BAB 2

TINJAUAN PUSTAKA

2.1 Teori yang Berkaitan dengan *Database*

2.1.1 Data

Menurut Connolly dan Begg (2005:20) data adalah komponen terpenting dalam sistem manajemen basis data atau *Basis data Management System* (DBMS), yang berasal dari sudut pandang *end-user*. Data berperan sebagai penghubung antara komponen mesin dan komponen manusia.

Menurut Indrajani (2011:2) data adalah fakta atau observasi mentah yang biasanya mengenai fenomena fisik atau transaksi bisnis. Lebih khusus lagi, data adalah ukuran objektif atribut (karakteristik) dari entitas seperti orang-orang, tempat, benda, atau kejadian.

2.1.2 Basis Data

Menurut Connolly dan Begg (2005:15), basis data (*database*) adalah sekumpulan data yang berelasi secara logis beserta deskripsi datanya, yang dirancang untuk memenuhi kebutuhan informasi dari suatu organisasi.

Basis data merupakan suatu penyimpanan data yang besar dan dapat digunakan secara bersamaan oleh *user d*an departemen. Basis data tidak hanya dimiliki oleh satu departemen saja tetapi basis data merupakan sumber daya organisasi yang digunakan bersama. Basis data juga tidak hanya menyimpan data operasional organisasi tetapi basis data juga menyimpan deskripsi dari data itu sendiri.

2.1.3 Sistem Basis Data

Menurut Connolly dan Begg (2005:4), sistem basis data adalah kumpulan dari program aplikasi yang berinteraksi dengan basis data bersamaan dengan sistem manajemen basis data atau *Database Management System* (DBMS) dan basis data itu sendiri.

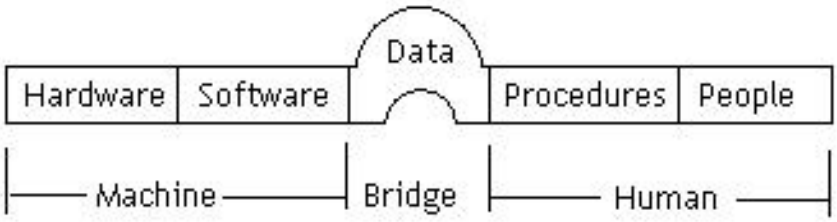
2.1.4 Sistem Manajemen Basis Data

Menurut Connolly dan Begg (2005:16), sistem manajemen basis data adalah suatu sistem perangkat lunak yang memungkinkan *user* untuk mendefinisikan, menciptakan, memelihara, dan mengontrol akses ke basis data.

DBMS merupakan perangkat lunak yang berinteraksi dengan program aplikasi *user* dan basis data. Sebuah DBMS menyediakan fasilitas-fasilitas sebagai berikut:

1. Memungkinkan *user* untuk mendefinisikan basis data menggunakan *Data Definition Language* (*DDL*). *DDL* membantu *user* dalam menentukan tipe data, struktur, dan batasan data yang akan disimpan dalam basis data.
2. Memungkinkan *user* untuk melakukan *insert*, *update*, *delete*, dan *retrieve* data dari basis data menggunakan *Data Manipulation Language* (*DML*).
3. Menyediakan akses terkontrol pada basis data. Contohnya:
    a. Sistem keamanan (*security system*), mencegah *user* yang tidak berwenang mengakses basis data;
    b. Sistem integritas (*integrity system*), memelihara konsistensi data yang disimpan;
    c. Sistem kontrol konkurensi (*concurrency control system*), memungkinkan agar data dapat dipakai bersama-sama oleh *user* lainnya:
    d. Sistem kontrol perbaikan (*recovery control system*), mengembalikan basis data ke kondisi sebelumnya jika terjadi kegagalan atau kerusakan pada perangkat keras atau perangkat lunak;
    e. Katalog yang dapat diakses *user* (*user-accessible catalog*), berisi deskripsi data dalam basis data.

Menurut Connolly dan Begg (2005:18-21), terdapat lima komponen utama dalam lingkungan DBMS, yaitu:



Gambar 2.1 Komponen Utama DBMS

(Sumber: www.noucamp.org/it1/access/images/dbmsComponents.PNG)

1. Perangkat Keras (*Hardware*)

   DBMS dan aplikasinya membutuhkan *hardware* untuk dapat beroperasi. Contoh *hardware* yang digunakan dapat berupa *single personal computer*, *single mainframe*, dan jaringan komputer. *Hardware* yang digunakan harus berdasarkan kebutuhan organisasi dan DBMS yang digunakan dalam sebuah perusahaan.

2. Perangkat Lunak (*Software*)

   Komponen *software* meliputi *software* DBMS itu sendiri dan program aplikasi, serta sistem operasi termasuk *software* jaringan jika DBMS sedang dijalankan melalui jaringan.

3. Data

   Data merupakan komponen terpenting dalam lingkungan DBMS, khususnya sudut pandang dari *end-user* mengenai data yang akan diolah dalam DBMS. Data berperan sebagai penghubung antara komponen mesin dan komponen manusia.

4. Prosedur (*Procedures*)

   Prosedur merupakan instruksi dan aturan yang mengatur perancangan dan penggunaan basis data. *User* sistem dan *staff* yang mengatur basis data membutuhkan prosedur yang terdokumentasi dalam menjalankan suatu sistem. Prosedur tersebut dapat berisi instruksi-instruksi berupa:

   a. *Login* ke DBMS;

b. Menggunakan fasilitas DBMS atau program aplikasi tertentu;
c. Menjalankan dan menghentikan DBMS;
d. Melakukan *backup copies* dari basis data;
e. Mengatasi kegagalan pada *hardware* atau *software*;
f. Mengubah struktur basis data, meningkatkan kinerja, atau membuat arsip penyimpanan data sekunder.

5. Manusia (*People*)

Manusia merupakan komponen terakhir yang terlibat dengan sistem. Menurut Connolly dan Begg (2005:21-24), terdapat 4 macam tipe komponen manusia yang berpartisipasi dalam DBMS yaitu:

a. *Data and Database Administrators*

Data Administrator (DA) bertanggung jawab untuk manajemen sumber daya data termasuk perencanaan basis data, pengembangan dan standar pemeliharaan, kebijakan dan prosedur, dan perancangan basis data secara konseptual/logikal.

Database Administrator (DBA) bertanggung jawab untuk realisasi fisik dari basis data, termasuk perancangan basis data secara fisikal dan implementasinya, kontrol keamanan dan integritas, pemeliharaan sistem operasional, dan memastikan kepuasan *user* atas kinerja aplikasi.

b. *Database Designers*

Database Designers terdiri dari dua tipe yaitu:

1) *Logical Database Designer*

Mengidentifikasi data (yaitu entitas dan atribut), hubungan antar data, dan batasan data yang akan disimpan dalam basis data.

2) *Physical Database Designer*

Menentukan bagaimana perancangan basis data logikal dapat direalisasikan secara fisikal. Misalnya, memetakan rancangan basis data logikal ke dalam sebuah tabel dan terintegritas dengan batasan, pemilihan spesifikasi struktur penyimpanan dan pemilihan metode

akses untuk kinerja yang lebih baik, serta merancang beberapa ukuran keamanan yang dibutuhkan oleh data.

c. *Application Developers*

Application Developers bertanggung jawab untuk membuat dan mengembangkan program aplikasi yang telah diimplementasikan, sehingga dapat digunakan oleh *end-users*.

d. *End-Users*

End-users adalah *user* akhir basis data yang telah dirancang, diimplementasikan, dan sedang dipelihara untuk menyajikan kebutuhan informasi yang diperlukan.

End-users dapat diklasifikasikan menjadi:

1) *Naïve Users*, yaitu *user* yang tidak peduli akan DBMS. Mereka mengakses basis data melalui program aplikasi yang ditulis khusus untuk membuat operasi sesederhana mungkin.
2) *Sophisticated Users*, yaitu *user* yang lebih canggih, mengerti struktur basis data dan fasilitas dalam DBMS.

Menurut Connolly dan Begg (2005:48-52) DBMS memiliki fungsi sebagai berikut:

1. *Data storage, retrieval, and update*

   Sebuah DBMS pasti menyediakan kemampuan bagi user untuk menyimpan, mengambil, dan mengubah data dalam basis data.

