

BAB 2

TINJAUAN PUSTAKA

2.1. Pengertian Basis Data

Teori - teori yang akan digunakan sebagai landasan penyusunan skripsi ini akan dijelaskan dalam sub-bab berikut.

2.1.1. Data

Menurut Indrajani (2015:69), data adalah fakta-fakta mentah kemudian dikelola sehingga menghasilkan informasi yang penting bagi sebuah perusahaan atau organisasi.

2.1.2. Basis Data dan Sistem Basis Data

Menurut Connolly dan Begg (2010:65), basis data adalah sebuah kumpulan data yang secara logis terkait dan dirancang untuk memenuhi suatu kebutuhan informasi dari sebuah organisasi.

Menurut Indrajani (2015:70), basis data adalah kumpulan data yang saling berhubungan secara logis dan didesain untuk mendapatkan data yang dibutuhkan oleh suatu organisasi.

Menurut Connolly dan Begg (2010:54), sistem basis data adalah kumpulan dari program aplikasi yang berinteraksi dengan basis data bersama dengan *Database Management System (DBMS)* dan basis data itu sendiri.

2.1.3. Database Management System (DBMS)

DBMS adalah sebuah sistem perangkat lunak yang mengizinkan pengguna untuk mendefinisikan, membuat, memelihara, dan mengontrol akses ke dalam basis data. (Connolly dan Begg, 2010 :66).

a. Fasilitas yang disediakan oleh *DBMS*

1. Mengizinkan pengguna untuk mendefinisikan basis data, dengan melalui *Data Definition Language (DDL)*. *DDL* mengizinkan pengguna untuk menentukan tipe, struktur, serta

kendala data yang nantinya akan disimpan ke dalam basis data.

2. Mengizinkan pengguna untuk melakukan menambah, mengubah, menghapus dan mengambil data dari basis data tersebut, dengan menggunakan *Data Manipulation Language (DML)*. *Standard* bahasa dari *DBMS* ialah *Structured Query Language (SQL)*.
3. Menyediakan akses kontrol ke dalam basis data, seperti :
 - a. Sistem keamanan, yang dapat mencegah pengguna yang tidak diberi kuasa untuk mengakses basis data.
 - b. Sistem integritas, yang dapat menjaga konsistensi dari data yang tersimpan.
 - c. Sistem kontrol konkurensi, yang mengizinkan berbagi akses dengan basis data.
 - d. Sistem kontrol pemulihan, jika terjadi kegagalan perangkat keras atau perangkat lunak maka sistem kontrol pemulihan ini dapat mengembalikan basis data ke keadaan yang konsisten dari yang sebelumnya.

b. Komponen Utama dalam *DBMS*

1. *Hardware*

Hardware yang digunakan dapat berupa *Personal Computer (PC)* yang akan disesuaikan dengan kebutuhan perusahaan dan *DBMS* yang akan digunakan.

2. *Software*

Komponen *software* terdiri dari *software DBMS* itu sendiri dan program aplikasi, bersamaan dengan sistem operasinya, serta termasuk *software* jaringan, apabila *DBMS* yang akan digunakan melalui sebuah jaringan.

3. Data

Data adalah komponen yang terpenting pada *DBMS*, karena data merupakan sebuah jembatan penghubung antara komponen mesin dengan manusia.

4. *Procedures*

Prosedur berisikan instruksi serta aturan yang digunakan untuk merancang dan menggunakan sebuah basis data.

5. *People*

Komponen terakhir adalah manusia yang dapat terlibat langsung dengan sistem tersebut.

c. Keuntungan *DBMS*:

1. Mengendalikan redundansi data

Menghilangkan redundansi dengan cara mengintegrasikan *file-file* tersebut agar salinan dari data yang sama tidak disimpan. Dikarenakan apabila data yang sama dengan data tersebut ditemukan lebih dari satu tabel di dalam basis data maka akan terjadi redundansi.

2. Meningkatkan Integritas data

Integritas basis data megacu pada validitas dan konsistensi data yang tersimpan. Integritas dinyatakan dalam *constraints*.

3. Meningkatkan keamanan

Keamanan sebuah basis data melindungi basis data dari pengguna yang tidak berwenang. dengan cara membuat *username* dan *password* untuk mengidentifikasi pengguna yang memiliki hak akses dalam menggunakan basis data. Akses yang diberikan pada pengguna yang telah memiliki hak akses dapat melakukan operasi seperti, *retrieval*, *insert*, *update*, *delete*.

4. Meningkatkan pelayanan *backup* dan *recovery*

menyediakan fasilitas untuk pemulihan data bila terjadi kegagalan pada *software* atau *hardware* dapat di pulihkan sehingga dapat meminimalkan jumlah pemrosesan yang hilang.

5. Berbagi Data

DBMS memungkinkan pengguna untuk menggunakan data yang sama secara bersamaan tentunya dengan pengguna yang berwenang.

d. Kerugian *DBMS*:

1. Kompleksitas

DBMS merupakan sebuah perangkat lunak yang sangat kompleks. *Database Designers*, *Database Developer*, *Database Administrator* dan *End-user* harus mengerti fungsionalitasnya.

2. Ukuran

Karena besarnya kompleksitas pada *DBMS* membuat *DBMS* membutuhkan sebuah kapasitas penyimpanan yang besar agar dapat menjalankan aplikasinya.

3. Biaya dalam *DBMS*

Biaya dalam *DBMS* sangat bervariasi, dan itu tergantung dari lingkungan dan fungsionalitas yang diinginkan.

4. Biaya tambahan dari *hardware*

Membutuhkan biaya tambahan untuk kapasitas penyimpanan dan agar dapat mencapai kinerja yang diinginkan sehingga membutuhkan mesin yang lebih besar.

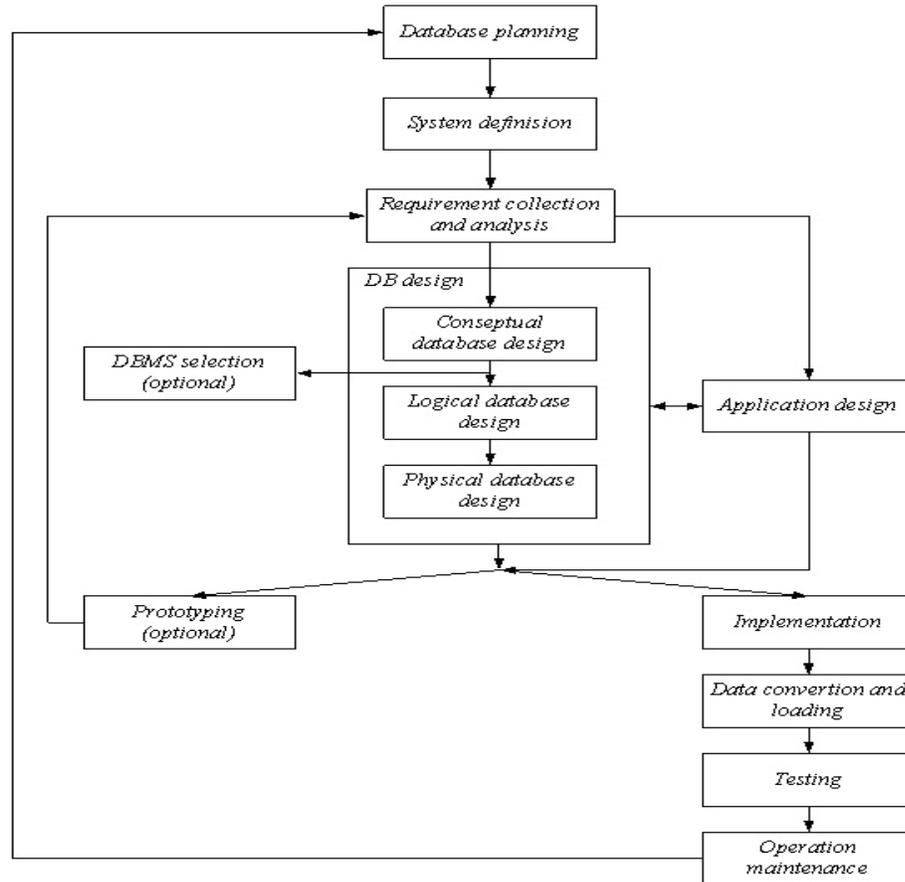
5. Biaya Konversi

Biaya konversi relatif lebih kecil. Biaya konversi meliputi pelatihan karyawan untuk menggunakan sistem yang baru dan memungkinkan karyawan dengan keahlian yang khusus

untuk membantu konversi dan menjalankan sistem.

2.2. Database System Development Lifecycle

Menurut Connolly dan Begg (2010:313), *database system development lifecycle* merupakan komponen yang penting di dalam sistem basis data dan aplikasi dari *database lifecycle* berkaitan dengan sistem informasi yang ada. Untuk sistem basis data yang kecil dengan pengguna yang jumlahnya sedikit membuat siklus hidupnya menjadi tidak harus sangat kompleks, tetapi jika sistem basis data yang didesain untuk ukuran skala menengah hingga besar dengan ribuan pengguna maka siklus hidupnya akan menjadi kompleks.



