BAB 2

LANDASAN TEORI

2.1 Teori-teori *Video Streaming*

2.1.1 *Video Streaming*

Video merupakan suatu media yang sangat penting untuk komunikasi dan hiburan selama puluhan tahun ini. Pertama kali, video diolah dan ditransmisikan dalam bentuk *analog*. Perkembangan di bidang komputer telah membantu terbentuknya video *digital*. Salah satu penerapan video *digital* yang digunakan dalam transmisi data adalah *video streaming*.

Video streaming adalah teknologi pengiriman data, video atau audio dalam bentuk yang telah dikompresi melalui jaringan *internet* yang ditampilkan oleh suatu *player* secara *realtime*. Pengguna memerlukan *player* yang merupakan aplikasi khusus untuk melakukan dekompresi dan mengirimkan data berupa video ke tampilan layar monitor dan data berupa suara ke *speaker*. Sebuah *player* dapat berupa suatu bagian dari *browser* atau sebuah perangkat lunak. Inti dari *streaming* adalah membagi data dan *encoding*, kemudian mengirimkannya melalui jaringan dan pada saat data sampai pada pengguna maka akan dilakukan *decoding* serta pembacaan data. Ciri-ciri aplikasi *streaming* yaitu distribusi audio, video dan multimedia pada jaringan secara *realtime* atau *on demand*,

transfer media data *digital* dari *server* dan diterima oleh pengguna sebagai *realtime stream* simultan sehingga pengguna tidak perlu menunggu keseluruhan data di-*download* karena *server* mengirimkan data yang diperlukan setiap selang waktu tertentu. Hal ini memungkinkan pengguna untuk menjalankan *file content* seketika dengan periode *buffer* pendek.

Ada beberapa tipe *video streaming* antara lain *webcast*, dimana tayangan yang ditampilkan merupakan siaran langsung (*live*) dan *Video on Demand* (VOD), di mana tayangan yang akan ditampilkan sudah terlebih dahulu disimpan dalam *server*. Faktor-faktor yang mempengaruhi distribusi *video streaming* melalui jaringan antara lain : besarnya *bandwidth*, waktu tunda (*delay*), *lost packet*, dan juga teknik mendistribusikan video tersebut ke beberapa tujuan secara merata dan efisien. (Apostolopoulos, 2002, p1).

Ada tiga cara umum yang biasa digunakan dalam menerima *stream* data yaitu :

1. *Download*

Pada penerimaan *stream* data dengan cara *download*, akses video dilakukan dengan cara melakukan *download* terlebih dahulu suatu *file* multimedia dari *server*. Penggunaan cara ini mengharuskan keseluruhan suatu *file* multimedia harus diterima secara lengkap pada pengguna. *File* multimedia yang sudah diterima kemudian disimpan pada tempat penyimpanan yang ada di komputer. Pengguna baru dapat mengakses video tersebut setelah berhasil menerima *file* multimedia tersebut secara lengkap. Keuntungan dari penggunaan

cara *download* ini adalah akses yang lebih cepat ke salah satu bagian dari *file* tersebut. Sedangkan kekurangannya adalah pengguna yang ingin mengakses video tersebut harus menunggu terlebih dahulu sampai keseluruhan *file* multimedia tersebut diterima secara lengkap.

2. *Streaming*

Pada penerimaan video secara *streaming*, pengguna dapat melihat suatu *file* multimedia hampir bersamaan ketika *file* tersebut mulai diterima. Penggunaan cara ini mengharuskan pengiriman suatu *file* multimedia ke pengguna secara konstan. Hal ini bertujuan agar pengguna dapat menyaksikan video yang diterima secara langsung tanpa ada bagian yang hilang. Keuntungan dari cara ini adalah pengguna tidak perlu menunggu hingga suatu *file* multimedia dikirimkan secara lengkap. Dengan demikian, penggunaan cara ini memungkinkan sebuah *server* untuk melakukan pengiriman siaran secara langsung kepada pengguna.

3. *Progressive Downloading*

Progressive downloading adalah suatu metode *hybrid* yang merupakan hasil penggabungan antara metode *download* dan metode *streaming*, dimana video yang sedang diakses dapat diterima dengan cara *download* sehingga *player* yang ada pada pengguna sudah dapat mulai menampilkan video tersebut sejak

sebagian dari *file* tersebut diterima walaupun *file* tersebut belum diterima secara lengkap.

Secara umum, terdapat empat buah komponen dari *streaming* yaitu :

- *Input*

Sumber dari video yang akan di-*streaming*. Sumber tersebut dapat berupa *file* video, DVD, MPEG, dan lain-lain.

- *Encoder*

Bagian dari aplikasi *server* yang bertugas untuk mengubah video sumber menjadi sebuah format yang sesuai dengan transmisi *streaming*, dimana format ini umumnya memiliki tingkat kompresi tinggi sehingga dapat ditransmisikan dengan baik pada suatu media jaringan.

- *Server*

File hasil *encoding* kemudian didistribusikan oleh *server* kepada pengguna. Pada aplikasi yang digunakan, *encoder* dan *server* berada pada satu aplikasi yang sama yang terintegrasi satu sama lain.

- *Player* / *output*

Player berfungsi untuk melakukan *decoding* terhadap *file* hasil *streaming* dan menampilkannya pada pengguna.

Penerapan teknologi *video streaming* mengharuskan dilakukannya perancangan sistem dan jaringan secara matang untuk memungkinkan pengiriman *video streaming* yang berkualitas baik. Faktor yang mempengaruhi proses *video streaming* pada jaringan adalah *bandwidth*.

Bandwidth didefinisikan sebagai jumlah *bit-bit* informasi yang melalui suatu jaringan dalam periode waktu tertentu. *Bandwidth* yang tersedia di *internet* pada umumnya tidak dapat diketahui secara pasti dan sangat bervariasi terhadap waktu. Besarnya *bandwidth* yang tersedia pada jaringan sangat mempengaruhi proses kerja suatu *video streaming*. Jika *server* melakukan pengiriman sebuah video dengan *bit rate* tinggi yang melebihi kapasitas *bandwidth* yang tersedia, maka akan terjadi kemacetan sehingga paket-paket tersebut akan di-*drop*. Hal ini akan menyebabkan terjadinya penurunan kualitas video yang diterima. Hal ini juga berlaku apabila *server* melakukan pengiriman dengan *bit rate* yang lebih rendah. Oleh karena itu, seorang perancang jaringan harus mampu memperkirakan besar kapasitas *bandwidth* yang tersedia dan menyesuaikannya dengan *bit rate* video yang dikirimkan.

2.1.2 VOD (*Video on Demand*)

Sistem VOD memungkinkan pengguna untuk memilih dan menyaksikan video yang hendak diakses dalam jaringan sebagai bagian dari sistem interaktif. VOD dapat memanfaatkan proses *streaming*, *progressive downloading* ataupun *download*. Sistem VOD juga memungkinkan pengguna untuk melakukan kendali

pada protokol RTSP, seperti : *pause, fast-forward, fast-rewind, slow-forward*, dan lain-lain. Namun pada sistem yang menggunakan metode *streaming*, hal ini akan membebani *server* dan memerlukan pemakaian *bandwith* yang lebih besar.

2.1.3 *Video Encoding*

2.1.3.1 Pengertian *Video Encoding*

Menurut David Austerberry (2005, p154), *video encoding* adalah konversi sebuah sinyal video menjadi sebuah *file* media *streaming*. *Encoding* terdiri dari beberapa tahap, antara lain :

1. Mengambil sebuah video kenvensional atau sinyal televisi dan mengkonversinya menjadi sebuah format *file* yang dapat diproses oleh *software* komputer.
2. Mengurangi *data rate* dengan penyekalaan (*scalling*) dan kompresi menjadi sebuah *bit rate* yang dapat dikirim melalui *circuit dial-up* atau *broadband*.
3. Membungkus video yang terkompresi dalam sebuah format yang terpaket yang dapat di-*stream* melalui sebuah jaringan IP.

2.1.4 *Video Compression*

2.1.4.1 Pengertian *Video Compression*

Video compression atau kompresi video adalah suatu metode mengurangi jumlah data yang digunakan untuk menampilkan video tanpa mengurangi

kualitas gambar secara signifikan dan mengurangi jumlah bit yang digunakan untuk menyimpan dan atau mengirimkan gambar *digital*.

Pada dasarnya, video terdiri dari susunan titik warna secara tiga dimensi. Dua dimensi digunakan untuk menentukan arah *horizontal* dan *vertical* pada gambar bergerak, dan satu dimensi digunakan untuk menentukan posisi waktu.

2.1.5 *Bit Rate*

Bit rate adalah jumlah *bit* yang diproses per satu satuan waktu. *Bit rate* dapat disamakan dengan *transfer speed*, kecepatan koneksi, *bandwith*, ataupun *throughput maximum*. *Bit rate* juga dapat diartikan sebagai jumlah *bit* yang diproses dalam satu satuan waktu untuk mewakili media yang kontinu seperti video dan audio setelah dilakukan kompresi. Satuannya adalah *bit per second* (bps).

2.1.6 *Streaming Bandwidth and* S*torage*

Ukuran *streaming media storage* yang dibutuhkan diukur dari *streaming bandwidth* dan durasi dari video tersebut dengan rumus (berlaku untuk satu *user* dan satu *file*) :

storage size (megabytes) = durasi (detik) · *bit rate* (in kbit/s) / 8,388.608

(1 megabyte = 8 * 1,048,576 bits = 8,388.608 kilobits)

2.2 Teori-teori *Internet*

2.2.1 Pengertian *Internet* dan Sejarah Perkembangannya

Menurut Strauss (2003, p8), *internet* adalah sebuah jaringan global dari jaringan-jaringan yang saling berhubungan. Media penghubung tersebut bisa melalui kabel, satelit, maupun frekuensi radio. Dengan demikian, komputer-komputer yang saling terhubung dapat saling berkomunikasi. *Internet* berasal dari kata *Interconnection Networking* yang memiliki arti hubungan berbagai komputer dengan berbagai tipe yang membentuk suatu sistem jaringan yang mencakup seluruh dunia (jaringan komputer global) dengan melalui jalur telekomunikasi seperti telepon. Jaringan *internet* terbentuk melalui berjuta-juta komputer yang berintegrasi yang letaknya tersebar di seluruh dunia.

Internet pertama kali digunakan untuk keperluan militer. Pada waktu itu Departemen Pertahanan Amerika Serikat membangun sebuah sistem jaringan dengan menghubungkan semua komputer di daerah-daerah yang penting untuk mengatasi masalah yang berupa ancaman dari luar. Untuk itu, dibentuklah ARPANET yang dulunya merupakan suatu proyek kecil yang selanjutnya berkembang dengan pesat dan dipecah menjadi dua yaitu : MILNET untuk keperluan militer dan ARPANET untuk keperluan non-militer seperti universitas. Gabungan kedua jaringan ini dikenal dengan nama DARPA Internet, yang selanjutnya disederhanakan menjadi *internet* saja.

