**BAB II**

**LANDASAN TEORI**

**2.1 Konsep Dasar Sistem**

Defini sistem dalam buku analisa dan design, terhadap dua kelompok pendekatan di dalam mendefinisikan sistem yaitu yang menekankan pada prosedur dan yang menekankan pada komponen atau elemennya.

* 1. Berdasarkan Penekanan Prosedur

Sistem adalah suatu jaringan kerja dari prosedur-prosedur yang saling berhubungan, berkumpul bersama-sama untuk melakukan suatu kegiatan atau untuk menyelesaikan suatu sasaran tertentu.

* 1. Berdasarkan Penekanan Komponen

Sistem adalah kumpulan dari elemen-elemen yang berinteraksi untuk mencapai suatu tujuan tertentu. (Jogiyanto: 2008).

* + 1. **Pengertian Sistem**

Sistem adalah suatu jaringan kerja dari prosedur-prosedur yang saling berhubungan, berkumpul bersama-sama untuk melakukan suatu kegiatan atau menyelesaikan suatu sasaran tertentu. (Yasin: 2012). Sistem adalah suatu kumpulan atau himpunan dari unsur atau variable-variabel yang saling terorganisasi, saling berinteraksi, dan saling bergantung satu sama lain. (Hanif Al Fattan: 2012).

* + 1. **Karakteristik Sistem**

Suatu sistem mempunyai karakteristik .karakteristik sistem adalah sebagai berikut:

1. Komponen-Komponen (*components*)

Suatu sistem terdiri dari sejumlah komponen yang saling berinteraksi artinya saling bekerja sama membentuk satu kesatuan. Komponen-komponen sistem atau elemen dapat berupa satu subsistem atau bagian-bagian dari sistem.

1. Batas Sistem (*boundary*)

Batas sistem merupakan daerah yang membatasi antara suatu sistem dengan sistem yang lainnya atau dengan lingkungan luar.Batas suatu sistem menunjukan ruang lingkup (*scope*) dari sistem tersebut.

1. Lingkungan Luar (*environment*)

Lingkungan luar dari suatu sistem adalah apapun diluar batas dari sistem yang mempengaruhi oprasi sistem.

1. Penghubung (*interface*)

Penghubung sistem merupakan media penghubung antara satu subsistem dengan subsistem yang lain untuk dapat berinteraksi membentuk satu kesatuan.

1. Masukan (*input*)

Masukan sistem adalah energy yang dimasukan kedalam sistem yang berupa masukan perawatan (*maintenance input*) dan sinyal masukan (*signal input*). *Maintenance input* adalah energy yang dimasukan supaya sistem tersebut dapat beroprasi. *Signal input* adalah energi yang diproses untuk didapatkan keluaran.

1. Keluaran (output)

Keluaran sistem adalah hasil dari energi yang diolah dan diklasifikasikan menjadi keluaran yang berguna sisa pembuangan.

1. Pengolah (*proces*)

Suatu sistem dapat mempunyai bagian pengolahan atau sistem itu sendiri sebagai pengolahnya. Pengolah yang akan merubah masukan menjadi keluaran.

1. Sasaran (*Objectives*) atau tujuan (*goal*)

Suatu sistem harus mempunyai sasaran, karena sasaran sangat menentukan sekali masukan yang dibutuhkan sistem dan keluaran yang akan dihasilkan sistem. Suatu sistem dikatakan berhasil bila mengenai sasaran atau tujuannya. (Jogiyanto: 2008).

## 2.1.3 Definisi Informasi

Informasi merupakan hasil pengolahan data dari satu atau berbagai sumber, yang kemudian diolah, sehingga memberikan nilai, arti dan manfaat. Proses pengelolaan ini memerlukan teknologi. Berbicara teknologi memang tidak harus selalu berkaitan dengan komputer, namun komputer sendiri merupakan salah satu bentuk teknologi. Dengan kata lain, alat tulis dan mesin ketik pun dapat dimasukkan sebagai salah satu teknologi yang digunakan selain komputer dan jaringan komputer.

Pada proses pengolahan data , untuk dapat menghasilkan informasi, juga dilakukan proses *verifikasi* secara akurat, spesifik, dan tepat waktu. Hal ini penting agar informasi dapat memberikan nilai dan pemahaman kepada pengguna. (Pratama : 2014).

## Definisi Sistem Informasi

Sistem Informasi merupakan gabungan dari empat bagian utama. Keempat bagian utama tersebut mencakup perangkat lunak (*software*), perangkat keras (*hardware*), *infrastruktur*, dan sumber daya manusia (SDM) yang terlatih. Keempat bagian utama ini saling berkaitan untuk menciptakan sebuah sistem yang dapat mengolah menjadi informasi yang bermanfaat.

