BAB 2

TINJAUAN PUSTAKA

2.1 Teori yang berkaitan dengan *database*

2.1.1 Sistem

Suatu sistem sangatlah dibutuhkan dalam suatu perusahaan atau instansi pemerintahan, karena sistem sangat menunjang terhadap kinerja perusahaan atau instansi pemerintah, baik yang berskala kecil maupun besar.Agar dapat berjalan dengan baik diperlukan kerjasama antara pihak-pihak yang terkait dalam sistem tersebut. Menurut HM (2005 : 1) ada berbagai pendapat yang mendefinisikan pengertian sistem diantaranya, sistem adalah suatu jaringan kerja dari prosedur-prosedur yang saling berhubungan, berkumpul bersama-sama untuk melakukan suatu kegiatan atau untuk menyelesaikan suatu sasaran yang tertentu. Pendapat lain mengatakan sistem adalah kumpulan dari elemen-elemen yang berinteraksi untuk mencapai suatu tujuan tertentu. Menurut HM (2005 : 195-197) ada beberapa definisi dari perancangan sistem diantaranya :

1. Pendefinisian dari kebutuhan–kebutuhan fungsional.
2. Penggambaran bagaimana suatu sistem dibentuk.
3. Penggambaran, perencanaan dan pembuatan sketsa atau pengaturan dari beberapa elemen yang terpisah ke dalam suatu kesatuan yang utuh dan berfungsi.

Perancangan sistem dapat dibagi dalam dua bagian, yaitu:

1. Perancangan sistem secara umum atau perancangan konseptual, perancangan *logical* atau perancangan secara makro.
2. Perancangan sistem terinci atau perancangan sistem secara fisik.

Tujuan utama perancangan sistem mempunyai dua maksud dan tujuan utama, yaitu :

1. Untuk memenuhi kebutuhan kepada pengguna sistem
2. Untuk memberikan gambaran yang jelas dan rancang bangun yang lengkap kepada *programmer* dan ahli-ahli teknik yang terlibat dalam perancangan sistem.

Untuk mencapai dua tujuan utama dari perancangan sistem, analisis sistem harus memenuhi sasaran-sasaran berikut ini :

1. Perancangan sistem harus berguna, mudah dipahami dan nantinya mudah digunakan. Ini berarti data harus mudah ditangkap, metode-metode harus

mudah diterapkan dan informasi harus mudah dihasilkan serta mudah dipahami dan digunakan.

2. Perancangan sistem harus dapat mendukung tujuan utama perusahaan sesuai dengan yang telah didefinisikan pada tahap perencanaan sistem yang dilanjutkan pada tahap analisis sistem.
3. Perancangan sistem harus efisien dan efektif untuk dapat mendukung pengolahan transaksi, pelaporan manajemen dan mendukung keputusan yang akan dilakukan oleh manajemen, termasuk tugas-tugas lainnya yang tidak dilakukan oleh *computer*.
4. Perancangan sistem harus dapat mempersiapkan rancangan yang terperinci untuk masing-masing komponen dari sistem informasi yang meliputi data dan informasi, data yang tersimpan, metode-metode, prosedur-prosedur, *software* dan *hardware*.

2.1.2 Aplikasi

Aplikasi merupakan salah satu dari komponen sistem informasi. Menurut Shelly & Vermaat (2011 : 108) aplikasi adalah seperangkat instruksi khusus dalam komputer yang dirancang agar kita dapat menyelesaikan tugas-tugas tertentu dengan lebih produktif.

2.1.3 Informasi

MenurutHM (2005 : 8-11), informasi diartikan sebagai data yang diolah menjadi bentuk yang lebih berguna dan lebih berarti bagi yang menerimanya. Nilai dari informasi ditentukan dari dua hal, yaitu manfaat dan biaya untuk mendapatkannya. Suatu informasi dikatakan bernilai jika manfaatnya lebih efektif dibandingkan dengan biaya mendapatkannya. Sebagian besar informasi tidak dapat persis ditaksir keuntungannya dengan nilai uang, tetapi dapat ditaksir nilai efektivitasnya. Kualitas informasi tergantung dari tiga hal, yaitu :

1. Akurat

   Informasi harus bebas dari kesalahan dan keambiguan. Informasi harus akurat karena dari sumber informasi ke penerima informasi kemungkinan banyak terjadi gangguan (*noise*) yang dapat merubah atau merusak informasi tersebut.

2. Tepat pada waktunya

Informasi yang datang pada penerima tidak boleh terlambat. Informasi yang sudah usang tidak akan memiliki nilai lagi, karena informasi merupakan landasan di dalam pengambilan keputusan. Bila pengambilan keputusan terlambat, maka dapat berakibat fatal untuk organisasi. Saat ini mahalnya nilai informasi disebabkan informasi harus secara cepat didapatkan, sehingga perlu teknologi canggih untuk mendapatkan, mengolah dan mengirimkannya.

3. Relevan

Informasi tersebut mempunyai manfaat bagi pemakainya. Relevansi informasi untuk tiap-tiap orang satu dengan yang lainnya berbeda. Misalya, informasi mengenai sebab-akibat kerusakan mesin produksi kepada perusahaan akuntan kurang relevan dan akan lebih relevan bila ditunjukkan kepada ahli teknik perusahaan. Sebaliknya informasi mengenai harga pokok produksi untuk ahli teknik merupakan informasi yang kurang relevan, tetapi relevan untuk akuntan.

2.1.4 Sistem Informasi

Sistem informasi adalah suatu sistem di dalam suatu organisasi yang mempertemukan kebutuhan pengolahan transaksi harian, mendukung operasi, bersifat manajerial dan kegiatan strategi dari suatu organisasi dan menyediakan pihak luar tertentu dengan laporan - laporan yang diperlukan HM (2005 : 11). Menurut O`Brien, James A (2005 : 5), “sistem informasi dapat merupakan kombinasi teratur apa pun dari orang-orang, *hardware*, *software*, jaringan komunikasi, dan sumber daya data yang mengumpulkan, mengubah, dan menyebarkan informasi dalam sebuah organisasi”. Orang bergantung pada sistem informasi untuk berkomunikasi antara satu sama lain dengan menggunakan berbagai jenis *hardware, software*, *network* dan sumber daya data.

2.1.5 Data

Data merupakan catatan atas kumpulan fakta yang dicatat dan disimpan oleh perusahaan untuk diolah sehingga dapat menghasilkan suatu informasi yang berguna bagi suatu organisasi. Data yang dikumpulkan di dalam *database* akan digunakan oleh pengguna akhir untuk mendapatkan informasi.

Seperti pendapat yang diungkapkan oleh Laudon & Laudon (2007 : 375), data merupakan kumpulan fakta – fakta kasar yang menunjukkan kejadian yang terjadi dalam organisasi atau lingkungan fisik sebelum fakta tersebut diolah dan ditata menjadi bentuk yang dapat dipahami.

2.1.6 *Database*

Seperti yang disampaikan oleh Connolly & BEGG (2010 : 65) bahwa "*database* adalah kumpulan data yang saling terhubung secara logis dan deskripsi dari data tersebut dirancang untuk menemukan informasi yang dibutuhkan oleh sebuah organisasi". Dalam merancang *database*, salah satu hal yang perlu diperhatikan adalah efisiensi. Banyaknya data yang berulang dapat mengurangi efisiensi pada *database* sehingga perlu dilakukan normalisasi. *Database* ini digunakan tidak hanya oleh satu orang maupun satu departemen, *database* dapat digunakan oleh seluruh departemen dalam perusahaan. *Database* akan menjadi sumber data yang digunakan secara bersama dalam perusahaan. Hal ini kembali ditegaskan oleh Connolly & BEGG (2010 : 65) bahwa *database* tidak lagi dimiliki oleh satu departemen tetapi sumber perusahaan yang saling berbagi. Untuk mendapatkan *database* sebaik mungkin, dengan hanya *database* saja tidak cukup, diperlukan *database management system* (DBMS) untuk dapat menggunakan *database*.

2.1.7 *Database Management System*

Database management system merupakan *software* yang membantu pengguna untuk mengakses *database*. Dengan DBMS, seseorang dapat melakukan interaksi dengan *database*, misalnya untuk menambah, mengubah, menghapus dan melihat data di dalam *database*. Hal ini juga dikemukakan oleh Connolly & BEGG (2010 : 66), DBMS merupakan sebuah sistem *software* yang memungkinkan pengguna untuk menjelaskan, membuat, memelihara dan mengontrol akses dalam *database*.

2.1.7.1 Fasilitas DBMS

Dari penerapan sebuah DBMS, tentunya akan mendapatkan fasilitas yang akan menguntungkan perusahaan yang menerapkan DBMS itu sendiri. Diantaranya fasilitas yang akan di dapatkan seperti

yang dikutip dari Connolly & BEGG (2010 : 66) fasilitas yang disediakan oleh DBMS antara lain :

1. *Data definition language* (DDL)

   Memungkinkan pengguna untuk menspesifikasikan tipe dan struktur data, serta batasan pada data yang disimpan dalam *database*.

2. *Data manipulation language* (DML)

   Memungkinkan pengguna untuk *insert, update*, *delete* dan *view* data dari *database*.

3. Kontrol akses ke basis data, antara lain :

   a. *Security*, melindungi basis data dari perusakan dan mencegah pengaksesan basis data oleh pengguna yang tidak berwenang.

   b. *Integrity*, mengatur mengenai konsistensi dari data yang disimpan.

   c. *Concurency control*, mengatur pengaksesan basis data dengan multipenggunasecara bersama-sama, tanpa menggangu operasi masing-masing.

   d. *Recovery control system*, memperbaiki basis data ke bentuk yang dianggap benar setelah terjadinya suatu kegagalan perangkat keras ataupun perangkat lunak.

   e. Pengguna *accessible catalog, catalog* yang memberikan gambaran dari data yang terdapat pada *database*.

2.1.7.2 Komponen dari Lingkungan *DBMS*

Untuk dapat membuat dan menjalankan DBMS dengan baik, maka diperlukan komponen-komponen DBMS. Menurut Connolly & BEGG (2010 : 68-71) kita dapat mengidentifikasi 5 komponen utama dalam lingkungan DBMS yaitu :

1. *Hardware*

   Untuk menjalankan DBMS dan aplikasi diperlukan *hardware*. *Hardware* yang diperlukan tergantung pada kebutuhan organisasi. Beberapa DBMS bekerja hanya pada *hardware* atau *operating system* tertentu, sementara yang lain berjalan pada *hardware* dan *operating system* yang beragam.

2. *Software*

   Untuk menjalankan DBMS, tidak dapat hanya dengan *hardware* namun diperlukan *software* dan aplikasi program untuk bekerja dengan *operating system* agar DBMS dapat dijalankan. Biasanya, aplikasi program ditulis dalam *third – generation language* (3GL), seperti C, C++, C#, Java, Visual Basic, COBOL, Fortran, Ada, Pascal, atau *forth – generation language* seperti SQL.

3. Data

   Salah satu komponen yang paling penting dalam DBMS adalah data. Data adalah apa yang akan dilihat oleh pengguna akhir nantinya. Data menjadi jembatan antara komponen manusia dan komponen mesin.

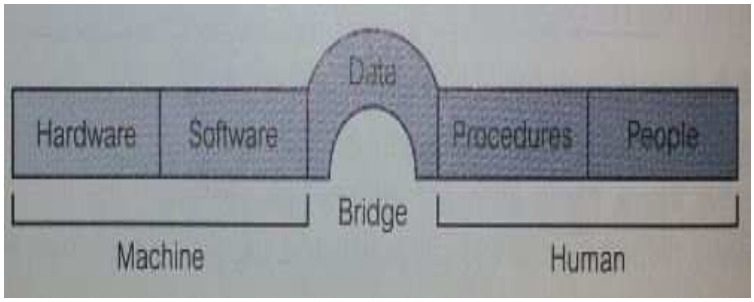
4. Prosedur

   Prosedur mengacu pada instruksi dan aturan yang mengatur desain dan penggunaan *database*. Pengguna dari sistem dan anggota yang mengelola membutuhkan dokumentasi prosedur tentang cara menjalankan dan menggunakan sistem. Langkah-langkah dalam menggunakan sistem :

   a. Masuk ke dalam DBMS
   b. Gunakan fasilitas DBMS atau program aplikasi
   c. Mulai dan berhenti DBMS
   d. Membuat cadangan *database*
   e. Menangani perangkat lunak atau perangkat lunak yang rusak (meliputi prosedur bagaimana mengidentifikasi komponen yang rusak, cara mengatasi)
   f. Mengubah struktur tabel, mengelola *database* silang, menambah performa, membuat data cadangan.

5. Manusia

   Komponen terakhir yang terlibat dengan sistem adalah manusia. Komponen manusia ini terbagi lagi berdasarkan peran mereka dalam lingkungan *database*.



Gambar 2.1 Komponen *Database*
Sumber :(Connolly & BEGG, 2010, p. 68)

2.1.7.3 Peran dalam Lingkungan *Database*

Dalam menjalankan DBMS, manusia sebagai salah satu komponen adalah yang memiliki peran untuk merancang dan menggunakan DBMS. Berdasarkan peran–peran tersebut, komponen manusia dapat dibagi. Hal ini juga dikatakan oleh Connolly & BEGG (2010 : 71-73) kita dapat mengidentifikasi 4 tipe dari orang yangberpartisipasi dalam lingkungan DBMS :

1. Data dan *database administrators*

   Peran yang secara umumnya terhubung dengan pengaturan dan pengontrolan dari DBMS dan datanya. *Data administrator* (DA) bertanggung jawab untuk mengatur sumber data, termasuk perencanaan *database,* standarisasi pengembangan dan perawatan, kebijakan dan prosedur, desain *database conceptual/logical. Database administrator* (DBA) bertanggung jawab atas terwujudnya fisik rancangan *database,* termasuk rancangan fisik *database* dan implementasi, keamanan dan pengontrolan integritas, perawatan sistem operasional, dan memastikan kinerja yang memuaskan dari aplikasi untuk pengguna.

2. *Database designers*

   Kita dapat membedakan *database designers* menjadi dua tipe, yaitu *logical database designer dan physical database designer. Logical database designer* berhubungan dengan pengidentifikasian data (*entities* dan *attributes*), relasi antara data dan batasan pada data yang disimpan dalam *database*. *Logical database designer* harus memliki pemahaman yang mendalam dan lengkap dari data

organisasi dan batasan-batasan pada data (batasan sering disebut dengan *business rules*). *Physical database designer* memutuskan bagaimana rancangan *logical database* akan diwujudkan secara fisik.

3. *Application developers*

   Bertanggung jawab untuk memastikan program aplikasi dari *database* yang telah diwujudkan menyediakan kebutuhan fungsional untuk pengguna akhir.

4. *End users*

   End users adalah *clients* dari *database* yang telah dirancang dan diwujudkan untuk menyediakan informasi yang dibutuhkan. *End user* dapat diklasifikasikan berdasarkan cara penggunaan sistem, yaitu *naive users* dan *sophisticated users. Naive users* adalah *users* dengan tipe yang tidak menyadari DBMS, mereka mengakses *database* berdasarkan program aplikasi yang tertulis dan berusaha untuk membuat operasi sesederhana mungkin. Mereka menjalankan operasi *database* dengan memasukkan perintah sederhana atau memilih pilihan dari menu, artinya mereka tidak perlu tahu tentang *database* atau DBMS. Berbeda dengan *naïveusers*, *sophisticated users* mengenal struktur *database* dan fasilitas yang disediakan oleh DBMS. *Sophisticated users* dapat menggunakan bahasa *query* tingkat tinggi seperti SQL untuk menampilkan operasi yang dibutuhkan.

2.1.7.4 Keuntungan dan Kekurangan DBMS

DBMS menjanjikan banyak keuntungan yang dapat digunakan, namun tentunya DBMS juga memiliki kekurangan yang perlu diperhatikan. Pertama-tama kita perlu melihat keuntungan apa saja yang diberikan oleh DBMS. Menurut Connolly & BEGG (2010 : 77-80) keuntungan dalam menggunakan DBMS ialah :

1. Mengendalikan pengulangan data

   Database merupakan salah satu cara untuk mengontrol pengulangan data, dengan banyaknya data yang ada, maka kemungkinan untuk data yang sama saling berulang sangatlah

besar, oleh karena itu dibutuhkan *database* dalam mengendalikan pengulangan data dengan cara mengintegrasikan *file* sehingga berbagai data yang sama tidak akan disimpan dalam *database*, akan tetapi hal ini tidak menghilangkan semua pengulangan data yang ada di *database* secara menyeluruh, melainkan hanya membuat jumlah pengulangan data dapat dikontrol.

2. Konsistensi data

   Dengan mengontrol perulangan data, dapat mengurangi resiko terjadinya ketidakkonsistennan yang terjadi. Jika data yang disimpan hanya sekali dalam *database*, maka berbagai perubahan bagi nilai data tersebut juga hanya dilakukan satu kali, dan nilai tersebut harus tersedia kepada semua pengguna. Jika *item* data yang disimpan lebih dari sekali dan sistem menyadari hal ini, sistem ini dapat memastikan bahwa semua salinan *item* disimpan secara konsisten.

3. Banyak informasi yang didapat dari jumlah data yang sama

   Dengan mengintegrasikan data operasional, dapat memungkinkan perusahaan untuk memperoleh informasi tambahan dari data yang sama.

4. *Sharing of data*

   Files dimiliki oleh orang atau perusahaan yang menggunakannya. *Database* merupakan bagian dari keseluruhan organisasi dan dapat dibagikan ke semua pengguna yang mempunyai wewenang.

5. Meningkatkan integritas data

   Integritas data mengacu pada validitas dan konsistensi data yang disimpan. Integritas biasanya dinyatakan dalam istilah *constraints* (batasan), yang berupa aturan konsisten yang tidak boleh dilanggar oleh *database. Constraints* dapat diterapkan ke data *items* dengan *record* tunggal atau hubungan dengan antar *record.* Integrasi memungkinkan DBA untuk menjelaskan, dan memungkinkan DBMS untuk membuat batasan integritas.

6. Meningkatkan keamanan

   Meningkatkan keamanan data dimana memproteksi *database* dari *user* yang tidak memiliki wewenang. Biasanya keamanan yang

dipakai dengan cara mengambil *username* dan *password* untuk mengidentifikasi orang yang memiliki wewenang untuk menggunakan *database*.

