BAB II

LANDASAN TEORI

2.1 Teori – Teori Umum Basis Data

Teori – teori yang terdapat di sini merupakan teori – teori umum yang digunakan dalam penulisan skripsi ini.

2.1.1 Data

Data adalah informasi yang disimpan dalam sistem katalog, yang berisi informasi tentang struktur tiap berkas, tipe, dan format penyimpanan setiap *item* data. Semua informasi yang disimpan dalam sistem katalog ini biasa disebut *meta-data*.

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

2.1.2 *Database*

Menurut Connolly dan Begg (2005, p15), *database* adalah koleksi dari data-data yang terkait secara logis dan deskripsi dari data-data tersebut yang didesain untuk memenuhi kebutuhan informasi dari organisasi. Suatu basis data adalah kumpulan data yang telah terorganisasi secara menyeluruh sehingga memiliki koneksi logis dan diterapkan secara sistematis di dalam komputer.

Berikut ini merupakan alasan dari penggunaan *database* menurut Date (2000, p15) :

- Padat

  Tidak diperlukan lagi membuat arsip kertas dalam ukuran besar.

- Kecepatan

  Mesin dapat memperoleh kembali dan mengubah data jauh lebih cepat daripada yang manusia yang dapat lakukan.

- Mengurangi pekerjaan yang membosankan

  Rasa bosan dari proses memelihara arsip – arsip berupa kertas dapat dikurangi.

- Aktual

  Informasi yang terbaru dan akurat selalu tersedia di setiap waktu ketika dibutuhkan.

Pendekatan basis data adalah memisahkan struktur data dari program aplikasi dan menyimpannya dalam *database*. *Database* merupakan sistem penyimpanan *record* terkomputerisasi yang bertujuan untuk pemeliharaan informasi dan tersedia pada saat dibutuhkan. Dalam menganalisa kebutuhan informasi suatu organisasi, kita menentukan entitas, atribut, dan relasi.

Menurut Date (2000, p16), ada delapan keuntungan dengan menggunakan pendekatan *database*, yaitu :

- Redundansi dapat dikurangi
- Ketidakkonsistenan dapat dihindari
- Data dapat di-*shared*
- Standar-standar dapat diciptakan
- Pembatasan keamanan dapat diciptakan
- Integritas dapat dipertahankan
- Keperluan yang bertentangan dapat diseimbangkan
- Tersedianya dukungan untuk transaksi

Teknologi basis data memperbolehkan sekumpulan data dengan berbagai tipe (teks, angka, gambar, suara, dan lain-lain) dan disimpan dalam komputer dan digunakan secara efisien tanpa adanya duplikasi oleh aplikasi yang berhubungan.

2.1.3 *Database Management System* (DBMS)

2.1.3.1 Pengertian DBMS

Menurut Connolly and Begg (2005, p16), *Database Management System* (DBMS) adalah suatu sistem perangkat lunak yang memungkinkan *user* untuk mendefinisikan, membuat, memelihara, dan mengontrol akses ke basis data. Fasilitas-fasilitas yang disediakan oleh DBMS antara lain:

1. *Data Definition Language* (DDL)

   DDL adalah suatu bahasa yang memperbolehkan *Database Administrator* (DBA) atau *user* untuk mendeskripsikan dan memberi nama suatu entitas, atribut, dan relasi data yang dibutuhkan untuk aplikasi, bersama dengan integritas data yang diasosiasikan dan batasan (*constraint*) keamanan data.

2. *Data Manipulation Language* (DML)

   DML adalah suatu bahasa yang menyediakan seperangkat operasi untuk mendukung manipulasi data yang berada pada basis data. Pengoperasian data yang akan dimanipulasi biasanya meliputi:

   - Penambahan data baru ke dalam basis data
   - Modifikasi data yang disimpan ke dalam basis data
   - Pengembalian data yang terdapat di dalam basis data
   - Penghapusan data dari basis data

   DML dibagi ke dalam dua jenis yaitu *Procedural* dan *Non-Procedural*. *Procedural* DML adalah suatu bahasa yang memperbolehkan *user* untuk mendeskripsikan ke sistem data apa yang dibutuhkan dan bagaimana mendapatkan data tersebut secara tepat, sedangkan *non-procedural* DML adalah sebuah bahasa yang

memperbolehkan *user* untuk menentukan data apa yang dibutuhkan tanpa memperhatikan bagaimana data diperoleh.

2.1.3.2 Fungsi DBMS

Menurut Connolly dan Begg (2005, p48), DBMS memiliki sepuluh fungsi yaitu:

1. *Data storage, retrieval, and update*

   DBMS harus dapat memungkinkan *user* untuk menyimpan, mengambil, dan meng-*update* data dalam basis data.

2. *A user-accessible catalog*

   Sebuah DBMS harus menyediakan katalog yang mendeskripsikan lokasi penyimpanan data *item* dan dapat diakses oleh *user*.

3. *Transaction support*

   DBMS harus menyediakan sebuah mekanisme yang akan menjamin baik seluruh *update* yang berhubungan dengan sebuah transaksi dapat dilakukan ataupun keseluruhan *update* tersebut dapat dilakukan.

4. *Concurrency control services*

   DBMS harus memiliki sebuah mekanisme untuk menjamin bahwa basis data dapat diubah dengan benar ketika beberapa *user* mengubah *database* pada waktu yang bersamaan.

5. *Recovery services*

   DBMS harus menyediakan sebuah mekanisme untuk memperbaiki basis data apabila terjadi kerusakan.

6. *Authorization services*

   DBMS harus menyediakan sebuah mekanisme untuk menjamin bahwa hanya *user* yang diberi otoritas lah yang dapat mengakses basis data.

7. *Support for data communication*

   DBMS harus dapat terintegrasi dengan *software* komunikasi, sehingga dapat mengakses *database* dari lokasi yang jauh.

8. *Integrity services*

   DBMS harus menyediakan sarana untuk menjamin baik data di dalam basis data maupun pengubahan terhadap data mengikuti aturan-aturan tertentu (*constraint*).

9. *Services to promote data independence*

   DBMS harus meliputi fasilitas-fasilitas yang mendukung program-program independensi dari struktur basis data aktual.

10. *Utility services*

    DBMS seharusnya menyediakan serangkaian layanan kegunaan seperti program analisis statistik, pengawasan fasilitas, fasilitas reorganisasi indeks, dan lain-lain.

2.1.3.3 Komponen-Komponen DBMS

Menurut Connolly dan Begg (2005, p18), DBMS memiliki lima komponen penting yaitu:

1. *Hardware* (Perangkat Keras)

   Yaitu dapat berupa *single personal computer*, *single mainframe,* sampai jaringan komputer.

2. *Software* (Perangkat Lunak)

   Yaitu DBMS, sistem operasi, *software* jaringan (bila diperlukan) dan juga program-program aplikasi.

3. Data

   Data merupakan komponen terpenting dari DBMS dan juga merupakan komponen penghubung antar komponen mesin (*hardware* dan *software*) dengan komponen *human*

(*Procedures* dan *People*). Data yang digunakan oleh organisasi dan deskripsi dari data disebut *schema*.