Gambar 2. 1 Tahapan Database System Development Lifecycle

Berikut penjelasan dari tahap-tahapan siklus dari *database* di atas :

1. ***Database Planning***

Aktifitas manajemen dalam merealisasikan tahapan awal dari siklus hidup sistem basis data secara efisien dan efektif. Ada tiga hal pokok yang harus diperhatikan dalam menentukan strategi sistem informasi :

- a. Identifikasi rencana serta tujuan perusahaan atau organisasi dengan menentukan sistem informasi yang diperlukan.
- b. Mengevaluasi sistem informasi yang ada untuk mengetahui kelebihan beserta kekurangannya.
- c. Penilaian tentang peluang IT yang nantinya mungkin dapat menghasilkan keuntungan yang kompetitif.

Terdapat dua metodologi yang digunakan untuk memecahkan masalah tersebut, yaitu mendefinisikan *mission statement* dan mendefinisikan *mission objectives*

2. ***Definition System***

Definisi sistem ini bertujuan untuk menggambarkan ruang lingkup dan batasan dari sistem basis data dari sudut pandang pengguna. Jadi dalam tahap ini membahas apa yang nantinya dapat diharapkan dari aplikasi basis data berdasarkan peran penggunanya.

3. ***Requirement Collection and Analysis***

Dalam tahap ini bertujuan melakukan pengumpulan dan analisis informasi tentang kebutuhan pengguna yang nantinya dapat dilayani oleh sistem basis data yang baru.

4. ***Database Design***

Database design adalah proses untuk membuat desain yang dapat mendukung operasional dan tujuan perusahaan.

Tujuan desain basis data, antara lain :

- a. Menggambarkan relasi data antara data yang dibutuhkan oleh aplikasi dan *user view*
- b. Menyediakan model data yang mendukung seluruh transaksi yang diperlukan
- c. Menspesifikasikan desain dengan struktur yang sesuai dengan kebutuhan sistem

Terdapat tiga fase yang akan dilalui dalam membuat desain basis data, yaitu:

1. *Conceptual Database Design*

Proses membangun suatu model data, informasi berskala perusahaan dan bebas dari pertimbangan fisik, di mana akan lebih menekankan konsep.

2. *Logical Database Design*

Proses membangun model dari informasi yang telah diperoleh berdasarkan model data khusus, tetapi bebas dari hal yang berkaitan dengan *DBMS* dan pertimbangan fisik lain.

3. *Physical Database Design*

Proses dari pembuatan yang menjelaskan implementasi basis data pada penyimpanan sekunder, yang menggambarkan struktur penyimpanan dan metode akses yang akan digunakan mencapai akses yang efektif.

5. ***DBMS Selection***

Memilih *DBMS* yang tepat sehingga dapat mendukung sistem basis data yang akan dibuat. Langkah-langkah yang digunakan dalam memilih *DBMS*:

- a. Definisikan waktu untuk melakukan studi referensi.
- b. Catat dua/tiga produk yang akan di evaluasi untuk digunakan.
- c. Evaluasi produk tersebut.
- d. Rekomendasikan produk yang dipilih dan buat laporan yang mendukung produk.

6. *Application Design*

Melakukan perancangan *user interface* dan program aplikasi yang digunakan dan memproses basis data tersebut. Terdapat dua aktivitas yang harus diperhatikan, yaitu:

- a. *Transaction design*
- b. *User interface design*

7. *Prototyping*

Prototyping adalah membangun sebuah model kerja dari aplikasi basis data tersebut (bersifat opsional). Tujuan utama dari tahapan ini adalah:

- a. Mengidentifikasi fitur sistem yang sedang berjalan.
- b. Memberikan perbaikan atau penambahan dari fitur baru.
- c. Mengklarifikasi kebutuhan *user*.
- d. Evaluasi kelayakan dan kemungkinan apa yang terjadi dari desain sistem.

8. *Implementation*

Merupakan realisasi secara fisik dari basis data dan rancangan aplikasinya. Implementasi basis data dapat dicapai dengan menggunakan:

- a. *DDL* untuk membuat skema basis data dan *database file* yang kosong.
- b. *DDL* untuk membuat *user view* yang diinginkan.
- c. *3GL* atau *4GL* untuk membuat program aplikasi.

9. *Conversion Data and Loading*

Proses pemindahan data lama ke dalam basis data yang baru serta mengkonversikan aplikasi yang sudah ada sehingga dapat digunakan ke basis data yang baru. Tahap ini diperlukan ketika sistem basis data baru akan menggantikan sistem yang lama.

10. *Testing*

Suatu proses dengan menjalankan sistem basis yang bertujuan untuk menemukan *error*.

11. *Operational Maintenance*

Proses memonitori serta memelihara instalisasi sistem basis data. Setelah sistem basis data telah beroperasi secara penuh, diperlukan untuk memonitori secara dekat agar memastikan performa berada pada *level* yang bisa diterima.

Pada sistem ini kegiatan yang termasuk di dalamnya adalah

- a. Mengawasi kinerja sistem.
- b. Pemeliharaan dan pembaruan aplikasi dari basis data tersebut.
- c. Penggabungan kebutuhan yang baru ke dalam aplikasi basis data.

2.3. *Entity Relationship Modeling*

Menurut Indrajani (2015:17) *Entity Relationship modeling* adalah sebuah pendekatan *top-bottom* dalam merancang sebuah basis data, dimulai dengan mengidentifikasi data yang penting dan digambarkan dalam suatu model.

Menurut Connolly dan Begg (2010:371), *Entity Relationship modeling* merupakan pemodelan yang berguna untuk digunakan agar mendapatkan pemahaman yang tepat terhadap data dan penggunaannya di dalam suatu perusahaan.

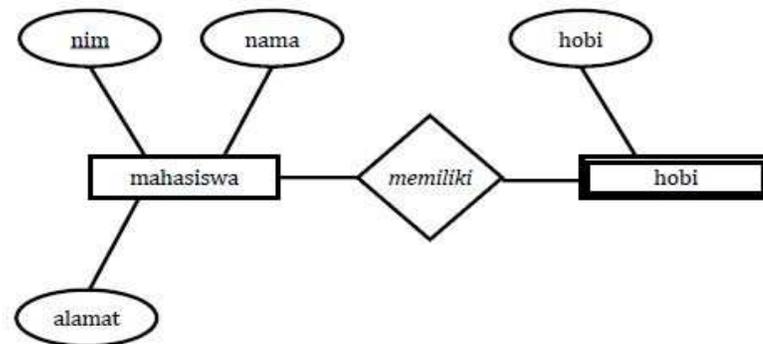
Aspek utama dari *Entity Relationship modeling* yaitu:

2.3.1. *Entity*

Entitas adalah sekumpulan objek dengan properti yang sama, yang diidentifikasi di dalam organisasi atau perusahaan dikarenakan keberadaannya yang bebas (*independen*).

Terdapat dua tipe entitas:

1. Entitas Lemah (*Weak Entity*) merupakan entitas yang keberadaanya bergantung pada keberadaan entitas lain. Contoh, entitas hobi bergantung pada entitas mahasiswa.



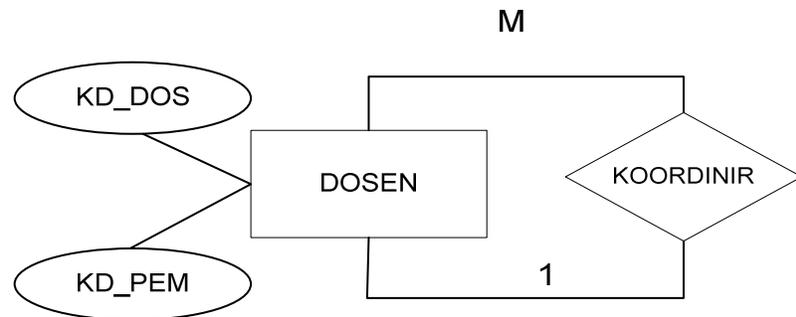
Gambar 2. 2 Entitas Lemah & Kuat

2. Entitas Kuat (*strong Entity*) merupakan entitas yang berdiri sendiri tanpa bergantung dengan entitas lain. Contoh entitas mahasiswa, hobi.

2.3.2. Relationship Type

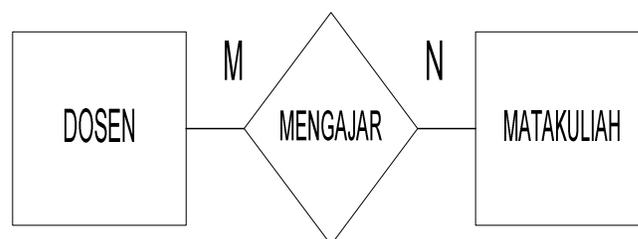
Relationship type adalah sebuah set asosiasi yang memiliki hubungan antar entitas. Derajat tipe hubungan adalah jumlah jenis entitas yang berpartisipasi dalam satu hubungan.