2. *A user-accessible catalog*

   Sebuah DBMS pasti menyediakan katalog dimana deskripsi data disimpan di dalamnya dan katalog tersebut dapat diakses oleh *user*.

3. *Transaction support*

   Sebuah DBMS pasti menyediakan suatu mekanisme yang dapat menjamin semua kegiatan perubahan baik yang berhubungan dengan transaksi yang dibuat atau tidak sama sekali dibuat.

4. *Concurrency control services*

   Sebuah DBMS pasti menyediakan mekanisme untuk menjamin bahwa basis data ter-*update* dengan benar ketika beberapa *user* melakukan perubahan basis data tersebut secara bersamaan.

5. *Recovery services*

   Sebuah DBMS pasti menyediakan mekanisme untuk memperbaiki basis data apabila terjadi kerusakan.

6. *Authorization services*

   Sebuah DBMS pasti menyediakan mekanisme untuk menjamin bahwa hanya *user* yang memiliki otoritas yang dapat mengakses basis data.

7. *Support for data communication*

   Sebuah DBMS dapat terintegrasi dengan perangkat lunak komunikasi, misalnya dapat pengirim permintaan sebagai pesan dan tanggapan.

8. *Integrity services*

   Sebuah DBMS pasti menyediakan cara untuk menjamin bahwa data dalam basis data dan perubahannya sesuai dengan aturan tertentu.

9. *Services to promote data independence*

   Sebuah DBMS pasti memiliki fasilitas-fasilitas yang mendukung program-program yang tidak tergantung dari struktur basis data sebenarnya.

10. *Utility services*

    Sebuah DBMS harus menyediakan sekumpulan layanan utilitas agar basis data dapat diatur atau dikelola secara efektif. Layanan utilitas ini biasanya meliputi:

    a. Fasilitas meng-*import* basis data;
    b. Fasilitas memonitor penggunaan basis data dan operasinya;
    c. Program analisis statistikal untuk analisis kinerja dan statistik penggunaan;
    d. Fasilitas pengaturan kembali indeks.

Menurut Connolly dan Begg (2005:26-29), keuntungan DBMS adalah:

1. *Control of data redundancy*

   Pendekatan basis data yang berusaha untuk menghilangkan redundansi dengan mengintegrasikan *file* sehingga data yang sama tidak disimpan. Pendekatan basis data tidak menghilangkan

redundansi sepenuhnya, tetapi mengendalikan jumlah redundansi yang ada dalam basis data.

2. *Data consistency*

   Dengan menghilangkan atau mengontrol redundansi dapat mengurangi risiko terjadinya inkonsistensi data.

3. *Improved data integrity*

   Jika data diakses melalui DBMS, maka DBMS dapat menjalankan batasan integritas data. Integritas dalam basis data meliputi validitas dan konsistensi data yang tersimpan. Umumnya integritas dinyatakan dalam *constraint*.

4. *Improved security*

   Dengan meningkatkan keamanan basis data, maka basis data akan terlindungi dari user yang tidak berwenang mengakses basis data.

5. *Enforcement of standards*

   Integrasi memungkinkan *DBA* untuk mendefinisikan dan menerapkan standar yang diperlukan. Dengan adanya penggunaan data secara bersamaan, maka apabila terdapat penambahan tabel, *field,* tipe data, hak akses harus dibuat standar dan dokumentasinya.

6. *Increased productivity*

   DBMS menyediakan banyak fungsi standar dan semua tingkat rendah penanganan *file* dalam program aplikasi. Penyediaan fungsi-fungsi ini memungkinkan *programmer* untuk berkonsentrasi pada fungsi tertentu yang diperlukan oleh *user*. Hal tersebut akan meningkatkan produktivitas *programmer* serta mengurangi waktu pengembangan.

7. *Increased concurrency*

   Dalam beberapa sistem berbasis *file*, apabila dua atau lebih user mengakses *file* yang sama secara bersamaan, akan menyebabkan hilangnya informasi atau bahkan hilangnya integritas. Dengan menggunakan DBMS, hal-hal tersebut dipastikan tidak akan terjadi karena DBMS melakukan penjadwalan akses data secara bersamaan.

8. *Improved backup and recovery data*

   Dengan meningkatkan *backup* data maka dapat mengurangi kegagalan sistem atau program aplikasi. Jika terjadi kesalahan maka *backup* data dapat di-*restored*.

Sedangkan kerugian DBMS menurut Connolly dan Begg (2005:29-30) adalah sebagai berikut:

1. *Complexity*

   Banyaknya fungsi yang dimiliki oleh suatu DBMS membuat DBMS menjadi kompleks. Perancang dan pengembang basis data, DA, DBA, dan *end-users* harus benar-benar memahami semua fungsi tersebut.

2. *Size*

   Fungsi yang kompleks dan luas membuat DBMS menjadi *software* yang sangat besar, memerlukan banyak ruang *harddisk*, dan membutuhkan sejumlah besar memori untuk dapat menjalankannya secara efisien.

3. *Additional hardware costs*

   Kebutuhan tempat penyimpanan bagi DBMS dan basis data memerlukan tempat penyimpanan tambahan. Untuk mencapai kinerja yang maksimal, DBMS memerlukan mesin yang lebih besar lagi sehingga akan membutuhkan tambahan biaya yang tidak sedikit.

4. *Cost of conversion*

   Dalam beberapa situasi, biaya DBMS dan *hardware* tambahan mungkin tidak signifikan jika dibandingkan dengan biaya konversi aplikasi untuk dijalankan dalam DBMS baru. Biaya ini juga meliputi biaya pelatihan *staff* untuk menggunakan sistem baru dan mungkin biaya kerja *staff* tertentu untuk membantu proses konversi dan menjalankan sistem baru.

5. *Performance*

   Pada umumnya, sistem berbasis *file* ditulis untuk aplikasi spesifik, sehingga kinerja sistem yang dicapai sangat baik. Sedangkan DBMS digunakan untuk melayani banyak aplikasi

sehingga beberapa aplikasi mungkin menjadi tidak secepat sebelumnya.

6. *Higher impact of a failure*

   Sentralisasi sumber daya meningkatkan keamanan sistem. Karena semua *user* dan aplikasi bergantung pada ketersediaan DBMS, maka kegagalan komponen tertentu dapat menyebabkan terhentinya operasi.

2.1.5 *Structured Query Language*

Menurut Connolly dan Begg (2005:113), *Structured Query Language* (*SQL*) merupakan *transform oriented language*, atau bahasa yang dirancang dengan penggunaan relasi untuk mengubah *input* menjadi *output* yang dibutuhkan.

Ada dua komponen utama dalam SQL yaitu:

1. *Data Definition Language*

   Data Definition Language (DDL) adalah bahasa yang mengizinkan user untuk membuat dan menghapus objek basis data, misalnya *schema*, *table*, *view*, *index*, dan *domain*. Perintah utama yang ada di dalam SQL untuk *Data Definition Language* (DDL) adalah sebagai berikut:

   a. *CREATE*

      Perintah *CREATE* digunakan untuk membuat objek dalam basis data. Pada umumnya, yang memiliki wewenang untuk menggunakan perintah ini adalah *DBA*. Seorang *DBA* memiliki wewenang untuk *CREATE schema*, *CREATE domain*, *CREATE table*, dan *CREATE view*.

   b. *ALTER*

      Perintah *ALTER* digunakan untuk mengubah objek dalam basis data. Perubahan yang dapat dilakukan oleh perintah ini seperti menambah dan menghapus kolom dalam tabel, membuat dan menghapus batasan pada tabel, serta menetapkan dan menghapus standar dari suatu kolom.

c. *DROP*

Perintah *DROP* digunakan untuk menghapus objek dalam basis data. Perintah *DROP* dapat dibagi menjadi dua tipe, yaitu *RESTRICT* dan *CASCADE*. Apabila *user* menggunakan tipe *RESTRICT,* maka perintah *DROP* akan ditolak jika terdapat objek lain yang bergantung pada objek tersebut. Sedangkan untuk tipe *CASCADE,* operasi *DROP* akan dilakukan sesuai dengan perintah yang diinginkan.

