BAB II

TINJAUAN PUSTAKA

2.1 Tinjauan Umum Perusahaan

2.1.1 Sejarah Singkat Perusahaan

Badan Kepegawaian Daerah (BKD) Kabupaten Bandung dibentuk berdasarkan Peraturan Daerah Nomor 10 Tahun 2002 Tentang Pembentukan Lembaga Teknis Daerah. Pembentukan BKD mengacu kepada kebijakan Pemerintah Pusat, yaitu Undang – Undang Nomor 43 Tahun 1999 Pasal 34 A ayat (1) ”Untuk kelancaran pelaksanaan Manajemen Pegawai Negeri Sipil, dibentuk Badan Kepegawaian Daerah dan yang dimaksud Pasal 34 A tersebut bahwa Badan Kepegawaian Daerah adalah Perangkat Daerah yang dibentuk oleh Kepala Daerah “ dan kemudian Keputusan Presiden RI. Nomor 159 Tahun 2000 Tentang Pedoman Pembentukan Badan Kepegawaian Daerah.

Beberapa Tugas, Pokok dan Fungsi BKD adalah menyelenggarakan manajemen Pegawai Negeri Sipil (PNS) yang mencakup Perencanaan, Pengembangan Kualitas Sumber Daya Manusia (SDM) PNS dan Administrasi Kepegawaian, Pengawasan dan Pengendalian, Penyelenggaraan dan Pemeliharaan Informasi Kepegawaian, mendukung Perumusan Kebijaksanaan Kesejahteraan PNS serta memberikan bimbingan teknis kepada unit organisasi yang menangani Kepegawaian pada Dinas/Badan/Lembaga di Daerah. Secara kelembagaan Badan Kepegawaian Daerah (BKD) Kabupaten Bandung mempunyai persamaan dengan BKD yang terdapat di Kabupaten/ Kota lainnya, namun mempunyai

perbedaan dalam Struktur Bidang yang disebabkan oleh adanya kebutuhan organisasi pemerintahan, diantaranya mempunyai karakteristik yang berbeda dalam sisi penekanan Tugas, Pokok dan Fungsi (TUPOKSI).

Selanjutnya sesuai dengan Peraturan Daerah Kabupaten Bandung Nomor 21 Tahun 2006 BKD digabung dengan Kantor Pendidikan dan Pelatihan sehingga berubah nama menjadi Badan Kepegawaian, Pendidikan dan Pelatihan sejak ditetapkan pada Lembaran Daerah Kabupaten bandung pada tanggal 17 Desember 2007 dan diundangkan pada tanggal 9 Januari 2008.

2.1.2 Visi dan Misi Perusahaan

1. Visi

Visi Badan Kepegawaian, Pendidikan dan Pelatihan (BKPP) Kabupaten Bandung adalah " *Terwujudnya Kualitas Pelayanan Kepegawaian yang Efektif dan Efisien "*.

Adapun maknanya adalah sebagai berikut :

1. Kualitas pelayanan kepegawaian yaitu Pengelolaan Kepegawaian dalam melaksanakan fungsi dan tugasnya mampu memberikan pelayanan administrasi kepegawiaan secara cepat, tepat dan akurat.
2. Pengelolaan Yang Efektif dan Efisien yaitu Pengelolaan kepegawaian yang dapat memberikan pelayanan secara optimal dari seluruh Fungsi Badan dengan menggunakan sumber daya yang tepat (pembiayaan, waktu dan tenaga)

2. Misi

Misi Badan Kepegawaian, Pendidikan dan Pelatihan (BKPP) Kabupaten Bandung adalah sebagai berikut

1. Meningkatkan Kualitas Pelayanan Administrasi Kepegawaian.

   Titik akhir dari kegiatan BKPP secara eksplisit dapat diukur melalui sejauhmana pelayanan yang telah diberikan kepada aparatur dalam urusan atau hal yang menyangkut Adimistrasi Kepegawaian, baik dari sisi waktu, kemudahan dan informasi.

   Tujuan dari misi ini adalah Pelayanan prima administrasi kepegawaian. Sasaran dari misi ini adalah Meningkatnya kualitas pelayanan administrasi kepegawaian dan Meningkatnya kualitas informasi kepegawaian (SIMPEG). Sedangkan Kebijakan dari misi ini adalah Pengelolaan Sumber Daya Manusia Aparatur

2. Meningkatkan Pembinaan dan Kesejahteraan Pegawai.

   Dengan mempunyai tugas merencanakan, mengkoordinasikan dan mengevaluasi dalam hal ini, BKPP merespon Visi, Misi Kabupaten Bandung untuk mewujudkan pelaksanaan *reward* dan *punishment*..

   Tujuan dari misi ini adalah Terwujudnya pembinaan dan kesejahteraan pegawai. Sasaran dari misi ini adalah Meningkatnya Kinerja dan Kesejahteraan Aparatur. Sedangkan Kebijakan dari misi ini adalah Pengelolaan Sumber Daya Manusia Aparatur

3. Meningkatkan Kualitas Sumber Daya Aparatur.

Bahwa BKPP sebagai unsur staf memiliki fungsi merumuskan rencana pengembangan kualitas SDM aparatur dalam rangka mendukung penyelenggaraan kepemerintahan yang baik/*Good Governance*.

Tujuan dari misi ini adalah Terwujudnya SDM aparatur yang profesional dalam mendukung penyelenggaraan pemerintahan. Sasaran dari misi ini adalah Meningkatnya kualitas SDM Aparatur berlandaskan Iman Taqwa. Sedangkan Kebijakan dari misi ini Pengelolaan Sumber Daya Manusia Aparatur.