1. *Relationship* berderajat satu (Unary)



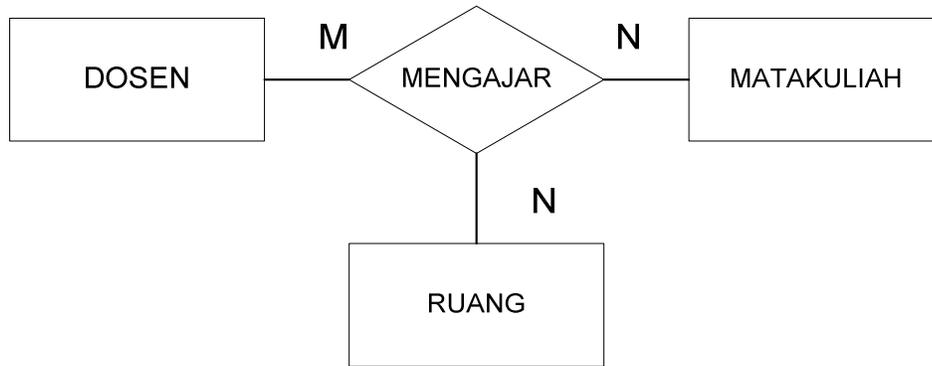
Gambar 2. 3 *Relationship* berderajat satu *one to many*

2. *Relationship* berderajat dua (Binary)



Gambar 2. 4 *Relationship* berderajat dua *Many to Many(M:N)*

3. *Relationship* berderajat tiga (Teranry)



Gambar 2. 5 *Relationship* berderajat tiga dua *Many to Many*(M:N)

2.3.3. *Attributes*

Atribut adalah sebuah kolom yang memiliki nama pada sebuah relasi.

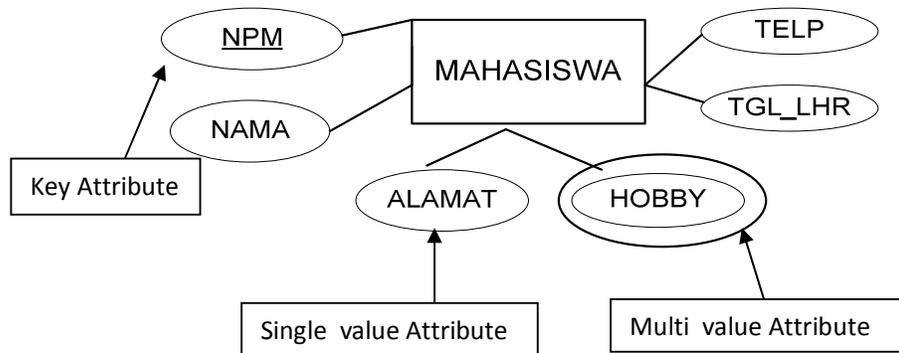
Berikut ada tipe dari atribut yaitu:

1. *Single – Valued Attribute*

Sebuah atribut yang mempunyai nilai tunggal untuk setiap kejadian dari sebuah entitas.

2. *Multi valued attribute*

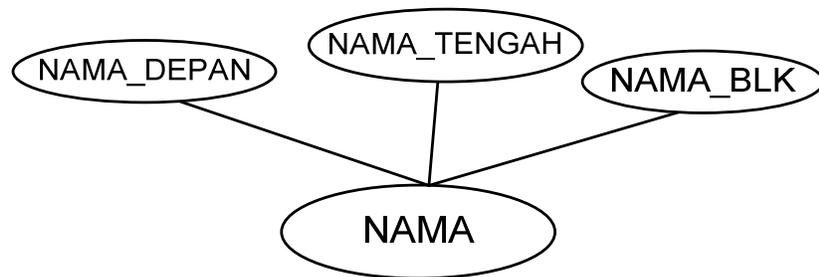
Nilai dari suatu *attribute* yang mempunyai lebih dari satu (*multivalue*) nilai dari *attribute* yang bersangkutan



Gambar 2. 6 Single & Multivalued attributes

3. *Composite Attribute*

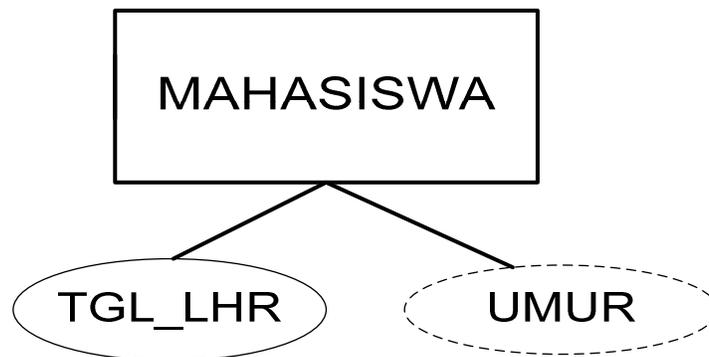
Atribut *composite* adalah suatu atribut yang terdiri dari beberapa atribut yang lebih kecil yang mempunyai arti tertentu yang masih bisah dipecah lagi atau mempunyai *sub attribute*.



Gambar 2. 7 Composite Attribute

4. Derivatif *Attribute*

Atribut yang tidak harus disimpan dalam *database* Ex. Total. atau atribut yang dihasilkan dari atribut lain atau dari suatu *Relationship*. Atribut ini dilambangkan dengan bentuk oval yang bergaris putus-putus



Gambar 2. 8 Derivatif attribute

2.3.4. Key

Key adalah sebuah *field* yang digunakan untuk mengidentifikasi satu atau lebih atribut secara unik untuk mengidentifikasi setiap *record*. Terdapat lima jenis *key* yang bisa digunakan, yaitu:

1. *Candidate Key*

Merupakan set atribut minimal yang secara unik mengidentifikasi setiap kejadian dari sebuah tipe entitas.

2. *Primary Key*

Merupakan *candidate key* yang dipilih untuk mengidentifikasi setiap kejadian dari suatu tipe entitas secara unik.

3. *Composite Key*

Merupakan sebuah *candidate key* yang terdiri dari dua atau lebih atribut.

4. *Foreign Key*

Merupakan sebuah atribut pada suatu relasi yang sama dengan *candidate key* dari relasi lainnya.

5. *Alternate Key*

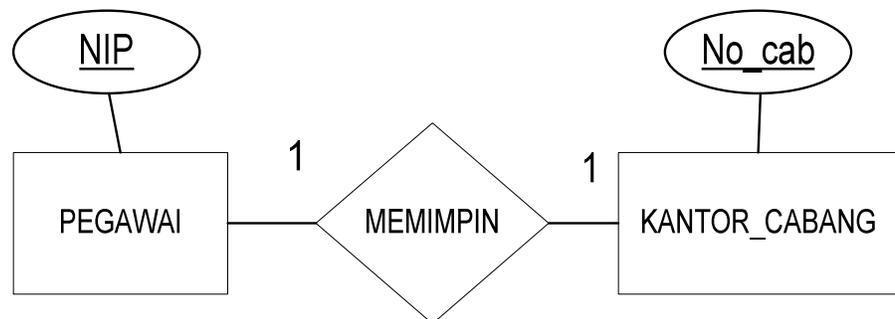
Merupakan kumpulan sebuah atribut dari *candidate key* yang tidak terpilih menjadi *primary key*.

2.3.5. Kardinalitas Rasio

Kardinalitas rasio adalah sejumlah kemungkinan entitas A berpartisipasi dengan entitas B dalam satu *Relationship*. Ada tiga jenis yakni:

1. ***One to One (1:1)***

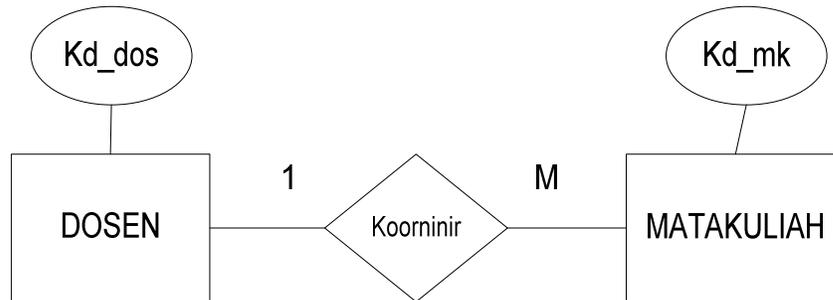
Setiap anggota entitas A hanya boleh berhubungan dengan satu anggota entitas B, begitu pula sebaliknya.