2.2.2 *Client-Server*

Client-server adalah suatu arsitektur yang memisahkan sebuah *client* dari sebuah *server* dimana hal ini banyak diimplementasikan pada jaringan komputer. Tipe arsitektur ini memungkinkan *device*-nya untuk berbagi *file* dan *resource*. Masing-masing *client* bisa mengirimkan *request* data ke satu atau lebih *server* yang terkoneksi. Kemudian *server* bisa menerima *request* tersebut, memprosesnya, dan mengembalikan informasi yang diminta kepada *client*. Contoh *client* misalnya adalah *web browser*, sedangkan contoh *server* misalnya adalah *web server*, *database server*, dan *mail server*. Contoh lainnya adalah *online game* yang juga merupakan tipe arsitektur *client-server*.

Karakteristik dari *client* antara lain menginisialisasi *request*, menunggu dan menerima balasan, biasanya terkoneksi dengan beberapa *server* pada suatu waktu, berinteraksi langsung dengan *end-user* dengan menggunakan sebuah *graphical user interface*. Sedangkan karakteristik dari *server* antara lain pasif, menunggu *request* dari *client*, ketika menerima *request* maka akan diproses dan kemudian membalasnya, biasanya menerima koneksi dari banyak *client*, tidak berinteraksi secara langsung dengan *client*.

2.2.3 *Web Browser*

Web browser adalah suatu *software* yang dijalankan pada komputer pengguna, yang meminta informasi dari *web server* dengan menampilkannya sesuai dengan *file* data itu sendiri.

2.2.4 *Hypertext Transfer Protocol* (HTTP)

Hypertext Transfer Protocol (HTTP) adalah suatu protokol yang digunakan untuk komunikasi atau mengirim informasi pada *World Wide Web* (WWW). HTTP juga merupakan suatu protokol *request* antara *client* dan *server*. *Client* membuat suatu HTTP *request* seperti *web browser* sedangkan *server* berguna untuk menyimpan dan membuat *resources* seperti *file* dan gambar pada HTML.

2.2.5 *World Wide Web* (WWW)

Menurut Oetomo et al (2003, p73), WWW merupakan aplikasi *internet* yang paling diminati oleh pengguna. WWW tidak lagi disertai dengan utilitas baris dan instruksi yang merupakan cara paling umum untuk menjelajah *internet*, tetapi dirancang dari ribuan halaman atau dokumen yang saling berhubungan yang dapat ditampilkan di layar monitor. Dengan menggunakan aplikasi ini, pengguna dapat dengan mudah mendapatkan informasi, tidak hanya teks tetapi juga gambar maupun multimedia.

Salah satu istilah yang berkaitan dengan WWW adalah *website* atau situs *web*. Menurut Sardi (2004, p4), *website* atau situs *web* adalah sekumpulan dokumen yang dipublikasikan melalui jaringan *internet* sehingga dapat diakses oleh pengguna melalui *web browser*. Dokumen tersebut dapat terdiri dari satu atau lebih kombinasi dari beberapa jenis *file* seperti *file* teks, gambar, suara, maupun video.

Informasi yang diletakkan di WWW sering disebut juga dengan "*HomePage*", dimana setiap *homepage* memiliki alamatnya masing-masing. Suatu *homepage* harus

dibuat semenarik mungkin dan banyak menyajikan informasi yang jelas agar dapat menarik perhatian dari para pengguna *internet* sehingga situs *web* tersebut menjadi sering dikunjungi.

WWW terdiri dari dua komponen dasar, yaitu :

1. *Web server*

Sebuah komputer dan perangkat lunak yang menyimpan dan mendistribusikan data ke komputer lainnya melalui *internet*.

2. *Web browser*

Perangkat lunak yang dijalankan pada komputer pengguna (*client*) yang meminta informasi dari *web server* dan menampilkannya sesuai dengan *file* data itu sendiri.

2.3 Teori-teori Web

2.3.1 Teori Interaksi Manusia dan Komputer

Interaksi manusia dan komputer atau *human computer interaction* adalah suatu disiplin ilmu yang berhubungan dengan perancangan, evaluasi, dan implementasi sistem komputer interaktif yang digunakan oleh manusia. Adapun delapan aturan emas yang umum digunakan dalam perancangan suatu *user interface* adalah :

1. Berusaha untuk konsisten

Desain tampilan yang ada harus dibuat sekonsisten mungkin dalam hal penamaan label, grafik, singkatan, *header*, *footer*, tampilan menu, dan lain sebagainya.

2. Memungkinkan *frequent user* menggunakan *shortcut*

Dengan semakin meningkatnya frekuensi penggunaan, maka semakin tinggi juga keinginan pengguna untuk mengurangi jumlah interaksi dan untuk meningkatkan kecepatan interaksi melalui penggunaan *shortcut*.

3. Memberikan umpan balik yang informatif

Untuk setiap aksi yang dijalankan pengguna, perlu diberikan umpan balik dari sistem. Untuk aksi yang minor dan sering dilakukan, respon sistem bisa dalam bentuk yang sederhana. Sedangkan untuk aksi yang utama dan jarang dilakukan, respon dari sistem bisa lebih khusus.

4. Merancang dialog untuk menghasilkan keadaan akhir (sukses/selesai)

Urutan aksi dapat dikelompokkan menjadi bagian awal, tengah, dan akhir. Umpan balik yang informatif pada penyelesaian sekelompok aksi dapat memberikan kepuasan serta memudahkan pengguna untuk masuk ke kelompok aksi yang berikutnya.

5. Memberikan penanganan kesalahan yang sederhana

Sedapat mungkin merancang suatu sistem dimana seorang pengguna tidak dapat membuat kesalahan yang serius. Contohnya, menonaktifkan karakter alfabetik dalam *field* untuk *entry* data numerik. Jika pengguna membuat kesalahan, sistem harus dapat

mendeteksi kesalahan tersebut serta menawarkan perintah yang sederhana, membangun, dan spesifik untuk memperbaiki kesalahan tersebut.

6. Mengizinkan pembalikkan aksi (*undo*) dengan mudah

Sedapat mungkin aksi yang dilakukan oleh pengguna dapat dibalik (di-*undo*). Fitur ini dapat mengurangi kekhawatiran pengguna karena pengguna mengetahui bahwa kesalahan yang diperbuat dapat dibalik.

7. Mendukung pengendalian secara internal

Pengguna yang berpengalaman menginginkan kesan bahwa dia dapat mengendalikan sistem dan sistem tersebut dapat merespon aksi yang telah dilakukannya. Hal-hal seperti aksi sistem yang mengejutkan, urutan *entry* data yang membosankan, kesulitan dalam memperoleh informasi yang dibutuhkan dan ketidakmampuan menghasilkan aksi yang diinginkan, dapat membuat pengguna menjadi tidak puas.

8. Mengurangi beban ingatan jangka pendek

Agar memudahkan beban ingatan pengguna, maka diperlukan tampilan yang sederhana dan frekuensi pergerakan *window* yang dikurangi.

2.3.2 *Top Ten Mistakes*

Top ten mistakes dalam suatu perancangan *user interface* adalah sebagai berkut : (**sumber http://www.useit.com/alertbox/9605.html**)

1. *Bad search*

Kesalahan yang sering terjadi adalah *search* tersebut tidak mampu mengatasi kalimat yang salah cetak. Kesalahan yang lain adalah jika *search* tersebut hanya dapat memanggil keseluruhan kata yang ingin dicari tanpa memisahkan mana yang paling sering dicari dan yang tidak.

2. *PDF Files for Online Reading*

Pada umumnya pengguna tidak menyukai adanya *file* PDF pada saat *browsing* karena kapasitas *file* PDF yang cukup besar dan membutuhkan waktu yang cukup lama untuk membukanya. Hal ini juga akan menyulitkan apabila koneksi *internet* yang ada tidak stabil.

3. *Not Changing the Color for Visited Link*

Tidak mengubah warna dari *link* yang telah dikunjungi akan membuat pengguna susah untuk menentukan arah mereka. Hal ini akan mempersulit karena pengguna tidak mengetahui *link* mana yang telah dikunjungi dan mana yang belum.

4. *Non-Scanable Text*

Tulisan yang dianggap penting bagi pengguna sebaiknya dipertebal atau diperbesar sehingga pengguna dapat mengetahui apa yang penting dalam informasi tersebut. Selain itu, sebaiknya menggunakan teknik penulisan yang sederhana karena akan lebih mempermudah pengguna untuk melihatnya.

5. *Fixed Font Sized*

Penggunaan *CSS style* akan membatasi pergerakan dari pengguna.

6. *Page Title with Low Search Engine Visibility*

Agar dapat mudah dibaca dan mudah dicari pada *search engine*, hindari penggunaan kata-kata yang umum dipakai seperti *welcome*, *the*, dan lain sebagainya pada judul halaman. Sebaiknya kata-kata yang digunakan berhubungan dengan sesuatu yang ditawarkan pada situs *web* tersebut.

7. *Anything that Look Like an Advertisement*

Hindari perancangan yang mirip dengan iklan-iklan yang ada pada umumnya seperti *pop-up menu*, *banne*r, dan animasi dimana hal ini akan dapat membuat pengguna menjadi malas untuk melihat apalagi untuk mempelajarinya lebih dalam.

8. *Violating Design Convention*

Inti dari suatu perancangan adalah konsistensi. Perancangan tanpa konsistensi akan menyulitkan pengguna untuk memakai hasil rancangan tersebut. Hal ini juga berarti melanggar aturan dasar dari perancangan yang telah ditetapkan.

9. *Opening New Browser Windows*

Perancangan sebaiknya tidak menampilkan dua *window* atau lebih karena hal itu dapat mengganggu pandangan dari pengguna. Selain itu, hindari penggunaan *pop-up menu* dan sejenisnya yang berpeluang untuk memperbanyak jumlah *window* yang ada.

10. *Not Answering Pengguna Question*

Feedback merupakan suatu hal yang sangat penting untuk menjawab keingintahuan pengguna. Penyediaan informasi sangat penting untuk menjawab segala yang dinginkan oleh pengguna dan informasi lainnya yang membuat pengguna tidak perlu bertanya-tanya lagi.

2.4 Teori-teori Basis Data

2.4.1 Sistem Basis Data

2.4.1.1 Pengertian Sistem

Menurut Pressman (2001, p276), sistem adalah tatanan elemen-elemen yang diatur untuk mencapai tujuan yang telah ditentukan sebelumnya melalui pemrosesan informasi. Setiap sistem terdiri dari unsur-unsur yang merupakan bagian terpadu dari sistem yang bersangkutan, unsur-unsur sistem tersebut bekerja sama untuk mencapai tujuan sistem. Maka pada dasarnya suatu sistem adalah sekelompok unsur yang erat berhubungan satu dengan yang lainnya, yang berfungsi bersama-sama untuk mencapai tujuan tertentu. Tujuan dari suatu sistem dibuat yaitu untuk menangani sesuatu yang berulang kali atau yang secara rutin terjadi.

2.4.1.2 Pengertian Data

Menurut Turban (2003, p15), data adalah fakta mentah atau deskripsi dasar dari sesuatu, kejadian, aktivitas, dan transaksi yang didapat, dicatat, disimpan, dan dikelompokkan, namun tidak terorganisasi sehingga tidak memberikan suatu arti yang spesifik.