2.2 Landasan Teori

2.2.1 Konsep Dasar Sistem

2.2.1.1 Pengertian Sistem

Untuk mengawali pembahasan tentang analisis dan perancangan sistem informasi, pemahaman akan sistem terlebih dahulu harus ditekankan. Definisi sistem berkembang sesuai dengan konteks dimana pengertian sistem itu digunakan. Berikut ini diberikan beberapa definisi sistem secara umum:

1. Kumpulan dari bagian-bagian yang bekerja sama untuk mencapai tujuan yang sama

Contoh :

a. Sistem Tata Surya

b. Sistem Pencernaan

c. Sistem Transfortasi Umum

d. Sistem Otomotif

e. Sistem Komputer

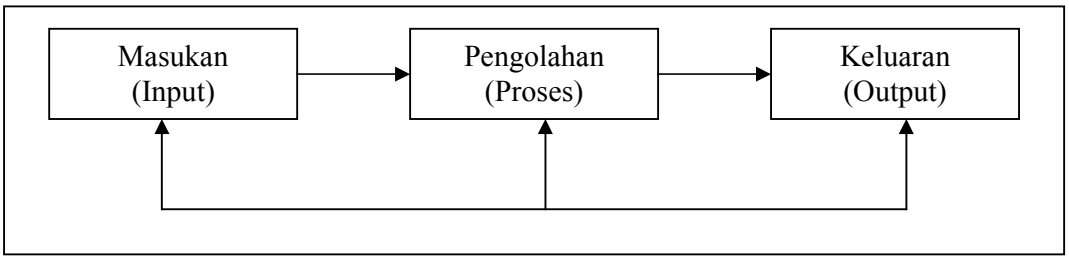
f. Sistem Informasi

2. Sekumpulan Objek-objek yang saling berelasi dan berinteraksi serta hubungan antar objek bisa di lihat sebagai suatu kesatuan yang dirancang untuk mencapai suatu tujuan.

Dengan demikian secara sederhana sistem dapat diartikan sebagai suatu kumpulan atau himpunan dari unsur atau variabel-variabel yang saling terorganisasi, saling terorganisasi dan saling bergantung satu sama lain.

Murdock dan Ross (1993) mendefinisikan sistem sebagai seperangkat elemen yang digabungkan satu dengan yang lainnya untuk suatu tujuan bersama. Sementara definisi sistem dalam kamus *Webster's Unbriged* adalah elemen-elemen yang saling berhubungan dan membentuk satu kesatuan atau organisasi.

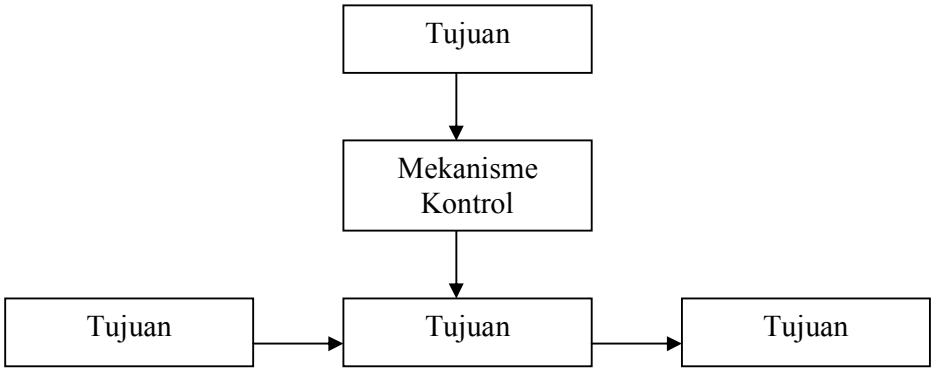
Menurut Scott (1996), sistem terdiri dari unsur-unsur seperti masukan(input), pengolahan(proses) dan keluaran(output). Ciri pokok sistem menurut Gapspert ada empat yaitu sistem itu beroperasi dalam suatu lingkungan terdiri atas unsur-unsur, ditandai dengan saling berhubungan dan mempunyai satu fungsi atau tujuan yang sama.



Gambar 2.1 Model Sistem

Gambar diatas menunjukan bahwa sistem atau pendekatan sistem minimal harus mempunyai empat komponen yakni masukan, pengolahan, keluaran, dan balikan atau control.

Sementera *Mc.Leod(1995)* mendefinisikan sistem sebagai sekelompok elemen-elemen yang terintegrasi dengan maksud yang sama untuk mencapai suatu tujuan. Sumber daya mengalir dari elemen output dan untuk menjamin prosesnya berjalan dengan baik maka dihubungkan dengan mekanisme control. Untuk lebih jelasnya elemen sistem tersebut dapat digambarkan dengan model sebagai berikut:



Gambar 2.2 Model Hubungan Elemen-elemen Sistem

Banyak ahli mengajukan konsep sistem dengan deskripsi yang berbeda, tetapi pada prinsipnya hampir sama dengan konsep dasar sistem pada umumnya. *Schronderberg(1971)* dalam *Suradinata(1996)* secara ringkas menjelaskan bahwa sistem adalah

1. Komponen-komponen yang saling berhubungan satu sama lain
2. Suatu keseluruhan tanpa memisahkan komponen pembentuknya
3. Bersama-sama dalam mencapai tujuan

4. Memiliki input dan output yang dibutuhkan oleh sistem lainnya.
5. Terdapat proses yang mengubah input menjadi output
6. Menunjukanadanya entropi
7. Memiliki aturan
8. Memiliki subsistem yang lebih kecil
9. Memiliki deferensiasi antar subsistem
10. Memiliki tujuan yang sama meskipun mulainya berbeda

2.2.1.2 Karakteristik Sistem

Untuk memahami atau mengembangkan suatu sistem maka perlu membedakan unsur-unsur dari sistem yang membentuknya. Berikut adalah karakteristik sistem yang dapat membedakan suatu sistem dengan sistem lainnya.

1. Batasan (*boundary*) adalah penggambaran dari suatu elemen atau unsur mana yang termasuk di dalam sistem dan mana yang di luar sistem.
2. Lingkungan (*environnment*) adalah segala sesuatu diluar sistem, lingkungan yang menyediakan asumsi, kendalan dan input terhadap suatu sistem
3. Masukan (*input*) adalah sumber daya (data, bahan baku, peralatan, energi) dari lingkungan yang dikonsumsi dan dimanipualasi oleh sistem.
4. keluaran (*output*) adalah sumber daya atau produk (informasi, laporan, dokumen, tampilan layar computer, barang jadi) yang disediakan untuk lingkungan sistem oleh kegiatan dalam suatu sistem.

5. Komponen (*component*) adalah kegiatan-kegiatan atau proses dalam suatu sistem yang mentransformasikan input menjadi bentuk setengah jadi(output). Komponen ini bisa merupakan subsistem dari sebuah sistem.
6. Penghubung (*interface*) adalah tempat dimana komponen atau sistem dan lingkungannya bertemu atau berinteraksi
7. Penyimpanan (*storage*) adalah area yang dikuasai dan digunakan untuk penyimpanan sementara dan tetap dari informasi, energi, bahan baku, dan sebagainya. Penyimpanan merupakan suatu media penyangga di antara komponen tersebut bekerja dengan berbagai tingkatan yang ada dan memungkinkan komponen yang berbeda dari berbagai data sama.