2. *Data Manipulation Language*

Data Manipulation Language (*DML*) adalah bahasa yang mengizinkan user untuk memasukkan, mengubah, menghapus, dan mengambil data dari basis data. Perintah utama yang ada di dalam SQL untuk *Data Manipulation Language* (DML) adalah sebagai berikut:

a. *SELECT*

Perintah *SELECT* digunakan untuk mengambil dan menampilkan data yang berasal dari satu atau lebih tabel yang ada dalam basis data. Dalam penggunaannya, perintah *SELECT* harus diikuti dengan *FROM*.

b. *INSERT*

Perintah *INSERT* digunakan untuk memasukkan data baru ke dalam basis data. Sebelum perintah *INSERT* digunakan, *user* harus membuat tabel terlebih dahulu dengan menggunakan perintah *CREATE* pada *Data Definition Language* (DDL).

c. *UPDATE*

Perintah *UPDATE* digunakan untuk memperbarui data dalam basis data. Dalam penggunaannya, perintah *UPDATE* harus diikuti *SET* untuk menjelaskan dan menspesifikasikan nama kolom yang ingin diperbarui atau di-*update*.

d. *DELETE*

Perintah *DELETE* digunakan untuk menghapus data dalam basis data. Dalam penggunaannya, perintah *DELETE* harus

diikuti *FROM* untuk menentukan nama tabel yang datanya akan dihapus.

2.1.6 *Entity-Relationship Modeling*

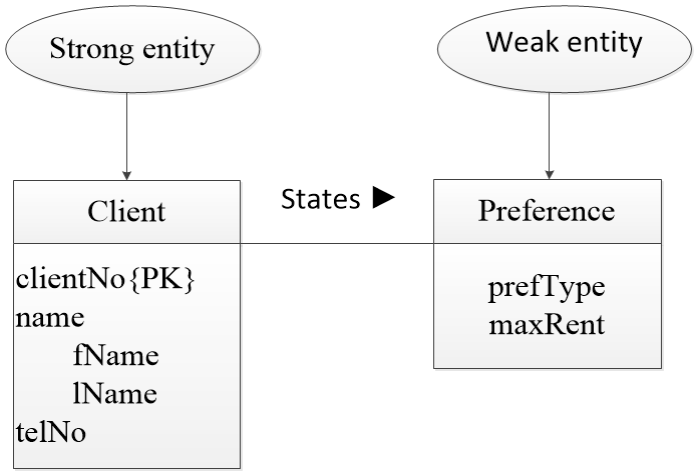
Menurut Connolly dan Begg (2005:342), *Entity-Relationship Modeling* (*ER Modeling*) adalah suatu pendekatan *top-down* untuk merancang basis data yang dimulai dengan mengidentifikasi data-data penting yang disebut entitas dan relasi (*relationship*) antar data yang direpresentasikan dalam sebuah model.

1. Tipe Entitas (*Entity Types*)

   Menurut Connolly dan Begg (2005:343), tipe entitas adalah sekelompok objek dengan properti yang sama, yang diidentifikasi oleh perusahaan, dan memiliki keberadaan yang independen. Keberadaan tipe entitas dapat berupa keberadaan fisik (*physical existence*) atau abstrak (*conceptual existence*).

   Tipe entitas dapat diklasifikasi menjadi dua jenis (Connolly dan Begg, 2005:354) yaitu:

   a. Tipe entitas kuat (*strong entity type*), yaitu tipe entitas yang keberadaannya tidak bergantung pada tipe entitas lainnya. Tipe entitas kuat disebut juga sebagai *parent*, *owner*, ataupun *dominant*.
   b. Tipe entitas lemah (*weak entity type*), yaitu tipe entitas yang keberadaannya bergantung pada tipe entitas lainnya. Tipe entitas lemah disebut juga sebagai *child*, *dependent*, ataupun *subordinate*.



Gambar 2.2 Contoh Tipe Entitas

2. Tipe Relasi (*Relationship Types*)

   Menurut Connolly dan Begg (2005:346), tipe relasi adalah sekumpulan gabungan dari satu atau lebih tipe entitas.

   Relationship occurence adalah sekumpulan asosiasi yang dapat diidentifikasikan secara unik, yang meliputi satu kejadian dari setiap tipe entitas yang berpartisipasi.

   Degree of relationship type (derajat relasi) adalah jumlah tipe entitas yang berpartisipasi dalam sebuah relasi. Tipe entitas yang berpartisipasi dalam sebuah relasi disebut sebagai *participant* dalam relasi tersebut.

   Recursive relationship adalah sebuah relasi dimana tipe entitas yang sama dapat berpartisipasi lebih dari satu kali dengan peran yang berbeda.

3. Atribut (*Attributes*)

   Atribut adalah properti dari sebuah entitas atau sebuah relasi. Atribut dapat diklasifikasikan menjadi:

   a. *Simple* and *Composite Attributes*

      Simple attribute adalah suatu atribut yang terdiri dari satu komponen dengan keberadaan yang independen.

Composite attribute adalah suatu atribut yang terdiri dari banyak komponen, setiap komponen memiliki keberadaan yang independen.

b. *Single-Valued* and *Multi-Valued Attributes*

Single-valued attribute adalah suatu atribut yang memiliki *single value* untuk setap tipe entitas.

Multi-valued attribute adalah suatu atribut yang memiliki *multiple value* untuk setiap tipe entitas.

c. *Derived Attributes*

Derived attribute adalah suatu atribut yang mewakili suatu nilai yang diturunkan dari nilai atribut yang berhubungan, dan tidak harus berada di dalam satu tipe entitas.

4. *Keys*

a. *Candidate Key*

Menurut Indrajani (2011:112), *candidate key* merupakan jumlah minimal atribut-atribut yang dapat mengidentifikasikan setiap kejadian atau *record* secara unik.

b. *Primary Key*

Primary key merupakan *candidate key* yang dipilih untuk mengidentifikasikan setiap kejadian atau *record* suatu entitas secara unik.

c. *Composite Key*

Composite key merupakan *candidate key* yang terdiri atas dua atau lebih atribut.

d. *Foreign Key*

Menurut Indrajani (2011:42), *foreign key* merupakan atribut atau himpunan atribut dalam relasi yang dibandingkan dengan *candidate key* pada beberapa relasi.

e. *Alternate Key*

Alternate key merupakan *candidate key* yang tidak terpilih sebagai *primary key*.

5. *Structural Constraints*

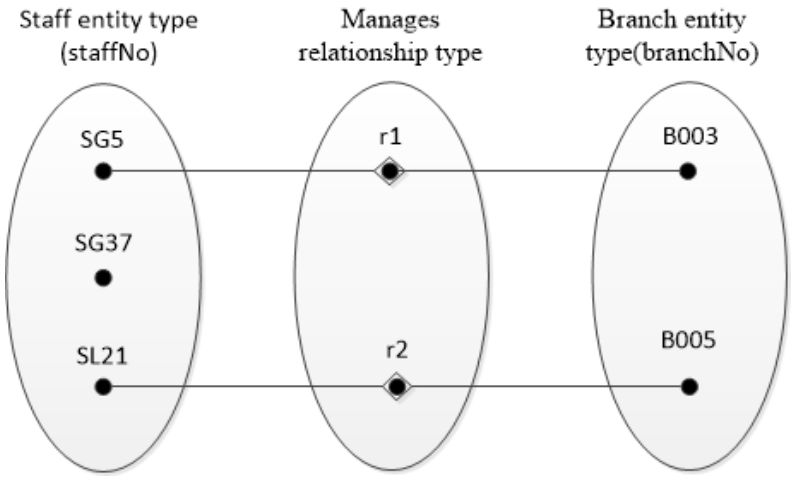
Ada dua komponen dalam *structural constraints*, yaitu *multiplicity* serta *cardinality* dan *participation constraints*.

a. *Multiplicity*

Menurut Connolly dan Begg (2005:356), *Multiplicity* adalah jumlah kejadian yang mungkin terjadi pada sebuah tipe entitas yang berhubungan dengan kejadian dari tipe entitas lainnya melalui suatu relasi. Derajat relasi yang biasa digunakan adalah *binary relationship*. Ada tiga tipe *binary relationship* yang biasa digunakan yaitu:

1) Derajat hubungan *one-to-one* (1:1)