2.4.1.3 Pengertian Basis data

Menurut Connolly dan Begg (2005, p15), basis data adalah sebuah kumpulan data yang saling berhubungan secara logis serta dapat dipakai secara bersama dan penjelasan mengenai data tersebut dirancang untuk memenuhi kebutuhan informasi dari suatu organisasi.

2.4.1.4 Komponen Sistem Basis data

Menurut Date (2000, p1), terdapat empat komponen penting dalam sistem basis data yaitu data, *hardware* (piranti keras), *software* (piranti lunak), dan pengguna.

1. Data

Data dalam basis data harus terintegrasi (*integrated*) dan dapat dipakai bersama (*shared*). Pengertian terintegrasi di sini adalah suatu basis data dapat dipandang sebagai suatu kumpulan *file-file* yang terkait satu sama lain dengan

menghilangkan redundansi yang ada. Pengertian dipakai bersama adalah setiap bagian data yang terdapat dalam basis data dapat digunakan oleh lebih dari satu pengguna dengan fungsi yang sama atau berbeda satu sama lain.

2. *Hardware* (piranti keras)

Untuk manajemen basis data, hanya dibutuhkan mesin standar. Namun yang harus diperhatikan adalah kapasitas penyimpanan karena basis data akan membutuhkan kapasitas yang besar. Komponen perangkat keras dari sistem terdiri dari:

- *Secondary storage* yang digunakan untuk menyimpan data. Bersama dengan peralatan I/O, *device controllers*, I/O *channels* yang saling berhubungan.
- *Processor* dan *main memory* yang digunakan untuk mendukung eksekusi perangkat lunak sistem basis data.

3. *Software* (piranti lunak)

Piranti lunak untuk sistem basis data disebut dengan DBMS (*Database Management System*). DBMS memungkinkan pengguna untuk membentuk *file* (*create*), penambahan data (*insert*), penghapusan (*delete*), dan lain-lain.

4. Pengguna

Pengguna basis data dibagi menjadi tiga kelompok, yaitu:

a.) Database Administrator (DBA)

Seorang atau grup personil pengolahan data yang bertanggung jawab terhadap kontrol keseluruhan basis data.

Tugas-tugas DBA yaitu:

- Menentukan isi informasi basis data
- Menentukan struktur penyimpanan dan strategi akses
- Menjadi penghubung antar pengguna
- Menentukan prosedur cek otorisasi dan validasi
- Menentukan strategi *backup* dan *recovery*
- Memonitor penampilan dan memberikan respon terhadap permintaan perubahan oleh pengguna

b.) Programmer

Programmer adalah seorang atau sekelompok orang yang menjadi tenaga ahli komputer yang berfungsi untuk mengembangkan program-program aplikasi yang diperlukan dalam manajemen basis data.

c.) End-user

Yang termasuk dalam kategori pengguna akhir adalah pemilik sistem, para manager, operator, dan sebagainya yang terlibat langsung dalam penggunaan basis data.

2.4.1.5 Keuntungan Penggunaan Basis data

Penggunaan sistem basis data memberikan banyak keuntungan, antara lain:

- Kontrol terpusat data operasional
- Redundansi data dapat dikurangi dan dikontrol
- Ketidakkonsistenan data dapat dihindarkan
- Data dapat dipakai bersama (*sharing*)
- Penerapan standarisasi
- Penerapan pembatasan keamanan data
- Integritas data dapat dipelihara
- Kebutuhan yang berbeda dapat diselaraskan
- Independensi data atau program

2.4.1.6 Kerugian Penggunaan Basis data

Selain memiliki banyak keuntungan, penggunaan basis data juga memiliki beberapa kerugian, antara lain:

- Mahal, karena membutuhkan biaya yang lebih besar untuk perangkat keras, perangkat lunak, dan personil yang lebih berkualitas.
- Kompleks, karena kemampuan perangkat lunak yang lebih besar, menjadi terlihat lebih rumit dan penguasaan yang lebih tinggi antara lain untuk kebutuhan sistem administrasi, prosedur *recovery*, prosedur *back up*, penataan keamanan data, penataan dalam rangka proses yang konkuren.

2.4.2 *Database Management System* (DBMS)

2.4.2.1 Pengertian DBMS

Menurut Connoly dan Begg (2005, p16), DBMS (*Database Management System*) adalah sebuah sistem piranti lunak yang memungkinkan pengguna untuk mendefinisikan, menciptakan, memelihara dan mengontrol akses ke basis data. Secara khusus, DBMS menyediakan fasilitas-fasilitas berikut :

- DBMS mengijinkan pengguna untuk mendefinisikan basis data, biasanya melalui *Data Definition Language* (DDL). Menurut Connoly dan Begg (2005, p40), DDL adalah sebuah bahasa yang memperbolehkan DBA atau pengguna untuk mendeskripsikan dan menyebutkan entitas, atribut, dan relasi yang diperlukan untuk aplikasi, bersama dengan batasan keamanan dan integritas datanya.

- DBMS mengijinkan pengguna untuk *insert, update, delete,* dan *retrieve* data dari basis data, biasanya melalui *Data Manipulation Language* (DML). Menurut Connoly dan Begg (2005, p40), DML adalah sebuah bahasa yang menyediakan sekumpulan operasi untuk mendukung operasi manipulasi data yang mendasar pada data yang tersimpan pada basis data. Operasi manipulasi data biasanya meliputi :

  - Penyisipan (*insert*) data baru ke dalam basis data
  - Modifikasi data yang tersimpan di dalam basis data
  - Pengambilan data yang terdapat di dalam basis data
  - Penghapusan data dari basis data

- DBMS menyediakan akses kontrol ke basis data. Sebagai contohnya, DBMS menyediakan :

1. *Security system,* yang mencegah pengguna yang tidak diberi kuasa untuk mengakses basis data.
2. *Integrity system,* yang memelihara konsistensi penyimpanan data.
3. *Concurrency control system,* yang mengijinkan basis data untuk diakses secara *share*.
4. *Recovery control system,* yang mengembalikan basis data ke sebuah *state* konsisten terdahulu pada saat terjadi kesalahan pada piranti keras ataupun piranti lunak.
5. *User-accesible catalog*, yang berisi deskripsi data pada basis data.

2.4.2.2 Komponen DBMS

Menurut Connoly dan Begg (2002, p18), ada lima komponen *Database Management System* (DBMS) yaitu:

- *Hardware* (piranti keras)

Piranti keras dibutuhkan untuk menjalankan DBMS dan aplikasi-aplikasi. Piranti keras dapat mencakup dari *single personal computer, single mainframe*, atau jaringan komputer.

- *Software* (piranti lunak)

Komponen piranti lunak terdiri dari piranti lunak DBMS itu sendiri dan program-program aplikasi, bersama dengan sistem operasi, termasuk piranti lunak jaringan jika DBMS digunakan melalui jaringan. Secara khas, program-program aplikasi ditulis dalam *third-generation programming language* (3GL), seperti : C, C++, Java, Visual Basic, COBOL, Fortran, Ada, Pascal, atau menggunakan *fourth-generation programming language* (4GL), seperti : SQL, yang ditempelkan ke dalam 3GL.

- Data

Data merupakan komponen yang paling penting dari DBMS, khususnya dari sudut pandang pengguna akhir. Data bertindak sebagai penghubung antara komponen-komponen mesin (*hardware* dan *software*) dan komponen-komponen manusia (*procedure* dan *people*).

- *Procedure* (prosedur)

Prosedur menunjuk kepada instruksi-instruksi dan aturan-aturan yang mengatur perancangan dan penggunaan basis data. Pengguna sistem dan staf yang mengatur basis data memerlukan dokumentasi prosedur tentang bagaimana menggunakan atau menjalankan sistem. Prosedur ini dapat berisi instruksi mengenai bagaimana masuk ke dalam DBMS, bagaimana menggunakan fasilitas atau aplikasi program DBMS tertentu, bagaimana memulai dan menghentikan DBMS, bagaimana membuat salinan *backup* dari basis data, bagaimana mengatasi kegagalan piranti keras atau piranti lunak, dan bagaimana mengubah struktur sebuah tabel, mengatur

ulang basis data melalui banyak *disk*, meningkatkan unjuk kerja, atau menyimpan data ke *secondary storage*.

- *People* (Orang)

Komponen terakhir adalah orang yang terlibat dengan sistem, termasuk di dalamnya adalah *Database Administrator* (DBA), perancang basis data, pengembang aplikasi, dan pengguna akhir.

2.4.2.3 Keuntungan DBMS

Keuntungan dari DBMS, antara lain :

- Terdapat kontrol terhadap pengulangan data
- Konsistensi data
- Semakin banyak informasi yang didapat dari jumlah data yang sama
- Penggunaan data secara bersama-sama
- Peningkatan integritas data
- Peningkatan keamanan data
- Penetapan standarisasi
- Pengurangan biaya
- Penyesuaian kebutuhan yang berlawanan
- Peningkatan pengaksesan data dan kemampuan respon
- Peningkatan produktivitas
- Peningkatan perawatan melalui independensi data
- Peningkatan konkurensi

- Peningkatan layanan *backup* dan *recovery*

2.4.2.4 Kerugian DBMS

Selain mempunyai banyak keuntungan, DBMS juga memiliki beberapa kerugian yaitu:

- Kompleksitas

  Penetapan fungsionalitas yang diharapkan dari DBMS yang baik menyebabkan DBMS menjadi bagian piranti lunak yang sangat rumit. Perancang dan pengembang basis data, administrator data dan basis data, serta pengguna akhir harus memahami fungsi-fungsi yang ada dengan baik sehingga dapat memperoleh keseluruhan manfaatnya.

- Ukuran

  Kompleksitas dan banyaknya fungsi yang ada menyebabkan DBMS menjadi bagian piranti lunak yang berukuran sangat besar, menempati banyak *megabytes* dari *disk space* dan memerlukan sejumlah memori tambahan agar dapat berjalan secara efektif.

- Biaya DBMS

  Biaya dari DBMS sangat bervariasi tergantung dari lingkungan dan fungsi-fungsi yang disediakan. Contohnya : sebuah *single-user* DBMS untuk *Personal Computer* (PC) hanya berharga US$100, sedangkan sebuah *mainframe multi-user* DBMS yang dapat melayani ratusan pengguna bisa sangat mahal, sekitar US$100,000 atau bahkan US$1,000,000.

- Biaya tambahan piranti keras

Kebutuhan penyimpanan *disk* untuk DBMS dan basis data mengharuskan pembelian ruang penyimpanan tambahan. Selanjutnya, untuk mencapai unjuk kerja yang diperlukan, mungkin perlu membeli mesin yang lebih besar. Dengan demikian, pengadaan piranti keras tambahan mengakibatkan pembelanjaan lebih lanjut.

- Biaya konversi

Dalam beberapa situasi, biaya DBMS dan piranti keras tambahan mungkin tidak penting dibandingkan dengan biaya untuk mengkonversi aplikasi yang ada untuk dijalankan pada DBMS dan piranti keras yang baru. Biaya ini juga termasuk biaya untuk melatih staf untuk menggunakan sistem yang baru ini dan mungkin juga biaya untuk mempekerjakan staf spesialis untuk membantu mengkonversi dan menjalankan sistem baru. Biaya ini menjadi salah satu alasan utama mengapa beberapa organisasi puas dengan sistem mereka yang sekarang dan tidak dapat menggantinya ke teknologi basis data yang lebih modern.