7. Penerapan standarisasi

   Integrasi memungkinkan DBA untuk mendefinisikan DBMS dan membuat *standard* yang diperlukan. Standarisasi ini termasuk standarisasi departemen, organisasi, nasional, dan internasional dalam hal format data, dan berguna untuk memfasilitasi pertukaran data antara sistem, ketepatan, penamaan standarisasi dokumentasi, prosedur *update*, dan aturan pengaksesan.

8. Pengurangan biaya

   Dimana semua data operasional organisasi dipusatkan ke dalam satu *database* dan membuat sekumpulan aplikasi bekerja pada satu sumber data sehingga dapat menghasilkan pengurangan biaya. Jadi dengan penyatuan biaya untuk pengembangan dan pemeliharaan sistem pada setiap departemen akan menghasilkan total biaya yang dikeluarkan akan lebih rendah. Sehingga sisa biaya yang merupakan penghematan sebelumnya dapat digunakan untuk hal lain yang dapat meningkatkan performa bagi kebutuhan organisasi.

9. Menyeimbangkan konflik kebutuhan

   Setiap pengguna atau departemen mempunyai kebutuhan yang mungkin berbeda dengan kebutuhan pengguna lain. Oleh karena itu *database* dikendalikan oleh DBA (*database administrator),* DBA dapat membuat keputusan berkaitan dengan perancangan dan penggunaan operasional *database* yang menyediakan penggunaan terbaik dari sumber daya bagi keseluruhan organisasi.

10. Meningkatkan kemampuan pengaksesan dan respon pada data

    Dengan mengintegrasikan data yang melintasi batasan departemen dapat langsung diakses oleh pengguna akhir, hal ini dapat menyediakan sebuah sistem dengan lebih banyak fungsi.

11. Meningkatkan produktivitas

    DBMS menyediakan banyak fungsi *standard* yang biasanya ditulis oleh *programmer* dalam aplikasi berbasis *file*. Pada tingkat dasar, DBMS juga menyediakan semua rutinitas penanganan *file* tingkat

rendah yang khas dalam program aplikasi. Penyediaan fungsi–fungsi ini memungkinkan seorang *programmer* untuk berkonsentrasi pada fungsi tertentu yang dibutuhkan oleh seorang pengguna tanpa perlu khawatir tentang rincian implementasi tingkat rendah. DBMS banyak juga menyediakan lingkungan generasi keempat, yang digunakan untuk menyederhanakan pengembangan aplikasi *database*. Hal ini menyebabkan produktivitas pemmrograman meningkat dan waktu pengembangan berkurang (dengan penghematan biaya yang terkait).

12. Meningkatkan pemeliharaan melalui kemandirian data

    Dalam sistem berbasis *file*, deskripsi data dan *logical* untuk mengakses data yang dibangun ke dalam setiap program aplikasi, membuat program bergantung pada data. Perubahan struktur data, seperti membuat sebuah alamat menjadi 41 karakter dari 40 karakter atau perubahan cara penyimpanan data pada *disk* dapat memerlukan perubahan besar untuk program yang dipengaruhi oleh perubahannya. Sebaliknya, DBMS memisahkan deskripsi data dari aplikasi, sehingga membuat aplikasi tidak terganggu oleh perubahan dalam deskripsi data, dimana dikenal sebagai data *independence.*

13. Meningkatkan konkurensi

    Dalam sistem yang berbasis *file*, jika dua atau lebih *user* diperbolehkan mengakses *file* yang sama secara bersama-sama, memungkinkan akses akan terganggu satu sama lain, akibatnya kehilangan informasi dan integritas.

14. Meningkatkan pelayanan *backup* dan *services*

    Banyak sistem berbasis *file* menempatkan tanggung jawab kepada *user* dalam memberi tindakan untuk melindungi data dari kegagalan sistem komputer atau program aplikasi. *Backup* harus sering dilakukan seiring dengan proses datanya dan apabila selama pekerjaan telah terjadi kegagalan maka *backup* dipergunakan. DBMS modern memberikan fasilitas untuk meminimalkan jumlah pengolahan yang hilang setelah terjadi kegagalan.

Setelah melihat keuntungan yang dimiliki DBMS, kita perlu melihat kerukurangan yang dimiliki oleh DBMS untuk dipertimbangkan. Menurut Connolly & BEGG (2010:80-81) kerugian dalam menggunakan DBMS ialah :

1. Kompleksitas

   Pemberian dari fungsi yang diharapkan dari DBMS yang baik membuat DBMS menjadi sebuah *software* yang sangat kompleks. *Database designers* dan *developers*, *data administrators* dan *database administrators*, serta *end users* harus memahami fungsi tersebut untuk mendapatkan banyak keuntungan dari DBMS tersebut. Kesalahan dalam memahami sistem dapat membawa ke keputusan desain yang buruk, dimana dapat berdampak serius bagi sebuah organisasi.

2. Ukuran data

   Fungsi yang kompleks dan luas membuat DBMS menjadi *software* yang sangat besar, sehingga memerlukan banyak ruang *harddisk* dan jumlah memori yang digunakan menjadi sangat besar untuk berjalan dengan efisien.

3. Biaya dari DBMS

   Biaya DBMS bervariasi, tergantung pada lingkungan dan fungsi yang disediakan. Tedapat juga biaya pemeliharaan tahunan yang juga dimasukan dalam daftar harga DBMS.

4. Biaya penambahan perangkat keras

   Kebutuhan tempat penyimpanan bagi DBMS dan *database* sangat membutuhkan pembelian tempat penyimpanan tambahan lebih lanjut untuk mencapai performa yang diperlukan, dan akan membutuhkan spesifikasi perangkat keras yang lebih canggih dan sebagainya yang memerlukan biaya yang tidak sedikit. Tergantung pada spesifikasi perangkat keras yang diperlukan.

5. Biaya konversi

   Di dalam situasi tertentu, biaya DBMS dan *hardware* tambahan mungkin relatif kecil dibandingkan dengan biaya mengkonversi aplikasi yang ada untuk berjalan di DBMS baru dan perangkat

keras. Biaya ini juga termasuk biaya pelatihan karyawan untuk menggunakan sistem baru, dan mungkin karyawan spesialis untuk membantu mengkonversi dan menjalankan sistem. Biaya ini merupakan salah satu alasan utama mengapa beberapa organisasi merasa terikat pada sistem mereka saat ini dan tidak bisa beralih ke teknologi *database* yang lebih modern.

6. Kinerja

   Sistem berbasis *file* ditulis untuk aplikasi yang khusus, seperti *invoice*. Secara umum kinerjanya sangat baik. Namun, DBMS ditulis lebih umum, untuk melayani banyak aplikasi bukan hanya satu. Hasilnya adalah bahwa beberapa aplikasi mungkin berjalan secepat dulu.

7. Dampak kegagalan lebih besar

   Pemusatan sumber daya akan meningkatkan kerentanan sistem. Karena semua pengguna dan aplikasi bergantung pada ketersediaan DBMS, sehingga apabila terjadi kegagalan komponen tertentu dapat mengakibatkan operasi berhenti.

2.1.7.5 Fungsi DBMS

Ketika menggunakan DBMS, kita tentunya mengharapkan fungsi dari DBMS yang dapat kita gunakan DBMS menyediakan beberapa tipe fungsi dan layanan yang dapat kita gunakan. Seperti yang dikatakan oleh Connolly & BEGG (2010 : 99-104) fungsi daripada DBMS antara lain :

1. Menyimpan, menampilkan, dan mengubah data

   Sebuah DBMS diharuskan memiliki kemampuan untuk menyimpan, menampilkan dan mengubah data di dalam basis data. DBMS akan menyembunyikan rincian dari implementasi *physical internal* DBMS (seperti *file* organisasi dan struktur penyimpanan) dari pengguna lainnya.

2. Sebuah *user accessible catalog*

   Dalam DBMS harus menyediakan katalog yang mana berisi deskripsi dari data yang disimpan di dalamnya dan informasi mengenai siapa saja yang diberi hak akses untuk mengakses data di

dalamnya. Pada dasarnya yang disimpan dalam sistem katalog terdiri dari :

a. Nama, tipe dan ukuran data
b. Nama *relationship*
c. Batasan integritas dalam data
d. Nama pengguna yang memiliki hak akses ke basis data
e. Data yang dapat diakses pengguna dan tipe hak akses dan tipe akses yang diizinkan (seperti *insert*, *update*, *delete* atau *read*)
f. Skema eksternal, konseptual, internal serta pemetaan antar skema
g. Statistik pemakaian, seperti frekuensi transaksi dan jumlah akses ke objek di *database*

3. Mendukung transaksi

   Sebuah DBMS harus menyediakan sebuah mekanisme yang dapat melakukan perubahan terhadap transaksi yang sudah terjadi untuk *user*.

4. Layanan kontrol konkurensi

   DBMS harus dapat menyediakan mekanisme untuk memastikan bahwa *database* di ubah secara benar ketika banyak *user* sedang perubahan *database* secara bersamaan.

5. Layanan perbaikan

   Dalam sebuah DBMS harus ada sebuah mekanisme untuk memperbaiki basis data yang bermasalah dengan menggunakan cara apa pun.

6. Layanan otorisasi

   DBMS harus dapat menyediakan validasi untuk memastikan bahwa *user* yang hendak mengakses DBMS adalah *user* yang sudah memiliki otoritas.

7. Mendukung komunikasi data

   Sebuah DBMS harus dapat menghubungkan antar perangkat lunak dimana digunakan untuk komunikasi data.

8. Layanan integritas

Sebuah DBMS harus dapat menyediakan cara untuk memastikan data dalam basis data dan perubahan data tersebut dapat mengikuti peraturan yang telah diatur sebelumnya.

9. Memberikan kemampuan independensi data

   Pada DBMS harus memiliki fasilitas dimana mendukung kemandirian program dari struktur *database* yang sebenarnya.

10. Layanan utilitas

    DBMS menyediakan sekumpulan layanan utilitas, seperti :

    a. Fasilitas *import*

       Memuat *database* dari *flat files,* dan fasilitas *export* untuk memindahkan *database* ke *flat files*. *Flat files* adalah tabel yang tidak memiliki kolom yang diulang.

    b. Fasilitas *monitoring*

       Memantau penggunaan *database* dan operasi.

    c. Program analisis *statistic*

       Menguji kinerja atau statistik penggunaan.

    d. Fasilitas pengaturan ulang indeks

       Menyusun kembali indeks dan *overflow*.

    e. *Garbage collection* dan realokasi

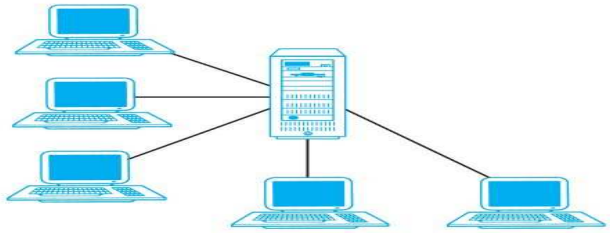
       Menghapus *record* yang secara fisik sudah di hapus dari perangkat penyimpanan dan untuk mengalokasikan kembali ketika dibutuhkan.

2.1.7.6 Arsitektur DBMS

Arsitektur dari aplikasi DBMS terdiri dari beberapa jenis yang digunakan untuk mengimplementasikan DBMS, seperti yang dikatakan oleh Connolly & BEGG (2010 : 108) arsitektur dalam DBMS terdiri atas :

1. Arsitektur aplikasi *teleprocessing*

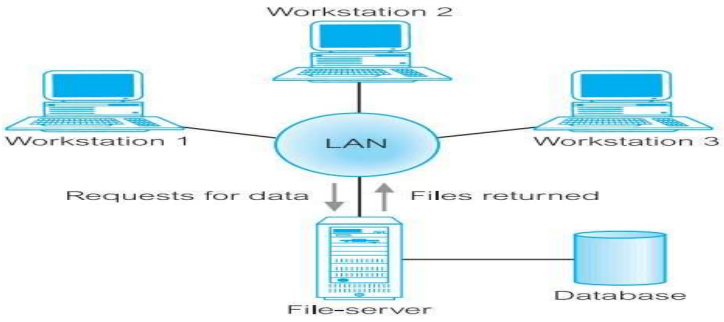
   Aplikasi *teleprocessing* adalah aplikasi komputer yang dapat dijalankan hanya pada satu komputer. *Database* dan programnya berada dalam satu komputer itu sendiri.



Gambar 2.2 *Teleprocessing*
Sumber :(Connolly & BEGG, 2010, p. 108)

2. Arsitektur aplikasi *file server*

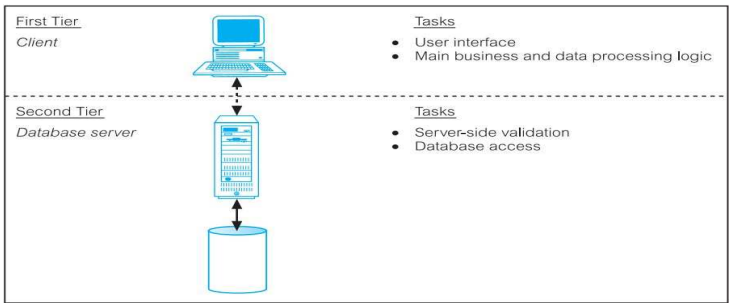
   Arsitektur aplikasi *file server* didasarkan pada hal sederhana yaitu komputer yang berbeda melakukan tugas yang berbeda juga. Aplikasi dipecah-pecah ke dalam dua komponen utama kemudian bekerja sama untuk mencapai satu tujuan.



Gambar 2.3 *File – Server Architecture*
Sumber :(Connolly & BEGG, 2010, p. 109)

3. Arsitektur aplikasi *two-tier client-server (traditional)*

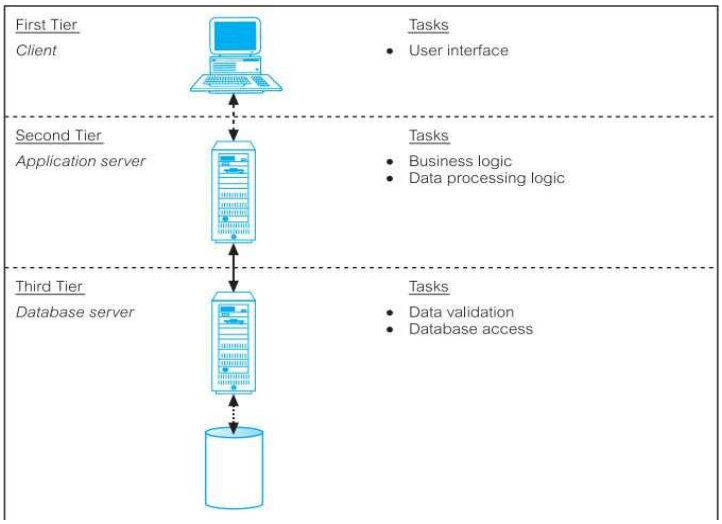
   Arsitektur aplikasi yang dijalankan pada sisi *client*. Mengandung kode yang menampilkan data dan berinteraksi dengan *user*.



Gambar 2.4 *Two–Tier Client Server Architecture*
Sumber :(Connolly & BEGG, 2010, p. 111)

4. Arsitektur aplikasi *three-tier client-server*

   Peran *database* disini adalah mengakses dan mengubah data. *Client* bertanggung jawab menampilkan data kepada pengguna dan mengirimkan *input* dari pengguna. Tingkat ini merupakan sebuah objek yang berada diantara aplikasi *client* dan *server*. Pada arsitektur ini, terdapat sebuah komputer yang mana disebut sebagai *server*, dan ada beberapa komputer lagi yang lainnya dimana biasanya disebut sebagai *client*.



Gambar 2.5 *Three–Tier Client–Server Architecture*
Sumber :(Connolly & BEGG, 2010, p. 113)

5. Arsitektur *N-Tier*

   Arsitektur *three tier* dapat diperlebar menjadi *n-tier*. Dengan menambah *tier* dapat menyediakan fleksibilitas dan skalabilitas yang lebih banyak.

6. *Middleware*

   Software komputer yang menghubungkan komponen *software* atau aplikasi.

7. *Transaction processing monitors*

   Sebuah program yang mengendalikan perpindahan data antara *clients* dan *servers* untuk menyediakan *environment* yang konsisten, khususnya untuk OLTP.

2.1.8 *Relational* model

Relational model berdasarkan konsep matematika *relation*, dimana digambarkan dengan *table*. Dalam melakukan perancangan *database*, kita perlu membuat model *relation* untuk mengetahui hubungan yang dimiliki dalam *database*. Hal ini akan membantu kita dalam menghadapi redudansi data, konsistensi data dan manipulasi data. Menurut Connolly & BEGG (2010 : 142) tujuan *relational model* dispesifikasikan sebagai berikut :

1. Memungkinkan tingkat kemandirian data yang lebih tinggi
2. Menyediakan alasan yang kuat untuk menangani masalah data semantik, konsistensi, dan redudansi
3. Memungkinkan perluasan dari bahasa manipulasi set data orientasi

2.1.8.1 *Relational* Data *structure*

Dalam membuat *relation model*, terdapat struktur yang perlu dipahami terlebih dahulu. Menurut Connolly & BEGG (2010 : 144-146) konsep struktur *relational model* dapat dibagi menjadi beberapa bagian yaitu :

1. *Relation*

   Relation adalah suatu tabel dengan kolom dan baris. Relation memiliki 2 arti yaitu *table* dan *file*.

2. *Attribute*

   Attribute adalah suatu yang dinamakan kolom dari sebuah *relation*. *Attribute* memiliki 2 arti yaitu *column* dan *field*.

3. *Domain*

   Suatu *domain* adalah sekumpulan nilai-nilai yang bisa diijinkan untuk satu *attribute* atau lebih.