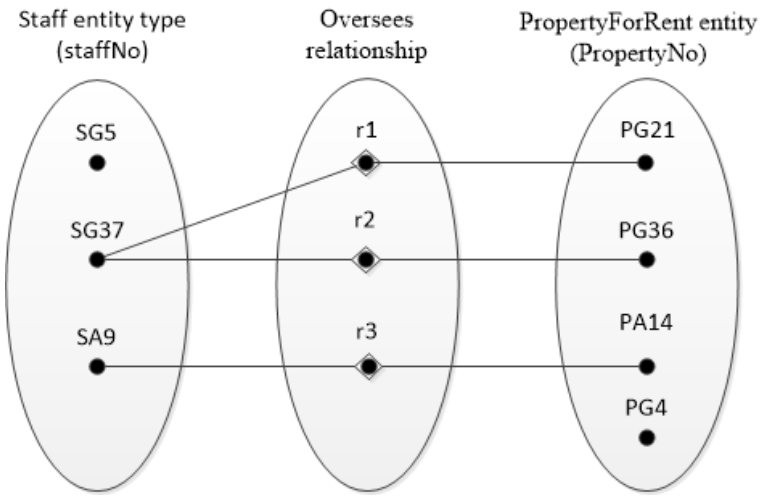
Derajat hubungan 1:1 terjadi bila tiap anggota suatu tipe entitas hanya boleh berpasangan dengan satu anggota dari tipe entitas yang lain. Begitu pula sebaliknya, anggota dari tipe entitas yang lain hanya boleh berpasangan dengan satu anggota dari tipe entitas tersebut.



Gambar 2.3 *Multiplicity* dengan Tipe Relasi (1:1)

2) Derajat hubungan *one-to-many* (1:*)

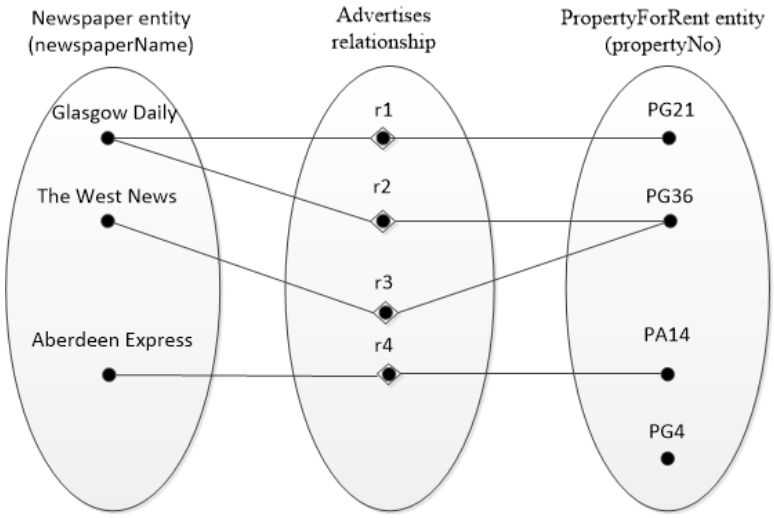
Derajat hubungan 1:* terjadi bila tiap anggota suatu tipe entitas boleh memiliki pasangan lebih dari satu anggota dari tipe entitas lain. Dan sebaliknya, anggota dari tipe entitas yang lain hanya boleh berpasangan dengan satu anggota dari tipe entitas tersebut.



Gambar 2.4 *Multiplicity* dengan Tipe Relasi (1:*)

3) Derajat hubungan many-to-many (*:*)

Derajat hubungan *:* terjadi apabila tiap anggota suatu tipe entitas boleh memiliki pasangan lebih dari satu anggota dari tipe entitas lain dan anggota dari tipe entitas yang lain juga boleh memiliki pasangan lebih dari satu anggota dari tipe entitas tersebut.



Gambar 2.5 *Multiplicity* dengan Tipe Relasi (*:*)

b. *Cardinality* dan *Participation Constraints*

Menurut Connolly dan Begg (2005:363), pada dasarnya *multiplicity* terdiri atas dua batasan yang berbeda yaitu:

1) *Cardinality*, mendeskripsikan nilai maksimum dari relasi yang mungkin terjadi untuk tipe entitas yang ikut serta pada relasi tersebut.
2) *Participation*, menentukan apakah semua atau hanya beberapa tipe entitas yang ikut berpartisipasi dalam sebuah relasi. *Participation constraint* dibagi menjadi dua jenis, yaitu *mandatory participation* dan *optional participation*. *Mandatory participation* melibatkan semua tipe entitas pada relasi tertentu. Sedangkan *optional participation* melibatkan beberapa tipe entitas pada relasi tertentu.

2.1.7 Normalisasi

Menurut Connolly dan Begg (2005:388), normalisasi adalah suatu teknik yang digunakan untuk menghasilkan seperangkat relasi yang memiliki properti yang sesuai dengan kebutuhan perusahaan. Tujuan utama dari normalisasi adalah untuk mengurangi terjadinya redundansi data dan mengurangi masalah yang terjadi pada suatu relasi, yang dikenal dengan sebutan *anomaly*.

Anomaly adalah suatu masalah yang timbul, seperti data ganda, data hilang, tempat penyimpanan memori yang boros, dan data yang tidak konsisten akibat dari proses penghapusan data, pembaruan data, pemasukan data, dan pergantian data.

Berikut ini adalah beberapa tingkatan normalisasi:

1. *Unnormalized Form* (UNF)

Unnormalized Form (UNF) adalah suatu tabel yang terdiri dari satu atau lebih kelompok berulang (*repeating group*). Kelompok berulang adalah sebuah atribut atau himpunan atribut di dalam tabel yang memiliki lebih dari satu nilai untuk keberadaan tunggal.

2. *First Normal Form* (1NF)

   Suatu relasi dikatakan 1NF jika titik temu tiap baris dan kolom pada relasi tersebut mengandung satu dan hanya satu nilai. Atau dapat dikatakan sebuah relasi akan berada dalam bentuk 1NF jika kelompok perulangan telah hilang. Proses-proses yang dilakukan untuk mengubah UNF menjadi 1NF adalah sebagai berikut:

   a. Pilih satu atau sekumpulan atribut sebagai *key* untuk unnormalized table;
   b. Identifikasi kelompok perulangan yang terdapat pada *unnormalized table*;
   c. Hilangkan kelompok perulangan yang ada, dengan cara memindahkan kelompok perulangan tersebut ke dalam relasi baru yang terpisah dan meninggalkan salinan dari *primary key*.

3. *Second Normal Form* (2NF)

   Relasi dikatakan 2NF jika relasi tersebut telah berada pada 1NF dan setiap atribut bukan *primary key* memiliki ketergantungan penuh (*fully functionally dependent*) terhadap *primary key*. Ketergantungan penuh terjadi jika A dan B merupakan atribut dari suatu relasi, dan B dikatakan bergantung terhadap A (A→B), namun bukan merupakan subset dari A. Proses-proses yang dilakukan untuk mengubah 1NF menjadi 2NF adalah sebagai berikut:

   a. Identifikasi *primary key* untuk relasi 1NF;
   b. Identifikasi ketergantungan fungsional dalam relasi;
   c. Jika terdapat ketergantungan parsial terhadap *primary key*, maka hilangkan dengan menempatkannya dalam relasi yang baru dan meninggalkan salinan dari *primary key*.

4. *Third Normal Form* (3NF)

   Suatu relasi dikatakan 3NF jika relasi tersebut berada dalam bentuk 1NF dan 2NF, serta tidak ada atribut yang bukan *primary key* bergantung secara transitif (*transitively dependent*) terhadap *primary key*. Ketergantungan transitif terjadi dalam kondisi dimana A,B, dan C merupakan atribut dari suatu relasi, serta A→B dan B→C. Maka dapat dikatakan C bergantung secara transitif terhadap

A melalui B. Proses-proses yang dilakukan untuk mengubah 2NF menjadi 3NF adalah sebagai berikut:

a. Identifikasi *primary* key untuk relasi 2NF;
b. Identifikasi ketergantungan fungsional dalam relasi;
c. Jika terdapat ketergantungan transitif terhadap *primary key*, maka hilangkan dnegan menempatkannya dalam relasi yang baru dan meninggalkan salinan dari *primary key*.

2.1.8 Metodologi Perancangan Basis Data

Menurut Connolly dan Begg (2005:291), perancangan basis data merupakan suatu proses menciptakan rancangan basis data yang akan mendukung operasi dan tujuan perusahaan.