- Unjuk kerja

Secara khusus, sebuah sistem yang berbasis *file* ditulis untuk sebuah aplikasi spesifik seperti *invoicing.* Hasilnya, unjuk kerja secara umum sangat baik. Bagaimanapun, DBMS harus ditulis menjadi lebih umum lagi untuk menyediakan banyak aplikasi daripada satu aplikasi. Akibatnya, beberapa aplikasi mungkin tidak berjalan secepat biasanya.

- Dampak lebih besar dari kegagalan

Pemusatan dari sumber daya dapat meningkatkan kemudahan rusak dari sebuah sistem karena semua pengguna dan aplikasi bergantung pada ketersediaan dari DBMS, kegagalan beberapa komponen bisa membuat operasi terhenti.

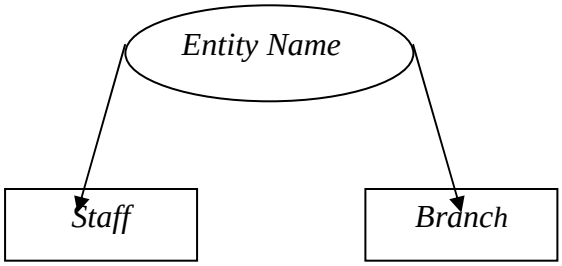
2.4.3 *Entity Relationship* (ER) *Modeling*

Menurut Connoly dan Begg (2002, p342), *Entity Relationship Modeling* merupakan pendekatan *top-down* dalam perancangan basis data yang dimulai dengan mengidentifikasi data penting yang dinamakan *entity* dan *relationship* antara data yang harus direpresentasikan dalam model.

2.4.3.1 Entity Types (tipe *entity*)

Konsep dasar dari *Entity Relationship Modeling* adalah *entity types,* yakni kumpulan dari objek-objek dengan sifat (*properties*) yang sama, yang diidentifikasikan oleh perusahaan yang keberadaannya tidak tergantung dengan yang lain. Keberadaannya dapat berupa obyek dengan sebuah keberadaan fisik (nyata) ataupun obyek dengan keberadaan konseptual (abstrak).

Entity occurence adalah obyek yang dapat diidentifikasikan secara unik dari sebuah tipe *entity*.

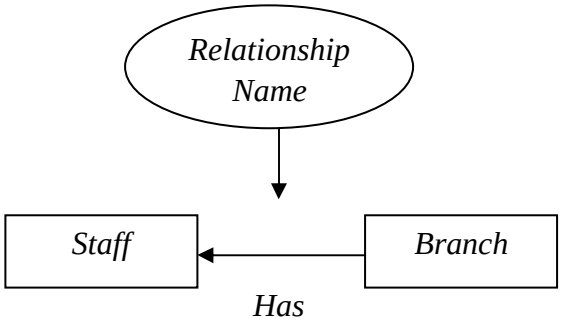


Gambar 2.1 Contoh Tipe *Entity*

(Sumber Connoly dan Begg, 2005, p345)

2.4.3.2 Relationship Types

Relationship types adalah sekumpulan keterhubungan yang mempunyai arti antara tipe *entity* yang ada. Setiap *relationship type* diberi nama yang mendeskripsikan fungsinya. *Relationship occurence* adalah sebuah keterhubungan yang dapat diidentifikasikan secara unik, yang meliputi satu keberadaan dari tiap tipe *entity* yang berpartisipasi.



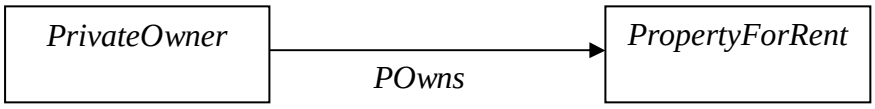
'Branch has staff'

Gambar 2.2 Contoh Tipe *Entity*

(Sumber Connoly dan Begg, 2005, p345)

Derajat *relationship* adalah jumlah tipe *entity* yang berpartisipasi dalam suatu *relationship*. Derajat *relationship* terdiri dari:

- *Binary relationship*, yaitu keterhubungan antar dua tipe *entity*. Contoh *binary relationship* antara *PrivateOwner* dengan *PropertyForRent* yang disebut *POwns*.

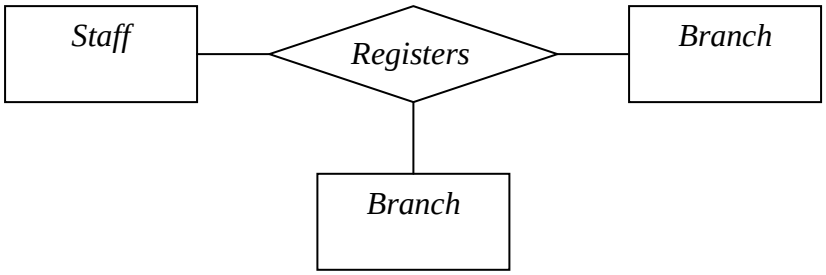


'Private owner owns property for rent'

Gambar 2.3 Contoh *Binary Relationship*

(Sumber Connoly dan Begg, 2005, p348)

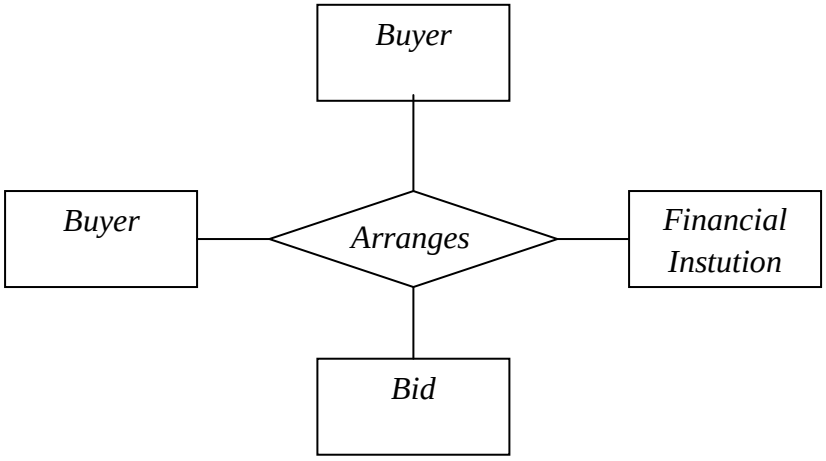
- *Ternary relationship*, yaitu keterhubungan antar tiga tipe *entity*. Contoh *ternary relationship* yang dinamakan *Registers*. Relasi ini melibatkan tiga tipe *entity* yaitu *Staff*, *Branch* dan *Client* yang menggambarkan *Staff* mendaftarkan *Client* pada *Branch*.



Gambar 2.4 Contoh *Ternary Relationship*

(Sumber Connoly dan Begg, 2005, p348)

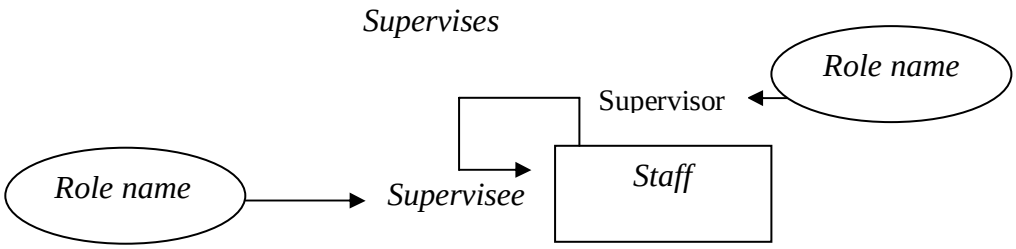
- *Quaternary relationship,* yaitu keterhubungan antar empat tipe *entity.* Contoh *quaternary relationship* yang dinamakan *Arranges.* Relasi ini melibatkan empat tipe *entity* yaitu *Buyer*, *Solicitor*, *Financial Instution* dan *Bid* yang menggambarkan *Buyer* diberi masukkan oleh *Solicitor* dan didukung oleh *Financial Instution*, melakukan penawaran (*Bid*).



Gambar 2.5 Contoh *Quaternary Relationship*

(Sumber Connoly dan Begg, 2005, p349)

- *Unary relationship* (*Recursive relationship*), yaitu keterhubungan antar satu tipe *entity*, dimana tipe *entity* tersebut berpartisipasi lebih dari satu kali dengan peran yang berbeda. Relasi ini dapat diberikan *role names* (aturan penamaan) untuk mengidentifikasikan keterkaitan tipe *entity* dalam relasi. Contoh : *entity Staff* yang berperan sebagai *Supervisor* dan *Staff* yang di-*supervisor*-i.



Gambar 2.6 Contoh *Unary Relationship*

(Sumber Connoly dan Begg, 2005, p349)

2.4.3.3 *Attributes* (atribut) dan *Keys*

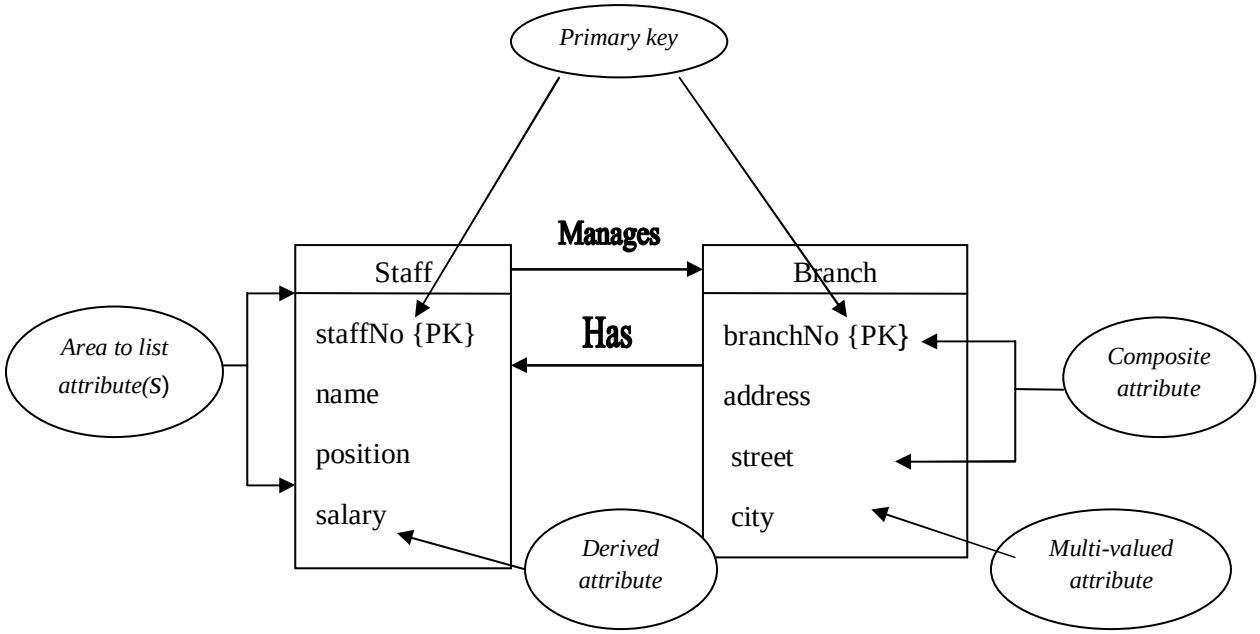
Attribute merupakan sifat-sifat (*property*) dari sebuah tipe *entity* atau tipe *relationship*. Contohnya: sebuah tipe *entity Staff* digambarkan oleh atribut *staffNo*, *name*, *position* dan *salary*. *Attribute domain* adalah himpunan nilai yang diperbolehkan untuk satu atau lebih atribut. Setiap atribut yang dihubungkan dengan sekumpulan nilai disebut *domain*. *Domain* menetapkan nilai potensial yang dapat disimpan oleh sebuah atribut atau sama dengan konsep *domain* pada model relasional. Macam-macam atribut, antara lain:

- *Simple Attribute*, yaitu atribut yang terdiri dari satu komponen tunggal dengan keberadaan yang tidak tergantung dan tidak dapat dibagi menjadi bagian yang lebih kecil lagi. Dikenal juga dengan nama *Atomic Attribute*.
- *Composite Attribute*, yaitu atribut yang terdiri dari beberapa komponen, dimana masing-masing komponen memiliki keberadaan yang tidak tergantung. Contohnya atribut *Address* dapat terdiri dari *Street, City, PostCode*.

