

## **2 LANDASAN TEORI**

### **2.1 Definisi Bakso**

Bakso adalah jenis bola daging yang lazim ditemukan pada masakan Indonesia. Bakso umumnya dibuat dari campuran daging sapi giling dan tepung tapioka, akan tetapi ada juga bakso yang terbuat dari daging ayam, ikan, atau udang. Dalam penyajiannya, bakso umumnya disajikan panas-panas dengan kuah kaldu sapi bening, dicampur mi, bihun, taoge, tahu, terkadang telur, ditaburi bawang goreng dan seledri. Bakso sangat populer dan dapat ditemukan di seluruh Indonesia dari gerobak pedagang kaki lima hingga restoran besar.

Bakso memiliki akar dari seni kuliner Tionghoa Indonesia hal ini ditunjukkan dari istilah ‘bakso’ berasal dari kata Bak-So, dalam Bahasa Hokkien yang secara harfiah berarti ‘daging giling’. Karena kebanyakan penduduk Indonesia adalah muslim, maka bakso lebih umum terbuat dari daging halal seperti daging sapi, ikan atau ayam [1].

### **2.2 Definisi Kasir**

Kasir adalah seseorang yang pekerjaannya menerima uang pembayaran atas pembelian produk barang atau jasa dan melakukan pengembalian uang sisa pembayaran, sekaligus menyerahkan produk barang atau jasa kepada pelanggan (*customer*) di loket-loket kasir suatu toko, supermarket, minimarket, hotel, mall, restoran, rumah sakit ataupun department store. Selain itu, tugas kasir juga melakukan penghitungan jumlah total penjualan per hari, per minggu, ataupun per bulan serta mengenali barang yang paling laris dijual [2].

### 2.3 Definisi Aplikasi

Aplikasi adalah program komputer yang didesain untuk menangani aktivitas. Sehingga aplikasi berbeda dengan sistem operasi yang menjalankan komputer, atau utility yang melakukan maintenance atau perawatan komputer. Ada juga aplikasi pemrograman yang berguna untuk membuat program untuk komputer.

Aplikasi bisa sangat bervariasi tergantung kepada kebutuhan, sebuah aplikasi bisa memanipulasi teks, angka, gambar atau kombinasi dari elemen-elemen di atas.

Beberapa paket aplikasi mensyaratkan kemampuan komputasi yang tidak terlalu besar, misalnya yang hanya untuk melakukan suatu task tertentu, seperti editor teks, dan lainnya lebih terintegrasi, seperti software spreadsheet, macro word prosesor, dan software simulasi sains dan lain sebagainya.

Ada paket aplikasi yang terdiri dari beberapa aplikasi yang dibundel bersamaan, biasanya memiliki fungsi, fitur dan antarmuka yang berkaitan dan hampir sama, serta bisa berinteraksi satu dengan lainnya, misalnya MS Office, LibreOffice dan iWork yang memaketkan word processor, spreadsheet, software RDBMS menjadi satu [3].

### 2.4 Definisi Basis Data

Database dalam pengertian Microsoft Visual Foxpro adalah keterangan mengenai kumpulan sejumlah tabel, prosedur tersimpan (*stored procedure*) dan hubungan relasi antar tabel yang saling berhubungan dalam membentuk suatu program aplikasi. Jadi file database dalam Microsoft Visual Foxpro hanya menampung nama file, hubungan relasi dan keterangan dari file-file tabel lainnya [4].

Database merupakan tempat untuk menampung data susunan yang teratur sehingga kita dapat memperoleh informasi data kembali dengan mudah dan cepat. Setiap kolom dalam database merupakan sebuah field yang harus berisi data dengan jenis yang sama, sedangkan setiap baris dalam database merupakan record data

berisi kumpulan data yang terdiri atas beberapa field. Nama field (judul kolom) harus dalam satu baris serta mempunyai nama unik, artinya nama satu field harus berbeda dengan field lainnya [5].

## **2.5 Microsoft Visual FoxPro**

Microsoft Visual Foxpro 9 Prof. merupakan perangkat lunak PME (Properties-Method-Event) yang sangat cepat dan mudah dalam membuat program aplikasi memakai database. Bahasa pemrograman Foxpro merupakan bahasa pemrograman yang sangat mudah dipelajari dan dipakai untuk pembuatan aplikasi. Kekurangan dari Foxpro, database yang digunakan adalah database dbc sehingga mudah untuk di Hack. Cara menutupi kekurangannya, kita dapat menggunakan database MySQL. Hingga kini versi Microsoft Visual Foxpro sudah mencapai Versi 9 dan SP 2. Dari awal perkembangan bahasa xBase dari dBASE III Plus dengan FoxBASE, persaingan bahasa xBASE semakin ketat. FoxBASE berkembang menjadi FoxPro, dBASE III Plus menjadi dBASE IV. FoxPro dilanjutkan menjadi FoxPro Plus. Dengan berkembangnya perangkat lunak pemrograman Visual, FoxPro Plus menjadi Visual Foxpro.