Gambar 2. 9 Relasi One To One

2. **One to many (1...*)**

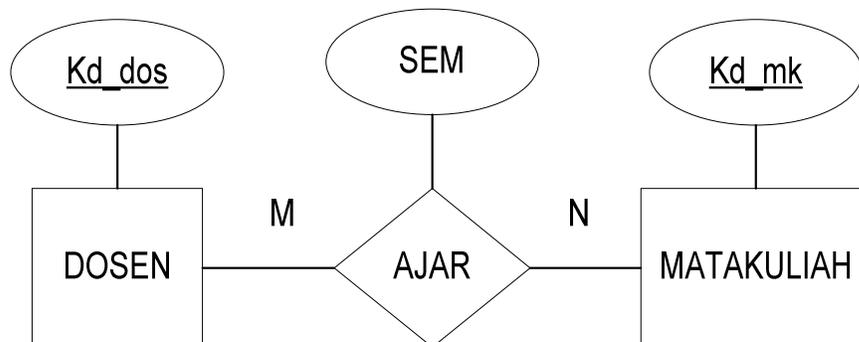
Setiap anggota entitas A dapat berhubungan dengan lebih dari satu anggota entitas B tetapi tidak sebaliknya.



Gambar 2. 10 Relasi 1 ... *

3. **Many to Many (*...*)**

Setiap entitas A dapat berhubungan dengan banyak entitas himpunan entitas B dan demikian pula sebaliknya



Gambar 2. 11 Relasi *...*

2.4. Normalisasi

Menurut Indrajani (2015:7), normalisasi adalah teknik dengan melakukan sebuah pendekatan *bottom-up* yang digunakan dalam membantu mengidentifikasi hubungan. Sedangkan menurut Connolly dan Begg (2010:416), normalisasi adalah sebuah teknik yang menghasilkan suatu kumpulan relasi dengan *property* yang diinginkan dengan memberikan suatu kebutuhan data pada perusahaan.

Tujuan normalisasi adalah

1. Menghilangkan kerangkapan data
2. Mengurangi kompleksitas
3. Mempermudah pemodifikasian data

Tahapan-tahapan dalam normalisasi sebagai berikut

1. *Unnormalized Form (UNF)*

Adalah sebuah tabel yang memuat satu atau lebih kelompok yang berulang.

2. *First Normal Form (1NF)*

Adalah sebuah relasi yang terdiri dari perpotongan dari setiap baris dan kolom berisi satu dan hanya satu buah nilai saja. Aturan dari 1NF yaitu:

- a. Tidak ada atribut *multi-value*, atribut komposit atau kombinasinya
- b. Mendefinisikan atribut kunci
- c. Setiap atribut dalam *table* tersebut harus bernilai *atomic* (tidak dapat dibagi-bagi lagi)

3. *Second Normal Form (2NF)*

Adalah sebuah relasi yang berada dalam bentuk 1NF di mana setiap atribut yang bukan *primary key* bergantung secara fungsional penuh kepada *primary key*. Aturan dari 2NF yaitu :

- a. Sudah memenuhi dalam bentuk normal kesatu (1NF)
- b. Semua atribut bukan kunci hanya boleh tergantung (*functional dependency*) pada atribut kunci

- c. Jika ada ketergantungan parsial maka atribut tersebut harus dipisah pada *table* yang lain
- d. Perlu ada *table* penghubung ataupun kehadiran *foreign key* bagi atribut-atribut yang telah dipisah tadi.

4. **Third Normal Form (3NF)**

Adalah relasi yang berada dalam bentuk 1NF dan 2NF di mana tidak ada lagi atribut yang bukan *primary key* yang bergantung secara transitif kepada *primary key*. Aturan dari 3NF yaitu:

- a. Sudah berada dalam bentuk normal kedua (2NF)
- b. Tidak ada ketergantungan transitif (di mana atribut bukan kunci tergantung pada atribut bukan kunci lainnya)

2.5. Metodologi Perancangan Basis data

Menurut Connolly dan Begg (2010:466) metodologi perancangan adalah pendekatan terstruktur yang menggunakan bantuan prosedur, teknik, peralatan, dan dokumentasi untuk mendukung dan memfasilitasi proses perancangan.

2.5.1 Perancangan basis data konseptual

Menurut Connolly dan Begg (2010:467) rancangan basis data konseptual adalah proses pembangunan model data yang digunakan pada suatu perusahaan, dan bebas dari seluruh pertimbangan fisikal. Sebuah rancangan basis data konseptual memiliki *Entity types*, *Relationship type*, *attributes* dan *attribute domains*, *primary keys* dan *alternate keys*, dan *integrity constraint*. Langkah-langkah dalam pembangunan rancangan basis data konseptual antara lain:

1. Mengidentifikasi *Entity type*

Mengidentifikasi spesifikasi kebutuhan pengguna dan mengambil kata-kata benda dari spesifikasi tersebut. Selain itu juga dapat diambil objek besar seperti orang, tempat, dan lainnya, kecuali kata-kata benda yang menjadi ukuran dari objek lainnya. Dan juga jangan ambil kata-kata benda yang

merupakan sinonim dari objek lainnya. Contoh hasil dari langkah ini antara lain Karyawan, Pelanggan, Servis, Barang dan lain-lain.

2. Mengidentifikasi *Relationship types*

Langkah selanjutnya yaitu mengidentifikasi *Relationship* di antara *Entity types* yang sudah ada. *ER* diagram digunakan sehingga dapat terlihat hubungan antar *Entity type* dengan jelas.

3. Mengidentifikasi dan mengasosiasikan *attributes* dengan *Entity* atau *Relationship types* tertentu. Cara termudah untuk melakukan identifikasi dan menghubungkan *attributes* dengan *Entity* dan *Relationship types* adalah dengan menanyakan pertanyaan “Informasi apa yang kita butuhkan untuk dipegang”.

4. Menentukan *domain* atribut

Domain adalah kumpulan nilai di mana satu atau lebih *attributes* menggambarkan nilainya.

5. Menentukan *candidate*, *primary*, dan *alternate key*

Dari masing-masing *domain* atribut yang telah dibuat di tahap sebelumnya, dapat dilanjutkan ke tahap selanjutnya dengan menentukan *candidate*, *primary*, dan *alternative key*. *Candidate key* merupakan *key* yang berpeluang untuk menjadi *primary key*. *Primary key* adalah *key* unik yang menjadi kunci utama sebuah entitas untuk relasi ke entitas lain dalam sebuah tabel. Sedangkan *alternative key* merupakan atribut-atribut pada *candidate key* yang tidak terpilih menjadi *primary key*.

6. Mempertimbangkan penggunaan *enhanced ER Modelling (optional)*. Tahap ini merupakan proses pertimbangan akan penggunaan *enhanced ER Modelling* dan bersifat opsional untuk dipakai atau tidak.

7. Mengecek model untuk pengulangan

Dalam tahap ini, ada tiga langkah yang dikerjakan antara lain:

- a. Mengecek ulang *one-to-one* (1:1) *Relationship*.
 Dalam melakukan perancangan, mungkin saja terjadi dua *Entity* yang sebenarnya merepresentasikan objek yang sama di dunia nyata. Hal ini dapat dihilangkan dengan menggabungkan kedua *Entity* tersebut dan memilih salah satu *primary key* menjadi *primary key* pada *Entity* hasil gabungan.
- b. Menghilangkan *Relationship* yang mengulang
Relationship yang mengulang bisa didapatkan dengan melihat apakah ada informasi dari suatu *Relationship* dapat didapatkan dari *Relationship* lainnya.
- c. Mempertimbangkan dimensi waktu
 Agar tidak terjadi redundansi, kita harus memastikan juga bahwa model yang sudah ada agar tetap valid di masa depan nantinya.

8. Memvalidasi model data konseptual dengan transaksi *user*

Data model yang sudah merepresentasikan kebutuhan perusahaan sudah jadi, tapi harus dipastikan kalau transaksi yang dikerjakan perusahaan dapat terpenuhi dengan model tersebut. Hal ini dapat dikerjakan dengan mendeskripsikan transaksi, dan membuat jalur transaksi.

9. Melakukan *review* model data konseptual dengan *user*

Hal ini dilakukan dengan bertujuan untuk memastikan apakah model yang telah dibuat ini sudah sesuai dengan yang diinginkan user atau belum.