Didalamnya juga termasuk proses perencanaan, kontrol, koordinasi, dan pengambilan keputusan. Sehingga, sebagai sebuah system yang mengolah data menjadi informasi yang akan disajikan dan digunakan oleh pengguna, maka sistem informasi merupakan sebuah sistem yang kompleks. Bukan hanya komputer saja yang bekerja (beserta *software* dan *hardware* didalamnya), namun juga manusia (dengan *brainware* yang dimiliki). Manusia (pengguna/aktor) dalam hal ini menggunakan seluruh ide, pemikiran, perhitungan, untuk dituangkan kedalam sistem informasi yang digunakan.

Dalam penerapannya, sebuah sistem informasi dapat berupa sebuah mainframe, sebuah *server* dari komputer biasa, maupun *hosting* di *internet* pada sebuah komputer *server*. Namun tetap saja ada kesamaan diantara ketiga penerapan berbeda ini. Kesamaannya yaitu sama-sama menggunakan sarana jaringan komputer *(intranet* maupun *internet*) untuk melakukan pemrosesan data secara bersama (terdistribusi), baik oleh beberapa pengguna maupun beberapa grup pengguna , menggunakan layanan atau fitur atau aplikasi yang disertakan. (Pratama : 2014).

**2.1.5 Konsep Dasar Sistem Informasi**

Sistem informasi memiliki komponen-komponen yang saling terintegrasi membentuk satu kesatuan dalam mencapai sasaran sistem yaitu sebagai berikut : (Wahyono : 2004).

1. Blok masukan *(Input Block)*

Blok masukan dalam sebuah sistem informasi meliputi metode-metode dan media untuk menangkap data yang akan dimasukkan, dapat berupa dokumen-dokumen dasar.

1. Blok Model *(Model Block)*

Blok model ini terdiri dari kombinasi prosedur, logika dan model matematik yang berfungsi memanipulasi data untuk keluaran tertentu.

1. Blok Keluaran *(Output Block)*

Blok keluaran berupa data-data keluaran seperti dokumen *output* dan informasi yang berkualitas.

1. Blok Teknologi (*Technology Block*)

Blok teknologi digunakan untuk menerima *input*, menjalankan model, menyimpan dan mengakses data, menghasilkan dan mengirimkan keluaran serta membantu pengendalian dari sistem secara keseluruhan.

1. Blok Basis Data (*Database Block*)

Merupakan kumpulan data yang berhubungan satu dengan lainnya, tersimpan diperangkat keras komputer dan perangkat lunak untuk memanipulasinya.

1. Blok Kendali (*Control Blok*)

Meliputi masalah pengendalian terhadap operasional sistem yang berfungsi mencegah dan menangani kesalahan atau kegagalan sistem.

* 1. **Rekam Medis dan Klinik**

**2.2.1 Pengertian Rekam Medis**

Menurut surat Keputusan Menteri Kesehatan RI No: 269/MENKES/PER/III/2008 yang dimaksud rekam medis adalah berkas yang berisi catatan dan dokumen antara lain identitas pasien, hasil pemeriksaan, pengobatan yang telah diberikan, serta tindakan dan pelayanan lain yang telah diberikan kepada pasien. Catatan merupakan tulisan-tulisan yang dibuat oleh dokter atau dokter gigi mengenai tindakan-tindakan yang dilakukan kepada pasien dalam rangka palayanan kesehatan.

Dalam pelayanan kesehatan di tempat praktek, Rumah Sakit, maupun KLINIK, dokter membuat catatan mengenai berbagai informasi yang berkaitan dengan pasien yang melakukan pemeriksaan ditempat pelayanan kesehatan tersebut ke dalam suatu berkas yang dikenal sebagai Status, Rekam Medis, Rekam Kesehatan atau Medical Record. Berkas ini merupakan suatu berkas yang memiliki arti penting bagi pasien, dokter, tenaga kesehatan serta tempat pemeriksaan tersebut, baik rumah sakit, puskesmas, ataupun KLINIK.

Rekam Medis merupakan catatan yang disimpan oleh dokter pada rumah sakit, KLINIK, puskesmas, ataupun pusat pelayanan kesehatan lainnya yang menyangkut pasien yang melakukan pengobatan ditempat yakni berupa data pasien, pemeriksaan, pengobatan, dan tindakan yang diberikan kepada pasien demi kesinambungan pelayanan, biasanya rekam medis tersebut berbentuk kartu.