- *Single-valued Attribute*, yaitu atribut yang mempunyai nilai tunggal untuk setiap kejadian dari sebuah tipe *entity*. Contohnya tipe *entity Branch* memiliki satu nilai untuk atribut *branchNo* pada setiap kejadian.
- *Multi-vallued Attribute*, yaitu atribut yang mempunyai beberapa nilai untuk setiap kejadian dari sebuah tipe *entity*. Contohnya tipe *entity Branch* memiliki beberapa nilai untuk atribut *telpNo* pada setiap kejadian.
- *Derived Attribute*, yaitu atribut yang mewakili sebuah nilai yang dihasilkan dari nilai dari satu atau kumpulan atribut yang berhubungan, dan tidak harus berasal dari tipe *entity* yang sama.

Keys dibagi menjadi tiga jenis, antara lain :

-*Candidate Key*, yaitu kumpulan minimal atribut-atribut yang dapat mengidentifikasikan setiap kejadian dari sebuah tipe *entity* secara unik.
-*Primary key*, yaitu c*andidate key* yang dipilih untuk mengidentifikasikan setiap kejadian dari sebuah tipe *entity* secara unik.
- *Composite key*, yaitu sebuah *candidate key* yang terdiri dari dua atau lebih atribut.



Gambar 2.7 Contoh Representasi Atribut

(Sumber Connoly dan Begg, 2005, p354)

2.4.3.4 *Strong and Weak Entity Types*

Strong entity type adalah tipe *entity* yang keberadaannya tidak tergantung pada tipe *entity* lain, sedangkan *weak entity type* adalah tipe *entity* yang keberadaannya bergantung pada tipe *entity* lain. *Strong entity type* terkadang disebut sebagai *entitty parent, owner,* atau *dominant* dan *weak entitty type* sebagai *entitty child, dependent,* atau *subordinate*.

2.4.3.5 *Attributes on Relationship*

Representasi atribut yang berhubungan dengan tipe relasi menggunakan simbol yang sama dengan tipe *entity*. Dalam membedakan antara sebuah relasi dengan atribut dan *entity*, segi empat yang merepresentasikan atribut dari relasi menggunakan garis putus-putus.

2.4.3.6 *Structural Constraints*

Batasan utama pada *relationship* disebut *multiplicity*, yaitu jumlah (atau jangkauan) dari kejadian yang mungkin terjadi pada sebuah tipe *entity* yang mungkin terhubung ke satu kejadian tunggal dari tipe *entity* lain yang berhubungan melalui suatu *relationship* tertentu. Derajat *relationship* yang paling umum adalah *binary*. Macam-macam *binary relationship*, antara lain :

- *One-to-one* (1:1) *Relationship*

  Relationship ini terjadi bila tiap anggota *entity Staff* hanya boleh berpasangan dengan satu anggota dari *entity Branch*. Sebaliknya, tiap anggota dari *entity Branch* hanya boleh berpasangan dengan satu anggota dari *entity Staff*.

- *One-to-many* (1:*) *Relationship*

  Relationship ini terjadi bila tiap anggota *entity Staff* boleh berpasangan dengan lebih dari satu anggota *entity PropertyForRent*. Akan tetapi, tiap anggota *entity PropertyForRent* hanya boleh berpasangan dengan satu anggota *entity Staff*.

- *Many-to-many* (*:*) *Relationship*

Relationship ini terjadi bila tiap anggota *entity Newspaper* boleh berpasangan dengan lebih dari satu anggota *entity PropertyForRent*. Sebaliknya tiap anggota *entity PropertyForRent* juga boleh berpasangan dengan lebih dari satu anggota *entity Newspaper*.

Multiplicity for Complex Relationship adalah jumlah (atau jangkauan) dari kejadian yang mungkin dari sebuah tipe *entity* dalam *n-ary relationship* ketika nilai *entity* yang lain (n-1) diketahui.

Multiplicity dibentuk dari dua macam batasan pada *relationship*, yaitu : *cardinality* dan *participation*. *Cardinality* menjelaskan jumlah maksimum dari kejadian *relationship* yang mungkin untuk sebuah *entity* yang berpartisipasi di dalam tipe *relationship* tersebut. Sedangkan *participation* menetapkan apakah seluruh atau hanya beberapa kejadian *entity* yang berpartisipasi dalam suatu *relationship*. Jika seluruh *entity* kejadian *entity* terlibat dalam suatu *relationship* tertentu, maka disebut *mandatory participation*. Jika hanya beberapa kejadian *entity* yang terlibat, maka disebut *optional participation*.

2.4.4 Normalisasi

2.4.4.1 Pengertian Normalisasi

Menurut Connolly dan Begg (2005, p388), normalisasi adalah sebuah teknik untuk menghasilkan sekumpulan tabel dengan *properties* yang diinginkan,

sesuai dengan kebutuhan data dari perusahaan. Normalisasi sering dilakukan sebagai rangkaian dari pengujian pada suatu hubungan untuk menentukan apakah hubungan tersebut benar atau melanggar persyaratan dari bentuk normal yang ditentukan.

Normalisasi merupakan sebuah teknik dalam *logical design* (rancangan logikal) sebuah basis data, teknik pengelompokkan atribut dari suatu relasi sehingga membentuk struktur relasi yang baik (tanpa redudansi).

2.4.4.2 Tujuan Normalisasi

Tujuan dari normalisasi antara lain :

- Untuk menghilangkan kerangkapan data
- Untuk mengurangi kompleksitas
- Untuk mempermudah akses, modifikasi, dan perawatan data
- Untuk meminimalkan penggunaan ruang penyimpanan pada komputer.

2.4.4.3 Bentuk Normal

Secara umumnya normalisasi dibagi menjadi tingkatan, yaitu bentuk normal pertama (1NF) berdasarkan penghilangan *repeating group*, bentuk normal kedua (2NF) berdasarkan pada ketergantungan fungsional (*functional dependency*), bentuk normal ketiga (3NF) yang berdasarkan pada ketergantungan transitif (*transitive dependency*).

2.4.4.3.1 *First Normal Form* (1NF)

Sebuah relasi dimana setiap persimpangan antara masing-masing baris dan kolom mengandung satu dan hanya satu nilai.

Aturan :

- Mendefinisikan atribut kunci
-Tidak adanya kelompok yang berulang. Tabel yang masih mengandung kelompok berulang (*repeating group*) berarti masih dalam bentuk *Unnormalized Form* (UNF)
- Semua atribut bukan kunci tergantung pada atribut kunci

2.4.4.3.2 *Second Normal Form* (2NF)

Sebuah relasi yang telah memenuhi bentuk normal pertama (1NF) dan setiap atribut yang bukan *primary key* harus *fully functionally dependent* terhadap *candidate key* mana pun.

Aturan :

- Sudah memenuhi bentuk normal pertama (1NF)
- Sudah tidak ada ketergantungan *parsial*, dimana seluruh *field* hanya tergantung pada sebagian *field* kunci

2.4.4.3.3 *Third Normal Form* (3NF)

Sebuah relasi yang telah memenuhi normal pertama (1NF) dan normal kedua (2NF) dan tidak ada atribut bukan *primary key* yang *transitively dependent* terhadap primary key.

Aturan:

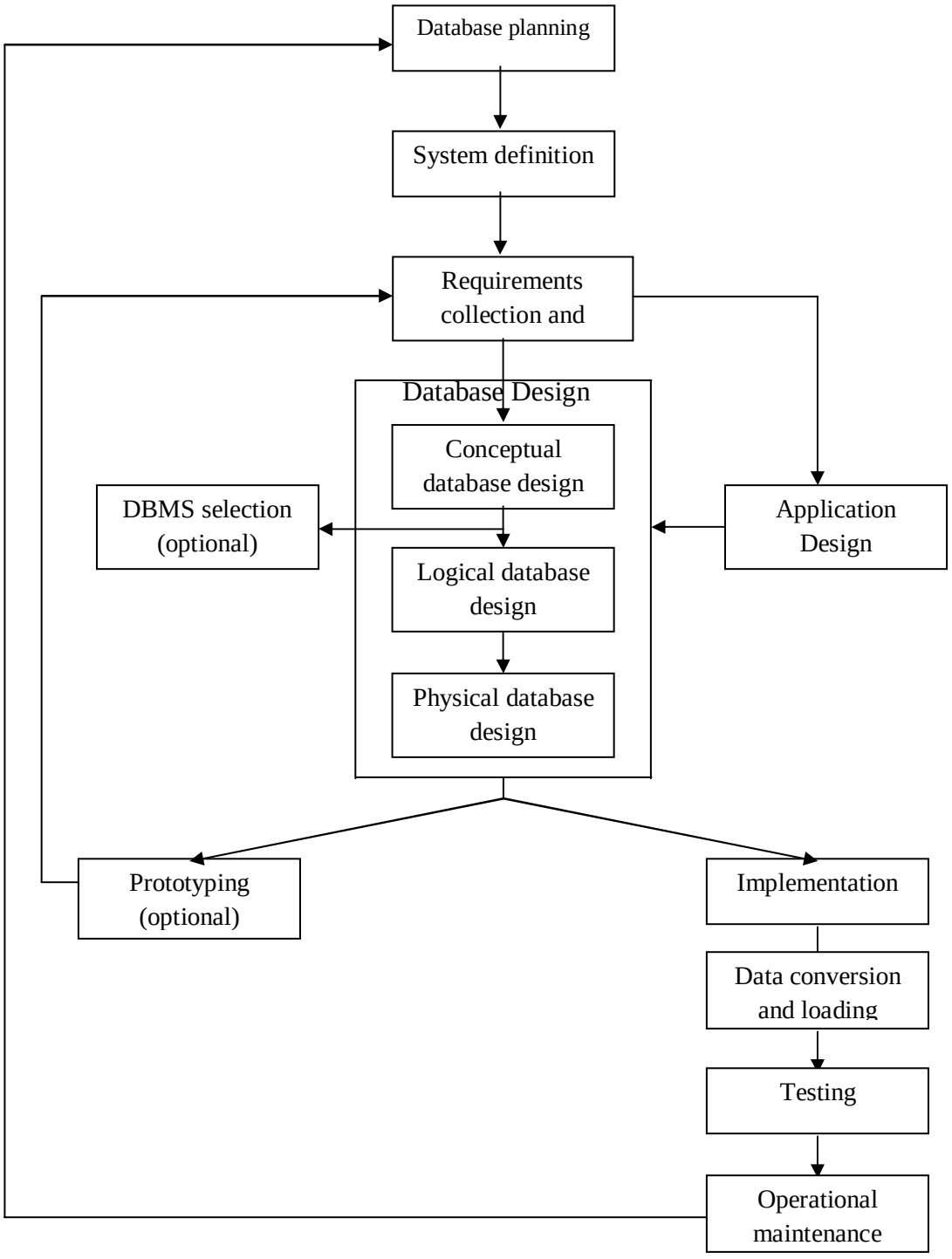
- Sudah berada dalam bentuk normal kedua (2NF)
- Tidak ada ketergantungan transitif, dimana *field* bukan kunci tergantung pada *field* bukan kunci lainnya

Bentuk normal pertama hingga ketiga merupakan bentuk normal yang umum dipakai. Artinya, bahwa pada kebanyakan relasi bila ketiga bentuk normal tersebut telah dipenuhi maka persoalan anomali tidak akan muncul lagi.