Visual Foxpro 5.0 direlease menjadi dua versi, yaitu Versi berdiri sendiri dengan nama Microsoft Visual Foxpro 5.0 dan Microsoft Visual Foxpro 5.0A yang include dengan Visual Studio 97. Visual Studio 6 direlease, di dalamnya terdapat Microsoft Visual Foxpro 6 (dan jangan lupa di dalam paket ini juga terdapat pemrograman yang sangat populer Microsoft Visual Basic 6.0), disusul Microsoft Visual Foxpro 7 yang berdiri sendiri (tidak terdapat dalam Visual Studio), kemudian Microsoft Visual Foxpro 8, dan Microsoft Visual Foxpro 9 [6].

## **2.6 Flow of Document**

Bagan alir dokumen mengilustrasikan arus dokumen dan informasi di antara bidang tanggung jawab dalam suatu organisasi. Bagan alir dokumen melacak dokumen dari awal dibuatnya hingga dokumen tersebut tidak dipergunakan lagi. Bagan alir tersebut memperlihatkan tempat asal dokumen, distribusinya, tujuan






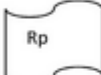







penggunaannya, tempat akhirnya dan segala sesuatu yang terjadi selama dokumen tersebut mengalir melewati sistemnya.

Bagan alir dokumen berguna untuk menganalisis kecukupan prosedur pengendalian di dalam suatu sistem seperti pemeriksaan internal dan pemisah fungsi. Bagan alir dokumen dapat membuka kelemahan dan ketakefisienan di dalam sistem seperti arus komunikasi yang tidak memadai, kerumitan arus data yang seharusnya tidak perlu terjadi atau atas prosedur yang menyebabkan penundaan yang tidak berguna [7].

### 2.6.1 Simbol-Simbol Diagram Alir Dokumen

Simbol-simbol dalam Diagram Alir Dokumen adalah sebagai berikut:

Tabel 2.1 Simbol-Simbol Diagram Alir Dokumen










	Dokumen/ Formulir		Dokumen/ Formulir n Rangkap
	Tanda Dimulainya Prosedur		Tanda Dimulainya Prosedur
	Daftar/Rekap		Uang
	Arsip diurut berdasarkan n : nomor t : tanggal a : abjad		Jurnal
	Konektor / Penghubung Antarhalaman		Penjelasan Proses
	Konektor / Penghubung Satu Halaman		Barang
	Penghubung dengan Arah dari Kiri ke Kanan		

## 2.7 Flowchart

Dalam membuat algoritma, diperlukan suatu mekanisme atau alat bantu untuk menuangkan hasil pemikiran mengenai langkah-langkah penyelesaian masalah yang sistematis dan terurut. Pada dasarnya untuk bisa menyusun solusi diperlukan kemampuan *problem-solving* yang baik. Oleh karena itu, sebagai sarana untuk melatih kemampuan tersebut terdapat sebuah *tool* (alat) yang dapat digunakan, yakni *flowchart*.

Secara formal, *flowchart* didefinisikan sebagai skema penggambaran dari algoritma atau proses. Tabel berikut menampilkan simbol-simbol yang digunakan dalam menyusun *flowchart*.

Tabel 2.2 Simbol-Simbol Dalam *Flowchart*

	<b>Terminator</b> Sebagai simbol 'START' atau 'END' untuk memulai atau mengakhiri flowchart.
	<b>Input/Output</b> Digunakan untuk menuliskan proses menerima data atau mengeluarkan data
	<b>Proses</b> Digunakan untuk menuliskan proses yang diperlukan, misalnya operasi aritmatika
	<b>Conditional / Decision</b> Digunakan untuk menyatakan proses yang membutuhkan keputusan
	<b>Preparation</b> Digunakan untuk memberikan nilai awal
	<b>Arrow</b> Sebagai penunjuk arah dan alur proses
	<b>Connector (On-page)</b> Digunakan untuk menyatukan beberapa arrow
	<b>Connector (Off-page)</b> Digunakan untuk menghubungkan flowchart yang harus digambarkan pada halaman yang berbeda. Biasanya pada simbol ini diberi nomor sebagai penanda, misalnya angka 1.
	<b>Display</b> Digunakan untuk menampilkan data ke <i>monitor</i>

Dengan menggunakan flowchart, tahapan-tahapan penting dalam algoritma dapat ditunjukkan dengan diagram di atas. Aliran proses ditunjukkan dengan arah panah atau disebut dengan ‘flowlines’.