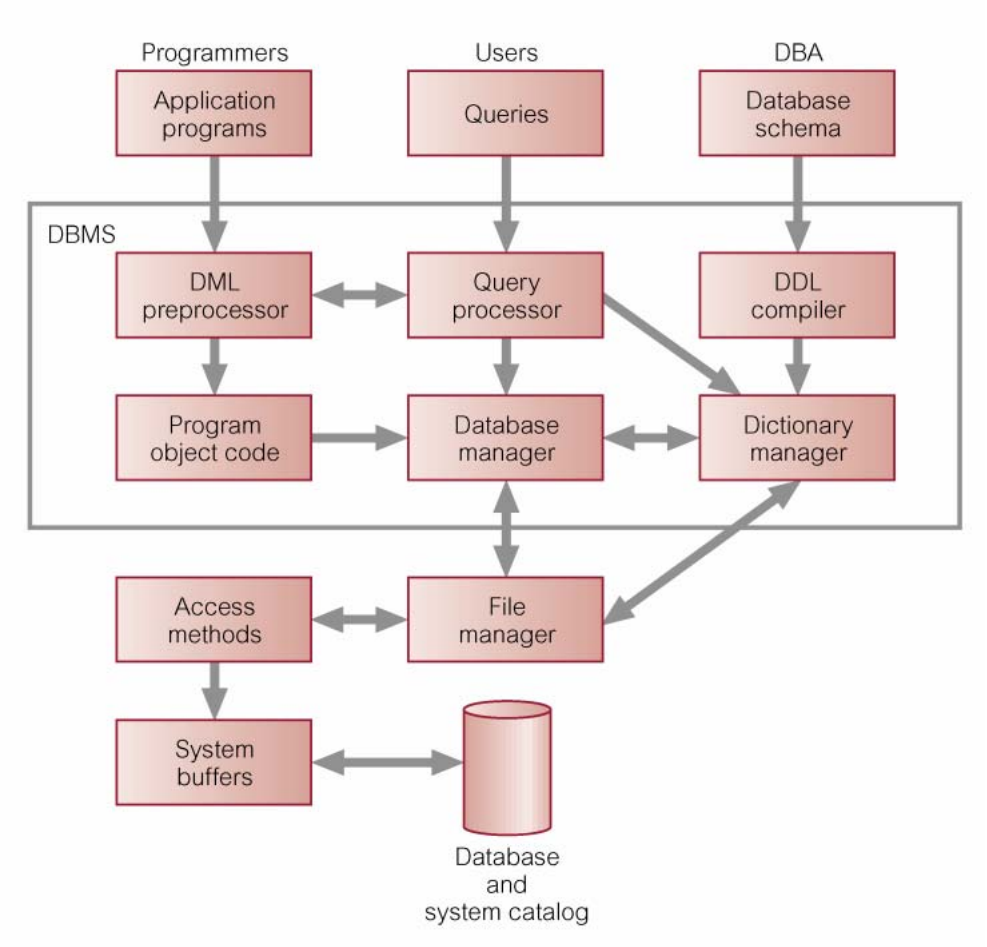
4. Prosedur

   Merupakan panduan dan instruksi dalam membuat desain dan menggunakan basis data. Penggunaan dari sistem dan staf dalam mengelola basis data membutuhkan prosedur dalam menjalankan sistem dan mengelola basis data itu sendiri.

5. *People*, orang yang terlibat dalam sistem yaitu:

   - *Data Administrator* adalah orang yang berwenang untuk membuat keputusan strategis dan kebijakan mengenai data yang ada.
   - *Database Administrator* adalah orang yang menyediakan dukungan teknis untuk implementasi keputusan tersebut, dan bertanggung jawab atas keseluruhan kontrol sistem pada *level* teknis.
   - *Desainer Database* (logikal dan fisikal) adalah perancang sistem *database* pada tahap logikal dan fisikal.
   - *Programmer* Aplikasi adalah orang yang bertanggung jawab untuk membuat aplikasi *database* dengan menggunakan bahasa pemrograman yang ada.

- *End Users*
- *Naive* adalah u*ser* yang tidak perlu tahu mengenai *database* dan DBMS, tetapi hanya menggunakan program aplikasi.
- *Sophisticated* adalah u*ser* yang familiar dengan struktur *database* dan DBMS.



Gambar 2.1 Komponen dalam DBMS (Connolly dan Begg)

2.1.3.4 Keuntungan dan Kerugian DBMS

Menurut Connolly dan Begg (2005, p26), DBMS memiliki beberapa keuntungan yaitu:

1. Mengontrol redundansi data
2. Konsistensi data
3. Lebih banyak informasi dari jumlah data yang sama
4. *Share data*
5. Meningkatkan integritas data
6. Meningkatkan sekuritas
7. *Standard* pelaksanaan (format data, penamaan, dan prosedur *update*)
8. *Economy of scale* yaitu data operasional perusahaan dijadikan satu, kemudian aplikasi dibuat dengan menggunakan *data source* yang tunggal tersebut sehingga akan menghemat biaya
9. Keseimbangan konflik kebutuhan (*database* untuk berbagai kepentingan)
10. Meningkatkan aksesibilitas data dan *responsiveness*
11. Meningkatkan produktivitas
12. Meningkatkan *maintenance* melalui *data independence* (data menjadi global)
13. Meningkatkan konkurensi data (mengurangi *loss information* dan *loss integration*)

14. Meningkatkan layanan *backup* dan *recovery*

DBMS juga memiliki kerugian yaitu:

1. Rumit (*complexity*)

   Penetapan fungsi dari DBMS yang baik, menyebabkan DBMS menjadi *software* yang cukup rumit. Seluruh *user* harus mengetahui fungsi-fungsi yang ada sehingga dapat memperoleh manfaatnya.

2. Ukuran (*size*)

   Kerumitan dan banyaknya fungsi yang ada menyebabkan DBMS memerlukan banyak *software* pendukung yang mengakibatkan penambahan tempat penyimpanan dan *memory*.

3. Biaya DBMS (*Cost* of DBMS) yang tinggi
4. Biaya tambahan *hardware* (*additional hardware costs*)
5. Biaya konversi (Biaya pelatihan, biaya staf spesialis)
6. Performa (*performance*)

   Pada dasarnya DBMS dibuat untuk menyediakan banyak aplikasi, akibatnya mungkin beberapa aplikasi tidak berjalan seperti biasanya.

7. Dampak kegagalan lebih besar (*higher impact of failure*)

   Sistem yang terpusat, jika seluruh *user* dan aplikasi terakses dari DBMS maka kerusakan pada bagian manapun

dari sistem, akan menyebabkan tidak bekerjanya proses operasi tersebut.

2.1.3.5 Pemilihan DBMS (DBMS *Selection*)

Menurut Connolly dan Begg (2005, p295), pengertian pemilihan DBMS adalah menyeleksi DBMS yang tepat untuk mendukung aplikasi basis data. Seleksi DBMS dilakukan diantara tahapan perancangan *database* logikal dan perancangan *database* fisikal dengan tujuan untuk kebutuhan sekarang dan kebutuhan yang akan datang pada suatu organisasi. Tahap-tahap utama untuk memilih DBMS yaitu:

- Mendefinisikan persyaratan studi referensi
- Mendaftar dua atau tiga produk
- Evaluasi produk
- Rekomendasi pilihan dan laporan produk

2.1.4 *Requirement Collection and Analysis* (Pengumpulan dan Analisis Kebutuhan)

Menurut Connolly dan Begg (2005, p288), pengumpulan dan analisis kebutuhan adalah proses pengumpulan dan analisis informasi tentang bagian organisasi yang didukung oleh aplikasi basis data dan yang menggunakan informasi ini untuk mengidentifikasikan kebutuhan

– kebutuhan *user* dari sistem yang baru. Teknik pengumpulan kebutuhan atau *fact-finding* memiliki 5 jenis teknik yang disesuaikan dengan kebutuhan *user* yaitu:

1. **Wawancara**

   Menurut Connolly dan Begg (2005, p317), wawancara bertujuan untuk mengumpulkan fakta - fakta, memeriksa kebenaran fakta yang ada dan mengklarifikasinya untuk membangkitkan semangat, melibatkan pengguna akhir, mengidentifikasi kebutuhan – kebutuhan, dan mengumpulkan ide – ide dan pendapat. Teknik ini memerlukan kemampuan komunikasi yang baik untuk menghadapi *user* yang memiliki nilai, prioritas, pendapat, motivasi, dan kepribadian yang beragam.