Tabel 2.1 Domain

Attribute	Domain Name	Meaning	Domain Definition
branchNo	BranchNumbers	The set of all possible branch numbers	character: size 4, range B001–B999
street	StreetNames	The set of all street names in Britain	character: size 25
city	CityNames	The set of all city names in Britain	character: size 15
postcode	Postcodes	The set of all postcodes in Britain	character: size 8
sex	Sex	The sex of a person	character: size 1, value M or F
DOB	DatesOfBirth	Possible values of staff birth dates	date, range from 1-Jan-20, format dd-mmm-yy
salary	Salaries	Possible values of staff salaries	monetary: 7 digits, range 6000.00–40000.00

Sumber :(Connolly & BEGG, 2010, p. 145)

4. *Tuple*

   Suatu *tuple* adalah suatu baris dari suatu *relation*. Pada Gambar 2.6, setiap baris dari *relation branch* memiliki 4 nilai, dimana satu untuk setiap *attribute*. *Tuple* bisa memiliki 2 arti yaitu *row* dan *record.*

5. *Degree*

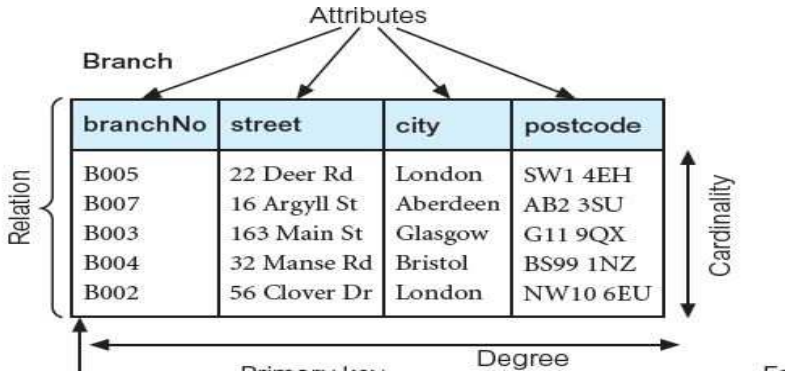
   Degree adalah jumlah *attribute* yang dimilikinya. Relasi memiliki 4 *attribute* atau 4 *degree*, artinya setiap baris dari tabel adalah 4 *tuple*. *Relation* dengan satu *attribute* akan memiliki 1 *degree* dan disebut *relation unary* atau 1 *tuple. Relation* dengan 2 *attribute* disebut *binary, relation* dengan 3 *attribute* disebut *ternary,* dan selanjutnya disebut *n-ary*.

6. *Cardinality*

   Cardinality adalah suatu *relation* yang berisi sejumlah *tuples*. Singkatnya, jumlah dari *tuple* disebut *cardinality.*

7. *Relational database*

   Suatu koleksi dari *relation* yang telah dinormalisasi dengan nama *relation* yang berbeda.



Gambar 2.6 *Relational* data *structure*
Sumber :(Connolly & BEGG, 2010, p. 145)

2.1.8.2 *Database Relations*

Menurut Connolly & BEGG (2010 : 148) *database relations* terbagi menjadi 2 yaitu :

1. *Relational schema*

   Nama *relation* yang dijelaskan oleh sekumpulan *attribute* dan *domain name.*

2. *Relational database schema*

   Sekumpulan *relational schema* dimana memiliki nama yang masing-masing berbeda.

2.1.8.3 *Properties of Relations*

Untuk dapat membuat *relational model*, perlu dipahami sifat-sifat yang dimiliki oleh *relation*. Dengan adanya sifat *relation*, penggambaran *relation model* akan lebih mudah. Menurut Connolly & BEGG (2010 : 148-149) *relation* mempunyai sifat-sifat sebagai berikut:

1. *Relation* memiliki nama yang berbeda satu sama lain dalam *relational schema*
2. Setiap *cell* dari *relational* memiliki satu nilai
3. Setiap *attribute* memiliki nama yang berbeda
4. Nilai satu *attribute* berasal dari *domain* yang sama
5. Setiap *tuple* berbeda dan tidak ada duplikasi *tuple*
6. Urutan *attribute* tidak memiliki makna
7. Secara teoritis urutan *tuple* tidak mempunyai makna

2.1.8.4 *Relational Keys*

Relational keys menjelaskan satu atau lebih *attribute* yang secara unik mengidentifikasikan setiap *tuple* dalam *relation*. Menurut Connolly & BEGG (2010 : 150 - 151) ada beberapa *terminology* yang digunakan untuk *relational key*, yaitu:

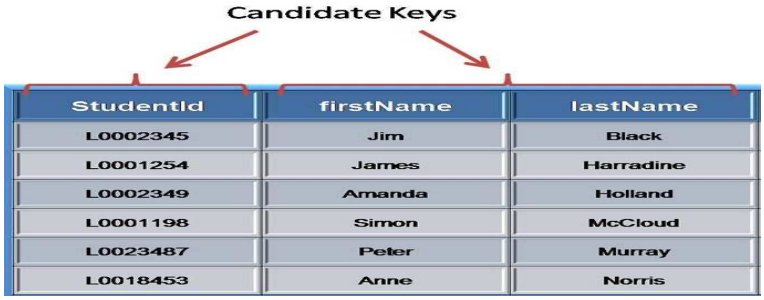
1. *Superkey*

   Attribute yang secara unik mengidentifikasikan sebuah *tuple* dengan sebuah *relation.*

2. *Candidate key*

   Sebuah *superkey* yang sedemikian rupa sehingga tidak ada bagian yang tepat adalah *superkey* di dalam *relation*. Ada beberapa

candidate key untuk *relation*, ketika sebuah *key* berisi dari satu atau lebih *attribute*, sering disebut *composite key*.



Gambar 2.7 Candidate keys (Sumber :rdbms.opengrass.net)

3. *Primary key*

   Candidate key yang dipilih untuk mengidentifikasikan *tuples* secara unik di dalam *relation*. Dalam keadaan terjelek, seluruh kumpulan *attribute* dapat dijadikan *primary key*, tetapi biasanya beberapa *subset* yang lebih kecil sudah cukup untuk membedakan *tupel*. *Candidate key* yang tidak terpilih menjadi *primary key* disebut *alternate key*.

4. *Foreign key*

   Sebuah *attribute* atau sekumpulan *attribute* di dalam satu *relation* yang cocok dengan *candidate key* dari beberapa *relation*.

2.1.8.5 Integrity Constraints

Menurut Connolly & BEGG (2010 : 153-155) dalam *integrity constraints*, ada konsep *null* yang berarti, nilai untuk *attribute* yang saat ini tidak diketahui atau tidak berlaku untuk suatu *tupel*. Ada 2 hal yang penting dalam *integrity rules*, yang merupakan kendala atau batasan yang berlaku untuk *database*, yaitu :

1. *Entity integrity*

   Dalam *relation* dasar, tidak ada *attribute* dari sebuah *primary key* yang diperbolehkan *null*. *Primary key* pada dasarnya merupakan *attribute* yang unik dari sebuah *relation*, maka jika *value* dari *primary key* diperbolehkan *null* tidak akan menjadi suatu pembeda antara satu *record* dengan *record* lain.

2. *Referential integrity*

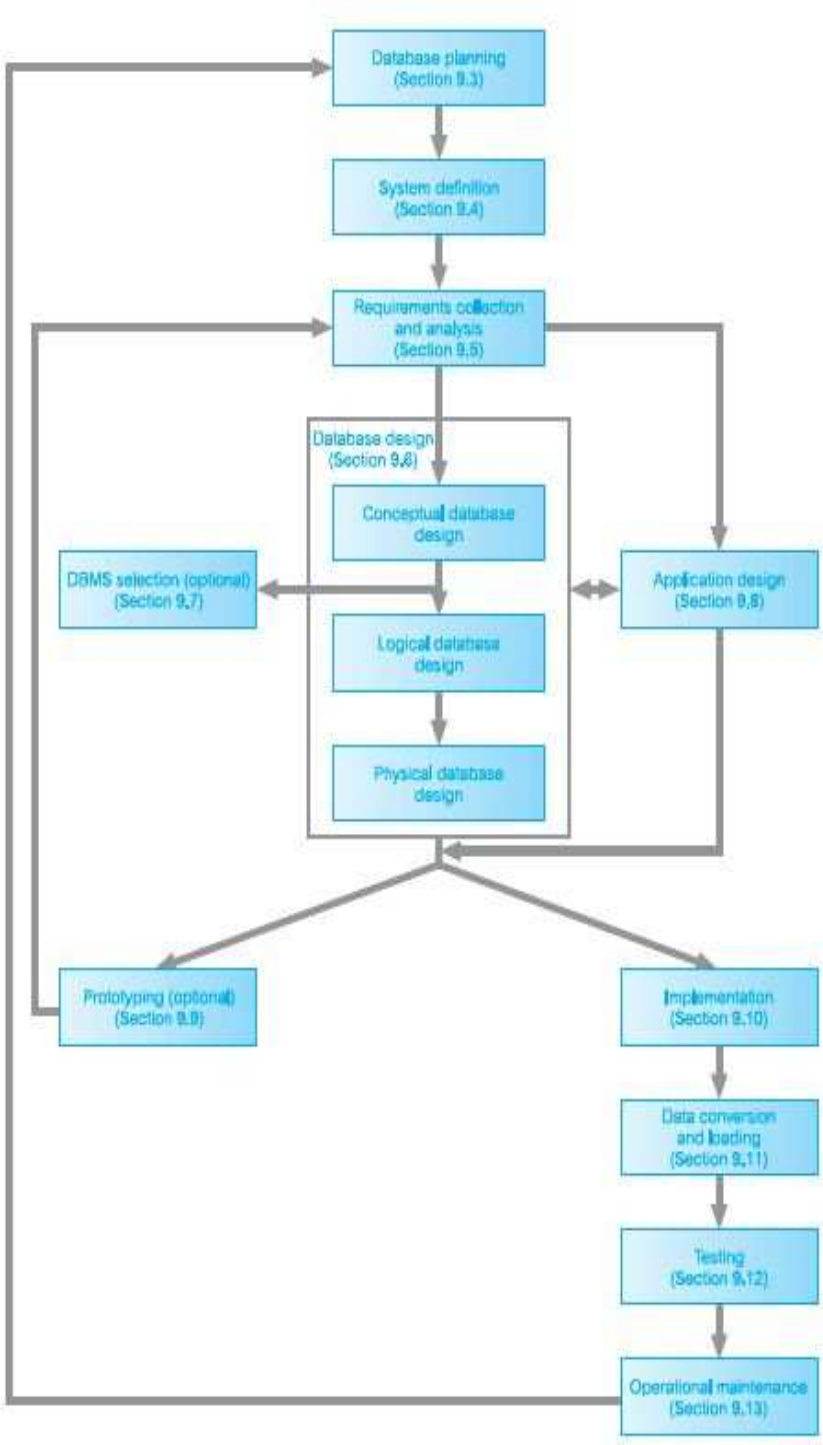
   Jika terdapat *foreign key* dalam *relation*, maka *value* itu harus sesuai dengan *candidate key dari tuple* pada tabel asalnya atau *foreign key* tersebut harus memiliki *value null*.

3. *General constraints*

   Aturan tambahan yang ditentukan oleh pengguna atau *database administrator* yang mendefinisikan atau membatasi beberapa aspek dari suatu perusahaan.

2.1.9 *Database System Development Lifecycle*

Untuk melakukan perancangan basis data, terdapat langkah-langkah yang akan menjadi dasar dalam perancangan. Penting untuk mengetahui dasar dari perancangan agar dapat mengatasi kendala-kendala yang mungkin terjadi selama perancangan. Menurut Connolly & BEGG (2010 : 314) tahap perancangan basis data berdasarkan siklus hidup seperti di bawah ini :



Gambar 2.8 Database system development lifecycle
Sumber :(Connolly & BEGG, 2010, p. 314)

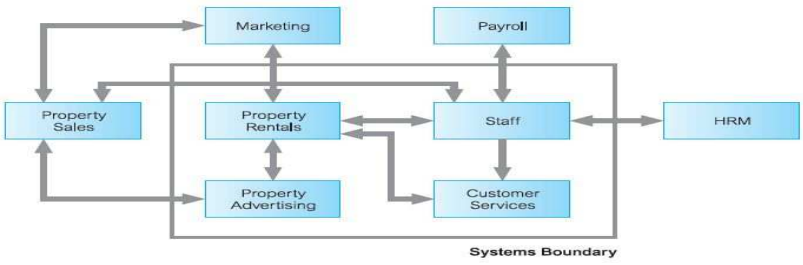
1. *Database planning*

   Dalam melakukan suatu perancangan, tahap pertama yang perlu dilakukan adalah merencanakan. Menurut Connolly & BEGG (2010 : 313) *database*

planning yaitu tahapan perencanaan bagaimana siklus kehidupan dari basis data dapat direalisasikan dengan cara yang paling efektif dan efisien. Langkah awal pada tahap ini adalah menentukan *mission statement*, yaitu menentukan tujuan utama dari sistem basis data yang akan dirancang. *Mission statement* membantu mengklarifikasi tujuan dari sistem *database* dan menyediakan cara membuat sistem *database* yang paling efisien dan efektif. Sekali *mission statement* didefinisikan, aksi selanjutnya mengidentifikasikan *mission objectives*. Setiap *mission objectives* harus dapat mengidentifikasikan tugas tertentu yang harus didukung oleh sistem *database*. *Database planning* juga harus mencakup pengembangan standar yang mengatur bagaimana data akan dikumpulkan, bagaimana format harus ditentukan, dokumentasi penting apa yang dibutuhkan, dan bagaimana desain dan implementasi selanjutnya. Standar bisa jadi sangat memakan waktu untuk mengembangkan dan memelihara, membutuhkan sumber daya untuk mengatur mereka pada awalnya, dan bagaimana desain dan implementasi berlanjut. Standar yang secara baik dirancang akan menyediakan dasar untuk pelatihan karyawan dan kontrol kualitas pengukuran, dan dapat memastikan pekerjaan yang sesuai dengan pola, terlepas dari keterampilan karyawan dan pengalaman.

2. *System definition*

   Tahapan untuk menjelaskan ruang lingkup dan batasan-batasan dari sistem basis data, termasuk sudut pandang *user*, siapa saja *user*-nya, area dari aplikasi dan jangka panjang dari sistem tersebut. Di *system definition* juga membahas *user view*, yang berarti pendefinisian yang dibutuhkan sistem *database* dari perspektif pekerjaan tertentu atau wilayah aplikasi perusahaan.



Gambar 2.9 System definition
Sumber :(Connolly & BEGG, 2010, p. 315)

3. *Requirement collections and analysis*

   Proses mengumpulkan dan menganalisis informasi tentang bagian dari organisasi yang akan didukung oleh sistem *database*, dan menggunakan informasi ini untuk mengidentifikasi kebutuhan sistem baru. Ada 3 pendekatan utama dalam melaksanakan kebutuhan sistem *database* dengan banyak *user view*, yaitu :

   a. *Centralized approach*

      Connolly & BEGG (2010 : 318) mengatakan bahwa, "*centralized approach* adalah kebutuhan setiap *user view* digabung menjadi sekumpulan kebutuhan untuk sistem *database* baru. Data model mewakili semua *user view* yang dibuat selama tahap desain *database"*.

   b. *View integration approach*

      Connolly & BEGG (2010 : 318) mengatakan bahwa, "*view integration approach* adalah kebutuhan setiap *user view* tetap sebagai daftar yang terpisah. Data model mewakili setiap *user view* diciptakan dan kemudian digabungkan selama tahap desain *database*".

   c. *Combination of both aproaches*

      Pendekatan yang menggabungkan antara *centralized approach* dan *view integration approach.*

   Terdapat beberapa teknik yang digunakan untuk mendapatkan informasi, disebut *fact finding techniques.* Ada 5 teknik umum *fact finding techniques*, yaitu :

   a. *Examining documentation*

      Connolly & BEGG (2010 : 344) mengatakan bahwa, "menguji dokumentasi dapat membantu ketika kita mencoba untuk mendapatkan pandangan tentang perlunya *database*. Kita juga dapat menemukan dokumentasi yang dapat membantu memberikan informasi mewakili perusahaan yang terkait dengan masalah tersebut. Jika masalah terkait dengan sistem yang sekarang, seharusnya ada dokumentasi yang terkait dengan sistem itu, contohnya dengan menguji dokumen, formulir, laporan, dan *file* yang terkait dengan sistem saat ini, kita dapat dengan cepat mendapatkan beberapa pemahaman mengenai sistem". Berikut jenis dokumentasi yang harus diuji :

Tabel 2.2 Pengujian dokumentasi

Tujuan	Contoh sumber
Menjelaskan masalah dan kebutuhan *database*.	Memo internal, e-mail, hasil rapat, keluhan pelanggan / karyawan, dokumen yang menjelaskan masalah, dan ulasan kinerja / laporan.
Menjelaskan bagian dari perusahaan yang terpengaruh oleh masalah.	Bagan organisasi, pernyataan misi, rencana strategis perusahaan, tujuan perusahaan untuk diteliti, deskripsi tugas / pekerjaan, contoh panduan dan laporan, dan contoh bentuk komputerisasi dan laporan.
Menjelaskan sistem yang sedang digunakan.	Berbagai jenis flowchart, kamus data, desain sistem *database*, dokumentasi program, dan pelatihan manual.

b. Interviewing

Ada beberapa tujuan menggunakan wawancara, seperti mencari tahu fakta-fakta, membuktikan fakta, mengklarifikasi fakta, membangkitkan antusiasme, mengetahui *end user* yang terlibat, mengidentifikasi kebutuhan, dan mengumpulkan ide-ide dan pendapat. Ada 2 jenis dari *interview* yaitu *interview* terstruktur dan *interview* tidak terstruktur. Maksud dari *interview* tidak terstruktur adalah pertanyaan yang diajukan oleh *interviewer* adalah pertanyaan yang kebanyakan bersifat umum dan sedikit mengenai yang khusus. Sedangkan maksud *interview* terstruktur adalah *interview* yang pertanyaannya lebih khusus, bergantung pada tanggapan dari yang diwawancara. Ada 2 cara untuk mengklarifikasi atau memperdalam tanggapan yang diterima, yaitu *open ended questions* dan *closed ended questions* (Connolly & BEGG, 2010: 344-345).

Tabel 2.3 Keuntungan dan kerugian *interview*

Keuntungan	Kerugian
Mungkin yang diwawancarai untuk merespon bebas dan	Sangat memakan waktu dan mahal, dan oleh karena itu

terbuka terhadap pertanyaan.	mungkin tidak praktis.
Mungkin yang diwawancarai untuk merasa menjadi bagian dari proyek.	Sukses tergantung pada komunikasi dan keterampilan pewawancara.
Memungkinkan pewawancara untuk beradaptasi atau re-kata pertanyaan selama wawancara.	Sukses dapat bergantung pada kesediaan yang diwawancarai untuk berpartisipasi dalam wawancara
Memungkinkan pewawancara untuk beradaptasi atau mengulang pertanyaan selama wawancara.	
Memungkinkan pewawancara untuk mengamati bahasa tubuh diwawancarai.	

c. *Observing the enterprise in operation*

Dengan teknik ini memungkinkan untuk ikut serta atau menonton seseorang melakukan kegiatan mempelajari sistem. Teknik observasi membutuhkan persiapan. Untuk memastikan bahwa observasi berhasil, penting untuk mengetahui banyak tentang individu dan kegiatan yang akan diamati (Connolly & BEGG, 2010 : 345-346).