**2.2.2 Pengertian Klinik**

Klinik adalah organisasi kesehatan yang bergerak dalam penyedian pelayanan kesehatan kuratif (diagnose dan pengobatan), biasanya terhadap satu macam gangguan kesehatan. (Dendy Sugono : 2008).

**2.3 Unified Modeling Language (UML)**

**2.3.1 Definisi UML**

UML adalah salah satu standar bahasa yang banyak digunakan di dunia industri untuk mendefinisikan *requiretment*, membuat analisis dan desain, sertamengambarkan arsitektur dalam pemrograman berorientasi objek. (M.Shalahuddin : 2013).

**2.3.2 Pengenalan UML**

Pada perkembangan teknologi perangkat lunak, diperlukan adanya bahasa yang digunakan untuk memodelkan perangkat lunak yang akan dibuat dan perlu adanya standarisasi agar orang di berbagai negara dapat mengerti pemodelan perangkat lunak.

Pada perkembangan teknik pemrograman berorientasi objek, muncullah sebuah standarisasi bahwa pemodelan untuk pembangunan perangkat lunak yang dibangun dengan menggunakan teknik pemrograman berorientasi objek, yaitu Unified Modeling Language (UML). UML muncul karena adanya kebutuhan membangun, dan dokumentasi dari sistem perangkat lunak. UML merupakan bahasa visual untuk pemodelan diagram dan teks-teks pendukung. UML hanya berfungsi untuk melakukan pemodelan. Jadi penggunaan UML tidak terbatas pada metodologi tertentu, meskipun pada kemyataannya UML paling banyak digunakan pada metodologi berorientasi objek. (M.Shalahuddin: 2013).

* 1. ***Use Case* Diagram**

*Use case* atau diagram *use case* merupakan pemodelan untuk kelakuan system informasi yang akan dibuat. *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih actor dengan sistem informasi yang akan dibuat. *Use case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sebuah system informasi dan siapa saja yang berhak menggunakan fungsi-fungsi itu.

Syarat penanaman pada *use case* adalah nama didefinisikan sesimpel mungkin dan dapat dipahami. Ada dua hal utama pada *use case* yaitu pendefinisian apa yang disebut *actor* dan *use case*.

1. *Aktor* merupakan orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi
2. *Use case* merupakan fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor.

Tabel 2.1 Simbol Pemodelan *Use Case Diagram*

|  |  |
| --- | --- |
| **Simbol** | **Deskripsi** |
| Use case | Fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor, biasanya dinyatakan dengan menggunakan kata kerja di awal frase nama *use case*. |
| Aktor / *actor* | Orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi biasanya dinyatakan menggunakan kata benda di awal frase nama aktor |
| Asosiasi / *association* | Komunikasi antara aktor dan *use case* yang berpartisipasi pada *use case* atau *use case* memiliki interaksi dengan aktor |
| ekstensi / *extend*  <<extend>>  ------------------->  <<extend>>  <<extend>> | Relasi *use case*  tambahan ke sebuah *use case* dimana *use case* yang ditambahkan dapat berdiri sendiri  walau tanpa *use case* tambahan itu, mirip dengan prinsip *inheritance* pada pemrograman berorientasi objek, biasanya *use cae* tambahan memiliki nama depan yang sama dengan *use case* yang ditambahkan, misal arah panah mengarah pada *use case* yang ditambahkan. Biasanya *use case* yang menjadi *extend-*nya merupakan jenis yang sama dengan *use case* yang menjadi induknya. |
| Generalisasi / *generalization* | Hubungan generalisasi dan spesialisasi (umum – khusus) antara dua buah *use case* dimana fungsi yang satu adalah fungsi yang lebih umum dari lainnya, misalnya :  Arah panah mengarah pada *use case* yang menjadi generalisasinya. |
| Menggunakan / *include / uses*  <<include>>  ------------------->  *<<*uses>> | Relasi *use case* tambahan ke sebuah *use case* di mana *use case*  yang ditambahkan memerlukan *use case* ini untuk menjalankan fungsinya. Ada dua sudut pandang yang cukup besar menegnai include di *use case*   1. *Include* berarti *use case* yang ditambahkan akan selalu dipanggil saat use case tambahan dijalankan, misal pada kasus berikut :   <<include>>   1. Include berarti *use case* yang tambahan akan selalu melakukan pengecekan apakah *use case* yang ditambahkan telah dijalankan sebelum use case tambahan dijalankan, misalnya pada kasus berikut :   <<include>>  Kedua interpretasi di atas dapat dianut salah satu atau keduanya tergantung pada pertimbangan dan interprestasi yang dibutuhkan. |

* 1. ***Activity* Diagram**

Diagram aktivitas atau *activity diagram* menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis atau menu yang ada pada perangkat lunak. Yang perlu diperhatikan disini adalah bahwa diagram aktivitas menggambarkan aktivitas sistem bukan apa yang dilakukan aktor, jadi aktivitas yang dapat dilakukan oleh sistem. (M.Shalahuddin: 2013).