2.5.2. Perancangan basis data logikal

Tahapan yang dilakukan saat merancang basis data logikal adalah:

1. Menurunkan *Relationship* untuk data model logikal

Pada tahap ini, *Relationship* yang ada pada data model konseptual diturunkan menggunakan DBDL untuk *relational database*. Dengan menggunakan DBDL, kita memberikan nama dari relasi, diikuti dengan *simple attribute* yang dibungkus dengan tanda kurung. Lalu mengidentifikasi *primary key* dan *alternate key* serta *foreign key* jika ada. Untuk menentukan *foreign key*, harus diketahui antara “*parent*” dan “*child*” *Entity*. Lalu mendeskripsikan bagaimana relasi diturunkan untuk struktur yang terjadi pada data model konseptual:

a. *Strong Entity types*

Untuk tiap *strong Entity type*, buat sebuah relasi yang terdiri dari semua *simple attribute* pada *Entity* tersebut.

b. *Weak Entity types*

Untuk tiap *weak Entity type*, buat sebuah relasi yang terdiri dari semua *simple attribute* dari *Entity* tersebut. *Primary key* dari *Entity* ini diturunkan sebagian atau sepenuhnya dari tiap *Entity* pemilik sehingga identifikasi dari *primary key* tidak dapat dilakukan hingga setelah semua *Relationship* dengan *Entity* pemilik telah dipetakan.

c. *One-to-many binary Relationship types*

Untuk relasi 1:* *binary Relationship*, *Entity* yang berada di “*one side*” menjadi *parent*, dan yang berada di “*many side*” menjadi *child*. Pada *child*, terdapat *attribute primary key* dari *parent*, yang dijadikan sebagai *foreign key*.

d. *One-to-one binary Relationship types*

Dalam menentukan representasi 1:1, yang dilihat bukanlah *cardinality*, melainkan *participation*.

i. *Mandatory participation* pada kedua sisi dari 1:1 *Relationship*

Menggabungkan kedua *Entity* kedalam satu relasi. Salah satu *primary key* dijadikan *primary key* dari relasi baru, dan yang satunya dijadikan *alternate key*.

ii. *Mandatory participation* pada satu sisi 1:1 *Relationship*

Pada sisi yang *mandatory*, dijadikan *child*, sedangkan pada sisi yang *optional*, dijadikan *parent*.

iii. *Optional participation* pada kedua sisi 1:1 *Relationship*

Untuk kasus ini, penentuan *parent* dan *child* tidak diharuskan kecuali didapatkan hubungan yang dapat membantu keputusan.

e. *One-to-one recursive Relationships types*

Untuk 1:1 *recursive Relationship*, tetap menggunakan *Entity* yang sama untuk relasi tersebut, namun ditambahkan satu *attribute* baru yang sebenarnya sama dengan *primary key*, namun diberi nama lain dan dijadikan *foreign key*.

f. *Many-to-many binary Relationship types*

Untuk *:* Relationship type*, buat satu relasi baru yang merepresentasikan *Relationship* dan memasukkan *attribute* yang menjadi bagian dari *Relationship*. Lalu, memasukkan *primary key* dari kedua *Entity* untuk dijadikan *foreign keys*.

g. *Multi-valued Attributes*

Merupakan sebuah atribut yang menyimpan banyak nilai-nilai untuk setiap kejadian dari sebuah tipe entitas.

2. Validasi relasi dengan normalisasi

Relasi yang telah dibuat dibandingkan dengan hasil normalisasi. Normalisasi bertujuan untuk memastikan bahwa set relasi memiliki jumlah *attribute* minimal yang dibutuhkan untuk menunjang kebutuhan dari perusahaan. Normalisasi melalui beberapa proses untuk memeriksa penggabungan dari *attribute-attribute* pada relasi dengan langkah-langkah 1NF, 2NF, dan 3NF.

3. Validasi relasi dengan transaksi pengguna

Tujuan dari langkah ini adalah untuk memvalidasi model data logikal untuk memastikan bahwa model data mampu mendukung transaksi-transaksi yang dibutuhkan, seperti yang ada pada *user's requirements*.

4. Memeriksa *integrity constraint*

Menurut Connolly dan Begg (2010:502) *integrity constraint* adalah *constraint* yang kita harapkan dapat melindungi basisdata dari perubahan yang membuat basis data menjadi tidak lengkap, tidak akurat, dan tidak konsisten. Tipe-tipe *integrity constraint* antara lain:

a. *Required data*

Beberapa *attribute* harus selalu memiliki nilai yang valid, dengan kata lain, *attribute* tersebut tidak diizinkan untuk bernilai *null*.

b. *Attribute domain constraints*

Tiap *attribute* memiliki domain, yaitu suatu kumpulan nilai yang diperbolehkan.

c. *Multiplicity*

Multiplicity merupakan *constraint* yang berada pada *Relationship* diantara data dalam basis data.

d. *Entity integrity*

Primary key dari suatu *Entity* tidak boleh bernilai *null*.

e. *Referential integrity*

Sebuah *foreign key* menghubungkan tiap *tuple* pada relasi *child* ke *tuple* pada relasi *parent* mengandung nilai *candidate key* yang sama. Maksud dari *referential integrity* adalah jika pada *foreign key* mengandung nilai, maka nilai tersebut harus merujuk ke *tuple* yang ada pada *parent*. Terdapat dua permasalahan mengenai *foreign key*. Permasalahan pertama adalah menentukan apakah nilai *null* diperbolehkan pada *foreign key*. Masalah kedua adalah bagaimana cara memastikan *referential integrity*. Untuk melakukannya, tentukan

existence constraints yang mendefinisikan kondisi dimana suatu *candidate key* atau *foreign key* dapat di-*insert*, *update*, atau *delete*. Ada beberapa strategi yang dapat dipertimbangkan:

- i. *NO ACTION*: Menahan penghapusan dari relasi *parent* jika ada *child tuple* yang terhubung.
 - ii. *CASCADE*: Saat *parent tuple* dihapus, secara otomatis menghapus tiap *child tuples* yang terhubung. Jika *child tuple* tersebut juga memiliki *child tuple* yang terhubung, maka juga akan ikut terhapus.
 - iii. *SET NULL*: Saat *parent tuple* dihapus, nilai *foreign key* pada semua *child tuple* yang terhubung akan secara otomatis berubah menjadi *null*.
 - iv. *SET DEFAULT*: Saat *parent tuple* dihapus, nilai *foreign key* pada semua *child tuple* yang terhubung akan secara otomatis berubah menjadi nilai *default*.
 - v. *NO CHECK*: Saat *parent tuple* dihapus, tidak melakukan apa-apa.
- f. *General constraints*

Terkahir, pertimbangkan *constraint* yang diketahui sebagai *general constraints*. Perbaharuan pada *Entity* mungkin dikendalikan oleh *constraint* yang mengatur transaksi pada dunia nyata.

5. Melakukan *review* model data logikal dengan *user*

Data model logikal seharusnya sudah lengkap dan terdokumentasi secara penuh. Namun, untuk mengkonfirmasi kebenarannya, pengguna diminta untuk meninjau data model logikal untuk memastikan bahwa mereka menyatakan kalau model tersebut benar. Jika pengguna tidak puas dengan model tersebut, maka perlu dilakukan pengulangan pada beberapa langkah sebelumnya.

6. Mempertimbangkan perkembangan di masa depan
Suatu model data logikal, seharusnya dapat memenuhi perkembangan data di masa depan.

7. Pemilihan *Database Management System*

2.5.3. Perancangan Basis data Fisikal

Menurut Connolly dan Begg (2010:523) rancangan basis data fisikal adalah proses produksi sebuah deskripsi dari implementasi sebuah basis data pada penyimpanan sekunder. Rancangan ini mendeskripsikan relasi dasar, organisasi *file*, dan *index* yang dipakai untuk mencapai akses data dan tiap *integrity constraint* dan *security measures* yang efisien.

Dalam melakukan perancangan basis data fisikal, ada beberapa langkah, antara lain:

1. Mentranslasikan data model logikal untuk *DBMS* tujuan

Langkah awal perancangan model data fisikal adalah melakukan translasi pada model data logikal menjadi sebuah bentuk yang dapat diimplementasikan pada *DBMS* tujuan. Untuk mentranslasikannya, terdapat tiga langkah, yaitu:

- a. Merancang relasi dasar

Pada tahap ini dilakukan penyusunan dan asimilasi informasi tentang relasi yang terbentuk saat melakukan perancangan basisdata logikal. Informasi tersebut diambil dari model data logikal dan *data dictionary*. Informasi-informasi yang diambil dari model data logikal antara lain nama relasi, daftar *simple attribute*, *primary key*, *alternate key*, *foreign key*, dan *referential integrity constraint* untuk tiap *foreign key* yang teridentifikasi. Sedangkan dari *data dictionary*, diambil informasi untuk tiap *attribute*, antara lain *domain*, *optional default value*, apakah dapat memiliki nilai *null*, dan apakah *attribute* tersebut diturunkan, jika ya, bagaimana cara untuk menghitungnya

b. Merancang representasi data turunan

Derived data atau data turunan adalah data yang didapatkan dari hasil perhitunag dari data lain. Data ini memiliki tanda “/” pada model data logikal. Mungkin saja pada model tidak terdapat data turunan, tetapi pada *data dictionary* terdapat data turunan. Untuk memasukkan data turunan kedalam penyimpanan, tergantung keputusan dari perancang, apakah lebih *low-cost* jika dimasukkan atau tidak. Seberapa besar media penyimpanan yang terpakai bila data turunan disimpan. Berapa lama proses pengambilan data turunan jika tidak disimpan

c. Merancang *general constraints*

Pada dunia nyata, banyak batasan-batasan data yang perlu diikuti. Hal ini dinamakan *general constraint*. Untuk membuat *general constraint*, banyak *DBMS* yang sudah menyediakan fasilitas yang memudahkan pembuatannya. Pembuatan *general constraint* dapat dilakuakn dengan menuliskan SQL pada SQL CREATE TABLE. Contohnya:

CONSTRAINT StaffNotHandlingTooMuch

CHECK (

NOTEXISTS (

SELECT StaffNo

FROM PropertyForRent

GROUPBY StaffNo

HAVINGCOUNT(*)>100

)

)