2. **Kepustakaan**

   Dengan membaca jurnal komputer, buku – buku referensi, buku – buku dari perpustakaan, dan internet merupakan sumber informasi yang baik yang menyediakan informasi bagaimana orang lain memecahkan masalah yang serupa.

3. **Observasi (*Survey*)**

   Memungkinkan untuk ikut serta atau mengamati seseorang dalam melakukan kegiatan untuk mempelajari sistem. Salah satu faktor pengamatan dapat berhasil adalah dengan mencari informasi sebanyak mungkin tentang aktivitas yang akan diamati serta orang yang melakukan aktivitas tersebut.

4. **Penelitian (*Research*)**

Dengan melakukan riset atau penelitian di luar organisasi yang menjadi tujuan.

5. **Kuesioner**

Menurut Connolly dan Begg (2005, p320), kuesioner adalah suatu dokumen dengan tujuan khusus yang memungkinkan fakta – fakta dikumpulkan dari banyak orang sekaligus menjaga kontrol terhadap tanggapan yang diberikan.

2.1.5 *Database Design Methodology*

Menurut Connolly dan Begg (2005, p438), metodologi perancangan basis data adalah suatu pendekatan terstruktur yang menggunakan prosedur, teknik, alat-alat, dan bantuan dokumentasi untuk mendukung dan memfasilitasi proses perancangan. Menurut Connolly dan Begg (2005, p439) proses perancangan *database* terdiri atas tiga bagian, yaitu:

2.1.5.1 Perancangan Basis Data Konseptual

Menurut Connolly dan Begg (2005, p439), perancangan basis data konseptual adalah proses pembentukan model dari informasi yang digunakan dalam *enterprise*, *independent* dari keseluruhan aspek fisik. Model data konseptual merupakan sumber informasi untuk fase *design logical*. Keseluruhan

proses dari pengembangan model data adalah diuji dan divalidasikan dari kebutuhan pemakai. Hasil akhir dari perancangan basis data konseptual yaitu berupa identifikasi tipe entitas, identifikasi tipe *relationship*, identifikasi dan hubungan atribut dengan tipe entitas atau tipe relasi, menentukan atribut domain, dan menentukan *candidate key* dan *primary key* yang didasarkan pada spesifikasi kebutuhan. Perancangan basis data konseptual memiliki beberapa langkah penting yaitu :

1. Menentukan tipe *class* yang akan digunakan dalam *semantic object diagram*

   Menentukan objek-objek yang diinginkan oleh *user*, yaitu jenis-jenis entitas yang diperlukan dalam *view*. Dalam penentuan ini, biasanya berdasarkan kepada kata benda atau frasa kata benda yang disebutkan.

2. Menentukan tipe *relationship*

   Menentukan relasi yang penting yang terjadi di antara entitas yang telah ditentukan. Biasanya relasi yang terjadi ditandai dengan kata kerja atau ekspresi kata kerja.

3. Mengidentifikasi dan mengasosiasikan *attribute* dengan tipe *object diagram*.

4. Menentukan domain atribut

   Menetapkan domain (kisaran nilai yang diperbolehkan) untuk setiap atribut yang ada dalam *local conceptual data model.*

5. Menentukan atribut *candidate key* dan atribut *object identifier (primary key)*

   Menentukan semua *candidate key* yang mungkin untuk setiap jenis entiti dan kemudian dipilih salah satu untuk menjadi *primary key.* Apabila *candidate key* masih diperlukan maka disebut *alternate key.*

6. Mempertimbangkan pemakaian *enhanced modeling concepts* (*optional*)

   Langkah ini akan melanjutkan pengembangan dengan E-R *modelling* seperti *specialization / generalization, aggregation* dan *composition.*

7. Memeriksa model terhadap *redundancy*

   Memeriksa kembali hubungan *one-to-one* (1:1) dan menghilangkan *redundancy relation.*

8. Memvalidasikan *local conceptual model* terhadap transaksi *user*

   Langkah ini memastikan *local conceptual model* mendukung transaksi-transaksi yang diperlukan pada *view*

dengan cara menggambarkan transaksi atau dengan menggunakan alur transaksi.

9. Meninjau ulang *local conceptual data model* yang dihasilkan terhadap *user*.
10. Meninjau kembali *local conceptual data model* dengan *user* untuk memastikan bahwa model tersebut telah sesuai untuk menggambarkan *view*.

2.1.5.2 Perancangan Basis Data Logikal

Menurut Connolly dan Begg (2005, p439), perancangan basis data logikal adalah proses membangun sebuah model dari informasi yang diperoleh dari sebuah organisasi berdasarkan model data khusus (*specific data model*), tetapi tidak bergantung kepada hal yang berkaitan dengan DBMS dan pertimbangan fisik lainnya. Keseluruhan proses dari pengembangan model data logikal adalah diuji berdasarkan kebutuhan *user*. Model data logikal merupakan sumber informasi untuk tahapan selanjutnya yang dinamakan perancangan basis data fisikal. Langkah-langkah dalam perancangan basis data logikal yaitu:

a) Membangun suatu hubungan *database* dalam *semantic object model*

b) Membangun dan memvalidasikan *local logical data model* untuk setiap *view*. Ada beberapa langkah yang harus dilakukan yaitu:

- Menghilangkan fitur yang tidak sesuai dengan *relational model*
- Menentukan hubungan untuk model data logikal
- Memvalidasi hubungan terhadap transaksi *user*
- Mendefinisikan *integrity constraints*
- Mengecek model data logikal lokal terhadap *user*
- Mendesain *enterprise constraints*

2.1.5.3 Perancangan Basis Data Fisikal

Menurut Connolly dan Begg (2005, p439), perancangan basis data fisikal adalah tahap berikutnya setelah model basis data logikal di mana perancangan ini menjadi penjelasan dari deskripsi relasi dasar, organisasi *file,* dan indeks yang digunakan untuk mengakses data secara lebih efisien, serta adanya batasan integritas tentang ukuran keamanan yang diimplementasi pada *secondary storage.* Perancangan basis data fisikal merupakan tahap terakhir dari perancangan *database* di mana perancang memutuskan bagaimana *database*

tersebut diimplementasikan secara fisik dari *logical database design*.

Langkah-langkah dalam merancang basis data secara fisikal adalah sebagai berikut ini:

1. Menerjemahkan model data logikal global untuk tujuan *Database Management System* (DBMS) , yaitu dengan:
   a. Mendesain *relational database*
   b. Mendesain representasi data dari turunan / *derived* data
   c. Mendesain *general constraints*
2. Memilih organisasi *file*
3. Mengestimasi *disc space* yang dibutuhkan
4. Mendesain *user views*
5. Mendesain mekanisme keamanan

2.1.6 Sistem

Menurut McLeod (2004, p11), sistem adalah sekelompok elemen yang terintegrasi dengan maksud yang sama untuk mencapai suatu tujuan. Berdasarkan pada penjelasan di atas, maka dapat disimpulkan bahwa sistem adalah kumpulan dari elemen-elemen yang saling terintegrasi untuk mencapai suatu tujuan tertentu.

2.1.6.1 Komponen Sistem

Menurut O'Brien (2003, p11), sebuah sistem memiliki tiga komponen dasar yaitu :

a. Masukan (*input*)

Masukan meliputi elemen-elemen yang ditangkap dan dirangkai untuk dimasukkan ke dalam sistem komputer dan diproses lebih lanjut. Biasanya dapat berupa perintah dari *keyboard* ataupun data.

b. Proses

Proses meliputi proses perubahan yang mengubah dari sebuah masukan menjadi sebuah *output.*

c. Keluaran (*Output*)

Output meliputi pemindahan elemen-elemen yang telah dihasilkan dari sebuah proses perubahan untuk tujuan akhir yang diinginkan. Biasanya dapat berupa informasi dalam bentuk yang beragam, seperti angka biner, karakter, dan gambar.