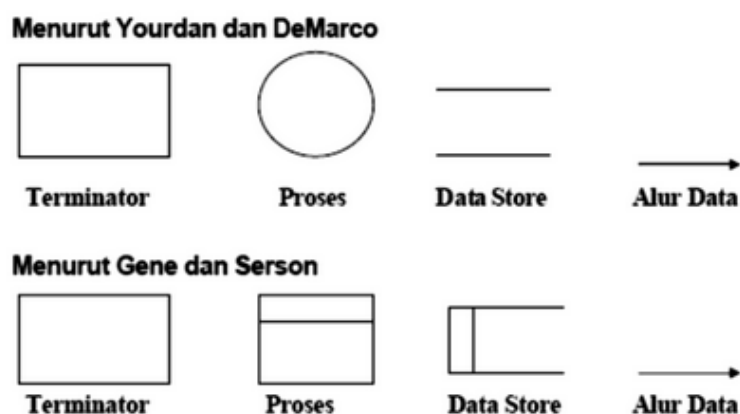
Keuntungan menggunakan *flowchart* adalah penggunaan diagram untuk menggambarkan tahapan proses sehingga lebih mudah dilihat dan dipahami. Namun demikian, flowchart juga memiliki kelemahan, yakni jika digunakan untuk menggambarkan proses atau algoritma untuk skala kasus yang besar, maka akan dibutuhkan banyak kertas [8].

## 2.8 Data Flow Diagram

Data Flow Diagram (DFD) “ *adalah alat pembuatan model yang memungkinkan profesional sistem untuk menggambarkan sistem sebagai suatu jaringan proses fungsional yang dihubungkan satu sama lain dengan alur data, baik secara manual maupun komputerisasi* “. DFD ini sering disebut juga dengan nama *Bubble chart, Bubble diagram, model proses, diagram alur kerja, atau model fungsi*.

DFD ini merupakan alat perancangan sistem yang berorientasi pada alur data dengan *konsep dekomposisi* dapat digunakan untuk *penggambaran analisa maupun rancangan sistem* yang mudah dikomunikasikan oleh profesional sistem kepada pemakai maupun pembuat program.

### 2.8.1 Komponen Data Flow Diagram (DFD).



Gambar 2.1 Simbol Data Flow Diagram

### 2.8.1.1 Komponen Terminator / Entitas Luar

*Terminator* mewakili entitas eksternal yang berkomunikasi dengan sistem yang sedang dikembangkan. Biasanya terminator dikenal dengan nama entitas luar (*external entity*).

Terdapat dua jenis terminator dalam *data flow diagram*:

1. Terminator Sumber (*source*): merupakan terminator yang menjadi sumber.
2. Terminator Tujuan (*sink*): merupakan *terminator* yang menjadi tujuan data / informasi sistem.

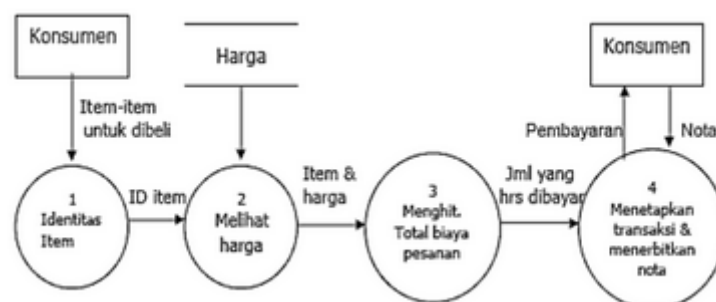
### 2.8.1.2 Komponen Data Flow / Alur Data

Konsep Sumber atau Tujuan Alur Data. Semua alur data harus *minimal mengandung satu proses*. Maksud kalimat ini adalah :

- a) Suatu alur data dihasilkan dari suatu *proses* dan menuju ke *suatu data store* dan/atau *terminator*
- b) Suatu alur data dihasilkan dari suatu *data store* dan/atau *terminator* dan menuju ke suatu proses.
- c) Suatu alur data dihasilkan dari suatu *proses* dan menuju ke suatu *Proses*.

### 2.8.1.3 Bentuk Data Flow Diagram

Terdapat dua bentuk DFD, yaitu **Diagram Alur Data Fisik**, dan **Diagram Alur Data Logika**. Diagram alur data fisik lebih menekankan pada bagaimana proses dari sistem diterapkan, sedangkan diagram alur data logika lebih menekankan proses-proses apa yang terdapat di sistem.



Gambar 2.2 Contoh Diagram Alur Data Fisik

#### 2.8.1.4 Syarat Pembuatan Data Flow diagram

Syarat-syarat pembuatan DFD ini adalah:

1. Pemberian nama untuk tiap komponen DFD
2. Pemberian nomor pada komponen proses
3. Penggambaran DFD sesering mungkin agar enak dilihat
4. Hindari penggambaran DFD yang rumit
5. Memastikan DFD yang dibentuk itu konsisten secara logika

#### 2.8.1.5 Teknik Penggambaran Data Flow Diagram

Secara garis besar langkah untuk membuat DFD adalah:

1. Identifikasi terlebih dahulu semua entitas luar yang terlibat di sistem.
2. Identifikasi semua input dan output yang terlibat dengan entitas luar.
3. Buat *Diagram Konteks (diagram context)*, diagram ini adalah diagram level tertinggi dari DFD yang menggambarkan hubungan sistem dengan lingkungan luarnya.