2. Merancang organisasi *file* dan *index*

Salah satu tujuan dari rancangan basisdata fisik adalah untuk menyimpan dan mengakses data secara efisien. Walaupun beberapa struktur penyimpanan dapat melakukan *bulk loading* kedalam basisdata, tapi akan tidak menjadi efisien setelah itu. Untuk itulah dibutuhkan

pemilihan struktur penyimpanan untuk mengatur basisdata dan pilih yang lain untuk operasional.

a. Menganalisa transaksi

Untuk menganalisa transaksi, dilakukan identifikasi kriteria performa seperti:

- i. Transaksi yang berjalan sering dan akan terkena dampak yang signifikan pada performa
- ii. Transaksi yang penting untuk operasi bisnis
- iii. Di hari/minggu saat ada permintaan *database* yang tinggi
- iv. Untuk mencari area basisdata yang memiliki masalah dalam akses yang banyak dengan cara Pemetaan semua jalur transaksi ke relasi; Mencari tahu frekuensi informasi; Menganalisa penggunaan data

b. Memilih organisasi *file*

Agar akses data lebih efisien, perlu diadakan pemilihan organisasi *file*. Beberapa pilihan organisasi *file* antara lain:

- i. *Heap*
- ii. *Hash*
- iii. *IndexedSequentialAccessMethod* (ISAM)
- iv. *B⁺-Tree*
- v. *Clusters*

c. Memilih *index*

Satu pendekatan untuk memilih organisasi *file* yang baik untuk sebuah relasi adalah dengan cara menjaga *tuple* tak terurut dan membuat sebanyak mungkin *secondary indexes*. Selain itu, bisa juga menurutkan *tuple* pada sebuah relasi dengan cara menentukan sebuah *primary* atau *cluster index*.

1. Menentukan *index*

SQL untuk membuat sebuah *index* :

```
CREATE UNIQUE INDEX PropertyNoInd ON
PropertyForRent(propertyNo);
```

SQL untuk membuat *clustering-index* :

```
CREATE INDEX StaffNoInd ON PropertyForRent(staffNo)
CLUSTER;
```

2. Memilih *secondary indexes*

Secondary indexes menyediakan mekanisme untuk menentukan *key* tambahan untuk sebuah relasi dasar yang bias dipakai untuk mengambil data dengan lebih efisien.

3. Panduan untuk memilih sebuah “*wish-list*” dari *index*

Panduan untuk memilih “*wish-list*” *index* antara lain :

- a. Jangan mengindex relasi kecil. Lebih efisien untuk melakukan pencarian relasi pada *memory* daripada membuat struktur *index* tambahan.
- b. Melakukan peng-*index*-an pada *primary key* dari sebuah relasi jika itu bukan sebuah *key* dari organisasi *file*.
- c. Menambahkan sebuah *secondary index* pada *foreign key*, jika sering diakses.
- d. Menambahkan *secondary key* pada tiap *attribute* yang sering dipakai sebagai *secondary key*.
- e. Menambahkan *secondary index* pada *attribute* yang sering terlibat pada kriteria *selection* atau *join*, *ORDER BY*, *GROUP BY*, dan operasi lain yang menyangkut dengan *sorting*.
- f. Menambahkan *secondary index* pada *attribute* yang terlibat di *built-in aggregate function*, bersamaan dengan tiap *attribute* yang dipakai untuk *function* itu.
- g. Menambahkan *secondary index* pada *attribute* yang mengeluarkan hasil di sebuah *index-only plan*.
- h. Menghindari peng-*index*-an pada *attribute* atau relasi yang sering diperbaharui.
- i. Menghindari peng-*index*-an pada *attribute* jika *query* akan mengambil banyak *tuple* pada sebuah relasi.
- j. Menghindari peng-*index*-an pada *attribute* yang berisi *string* karakter yang panjang.

4. Menghilangkan *index* dari *wish-list*

Index tidak selalu meningkatkan efisiensi pada saat mengakses data. Untuk itu diperlukan penghilangan pada *index* yang malah mengurangi efisiensi tersebut.

d. Memperkirakan kebutuhan *disk space*

Perancang harus memperkirakan banyaknya *disk space* yang diperlukan untuk menyimpan *database*, seandainya *hardware* baru harus diadakan.

3. Merancang *View* untuk pengguna

Langkah ini bertujuan untuk melakukan perancangan pada *view* untuk pengguna yang diidentifikasi saat pengumpulan kebutuhan dan analisa tahapan dalam siklus hidup pengembangan sistem basis data.

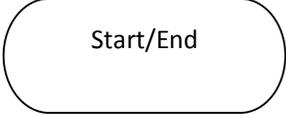
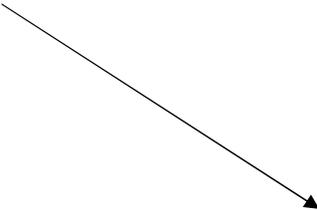
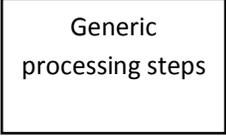
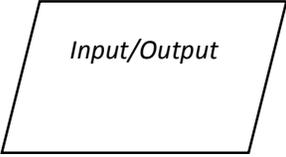
4. Merancang mekanisme *security*

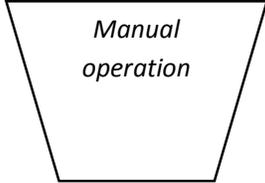
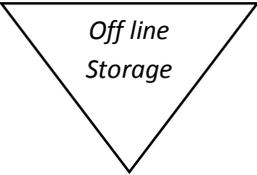
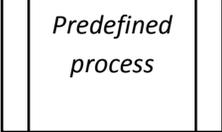
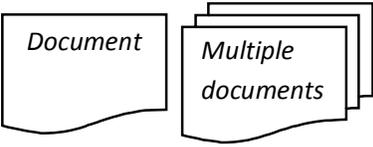
Langkah ini bertujuan untuk melakukan perancangan mekanisme keamanan untuk basisdata yang sudah ditentukan oleh pengguna saat tahap pengumpulan kebutuhan di siklus hidup pengembangan sistem basis data.

2.6. *Flowchart*

Flowchart adalah suatu diagram yang merepresentasikan sebuah aliran proses atau algoritma yang menggunakan simbol dari berbagai bentuk dan aliran prosesnya disimbolisasikan dengan garis panah. *Flowchart* memberikan solusi langkah demi langkah dari sebuah masalah yang ingin dipecahkan. Kegunaan *flowchart* adalah untuk menganalisis, merancang, mendokumentasikan serta mengelola suatu proses atau program di berbagai bidang. Sebuah proses atau *action* direpresentasikan dalam sebuah kotak, dan tanda panah yang menghubungkan kotak-kotak ini mewakili aliran atau arah aliran data. Berikut ini adalah simbol-simbol *flowchart* yang sering digunakan beserta deskripsinya:

Tabel 2. 1 Tabel komponen *Flowchart*

	<p><i>Start/End symbol:</i> Merepresentasikan awal mulai atau berhenti dari sebuah flowchart. Pada umumnya digambarkan sebagai sebuah oval.</p>
	<p><i>Arrow:</i> Merepresentasikan aliran dari simbol sebelumnya ke simbol selanjutnya untuk menggambarkan aliran proses yang terjadi. Disimbolkan sebagai sebuah panah solid atau putus-putus.</p>
	<p><i>Generic processing steps:</i> Merepresentasikan sebuah proses umum dalam sebuah penyelesaian masalah. Contoh : $X=X+1$. Disimbolkan sebagai sebuah persegi panjang.</p>
	<p><i>Input/Output:</i> Merepresentasikan masuk keluarnya data. Contoh: Baca x. Disimbolkan dengan sebuah jajar genjang.</p>
	<p><i>Prepare Conditional:</i> Merepresentasikan kondisi percabangan yang harus diambil sesuai dengan kondisi yang berlaku. <i>Arrow</i> yang akan keluar dari <i>prepare conditional</i> ada 2 yaitu ya dan tidak. Jika kondisi yang ditanyakan sesuai dengan ya atau tidak maka akan mengalir sesuai dengan yang bersangkutan. Disimbolkan dengan sebuah belah ketupat.</p>

 <p>Manual operation</p>	<p><i>Manual operation:</i> Merepresentasikan sebuah proses yang tidak dilakukan oleh komputer.</p>
 <p>Off page connector</p>	<p><i>Off page connector:</i> Merepresentasikan kelanjutan proses <i>flowchart</i> ke halaman berikutnya jika <i>flowchart</i> belum selesai dan <i>flowchart</i> berada di akhir halaman.</p>
 <p>Off line Storage</p>	<p><i>Off line storage :</i> Merepresentasikan bahwa data pada simbol ini akan disimpan.</p>
 <p>Predefined process</p>	<p><i>Predefined process:</i> Merepresentasikan sebuah proses yang kompleks dan proses tersebut bisa saja ditampilkan di <i>flowchart</i> yang terpisah.</p>
 <p>Document Multiple documents</p>	<p><i>Document/Multiple documents:</i> Merepresentasikan sebuah proses untuk menghasilkan satu dokumen atau banyak dokumen.</p>

2.7. T
e
o
r
i

y
a
n
g

t
e
r
k
a
i
t

t
e
m
a

penelitian

2.7.1. Internet

Menurut Conolly dan Begg (2010:1024) *Internet* adalah sebuah kumpulan jaringan komputer yang terpisah dan saling berhubungan di seluruh dunia.