Terdapat tiga tahapan utama dalam perancangan basis data, yaitu perancangan basis data konseptual (*conceptual database design*), perancangan basis data logikal (*logical database design*), dan perancangan basis data fisikal (*physical dataabse design*).

1. Perancangan Basis Data Konseptual (*Conceptual Database Design*)

   Perancangan basis data konseptual adalah proses membangun sebuah model dari data yang digunakan di perusahaan dan terlepas dari semua pertimbangan fisikal. Langkah-langkah yang dilakukan dalam perancangan basis data konseptual (Connolly dan Begg, 2005:442) adalah sebagai berikut:

   a. Mengidentifikasi tipe entitas

      Menentukan dan mendefinisikan objek-objek utama yang menjadi perhatian utama dari user. Identifikasi tipe entitas ini terdiri dari nama entitas, deskripsi dari entitas, alias, dan kejadian. Salah satu metode dalam mengidentifikasi entitas adalah dengan melihat rincian kebutuhan *user*.

   b. Mengidentifikasi tipe relasi

      Mengidentifikasi semua relasi yang ada antara masing-masing entitas. Identifikasi relasi ini terdiri dari nama entitas, multiplicity, dan nama relasi.

   c. Mengidentifikasi dan mengasosiasikan atribut dengan tipe entitas atau relasi

Menghubungkan atribut dengan entitas atau relasi yang sesuai. Identifikasi atribut ini terdiri dari nama entitas, atribut, deskripsi dari atribut, tipe data, *nulls*, dan *multi-valued*.

d. Menentukan domain atribut

Domain yang ditetapkan untuk setiap atribut meliputi kumpulan nilai yang diperbolehkan, ukuran, dan format dari setiap atribut tersebut.

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

Mengidentifikasi satu atau lebih atribut untuk menjadi *candidate key*. Kemudian jika terdapat lebih dari satu *candidate key*, maka pilihlah salah satu menjadi *primary key* dan yang lainnya menjadi *alternate key*.

f. Mempertimbangkan penggunaan *enhanced modeling concept* (*optional*)

Mempertimbangkan penggunaan *enhanced modeling concept*, seperti spesialisasi, generalisasi, agregasi, dan komposisi.

g. Memeriksa redundansi pada model data konseptual

Untuk menguji model data dari redundansi, yang harus dilakukan adalah sebagai berikut:

1) Memeriksa ulang relasi *one-to-one* (1:1);
2) Menghilangkan relasi yang berulang;
3) Mempertimbangkan dimensi waktu.

h. Memvalidasi model data konseptual dengan transaksi *user*

Memastikan bahwa model data konseptual dapat mendukung transaksi yang diperlukan. Dilakukan dengan mendeskripsikan transaksi dan menggunakan jalur transaksi.

i. Melakukan tinjauan model data konseptual dengan *user*

Memastikan model data konseptual sudah sesuai dengan kebutuhan *user* dan dapat menjadi representasi nyata dari kebutuhan perusahaan.

2. Perancangan Basis Data Logikal (*Logical Database Design*)

Perancangan basis data logikal adalah proses membangun sebuah model data yang digunakan di perusahaan berdasarkan pada

sebuah model data yang spesifik, tetapi terlepas dari DBMS tertentu dan pertimbangan fisikal lainnya.

Langkah-langkah yang dilakukan dalam perancangan basis data logikal (Connolly dan Begg, 2005:462) adalah sebagai berikut:

a. Menurunkan relasi untuk model data logikal

   Membuat relasi dalam model data logikal untuk menggambarkan entitas, relasi, dan atribut yang telah diidentifikasi.

b. Memvalidasi relasi dengan normalisasi

   Normalisasi adalah suatu teknik yang digunakan untuk menghasilkan seperangkat relasi yang memiliki properti sesuai dengan kebutuhan perusahaan. Tujuan utama dari normalisasi adalah untuk mengurangi terjadinya redundansi data dan mengurangi masalah yang terjadi pada suatu relasi yang dikenal dengan sebutan *anomaly*.

c. Memvalidasi relasi dengan transaksi *user*

   Memastikan model data logikal yang telah dibuat dapat mendukung transaksi yang dibutuhkan, seperti yang ada dalam rincian kebutuhan *user*.

d. Menentukan *integrity constraints*

   Integrity constraints yangperlu dipertimbangkan adalah:

   1) Data tidak boleh berupa *null*;
   2) Domain constraint dari setiap atribut;
   3) *Multiplicity*;
   4) *Primary key* dari entitas tidak boleh berupa *null*;
   5) Jika *foreign key* berisi suatu nilai, maka nilai tersebut harus menunjuk kepada *tupl*e yang ada pada relasi *parent*;
   6) *General constraints*.

e. Melakukan tinjauan model data logikal dengan *user*

   User diminta untuk memeriksa kembali model data logikal untuk memastikan bahwa model data logikal tersebut sudah sesuai untuk menjadi representasi nyata dari kebutuhan perusahaan.

f. Menggabungkan model data logikal menjadi model global (*optiona*l)

Meskipun setiap model data logikal haruslah benar, dapat dipahami, dan tidak ambigu, setiap model tersebut hanya mewakili sebagian dari basis data yang lengkap. Oleh karena itu, harus dilakukan penggabungan model data logikal menjadi model global yang digunakan untuk memecahkan konflik antar *view* dan *overlap* yang ada.

g. Memeriksa perkembangan di masa mendatang

Menduga apakah terjadi perubahan yang signifikan di masa depan dan untuk menaksir apakah model data logikal dapat menyesuaikan dengan perubahan tersebut.

3. Perancangan Basis Data Fisikal (*Physical Database Design*)

Perancangan basis data fisikal adalah proses untuk menghasilkan suatu deskripsi implementasi dari suatu basis data pada penyimpanan sekunder, juga menjelaskan relasi dasar, pengaturan *file* dan indeks yang digunakan untuk mencapai akses data yang efisien, *integrity constraint*, serta mekanisme keamanan.

Langkah-langkah yang dilakukan dalam perancangan basis data fisikal (Connolly dan Begg, 2005:497) adalah sebagai berikut:

a. Menerjemahkan model data logikal untuk target DBMS

Menghasilkan skema basis data yang saling berelasi dari model data logikal, yang dapat diimplementasikan pada target DBMS. Ada tiga aktivitas dalam tahapan ini yaitu:

1) Merancang relasi dasar;
2) Merancang representasi dari *derived data*;
3) Merancang batasan umum (*general constraints*).

b. Menentukan organisasi *file* dan indeks

Menentukan organisasi *file* yang paling baik untuk menyimpan relasi dasar dan indeks yang diperlukan untuk mencapai akses data yang efisien, serta bagaimana relasi dan tuple disimpan pada penyimpanan sekunder. Berikut ini adalah empat tahapan dalam merancang organisasi *file* dan indeks:

1) Menganalisis transaksi;

2) Menentukan organisasi *file*;
3) Menentukan indeks;
4) Memperkirakan kebutuhan ruang *disk*.

c. Merancang *user view*

Merancang *user view* yaitu merancang pandangan user yang telah diidentifikasi selama tahap analisis dan pengumpulan kebutuhan dalam siklus hidup pengembangan sistem basis data.

d. Merancang mekanisme keamanan

Merancang mekanisme keamanan untuk basis data yang ditetapkan oleh user selama proses pengumpulan kebutuhan perusahaan dan tingkatan dalam siklus hidup basis data.

e. Mempertimbangkan petunjuk *controlled redundancy*

Yaitu untuk menentukan apakah pengenalan redundansi dengan cara yang terkontrol dengan aturan normalisasi akan meningkatkan kinerja sistem.