2.4.5 *Database System Development Lifecycle*

Menurut Connolly dan Begg (2005, p283), sistem basis data merupakan komponen dasar dari organisasi yang besar dengan sistem informasi yang luas. Hal penting yang perlu diperhatikan dalam *database system development lifecycle* adalah bahwa tingkatannya tidak sepenuhnya berurutan (*sequential*), tapi melibatkan sejumlah pengulangan langkah-langkah sebelumnya melalui *feedback loop*. Sebagai contoh, masalah ditemukan selama perancangan basis data yang membutuhkan pengumpulan dan analisis kebutuhan tambahan. Untuk sistem basis data yang kecil

dengan jumlah pemakai yang sedikit maka *lifecycle*-nya tidak terlalu kompleks. Sebaliknya, ketika merancang sistem basis data yang berukuran sedang sampai ke basis data yang besar dengan puluhan ribu pemakai, menggunakan ratusan *query* dan program aplikasi, maka *lifecycle* akan menjadi sangat kompleks. Langkah-langkah dari *database system development lifecycle* dapat dilihat pada gambar berikut ini.



Gambar 2.8 ***Database System Development Lifecycle***

(Sumber Connoly dan Begg, 2005, p284)

2.4.5.1 *Database Planning*

Menurut Connolly dan Begg (2005, p285), *database planning* adalah aktivitas manajemen yang memperbolehkan langkah-langkah pada *database system development lifecycle* untuk direalisasikan seefisien dan seefektif mungkin. Perencanaan basis data harus terintegrasi dengan keseluruhan strategi sistem informasi dari organisasi atau perusahaan yang bersangkutan. Ada tiga masalah pokok dalam merumuskan suatu strategi sistem informasi, yaitu :

1. Identifikasi rencana dan tujuan perusahaan dengan penentuan sistem informasi yang diperlukan.
2. Evaluasi sistem informasi saat ini untuk menentukan kelemahan dan kekuatan yang ada.
3. Penilaian tentang peluang IT yang mungkin memberikan keuntungan yang kompetitif.

Kegiatan utama dari perencanaan basis data adalah merencanakan supaya tingkat *lifecycle* dapat menjadi efisien dan efektif. Hal pertama yang harus dilakukan adalah menentukan *mission statement* dari sistem basis data. *Mission statement* mendefinisikan tujuan utama dari sistem basis data. Kemudian tahap selanjutnya adalah mengidentifikasikan *mission objective*

dimana setiap *mission objective* harus mengidentifikasikan tugas tertentu yang harus didukung oleh sistem basis data.

2.4.5.2 *System Definition*

Menurut Connolly dan Begg (2005, p286), *system definition* bertujuan untuk mendeskripsikan ruang lingkup dan batasan-batasan aplikasi basis data serta *major user view*. Sebelum mencoba untuk merancang suatu sistem basis data, pertama-tama harus mengidentifikasi batasan-batasan sistem yang akan dikembangkan dan bagaimana sistem tersebut dapat terhubung dengan bagian lain dari sistem informasi organisasi. Penting juga bahwa hal ini tidak hanya meliputi area aplikasi dan para pengguna yang sekarang ke dalam batasan-batasan sistem, tetapi juga aplikasi dan para pengguna yang akan datang.

Suatu sistem basis data mungkin mempunyai satu atau lebih *user views*. Mengidentifikasi *user views* adalah suatu aspek yang penting dalam mengembangkan suatu sistem basis data karena *user views* dapat membantu untuk memastikan bahwa tidak ada pengguna utama dalam basis data tersebut yang terlupakan ketika mengembangkan kebutuhan untuk sistem basis data yang baru. *User views* sangat membantu dalam pengembangan sistem basis data yang relatif kompleks dengan memecahkan kebutuhan-kebutuhan menjadi bagian-bagian yang dapat dikendalikan.

User views mendefinisikan apa yang diperlukan dari suatu sistem basis data dalam kaitannya dengan data yang disimpan dan transaksi yang akan

dilakukan terhadap data (dengan kata lain, apa yang akan dilakukan oleh pengguna dengan data tersebut). Kebutuhan *user views* mungkin beda dengan *view* yang bersangkutan atau tumpang-tindih dengan *view* lain.

2.4.5.3 *Requirement Collection and Analysis*

Menurut Connolly dan Begg (2005, p288), *requirement collection and analysis* adalah proses pengumpulan dan analisis informasi mengenai bagian dari organisasi yang akan didukung oleh sistem basis data, dan menggunakan informasi ini untuk mengidentifikasi kebutuhan untuk sistem baru. Banyak teknik yang dapat digunakan untuk mengumpulkan informasi. Teknik–teknik ini dinamakan sebagai *fact-finding techniques.*

Terdapat lima teknik pengumpulan informasi yang digunakan dalam pengembangan basis data, antara lain :

1. Mempelajari dokumentasi

Teknik ini sangat bermanfaat ketika berusaha memperoleh pengetahuan untuk membangun kebutuhan dari basis data. Dokumentasi juga membantu menyediakan informasi pada bagian dari perusahaan yang berkaitan dengan masalah yang dihadapi. Dengan mempelajari dokumen-dokumen, formulir, laporan dan *file* yang berkaitan dengan sistem yang ada, dapat dengan cepat diperoleh beberapa pemahaman tentang sistem.

2. Wawancara

Teknik ini sangat populer dan umum digunakan serta memungkinkan pengumpulan informasi dari individual secara *face-to-face*. Tujuan kegiatan ini yaitu menemukan fakta baru, verifikasi fakta, klarifikasi fakta, generalisasi antusiasme, melibatkan *end-user*, identifikasi kebutuhan dan pengumpulan ide dan pendapat**.** Akan tetapi, menggunakan teknik wawancara membutuhkan kemampuan komunikasi yang baik untuk menghadapi orang-orang yang mempunyai pandangan yang berbeda dalam hal prioritas pendapat, motivasi dan kepribadian.

Keuntungan dari wawancara :

- Memungkinkan orang yang diwawancarai untuk merespon secara bebas dan terbuka terhadap suatu pertanyaan.
- Memungkinkan orang yang diwawancarai untuk menjadi bagian dari proyek.
- Memungkinkan pewawancara untuk mengadaptasi atau menyusun ulang pertanyaan selama proses wawancara.
- Memungkinkan pewawancara untuk mengamati gerak tubuh dari orang yang diwawancarai.

Kekurangan dari wawancara :

- Banyak menghabiskan waktu dan biaya.
- Kesuksesan tergantung kemampuan komunikasi dari si pewawancara.
- Kesuksesan dapat bergantung kepada kemauan dari orang yang diwawancara

untuk berpartisipasi dalam wawancara.

3. Observasi

Teknik ini merupakan salah satu teknik *fact finding* yang paling efektif untuk memahami sebuah sistem. Teknik ini memungkinkan untuk berpartisipasi atau mengawasi seseorang dalam beraktivitas untuk mempelajari sistem.

Keuntungan dari observasi :

- Memungkinkan pengecekan valid tidaknya suatu fakta dan data
- Pengamat dapat melihat secara langsung apa saja yang sedang dilakukan.
- Pengamat dapat juga memperoleh data dengan mendeskripsikan lingkungan fisik dari suatu pekerjaan.
- Relatif tidak mahal

Kekurangan dari observasi :

- Seseorang dapat bertindak secara berbeda ketika tahu bahwa ia sedang diamati.
- Adanya kemungkinan kehilangan tugas pengamatan tergantung dari tingkat kesulitan yang berbeda.
- Beberapa tugas mungkin tidak selalu dilakukan ketika sedang perlu diamati.
- Kurang praktis.

4. Penelitian

Salah satu teknik *fact finding* yang berguna adalah melakukan penelitian terhadap aplikasi dan masalahnya. Jurnal-jurnal komputer, buku-buku petunjuk dan internet merupakan sumber-sumber informasi yang bagus. Mereka dapat

menyediakan informasi tentang bagaimana orang lain memecahkan masalah yang serupa.

Kekurangan dari penelitian :

- Menghabiskan waktu.
- Membutuhkan akses ke sumber informasi yang dibutuhkan.
- Mungkin tidak dapat membantu dalam menyelesaikan masalah sebab masalahnya tidak didokumentasikan.

Keuntungan dari penelitian :

- Dapat menghemat waktu jika solusinya sudah ada.
- Peneliti dapat melihat orang lain dalam menyelesaikan masalah yang serupa.
- Peneliti dapat memperoleh informasi yang *up-to-date*.

5. Kuesioner

Teknik *fact finding* yang lain adalah melakukan survei dengan menggunakan kuesioner. Kuesioner adalah dokumen dengan tujuan khusus untuk mengumpulkan fakta-fakta dari sejumlah orang. Jika ingin mengumpulkan informasi dari orang banyak, teknik yang paling efisien adalah kuisioner. Ada dua tipe jenis pertanyaan pada kuesioner, yaitu :

- *Free format questions*

Free format questions memberikan kebebasan yang lebih luas dari responden untuk memberikan jawaban.

- *Fix format questions*

Fix format questions membutuhkan jawaban yang lebih spesifik dari responden. Setiap pertanyaan yang diberikan sudah ada pilihan jawaban sehingga jawaban dari responden lebih terkontrol.

Kekurangan dari kuesioner :

- Jumlah responden mungkin rendah, mungkin hanya 5% sampai 10%.
- Kuesioner mungkin dikembalikan dengan jawaban yang tidak lengkap.
- Mungkin tidak menyediakan kesempatan untuk beradaptasi atau mengubah pertanyaan yang salah ditaksirkan.
- Tidak bisa meninjau dan meneliti bahasa tubuh responden.
- Dapat memakan waktu untuk menyiapkan kuesioner.