2.1.7 Informasi

Menurut O'Brien (2003, p13), informasi adalah data yang sudah dikonversi menjadi konteks yang lebih berarti dan berguna untuk *user* akhir tertentu.

Sedangkan menurut Whitten et. al. (2004, p23), informasi adalah data yang diproses atau diorganisasi ulang menjadi bentuk yang lebih berarti. Informasi dibentuk dari kombinasi data yang diharapkan memiliki arti bagi penerima.

2.1.8 Perancangan Aplikasi

Menurut Connolly dan Begg (2005, p299), perancangan aplikasi adalah merancang *user interface* dan program aplikasi, yang akan memproses basis data dan menerjemahan kebutuhan *user* informasi ke dalam alternatif rancangan sistem yang diajukan kepada pemakai informasi untuk dipertimbangkan.

Berdasarkan penjelasan tersebut dapat disimpulkan bahwa perancangan sistem merupakan proses pembuatan atau pengembangan sistem berdasarkan kebutuhan pemakai informasi.

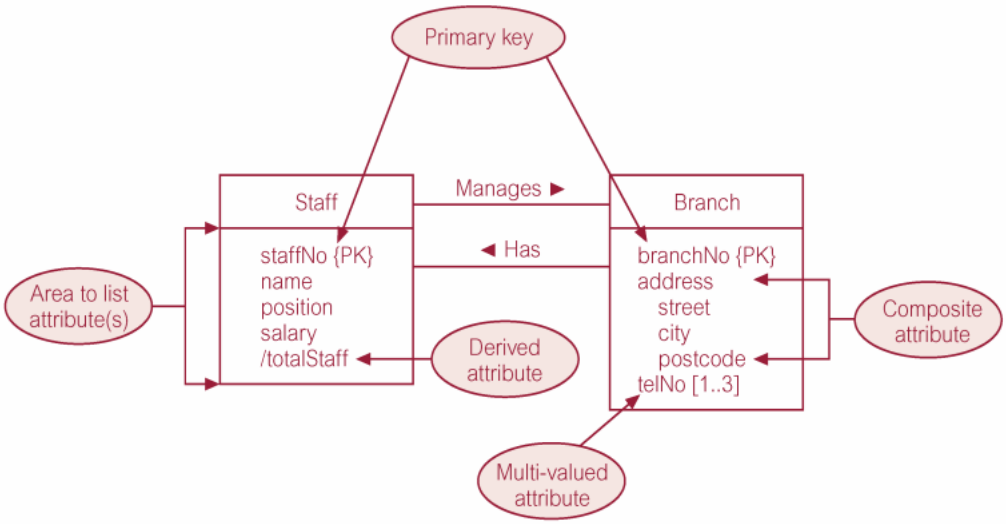
2.1.9 *Prototyping*

Menurut Connolly dan Begg (2005, p303), *prototyping* adalah membuat model kerja dari aplikasi basis data. Tujuannya adalah untuk memungkinkan *user* menggunakan *prototype* untuk mengidentifikasikan

fitur-fitur sistem yang berjalan dengan baik atau tidak, dan bila memungkinkan untuk menyarankan peningkatan atau bahkan penambahan fitur-fitur baru ke dalam sistem *database*.

2.1.10 *Entity Relationship Modelling* (E-R *Modelling*)

Dalam merancang sebuah *database* digunakan *Entity Relationship Model* dengan menggunakan tabel-tabel yang dideskripsikan dalam bentuk entitas. Contoh desain *database* menggunakan E-R *modelling*:



Gambar 2.2 Contoh *Entity Relationship Model* (Connolly dan Begg)

2.1.10.1 *Entity Type*

Menurut Connolly dan Begg (2005, p343), *entity* adalah kumpulan dari objek yang memiliki sifat yang sama, yang diidentifikasi oleh *enterprise* yang mempunyai eksistensi

yang independen. Keberadaannya dapat berupa fisik maupun abstrak. Hubungan antar tabel disebut dengan *relationship* yang merupakan asosiasi antar *entity* yang memiliki arti tertentu.

2.1.10.2 Atribut

Menurut Connolly dan Begg (2005, p350), atribut adalah sifat-sifat (*property*) dari sebuah *entity* atau *relationship type*. *Attribute domain* adalah sekumpulan nilai yang hanya diperbolehkan untuk satu atau lebih atribut.

Atribut terdiri atas beberapa macam yaitu:

1. *Simple Attribute*, yaitu atribut yang terdiri atas satu komponen tunggal dengan keberadaan yang independen dan tidak dapat dibagi menjadi bagian yang lebih kecil lagi.
2. *Composite Attribute*, yaitu atribut yang terdiri atas beberapa komponen di mana masing-masing komponen memiliki keberadaan yang independen.
3. *Single-valued Attribute*, yaitu atribut yang mempunyai nilai tunggal untuk setiap kejadian.
4. *Multi-valued Attribute*, yaitu atribut yang mempunyai beberapa nilai untuk setiap kejadian.
5. *Derived Attribute*, yaitu atribut yang memiliki nilai yang

dihasilkan dari satu atau beberapa atribut lainnya, dan tidak harus berasal dari satu entitas.

2.1.10.3 *Keys*

Menurut Connolly dan Begg (2005, p352), *key* digunakan untuk menghubungkan antar tabel yang digunakan. Ada beberapa macam *key* yaitu:

1. *Super Key* yaitu atribut unik yang digunakan untuk mengidentifikasi *row*.
2. *Candidate Key* yaitu atribut unik yang digunakan untuk mengidentifikasi tabel.
3. *Primary Key* adalah atribut unik yang digunakan untuk mengidentifikasi setiap *row* dalam tabel.
4. *Alternate Key* adalah *candidate key* yang tidak terpilih menjadi *primary key*.
5. *Composite Key* adalah *candidate key* yang terdiri atas dua atau lebih atribut.
6. *Foreign Key* adalah atribut dari sebuah tabel yang berada pada tabel lain.

2.1.10.4 *Strong* dan *Weak Entity / Class Types*

Menurut Connolly dan Begg (2005, p354), *Strong Entity / Class Type* adalah entitas yang keberadaannya tidak

bergantung pada entitas lain sedangkan *Weak Entity / Class Type* adalah entitas yang keberadaannya bergantung pada entitas lain. *Strong Entity Type* biasanya disebut dengan *parent* atau *owner dominant* dan *Weak Entity Type* biasa disebut dengan *child, dependent,* atau *subordinate.*

2.1.11 Estimasi *Disc Space*

Menurut Connolly dan Begg (2005, p514), langkah ini bertujuan untuk menghitung kapasitas penyimpanan yang dibutuhkan oleh *database*. Kapasitas yang ditentukan ini sangat bergantung pada DBMS yang digunakan, yaitu Mircosoft SQL Server 2005. Langkah-langkah yang diperlukan untuk menhitung kapasitas *space* yang diperlukan adalah:

1. Menentukan jumlah baris

   Jumlah baris di tabel = *Num_Rows*

2. Jika ada kolom *fixed length* dan *variable length* di tabel, perlu dihitung kapasitas masing – masing kolom sesuai dengan datanya. Ukuran kolom tergantung dari tipe data dan panjangnya.

   a. Jumlah kolom = *Num_Cols*

   b. Jumlah *byte* untuk kolom *fixed length = Fixed_Data_Size*

   c. Jumlah kolom *variable length = Num_Variable_Cols*

   d. Ukuran maksimum kolom *variable length = Max_Var_Size*

3. Untuk kolom *fixed length* di tabel, perhitungan *null bitmap* (bagian dari baris) diperlukan untuk mengatur *nullability* kolom.