Tabel 2.4 Keuntungan dan kerugian observasi

Keuntungan	Kerugian
Memungkinkan validitas fakta dan data yang akan diperiksa.	Orang-orang sadar atau tidak sadar dapat melakukan hal yang aneh ketika sedang diamati.
Pengamat dapat melihat apa yang sedang dilakukan.	Mungkin kelilangan pada saat mengamati tugas yang melibatkan tingkat kesulitan yang berbeda atau volume yang biasanya dialami selama jangka waktu tertentu.
Pengamat juga dapat memperoleh data yang menggambarkan lingkungan	Beberapa tugas tidak selalu dilakukan dalam cara di mana mereka diamati.

fisik dari tugas.	
Relatif murah.	Tidak Praktis.
Pengamat dapat melakukan pengukuran kerja.	

d. Research

Menurut Connolly & BEGG (2010 : 346) mengatakan bahwa, "sebuah teknik pencarian fakta yang berguna adalah meneliti aplikasi dan masalah. Jurnal, buku referensi, dan *internet* adalah sumber informasi yang baik. Mereka dapat memberikan informasi tentang bagaimana orang lain telah memecahkan masalah yang sama dan ditambah perangkat lunak apa yang ada untuk menyelesaikan semua atau bahkan sebagian memecahkan masalah".

Tabel 2.5 Keuntungan dan kerugian research

Keuntungan	Kerugian
Dapat menghemat waktu jika solusi sudah ada.	Membutuhkan akses ke sumber-sumber informasi yang tepat.
Peneliti dapat melihat bagaimana orang lain telah memecahkan masalah yang sama atau bertemu dengan persyaratan yang sama.	Pada akhirnya tidak dapat membantu dalam memecahkan masalah karena masalah tersebut tidak didokumentasikan di tempat lain.
Membuat peneliti tidak ketinggalan dengan arus perkembangan sekarang ini.	

e. Questionnaires

Pengumpulan fakta dengan cara mengumpulkan informasi dari sejumlah besar orang. Ketika berhadapan dengan orang yang banyak, tidak ada teknik pencarian fakta lain dimana dapat mentabulasi fakta yang sama secara efisien (Connolly & BEGG, 2010 : 346-347).

Tabel 2.6 Keuntungan dan kerugian kuesioner

Keuntungan	Kerugian

Keuntungan	Kerugian
Orang bisa mengisi dan mengembalikan kuesioner sesuka mereka.	Jumlah responden bisa rendah, mungkin hanya 5% sampai 10%.
Cara yang relatif murah untuk mengumpulkan data dari sejumlah besar orang.	Kuesioner dapat dikembalikan tidak lengkap.
Orang lebih cenderung untuk memberikan fakta nyata sebagai tanggapan yang dapat dijaga kerahasianya.	Mungkin tidak memberikan kesempatan untuk beradaptasi atau menanyakan pertanyaan yang telah disalah artikan.
Tanggapan dapat ditabulasikan dan dianalisis dengan cepat.	Tidak dapat mengamati dan menganalisis bahasa tubuh responden.

4. *Database* design

Menurut Connolly & BEGG (2010 : 320-321) mengatakan bahwa, "desain *database* adalah proses menciptakan desain yang akan mendukung *mission statement* dan *mission objectives* perusahaan untuk sistem *database* yang diperlukan". Ada beberapa pendekatan dalam desain *database* yaitu

a. *Bottom-up*

Bottom-up approach dimulai pada tingkat dasar *attribute* (sifat entitas dan *relationship*), setelah dianalisis asosiasi antara *attribute*, dikelompokkan ke dalam *relationship* yang mewakili jenis entitas dan *relationship* antara entitas. Pendekatan *bottom-up* cocok untuk desain *database* sederhana dengan jumlah *attribute* relatif kecil. Namun, pendekatan ini menjadi sulit ketika diterapkan pada desain *database* yang lebih kompleks dengan sejumlah besar *attribute*, di mana sulit untuk menetapkan semua dependensi fungsional antara *attribute*. Contoh untuk pendekatan ini adalah proses normalisasi.

b. *Top-down*

Pendekatan *top-down* dimulai dengan pengembangan model data yang mengandung beberapa entitas dan hubungan tingkat tinggi dan kemudian menerapkan penyempurnaan *top-down* berturut-turut untuk mengidentifikasi entitas, hubungan, dan *attribute* lebih rendah yang

terkait. Pendekatan ini diilustrasikan menggunakan konsep *entity relationship (ER)* model, dimulai dengan identifikasi entitas dan hubungan antara entitas.

c. *Inside-out*

Pendekatan ini tidak beda jauh dengan pendekatan *bottom-up*, perbedaannya adalah pada tahap awal mengidentifikasi entitas besar lalu dipecah menjadi entitas relasi dan *attribute* yang berhubungan dengan entitas besar.

d. *Mixed*

Menggunakan pendekatan *bottom-up* dan *top-down.*

Tahap dari desain *database* sendiri ada 3, yaitu :

a. *Conceptual database design*

Connolly & BEGG (2010 : 321-322) mengatakan bahwa, "*conceptual database design* adalah proses pembangunan sebuah model dari data yang digunakan dalam sebuah perusahaan, tidak bergantung pada semua pertimbangan fisik". *Conceptual database design* sepenuhnya tidak bergantung dari rincian pelaksanaan seperti perangkat lunak DBMS *target*, program aplikasi, bahasa pemrograman, platform perangkat keras, atau pertimbangan fisik lainnya. Ada 9 langkah untuk membuat *conceptual database design,* yaitu :

1) Mengidentifikasi tipe entitas

Connolly & BEGG (2010 : 471) mengatakan bahwa, tujuan dari mengidentifikasi tipe entitas adalah "mengidentifikasi kebutuhan tipe entitas". Pertama-tama dilakukan pengidentifikasian entitas dengan melihat objek-objek yang digunakan oleh *user*, tujuannya adalah untuk mengidentifikasi entitas yang diperlukan. Satu cara dalam mengidentifikasi entitas adalah menguji kebutuhan *user* secara detil. Contoh hasil dari *Staff Client user views DreamHome* adalah *staff, private owner, client, lease, propertyforrent, business owner, preference entitas.*

2) Mengidentifikasi tipe relasi

Connolly & BEGG (2010 : 472 - 473) mengatakan bahwa tujuan dari mengidentikasi tipe relasi adalah “mengetahui relasi penting yang ada antara entitas-entitas”.

Tabel 2.7 Identifikasi tipe relasi

Entity Name	*Description*	*Aliases*	*Occurrence*
Staff	Peraturan umum menjelaskan semua *staff* yang dipekerjakan oleh DreamHome	Employee	Setiap *staff* bekerja pada satu *branch*
Property For Rent	Peraturan umum yang menjelaskan semua property untuk dipinjam	Property	Setiap property memiliki pemilik tunggal dan ada pada cabang yang khusus, dimana property dikelola oleh satu *staff*. Properti dapat dilihat oleh banyak *client*, tetapi dipinjam oleh *client* tunggal pada satu waktu.

3) Mengidetifikasikan dan menghubungkan *attribute-attribute* dengan entitas atau relasi

Connolly & BEGG (2010 : 474-475) mengatakan bahwa, tujuan dari tahap ini adalah “mengasosiasikan *attribute* dengan entitas atau jenis hubungan yang sesuai”.

Tabel 2.8 Asosiasi atribut dengan entitas atau tipe relasi

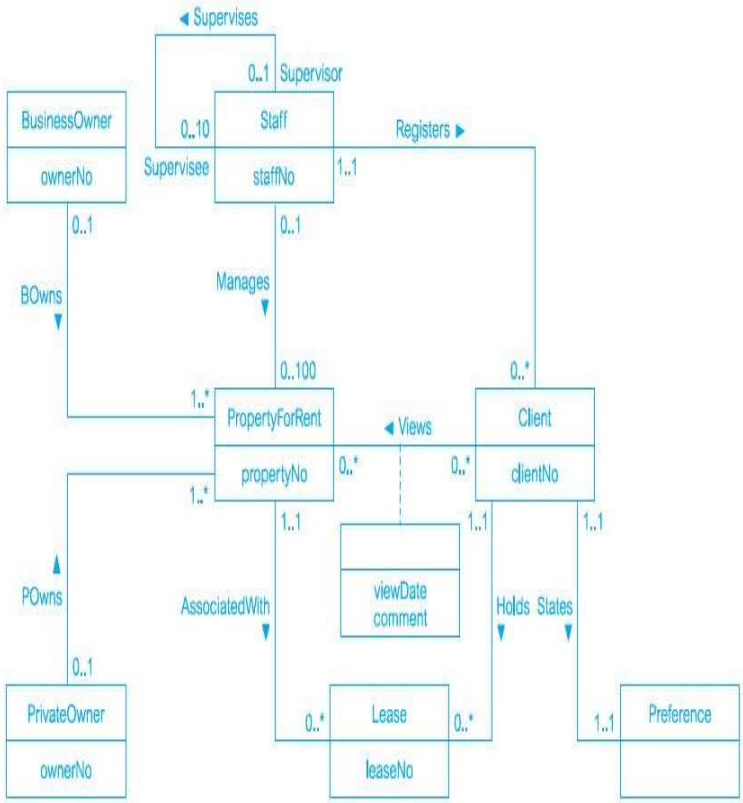
Entity name	*Multiplicity*	*Relationship*	*Multiplicity*	*Entity name*
Staff	0..1	*Manages* *Supervises*	0..100 0..10	*PropertyForRent* *Staff*
PropertyForRent	1..1	*AssociatedWith*	0..*	*Lease*

4) Menentukan domain *attribute*

   Connolly & BEGG (2010 : 478) mengatakan bahwa, "domain adalah sebuah kumpulan nilai-nilai dari satu atau lebih *attribute* yang diambil nilainya. Tujuan dari langkah ini adalah menentukan domain untuk *attribute* pada model data *conceptual*". Contohnya, untuk sebuah nomor induk pegawai sebuah perusahaan memiliki 10 karakter yang diawali dengan 2 huruf dan 8 digit angka, range angka (1-99999999).

5) Menentukan *attribute candidate*, *primary*, dan *alternate key*

   Connolly & BEGG (2010 : 479) mengatakan bahwa, tujuan dari tahap ini adalah "mengidentifikasi *candidate key* untuk setiap entitas dan, jika ada lebih dari satu *candidate key*, maka satu akan dipilih sebagai *primary key* sementara sisanya menjadi *alternate key*". Ada beberapa panduan ketika memilih *primary key* dari *candidate key*, yaitu *candidate key* dengan *attribute* yang paling sedikit, *candidate key* yang paling tidak mungkin berubah nilainya, *candidate key* dengan paling sedikit karakter (*attribute* huruf) *candidate key* dengan nilai maksimum terkecil (*attribute* angka) dan *candidate key* yang paling mudah dalam membentuk sudut pandang pengguna.



Gambar 2.10 ER Diagram dengan primary key
Sumber :(Connolly & BEGG, 2010, p. 480)

6) Mempertimbangkan penggunaan konsep *enhanced modeling* (*optional*)

   Connolly & BEGG (2010 : 480 - 481) mengatakan bahwa, tujuan dari tahap ini adalah "mempertimbangkan penggunaan *enhanced modeling*, seperti *spezialitation* / *generalization*, *aggregation*, dan *composition"*.

7) Melakukan pengecekan redudansi pada model

   Connolly & BEGG (2010 : 482 - 483) mengatakan bahwa, tujuan dari tahap ini adalah "melakukan pemeriksaan model data *conceptual* untuk menidentifikasi adanya redudansi dan menghilangkannya". Ada 3 aktivitas dalam tahap ini, yaitu :

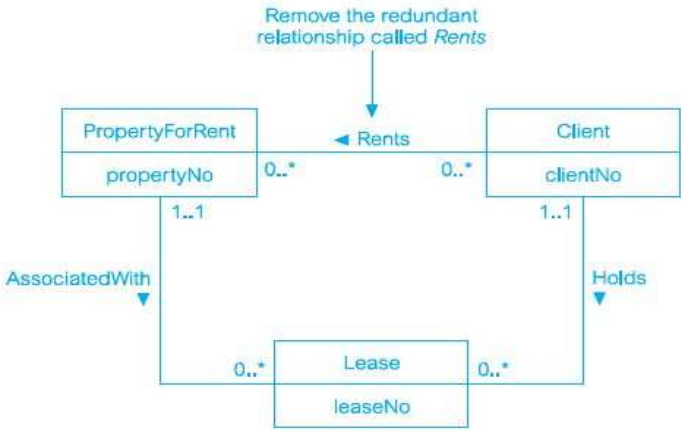
   a) Re-examine one-to-one relationships

      Dalam mengidentifikasikan entitas, ada kemungkinan kita mengidentifikasikan 2 entitas yang mewakili objek yang sama. Contoh, kita sudah mengidentifikasi 2 entitas yaitu *client* dan

renter, yang berbeda hanya dalam penamaan. Kedua entitas tersebut harus di gabungkan, jika *primary key* berbeda, pilih satu untuk dijadikan *primary key* dan sisanya *alternate key*.

b) Remove redundant relationships

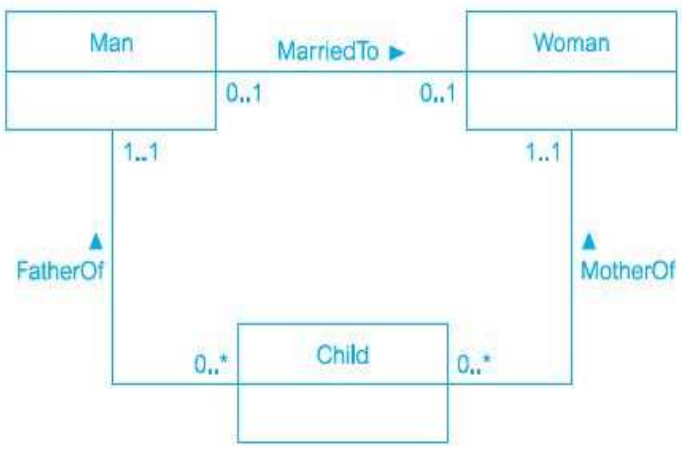
Relationship dikatakan *redundant* jika informasi yang sama didapatkan dari *relationship* lain. Contoh, *relationship* antara entitas *property for rent, lease* dan *client*. Ada 2 cara untuk mengetahui *property* yang dipinjam *client*, yang pertama hubungan langsung dengan hubungan *rents* antara entitas *client* dan *property for rent* atau hubungan tidak langsung, menggunakan *holds* dan berhubungan melalui entitas *lease*.



Gambar 2.11 Remove redundant relationships
Sumber :(Connolly & BEGG, 2010, p. 482)

c) Consider time dimension

Waktu dimensi hubungan penting ketika menilai redundansi. Ada 2 jalan antara *man* dan *child*, yang pertama hubungan langsung *father of* atau hubungan *married to* dan *mother of*. Akibatnya, kita mungkin berpikir bahwa hubungan *father of* tidak perlu. Namun ini akan menjadi salah untuk dua alasan, alasan pertama ayahnya mungkin memiliki anak dari pernikahan sebelumnya dan kita memodelkan pernikahan ayah yang sekarang dengan 1:1 *relationship*. Alasan kedua, ayah dan ibu tidak menikah.



Gambar 2.12 Mempertimbangkan dimensi waktu
Sumber :(Connolly & BEGG, 2010, p. 483)

8) Menvalidasi model data *conceptual* terhadap transaksi *user*

   Connolly & BEGG (2010 : 483-485) mengatakan bahwa, tujuan dari tahap ini adalah "meyakinkan bahwa model data *conceptual* mendukung transaksi". Ada 2 cara untuk meyakinkan bahwa model data *conceptual* mendukung transaksi, yaitu :

   a) Menjelaskan transaksi

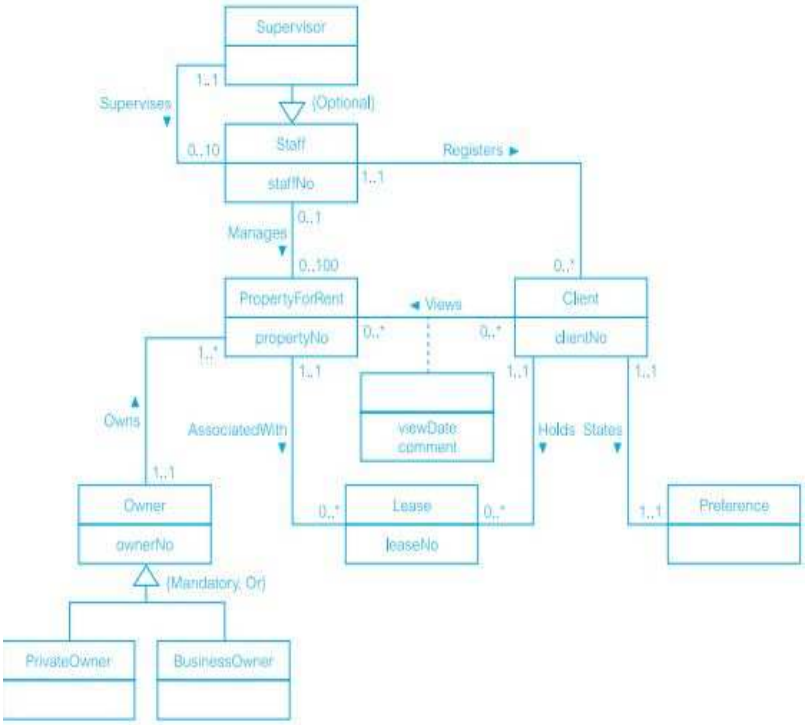
      Memeriksa bahwa semua informasi (entitas, hubungan, dan *attribute*) yang dibutuhkan oleh setiap transaksi disediakan oleh model, dengan mendokumentasikan deskripsi kebutuhan setiap transaksi.

   b) Jalur transaksi

      Pendekatan kedua untuk memvalidasi model data terhadap transaksi yang dibutuhkan melibatkan diagram yang mewakili jalur yang diambil langsung oleh setiap transaksi pada ER diagram.

9) Melakukan *review* model data *conceptual* dengan *user*

   Connolly & BEGG (2010 : 485) mengatakan bahwa, tujuan dari tahap ini adalah "meninjau model data *conceptual* dengan pengguna untuk memastikan bahwa mereka mempertimbangkan model untuk menjadi representasi benar dari persyaratan data perusahaan".