2.2.1.3 Pengertian Subsistem

Suatu sistem yang kompleks biasanya tersusun atas beberapa subsistem. Subsistem bisa dijelaskan sebagai sebuah sistem dalam sistem yang lebih besar. Sebagai contoh:

Automobile adalah sistem yang terdiri dari beberapa subsistem berikut:

a. Sistem Mesin

b. Sistem Body

c. Sistem Roda

Setiap susbsistem bisa terdiri dari beberapa subsistem berikut: Dalam Sistem mesin adalah Sistem karburator, sistem generator, sistem bahan bakar dan lain-lain.

2.2.1.4 Sistem Yang buruk

Untuk menghindari pengembangan suatu sistem yang buruk, perlu diketahui beberapa ciri dari sistem yang buruk :

1. Tidak memenuhi kebutuhan pengguna
2. *Performance* buruk
3. Reliabilitas rendah
4. Kegunaan rendah
5. Contoh-contoh kesulitan : tidak terjadwal, tidak ada rencana anggaran, walaupun bisa jalan tetapi over budget atau over jadwal.

2.2.1.5 Beberapa konsep sistem yang penting

Untuk lebih mudah memahami pengertian sistem dan sistem informasi lebih jauh maka perlu diingat beberapa konsep yang penting dalam pengembangan sistem, yaitu:

1. Dekomposisi

Untuk menganalisa dan memahami secara menyeluruh sebuah sistem yang besar, biasanya membutuhkan waktu yang cukup lama. Untuk mempermudah pekerjaan ini digunakan konsep dekomposisi. Dekomposisi adalah pembagian sistem kedalam komponen-komponen yang lebih kecil(subsistem). Dekomposisi memiliki beberapa keuntungan diantaranya:

a. Analis menjadi lebih mudah mengatur dan menganalisa setiap subsistem secara detail

b. Pada pengembangan sistem, sistem bisa dibagi menjadi beberapa modul, pengembangan modul bisa dilakukan secara paralel dengan syarat tidak ada ketergantungan antar modul yang dibangun.

2. Modularitas

Konsep modularitas berhubungan dengan dekomposisi, pada saat dekomposisi, diharapkan sistem yang besar terbagi menjadi subsistem-subsistem yang relatif sama ukurannya. Dengan modul-modul ini maka beban kerja mengembangkan sistem bisa didistribusi secara merata pada semua sumber daya yang ada. Pengembangan sistem bisa lebih sederhana karena hanya fokus pada satu modul terlebih dahulu, baru dilakukan integrasi antar modul.

3. Coupling

Dari modul-modul yang kita perloleh kadang-kadang ditemukan beberapa modul yang memiliki ketergantungan dengan modul yang lain. Pada kasus ini, modul-modul yang bergantung harus dipasangkan. Dengan cara ini bisa diketahui modul yang bisa bekerja secara independen dan modul-modul yang harus diselesaikan terlebih dahulu sebelum modul yang lain bisa bekerja.

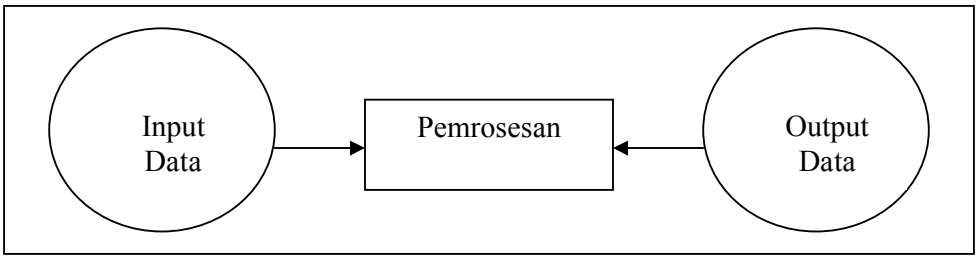
4. Kohesi

Dengan proses coupling, kita bisa dapatkan kelompok-kelompok modul dengan karakteristik yang hampir sama. Di sini muncul konsep kohesi dimana kelompok modul itu harus dianalisis bersama-sama dengan kelompok modul yang saling berkohesi.

2.2.1.7 Pengertian Sistem Informasi

Untuk memahami pengertian sistem informasi, harus dilihat keterkaitan antara data dan informasi sebagai entitas penting pembentuk sistem informasi. Data merupakan nilai, keadaan atau sifat yang berdidi sendiri lepas dari konteks apapun. Sementara informasi adalah data yang telah diolah menjadi sebuah bentuk yang berarti bagi penerimanya dan bermanfaat bagi pengambilan keputusan saat ini maupun yang akan datang (*Davis, 1995)* mengatakan bahwa informasi adalah data yang telah diproses, atau data yang memiliki arti.

Akhirnya sistem informasi manajemen (SIM) dapat didefinisikan sebagai suatu alat untuk menyajikan informasi dengan cara sedemikian rupa sehingga bermanfaat bagi penerimanya. Tujuannya adalah untuk menyajikan informasi guna pengambilan keputusan pada perencanaan, pemrakarsaan, pengeorganisasian, pengendalian kegiatan operasi subsistem suatu perusahaan dan menyajikan sinergi organisasi pada proses. Dengan demikian sistem informasi berdasarkan konsep IPO / input proses output yang dapat dilihat pada gambar dibawah ini :



Gambar 2.3 Konsep Sistem Informasi

2.2.1.8 Komponen Sistem Informasi

Stairs (1992) menjelaskan bahwa sistem informasi berbasis komputer (CBIS) dalam suatu organisasi terdiri dari komponen-komponen berikut:

1. Perangkat keras, yaitu perangkat keras komponen untuk melengkapi kegiatan memasukan data, memproses data, dam keluaran data.
2. Perangkat lunak, yaitu program dan instruksi yang diberikan ke komputer.
3. *Database*, yaitu kumpulan data dan informasi yang diorganisasikan sedemikian rupa sehingga mudah diakses pengguna sistem informasi.
4. Telekomounikasi, yaitu komunikasi yang menghubungkan antara pengguna sistem dengan sistem komputer secara bersama-sama kedalam suatu jaringan kerja yang efektif.
5. Manusia, yaitu personal dari sistem informasi, meliputi manager, analis, programmer, dan operator serta bertanggung jawab terhadap perawatan sistem.