Hal terpenting yang perlu diperhatikan dalam penggambaran levelisasi DFD, yaitu dalam diagram konteks, ada beberapa hal yang perlu diperhatikan seperti hubungan sistem dengan dunia luar yang mempengaruhinya, penggambaran sistem dalam satu proses, dan penggambaran data store (optional) yang dikenal dengan *data store eksternal* atau *data store master* [9].

### 2.9 Entity Relationship Diagram

ERD menjadi salah satu pemodelan data konseptual yang paling sering digunakan dalam proses pengembangan basis data bertipe relasional. Penggunaannya yang sangat luas diakibatkan beberapa faktor, yaitu kemudahan, penggunaan secara luas *Computer Aided Software Engineering* (CASE), dukungan



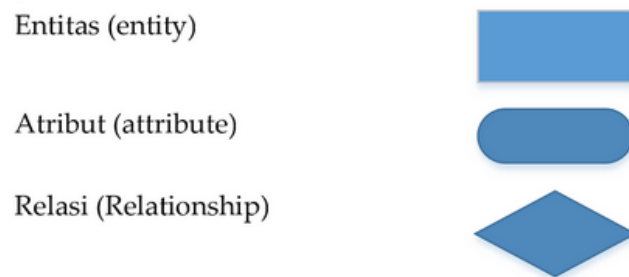
konsep matematika (kalkulus relasional) yang tangguh, hubungan entitas antar entitas merupakan konsep pemodelan alamiah yang sesuai dengan keadaan dunia nyata.

Model E-R sering digunakan sebagai sarana komunikasi antara perancang basis data dan pengguna sistem selama tahap analisis dari proses pengembangan basis data dalam kerangka pengembangan sistem informasi secara utuh. Model E-R digunakan untuk mengkonstruksi model data konseptual, yang mencerminkan struktur data dan batasan dari basis data, yang mandiri dari perangkat lunak pengelola basis data (DBMS) dan berhubungan erat dengan model data yang langsung bisa digunakan untuk mengimplementasikan basis data secara logika maupun secara fisik dengan DBMS yang dipilih pada tahap implementasi.

Model data E-R pertama kali diperkenalkan oleh Chen (1976) pada artikelnya yang mendiskusikan konstruksi utama dari model E-R, hubungan antar entitas (relationships) serta atribut-atribut yang bersesuaian dengan tiap entitas. Model yang diperkenalkan oleh Chen kemudian diperluas dan dikembangkan oleh Teorrey, Yang, Fry (1986), serta Story (1991) saat ini model E-R masih berkembang, namun tidak ada notasi baku untuk pemodelan E-R.

ERD adalah suatu diagram untuk menggambarkan desain konseptual dari model konseptual suatu basis data relasional. ERD juga merupakan gambaran yang merelasikan antara objek yang satu dengan objek yang lain dari objek di dunia nyata yang sering dikenal dengan hubungan antar entitas. Sebagai contoh jika membuat

ERD dari Sistem Perpustakaan maka bahan sebagai objek ERD bisa berupa anggota, buku, peminjaman, pengembalian dan sebagainya. ERD terdiri dari 3 Komponen Utama, yaitu:



Gambar 2.3 Komponen *Entity Relationship Diagram*

### 2.9.1 Entitas (*Entity*)

Entitas adalah suatu objek di dunia nyata yang dapat dibedakan dengan objek lainnya. Objek tersebut dapat berupa orang, benda ataupun hal lainnya. Entitas digambarkan dalam bentuk persegi panjang seperti pada gambar 2.4. Dilihat dari jenisnya entitas terbagi atas dua yaitu:

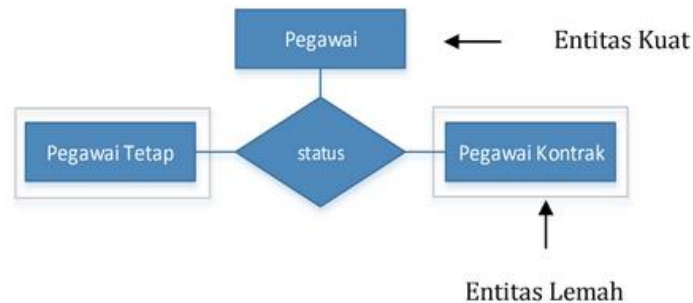
a. Entitas Kuat (*strong Entity*)

Entitas kuat adalah entitas yang dapat berdiri sendiri tidak bergantung pada entitas lainnya, entitas kuat memiliki atribut key dan entitas kuat digambarkan sebagai kotak persegi panjang bergaris tunggal. Contoh entitas kuat adalah entitas pegawai.

b. Entitas Lemah (*Weak Entity*)

Entitas lemah adalah entitas yang tidak dapat berdiri sendiri. Entitas lemah merupakan hasil pembentukan entitas kuat, entitas lemah tidak memiliki atribut key dan entitas lemah digambarkan sebagai kotak persegi panjang bergaris ganda. Jika entitas kuat yang membentuk entitas lemah dihapus maka secara otomatis entitas lemah akan terhapus.