Gambar 2.13 Conceptual data model
Sumber :(Connolly & BEGG, 2010, p. 481)

b. Logical database design

Connolly & BEGG (2010 : 490) mengatakan bahwa, tujuan dari tahap ini adalah "menerjemahkan model data *conceptual* ke dalam model data *logical* yang kemudian divalidasi untuk memeriksa kebenarannya secara struktural dan kemampuannya untuk mendukung kebutuhan transaksi". Ada 7 tahap dalam desain *database logical*, yaitu :

1) Menurunkan relasi untuk model data *logical*

Connolly & BEGG (2010 : 492) mengatakan bahwa, tujuan dari tahap ini adalah "membuat *relations* untuk model data *logical* untuk mewakili entitas, hubungan dan *attribute* yang sudah diidentifikasi". Kemungkinan yang terjadi pada model data *conceptual* pada *relations* yang diturunkan,yaitu :

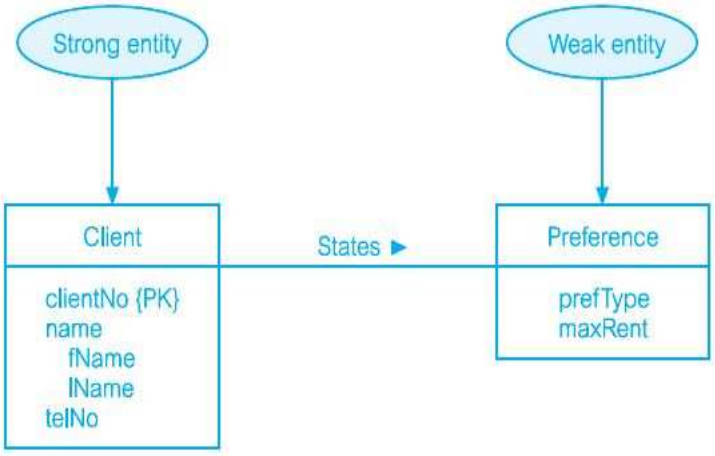
a) Strong entity types

Menurut Connolly & BEGG (2010 : 383) mengatakan bahwa, "*strong entity types* adalah entitas yang keberadaannya tidak bergantung pada keberadaan entitas lainnya". Setiap kejadian

entitas teridentifikasi secara unik yang memiliki *attribute primary key* yang dapat membedakan dari entitas yang lain. Contoh, ketika ingin mengidentifikasi secara unik setiap siswa dalam sekolah, dapat menggunakan *attribute* NIS(Nomor Induk Siswa) untuk dapat membedakan. Siswa (NIS, nama, alamat, jenis kelamin, agama) *primary key* : NIS

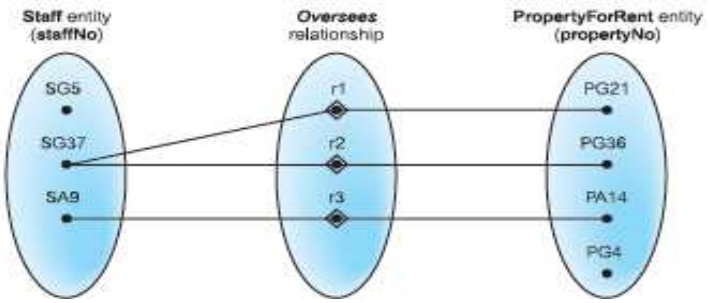
b) *Weak entity types*

Connolly & BEGG (2010 : 383 - 384) mengatakan bahwa, "*weak entity types* adalah entitas yang keberadaanya tergantung oleh beberapa entitas yang lain". Setiap kejadian entitas tidak dapat teridentifikasi secara unik hanya dengan menggunakan *attribute* entitas tersebut, tidak seperti halnya tipe entitas kuat yang menggunakan *primary key*. *Primary key* dari entitas lemah sebagian atau seluruhnya berasal dari setiap entitas *owner* sehingga identifikasi *primary key* dari entitas lemah tidak dapat dibuat sampai semua hubungan dengan entitas *owner* dipetakan. Contoh, pada entitas *preference* tidak memiliki *primary key*, artinya tidak dapat mengidentifikasi setiap kejadian dari entitas *preference* hanya dengan menggunakan *attribute* yang sudah ada.



Gambar 2.14 Entitas kuat dan lemah
Sumber :(Connolly & BEGG, 2010, p. 384)

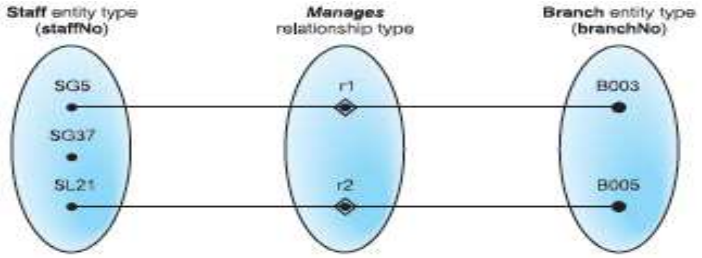
c) One to many relationship types



Gambar 2.15 Hubungan one to many
Sumber :(Connolly & BEGG, 2010, p. 387)

Contoh diatas menggambarkan 3 tipe *relationship staff* yang mengawasi suatu *property* yang akan disewa. *Staff* SG37 mengawasi *property* PG21 dan PG36 dengan *relationship* r1 dan r2, *staff* SA9 mengawasi *property* PA14 dengan *relationship* r3, namun *staff* SG5 tidak mengawasi *property* manapun dan *property* PG4 tidak diawasi oleh *staff*. Dapat disimpulkan bahwa seorang *staff* dapat mengawasi nol atau lebih *property*, dan tiap *property* dapat diawasi oleh nol atau satu orang *staff*. Tipe *relationship* seperti ini disebut *one – to – many*.

d) One to one relationship types



Gambar 2.16 Hubungan one to one
Sumber :(Connolly & BEGG, 2010, p. 386)

Contoh diatas menggambarkan 2 tipe *entitas staff* yang mengelola *branch*. *Staff* SG5 mengatur cabang B003 dengan *relationship* r1 dan *staff* SL21 mengatur *branch* B005 dengan *relationship* r2, tetapi *staff* SG37 tidak mengatur cabang manapun. Dapat disimpulkan bahwa seorang *staff* hanya dapat

mengatur nol atau satu *branch* dan setiap *branch* hanya dapat diatur oleh satu orang *staff*. Tipe *relationship* seperti ini disebut *one – to – one*.

e) One to one recrusive relationship types

Connolly & BEGG (2010 : 495) mengatakan bahwa, "untuk 1:1 *recursive relationship with mandatory participation on both sides*, wakili hubungan rekursif sebagai *relation* tunggal dengan dua salinan *primary key*, yang mana satu dari salinan *primary key* merupakan *foreign key* dan harus diganti namanya untuk menunjukkan hubungan yang diwakilinya. Untuk 1:1 *recursive relationship with mandatory participation on only one side*, memiliki pilihan untuk membuat *relation* tunggal dengan dua salinan *primary key* atau membuat *relation* baru untuk mewakili hubungan. *Relation* baru hanya akan memiliki dua *attribute* yaitu kedua salinan *primary key*, dimana salinan *primary key* bertindak sebagai *foreign key* dan harus diganti namanya untuk menunjukkan tujuan masing-masing dalam *relation*. Untuk 1:1 *recursive relationship with optional participation on both sides*, sekali lagi membuat *relation* baru".

f) Superclass/subclass relationship types

Super class entity seperti *parent entity* dan *subclass entity* seperti *child entity*.

Participation constraint	Disjoint constraint	Relations required
Mandatory	Nondisjoint {And}	Single relation (with one or more discriminators to distinguish the type of each tuple)
Optional	Nondisjoint {And}	Two relations: one relation for superclass and one relation for all subclasses (with one or more discriminators to distinguish the type of each tuple)
Mandatory	Disjoint {Or}	Many relations: one relation for each combined superclass/subclass
Optional	Disjoint {Or}	Many relations: one relation for superclass and one for each subclass

Gambar 2.17 Panduan untuk superclass / subclass
Sumber :(Connolly & BEGG, 2010, p. 496)

Option 1 – Mandatory, nondisjoint

AllOwner (ownerNo, address, telNo, fName, lName, bName, bType, contactName, pOwnerFlag, bOwnerFlag)
Primary Key ownerNo

Option 2 – Optional, nondisjoint

Owner (ownerNo, address, telNo)
Primary Key ownerNo

OwnerDetails (ownerNo, fName lName, bName, bType, contactName, pOwnerFlag, bOwnerFlag)
Primary Key ownerNo
Foreign Key ownerNo **references** Owner(ownerNo)

Option 3 – Mandatory, disjoint

PrivateOwner (ownerNo, fName, lName, address, telNo)
Primary Key ownerNo

BusinessOwner (ownerNo, bName, bType, contactName, address, telNo)
Primary Key ownerNo

Option 4 – Optional, disjoint

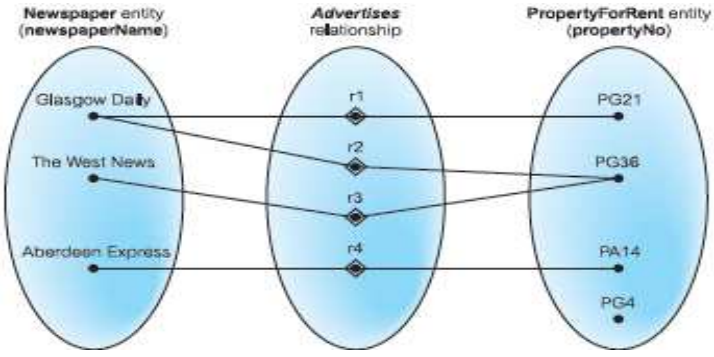
Owner (ownerNo, address, telNo)
Primary Key ownerNo

PrivateOwner (ownerNo, fName, lName)
Primary Key ownerNo
Foreign Key ownerNo **references** Owner(ownerNo)

BusinessOwner (ownerNo, bName, bType, contactName)
Primary Key ownerNo
Foreign Key ownerNo **references** Owner(ownerNo)

Gambar 2.18 Tipe hubungan superclass / subclass
Sumber :(Connolly & BEGG, 2010, p. 497)

g) *Many to many relationship types*



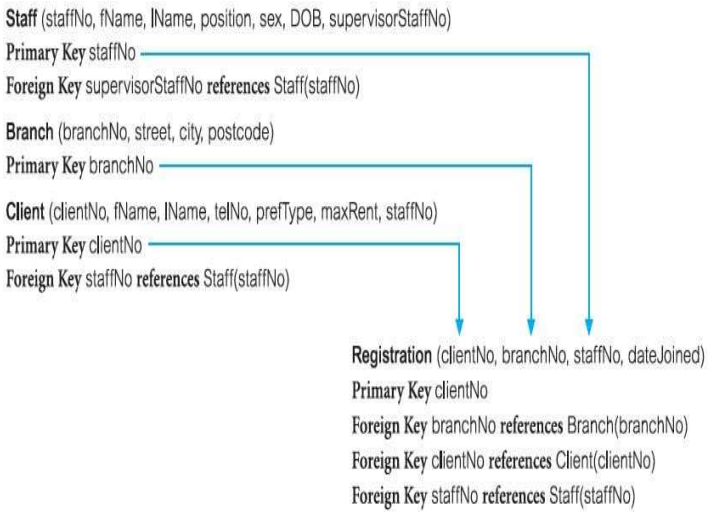
Gambar 2.19 Hubungan many to many
Sumber :(Connolly & BEGG, 2010, p. 388)

Contoh diatas menggambarkan 4 tipe *relationship* suatu koran yang mengiklankan *property*. *Glasgow Daily* mengiklankan

property PG21 dan PG36 dengan *relationship* r1 dan r2, *The West News* juga mengiklankan *property* PG36 dengan *relationship* r3, *Aberdeen Express* mengiklankan *property* PA14 dengan *relationship* r4 dan *property* PG4 tidak diiklankan. Disimpulkan bahwa satu koran dapat mengiklankan satu atau lebih *property* dan satu *property* dapat diiklankan oleh nol atau lebih koran. Tipe *relationship* seperti ini disebut *many – to – many relationship.*

h) Complex relationship types

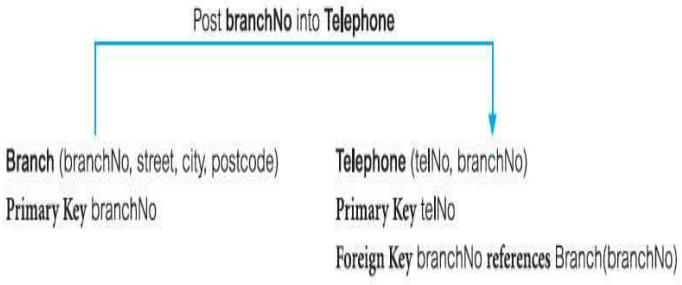
Connolly & BEGG (2010 : 498) mengatakan bahwa, "untuk setiap hubungan yang kompleks, buat *relation* untuk mewakili hubungan dan setiap *attribute* yang merupakan bagian dari hubungan. Kirimkan salinan *primary key attribute* dari entitas yang terlibat dalam hubungan yang kompleks ke dalam hubungan baru, untuk bertindak sebagai *foreign key*". Contoh, *registers* memilik *ternary relationship* dalam *branch user view* yang mewakili hubungan antara *staff* yang mendaftar sebagai *client* baru di *branch.* Untuk mewakili ini, kita menciptakan hubungan untuk entitas kuat *branch*, *staff*, dan *client*, dan menciptakan hubungan *registration* untuk mewakili hubungan *register.*



Gambar 2.20 Complex relationship type
Sumber :(Connolly & BEGG, 2010, p. 498)

i) *Multi-valued attributes*

Connolly & BEGG (2010 : 499) mengatakan bahwa, "untuk setiap *attribute multi-valued* dalam suatu entitas, buatlah *relation* baru untuk mewakili *attribute multi-valued* dan *primary key* dari entitas dalam *relation* baru, untuk bertindak sebagai *foreign key*. Kecuali *attribute multi-valued* itu sendiri merupakan *alternate key* dari entitas, *primary key* dari relasi baru adalah kombinasi dari *attribute multi-valued* dan *primary key* dari entitas. Misalnya, dalam *user view branch* untuk mewakili situasi di mana satu *branch* telah sampai tiga nomor telepon, *attribute* telpNo dari entitas *branch* telah didefinisikan sebagai *attribute multi-valued*. Untuk mewakili ini, kita membuat hubungan untuk entitas *branch* dan menciptakan hubungan baru yang disebut Telepon untuk mewakili *attribute multi-valued* telpNo.



Gambar 2.21 Multi – valued attributes
Sumber :(Connolly & BEGG, 2010, p. 499)

2) Validasi *relations* menggunakan normalisasi

Connolly & BEGG (2010 : 501) mengatakan bahwa, tujuan dari tahap ini adalah "memvalidasi *relation* di model data *logical* menggunakan normalisasi. Tujuan dari normalisasi adalah memastikan bahwa himpunan *relation* memiliki jumlah minimal dan belum memiliki *attribute* yang cukup untuk mendukung persyaratan data perusahaan, dan juga *relation* harus memiliki redundansi data minimal untuk menghindari masalah update". Menurut Connolly & BEGG (2010 : 428) "normalisasi adalah

teknik resmi untuk menganalisis*relation* berdasarkan *primarykey* (atau *candidate key*) dan ketergantungan secara fungsional (Codd , 1972b)". Teknik ini melibatkan serangkaian peraturan yang dapat digunakan untuk menguji *relation* individu sehingga sebuah *database* dapat dinormalisasi untuk tingkat apapun. Tiga bentuk normal awalnya diusulkan disebut bentuk normal pertama (1NF), bentuk normal kedua (2NF) , dan bentuk normal ketiga (3NF). Proses Normalisasi :

a) First Normal Form (1NF)

Connolly & BEGG (2010 : 430 - 431) mengatakanbahwa, *"Unnormalized Form* (UNF) adalah sebuah tabel yang berisi satu atau lebih *repeating group"*. *Repeatinggroup* adalah sebuah *attribute* atau himpunan *attribute* dalam tabel yang memiliki lebih dari satu nilai (*multiplevalue*) untuk sebuah *primary key* pada tabel tersebut.*First Normal Form* (1NF) adalah sebuah relasi dimana titik temu antara baris dan kolomnya hanya mengandung satu nilai.Untuk mengubah tabel UNF ke 1NF, dilakukan identifikasi dan penghapusan *repeatinggroup* dalam tabel.

b) Second Normal Form (2NF)

Connolly & BEGG (2010 : 434)mengatakanbahwa, "*Second Normal Form* (2NF) adalah suatu hubungan yang ada dalam 1NFdan setiap non-*primary-key attribute*, *functional dependency* sepenuhnya pada *primary key"*. Normalisasi 1NF ke 2NF melibatkan penghapusan *partialdependencies*. Hapus *attributepartialdependencies* dari *relation* dengan menempatkannya dalam *relation* baru bersama dengan salinan *determinant*, apabila terdapat *partialdependencies*. *Functionaldependency* menjelaskan hubungan antara *attribute* dalam suatu *relation*. Misalnya, X dan Y adalah *attribute* dari *relations* R, Y secara *functional dependency* pada X (dilambangkan $X \rightarrow Y$), jika setiap nilai X terkait dengan tepat satu nilai Y. (X dan Y mungkin masing-masing terdiri dari satu atau *attribute* lebih). *Full functional dependency* menunjukkan

bahwa jika X dan Y adalah *attribute* dari suatu hubungan, Y adalah *fully functionally dependent* pada X jika Y adalah *functional dependency* pada X, tetapi tidak pada setiap subset dari X.

c) Third Normal Form (3NF)

Connolly & BEGG (2010 : 435 - 437) mengatakan bahwa, "*Third Normal Form* (3NF) adalah suatu hubungan yang ada dalam 1NF dan 2NF dan dimana tidak ada non-*primary-key attribute* yang *transitively dependent*pada *primary key"*. Normalisasi 2NF ke 3NF melibatkan penghapusan *transitivedependency*. Hapus *relation* dengan cara menempatkan *attribute* dalam suatu *relation*baru bersama dengan salinan *determinant*, apabila terdapat *transitivedependency*. *Transitivedependency* adalah sebuah kondisi dimana X, Y dan Z adalah *attribute* dari sebuah hubungan seperti jika X → Y dan Y → Z, maka Z adalah *dependent* secara transitif pada XY melalui (disediakan bahwa X tidak *functionally dependent* pada Y atau Z).