   Null_Bitmap = 2 + ((*Num_Cols* + 7) / 8)

4. Untuk kolom *variable length* di tabel, tentukan berapa banyak kapasitas yang dibutuhkan untuk menyimpan kolom *variable length*.

   Variable_Data_Size = 2 + (*Num_Variable_Cols* * 2) + *Max_Var_Size*

5. Menghitung ukuran baris
   a. *Row_Size* = *Fixed_Data_size* + *Variable_Data_Size* + *Null_Bitmap* + 4 (Nilai untuk data *row header*)
6. Menghitung jumlah baris per halaman
   a. Untuk menghitung jumlah baris per halaman, nilai yang biasanya dipakai sebagai pembagi untuk ukuran halaman adalah 8 Kb (8192 *byte*), dan biasanya ukuran *byte* yang digunakan (*free bytes*) adalah 8096 *byte*
   b. *Rows_Per_Page* = (8096) / (*Row_Size* + 2)
7. Jika ada penambahan *index clustered* pada tabel, perlu dilakukan perhitungan jumlah baris yang bisa digunakan (*free*) per halaman, sesuai dengan *fill factor* yang ditentukan. Jika tidak ada penembahan *index clustered* maka *fill factor* diberi nilai 100.
   a. *Free_Rows_Per_page* = 8096 * ((100–*Fill_Factor*)/100) / (*Row_Size*+2)

b. Semakin tinggi nilai *fill factor*, maka data yang disimpan di tiap halaman semakin banyak.

8. Menghitung jumlah halaman yang dibutuhkan untuk menyimpan semua baris

   Num_Pages = *Num_Rows* / (*Rows_Per_Page* – *Free_Rows_Per_Page*)

9. Menghitung jumlah kapasitas yang dibutuhkan untuk menyimpan data pada tabel.

   Total_Size (bytes) = 8192 * *Num_Page*

2.1.12 Manajemen Proses Bisnis

Manajemen proses bisnis (BPM) dapat didefinisikan sebagai kumpulan dari proses dan berisi kumpulan aktivitas (*tasks*) yang saling berelasi satu sama lain untuk menghasilkan suatu keluaran yang mendukung pada tujuan dan sasaran strategis dari organisasi. Suatu proses bisnis yang baik harus memiliki tujuan-tujuan seperti mengefektifkan, mengefisienkan dan membuat mudah untuk beradaptasi pada proses-proses di dalamnya. Artinya proses bisnis tersebut harus merupakan proses bisnis yang berorientasikan pada jumlah dan kualitas produk *output*, minimal dalam menggunakan sumber daya dan dapat beradaptasi sesuai dengan kebutuhan bisnis dan pasar. Manajemen proses bisnis yang efektif dan efisien dapat menghasilkan nilai-nilai kompetitif bagi perusahaan. Proses bisnis yang dikelola dengan baik

akan mampu menumbuhkan peluang. Namun perusahaan terkadang kurang memahami dan tidak mampu mengontrol proses bisnis yang dimilikinya. Pihak manajemen mungkin telah berhasil membuat prosedur yang ideal untuk menjalankan proses bisnisnya, tapi pada kenyataannya, implementasi di lapangan dapat sangat berbeda dari apa yang telah dirancang sebelumnya. Pada pelaksanaan suatu proses bisnis kadang terjadi redundansi, ketidakefisienan, stagnasi, dan berbagi kesalahan-kesalahan lainnya yang tidak dapat diantisipasi sebelumnya. Bisnis yang tidak tangkas dalam mengontrol proses bisnis yang dimilikinya cenderung akan menghalangi usaha perusahaan dalam mencapai sasaran yang diinginkan. BPM membantu perusahaan dalam mengawasi dan mengontrol seluruh elemen pada proses bisnis, seperti karyawan, pelanggan, pemasok, dan workflow. BPM meningkatkan kualitas proses bisnis melalui penyediaan mekanisme *feedback* yang lebih baik. *Review* yang berkesinambungan dan *real-time* akan membantu perusahaan dalam mengidentifikasi masalah dan kemudian mengatasinya secara lebih cepat sebelum masalah tersebut berkembang menjadi lebih besar.

Setiap solusi Manajemen Proses Bisnis (BPM) memiliki empat komponen utama:

- **Permodelan**

  Pengguna dapat mendefinisikan dan mendesain struktur dari setiap proses bisnis secara grafis. Manajer proses dapat mendesain sebuah

proses beserta seluruh elemen, aturan, sub-proses, *parallel* proses, penanganan *exception*, penanganan *error*, dan *workflow* dengan mudah tanpa perlu memiliki kemampuan *programming* khusus dan tanpa membutuhkan bantuan dari staf IT.

- **Pengintegrasian**

  BPM dapat menghubungkan setiap elemen dalam proses sehingga elemen-elemen tersebut dapat saling berkolaborasi dan bertukar informasi untuk menyelesaikan tujuannya. Pada *level* aplikasi, hal ini bisa diartikan sebagai penggunaan *Application Programming Interface* (API) dan *messaging*. Bagi pengguna, hal ini berarti tersedianya sebuah *workspace* pada komputernya ataupun perangkat *wireless*-nya untuk mengerjakan tugas sesuai dengan perannya pada suatu proses bisnis.

- **Pengawasan**

  Pengguna dapat mengawasi dan mengontrol *performance* dari proses bisnis yang sedang berjalan dan *performance* dari setiap personil yang terlibat dalam proses bisnis tersebut. Pengguna juga dapat memperoleh informasi mengenai proses yang tengah berjalan, maupun yang telah selesai, beserta data-data yang ada di dalamnya.

- **Optimalisasi**

  Pengguna dapat menganalisa dan memonitor suatu proses bisnis, melihat ketidakefisienan, dan juga memungkinkan pengguna untuk

mengambil tindakan dengan cepat dan merubah proses tersebut untuk meningkatkan efisiensinya.

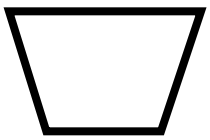
2.1.13 Bagan Alir (*Flowchart*)

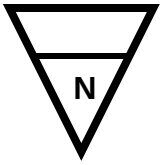
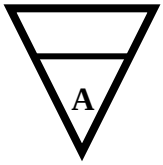
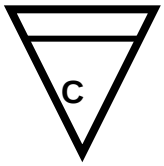
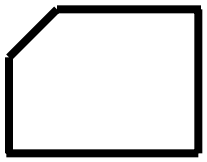
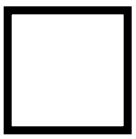
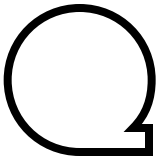
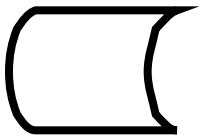
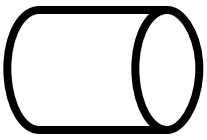
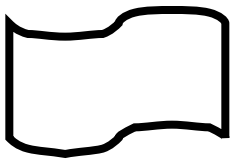
Menurut McLeod (2001, p40), bagan alir adalah suatu model yang menggambarkan aliran data atau dokumen serta proses untuk mengolah data atau dokumen ke dalam suatu proses. Bagan alir juga merupakan bagan (*chart*) yang menunjukkan alir (*flow*) di dalam program atau prosedur sistem secara logika. Bagan ini digunakan terutama sebagai alat bantu komunikasi dan untuk dokumentasi. Bagan alir yang digunakan dalam penulisan skripsi ini dibagi ke dalam dua jenis yaitu:

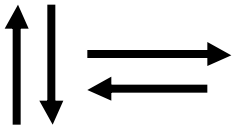
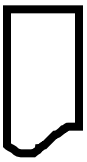
1. Bagan alir sistem (*system flowchart*)

   Merupakan bagan yang menunjukkan arus pekerjaan secara keseluruhan dari sistem, urutan dari prosedur-prosedur yang ada di dalam sistem, dan menunjukkan apa yang dikerjakan di sistem.