Prosedur, yakni tata cara yang meliputi strategi, kebijakan, metode, dan peraturan-peraturan dalam menggunakan sistem informasi berbasis komputer.

Sementara Burch dan Grudnistki (1986) berpendapat, sistem informasi yang terdiri dari komponen-komponen diatas disebut dengan istilah blok bangunan (*building block*), yaitu blok masukan, blok model, blok keluaran, blok teknologi, dan blok kendali. Sebagai suatu sistem, keenam blok tersebut masing-masing saling berinteraksi satu dengan yang lainnya membentuk satu kesatuan untuk mencapai sasarannya.

1. Blok masukan. Input mewakili data yang masuk kedalam sistem informasi. Input disini termasuk metode-metode dan media untuk menangkap data yang akan dimasukkan yang dapat berupa dokumen-dokumen besar
2. Blok model. Blok ini terdiri dari kombinasi prosedur, logika dan matematika yang akan memanipulasi data input dan data yang tersimpan dibasis data dengan cara tertentu untuk menghasilkan keluaran yang diinginkan.
3. Blok keluaran. Produk sistem informasi adalah keluaran yang merupakan informasi yang berkualitas dan dokumentasi yang berguna untuk semua tingkat manajemen serta semua pemakai sistem
4. Blok teknologi. Teknologi merupakan kotak atau tool-box dalam sistem informasi. Teknologi digunakan untuk menerima input, menjalakan model, menyimpan dan mengakses data, menghasilkan sekaligus mengirimkan keluaran dan membantu pengendalian dari sistem secara keseluruhan.
5. Blok database. Database merupakan kumpulan dari data yang saling berhubungan satu dengan yang lainnya, tersimpan diperangkat keras komputer dan digunakan perangkat lunak untuk memanipulasinya.
6. Blok kendali. Pengendalian perlu dirancang dan diterapkan untuk meyakinkan bahwa hal-hal yang dapat merusak sistem dapat dicegah atau bila terlanjur terjadi kesalahan-kesalahan dapat langsung cepat diatasi.

Sementara menurut *Davis (1995),* sistem informasi menajemen terdiri dari elemen-elemen berikut:

1. Perangkat keras komputer *(Hardware)*
2. Perangkat lunak *(Software)* yang terdiri dari perangkat lunak sistem umum, perangkat lunak terapan dan program aplikasi
3. Database
4. Prosedur
5. Petugas Operasional

2.2.2 Pemodelan Data

Proses model menggambarkan keseluruhan proses bisnis yang akan dilakukan oleh sistem informasi yang akan dibangun. Proses model juga menjelasan data-data yang terlibat dalam proses-proses tersebut. Tetapi proses model tidak menggambarkan bagaimana data diorganisir dan dikelompokkan. Juga tidak menggambarkan hubungan logis antar kelompok data. Untuk masalah seperti ini diperlukan pemodelan lain yaitu pemodelan data.

2.2.2.1 Data Model

Data model adalah cara formal untuk menggambarkan data yang digunakan dan diciptakan dalam suatu sistem bisnis. Model ini menunjukan orang, tempat atau benda dimana data diambil dan hubungan antar data tersebut. Pemodelan data juga dibedakan menjadi dua yaitu model data logis dan model data fisik. Model data logis menunjukan pengaturan data tanpa mengindikasikan bagaimana data tersebut disimpan, dibuat, dan dimanipulasikan. Model data fisik menunjukan bagaimana data akan disimpan sebenarnya dalam database atau file. Penyusunan

pemodelan data harus seimbang dengan pemodelan proses. Salah satu cara pemodelan data adalah dengan ERD *(entity Relationship Diagram).*

2.2.2.2.Entity Relationship Diagram (ERD)

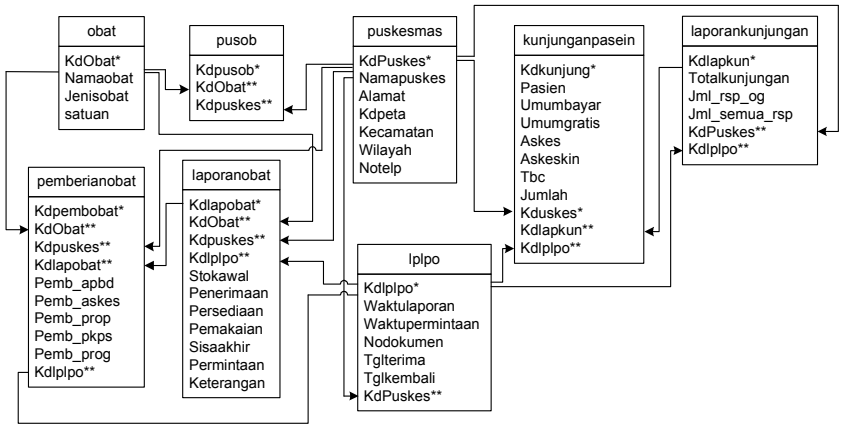
ERD adalah gambar atau diagram yang menunjukan informasi yang dibuat, disimpan, dan digunakan dalam sistem bisnis. Entitas biasanya menggambarkan jenis informasi yang sama. Dalam entitas biasanya digunakan untuk menghubungkan antar entitas yang sekaligus menunjukan hubungan antar data menunjukan hubungan antar data. Pada akhirnya ERD bisa digunakan untuk menunjukan aturan-aturan bisnis yang ada pada sistem informasi yang akan dibangun.

Ada beberapa poin yang bisa dilihat dalam menggunakan ERD untuk menunjukan aturan bisnis :

1. Aturan bisnis adalah batasan yang harus diikuti ketika sistem beroperasi.
2. Simbol ERD hanya menunjukan satu instance dari entitas harus ada sebelum instance lain dari suatu entitas. Sebagai contoh seorang dokter haru ada sebelum perjanjian ketemu dengan dokter dibuat.
3. Simbol ERD dapat menunjukan ketika salah satu instance dari suatu entitas dapat direlasikan dengan satu anggota atau lebih dari entitas lainnya. Sebagai contoh, satu dokter bisa memiliki banyak pasien, satu pasien bisa jadi hanya memiliki satu dokter utama.
4. Simbol ERD juga menunjukan ketika eksistensi dari suatu instance dalam suatu entity adalah opsional untuk sebuah relasi dengan instance lain dari

suatu entitas. Sebagai contoh, pasien mugkin memiliki atau mungkin tidak memiliki biaya asuransi.