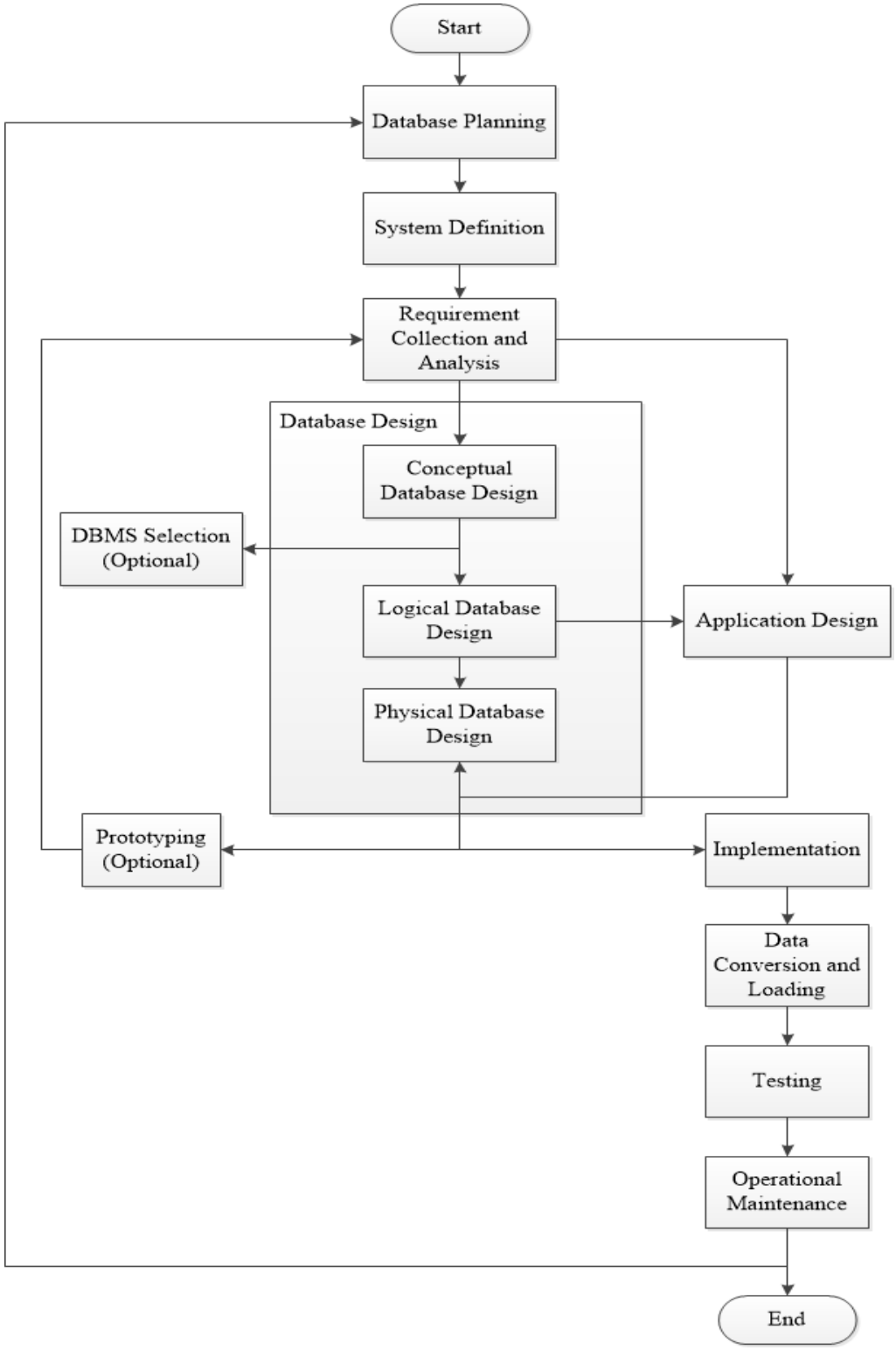
f. Memonitor dan memperbaiki sistem operasional

Memonitor dan meningkatkan performa dari sistem dengan memperbaikin rancangan yang tidak sesuai dnegan kebutuhan.

2.1.9 Siklus Hidup Pengembangan Sistem Basis Data

Menurut Connolly dan Begg (2005:283), siklus hidup pengembangan sistem basis data (*Database System Development Lifecycle*) merupakan komponen yang terpenting dalam sistem basis data.

Pada aplikasi dataabse berskala kecil yang jumlah *user*-nya sedikit, siklus hidup pengembangan aplikasi basis datanya tidak terlalu kompleks. Sebaliknya, apabila aplikasi basis data dibuat dalam skala yang lebih besar atau kompleks dengan jumlah *user* antara puluhan sampai ribuan, serta menggunakan ratusan *query* dan program aplikasi, siklus hidup pengembangan aplikasi basis datanya pun akan menjadi lebih kompleks.



Gambar 2.6 *Database System Development Lifecycle*

Berikut ini adalah penjelasan dari masing-masing tahapan siklus hidup pengembangan sistem basis data:

1. Perencanaan Basis Data (*Database Planning*)

   Perencanaan basis data adalah merencanakan bagaimana langkah-langkah dari siklus hidup pengembangan sistem basis data dapat diterapkan dalam sistem basis data secara efektif dan efisien. Ada tiga isu pokok dalam merumuskan suatu strategi sistem informasi yaitu:

   a. Mengidentifikasi rencana dan tujuan perusahaan dengan menentukan sistem informasi yang diperlukan;
   b. Mengevaluasi sistem informasi yang ada sekarang untuk menentukan kekuatan dan kelemahan yang ada;
   c. Memperkirakan peluang IT yang mungkin menghasilkan keuntungan kompetitif.

2. Definisi Sistem (*System Definition*)

   Definisi sistem adalah menjelaskan cakupan dan batasan-batasan dari aplikasi sistem basis data dan pandangan user yang utama. Pandangan *user* sangat diperlukan untuk mengidentifikasikan informasi-informasi yang dibutuhkan oleh *user*. Pandangan *user* akan mendefinisikan apa saja yang harus ada dalam suatu sistem basis data.

3. Analisis dan Pengumpulan Kebutuhan (*Requirement Collection and Analysis*)

   Analisis dan pengumpulan kebutuhan adalah proses pengumpulan dan analisis informasi tentang bagian organisasi yang akan didukung oleh sistem basis data, dan menggunakan informasi tersebut untuk mengidentifikasikan kebutuhan *user* terhadap sistem yang baru. Teknik pencarian fakta (*fact-finding technique*) adalah teknik yang digunakan untuk mendapatkan informasi yang dibutuhkan.

   Ada lima teknik pencarian fakta yang biasanya digunakan (Connolly dan Begg, 2005:317) yaitu:

a. Pemeriksaan dokumen (*examining documentation*)

Dokumentasi dapat digunakan untuk membantu menyediakan informasi perusahaan berkaitan dengan masalah yang sedang dihadapi. Dengan mempelajari dokumen, formulir, laporan dan *file* yang berhubungan dengan sistem yang ada, pemahaman tentang sistem dapat dengan mudah diperoleh.

b. Wawancara

Dengan wawancara, maka informasi akan didapatkan langsung dari individu-individu yang bersangkutan. Tujuan dilakukan wawancara adalah untuk menemukan fakta, memverifikasi fakta, mengklarifikasi fakta, menampilkan antusiasme, melibatkan *end-user*, mengidentifikasikan kebutuhan, dan mendapatkan opini dari orang yang berkepentingan.

c. Observasi

Observasi adalah salah satu teknik yang paling efektif untuk memahami sebuah sistem. Dengan teknik observasi, informasi dapat ditemukan dengan berpartisipasi dan mengawasi sistem yang ada secara langsung.

d. Penelitian

Penelitian adalah suatu teknik yang dilakukan dengan meneliti aplikasi dan masalah yang ada. Sebagai tambahan, dapat digunakan jurnal-jurnal, buku-buku, dan internet sebagai sumber informasi. Dari hasil riset, akan didapatkan informasi tentang bagaimana masalah serupa dapat dipecahkan.

e. Kuesioner

Kuesioner adalah salah satu cara pengumpulan fakta dari sejumlah orang dengan menggunakan suatu dokumen. Teknik ini sangat efisien ketika ingin mengumpulkan informasi dari banyak orang.

4. Perancangan Basis Data (*Database Design*)

Perancangan basis data merupakan proses menciptakan rancangan untuk basis data yang akan mendukung operasi dan

tujuan perusahaan. Proses perancangan basis data dibagi menjadi tiga tahapan utama yaitu:

a. Perancangan basis data konseptual

   Adalah proses membangun sebuah model dari data yang digunakan di perusahaan dan terlepas dari semua pertimbangan fisikal.

b. Perancangan basis data logikal

   Adalah proses membangun sebuah model data yang digunakan di perusahaan berdasarkan pada sebuah model data yang spesifik, tetapi terlepas dari DBMS tertentu dan pertimbangan fisikal lainnya.

c. Perancangan basis data fisikal

   Adalah proses untuk menghasilkan suatu deskripsi implementasi dari suatu basis data pada penyimpanan sekunder, juga menjelaskan relasi dasar, pengaturan *file* dan indeks yang digunakan untk mencapai akses data yang efisien, *integrity constraints*, serta mekanisme keamanan.