   Komponen - komponen dari *system flowchart* ini dilihat pada Tabel 2.1.

	Simbol dokumen; menunjukkan *input* dan *output* baik untuk proses manual, mekanik atau komputer.		Simbol manual; menunjukkan pekerjaan yang dilakukan secara manual.

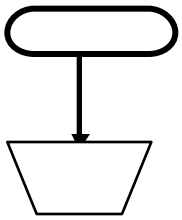
	Simbol untuk menyimpan secara *offline*; *file* non-komputer yang diarsip dengan mengurutkan angka (*numerical*).		Simbol untuk menyimpan secara *offline*; *file* non-komputer yang diarsip dengan mengurutkan huruf (*alphabetical*).
	Simbol untuk menyimpan secara *offline*; *file* non komputer yang diarsip dengan mengurutkan tanggal (*chronological*).		Simbol kartu *punch*; menunjukkan I/O yang menggunakan kartu *punch*.
	Simbol proses; untuk menunjukkan kegiatan proses dari operasi program komputer.		Simbol operasi luar; untuk menunjukkan operasi yang dilakukan di luar operasi komputer.
	Simbol *sort offline*; menunjukkan proses pengurutan data di luar proses komputer.		Simbol pita *magnetic*; menunjukkan I/O menggunakan pita *magnetic*.
	Simbol *disc* ; menunjukkan I/O dengan menggunakan *harddisk*.		Simbol disket; menunjukkan I/O dengan menggunakan disket.
	Drum magnetik; menunjukkan I/O dengan menggunakan drum *magnetic*.		Pita kertas berlubang; menunjukkan I/O dengan menggunakan pita kertas pita berlubang.

	*Keyboard*; menunjukkan *input* yang menggunakan *on-line keyboard*.		*Display*; menunjukkan *output* yang ditampilkan di *monitor*.
	Hubungan komunikasi; menunjukkan proses transmisi data *mell* untuk saluran komunikasi.		Garis alir; menunjukkan arus dari proses.
	Penjelasan; menunjukkan penjelasan dari suatu proses.		Penghubung; menunjukkan penghubung ke halaman yang sama atau halaman lain.
	Pita Kontrol; menunjukkan penggunaan pita kontrol (*control tape*) dalam *batch control* yang digunakan untuk pencocokan pada proses *batch processing*.		

Tabel 2.1 Notasi Bagan Alir Sistem

2. Bagan alir dokumen

Bagan alir dokumen (*document flowchart*) atau disebut juga bagan alir formulir (*form flowchart*) atau *paperwork flowchart* merupakan bagan alir yang menunjukkan arus dari laporan dan formulir termasuk dokumen – dokumen yang digunakan. Bagan alir dokumen menggunakan simbol-simbol yang sama dengan bagan alir sistem. Contoh dari bagan alir dokumen adalah sebagai berikut:

Pengendalian Persediaan	Bagian Pembelian	Bagian Pengiriman	Gudang	Hutang Dagang
				

Tabel 2.2 Contoh Bagan Alir Dokumen

2.1.14 Bahasa Pemrograman

Dalam mengembangkan aplikasi untuk penelitian pada skripsi ini, menggunakan perangkat lunak yaitu bahasa pemrograman untuk dapat membuat sistem yang sesuai dengan kebutuhan *user*. Bahasa pemrograman yang digunakan adalah C# (biasanya disebut juga sebagai "C *Sharp*") yang merupakan sebuah bahasa pemrograman komputer baru yang dikembangkan oleh Microsoft Corporation, USA. Menurut Deitel (2002, p58), C# merupakan bahasa pemrograman *object-oriented* seperti Java dan merupakan bahasa *component-oriented* yang pertama kali dikembangkan. Bahasa pemrograman C# dibuat sebagai bahasa pemrograman yang bersifat *general-purpose* (untuk tujuan jamak), berorientasi objek, modern dan sederhana.

Bahasa pemrograman C# ditujukan dalam mengembangkan komponen perangkat lunak yang mampu mengambil keuntungan dari lingkungan yang terdistribusi. C# ditujukan untuk program aplikasi baik

dari segi *client-server* (*hosted system*) maupun sistem *embedded* (*embedded system*), mulai dari program aplikasi sangat besar yang menggunakan sistem operasi canggih hingga program aplikasi sangat kecil yang memiliki fungsi-fungsi terdedikasi.

Bahasa pemrograman ini memiliki sifat sebagai berikut:

- C# adalah sebuah bahasa pemrograman baru yang diturunkan dari bahasa pemrograman C/C++
- Sintaks pada C# lebih sederhana dan lebih modern daripada C++
- Merupakan bahasa *component-oriented* yang pertama
- Merupakan bahasa yang didesain dari .Net Framework
- Mengkombinasikan fitur-fitur terbaik dari banyak bahasa yang umum digunakan seperti produktivitas dari Visual Basic, dan kekuatan dari C++ dan Java
- C# menjadi pilihan bahasa bagi pemrograman .Net

Selain menggunakan bahasa pemrograman untuk mengembangkan sistem yang sesuai, dibutuhkan suatu DBMS untuk dapat menyimpan basis data dan dapat memanipulasi data. Dalam pengembangan sistem ini, digunakan Microsoft SQL Server 2005 sebagai *Relational Database Management System* (RDBMS) yang cocok digunakan dan sesuai dengan bahasa pemrograman yang digunakan.

Microsoft SQL Server 2005 adalah sebuah sistem manajemen basis data relasional (RDBMS) yang dihasilkan oleh Microsoft. *Query language* utamanya adalah transact-SQL. Microsoft SQL Server 2005 memiliki fitur yang tidak dimiliki pendahulunya yaitu sebagai *platform* pengembangan aplikasi *business intelligence* (BI). *Business intelligence* (BI) merupakan proses ekstraksi data mentah dari sekumpulan data operasional lalu mengkombinasikan, membuat korelasi, dan membuat kesimpulan dari data tersebut, untuk kemudian dianalisa lebih lanjut untuk melakukan pengambilan keputusan.

Sebagai *platform* BI, maka ada tiga *service* yang menjadi fungsi utama yaitu :

1. *Integration Services*

   Digunakan untuk mengumpulkan, mengintegrasikan, dan membuat kesimpulan terhadap data operasional yang berasal dari berbagai sumber.

2. *Analysis Services*

   Memiliki dukungan terhadap kemampuan *Online Analytical Processing* (OLAP) dan *data mining*.

3. *Reporting Services*

   Digunakan untuk membangun infrastruktur pengelolaan dan pendistribusian laporan.

2.1.15 *Structured Query Langauge*

Menurut Connolly dan Begg (2005, p113) SQL adalah contoh dari *transform-oriented language* atau bahasa yang didesain untuk menggunakan relasi untuk mengubah *input* menjadi *output* yang diinginkan. *Database language* dapat memungkinkan *user* untuk:

- Membuat struktur relasi dan *database*
- Melakukan operasi penyisipan (*insertion*), perubahan (*modification*) dan penghapusan (*deletion*) data dari relasi
- Melakukan *query simple* dan kompleks.