Pada gambar dibawah ini dapat dilihat suatu ERD yang masih belum memiliki kardinalitas dan modalitas, biasanya disebut juga dengan Skema Relasi :

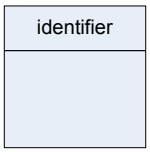


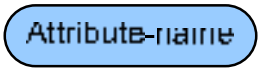
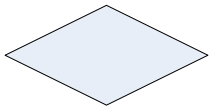
Gambar 2.4 Skema Relasi

2.2.2.3.Elemen-Elemen ERD

Seperti data flow diagram, ERD juga menggunakan simbol-simbol khusus untuk menggambarkan elemen-elemen ERD. Berikut simbol-simbol yang digunakan dalam ERD.

Tabel 2.1 Tabel Simbol-simbol dalam ERD

	IDEF1X	**Chen**	**Information Engineering**
Entitas: Orang, tempat, atau benda. Memiliki nama tunggal. Ditulis dengan huruf besar. Berisi lebih	Entity-Name 	Entity-Name 	Entity-Name 

dari satu instance			
Attribute : Properti dari entitas, Harus digunakan oleh minimal 1 proses bisnis, Dipecah dalam detail	Atribute-Name 	Atribute-Name 	Atribute-Name 
Relationship : Menunjukan hubungan antar 2 entitas. Dideskripsikan dengan kata kerja. Memiliki modalitas (null/not null). Memiliki kardinalitas (1:1, 1:N, N:N)	Relationship-Name 	Relationship-Name 	Relationship-Name

Keterangan

Entitas : Entitas bisa berupa orang, kejadian, atau benda dimana data akan dikumpulkan. Untuk menjadi sebuah entitas, sebuah objek harus menampilkan beberapa event. Sebagai contoh, jika sebuah firma memiliki satu gudang maka firma tersebut bukan sebuah entitas. Tetapi jika perusahaan memiliki banyak gudang, maka gudang bisa menjadi instance sebuah entitas jika perusahaan ingin menyimpan data untuk setiap anggota dari gudang.

***Attribute* :** aturan pembuatan attribute adalah sebagai berikut :

1. Informasi yang diambil tentang sebuah entitas
2. Hanya yang digunakan oleh organisasi yang dimasukkan dalam model
3. Nama attribute harus merupakan kata benda

***Identifier* :** aturan pembuatan identifier/pengenal :

1. Satu atau lebih attribute dapat menjadi identifier entitas, yang secara unik mengidentiikasi setiap anggota dari entitas.
2. Concatenated identifier /identifier gabungan terdiri dari beberapa attribute
3. Identifier bisa menjadi artifisial, seperti dengan membuat nomor ID
4. Identifier tidak akan dikembangkan sampai fase desain

***Relationship* :** aturan pembuatan relasi adalah

1. Hubungan antar entitas
2. Entitas pertama dalam relationship disebut entitas induk, entitas kedua disebut entitas anak
3. Relationship harus memiliki nama yang berupa kata kerja
4. Relationship berjalan arah
5. Sebagai contoh, jika dimiliki dua entitas yaitu buku dan toko buku, maka bisa dibuat relationship diantaranya :
    1. Toko buku memesan buku
    2. Toko buku menampilkan buku
    3. Toko buku menstock buku
    4. Toko buku menjual buku

5. Toko buku mengembalikan buku

Kardinalitas : aturan pembuatan kardinalitas adalah

1. Kardinalitas mengacu pada beberapa kali instance dari suatu entitas dapat berelasi dengan instance lain di entitas yang berbeda
2. Satu entitas dalam suatu instance mengacu pada satu dan hanya satu instance pada entitas lainnya (1:1)
3. Satu instance dalam satu entitas mengacu ke satu atau lebih instance yang berelasi (1:N)
4. Satu atau lebih instance dalam suatu entitas mengacu pada satu atau lebih instance pada entitas yang berelasi (M:N)

Modalitas : aturan pembuatan modalitas adalah

1. Mengacu pada apakah suatu instance dari entitas anak dapat ada tanpa suatu relasi dengan instance dari entitas induk atau tidak
2. Not null, berarti bahwa suatu instance pada entitas yang berelasi harus ada untuk suatu instance dari entitas lain untuk disebut vallid
3. Null, berarti bahwa tidak ada instance dalam entitas yang berelasi yang diperlukan untuk instance pada relasi lain untuk dikatakan valid

2.2.2.4.Tahap Pembuatan ERD

Untuk membuat ERD, diperlukan latihan dan jam terbang. Ada beberapa pedoman yang perlu diperhatikan untuk membuat ERD, diantaranya :

1. Entitas harus memiliki banyak realitas

2. Hindari penggunaan atribut yang tidak perlu
3. Berilah label yang jelas untuk setiap komponen
4. Pasangkan kardinalitas dan modalitas yang jelas dan benar
5. Pecah attribute menjadi level serendah mungkin yang diperlukan
6. Label harus merefesikan istlah-istilah bisnis yang umum
7. Asumsi harus disebutkan dengan jelas

Normalisasi adala teknik yang digunakan untuk memvalidasi model data. Serangkaian aturan diberlakukan pada data model logis untuk meningkatkan pengaturannya. Biasanya ada tiga aturan yang digunakan.