Menurut William dan Sawyer (2010:18) *Internet* adalah suatu jaringan komputer dari seluruh dunia yang dapat menghubungkan ratusan ribu jaringan yang lebih kecil.

2.7.2. Website

Menurut Connolly dan Begg (2010:1028) *website* adalah sistem yang menyediakan sarana dari informasi browsing di *Internet* dengan cara *non-sequensial* yang menggunakan *hyperlink*. Sedangkan menurut William dan Sawyer (2010:65) *website* adalah lokasi pada suatu komputer di *web* dan memiliki alamat yang unik antara satu dengan yang lainnya, sedangkan menurut Kotler dan Armstrong (2013:537) *website* harus dibangun untuk membantu pelanggan, mengumpulkan umpan balik dari pelanggan, dan melengkapi saluran penjualan daripada menjual produknya secara langsung. *Website* harus menawarkan informasi yang detail dan fitur-fitur lain yang dapat menjawab pertanyaan, membangun hubungan dengan pelanggan menjadi lebih dekat, dan menghasilkan eksistensi perusahaan.

2.7.3. e-Business

Menurut Conolly dan Begg (2010:1029), *e-bussiness* adalah integrasi lengkap dari sebuah teknologi internet ke dalam infrastruktur ekonomi bisnis, dalam segi penjualan, pelayanan, dan pemasaran akan memiliki keuntungan menjadikan komunikasi lebih cepat dan efisien sehingga membantu meningkatkan produktivitas.

Menurut Indrajani (2014:53) *e-bussiness* adalah penggunaan teknologi informasi serta komunikasi baik oleh organisasi, individu dan pihak-pihak terkait untuk mengelola proses bisnis yang utama dan dapat memberikan keuntungan, keamanan, fleksibilitas, integrasi, optimasi, efisiensi, dan peningkatan profit.

2.7.4. Hypertext Transfer Protocol (HTTP)

Menurut Conolly dan Begg (2010:1029), HTTP adalah sebuah aturan yang dipakai untuk mengirimkan halaman *web* melalui *internet*. Menurut William dan Sawyer (2010:66) *HTTP* adalah aturan yang mengizinkan *browser* dapat terhubung dengan *web server*. Kebanyakan semua *web* sudah di asumsikan dengan *http://*, jadi kita tidak harus mengetiknya.

2.7.5. Uniform Resource Locators (URL)

Menurut Conolly dan Begg (2010:1033), *URL* adalah serangkaian karakter *alphanumeric* yang dapat mempresentasikan alamat dari sumber yang ada di *internet* dan bagaimana cara sumber tersebut di akses.

Menurut William dan Sawyer (2010:65) *URL* adalah karakter *string* yang menunjuk bagian informasi spesifik dari *web*.

2.7.6. Browser

Menurut William dan Sawyer (2010:64) *browser* adalah perangkat lunak yang dapat menemukan dan mengizinkan untuk akses ke berbagai bagian dari *web*.

2.7.7. HTML

Menurut Conolly dan Begg (2010:1031), *HTML* merupakan format bahasa yang digunakan dalam merancang halaman *web*.

Menurut Indrajani (2014:54) *HTML* merupakan suatu bahasa standar untuk mendesain seluruh halaman *web*, dan tampilan *webnya* dapat di atur, mempublikasikan dokumen secara *online*, membuat sebuah *form online* untuk pendaftaran, transaksi serta dapat juga menambahkan objek-objek gambar, suara, *video*, ke dalam *HTML*.

Menurut William dan Sawyer (2010:68) *HTML* adalah suatu instruksi khusus untuk menentukan format dan struktur pada dokumen ke dokumen *multimedia* yang lainnya dalam *web*.

2.7.8. Hypertext Links

Menurut William dan Sawyer (2010:68) *hypertext links* adalah koneksi antara dokumen yang satu ke dokumen yang lainnya yang saling terkait, baik itu *web* yang sama atau tidak. Biasanya *hyperlink* ditandai dengan warna yang berbeda atau kata yang digaris bawahi.

2.7.9. PHP

Menurut Sidik (2012:4) PHP merupakan bahasa pemrograman *script-script* yang membuat HTML secara *on the fly* dan dieksekusi di dalam *server web*. Dengan menggunakan PHP akan memudahkan *maintenance* serta *update* suatu situs akan menjadi lebih mudah.

PHP banyak digunakan karena kemudahannya dan penggunaan *PHP* paling banyak digunakan untuk situs yang berisi konten bisnis, teknologi dan dewasa.

Menurut Sidik (2012:311) Salah satu dari keunggulan *PHP* sebagai bahasa pemrograman karena banyaknya fasilitas (*library* fungsi) yang dapat memungkinkan mengakses basis data.

Menurut Connolly dan Begg (2010:1043) *PHP* merupakan sebuah bahasa *script* yang didukung dari banyak *web server*, termasuk Apache HTTP Server dan Internet Information Server Microsoft.

Beberapa keunggulan dari *PHP* dibandingkan dengan bahasa-bahasa *scripting* lainnya ialah penulisan halaman dihasilkan dengan cepat dan dinamis, sangat mudah dipelajari dan digunakan, dan sejumlah modul ekstensi telah disediakan sehingga dapat mendukung hal-hal seperti konektivitas *database*, *mail*, dan *XML*.

2.7.10. XAMPP

Sebuah *software web server* yang berperan sebagai jembatan yang menjembatani antara browser yang khusus menerima input dari pengguna dan dikonversi ke *server* melalui *software web server* ini. *XAMPP* merupakan tool yang menyediakan paket perangkat lunak ke dalam satu buah paket.

2.7.11. Interaksi Manusia dan Komputer

Menurut Shneiderman dan Plaisant (2010:22), Interaksi Manusia dan Komputer (IMK) atau *Human Computer Interface* adalah disiplin ilmu yang berhubungan dengan pembuatan, evaluasi, dan implementasi sistem komputer yang interaktif sehingga memungkinkan manusia (*user*) untuk berkomunikasi dengan komputer itu sendiri. Tujuan penerapan disiplin ilmu ini adalah untuk

memberikan sebuah antarmuka pemakai (*user interface*) yang sesuai dengan kebutuhan si pemakai. Sedangkan, antarmuka pemakai itu sendiri adalah sebuah bagian dari sistem komputer yang bisa berinteraksi dengan manusia (*user*).

2.7.12. Perancangan Desain *Interface*

Menurut Shneiderman dan Plaisant (2010:88), dalam merancang desain antarmuka (*interface*) terdapat delapan prinsip aturan emas (*Eight Golden Rules of interface design*) yang harus diperhatikan yaitu :

1. Konsistensi

Istilah – istilah yang sejenis harus dipakai pada *prompt*, menu, layar bantuan (*help screen*) dan konsistensi dari warna, *layout*, kapitalisasi, *fonts* dan lain-lain harus sesuai dengan *interface* lainnya.

2. Memenuhi kegunaan yang universal

Mengenali kebutuhan pengguna yang beragam serta desain untuk perubahan konten. Menambahkan fitur untuk para pemula serta penjelasan fitur dan fitur untuk para ahli juga seperti akses jalan pintas dan akses cepat yang dapat juga dalam memperkaya desain *user interface* serta meningkatkan kualitas sistem.

3. Memberikan respon atau umpan balik yang informatif

Untuk setiap tindakan dari user, sebaiknya diberikan suatu sistem umpan balik. Untuk tindakan yang sering dilakukan dan tidak terlalu penting, dapat diberikan suatu umpan balik yang sederhana. Namun, ketika tindakan tersebut jarang dilakukan dan merupakan tindakan yang penting, respon atau umpan balik yang diberikan haruslah lebih substansial atau besar.

4. Merancang dialog untuk menghasilkan suatu penutupan

Urutan tindakan sebaiknya diorganisir ke dalam beberapa kelompok dengan bagian awal, tengah, dan akhir. Umpan balik yang informatif akan memberikan indikasi bahwa cara yang dilakukan oleh pemakai sudah benar dan bisa dijadikan sebuah persiapan untuk tindakan-tindakan selanjutnya.