3) Validasi *relations* terhadap transaksi *user*

Connolly & BEGG (2010 : 502) mengatakan bahwa, tujuan dari tahap ini adalah "Memastikan bahwa *relations* dalam model data *logical* mendukung transaksi yang diperlukan".

4) Memeriksa *integrityconstraints*

Connolly & BEGG (2010 : 502 - 505) mengatakan bahwa, tujuan dari tahap ini adalah "Memeriksa apakah *integrityconstraints* ditampilkan dalam model data *logical*. *Integrityconstraints* ini bertujuan untuk melindungi basis data dari ketidaklengkapan,ketidakakuratan, atau ketidakkonsistenan". Ada beberapa tipe *integrityconstraints* :

a) Required data

Beberapa *attribute* harus memiliki nilai pada datanya sehingga tidak diperbolehkan *null*. Contoh, seorang mahasiswa pasti memiliki jurusan yang diambilnya.

b) *Attribute domain constraints*

Attribute memiliki *domain* yang merupakan sekumpulan nilai yang sah. Contoh, jenis kelamin dari mahasiswa hanya ada 2 yaitu laki-laki dan perempuan.

c) *Multiplicity*

Multiplicity mewakili *constraints* yang ditempatkan pada *relationship* antara data di *database*. Contoh, banyak mahasiswa yang berada dalam satu kelas.

Tabel 2.9 Multiplicity

ultiplicity	Arti
0..1	Nol atau satu kejadian entitas
1..1 (or just 1)	Satu kejadian entitas
0..* (or just *)	Nol atau banyak kejadian entitas
1..*	One atau banyak kejadian entitas
5..10	Minimum 5 dan maksimum 10 kejadian entitas
0, 3, 6–8	Nol, tiga atau enam sampai delapan kejadian entitas

d) *Entity integrity*

Primary key dari sebuah entitas tidak dapat bernilai *null*. Contoh, setiap *tuple* dari *relation* mahasiswa harus memiliki nilai *attributeprimary key* yaitu nim.

e) *Referential integrity*

Foreign key menghubungkan setiap *tuple* dalam *childrelation* dengan tuple dalam *parentrelation* dengan mencocokkan nilai *candidate key. Referential integrity* berarti jika *foreign key* mengandung nilai, nilai tersebut yang harus merujuk pada suatu *tupel* yang ada di *parentrelation*. Sebagai contoh, perhatikan *Staff* mengatur *PropertyForRent*. *Attributestaff*No dalam relation *PropertyForRent* menghubungkan properti untuk disewakan kepada *tupel* dalam relasi *staff* yang berisi anggota *staff* yang mengelola properti itu. Jika *staff*No tidak *null*, pasti

berisi nilai *valid* yang ada dalam *attributestaff*No dari *relation staff*, atau properti akan ditugaskan ke anggota *staff* yang tidak ada.

Gambar 2.22 Referential integrity
Sumber (Connolly & BEGG, 2010, p. 505)

Ada dua masalah mengenai *foreign key* yang harus diatasi, yang pertama menganggap apakah *nulls* diperbolehkan untuk *foreign key*. Sebagai contoh, kita dapat menyimpan rincian properti untuk disewa tanpa memiliki anggota *staff* tertentu untuk mengelolanya. Masalahnya bukan jumlah *staff* yang ada, tetapi apakah jumlah *staff* harus ditentukan. Secara umum, jika partisipasi *child relation* dalam hubungan mandatory (*nulls* tidak diperbolehkan) dan optional (*nulls* diperbolehkan). Masalah kedua yang harus kita mengatasi adalah cara untuk memastikan *referential integrity*. Ada beberapa strategi yang

harus dipertimbangkan dalam menghapus *tuple* dari *relation parent* (*Staff*) :

A. NO ACTION

B. Strategi yang digunakan adalah mencegah penghapusan dari *parent relation* jika terdapat hubungan ke child *tuple.*

C. CASCADE

Apabila pada *tuple parent* dihapus, maka secara otomatis *tuple child*akan terhapus juga.

D. SET NULL

Jika pada *tuple parent* dihapus, maka nilai *foreign key* pada semua *tuple child* otomatis akan terisi nilai *null*.

E. SET DEFAULT

Jika pada *tuple parent* dihapus, maka *foreign key* pada semua *tuple child*akan menerima nilai *default*.

f) General constraints

Constraints tambahan yang dibuat oleh *user* atau seseorang *database administrator* dari basis data tersebut. Contoh, satu kelas melarang mengatur lebih dari 50 pelajar.

5) Melakukan *review* model data *logical* dengan *user*

Connolly & BEGG (2010 : 506) mengatakan bahwa, tujuan dari tahap ini adalah "*Review* model data *logical*bersama *user* untuk memastikan *user* bahwa model yang telah dibuat sesuai dengan kebutuhan data perusahaan".

6) Menggabungkan model data *logical* ke model data *global* (*optional*)

Connolly & BEGG (2010 : 506 - 517) mengatakan bahwa, tujuan dari tahap ini adalah "menggabungkan model data *logical* ke dalam data *logical global tunggal* yang menampilkan semua *user views* dari *database*".

7) Melakukan pemeriksaan untuk perkembangan di masa datang

Connolly & BEGG (2010 : 517 – 518) mengatakan bahwa, tujuan dari tahap ini adalah "Menentukan apakah ada perubahan yang besar di masa mendatang dan menilai apakah model data *logical*

tersebut dapat menyesuaikan terhadap perubahan-perubahan tersebut".

c. Physical database design

Connolly & BEGG (2010 : 523) mengatakan bahwa *physicaldatabasedeisgn* adalah "Proses menghasilkan deskripsi implementasi dari *database* pada secondary storage, menggambarkan *baserelation, fileorganizations* dan indeks yang digunakan untuk mencapai akses yang efisien terhadap data, dan setiap terkait *integrity constraints* dan langkah-langkah keamanan". Ada 6 tahap dari *physical database design*, yaitu :

1) Translate logical data model for target DBMS

Connolly & BEGG (2010 : 524) mengatakan bahwa, tujuan dari tahap ini adalah "Menghasilkan skema *databaserelational* dari model data *logical* yang dapat diimplementasikan pada *target* DBMS".

a) Design base relations

Connolly & BEGG (2010 : 525) mengatakan bahwa, tujuan dari tahap ini adalah "Memutuskan cara untuk mewakili *base relations* yang sudah diidentifikasi di model data *logical* ke dalam *target* DBMS". Setiap *relation* yang diidentifikasikan dalam model data *logical* memiliki definisi yang terdiri dari :

1) Nama *relation*
2) Daftar dari *simple attribute*
3) *Primary key*, *alternate keys*, dan *foreign keys*
4) *Refential integrity constraints* untuk setiap *foreign keys* yang diidentifikasi

Dari kamus data, setiap *attribute* juga memiliki :

1) *Domainnya*, yang terdiri dari tipe data, panjang, dan *constraints* pada *domain*
2) Sebuah *optional* bernilai *default* untuk *attribute*
3) Apa *attribute* boleh bernilai *null*
4) Apa *attribute* merupakan data yang diturunkan dan cara menghitung data

b) Design representation of derived data

Connolly & BEGG (2010 : 526 - 528) mengatakan bahwa, tujuan dari tahap ini adalah "Memutuskan cara merepresentasikan *derived* data yang di model data *logical* ke *target* DBMS". Contoh, jumlah mahasiswa yang kuliah di sebuah jurusan.

c) Design general constraints

Connolly & BEGG (2010 : 526 - 528) mengatakan bahwa , tujuan dari tahap ini adalah "Merancang *generalconstraints* untuk *target* DBMS".

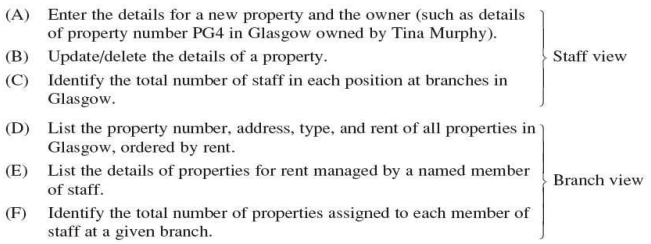
2) Design file organizations and indexes

Connolly & BEGG (2010 : 528 - 529) mengatakan bahwa, tujuan dari tahap ini adalah "Menentukan *file organizations* yang optimal untuk menyimpan relasi dasar dan index yang diperlukan demi mencapai kinerja yang sesuai, yaitu bagaimana cara dari *relations* dan *tuples* dapat disimpan dalam *secondary storage"*.

a) Analyze transactions

Connolly & BEGG (2010 : 529 - 534) mengatakan bahwa, tujuan dari tahap ini adalah "Memahami transaksi secara fungsional yang akan dijalankan dalam *database* dan untuk menganalisis transaksi yang penting". Untuk membantu mengidentifikasi transaksi mana yang perlu diteliti, dapat menggunakan *transaction relation cross-refrence matrix* yang menunjukkan relasi setiap akses transaksi. Untuk fokus pada area yang mungkin bermasalah, salah satu cara untuk melanjutkan :

1. *Map* seluruh jalan transaksi untuk relasi



Gambar 2.23 Map path
Sumber (Connolly & BEGG, 2010, p. 530)

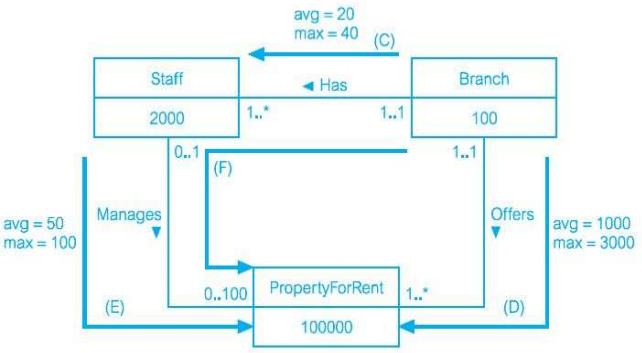
Table 17.1 Cross-referencing transactions and relations.

Transaction/ Relation	(A)				(B)				(C)				(D)				(E)				(F)			
	I	R	U	D	I	R	U	D	I	R	U	D	I	R	U	D	I	R	U	D	I	R	U	D
Branch										X				X								X		
Telephone																								
Staff		X				X				X								X				X		
Manager																								
PrivateOwner	X																							
BusinessOwner	X																							
PropertyForRent	X					X	X	X						X				X				X		
Viewing																								
Client																								
Registration																								
Lease																								
Newspaper																								
Advert																								

I = Insert; R = Read; U = Update; D = Delete

Gambar 2.24 Cross referencing transactions and relations
Sumber (Connolly & BEGG, 2010, p. 531)

2. Tentukan relasi mana yang sering diakses oleh transaksi



Gambar 2.25 Frequently accessed by transactions
Sumber (Connolly & BEGG, 2010, p. 532)

3. Analisis penggunaan data dari transaksi yang dipilih yang melibatkan transaksi lain

   Frekuensi informasi akan mengidentifikasi *relation* yang perlu pertimbangan cermat untuk memastikan struktur akses yang tepat untuk digunakan. Kondisi pencarian digunakan oleh transaksi yang memiliki *time constraints* menjadi prioritas yang lebih tinggi untuk struktur akses.

b) Choose file organizations

Connolly & BEGG (2010 : 534) mengatakan bahwa, tujuan dari tahap ini adalah "untuk menentukan *file organizations*yang efisien untuk setiap relasi dasar". Beberapa tipe *file organizations* yaitu:

1. MyISAM

   MyISAM adalah engine standar atau default dari MySQL, kelebihan dari engine ini adalah proses yang lebih cepat dalam proses select, sehingga engine ini sering sekali digunakan sebagai engine dari tabel acuan atau tabel yang jarang sekali berubahnya. Kelebihan MyIsam antara lain:

   a. Adanya suatu kode pembeda yang akan memberi tanda bila suatu tabel tidak ditutup dengan semestinya setelah dibuka. Bila kita menjalankan program MySQLServer, mysqld, dengan pilihan myisam-recover, maka secara otomatis tabel yang telah ditandai tersebut akan dipulihkan atau ditutup dengan semestinya. Hal ini sangat membantu untuk menjaga keutuhan dan validitas tabel.
   b. Kemampuan kolom AUTO_INCREMENT lebih handal dibandingkan dengan format tabel ISAM
   c. Mendukung *file* ukuran besar (63-bit) pada sistem operasi tertentu yang jugamendukung pembuatan dan pembacaan *file-file* data ukuran besar
   d. Pada kolom BLOB dan TEXT bisa dilakukan pengindeksan NULL diperkenankan pada kolom yang diindeks
   e. Jumlah maksimum *file* indeks adalah 32 buah per tabelnya, dan masih dapat dikembangkan hingga mencapai 64 buah per tabel dengan kondisi tertentu tanpa harus mengkompilasi ulang program myisamchk.
   f. Program bantu myisampack dapat memadatkan kolom BLOB dan VARCHAR

2. InnoDB

   Tipe tabel InnoDB merupakan tipe tabel MySQL yang mendukung proses transaksi. Tipe ini memiliki beberapa keunggulan, antara lain format tabel InnoDB mendukung proses transaksi dengan adanya fasilitas rollback, commit, dan juga kemampuan untuk memulihkan tabel bila terjadi

kerusakan pada tabel tersebut. Mampu melakukan penguncian (locking) pada tingkatan record dan juga mampu membaca pada perintah SELECT yang tidak dikunci. Kemampuan-kemampuan tersebut meningkatkan kecepatan dan kinerja penggunaan *multiuser*.

3. Heap

   Tabel dengan tipe HEAP tidak menyimpan datanya di hardisk, tetapi menyimpan di RAM (memori). Tipe tabel ini biasanya digunakan sebagai tabel sementara (temporary). Tabel secara otomatis akan dihapus (hilang) dari MySQL saat koneksi ke server diputus atau server MySQL dimatikan.

c) Choose indexes

Connolly & BEGG (2010 : 535) mengatakan bahwa, tujuan dari tahap ini adalah "Menentukan apakah penambahan index akan meningkatkan kinerja dari sistem". Pemilihan *attribute* untuk *primary* atau *clustering* adalah sebagai berikut :

1) *Attribute* yang banyak digunakan untuk operasi *join*, yang membuat operasi *join* semakin efisien
2) *Attribute* yang banyak digunakan untuk mengakses tupel dalam sebuah relasi yang memiliki *attribute* tersebut

d) Estimate disk space requirements

Connolly & BEGG (2010 : 541) mengatakan bahwa, tujuan dari tahap ini adalah "Memperkirakan jumlah kapasitas *disk* yang akan diperlukan oleh *database* sehingga dapat diperkirakanpenggunaan kapasitas *disk* setiap tahunnya".

3) Design user views

Connolly & BEGG (2010 : 542) mengatakan bahwa, tujuan dari tahap ini adalah "Merancang tampilan penggunayang sudah diidentifikasi selama pengumpulan kebutuhan dan analisis tahap dari pengembangan siklus kehidupan sistem *database*.

4) Design security mechanisms

Connolly & BEGG (2010 : 542 - 543) tujuan dari tahap ini adalah "Merancang mekanisme *security* untuk *database* seperti yang telah

dispesifikasikan oleh *user* selama tahap pengumpulan dan analisis kebutuhan dari pengembangan siklus kehidupan sistem *database*".

2.1.10 *Unified Modelling Language* (UML)

Menurut Britton &Doake (2005 : 13), *Unified Modeling Language (UML)* adalah satu kumpulan diagram yang dirancang secara khusus untuk pengembangan berorientasi objek, dan telah menjadi standar industri untuk pemodellan sistem berorientasi objek.

1. *Use Case Diagram*

   Menurut Britton & Doake (2005 : 40) *use case diagram* adalah diagram yang secara grafis menggambarkan interaksi antara *user* secara fungsional dengan sistem.

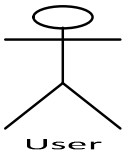
   a. *Use case*

      Use case digambarkan dalam bentuk elips dengan label nama *use case*. Penulisan nama *use case* menggunakan kata kerja yang menegaskan bahwa *use case* mewakili proses.

Gambar 2. 26 *Use Case*

   b. *Actor*

      Digambarkan dengan bentuk tongkat diberi label nama *actor*. Tujuannya adalah untuk memanfaatkan nama *actor* tersebut agar mudah untuk mengidentifikasikannya. Bentuk tersebut juga digunakan untuk *actor* yang bukan manusia, misalnya komputer.



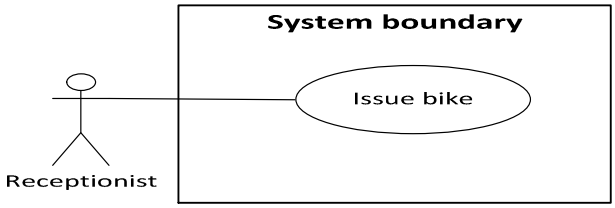
Gambar 2. 27 *Actor*

   c. *Relationship*

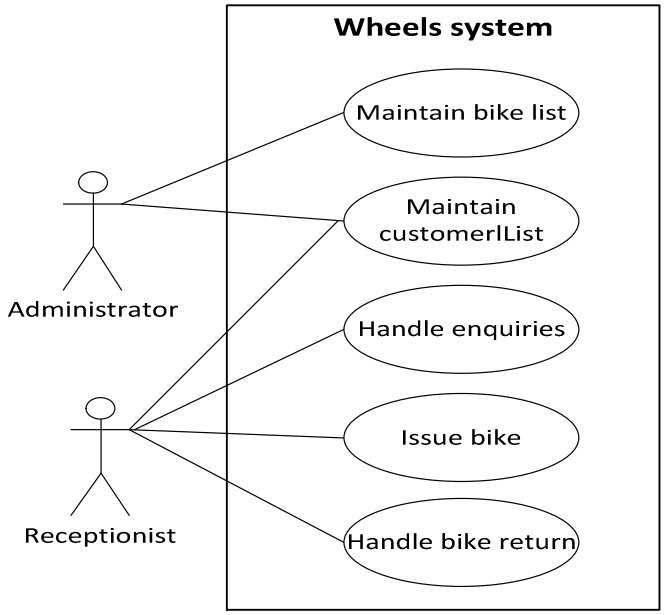
      Garis yang menghubungkan *actor* dengan *use case*. Garis tersebut menunjukkan *actor* yang berkaitan dengan *use case* yang digunakan. Hubungan ini dikenal juga sebagai asosiasi komunikasi.

d. *System boundary*

Garis yang digambarkan mengitari *use case* untuk memisahkan *use case* dengan *actor*. Dapat diberi label untuk mengindikasikan *domain diagram*.