Keuntungan dari kuesioner :

- Orang-orang dapat mengisi dan mengembalikan kuesioner sesuai dengan keinginan mereka.
- Secara relatif merupakan cara yang murah untuk memperoleh data dari orang banyak.
- Tanggapan dapat dikumpulkan dan dianalisa dengan cepat.

Hal penting yang perlu diperhatikan sehubungan dengan langkah ini adalah memutuskan bagaimana berhadapan dengan situasi dimana terdapat lebih

dari satu *user views* untuk sistem basis data. Ada tiga pendekatan utama untuk mengatur kebutuhan dari sistem basis data dengan banyak *user views*, yaitu :

1. Pendekatan terpusat (*centralized approach*)

Kebutuhan untuk tiap *user view* digabung ke dalam kumpulan kebutuhan tunggal untuk sistem basis data baru. Sebuah model data mewakili semua *user view* yang diciptakan selama langkah perancangan basis data.

2. Pendekatan integrasi *view* (*view integration approach*)

Kebutuhan untuk tiap *user view* tetap sebagai daftar yang terpisah. Model data-model data mewakili tiap *user view* yang diciptakan dan kemudian digabungkan selama langkah perancangan basis data.

3. Kombinasi kedua pendekatan

2.4.5.4 *Database Design*

Menurut Connolly dan Begg (2005, p291), *database design* (perancangan basis data) adalah proses pembuatan sebuah rancangan yang akan mendukung *mission statement* dan *mission objective* untuk sistem basis data yang diperlukan. Perancangan basis data dibagi menjadi tiga tahap utama yaitu *conceptual database design, logical database design,* dan *physical database design.*

2.4.5.4.1 *Conceptual Database Design*

Conceptual database design merupakan suatu proses untuk membangun sebuah model data yang digunakan dalam sebuah perusahaan, dimana proses tersebut tidak tergantung pada pertimbangan fisikal. *Conceptual database design* merupakan fase awal dari *database design*, dan meliputi pembentukan sebuah model data konseptual dari bagian perusahaan yang akan dimodelkan. Model data tersebut dibangun dengan menggunakan informasi yang didokumentasikan dalam spesifikasi kebutuhan pengguna.

2.4.5.4.2 *Logical Database Design*

Logical database design merupakan suatu proses untuk membangun suatu model data yang digunakan dalam suatu perusahaan yang berdasarkan pada model data spesifik, tapi tidak bergantung pada DBMS tertentu dan pertimbangan fisikal lainnya.

2.4.5.4.3 *Physical Database Design*

Physical database design merupakan suatu proses untuk membangun sebuah deskripsi dari implementasi basis data pada *secondary storage*, yang meliputi implemetasi fisikal dari basis data serta pengaturan hak aksesnya.

Pada tahap ini, isu penting yang juga perlu diperhatikan adalah mengenai mekanisme keamanan. Definisi dari keamanan basis data adalah suatu mekanisme yang memproteksi basis data dari suatu kejadian yang disengaja maupun tidak disengaja.

Suatu basis data merupakan sumber dari perusahaan yang penting yang perlu dilindungi dengan menggunakan suatu kontrol yang memadai.

2.4.5.5 *DBMS Selection* (langkah optional)

Menurut Connolly dan Begg (2005, p295), *DBMS selection* adalah pemilihan DBMS yang sesuai untuk mendukung sistem basis data. Langkah-langkah utama dalam pemilihan DBMS adalah :

- Mendefinisikan istilah dari acuan pembelajaran
- Menentukan beberapa produk DBMS
- Evaluasi produk
- Merekomendasikan pilihan dan menghasilkan laporan

2.4.5.6 *Application Design*

Menurut Connolly dan Begg (2005, p299), *application design* adalah perancangan *user interface* dan program aplikasi yang menggunakan dan memproses basis data. Aktivitas antara *database design* dan *application design* berjalan secara paralel dalam *database system development lifecycle*. Dalam

banyak kasus, tidaklah mungkin untuk menyelesaikan *application design* sampai perancangan dari basis data itu sendiri selesai. Di sisi lain, basis data ada untuk mendukung aplikasi dan dengan demikian pasti ada aliran informasi antara *application design* dan *database design*.

Seluruh fungsionalitas yang tercantum dalam spesifikasi kebutuhan pengguna harus ada dalam *application design* untuk sistem basis data. Perancangan *user interface* yang tepat ke dalam sistem basis data menjadi kebutuhan tambahan agar fungsionalitas yang dibutuhkan tercapai. *Interface* ini seharusnya menyajikan informasi yang diperlukan dalam cara yang *user-friendly*. Pentingnya perancangan *user interface* kadang-kadang tidak diperhatikan atau ditinggalkan selama tahapan perancangan. Akan tetapi, harus dikenali bahwa *interface* mungkin salah satu komponen yang penting dari sistem. Jika *interface* itu mudah dipelajari, mudah digunakan, bersifat langsung dan memaafkan, pengguna akan menggunakan informasi yang disajikan dengan baik.

2.4.5.7 *Prototyping* (langkah optional)

Menurut Connolly dan Begg (2005, p304), *prototyping* adalah proses membuat sebuah model kerja dari sebuah sistem basis data. Tujuan utama dalam pengembangan *prototype* adalah untuk mengijinkan pengguna menggunakan *prototype* untuk mengidentifikasi fitur-fitur sistem yang bekerja dengan baik atau yang tidak mencukupi, dan jika mungkin untuk menyarankan peningkatan atau penambahan fitur baru ke sistem basis data.

Terdapat dua macam strategi *prototyping* yang digunakan saat ini, yaitu :

- *Requirements prototyping*, menggunakan *prototype* untuk menentukan kebutuhan dari sistem basis data yang diusulkan dan ketika kebutuhan itu terpenuhi maka *prototype* akan dibuang.
- *Evolutionary prototyping*, digunakan untuk tujuan yang sama. Perbedaan yang penting adalah bahwa *prototype* tidak dibuang tetapi dengan pengembangan lebih lanjut menjadi sistem basis data yang bekerja.

2.4.5.8 *Implementation*

Menurut Connolly dan Begg (2005, p304), *implementation* adalah perwujudan fisik dari basis data dan rancangan aplikasi. Pada penyelesaian langkah-langkah perancangan (dimana dapat melibatkan pembuatan prototype atau tidak), dapat mengimplementasikan basis data dan program aplikasi. Implementasi basis data dicapai dengan menggunakan *Data Definition Language* (DDL) dari DBMS yang telah dipilih atau dengan menggunakan *Graphical User Interface* (GUI), yang menyediakan fungsionalitas yang sama sementara menyembunyikan *low-level* DDL *statement*. DDL *statement* digunakan untuk menciptakan struktur basis data dan mengosongkan *file* basis data. *User view* juga diimplementasikan pada langkah ini.

2.4.5.9 *Data Conversion and Loading*

Menurut Connolly dan Begg (2005, p305), *data conversion and loading* berarti memindahkan semua data yang ada ke dalam basis data yang baru dan mengkonversi semua aplikasi yang ada untuk dijalankan pada basis data yang baru. Langkah ini dibutuhkan hanya jika sistem basis data yang baru menggantikan sistem yang lama. Sekarang ini, sudah menjadi hal yang umum bagi sebuah DBMS untuk mempunyai *utility* yang dapat memuat keseluruhan *file* yang ada ke dalam basis data yang baru. *Utility* ini biasanya membutuhkan spesifikasi dari *file* sumber dan basis data yang dituju, dan kemudian mengkonversi data ke format yang sesuai dengan *file* basis data yang baru.

2.4.5.10 *Testing*

Menurut Connolly dan Begg (2005, p305), *testing* adalah suatu proses menjalankan sistem basis data dengan tujuan untuk menemukan kesalahan (*error*). Pengujian ini dilakukan dengan menggunakan strategi pengujian yang direncanakan dengan hati-hati dan menggunakan data yang realistis sehingga keseluruhan proses pengujian terselesaikan sesuai dengan metodenya. Pengujian mendemonstrasikan bahwa basis data dan program aplikasi kelihatan bekerja berdasarkan spesifikasinya dan bahwa kebutuhan unjuk kerja dapat dipenuhi. Sebagai tambahan, metrik yang dikumpulkan dari langkah pengujian menyediakan suatu ukuran reliabilitas dan kualitas *software*. Perlu diperhatikan juga bahwa pengguna sistem yang baru harus dilibatkan dalam proses pengujian.

2.4.5.11 *Operational Maintenance*

Menurut Connolly dan Begg (2005, p306), *operational maintenance* adalah proses memonitor dan memelihara sistem basis data setelah instalasi. Langkah ini meliputi kegiatan-kegiatan berikut :

- Memonitor unjuk kerja sistem. Jika unjuk kerja jatuh di bawah tingkat yang tidak dapat diterima, perbaikan atau reorganisasi basis data diperlukan.
- Memelihara dan meningkatkan mutu sistem basis data (jika dibutuhkan). Kebutuhan baru digabungkan ke dalam sistem basis data melalui langkah-langkah siklus hidup yang sebelumnya.