5. Memberikan pencegahan dan penanganan kesalahan yang sederhana kesalahan fatal yang mungkin dilakukan oleh *user* bisa dicegah dengan sistem yang sudah dirancang sedemikian rupa. Jika terjadi kesalahan, sistem dapat mendeteksi kesalahan dengan cepat dan memberikan penanganan yang sederhana dan mudah dipahami oleh *user*.
6. Mudah kembali ke tindakan sebelumnya
Hal ini dapat mengurangi kekhawatiran karena pemakai (*user*) tahu bahwa kesalahan yang dilakukan bisa dibatalkan dengan segera sehingga memberikan keberanian kepada pemakai (*user*) untuk mengeksplorasi pilihan-pilihan lain yang belum biasa digunakan.
7. Mendukung pusat kendali internal
Peran antara pemakai dan sistem sangat penting di sini. Pemakai ingin menjadi pengontrol sistem, sedangkan sistem akan merespon tindakan yang dilakukan oleh pengguna, bukan sebaliknya yaitu sistem yang mengontrol pengguna. Dengan demikian, sistem harus dirancang sedemikian rupa agar pengguna menjadi sebuah inisiator daripada menjadi sebuah responden.
8. Mengurangi beban ingatan jangka pendek
Tampilan yang sederhana sangat dibutuhkan mengingat keterbatasan ingatan manusia dalam memori jangka pendek. Informasi yang diberikan dalam beberapa halaman sebaiknya disatukan kedalam satu halaman. Selain itu, diberikan juga waktu pelatihan yang cukup untuk menangani urutan tindakan-tindakan yang kompleks.

Prinsip-prinsip tersebut memberikan penjelasan tentang bagaimana meningkatkan produktivitas dari pemakai dengan cara menyediakan prosedur-prosedur yang sederhana, tampilan yang mudah dimengerti dan umpan balik yang cepat untuk meningkatkan keahlian dari si pemakai.

2.7.13. Pengertian Pemasaran

Pemasaran menurut Kotler dan Armstrong (2013:28) adalah mengatur pengelolaan yang menguntungkan hubungan dengan pelanggan. Tujuan utama dari *marketing* itu sendiri ialah menarik setiap pelanggan yang baru dengan memberikan janji nilai yang unggul dari yang lainnya dan terus tumbuh berkembang dengan memberikan sebuah kepuasan yang tak ternilai kepada pelanggan.

2.7.14. Pengertian Pemasaran Online

Menurut Kotler dan Armstrong (2013:532) pemasaran *online* adalah suatu upaya dalam memasarkan produk serta pelayanan dengan membangun hubungan dengan pelanggan melalui *internet*. Menurut Kotler dan Armstrong (2013:534) pembeli di *internet* berbeda dengan konsumen tradisional. Pemasaran tradisional menargetkan konsumen yang pasif, berbeda dengan pemasaran secara *online* yang menargetkan orang-orang yang aktif dalam memilih *website* yang akan dikunjunginya dan informasi yang mereka terima tentang produk serta dalam kondisinya. Sekarang banyak orang yang melakukan pemesanan melalui *online* untuk memesan berbagai barang dan mereka akan mencari *website* yang menyediakan kebutuhan mereka.

Dengan menggunakan pemasaran *online* dapat menjangkau pelanggan baru, melayani pelanggan yang saat ini akan menjadi lebih efektif, dan membeli dengan efisiensi dan harga yang lebih baik karena menawarkan informasi produk yang terkait, pembelian pelanggan, dan dukungan pelayanan pelanggan secara *online*.

2.7.15. Pengertian Penjualan

Menurut Mulyadi (2008:160) Penjualan adalah “Suatu kegiatan yang terdiri dari transaksi penjualan barang atau jasa, secara kredit maupun tunai”.

Sedangkan Menurut Soemarso .S.R, (2009 :160) Penjualan adalah Jumlah yang dibebankan kepada pembeli untuk barang dagang yang diserahkan merupakan pendapatan perusahaan yang bersangkutan.

Berdasarkan kedua pernyataan di atas, maka dapat disimpulkan bahwa penjualan, khususnya penjualan barang merupakan kegiatan menjual barang yang diproduksi sendiri atau dibeli dari pihak lain untuk dijual kembali kepada konsumen secara kredit maupun tunai.

Jadi secara umum penjualan pada dasarnya terdiri dari dua jenis yaitu penjualan tunai dan kredit. Penjualan tunai terjadi apabila penyerahan barang atau jasa segera diikuti dengan pembayaran dari pembelian, sedangkan penjualan kredit ada tenggang waktu antara saat penyerahan barang atau jasa dalam penerimaan pembelian.

Keuntungan dari penjualan tunai adalah hasil dari penjualan tersebut langsung terealisasi dalam bentuk kas yang dibutuhkan perusahaan untuk mempertahankan likuiditasnya. Sedangkan dalam rangka memperbesar volume penjualan, umumnya perusahaan menjual produknya secara kredit. Penjualan kredit tidak segera menghasilkan pendapatan kas, tapi kemudian menimbulkan piutang. Kerugian dari penjualan kredit adalah timbulnya biaya administrasi piutang dan kerugian akibat piutang tak tertagih.

2.7.16. Pengertian Pelayanan

Menurut Kotler dan Armstrong (2013:248) Pelayanan adalah sesuatu yang terdiri dari kegiatan, manfaat bahkan kepuasan yang diberikan dan pada dasarnya tidak terwujud dan tidak juga mengakibatkan kepemilikan apapun. Pelayanan menjadi komoditas utama, banyak perusahaan berkembang ke tingkat yang lebih maju dalam menciptakan sebuah nilai yang berkesan bagi pelanggan dengan membedakan pelayanan mereka dengan yang lain sehingga menciptakan dan mengelola pengalaman yang dirasakan pelanggan dengan perusahaan.

2.8. Hasil Penelitian Sebelumnya

1. Jurnal Internasional

- a. *Web-based Dealership Management System* karya Jerome, Sandra L. (Port St Lucie, FL, US) Jerome, Keith (Port St Lucie, FL, US)

Tujuan penelitian untuk menyediakan laporan keuangan dan pelacakan penjualan layanan kendaraan dan suku cadang dengan aplikasi berbasis *web*. Metode penelitian yang digunakan adalah menggunakan relasional basis data pada *web server* untuk mengintegrasikan alat yang dapat melihat laporan keuangan, penjualan suku cadang, kendaraan dan persediaan. Metode dalam merancangan sistem dengan menggunakan UML. Hasil yang dicapai adalah desain arsitektur sistem ini yang mengintegrasikan aplikasi *web* dan memberikan sebuah penjelasan yang terkait agar meningkatkan kinerja dan juga dapat menampilkan catatan historis pengguna dan dapat di ubah atau ditambahkan. Simpulan dari penelitian ini adalah dengan adanya aplikasi berbasis web ini dapat membantu untuk mengembangkan usaha serta dapat membantu dalam hal melihat laporan penjualan, keuangan serta transaksi.

- b. Analisis Dan Perancangan Sistem Basis Data Penjualan, Pembelian, Dan Persediaan Bahan Baku Pada CV Cipta Selaras Plastik karya Doddy Koeswandy, Gandi Sutejo, Lius Hendra, Fanny Riyanti Wiros.

Tujuan penelitian untuk menganalisis dan merancang sistem basis data untuk transaksi penjualan, pembelian, dan persediaan bahan baku pada CV Cipta Selaras Plastik. Kemudian, akan dibuat juga aplikasi yang akan mendukung basis data tersebut. Metode penelitian yang digunakan adalah metode analisis dan perancangan. Metode analisis digunakan untuk mengetahui permasalahan yang ada pada CV Cipta Selaras Plastik khususnya dalam bidang penjualan, pembelian, dan persediaan bahan baku.

Sedangkan metode perancangan dilakukan dalam rangka merancang sistem basis data yang melalui tiga tahapan, yaitu perancangan konseptual, logikal, dan fisik. Hasil yang dicapai adalah terbuatlah suatu sistem basis data yang mengintegrasikan data-data yang berhubungan dengan transaksi penjualan, pembelian, dan persediaan bahan baku. Prosedur yang harus dilakukan adalah user harus meng-input data baru setiap kali ada transaksi yang terjadi sehingga tidak terjadi perbedaan antara kenyataan sebenarnya dengan data yang ada pada basis data. Simpulan yang dapat diambil adalah bahwa dengan adanya sistem basis data yang dibuat ini maka dapat membantu dalam hal pengorganisasian data, khususnya yang terkait dalam hal penjualan, pembelian, dan persediaan bahan baku. Saran yang dapat diberikan adalah agar data sebaiknya di-backup sehingga dapat menghindari hal yang tidak diinginkan.

- c. Aplikasi penjualan suku cadang dan jasa perbaikan sepeda motor(studi kasus: fans motor) karya Taufik Akbar, Ely Rosely, Rochmawati.

Tujuan Penelitian ini adalah membantu perusahaan memudahkan dalam pencarian laporan penjualan serta perbaikan. Metode yang digunakan adalah metode perancangan sistem dengan menggunakan DFD. Hasil yang dicapai adalah terbuatannya suatu aplikasi yang dapat membantu perusahaan dalam menjalankan usahanya, memudahkan dalam pengoperasian serta pencarian data, data yang terstruktur dan mudah dalam mengaksesnya sehingga proses transaksi menjadi lebih efektif dan efisien. Simpulan yang dapat diambil adalah dengan adanya aplikasi ini dapat memudahkan perusahaan dalam mengelola laporan penjualan dan perbaikan, menyimpan data penjualan dan perbaikan sehingga data dapat berguna untuk kemajuan perusahaan.