Contoh entitas lemah adalah entitas pegawai kontrak, pegawai tetap.



Gambar 2.4 Jenis Entitas

### 2.9.2 Atribut (*Attribute*)

Atribut merupakan semua informasi yang berkaitan dengan entitas. Atribut sering dikenal dengan *property* dari suatu entitas atau objek. Atribut digambarkan dalam bentuk lingkaran elips. Macam-macam atribut:

#### a. Atribut Sederhana (*Simple Attribute*)

Atribut sederhana adalah atribut yang nilainya tidak dapat dibagi lagi menjadi banyak atribut yang lebih kecil. Contoh atribut sederhana harga seperti gambar 2.5.

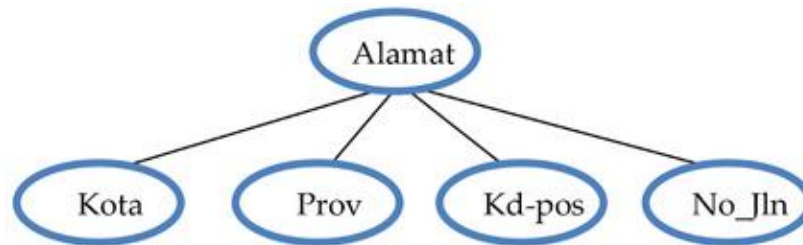


Gambar 2.5 Atribut Sederhana

#### b. Atribut Komposit (*Composite Attribute*)

Atribut komposit adalah atribut gabungan yang nilainya dapat dipecah menjadi bagian yang lebih kecil. Atau sering disebut atribut yang terdiri

dari beberapa atribut kecil di dalamnya. Contoh atribut komposit adalah alamat seperti gambar 2.6.



Gambar 2.6 Atribut Komposit

c. Atribut Bernilai Tunggal (*single Values Attribute*)

Atribut bernilai tunggal adalah jenis atribut yang nilainya hanya satu dari suatu entitas. Contoh atribut bernilai tunggal adalah tanggal\_lahir dari entitas mahasiswa. Telah bisa dipastikan bahwa setiap mahasiswa mempunyai satu tanggal\_lahir seperti gambar 2.7.



Gambar 2.7 Atribut Bernilai Tunggal.

d. Atribut Bernilai Banyak (*multivalues Attribute*)

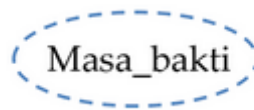
Atribut bernilai banyak adalah jenis atribut yang nilainya lebih dari satu dalam suatu entitas tertentu. Contoh atribut bernilai banyak adalah hobbi dimungkinkan bahwa mahasiswa memiliki lebih dari satu hobbi seperti gambar 2.8.



Gambar 2.8 Atribut Bernilai Banyak

e. Atribut Turunan (*Derived Attribute*)

Atribut turunan adalah jenis atribut yang nilainya diperoleh dari atribut yang lain. Contoh atribut turunan adalah masa\_bakti dari entitas pegawai. Atribut masa\_bakti akan muncul nilainya ketika atribut tanggal\_masuk\_kerja sudah ada nilainya. Pada dasarnya atribut masa bakti tidak akan dijadikan suatu kolom. Atribut masa\_bakti akan muncul dengan bantuan *query*.



Gambar 2.9 Atribut Turunan

a. Atribut Identitas (*Key Attribute*)

Atribut identitas adalah atribut yang dijadikan sebagai kunci pada suatu table. Sifat atribut identitas ini unik, tidak ada yang menyamai, atribut identitas terdiri dari beberapa jenis yaitu:

1. Super Key

Super Key adalah satu atribut atau kumpulan atribut yang secara unik mengidentifikasi sebuah baris di dalam relasi atau himpunan dari satu atau lebih entitas yang dapat digunakan untuk mengidentifikasi secara unik sebuah entitas dalam set entitas.

2. Candidate Key

Candidate Key adalah atribut yang menjadi determinan yang dapat dijadikan identitas baru pada sebuah relasi. Biasanya super key minimum.

3. Primary Key

Primary Key adalah kandidat key yang dipilih untuk mengidentifikasi baris data secara unik dalam relasi.

#### 4. Alternative Key

Alternative Key adalah candidate key yang tidak terpilih sebagai primary key atau atribut untuk menggantikan kunci utama.