5. Seleksi DBMS (DBMS *Selection*)

   Seleksi DBMS adalah pemilihan DBMS yang sesuai untuk mendukung sistem basis data. Menurut Connolly dan Begg (2005:296), tahapan dalam DBMS *selection* yaitu:

   a. Mempelajari DBMS yang ada dan yang sesuai dengan kriteria kebutuhan *user*;
   b. Membatasi pilihan DBMS menjadi dua atau tiga pilihan saja;
   c. Mengevaluasi produk DBMS;
   d. Merekomendasikan produk DBMS yang trebaik dan membuat dokumentasi dari tahapan pemilihan tersebut.

6. Perancangan Aplikasi (*Application Design*)

   Perancnagan aplikasi adalah merancang user interface dan program aplikasi yang akan memproses basis data. Harus dipastikan bahwa semua kebutuhan user tersedia di dalam rancangan aplikasi basis data.

7. *Prototyping* (*optional*)

   Prototyping adalah proses membangun sebuah model kerja dari aplikasi basis data. Tujuan utama *prototyping* adalah memungkinkan *user* menggunakan *prorotype* untuk mengidentifikasikan fitur-fitur yang bekerja dengan baik pada sistem, atau fitur-fitur yang masih kurang, dan memberikan masukan fitur-fitur baru ke dalam aplikasi basis data.

8. Implementasi (*Implementation*)

   Implementasi merupakan realisasi fisik dari basis data dan rancangan aplikasi. Implementasi basis data dapat dicapai dengan menggunakan:

   a. *Data Definition Language* (DDL), digunakan untuk membuat skema basis data atau *file* basis data yang masih kosong dan juga untuk mengimplementasikan pandangan *user* yang diinginkan;
   b. *Third Generation Language* atau *Fourth Generation Language* (3GL atau 4GL), digunakan untuk membuat program aplikasi, termasuk transaksi basis data yang disertakan dengan menggunakan *Data Manipulation Language* (DML).

9. Konversi Data (*Data Conversion and Loading*)

   Konversi data merupakan pemindahan data yang ada dan mengonversikannya dengan beberapa aplikasi, agar dapat dijalankan pada basis data yang baru. Tahapan ini dibutuhkan hanya ketika sistem basis data yang baru menggantikan sistem yang lama. Pada saat ini, DBMS sangat bermanfaat untuk memanggil *file* yang sudah ada ke dalam basis data yang baru.

10. Pengujian (Testing)

    Pengujian adalah suatu proses eksekusi program aplikasi dengan tujuan untuk mencari kesalahan, dengan menggunakan strategi tes yang direncanakan dan data yang sesungguhnya. Apabila ditemukan adanya kesalahan, maka akan dilakukan perbaikan terhadap basis data dan aplikasi yang telah dibuat sebelumnya.

11. Operasional Pemeliharaan (*Operational Maintenance*)

Operasional pemeliharaan merupakan proses mengawasi dan memelihara sistem setelah dilakukannya instalasi. Aktivitas yang terdapat pada operational maintenance yaitu:

a. Mengawasi kinerja sistem;
b. Memelihara dan memperbarui aplikasi basis data (jika diperlukan).

2.2 Teori yang Terkait Tema Penelitian (Tematik)

2.2.1 *Internet*

Menurut Williams dan Sawyer (2007:63), *internet* adalah jaringan yang sangat besar dari jaringan, menghubungkan jutaan komputer melalui protokol, perangkat keras dan jalur komunikasi. Internet merupakan infrastruktur yang tidak hanya mendukung *web*, tetapi juga sistem komunikasi seperti *e-mail*, *instant messaging*, *newsgroup*, dan aktivitas lain.

2.2.2 *Web Server*

"*Web Server* adalah sebuah komputer yang terdiri dari perangkat keras dan perangkat lunak. Secara bentuk fisik dan cara kerjanya, perangkat keras *web server* tidak berbeda dengan komputer rumah atau PC, yang membedakan adalah kapasitas dan kapabilitasnya. Perbedaan tersebut dikarenakan *web server* bekerja sebagai penyedia layanan yang dapat diakses oleh banyak pengguna, sehingga dibutuhkan kapasitas dan kapabilitas yang besar dibandingkan PC. Dukungan perangkat lunak sangat dibutuhkan agar *web server* dapat berjalan secara optimal. Setiap perangkat lunak *web server* memiliki karakteristik dan teknologi yang digunakan untuk mengatur kerja sistemnya." (Sibero, 2013:11)

2.2.3 Web Browser

"*Web browser* adalah aplikasi perangkat lunak yang digunakan untuk mengambil dan menyajikan sumber informasi *web*." (Sibero, 2013:12)

2.2.4 PHP

Menurut Welling dan Thomson (2009:2), PHP adalah *server-side scripting language* yang didesain secara spesifik untuk *web*. Kode PHP

dapat disatukan dalam halaman yang berisi *tag-tag* HTML yang akan dieksekusi setiap kali halaman itu dikunjungi. Kode PHP tersebut akan diinterpretasikan di *web server* dan menghasilkan HTML atau output lain yang dapat dilihat oleh pengunjung.

PHP adalah produk *Open Source*. Pengunjung memiliki akses ke *source code*. Karena bersifat *open source*, siapapun dapat menggunakan, mengubah, dan menambahkan fungsi-fungsi baru secara bebas.

PHP awalnya berdiri untuk *Personal Home Page*, tetapi kemudian diubah sesuai dengan konvensi penamaan *GNU* rekursif (*GNU = Gnu's Not Unix*) dan saat berdiri untuk PHP *Hypertext Preprocessor*.

Menurut Kadir (2013:32-33), PHP merupakan skrip yang digunakan untuk menangani pemrosesan di sisi *server*, misalnya untuk membaca basis data yang bersifat rahasia. Skrip PHP sangat bermanfaat untuk membuat aplikasi yang berhubungan dengan basis data. Dengan demikian, penggunaan PHP memungkinkan untuk membuat halaman *web* yang dinamis, karena melibatkan data yang tersimpan dalam basis data.

2.2.5 *MySQL*

Menurut Welling dan Thomson (2009:3), *MySQL* adalah *relational database management system* (RDBMS) yang sangat cepat dan kuat. Basis data memungkinkan untuk menyimpan, mencari, mengurutkan, dan mengambil data secara efisien. *MySQL server* mengontrol akses pada data untuk memastikan bahwa beberapa user dapat bekerja secara bersamaan, memberikan akses cepat, dan memastikan bahwa hanya user yang berwenang yang dapat memperoleh akses. *MySQL* menggunakan SQL (*Structured Query Language*), standar bahasa *query* basis data di seluruh dunia.

2.2.6 *JavaScript*

Menurut Negrino dan Smith (2004:2), *JavaScript* adalah bahasa pemrograman yang dapat digunakan untuk menambahkan interaktivitas ke dalam suatu halaman *web*.

2.2.7 *CodeIgniter*

Menurut Upton (2007:7-8), *CodeIgniter* membantu secara lebih baik dalam penulisan kode PHP. *CodeIgniter* akan mengurangi jumlah kode yang diketikkan. Selain itu, *script* akan menjadi lebih mudah untuk dibaca dan diperbarui. *CodeIgniter* tidak dapat digunakan jika:

1. Tidak memiliki pengetahuan yang cukup mengenai PHP dan HTML.
2. Akan menulis dasar *Content Management System* (CMS) dengan cepat dan sederhana.
3. Hanya akan menulis situs *web* sederhana dengan beberapa fitur standar.

2.2.5 *Cascading Style Sheets*

Menurut Pouncey dan York (2011:3) *Cascading Style Sheets* (CSS) adalah bahasa yang didesain untuk menggambarkan tampilan dokumen yang ditulis dalam bahasa *markup* seperti HTML. Dengan CSS dapat mengontrol warna teks, gaya huruf, jarak antara paragraf, warna latar belakang yang digunakan, dan berbagai efek visual lainnya.