2.1.16 Kontrol Akses

Ada beberapa kontrol akses yang digunakan dalam penggunaan *database* yaitu:

- *Privileges*

  Digunakan untuk membatasi penggunaan sintaks *insert, update, delete,* dan *references* untuk kolom yang ditentukan. *Admin* harus memberikan hak akses yang dianggap perlu kepada *user* lain dengan menggunakan perintah *grant.* Sintak *select* digunakan untuk membuat *view* bagi *user* yang akan menampilkan data yang tersedia pada kolom tertentu. Untuk mengambil kembali hak akses yang diberikan oleh *grant,* digunakan perintah *revoke.*

2.2 Teori - Teori Khusus

Berikut ini adalah teori – teori khusus yang digunakan dalam penulisan skripsi ini.

2.2.1 *Semantic Object Model*

Selain menggunakan permodelan E-R model untuk menggambarkan suatu *database*, dapat pula menggunakan permodelan *Semantic Object Model.*

2.2.1.1 Pengertian

Menurut Kroenke (2006, p610), *Semantic object modelling* adalah suatu permodelan *database* yang terdiri atas sekumpulan atribut-atribut yang digunakan untuk menjelaskan perbedaan identitas dari satu objek dengan objek yang lain. *Semantic object* bisa juga disebut objek. Seperti halnya entitas, *semantic objects* dikelompokkan ke dalam *class-class*. *Class object* mempunyai nama yang memberikan identitas dari satu *class* dengan *class* yang lain dan berkorespondensi dengan nama yang diwakilkan oleh *class* yang bersangkutan.

Seperti halnya entitas, sebuah *object* mempunyai sekumpulan atribut. Setiap atribut memiliki karakteristik dari identitas yang akan diwakilkan. Misalnya, *object* DOSEN mempunyai atribut seperti Nama, NIP, Alamat, dan Telepon. Kumpulan dari atribut ini biasa disebut dengan *sufficient description*, yang artinya atribut merepresentasikan semua

karakteristik yang dibutuhkan *user*. Model *semantic object* lebih mudah untuk dimengerti daripada E-R *modelling* karena dalam pembuatannya, atribut-atribut yang terdapat dalam *semantic object* dibuat mengikuti *user interface* yang atribut-atributnya dapat secara langsung dibuat *object*-nya sehingga tidak ada atribut yang tidak terpakai dalam perancangan *database*.

Dalam penggunaan E-R *modelling*, akan memiliki entitas dan *relationship*. Sedangkan jika model *semantic* yang digunakan, akan memiliki *object semantic* dan konstruksi yang berhubungan. Perbedaan yang terdapat di antara E-R *modelling* dan *semantic object model* yang utama terletak pada desain *database*-nya.

2.2.1.2 *Semantic Object Diagram*

Menurut Kroenke (2006, p615), dalam *semantic object*, terdapat 7 tipe objek dari *Semantic Object Diagram* yaitu:

1. *Simple object*

   Yaitu *object* yang memiliki *single value, simple,* atau *group attribute.*

2. *Composite object*

   Yaitu *object* yang terdiri atas satu atau lebih *multi value simple* atau *group attribute* tetapi bukan merupakan *object attribute.*

3. *Compound object*

   Yaitu *object* yang terdiri atas paling sedikit satu atribut *object.*

4. *Hybrid object*

   Yaitu objek yang merupakan kombinasi antara c*omposite object* dan *compound object.*

5. *Association object*

   Yaitu *object* yang berhubungan dengan dua atau lebih atribut objek dan memiliki hubungan yang khusus antar tabel.

6. *Parent* atau *subtype object*

   Sama halnya dengan konsep *class*, *parent* adalah *class* utama dan *subtype* adalah turunan dari *object* lain yang mengandung seluruh atribut dari *object* tersebut.

7. *Archetype* atau *version objects*

   Yaitu s*emantic object* yang menghasilkan *semantic object* lain yang mewakili versi, rilis, atau edisi dari *archetype.*

2.2.1.3 Atribut

Menurut Kroenke (2006, p611), atribut dalam *Semantic Object Model* mendefinisikan karakteristik dari *semantic object* itu sendiri. Ada 3 jenis atribut dari *semantic object* yaitu :

1. *Simple attribute*

   Yaitu atribut sederhana yang mempunyai satu elemen tunggal.

   Contoh: Kode_Dosen pada *object* Dosen

2. *Group attribute*

   Yaitu gabungan dari beberapa elemen tunggal yang memiliki sifat yang sama.

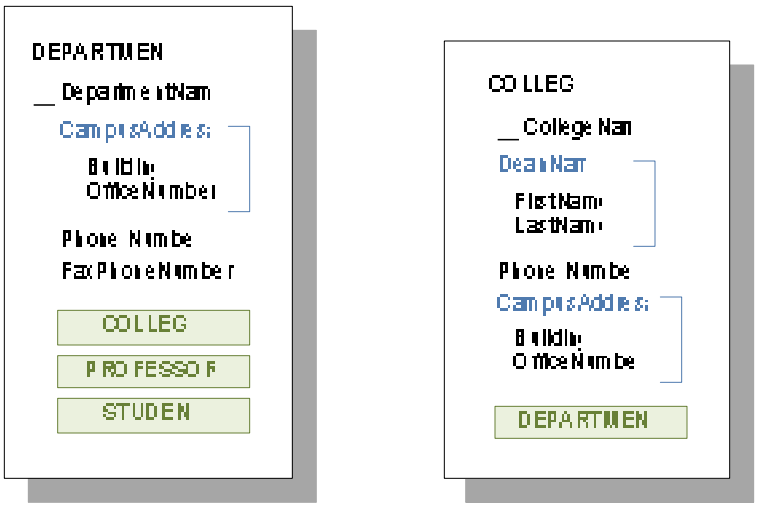
   Contoh: alamat pada *object* dosen terdiri atas kota, jalan, kode pos, dan propinsi.

3. *Semantic Object attribute*

   Yaitu atribut yang berfungsi untuk menghubungkan satu *semantic object* dengan *semantic object* lainnya.

Berikut ini adalah contoh dari s*emantic object diagram*:

Object Department berhubungan dengan *object* College maka nama *object* tersebut masuk atau tercantum dalam *object* diagram seperti contoh di bawah ini:



Gambar 2.3 Contoh *Semantic Object Diagram* (Kroenke)

Kelebihan menggunakan *semantic object model* adalah terdapat pada integrasi data, misalkan pada contoh *Semantic Object Diagram* di atas, data pada *object Department* yang akan ditampilkan, maka data *College* yang berada di *Department* tersebut akan otomatis termasuk di dalamnya sehingga data yang dihasilkan lebih akurat.

Keuntungan lain dalam menggunakan *semantic object model* adalah untuk meminimalisasi terjadinya *cycle* yang berulang seperti yang terdapat pada *Entity Relationship Diagram* pada permodelan E-R *modelling*. Selain itu juga akan mengurangi adanya redudansi data yang terjadi dikarenakan adanya *relationship* yang pasti yang menghubungkan *class* yang satu dengan *class* yang lainnya.

2.2.1.4 *Object Identifier*

Menurut Kroenke (2006, p613), o*bject identifier* adalah satu atau lebih dari atribut *object* yang digunakan untuk mengidentifikasi *object instances*. Dalam *semantic object diagram*, *object identifier* ditunjukkan dengan huruf ID di depan atribut. Jika *identifier*-nya unik, maka ID perlu digarisbawahi. Jika sebuah atribut digunakan sebagai *identifier*, maka atribut tersebut harus mempunyai sebuah nilai. Misalnya dalam *object* Dosen, *identifier* yang digunakan adalah kode_dosen sebagai *identifier* yang bersifat unik. *Object identifier* biasa disebut sebagai *primary key* yang biasa digunakan dalam E-R *modelling*. Yang dimaksud dengan *group identifier* adalah sebuah *identifier* yang mempunyai lebih dari satu atribut. Misalnya nama mahasiswa memiliki nama depan dan nama belakang.