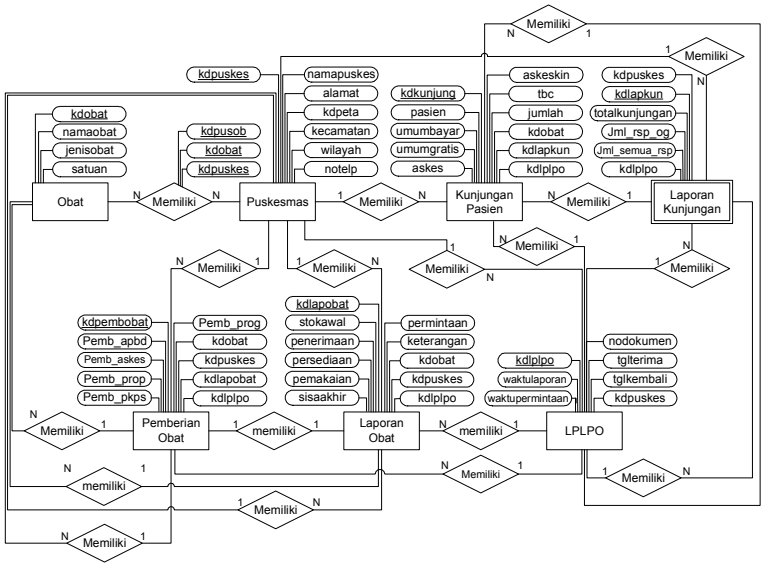
Langkah-langkah yang biasa digunakan untuk melakukan normalisasi terhadap data model yang telah kita peroleh adalah sebagai berikut :

1. **First Normal Form (1NF) :** cari kelompok-kelompok atribute yang berulang dan dipisahkan kedalam entitas yang berbeda.
2. **Second Normal Form (2NF) :** Jika ada entitas yang memiliki identifier gabungan, cari attribute yang hanya bergantung pada identifier. Jika ditemukan, pindahkan ke entitas baru.
3. **Third Normal Form (3NF) :** cari attribute yang bergantung hanya pada attribute lain yang bukan merupakan identifier, jika ditemukan pindahkan menjadi entitas baru. Pindahkan juga attribute-attribute yang dipindahkan.

Setelah membuat normal ERD, langkah selanjutnya adalah menyeimbangkan ERD dengan DFD. Semua aktivitas analisis merupakan aktivitas-aktivitas yang saling berkaitan termasuk pemodelan proses dan

pemodelan data. Proses model akan berisi dua hal, yaitu data flow dan data store. Komponen data dalam DFD ini harus diseimbangkan dengan ERD, dimana data store diseimbangkan dengan entitas dan elemen data diseimbangkan dengan dengan attribute. Untuk mempermudah, banyak tool case yang menyediakan fiture untuk mengecek ketidak seimbangan.

Berikut ini adalah contoh ERD yang sudah normal, ERD ini menangani pengelolaan data puskesmas dan data obat yang ada disebuah dinas kesehatan.



Gambar 2.5 Contoh ERD Pengelolaan Obat

2.2.3. MySQL

MySQL merupakan susunan salah “ satu konsep utama dalam *database* sejak lama, yaitu SQL (*Structure Query Language*). Kendala dari suatu sistem *database* (DBMS) dapat diketahui dari cara kerja *optimezer*nya dalam melakukan perintah-perintah SQL yang dibuat user ataupun program-program aplikasinya” KAD. Sebagai *database server*, MySQL dapat dikatakan lebih unggul dibanding *database server* lainnya dalam query data. MySQL adalah salah satu dari sekian

banyak sistem *database* yang merupakan terobosan solusi yang tepat dalam aplikasi *database*.

MySQL adalah multi *user database* yang menggunakan bahasa *Strucktured Query Language* (SQL). *MySQL* mampu menangani data yang cukup besar. Perusahaan yang mengembangkan *MySQL* yaitu TcX, mengaku menyimpan data lebih dari 40 *database*, 10.000 tabel dan sekitar 7 juta baris, totalnya kurang lebih 100 *Gigabyte* data. SQL adalah bahasa standar yang digunakan untuk mengakses *database server*. Bahasa ini pada awalnya dikembangkan oleh IBM, namun telah diadopsi 25 dan digunakan sebagai standar industri. Dengan menggunakan SQL, proses akses *database* menjadi lebih *user - friendly* dibandingkan dengan menggunakan *dBASE* atau *Clipper* yang masih menggunakan perintah - perintah pemrograman.*MySQL* merupakan *software database* yang paling populer di lingkungan Linux, kepopuleran ini karena ditunjang performa *query* dari databasenya yang saat ini bisa dikatakan paling cepat dan jarang bermasalah. *MySQL* ini juga sudah dapat berjalan pada lingkungan *Windows*.

Perintah untuk mengelola *database* dibagi menjadi 3 (tiga) kelompok, diantaranya :

1. Perintah untuk mendefinisikan data/DDL (*Data Definition Language*).
2. Perintah untuk memanipulasi data/DML (*Data Manipulation Language*).
3. Perintah untuk mengendalikan data/DCL (*Data Control Language*).

2.2.3.1. DDL *(Data Definition Language)*

DDL atau *Data Definition Language* adalah bagian dari SQL yang digunakan untuk mendefinisikan data dan objek database, dimana terdapat perintah-perintah sebagai berikut :

Tabel 2.2 Perintah DDL

Perintah	Keterangan
CREATE	Untuk mendefinisikan database, maupun tabel sebagai data yang akan disimpan maupun diakses
ALTER	Untuk memodifikasi tabel, baik itu menambah, menghapus, maupun mengganti kolom/field pada tabel
DROP	Untuk menghapus tabel dan database

2.2.3.2. DML (*Data Manipulation Language*)

DML atau *Data Manipulation Language* adalah bagian dari SQL yang digunakan untuk memulihkan dan memanipulasi data.

Tabel 2.3 Perintah DML

Perintah	Keterangan
SELECT	Untuk mengambil atau menampilkan data dari tabel pada database
INSERT	Untuk menyisipkan data pada tabel
UPDATE	Untuk memperbaharui nilai suatu data dalam database
DELETE	Untuk menghapus record pada tabel

2.2.4. Visual Basic

Microsoft Visual Basic (sering disingkat sebagai **VB** saja) merupakan sebuah bahasa pemrograman yang bersifat *event driven* dan menawarkan *Integrated Development Environment* (IDE) visual untuk membuat program aplikasi berbasis sistem operasi Microsoft Windows dengan menggunakan model pemrograman Common Object Model (COM). Visual Basic merupakan turunan bahasa BASIC dan menawarkan pengembangan aplikasi komputer berbasis grafik dengan cepat, akses ke basis data menggunakan Data Access Objects (DAO), Remote Data Objects (RDO), atau ActiveX Data Object (ADO), serta menawarkan pembuatan kontrol ActiveX dan objek ActiveX. Beberapa bahasa skrip seperti Visual Basic for Applications (VBA) dan Visual Basic Scripting Edition (VBScript), mirip seperti halnya Visual Basic, tetapi cara kerjanya yang berbeda.

Para programmer dapat membangun aplikasi dengan menggunakan komponen-komponen yang disediakan oleh Microsoft Visual Basic Program-program yang ditulis dengan Visual Basic juga dapat menggunakan Windows API, tapi membutuhkan deklarasi fungsi eksternal tambahan.