Gambar 2. 28 *System Boundary*



Gambar 2. 29 *Use case diagram*

Sumber Britton & Doake (2005:41)

2. Class Diagram

Britton & Doake (2005 : 117-137) menyatakan *class diagram* adalah sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek. *Class* menggambarkan keadaan (*attribute*) suatu sistem dan menawarkan layanan untuk memanipulasi keadaan tersebut (metoda atau fungsi). *Class diagram* menggambarkan struktur dan deskripsi *class*, *package* dan objek beserta hubungan satu sama lain seperti pewarisan, asosiasi, dan lain-lain. *Class* memiliki tiga area pokok, yaitu :

a. Nama, merupakan nama dari sebuah *class*.

b. *Attribute*, merupakan properti dari sebuah *class*. *Attribute* melambangkan batas nilai yang mungkin ada pada objek dari *class*.
c. Metoda, merupakan sesuatu yang dapat dilakukan oleh sebuah *class*atau yang dapat dilakukan oleh *class* lain terhadap sebuah *class.*

Berikut merupakan notasi dari *class diagram*:

a. *Class*

   Class adalah blok-blok pembangun pada pemrograman berorientasi objek. Sebuah *class* digambarkan pada sebuah kotak yang terbagi atas tiga bagian, yaitu: nama *class*, definisi *attribute* dan definisi fungsi.

b. *Association*

   Sebuah asosiasi merupakan sebuah hubungan paling umum antara dua *class*. Garis ini dapat melambangkan tipe-tipe hubungan dan juga *multiplicity* pada sebuah hubungan seperti *one to one, one to many*, *many to many*.

c. *Composition*

   Jika sebuah *class* tidak dapat berdiri sendiri dan merupakan bagian dari *class* yang lain, maka *class* tersebut memiliki relasi *composition* terhadap *class* tempat *class* tersebut bergantung. Sebuah hubungan *composition* digambarkan sebagai garis dengan ujung berbentuk jajaran genjang berisi atau *solid*.

d. *Dependency*

   Hubungan *dependency* dilakukan apabila sebuah *class* menggunakan *class* yang lain. Penggunaan *dependency* dilakukan untuk menunjukkan operasi pada suatu *class* yang menggunakan *class* yang lain. Sebuah *dependency* dilambangkan sebagai sebuah panah bertitik-titik.

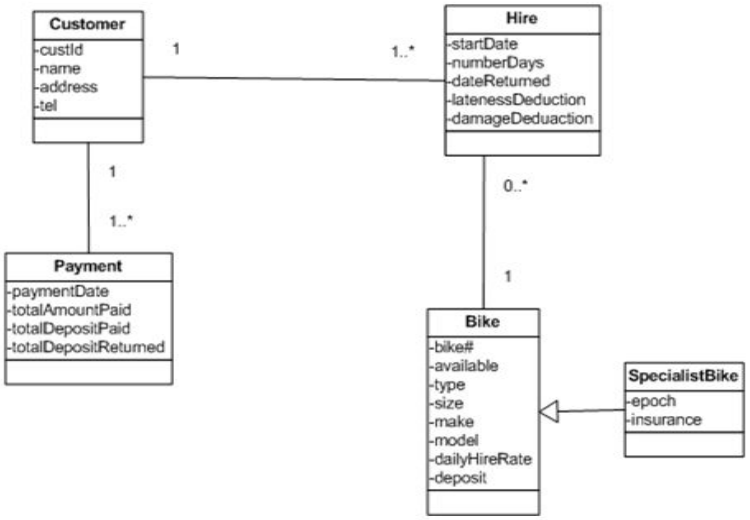
e. *Aggregation*

   Agregasi mengindikasikan keseluruhan bagian *relationship* dan biasanya disebut sebagai relasi (mempunyai sebuah / bagian dari). Sebuah agregasi dilambangkan sebagai sebuah garis dengan sebuah jajaran genjang yang tidak berisi atau tidak *solid.*

f. *Generalization*

   Sebuah relasi generalisasi sepadan dengan sebuah relasi *inheritance* pada konsep berorientasi objek. Generalisasi dilambangkan dengan

sebuah panah dengan kepala panah yang tidak solid yang mengarah ke kelas induk.



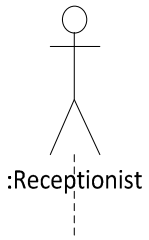
Gambar 2.30 Class diagram
Sumber (Britton & Doake, 2005, p. 131)

3. *Sequence Diagram*

Menurut Britton & Doake (2005 : 156), *sequence diagram* menggambarkan dengan jelas dan sederhana aliran kontrol antar objek yang diperlukan untuk melaksanakan skenario. Sebuah skenario menguraikan urutan langkah-langkah dalam satu contoh *use case* dari pengguna. Dari sisi layar komputer, *sequence diagram* menunjukkan bagaimana langkah-langkah tersebut diterjemahkan kedalam pesan antar objek di komputer. Berikut ini merupakan elemen yang digunakan :

a. *Actor*

Digambarkan dengan simbol *actor* yang ada didalam diagram *use case*, yang digunakan untuk mewakili *user*.



Gambar 2. 31 *Actor*

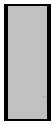
b. *Object lifeline*

Garis vertikal putus-putus yang memanjang kebawah dari *symbol actor* dan sistem yang mengindikasikan urutan kehidupan.

Gambar 2. 32 Lifeline

c. Activation bar

Bar didalam *lifeline* yang menunjukkan periode waktu ketika *actor* aktif dalam interaksi.



Gambar 2. 33 *Activation Bar*

d. Input Message

Anak panah *horizontal* yang berasal dari *actor* menuju objek yang mengindikasikan pesan masuk.

Gambar 2. 34 *Input Message*

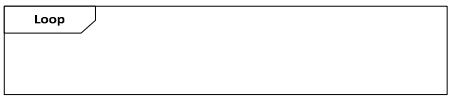
e. *Output message*

Garis anak panah *horizontal* putus-putus yang berasal dari objek menuju *actor*.

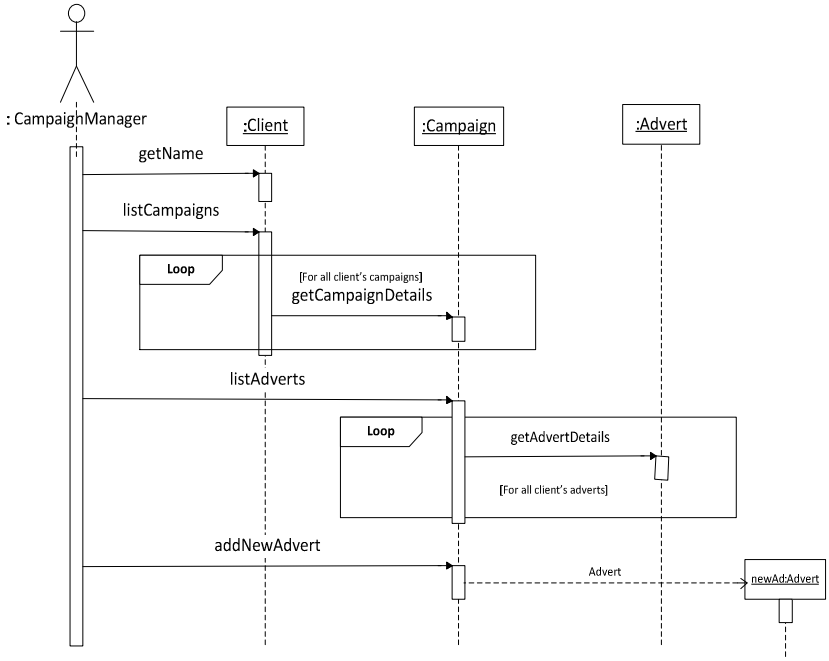
Gambar 2. 35 *Output Message*

f. Interaction operator

Menspefikasikan tipe dari *combined fragment(loop,alt,opt)*.



Gambar 2. 36 *Interaction Operator*



Gambar 2. 37 *Sequence diagram*

4. *State Diagram*

   Britton & Doake (2005 : 181-184) mengatakan bahwa, *state diagram* menunjukkan bagaimana objek dapat berubah dari satu keadaan ke keadaan yang lain dalam menanggapi peristiwa eksternal. *State diagram* biasanya model dari sebuah objek yang spesifik. Notasi yang ada di dalam *state diagram* :

   a. *States*

      Diwakili oleh kotak-kotak dengan sudut yang membulat.

Gambar 2. 38 *States*

   b. *Transition*

      Diwakili oleh anak panah yang padat antara *state-state* diberi label dengan *'EventName/action'* (*event* memicu transisi dan *action* adalah hasil dari transisi).

Gambar 2. 39 *Transition*

c. *Start tstate*

Keadaan objek sebelum transisi, yang mana diwakili oleh lingkaran padat dengan anak panah ke *state* awal.

Gambar 2. 40 *Start State*

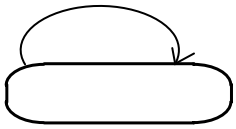
d. *Stop state*

State yang menandakan penghancuran objek, yang mana diwakili oleh lingkaran padat dengan lingkaran sekitarnya dan panah datang dari *state* sebelumnya.

Gambar 2. 41 *Stop State*

e. Self transition

Diwakili oleh kotak-kotak dengan sudut yang membulat dan anak panah yang kembali ke *state* itu sendiri.



Gambar 2. 42 Self Transition

f. Event[guard] / action

Label dari suatu *transition.*

2.1.11 *Internet*

Williams & Sawyer (2007 : 17) mengatakan bahwa, "*internet* adalah jaringan komputer di seluruh dunia yang menghubungkan ratusan bahkan ribuan jaringan yang lebih kecil, misalnya jaringan pendidikan, komersial, nirlaba, dan militer, bahkan jaringan individual". Kemudian terdapat juga komponen penting *internet* yang berupa media yang biasa disebut dengan www (*World Wide Web*), www memegang peranan penting dalam mempopulerkan *internet*, www atau *web* dapat didefinisikan sebagai sistem interkoneksi *computerinternet* (disebut server) yang mendukung dokumen dokumen bersifat multimedia. Agar dapat menjelajah *web*, perlulah perangkat lunak yang biasa disebut dengan *browser.* Menurut Williams & Sawyer (2007

: 64-65), “browser atau *web* browser adalah perangkat lunak yang memungkinkan anda mencari dan mengakses beragam komponen *web*. Eskplorasi *web* ini atau biasa disebut dengan *surfing* memungkinkan kita untuk dapat berpindah pindah dalam serangkaian jalur yang terkoneksi, atau disebut link dari lokasi yang satu, atau situs *web*, ke lokasi yang lain”. *Website* (situs *web*) atau biasa disebut situs adalah sebuah lokasi pada *computer* tertentu di *web* yang memiliki alamat unik (disebut URL), sedangkan *webpage* (halaman *web*) merupakan dokumen *pada world wide web* yang bisa berisi teks, gambar, suara, dan video.

2.1.12 *Eight Golden Rules*

Menurut Shneiderman & Plaisant(2010 : 88), terdapat delapan aturan emasdalam merancang sebuah *userinterface* untuk pengguna, yaitu:

1. Berusaha keras untuk konsisten

   Aturan ini adalah yang paling sering dilanggar, tapi nantinya bisa rumit karena ada banyak bentuk konsistensi. Konsisten urutan tindakan harus diperlukan dalam situasi yang sama; istilah yang sama harus digunakan dalam petunjuk, menu, dan *helpscreen*, dankonsisten warna, tata letak, kapitalisasi, *font*, tampilan, dan lain-lain.

2. Melayani penggunaan universal

   Memungkinkan tersedianya shortcut / fungsi yang pasti akan digunakan semua pengguna apabila pengguna sudah memasuki tahap ahli pada suatu aplikasi / terbiasa dengan aplikasi sehingga dapat mengurangi jumlah interaksi dan meningkatkan kecepatan interaksi.

3. Memberikan umpan balik yang informatif

   Untuk setiap aksi yang dilakukan oleh pengguna, beberapa di antaranya harus mempunyai sistem *feedback*. Untuk aksi yang sering dan sederhana, maka respon yang diberikan juga sederhana, tetapi jikaaksi yang jarang dan besar maka respon juga harus lebih banyak dan rinci.

4. Merancang dialog untuk menghasilkan keadaan akhir

   Urutan aksi harus dibagi menjadi awal,tengah, dan akhir. Dengan adanya umpan balik,pengguna dapat merasa lebih merasa aman dalam melakukan sebuah tindakan dengan memberikan gambaran hasil akhir dari suatu pilihan, sertapemberian banyak *pilihan* kepada pengguna sehingga bisa

ikut serta dalam mempengaruhi hasil akhir. Contoh, sebuah situs *webe-commerce* memindahkan pengguna dari pemilihan produk ke kasir, berakhir dengan sebuah halaman konfirmasi yang jelas untuk melengkapi transaksi.

5. Menghindari kesalahan

   Suatu sistem harus dirancang agar kesalahan yang dibuat pengguna dapat ditekan seminimal mungkin, dan pesan kesalahan yang dimunculkan harus dapat dimengerti oleh pengguna awam. Jika pengguna membuat kesalahan, antarmuka harus mendeteksi kesalahan dan menawarkan instruksi yang sederhana, membangun, dan spesifik untuk mengatasi kesalahan tersebut. Contoh, pengguna seharusnya tidak perlu mengetik ulang bentuk, seperti nama, alamat, tanggal lahir, jenis kelamin jika mereka memasukkan kode pos yang tidak valid, melainkan harus dibimbing untuk memperbaiki hanya bagian yang rusak, yaitu kode pos.

6. Mengijinkan pembalikan aksi (*undo*) dengan mudah

   Aksi harus dapat dibalikkan menjadi keadaan sebelumnya sehingga membuat pengguna merasa aman karena ia tahu bahwa kesalahan yang dibuat dapat diperbaiki.

7. Mendukung pengguna sebagai pusat kendali dari sistem

   Membuat pengguna merasa memegang kendali atas sistem tersebut,bukan sebagai responden. Kesulitan pengguna dalam menavigasi *site*, kesulitan dalam mendapatkan data yang diinginkan, antarmuka yang mengejutkan, ketidakmampuan untuk menghasilkan tindakan yang diinginkan akan menimbulkan rasa ketidakpuasan.

8. Mengurangi beban ingatan jangka pendek

   Manusia hanya dapat mengingat tujuh info ditambah atau dikurang dua info pada suatu waktu.Batasan ini berarti suatu sistem harus dibuat sesederhana mungkin sehingga tidak membuat seorang pengguna bingung karena terlalu banyak info. Dan apabila diperlukan, akses *online* ke bentuk perintah sintaks, singkatan, kode, dan informasi lainnya harus disediakan fasilitasnya.

2.1.15 Faktor manusia terukur

Menurut Shneiderman & Plaisant (2010 : 32) untuk membuat sistem perancangan antarmuka yang efisien, efektif dan memuaskan, ada faktor-faktor pengukur yang dijadikan sebagai evaluasi, yaitu :

1. Waktu pembelajaran

   Lamanya waktu yang diperlukan *user* untuk mempelajari cara penggunaan aksi yang berhubungan dengan tugas.

2. Kecepatan kinerja

   Lamanya waktu yang diperlukan *user* untuk menyelesaikan tugas.

3. Tingkat kesalahan yang dibuat *user*

   Tingkat banyaknya kesalahan dan jenis kesalahan apa yang dilakukan oleh *user*. Pengendalian kesalahan merupakan komponen kritis dalam pembuatan *interface.*

4. Daya ingat

   Bagaimana *user* dapat mempertahankan daya ingat mereka mengenai *interface* setelah jangka waktu tertentu. Frekuensi penggunaan *interface* akan meningkatkan daya ingat.

5. Tingkat kepuasan

   Tingkat kepuasan *user*akan beberapa segi *interface* yang dapat diketahui dengan melakukan kuesioner dan *interview*.

2.2 Teori yang terkait tema penelitian

2.2.1 *E-commerce*

Menurut Jr & Schell (2008 : 53-55) menyatakan bahwa perdagangan elektronik (*electronic - commerce*) adalah proses pembelian dan penjualan barang secara elektronik melalui transaksi bisnis ter-komputerisasi menggunakan *internet* atau teknologi jaringan digital lainnya. Menurut Gaol (2008 :275) ada 3 jenis utama *e-commerce* yaitu *busines-to-cosumer* (B2C), *Business-to-business* (B2B), dan *consumer-to-consumer* (C2C). Menurut Jr & Schell, (2008 : 59-61) definise sempit dari *e-commerce* hanya meliputi transaksi bisnis yang berhubungan dengan pelanggan dan pemasok, yang menghubungkan *computer* mereka masing – masing melalui *internet*, sedangkan definisi luas *e-commerce* adalah dapat memfasilitasi operasi internal maupun eksternal dari suatu perusahaan. Operasi perusahaan secara

internal dilaksanakan dalam batas perusahaan, oleh bidang bisnis keuangan, sumber daya manusia, layanan informasi, produksi, pemasaran, dan lain lain dengan menggunakan akses jaringan berbasis *computer* dan antarmuka sebuah browser *web*, sedangkan e - commerce di luar batas perusahaan dibagi menjadi 2 jenis yaitu, *e-commerce* bisnis ke konsumen (*business-to-consumer*—B2C) dimana mengacu pada transaksi yang terjadi antara sebuah bisnis dan konsumen akhir produk, dan bisnis ke bisnis (*business-to-business* - B2B) mengacu pada transaksi bisnis dimana pihak terlibat dalam bisnis tersebut tidak menjadi konsumen akhir (*end user*). Manfaat yang diharapkan dari *e-commerce* adalah

1. Perbaikan layanan pelanggan sebelum dan setelah penjualan
2. Perbaikan hubungan dengan pemasok dan komunitas keuangan
3. Peningkatan imbal hasil ekonomis atas pemegang saham dan investasi pemilik

Menurut Jr & Schell(2008 : 60) bahwa salah satu strategi yang terbaik dalam melakukan *e-commerce* adalah sitem antaroganisasi. Sistem antarorganisasi adalah sistem yang memperbolehkan perusahaan tersebut bekerja seperti suatu unit yang terkoordinasi agar tujuan-tujuan tertentu dapat tercapai oleh masing-masing perusahaan, dimana tadinya tidak bisa dicapai oleh suatu perusahaan. Perusahaan-perusahaan yang terlibat disebut dengan partner dagang, partner bisnis atau bisa diseuat aliansi bisnis. Sistem tersebut merupakan dasar dari *e-commerce* dikarenakan dalam *e-commerce* ada pertukaran data dalam jumlah yang besar dan harus cepat dan aman, hal-hal tersebut merupakan komponen penting yang menunjang produktivitas dari *e-commerce*.