#### 5. Foreign Key

Foreign Key adalah atribut dengan domain yang sama yang menjadi kunci utama sebuah relasi, tetapi pada relasi lain atribut tersebut sebagai atribut biasa.

#### 6. Composite Key

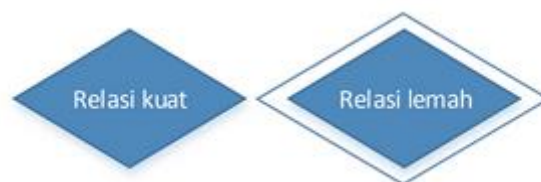
Composite Key adalah kunci yang terdiri dari dua atribut atau lebih. Atribut-atribut tersebut jika berdiri sendiri tidak menjadi identitas baris, tetapi bila dirangkakan menjadi satu kesatuan akan dapat mengidentifikasi secara unik.



Gambar 2.10 Atribut Identitas

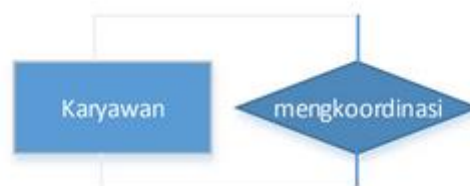
### 2.9.3 Tipe Relasi

Gambar belah ketupat merupakan perlambangan relasi antar entitas atau sering disebut kerelasian. Ada 2 macam penggambaran relasi yaitu relasi kuat dan relasi lemah. Relasi kuat adalah untuk menghubungkan antar entitas kuat sedangkan relasi lemah untuk menghubungkan antar entitas kuat dengan entitas lemah. Penggambaran relasi dapat dilihat pada gambar 2.11.



Gambar 2.11 Tipe Relasi

Ada tiga macam relasi menurut derajatnya, yaitu [1] Unary adalah relasi yang menghubungkan entitas yang sejenis, [2] Binary adalah relasi yang menghubungkan entitas yang tidak sejenis, [3] Ternary adalah relasi yang menghubungkan lebih dari dua entitas yang tidak sejenis. Contoh relasi berdasarkan derajat relasi seperti pada gambar 2.12, 2.13, 2.14 sebagai berikut:



Gambar 2.12 Derajat Relasi Unary

Derajat hubungan unary adalah entitas dosen hanya bekerjasama dengan entitas dosen yang entitas-nya sejenis, begitu juga dengan entitas karyawan (pimpinan) mengkoordinasi entitas karyawan (pekerja) yang entitas-nya sejenis.



Gambar 2.13 Derajat Relasi Binary

Derajat relasi binary pada gambar 2.14 adalah entitas kepala program studi berelasi dengan entitas program studi.





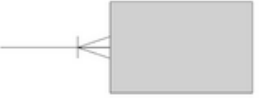


Gambar 2.14 Derajat Relasi Ternary

Derajat relasi ternary pada gambar 2.14 adalah entitas mahasiswa berelasi dengan dua entitas yang berbeda yaitu matakuliah nilai [10].

#### 2.9.4 Kardinalitas

*Cardinality* adalah jumlah minimum dan maksimum sebuah kejadian dalam entity yang dapat dihubungkan dengan kejadian dalam *entity* lainnya .

Tabel 2.3 Notasi *Cardinality*

Cardinality	Minimum	Maksimum	Notasi
1	1	1	
0 atau 1	0	1	
1 atau >1 (many)	1	>1 (many)	
0, 1 atau >1 (many)	0	>1 (many)	
>1 (many)	>1 (many)	>1 (many)	



### 2.9.5 Metodologi Membangun ERD

Setelah mengerti dan memahami konsep dan notasi yang digunakan untuk membangun ERD, berikut ini anda akan mempelajari metodologi dalam membangun ERD. Ada 10 (sepuluh) tahapan yang harus anda lakukan dalam membangun ERD, berikut ini tahapan tersebut.

1. Identifikasi *entity*, yaitu identifikasi peran, kejadian/kegiatan, lokasi, hal abstrak/konsep yang datanya perlu disimpan dalam sistem dan akan digunakan oleh *end-user*.
2. Tentukan *relationship*, tentukan hubungan antara 1 (satu) atau beberapa *entity* dengan menggunakan *relationship matrix*.
3. Gambarkan ERD secara kasar, gambarkan *entity* dengan *relationship* yang sudah diidentifikasi.
4. Tentukan *cardinality*, yaitu tentukan *cardinality* untuk setiap *relationship* pada ERD kasar yang sudah kita gambarkan pada langkah 3 (tiga).
5. Tentukan *primary key*, yaitu menentukan dan mengidentifikasi *attribute* yang menjadi *primary key* suatu *entity*
6. Gambarkan ERD berdasarkan *primary key*, yaitu sertakan *primary key* pada setiap *entity*.
7. Identifikasi *attribute* lainnya, yaitu mengidentifikasi *attribute* selain *primary key* (*detail data*) pada suatu *entity*.
8. Petakan *attribute*, yaitu meletakkan *attribute* pada suatu *entity* yang tepat dan mencari *attribute* lain dalam suatu *relationship*.
9. Gambarkan ERD lengkap dengan *attribute*, yaitu sesuaikan ERD hasil dari langkah 6 (enam) dengan *entity* atau *relationship* pada langkah 8 (delapan).
10. Periksa hasil, yaitu apakah ERD yang dihasilkan sudah secara tepat mencerminkan data yang dibutuhkan oleh sistem? [11]