Menurut Kadir (2013:8), CSS banyak dilibatkan dalam dokumen web. Kegunaannya adalah untuk memformat dokumen. Sebagai contoh, warna teks atau bahkan warna latar belakang bisa diatur melalui CSS. Namun, kegunaan CSS jauh lebih ampuh daripada sekadar mengatur warna. Membuat bingkai atau bahkan mengatur agar elemen di dokumen terletak pada posisi tertentu dapat dilakukan dengan CSS.

2.2.6 *jQuery*

Menurut Lindley (2010:1), *jQuery* merupakan *open source JavaScript library* yang menyederhanakan interaksi antara dokumen HTML, atau lebih tepatnya *Document Object Model* (DOM), dan *JavaScript.*

2.2.7 *Ajax*

Menurut Babin (2007:7), *Ajax* bukan merupakan suatu teknologi baru. *Ajax* hanyalah sebuah istilah untuk menggambarkan proses

menggunakan *JavaScript* untuk mengambil informasi dari suatu *web server* secara dinamis (asinkron).

2.2.8 *Easy UI jQuery*

Easy UI jQuery adalah sebuah *framework* yang membantu membuat halaman *web* menjadi lebih mudah. *Easy UI* adalah sebuah kumpulan dari *plugin user interface* berbasis *jQuery*. *Easy UI* menyediakan fungsi-fungsi penting untuk membangun aplikasi *JavaScript* yang *modern* dan interaktif. (www.jeasyui.com, 2010-2013)

2.2.9 JsTree

JsTree adalah sebuah *jquery plugin*, yang menyediakan pohon interaktif (*interactive trees*). *JsTree* mudah diperpanjang dan dikonfigurasi, *JsTree* mendukung sumber data HTML & JSON, serta AJAX *loading*. (www.jstree.com, 2013)

2.2.10 *Ace*

Ace merupakan *code editor* yang ditulis menggunakan *JavaScript*. *Ace* telah dibuat sesuai dengan fitur dan kinerja *editor* asli seperti *Sublime*, *Vim*, dan *TextMate* sehingga dapat dengan mudah untuk digunakan pada setiap halaman *web* dan aplikasi *JavaScript*. *Ace* dipertahankan sebagai *editor* utama untuk *Cloud9 IDE* dan merupakan penerus dari proyek *Mozilla Skywriter*. (www.ace.c9.io, 2010)

2.2.11 *Flowchart*

Menurut Indrajani (2011:22), *Flowchart* merupakan penggambaran secara fisik dari langkah-langkah dan urutan prosedur suatu program. Biasanya mempermudah penyelesaian masalah, khususnya yang perlu dipelajari dan dievaluasi lebih lanjut.

2.2.12 *State Transition Diagram*

Menurut Indrajani (2011:17), *State Transition Diagram* (STD) adalah suatu kondisi yang menunjukkan keadaan tertentu, dimana suatu sistem dapat ada dan transisi menghasilkan keadaan tertentu yang baru. Biasanya digunakan dalam sistem yang *real time*.

2.2.13 Interaksi Manusia dan Komputer

Menurut Indrajani (2011:74), interaksi manusia dan komputer adalah disiplin ilmu yang berhubungan dengan perancangan, evaluasi, dan implementasi sistem komputer interaktif untuk digunakan oleh manusia, serta studi fenomena-fenomena besar berhubungan dengannya. Terdapat delapan aturan emas dalam membuat sebuah tampilan *user interface*. Delapan aturan tersebut dapat digunakan dalam sebuah sistem yang interaktif dan membutuhkan sebuah validasi dan pengaturan untuk sebuah tampilan domain yang spesifik.

Tahapan	Penjelasan
1. *Strive for consistency*	Konsistensi dilakukan pada ukuran tindakan, perintah, dan istilah yang digunakan pada *prompt*, menu, serta layar bantuan.
2. *Cater for universal usability*	Memungkinkan *user* untuk menggunakan *shortcut*. Ada kebutuhan dari *user* yang telah ahli untuk meningkatkan kecepatan interaksi sehingga diperlukan singkatan, tombol fungsi perintah tersembunyi, dan fasilitas makro.
3. *Over informative feedback*	Untuk setiap tindakan operator, sebaiknya disertakan suatu sistem umpan balik. Untuk tindakan yang sering dilakukan dan tidak terlalu penting dapat diberikan umpan balik yang sederhana. Tetapi ketika tindakan berupa hal yang penting, maka umpan balik sebaiknya lebih substansial.
4. *Design dialog to yield closure*	Urutan tindakan sebaiknya

	diorganisasi dalam suatu kelompok dengan bagian awal, tengah, dan akhir. Umpan balik yang informatif akan memberikan indikasi bahwa cara yang dilakukan telah benar dan dapat mempersiapkan kelompok tindakan berikutnya.
5. *Prevent errors*	Sedapat mungkin sistem dirancang sehingga user tidak dapat melakukan kesalahan fatal. Jika kesalahan terjadi, sistem dapat mendeteksi kesalahan dengan cepat dan memberikan mekanisme yang sederhana dan mudah dipahami untuk penanggulangan kesalahan.
6. *Permit easy reversal of action*	Hal ini dapat mengurangi kekhawatiran *user* karena user mengetahui kesalahan yang dilakukan dapat dibatalkan, sehingga *user* tidak takut untuk mengeskplorasi pilihan-pilihan lain yang belum biasa digunakan.
7. *Support internal locus of control*	*User* ingin menjadi pengontrol sistem dan sistem akan merespon tindakan yang dilakukan *user*, daripada *user* merasa bahwa sistem mengontrol *user*. Sebaiknya sistem dirancang sedemikian rupa sehingga *user* menjadi inisiator daripada responden.

8. *Reduce short-term memory load*	Keterbatasan ingatan manusia membutuhkan tampilan yang sederhana atau banyak tampilan halaman yang sebaiknya disatukan, serta diberikan cukup waktu pelatihan untuk kode, *mnemonic*, dan urutan tindakan.

2.3 Hasil Penelitian atau Produk Sebelumnya

Penulis melakukan analisis pada produk sejenis sebelumnya dengan melakukan studi pustaka terhadap tiga jurnal, yaitu:

2.3.1 *Experiences with Eclipse IDE in Programming Courses*

Menurut Chen dan Marx (2005:104), *Eclipse for Java* adalah fitur *IDE* profesional yang dilengkapi dengan kecanggihan fitur lainnya. *Eclipse* merupakan *platform-centric IDE* dengan berbagai *tool* pengembangan yang terintegrasi di dalamnya.

Eclipse merupakan sebuah *open source* dan *IDE* universal yang dapat digunakan oleh siapapun.

2.3.2 *Real-Time Collaborative Coding in a Web IDE*

Menurut Goldman, Little, dan Miller (2011:155), *Collaborative Coding* (*Collabode*) adalah sebuah *web* IDE berbasis *Java* yang dirancang untuk mendukung serta mengsinkronisasikan kolaborasi antar *programmer*. *Programmer* menggunakan web browser untuk dapat terhubung ke *server Collaborative Coding* (*Collabode*).

Web IDE kolaboratif berbasis *Java* bertujuan untuk mempelajari tentang bagaimana lingkungan pemrograman dibangun untuk mendukung kolaborasi yang dapat meningkatkan kualitas, baik kualitas kolaborasi maupun perangkat lunak yang dihasilkan.

2.3.3 *Improving IDE Recommendations by Considering Global Implications of Existing Recommendations*

Menurut Muslu, Brun, Holmes, Ernst, dan Notkin (2012:349), *Modern IDE* menawarkan kemudahan untuk mendapatkan informasi

kontekstual bagi para pengembang. Contohnya, mekanisme *auto-complete* untuk melengkapi variabel atau nama *method* pada baris *code* yang sedang dikerjakan. *Eclipse's Quick Fix* yang memberikan rekomendasi cara untuk menyelesaikan error yang terjadi pada saat kompilasi dijalankan. *Modern IDE* bertujuan untuk mempermudah pengembang dalam mendapatkan rekomendasi untuk menyelesaikan kesalahan (*error*) yang terjadi dengan memanfaatkan *global information* yang berjalan cepat dan efektif tanpa mengurangi kualitas hasil akhir.

Penelitian dan permintaan *user* mendorong *IDE* untuk terus berkembang. Peneliti mengidentifikasi masalah pada *IDE* saat ini dengan menawarkan informasi kontekstual untuk menyederhanakan tugas pengembang pada umumnya, agar *IDE* dapat memudahkan pengembang aplikasi di masa mendatang.