Menurut Dewara, Nugraha, & Fachriz(2012), keuntungan dari *e-commerce* dapat dilihat dari berbagai sudut pandang, diantaranya :

1. Keuntungan *e-commerce* buat organisasi
    a. Jangkauan global
    b. Pengurangan biaya produksi, proses, distribusi dan penyimpanan
    c. Perbaikan rantai suplai
    d. Bisnis akan selalu buka di *web*, tanpa pertambahan waktu dan biaya
    e. Cara berbisnis yang baru, melalui *e-commerce* dapat memberikan keuntungan strategis dan meningkatkan laba

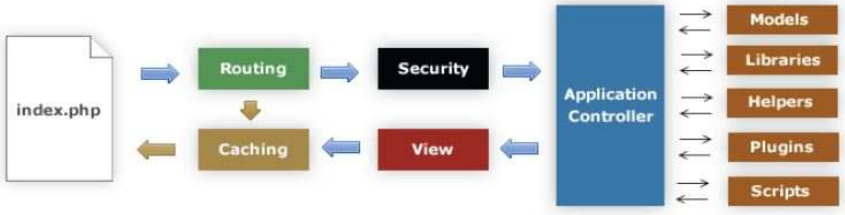
2. Keuntungan *e-commerce* buat konsumen
   a. Memberikan konsumen untuk berbelanja atau melakukan transaksi lainnya sepanjang tahun, 24 jam sehari
   b. Menyediakan banyak pilihan bagi konsumen
   c. Dalam kasus produk digital, *e-commerce* memungkinkan untuk pengiriman lebih cepat
   d. Pada banyak negara, bisnis *online* dibebaskan dari pajak penjualan
3. Keuntungan *e-commerce* buat masyarakat
   a. Berkurangnya polusi udara karena dengan *e-commerce* pembeli dapat melakukan transaksi tanpa harus datang ke toko.
   b. Orang yang tinggal di wilayah pedesaan kini dapat menikmati produk dan layanan yang tersedia di masa lalu
   c. Pelayanan public, seperti perawatan kesehatan, pendidikan dan distribusi pelayanan social pemerintah, dapat dilakukan dengan mengurangi biaya dan meningkatkan kualitas.

2.2.2 Bisnis

Ebert, Ronald J., Griffin, Ricky W. (2009 :4) mengatakan bahwa, “bisnis adalah organisasi yang menyediakan barang-barang atau pelayanan yang kemudian dijual untuk mendapatkan keuntungan”.

2.2.3 Code Igniter

Code igniter adalah *model view controller* (MVC) *framework* berbasis PHP yang membantu anda dalam struktur kode dan membuat tugas yang berulang tidak membosankan. *CodeIgniter* tidak memaksa konvensi apapun tetapi menyediakan banyak fitur yang umumnya diperlukan melalui serangkaian yang dibangun pada perpustakaan.



Gambar 2.43 Code igniter flow
Sumber :(Code Ignitier, p:7)

Penjelasan dari gambar diatas :

1. Index.php berfungsi sebagai pengatur bagian depan, menginisialisasi sumber daya dasar yang dibutuhkan untuk menjalankan *CodeIgniter*.
2. Router mengkaji permintaan HTTP untuk menentukan apa yang harus dilakukan dengan hal itu.
3. Jika sebuah *filecache* ada, ia akan dikirim langsung ke *browser*, melewati eksekusi sistem normal.
4. *Security*, sebelum aplikasi *controller* dimuat, permintaan HTTP dan setiap data pengguna yang dikirim akan disaring untuk keamanan.
5. *Controller* memuat model, *libraries, plugin, helpers*, dan sumber daya lainnya yang dibutuhkan untuk memproses permintaan khusus.
6. Tampilan akhir yang diberikan kemudian dikirim ke *web* browser untuk dilihat. Jika *caching* diperbolehkan, maka halaman view tersebut akan di simpan dalam *cache* dan kemudian jika halaman tersebut diakses kembali, maka tinggal mengaksesnya dari *cache*.

2.2.4 MySQL

Menurut Weiling & Thomson(2005 : 3) MySQL adalah "*relational database management system* yang sangat cepat dan kokoh. *Database* memperbolehkan untuk menyimpan, mencari, mengurutkan dan mendapatkan data secara efisien". MySQL *server* mengontrol akses ke data dan menjamin bahwa banyak *user* dapat mengakses secara bersamaan, menyediakan akses yang cepat dan menjamin hanya *user* yang berwenang mendapatkan akses.

2.2.5 PHP Hypertext Processor (PHP)

Weiling & Thomson (2005 : 2) mengatakan bahwa, "PHP adalah bahasa *scripting server side* yang dirancang khusus untuk *web*". Kode PHP akan diterjemahkan pada *web server* dan menghasilkan HTML atau output lainnya yang akan dilihat oleh pengunjung *web*. PHP adalah produk *open source* dimana pengguna mempunyai akses ke *source code*. Kode PHP dapat digunakan secara bebas, dirubah dan disebarkan tanpa biaya.

2.2.6 *Enquiry*

Menurut Dictionary, *enquiry* adalah sebuah pencarian atau permintaan untuk informasi atau pengetahuan dan kegiatan bertanya atau mencari informasi dengan cara bertanya.

2.2.7 *Framework*

Kumpulan pustaka-pustaka (*library*) perangkat lunak yang *script*-nya dapat digunakan kembali (*reusable*) yang terbungkus menjadi sebuah API. API (Application Programming Interface), sebuah perangkat lunak yang menyediakan layanan untuk interaksi antar perangkat lunak.

2.2.8 *Asynchronous JavaScript and XML* (AJAX)

Menurut (AJAX : W3Schools, 1999), AJAX adalah seni bertukar data dengan server, dan memperbarui bagian dari halaman *web* tanpa *reload* seluruh halaman. AJAX bukanlah bahasa pemrograman baru, tapi cara baru untuk menggunakan standar yang ada.

2.2.9 JQUERY

Menurut (jQuery : W3Schools, 1999), JQUERY adalah library javascript. jQuery membutuhkan banyak tugas umum yang membutuhkan banyak baris kode JavaScript untuk mencapai, dan membungkus mereka menjadi metode yang dapat anda panggil dengan satu baris kode.

2.2.10 Javascript

Menurut (JAVASCRIPT : W3Schools, 1999), JavaScript adalah bahasa pemorgraman yang dapat dimasukkan ke halaman HTML. Javascript merupakan bahasa pemrograman yang ringan.

2.3 Hasil Penelitian atau Produk Sebelumnya

2.3.1 Website Sejenis

Banyak *website* yang menjual dokumen-dokumen penting, yaitu :

1. www.legalakses.com

   Web ini menjual dokumen-dokumen legal yang sering dipakai di Indonesia, seperti surat perjanjian, surat kuasa, berita acara, surat pernyataan, surat resmi dan surat gugatan. Selain menjual dokumen dan

peraturan-peraturan yang berlaku di Indonesia, *web* ini juga menyediakan informasi yang berguna bagi *user*nya, seperti daftar alamat kantor pengacara, daftar kantor pengadilan di jabodetabek, daftar alamat pengadilan militer dan daftar alamat pengadilan PTUN.

Nilai positif dari *web* ini :

a. Search menurut *title* (apa yang *userinput*)
b. *User* bisa melihat *preview* dari dokumen yang akan dibeli

Nilai negatif dari *web* ini :

a. Sistem transaksi yang tidak aman, karena menambahkan nilai 3 digit nomor telepon *user* pada jumlah transfer ada kemungkinan sama. Sama dalam arti dokumen yang dibeli sama dan 3 digit nomor teleponnya sama.
b. Tidak ada pembedaan *user* (*customer / member*)

2. www.bkpm.go.id

Badan koordinasi penanaman modal adalah lembaga pemerintahan *non* departemen Indonesia yang bertugas untuk merumuskan kebijakan pemerintah di bidang penanaman modal, baik dari dalam negeri maupun luar negeri.

Fitur yang ada di dalam *web* ini :

a. Tutorial
b. Pencarian regulasi khusus indonesia
c. Data *statistic* terbaru
d. Pendaftaran secara langsung di *website*
e. Referensi ke beberapa *website* investasi
f. Dapat menampung dokumen yang akan kita baca
g. Ada beberapa bahasa
h. Melihat progress aplikasi

3. www.sciencedirect.com

ScienceDirect adalah *database* ilmiah lengkap terkemuka yang menawarkan artikel jurnal dan bab buku dari lebih dari 2.500 jurnal dan lebih dari 11.000 buku.

Fitur yang ada dalam *web* ini :

a. Search by title
b. Search by subject

c. *Newest articles*
d. *Subscription's alert*
e. *Subsription*
f. *Add to cart (Shopping cart)*
g. *Advanced Search & Expert Search*

2.3.2 Jurnal

Banyak jurnal yang membahas mengenai e-commerce, yaitu :

1. Irmawati (2011 : 13-14) mengatakan bahwa, "dalam proses *e-commerce*, perusahaan membutuhkan beberapa komponen utama agar operasi dan manajemen aktivitas *e-commerce* berjalan dengan baik. Komponen-komponen pokok yang memiliki peran penting dalam proses *e-commerce* dunia usaha tampak seperti pada gambar berikut:
   a. Pengendalian akses dan keamanan, situs *e-commerce* harus memberikan rasa percaya dan akses yang aman untuk berbagai pihak dalam transaksi *e-commerce*, misalkan dengan adanya kata kunci *(password)*, kunci enkripsi, sertifikasi, atau tanda tangan digital. Kemudian ada otorisasi akses yang hanya ke bagian tertentu saja sehingga hanya para pelanggan yang terdaftar saja yang dapat mengakses informasi dan aplikasi yang ada. Pengendalian akses dan keamanan ini perlu dilakukan untuk melindungi sumber daya situs *e-commerce* dari berbagai ancaman seperti peretas (*hacker)*, pencurian *password*atau nomor kartu kredit, atau menghindari kegagalan sistem.
   b. Membuat profil dan personalisasi**,** proses pembuatan profil dan personilasasi menggunakan alat pembuat profil seperti pendaftaran, *file cookie*, *software* penelusur perilaku dalam situs *web* dan respon pemakai. Profil ini digunakan untuk mengenali kita sebagai pemakai individual, memberikan tampilan personalisasi, saran atas produk dan iklan *web*. Tujuan proses pembuatan profile ini untuk tujuan manajemen rekening, pembayaran, mengumpulkan data mengenai manajemen hubungan pelanggan, perencanaan pemasaran, dan untuk manajemen situs *web* itu sendiri.
   c. Manajemen pencarian**,** *software e-commerce* harus meliputi komponen mesin pencari situs *web* untuk dapat membantu para pelanggannya

dalam menemukan produk dan jasa tertentu yang mereka inginkan untuk dievaluasi atau dibeli

d. Manajemen isi dan katalog**,** isi *e-commerce* sebagian besar berbentuk katalog multimedia yang memuat informasi produk sehingga membuat dan mengelola catalog merupakan rangkaian utama dari manajemen isi. *Software* manajemen isi tersebut bekerja dengan alat pembuat profile yang sudah disebutkan sebelumnya. Software manajemen isi akan membantu perusahaan *e-commerce* untuk mengembangkan, menghasilkan, mengirimkan, memperbaharui, dan menyimpan data teks serta informasi multimedia di situs *web e-commerce*. Selanjutnya manajemen isi dan katalog dapat diperluas untuk memasukkan proses konfigurasi produk yang akan mendukung layanan mandiri berbasis *web* dan penyesuaian massal atas berbagai produk perusahaan.
e. Manajemen arus kerja, sistem arus kerja *e-business* digunakan untuk membantu para karyawan secara elektronik bekerja sama untuk menyelesaikan tugas pekerjaan dengan menggunakan mesin *software* arus kerja *(workflow software engine).* Sistem ini memastikan bahwa transaksi, keputusan, dan aktivitas kerja yang tepat dilakukan, serta data dan dokumen yang benar telah dikirimkan ke para karyawan, pelanggan, pemasok, dan pihak *stakeholder*.
f. Pemberitahuan kegiatan**,** proses pemberitahuan kegiatan *(event notification)* memainkan peranan penting dalam sistem *e-commerce* karena sistem ini digunakan untuk memonitor semua proses *e-commerce* dan mencatat semua kegiatan yang relevan, termasuk perubahan mendadak atau ketika dalam masalah. Sistem ini akan memberitahukan kepada para pelanggan, pemasok, dan pegawai serta *stakeholder* mengenai semua kegiatan transaksi yang berkaitan dengan status mereka dengan melalui pesan elektronik seperti *e-mail*, *newsgroup*, penyeranta *(pager),* atau *fax*.
g. Kerjasama dan perdagangan**,** Tujuan utama *e-commerce* adalah untuk mendukung kesepakatan kerjasama dan layanan perdagangan yang dibutuhkan oleh para pelanggan, pemasok, dan *stakeholder* lainnya. Seperti halnya dalam *e-business*, sistem *e-commerce* juga fokus

menumbuhkan komunitas berkepentingan *online* untuk meningkatkan layanan pelanggan dan membangun loyalitasnya.

h. Proses pembayaran elektronik**,** pembayaran sebagai proses nyata dan penting dalam transaksi *e-commerce*. Sekarang ini sebagian besar sistem *e-commerce* yang terlibat dalam *web* dan bisnis B2C menggunakan proses pembayarannya dengan kartu kredit ".

2. Ahmadi (2011 : 15-16) mengatakan bahwa, "dampak perumusan pemasaran *e-commerce* sebagai berikut :
   a. Promosi *e-commerce* dapat mempertinggi produk dan layanan melalui kontak langsung, kaya informasi dan interaksi dengan pelanggan.
   b. Saluran pemasaran baru menciptakan saluran distribusi bagi produk yang ada sehingga banyak peluang menjangkau pelanggan dengan sifat komunikasi secara langsung dan dua arah.
   c. Penghematan langsung dalam pengiriman informasi kepada pelanggan.
   d. Pengurangan *cycle time*, pengiriman produk dan pelayanan digital dapat dikurangi hingga hanya dalam hitungan detik untuk sampai ke tujuan.
   e. Layanan konsumen ditingkatkan dengan cara pelanggan menemukan informasi detail secara *online*
   f. Citra merek perusahaan, dalam *web* pendatang baru bisa membangun citra perusahaan dengan cepat ".
3. Vayghan (2012 : 1, 4) mengatakan bahwa, "Kesusksesan dari operasi *e-business* sangat bergantung kepada data dan informasi, serta bagaimana data dan informasi tersebut dapat digunakan untuk mengoptimalkan operasional perusahaan, mendorong proses *sales* dan *marketing*, serta membuat bisnis menjadi semakin berkembang. Kemampuan untuk dapat mengolah dan melindungi data sebagai aset strategi, sehingga dapat diubah menjadi informasi yang dapat dijalankan dan menjadi *key contributor* terhadap kesuksesan suatu operasi bisnis sangatlah dibutuhkan. Dalam Operasi pemasaran elektonik, kualitas dari perancangan mekanisme dan kepercayaan daripada sebuah *marketplace* merupakan pertimbangan yang sangat penting bagi para pelanggan dalam melakukan suatu proses transaksi. Sebagai contoh, penipuan penawaran yang sering terjadi pada suatu *marketplace* tentunya akan membuat nilai dari suatu perusahaan

tersebut menjadi semakin berkurang, dan rasa kepercayaan pembeli untuk terus berpartisipasi dan membeli produk dari suatu perusahaan tersebut akan semakin berkurang dan pada akhirnya dapat menyebabkan kegagalan pada *e–marketplace*".

4. Rababah & Masoud (2010 : 3-4) menyatakan terdapat beberapa faktor yang mempengaruhi kualitas dari *e-commerce*, yaitu sebagai berikut :
   a) *Efficiency*

      Menjelaskan seberapa efisen waktu respon suatu *website* saat digunakan oleh *user*
   b) *User friendliness*

      Kemampuan *User interface* sebuah *website* untuk dapat menyediakan tampilan yang mendukung dan menarik bagi *user*
   c) *Navigability*

      Kemampuan navigasi suatu *website* untuk dapat mengakses konten
   d) *Maintanability*

      Usaha yang dilakukan untuk menjaga agar *software website* yang kita buat dapat selalu up to date dan berguna bagi *user*
   e) *Involvement capacity*

      Suatu pengukuran sejauh mana suatu *website* dapat beradaptasi dan menarik perhatian *user*
   f) *Functionality*

      Menjelaskan aspek operasional apa saja yang bisa disediakan oleh *website* tersebut
   g) *Security*

      Menjelaskan sejauh mana tingkat keamanan suatu *website* dalam menghadapi *threat* atau penyusupan yang dilakukan oleh *user* yang tidak seharusnya.
   h) *Reliability*

      Menjelaskan sejauh mana *website* tersebut dapat selalu *available* dan mampu bekerja dengan baik
   i) *Integrity*

      Reliability, konsistensi, serta kebenaran dari data yang ada

j) Trustworthiness

Menjelaskan sejauh mana *user* dapat yakin bahwa *website* yang digunakan mampu bekerja secara baik, konsisten, *reliable*, dan benar, sehingga dapat menciptakan *trusting relationship*

k) *Content adequacy*

Menjelaskan bahwa informasi yang ditampilkan dapat diaplikasikan secara benar oleh *user* dan dapat menjawab kebutuhan *user*

l) Scalability

Kesiapan suatu *website* agar dapat memenuhi permintaan *user*

m) Availability

Menjelaskan sejauh mana aksesibilitas *website* terhadap *user* dengan menggunakan *browser* yang berbeda

n) Readability

Menjelaskan bagaimana suatu *website* dapat ditampilkan dengan bahasa dan tampilan yang sesuai agar mudah dibaca

o) Standard conformance

Menjelaskan sejauh mana konsistensi *user interface* yang ditampilkan dalam *website*

p) Ease of manipulation

Menjelaskan sejauh mana *website* tersebut menyediakan fitur *help* untuk membantu mengoperasikan *website* tersebut.