Dalam pemrograman untuk bisnis, Visual Basic memiliki pangsa pasar yang sangat luas. Dalam sebuah survey yang dilakukan pada tahun 2005, 62% pengembang perangkat lunak dilaporkan menggunakan berbagai bentuk Visual Basic, yang diikuti oleh C++, JavaScript, C#, dan Java.

2.2.5. Konsep Dasar Pemrograman Berorientasi Objek (OOP)

Empat konsep yang biasanya dimiliki oleh suatu bentuk pemrograman yang berorientasi objek adalah

a. *Encapsulation*
b. *Abstraction*
c. *Inheritance*
d. *Polymorphism*

Encapsulation adalah penyatuan antara proses-proses yang terjadi dan sejumlah data yang dikelola kedalam suatu bentuk kelompok tertentu. Pengelompokan tersebut adalah dengan dibentuknya suatu objek secara abstrak.

Abstraction memiliki pengertian bahwa dalam mengembangkan perangkat lunak seorang programmer akan menggunakan atau menciptakan sejumlah objek dimana sejumlah objek tersebut akan ada dan berperilaku pada sistem.

Inheritance atau pewarisan adalah merupakan konsep dimana suatu objek memiliki sejumlah perilaku yang dimiliki oleh objek lain dalam suatu hirarki tertentu.

Polymorphism adalah konsep dimana beberapa objek dapat memiliki metode yang sama dengan aksi yang berbeda.

2.2.6. UML

2.2.6.1 Model-Model Diagram pada UML

UML terdiri atas banyak elemen-elemen grafis yang digabungkan membentuk diagram. Tujuan representasi elemen-elemen grafis ke dalam diagram adalah

untuk menyajikan beragam sudut pandang dari sebuah sistem berdasarkan fungsi masing-masing diagram tersebut. Kumpulan dari beragam sudut pandang inilah yang kita sebut sebuah **model**. Diagram-diagram yang tersedia pada UML versi 1.4 ada 9 diagram yang terbagi atas 2 kategori, yaitu :

Structural Diagram :

1. *Class diagram*
2. *Object diagram*
3. *Component diagram*
4. *Deployment diagram*

Behavioral Diagram :

1. *Use case diagram*
2. *Sequence diagram*
3. *Collaboration diagram*
4. *Statechart diagram*
5. *Activity diagram*

2.2.6.2 Class Diagram

Class Diagram adalah sebuah spesifikasi yang jika diinstansiasi akan menghasilkan

Sebuah obyek dan merupakan inti dari pengembangan dan desain berorientasi obyek. *Class* menggambarkan keadaan (atribut/properti) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (metoda/fungsi).

Class diagram menggambarkan struktur dan deskripsi *class, package* dan *object* beserta hubungan satu sama lain seperti *containment*, pewarisan, asosiasi, dan lainlain.

Sebuah *Class* memiliki tiga area pokok :

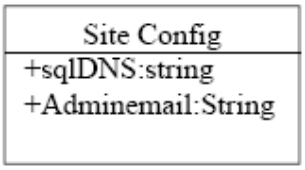
1. Nama, merupakan nama dari sebuah kelas
2. Atribut, merupakan peroperti dari sebuah kelas. Atribut melambangkan batas nilai yang mungkin ada pada obyek dari class
3. Operasi, adalah sesuatu yang bisa dilakukan oleh sebuah *class* atau yang dapat dilakukan oleh *class* lain terhadap sebuah *class*.

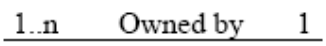
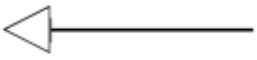
Atribut dan metoda dapat memiliki salah satu sifat berikut :

1. *Private*, tidak dapat dipanggil dari luar *class* yang bersangkutan.
2. *Protected*, hanya dapat dipanggil oleh *class* yang bersangkutan dan anak-anak yang mewarisinya.
3. *Public*, dapat dipanggil oleh siapa saja.
4. *Package*, hanya dapat dipanggil oleh instance sebuah class pada paket yang sama.

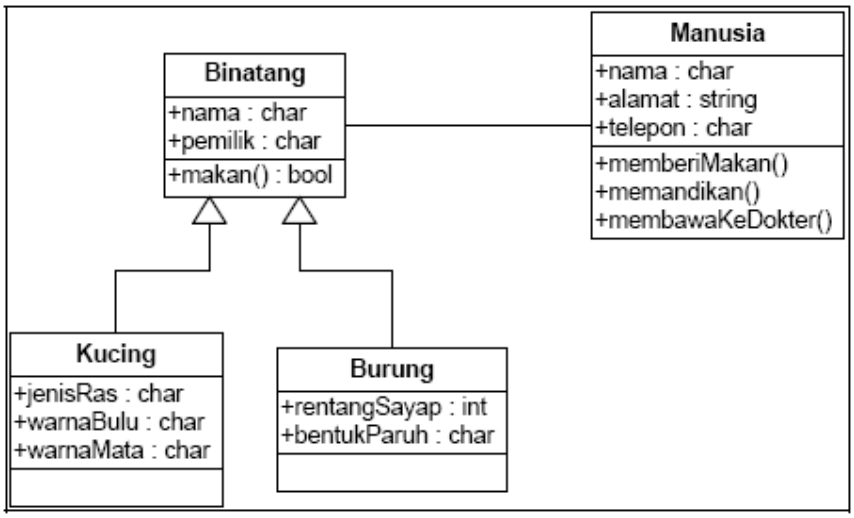
Berikut adalah notasi – notasi yang ada pada class diagram :