## 2.10 Normalisasi

Normalisasi adalah suatu teknik dengan pendekatan *bottom-up* yang digunakan untuk membantu mengidentifikasi hubungan, dimulai dari menguji hubungan, yaitu *functional dependencies* antara atribut. Pengertian lainnya adalah suatu teknik yang menghasilkan sekumpulan hubungan dengan sifat-sifat yang diinginkan dan memenuhi kebutuhan pada perusahaan [12].

Proses normalisasi pertama kali ditemukan oleh EF. Codd (1972). Normalisasi biasa dilakukan sebagai serangkaian tes pada relation untuk menentukan apakah itu sesuai atau melanggar kebutuhan pada normal form.

Codd mendefinisikan bentuk normal pertama, kedua, dan ketiga di makalah (Codd, 1970). Bentuk normal ketiga kemudian diperbaiki sehingga mempunyai bentuk normal yang lebih kuat, yaitu BCNF (Codd, 1974). Fagin memperkenalkan bentuk normal keempat (Fagin, 1977). Fagin juga memperkenalkan bentuk normal kelima (Fagin, 1979) [13].

### 2.10.1 Tujuan Normalisasi

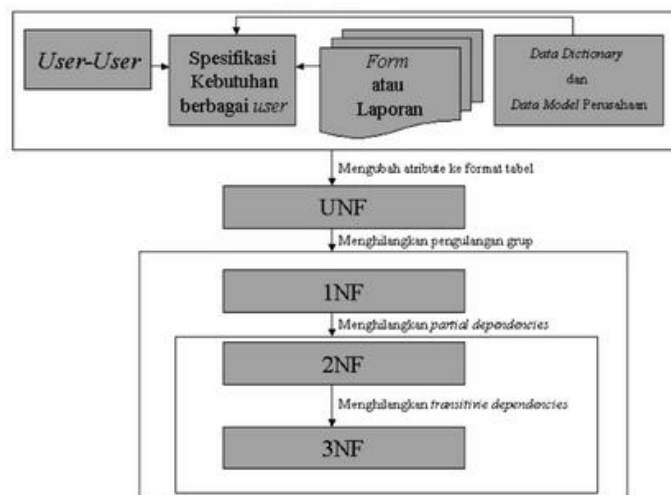
Tujuan utama normalisasi adalah mengidentifikasi kesesuaian hubungan yang mendukung data untuk memenuhi kebutuhan perusahaan. Adapun karakteristik hubungan tersebut mencakup:

- Minimal jumlah atribut yang diperlukan untuk mendukung kebutuhan perusahaan.
- Atribut dengan hubungan logika yang menjelaskan *functional dependencies*.
- Minimal duplikasi untuk tiap atribut.

### 2.10.2 Jenis Normalisasi

Terdapat empat bentuk normal yang biasa digunakan, yaitu:

- *First Normal Form* (1NF) atau Normalisasi Tingkat 1
- *Second Normal Form* (2NF) atau Normalisasi Tingkat 2
- *Third Normal Form* (3NF) atau Normalisasi Tingkat 3
- *Boyce-Codd Normal Form* (BCNF)
- *Four Normal Form* (4NF)
- *Five Normal Form* (5NF)



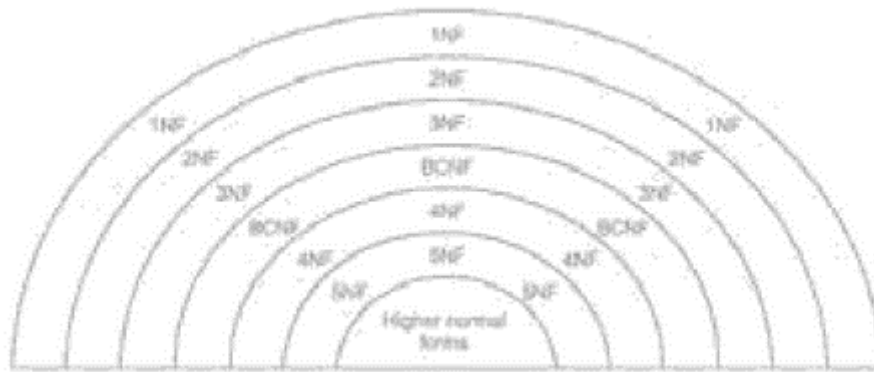
Gambar 2.15 Diagram Proses Normalisasi

### 2.10.3 Proses Normalisasi

Beberapa hal yang perlu diperhatikan dalam proses normalisasi adalah:

- Suatu teknik formal untuk menganalisis relasi berdasarkan *primary key* dan *functional dependencies* antar-atribut.
- Dieksekusi dalam beberapa langkah. Setiap langkah mengacu ke bentuk normal tertentu, sesuai dengan sifat yang dimilikinya.