2.4.6 *Web Database*

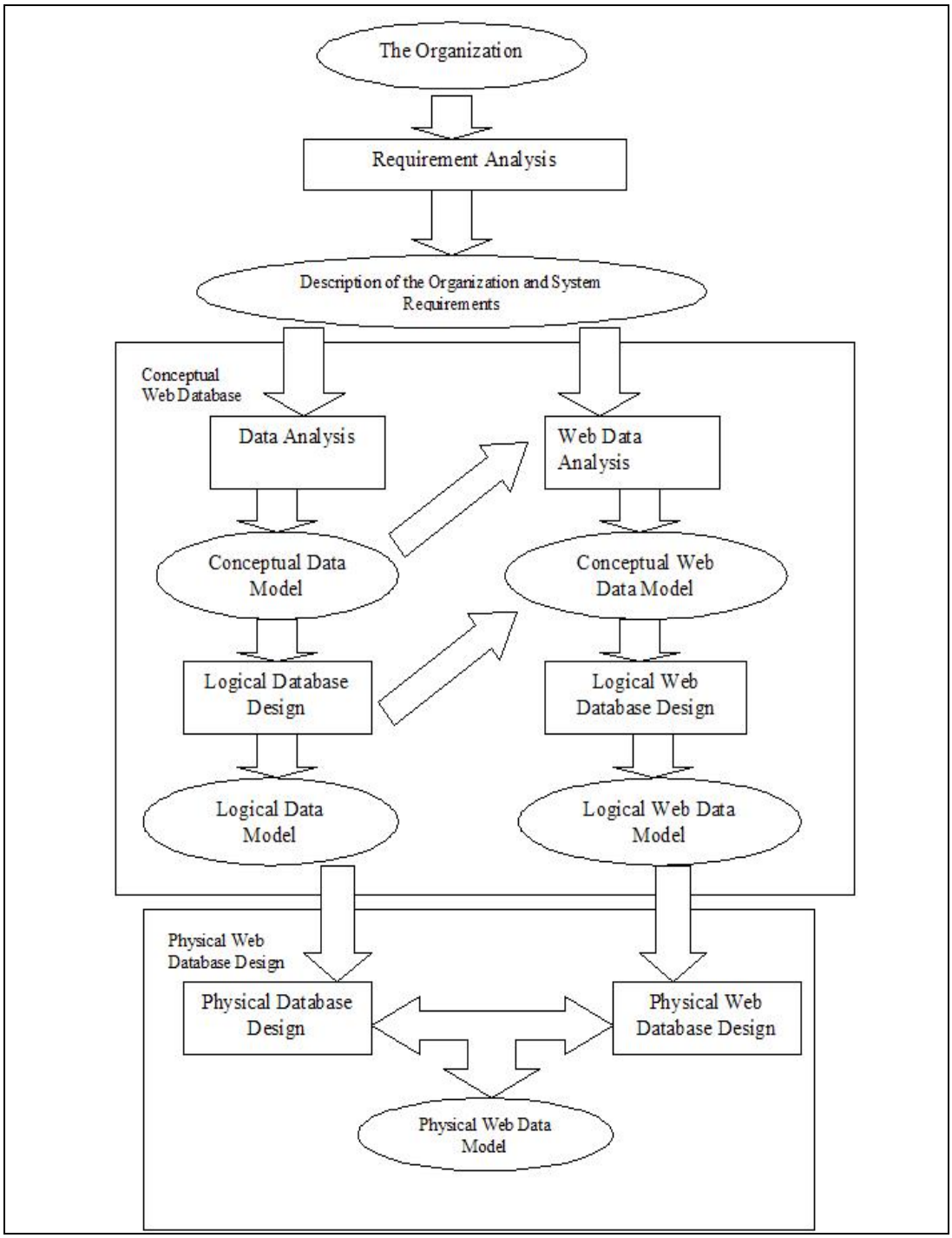
2.4.6.1 Pengertian *Web Database*

Web Database dapat diartikan sebagai berikut :

- Tempat penyimpanan (*respositories*) basis data atau informasi yang secara dinamis berinteraksi dengan halaman *web*.
- Sebuah metode untuk menyimpan *content* WWW, dalam format terstruktur atau *usable*, yang terhubung baik secara statis maupun dinamis dengan basis data lain.
- Membantu komunikasi antara *web server* dan basis data, serta memungkinkan pengguna menerbitkan atau mengumpulkan informasi dari manapun.

2.4.6.2 *Web Database Design*

Adapun langkah-langkah dalam *web database design* digambarkan sebagai berikut :



Gambar 2.9 *Web Database Design*

(Sumber Eaglestone dan Ridley, 2001, p264)

Pembahasan mengenai *web database lifecycle* hanya mengenai pemodelan konseptual halaman *web* dan *logical web page schema* karena secara garis besar, proses perancangan pada *web database* serupa dengan proses perancangan pada basis data relasional.

2.4.6.2.1 Model Konseptual Halaman *Web*

Perancangan *data content* pada halaman *web* sebenarnya mirip dengan perancangan basis data relasional yang telah diuraikan sebelumnya. Akan tetapi, pada tahap ini, ada beberapa hal yang ditambahkan pada diagram ER yang telah dihasilkan. Menurut Eaglestone dan Ridley (2001, p285), ada dua aspek dari halaman *web* yang dibutuhkan ketika memperluas suatu diagram ER untuk pemodelan data *web*, yaitu :

1. *Hypermedia link*, yang melambangkan jalur navigasi antara *entity* yang berhubungan. *Link* ini berfungsi seperti *relationship* pada diagram ER, yang membedakannya adalah garis *relationship*-nya harus berupa anak panah yang bolak-balik.
2. *Web application-specific concept*, yang meliputi titik akses (*entry point*) ke halaman-halaman *web*. Dapat juga diartikan sebagai *homepage* dari aplikasi *web* tersebut. Biasanya menggunakan simbol berbentuk belah ketupat, yang disebut *concept box*. *Concept box* ini akan memiliki arah panah ke halaman *web* yang dapat diakses dari *homepage* tersebut.

2.4.6.2.2 *Logical Web Page Schema*

Tahap ini mengambil *input* dari *web conceptual model* dan mendefinisikan sebuah *schema* untuk masing-masing halaman *web*. Berikut ini beberapa ketentuan dalam pendefinisian *web page schema* (Eaglestone dan Ridley, 2001, p312) :

- Setiap *web page schema* selalu diawali dengan *keyword* "PAGE-SCHEMA".
- Nama halaman *web*.
- Item data yang berulang, yang diindikasikan dengan *keyword* "LIST-OF".
- Tipe data, seperti STRING, INTEGER, dan sebagainya.

2.4.7 *Database Security*

Data merupakan sumber daya bernilai yang harus diatur dan diawasi secara ketat bersama dengan sumber daya perusahaan (*corporate resources*). Sebagian atau keseluruhan data perusahaan mempunyai kepentingan strategis dan karena itu harus dijaga agar tetap aman dan rahasia. Pertimbangan keamanan tidak hanya diaplikasikan pada data yang ada dalam basis data. Menurut Connolly dan Begg (2005, p542), *database security* adalah mekanisme yang melindungi basis data dari serangan atau ancaman yang disengaja maupun tidak disengaja. Pelanggaran terhadap keamanan dapat mempengaruhi bagian lain dari sistem, yang akan memberi akibat balik terhadap basis data. Kemananan basis data terkait dengan keadaan berikut :

- Pencurian data (*Theft and Fraud*)
- Kehilangan kerahasiaan suatu data (*Loss of Confindentially*)

- Kehilangan hak pribadi (*Loss of Privacy)*
- Kehilangan Integritas (*Loss Integrity*)
- Kehilangan ketersediaan data (*Loss of Avaliability*)

Secara umum, DBMS relasional menyediakan dua tipe keamanan basis data, yaitu :

- Keamanan sistem, melindungi pengaksesan dan penggunaan basis data pada tingkat sistem, seperti *username* dan *password*.
- Keamanan data, melindungi pengaksesan dan penggunaan obyek basis data (seperti relasi dan *view*) dan tindakan yang dapat dilakukan oleh pengguna.

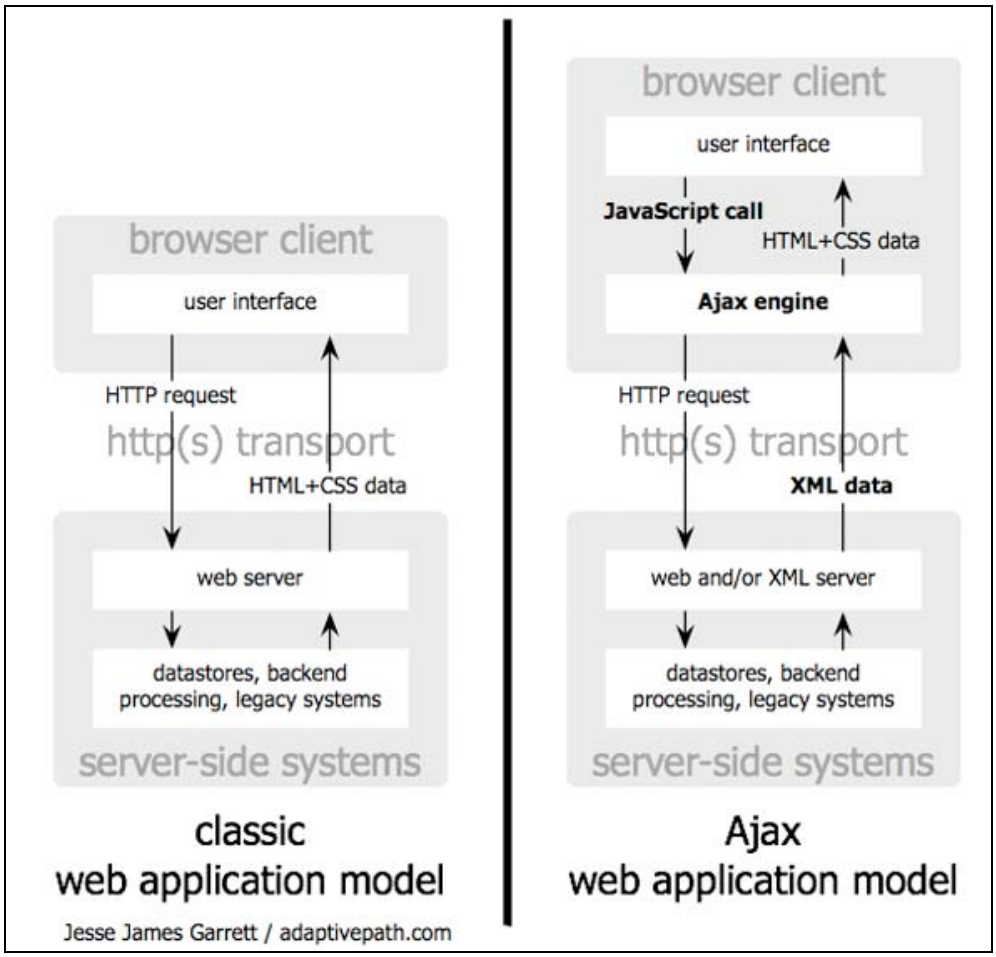
2.5 Teori AJAX

Asynchronous JavaScript and XMLHTTP, atau disingkat AJAX, adalah suatu teknik pemrograman berbasis web untuk menciptakan aplikasi *web* interaktif. Tujuannya adalah untuk memindahkan sebagian besar interaksi pada komputer *web server*, melakukan pertukaran data dengan *server* di belakang layar, sehingga halaman *web* tidak harus dibaca ulang secara keseluruhan setiap kali seorang pengguna melakukan perubahan. Hal ini akan meningkatkan interaktivitas, kecepatan, dan *usability*. Ajax merupakan kombinasi dari:

- DOM yang diakses dengan *client side scripting language*, seperti VBScript dan implementasi ECMAScript seperti JavaScript dan JScript, untuk menampilkan secara dinamis dan berinteraksi dengan informasi yang ditampilkan

- Objek XMLHTTP dari Microsoft atau XMLHttpRequest yang lebih umum di implementasikan pada beberapa *browser*. Obyek ini berguna sebagai kendaraan pertukaran data *asynchronous* dengan *web server*. Pada beberapa *framework* AJAX, elemen HTML IFrame lebih dipilih daripada XMLHTTP atau XMLHttpRequest untuk melakukan pertukaran data dengan *web server*.
- XML umumnya digunakan sebagai dokumen *transfer*, walaupun format lain juga memungkinkan, seperti HTML, plain text. XML dianjurkan dalam pemakaian teknik AJAX karena kemudahan akses penanganannya dengan memakai DOM.
- JSON dapat menjadi pilihan alternatif sebagai dokumen *transfer*, mengingat JSON adalah JavaScript itu sendiri sehingga penanganannya lebih mudah.

Seperti halnya DHTML, LAMP, atau SPA, Ajax bukanlah teknologi spesifik, melainkan merupakan gabungan dari teknologi yang dipakai bersamaan.



Gambar 2.10 Perbandingan antara model aplikasi web lama (kiri) dengan model AJAX (kanan)

(Sumber http://ww.adaptivepath.com /ideas/essays/archieves/000385.php)