Tabel 2.4 Notasi Pada Class Diagram

Class	*Class* adalah blok - blok pembangun pada pemrograman berorientasi obyek. Sebuah class digambarkan sebagai sebuah kotak yang terbagi atas 3 bagian. Bagian atas adalah bagian nama dari *class*. Bagian tengah mendefinisikan property/atribut *class*. Bagian akhir mendefinisikan method-method dari	

	sebuah *class*.	
Assosiation	Sebuah asosiasi merupakan sebuah *relationship* paling umum antara 2 class, dan dilambangkan oleh sebuah garis yang menghubungkan antara 2 *class*. Garis ini bisa melambangkan tipe-tipe *relationship* dan juga dapat menampilkan hukum-hukum multiplisitas pada sebuah *relationship* (Contoh: One-to-one, one-to-many, many-to-many).	
Dependency	Kadangkala sebuah *class* menggunakan *class* yang lain. Hal ini disebut *dependency*. Umumnya penggunaan *dependency* digunakan untuk menunjukkan operasi pada suatu *class* yang menggunakan *class* yang lain. Sebuah *dependency* dilambangkan sebagai sebuah panah bertitik-titik.	
Aggregation	*Aggregation* mengindikasikan keseluruhan bagian *relationship* dan biasanya disebut sebagai relasi "mempunyai sebuah" atau "bagian dari". Sebuah *aggregation* digambarkan sebagai sebuah garis dengan sebuah jajaran genjang yang tidak berisi/tidak solid.	
Generalization	Sebuah relasi *generalization* sepadan dengan sebuah relasi *inheritance* pada konsep berorientasi obyek. Sebuah *generalization* dilambangkan dengan sebuah panah dengan kepala panah yang tidak solid yang mengarah ke kelas "*paren*t"-nya/induknya.	

Contoh Class Diagram :



Gambar 2.6 Contoh Class Diagram

Perhatikan contoh *class diagram* pada gambar 2.8. Diagram ini mempunyai 2 kelas, yaitu kelas burung dan kelas manusia. Masing-masing kelas memiliki atribut dan operasi. Selain itu pada gambar 2.8 juga terdapat konsep inheritansi kelas, yaitu pada kelas binatang, yang inheritansinya diterapkan kepada kucing dan burung.

2.2.6.3 Use Case Diagram

Diagram Use Case adalah diagram yang menunjukkan fungsionalitas suatu sistem atau kelas dan bagaimana sistem tersebut berinteraksi dengan dunia luar dan menjelaskan sistem secara fungsional yang terlihat user, intinya use case merupakan gambaran fungsionalitas dari suatu sistem, sehingga customer atau pengguna sistem paham dan mengerti mengenai kegunaan sistem yang akan

dibangun.. Biasanya dibuat pada awal pengembangan. Use case diagram menggambarkan fungsionalitas yang diharapkan dari sebuah sistem.

Notasi Gambar Yang Diapakai Use Case diantaranya adalah :

1. *Actor*

   Seorang / sebuah aktor adalah sebuah entitas manusia atau mesin yang berinteraksi dengan sistem untuk melakukan pekerjaan-pekerjaan tertentu.

2. *case*

   Menggambarkan deskripsi yang melibatkan actor.

3. *Extend*

   Relasi yang digunakan jika use case yang satu mirip dengan use case yang lain.

4. *Include*

   Relasi jika terdapat perilaku yang mirip dengan beberapa use case.

Cara Menemukan Use Case :

1. Pola perilaku perangkat lunak aplikasi.
2. Gambaran tugas dari sebuah actor.
3. Sistem atau "benda" yang memberikan sesuatu yang bernilai kepada actor.
4. Apa yang dikerjakan oleh suatu perangkat lunak (bukan bagaimana cara mengerjakannya).

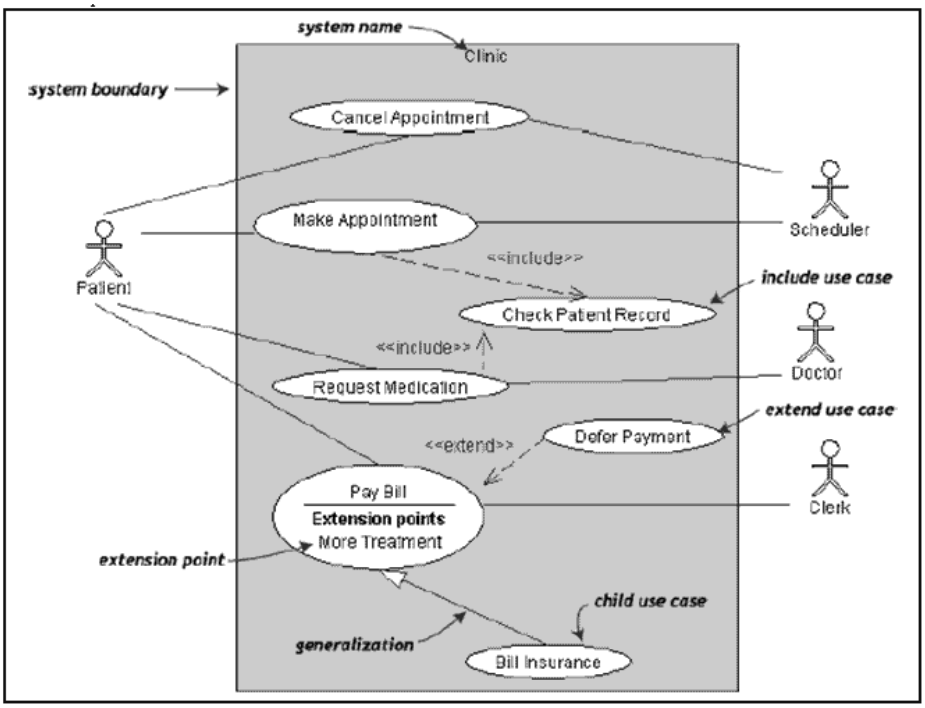
Ada beberapa relasi yang terdapat pada use case diagram:

1. *Association*, menghubungkan link antar element.

2. *Generalization*, disebut juga inheritance (pewarisan), sebuah elemen dapat merupakan spesialisasi dari elemen lainnya.
3. *Dependency*, sebuah element bergantung dalam beberapa cara ke element lainnya.
4. *Aggregation*, bentuk assosiation dimana sebuah elemen berisi elemen lainnya.

Tipe relasi / stereotype yang mungkin terjadi pada Use Case diagram :

1. <<*include*>>, yaitu kelakuan yang harus terpenuhi agar sebuah event dapat terjadi, dimana pada kondisi ini sebuah use case adalah bagian dari use case lainnya.
2. <<*extends*>>, kelakuan yang hanya berjalan di bawah kondisi tertentu seperti menggerakkan alarm.
3. <<*communicates*>>, mungkin ditambahkan untuk asosiasi yang menunjukkan asosiasinya adalah communicates association . Ini merupakan pilihan selama asosiasi hanya tipe relationship yang dibolehkan antara actor dan use case.



Gambar 2.7 Contoh Use Case pada Sistem Rumah Sakit