- Setelah normalisasi diproses, relasi menjadi secara bertahap lebih terbatas atau kuat mengenai bentuk formatnya dan juga mengurangi tindakan *update* yang anomali.



Gambar 2.16 Hubungan antara *Normal Forms*

#### 2.10.3.1 Unnormalized Form (UNF)

Merupakan suatu tabel yang berisikan satu atau lebih grup yang berulang. Membuat tabel yang *unnormalized* adalah dengan memindahkan data dari sumber informasi. Contohnya, nota penjualan yang disimpan ke dalam format tabel dengan baris dan kolom.

#### 2.10.3.2 First Normal Form (1NF)

Merupakan suatu relasi di mana setiap baris dan kolom berisikan hanya satu nilai.

##### Proses UNF ke 1NF

- Tentukan satu atau kumpulan atribut sebagai kunci untuk tabel *unnormalized*.
- Identifikasikan grup yang berulang dalam tabel *unnormalized* yang berulang untuk kunci atribut.
- Hapus grup yang berulang dengan cara:
  - Masukkan data yang semestinya ke dalam kolom yang kosong pada baris yang berisikan data berulang (*flattening the table*).

- Menggantikan data yang ada dengan menulis ulang dari kunci atribut yang sesungguhnya ke dalam relasi terpisah.

### 2.10.3.3 Second Normal Form (2NF)

- Berdasarkan pada konsep *full functional dependency*, yaitu A dan B merupakan atribut dari sebuah relasi, B dikatakan *fully dependent* terhadap A jika B *functionally dependent* pada A, tetapi tidak pada proper subset dari A.
- 2NF merupakan sebuah relasi dalam 1NF dan setiap atribut *non-primary key* bersifat *fully functionally dependent* pada *primary key*.
- 1NF ke 2NF
  - Identifikasikan *primary key* untuk relasi 1NF.
  - Identifikasikan *functional dependencies* dalam relasi.
  - Jika terdapat *partial dependencies* terhadap *primary key*, maka hapus dengan menempatkan dalam relasi yang baru bersama dengan salinan determinannya.

### 2.10.3.4 Third Normal Form (3NF)

- Berdasarkan pada konsep *transitive dependency*, yaitu suatu kondisi di mana A, B, dan C merupakan atribut dari sebuah relasi, maka  $A \rightarrow B$  dan  $B \rightarrow C$ , maka *transitively dependent* pada A melalui B (jika A tidak *functionally dependent* pada B atau C).
- 3NF adalah sebuah relasi dalam 1NF dan 2NF dan di mana tidak terdapat atribut *nonprimary key* yang bersifat *transitively dependent* pada *primary key*.
- 2NF ke 3NF
  - Identifikasikan *primary key* dalam relasi 2NF.
  - Identifikasikan *functional dependencies* dalam relasi.
  - Jika terdapat *transitive dependencies* terhadap *primary key*, hapus dengan menempatkannya dalam relasi yang baru bersama dengan salinan determinannya.

### 2.10.3.5 Boyce-Codd Normal Form (BCNF)

- Berdasarkan pada *functional dependencies* yang dimasukkan ke dalam hitungan seluruh *candidate key* dalam suatu relasi, bagaimana pun BCNF juga memiliki batasan-batasan tambahan disamakan dengan definisi umum dari 3NF.
- Suatu relasi dikatakan BCNF, hanya jika setiap determinan merupakan *candidate key*.
- Perbedaan antara 3NF dan BCNF, yaitu untuk *functional dependency*  $A \rightarrow B$ , 3NF memungkinkan *dependency* ini dalam suatu relasi jika B adalah atribut *primary key* dan A bukan merupakan *candidate key*.
- Sedangkan BCNF menetapkan dengan jelas bahwa untuk *dependency* agar ditetapkan dalam relasi A, maka A harus merupakan *candidate key*.
- Setiap relasi dalam BCNF juga merupakan 3NF, tetapi relasi dalam 3NF belum tentu termasuk ke dalam BCNF.
- Dalam BCNF kesalahan jarang sekali terjadi. Kesalahan dapat terjadi pada relasi yang:
  - Terdiri dari 2 atau lebih *composite candidate key*.
  - *Candidate key overlap*, sedikitnya satu atribut [12].