2.2.1.5 Atribut *Cardinality*

Menurut Kroenke (2006, p612), setiap atribut dalam *semantic object* mempunyai *minimum cardinality* dan *maximum cardinality*. *Cardinality* dalam permodelan E-R *modelling* biasa disebut dengan *multiplicity*. *Minimum cardinality* menunjukkan jumlah dari *instances* atribut pada sebuah objek biasanya bernilai 0 atau 1. Jika *minimum*

cardinality-nya bernilai 0, maka atribut tersebut tidak membutuhkan sebuah nilai. Sedangkan jika *minimum cardinality*-nya bernilai 1, maka harus mempunyai nilai. *Maximum cardinality* menunjukkan jumlah *maximum* dari *instances* atribut pada sebuah objek, biasanya antara 1 dan N. Jika *maximum cardinality*-nya bernilai 1, maka atribut hanya mempunyai nilai tidak lebih dari 1, sedangkan jika *maximum cardinality*-nya bernilai N, maka atribut dapat memiliki banyak nilai. Misalnya atribut kode_dosen pada *object* dosen mempunyai *minimum cardinality* dan *maximum cardinality* 1.1 yang menunjukkan bahwa minimal *object* mahasiswa harus mempunyai 1 kode dosen dan maksimal *object* dosen juga hanya terdapat satu.

2.2.1.6 *Object Instances*

Objek instances adalah nilai dari masing-masing atribut pada tiap-tiap objek.

2.2.1.7 *Paired Attributes*

Dalam *semantic object model* tidak hanya mempunyai satu *relationship* tetapi terdapat *two-ways relationship*. Jika dalam suatu *object* memiliki *object* yang lain, maka *object* yang kedua akan memiliki *object* yang pertama. Misalnya jika

dosen memiliki atribut pada *object* tipe_dosen, maka tipe_dosen juga akan mengandung atribut *object* dosen.

2.2.1.8 Perbandingan antara *Semantic Object Model* dengan *Entity Relationship Modelling*

Berikut ini adalah perbedaan yang ada pada *semantic object model* dengan E-R *modelling*, yaitu :

1. Perbedaan yang paling utama terletak pada orientasinya. *Semantic object model* menggunakan konsep *semantic object* sebagai dasarnya sedangkan E-R *modelling* menggunakan konsep *entity* dasar.
2. Pada *semantic object model* tidak terdapat *relationship* sedangkan pada E-R *modelling* terdapat *relationship*.
3. Pada *semantic object model* yang menjadi *key* yang *unique* disebut dengan *object identifier* sedangkan pada E-R *modelling* disebut dengan *primary key*.
4. *Semantic object model* adalah suatu permodelan basis data yang menggunakan *semantic object* dalam membangun suatu *relationship* sedangkan E-R *modelling* adalah suatu permodelan yang mempunyai *entity*, *relationship* dan lain-lain yang berhubungan satu sama lain.

Permodelan *semantic object model* lebih dapat disesuaikan dalam pengembangan *Enterprise Resource Planning* (ERP) pada sistem yang akan dirancang karena dalam permodelan *semantic object model* akan mengurangi adanya redudansi data yang terjadi dengan sifat permodelan *semantic object model* yang efektif dibandingkan dengan permodelan E-R *Modelling*. Dalam konsep E-R *Modelling* mengenal adanya *Entity Relationship Diagram* yang apabila digunakan dalam sebuah organisasi atau perusahaan berskala *enterprise* dengan proses bisnis yang cukup kompleks maka akan menimbulkan adanya *data cycle* (perputaran data) yang seharusnya tidak boleh terjadi dalam model basis data. *Semantic object* akan mengurangi adanya *data cycle* tersebut karena dengan adanya object identifier yang berada pada setiap tabel dan menghubungkan semua tabel dalam satu rancangan basis data, maka akan mengurangi terjadinya perputaran data dan *user* akan lebih mudah dalam memahami sistem yang sedang berjalan tersebut.

Maka dengan kata lain, konsep dalam *semantic object model* dapat lebih *compatible* dengan *Enterprise Resource Planning* (ERP) yang akan membantu DBMS dalam mengolah

dan memproses data menjadi lebih efektif dengan waktu yang lebih efisien.

2.2.2 *Enterprise Resource Planning* (ERP)

Menurut Lau (2005, p10), *Enterprise Resource Planning* (ERP) adalah proses mengintegrasi semua fungsi dan proses bisnis dari sebuah organisasi serta memberikan keunggulan yang kompetitif pada perusahaan. ERP merupakan singkatan dari *Enterprise* (perusahaan / organisasi), *Resource* (sumber daya), dan *Planning* (perencanaan). ERP digunakan untuk merencanakan dan mengelola sumber daya organisasi agar dapat dimanfaatkan secara optimal untuk menghasilkan nilai tambah bagi seluruh pihak yang memiliki kepentingan atas organisasi tersebut.

Berdasarkan definisi tersebut, dapat disimpulkan bahwa ERP merupakan suatu sistem informasi yang mendefinisikan dan merencanakan sumber daya yang dibutuhkan suatu organisasi untuk mengintegrasikan semua fungsi bisnis yang ada agar dapat lebih responsif dan objektif terhadap berbagai kebutuhan yang diperlukan.

2.2.2.1 Kelebihan dalam Implementasi ERP

Kelebihan dalam mengunakan ERP adalah sebagai berikut:

1. ERP berbasiskan *web* sehingga mengurangi dokumentasi dengan format kertas. Selain itu memberikan format presentasi yang lebih baik
2. Menyajikan informasi yang lebih akurat
3. Meningkatkan waktu *update* informasi
4. Memberikan respon dan *follow up* yang lebih baik kepada pelanggan
5. Pemantauan yang lebih baik kepada *query*
6. Memberikan respon yang cepat terhadap operasi bisnis dan perubahan pasar
7. Menyediakan *database customer* yang *unified* dan dapat digunakan oleh semua aplikasi
8. Meningkatkan akses informasi dan manajemen pada seluruh divisi perusahaan

2.2.2.2 Kekurangan dalam Implementasi ERP

Kekurangan dalam mengunakan ERP adalah sebagai berikut :

1. Biaya instalasi yang tinggi
2. Implementasi ERP pada suatu perusahaan membutuhkan waktu yang tidak sedikit, sehingga dapat menurunkan kinerja perusahaan tersebut pada waktu implementasi

3. Perlunya dilakukan pelatihan terhadap karyawan agar dapat beradaptasi dengan sistem yang baru

2.2.3 Sistem *Enterprise Resource Planning* (ERP)

Menurut Olson (2004, p9), sistem *Enterprise Resource Planning* (ERP) adalah sistem informasi terintegrasi yang dibangun untuk menerapkan praktik-praktik terbaik terhadap komputasi organisasi.

Sistem ERP dibagi atas beberapa sub-sistem yaitu sistem finansial, sistem distribusi, sistem manufaktur, sistem *maintenance,* dan sistem *human resource.*

Menurut Daniel E. O'Leary (2000, p27), sistem ERP memiliki beberapa ciri yaitu :

1. Sistem ERP adalah sistem perangkat lunak yang didesain untuk lingkungan pelanggan *user server*, apakah itu secara tradisional atau berbasis jaringan
2. Sistem ERP memadukan sebagian besar dari proses bisnis
3. Sistem ERP menggunakan *database* perusahaan yang secara selektif dalam menyimpan setiap data sekali saja
4. Sistem ERP memungkinkan mengakses data secara waktu nyata (*real time*)

5. Sistem ERP menunjang sistem multi mata uang dan bahasa, yang sangat diperlukan oleh perusahaan multinasional
6. Sistem ERP memungkinkan penyesuaian untuk kebutuhan khusus perusahaan tanpa melakukan pemrograman kembali
7. Sistem ERP merupakan *application service provider*