Diagram aktivitas juga banyak digunakan untuk mendefinisikan hal-hal berikut :

1. Rancangan proses bisnis dimana setiap urutan aktivitas yang digambarkan merupakan proses bisnis sistem yang didefiniskan.
2. Urutan atau pengelompokan tampilan dari sistem / *user interface* dimana setiap aktivitas dianggap memiliki sebuah rancangan antarmuka tampilan.
3. Rancangan pengujian dimana setiap aktivitas dianggap memerlukan sebuah pengujian yang perlu didefinisikan kasus ujinya.
4. Rancangan menu yang ditampilkan pada perangkat lunak

Tabel 2.2 Simbol Pemodelan *Activity Diagram*

|  |  |
| --- | --- |
| **Simbol** | **Deskripsi** |
| Status awal | status awal aktifitas sistem, sebuah diagram aktivitas memiliki sebuah status awal |
| Aktivitas  aktivitas | aktivitas yang dilakukan sistem, aktivitas biasanya diawali dengan kata kerja |
| Percabangan / *decision* | asosiasi percabangan dimana jika ada pilihan aktivitas lebih dari satu |
| Penggabungan / *join* | Asosiasi penggabungan dimana lebih dari satu aktivitas digabungkan menjadi satu |
| status akhir | Status akhir yang dilkaukan sistem, sebuah diagram aktivitas memiliki sebuah status akhir |

**2.6 *Sequence* Diagram**

Diagram *sequence* menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan *message* yang dikirimkan dan diterima antar objek. Oleh karena itu untuk menggambar diagram sequence maka harus diketahui objek-objek yang terlibat dalam sebuah *use case* beserta metode-metode yang dimiliki kelas yang diinstansiasi menjadi objek. (M.Shalahuddin : 2013).

Tabel 2.3 Simbol Pemodelan *Sequence Diagram*

|  |  |
| --- | --- |
| Aktor  nama aktor  atau  nama aktor  tanpa waktu aktif | Orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang, biasanya dinyatakan menggunakan kata benda di awal *frase* nama aktor |
| Garis hidup / *lifeline* | Menyatakan kehidupan suatu objek |
| Objek  nama objek : nama kelas | Menyatakan objek yang berinteraksi pesan |
| Waktu aktif | Menyatakan objek dalam keadaan aktif dan berinteraksi, semua yang terhubung dengan waktu aktif ini adalah sebuah tahapan yang dilakukan di dalamnya, misalnya  2: cekStatusLogin()  1: login()  3: open()  Maka cekStatusLogin() dan open() dilakukan di dalam metode login()  Aktor tidak memiliki waktu aktif |
| Pesan tipe create | Menyatakan suatu objek memanggil operasi/metode yang ada pada objek lain atau dirinya sendiri.  1: nama\_metode()  Arah panah mengarah pada objek yang memiliki operasi/metode, karena hal ini memanggil operasi/metode maka operasi/metode yang dipanggil harus ada pada diagram kelas sesuai dengan kelas objek yang berinteraksi. |
| Pesan tipe send  1 : masukan | Menyatakan bahwa suatu objek mengirimkan data/masukan/informasi ke objek lainnya, arah panah mengarah pada objek yang dikirim. |
| Pesan tipe return  1 : keluaran  ---------------------- | Menyatakan bahwa suatu objek yang telah menjalankan suatu operasi atau metode menghasilkan suatu kembalian ke objek tertentu, arah panah mengarah pada objek yang menerima kembalian |
| Pesan tipe destory  <<destory>> | Menyatakan suatu objek mengakhiri hidup objek yang lain, arah panah mengarah pada objek yang diakhiri, sebaiknya jika ada create maka ada *destory.* |

**2.7 *Class* Diagram**

*Diagram class* atau *class* diagram menggambarkan struktur sistem dari sagi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. Kelas memiliki apa yang disebut atribut dan metode atau operasi.

1. Atribut merupakan variabel-variabel yang dimiliki oleh suatu kelas
2. Operasi atau metode adalah fungsi-fungsi yang dimiliki oleh suatu kelas

Diagram kelas dibuat agar pembuat program atau programmer membuat kelas-kelas sesuai rancangan di dalam diagram kelas agar antara dokumentasi perancangan dan perangkat lunak sinkron. Kelas-kelas yang ada pada struktur sistem harus dapat melakukan fungsi-fungsi sesuai dengan kebutuhan sistem sehingga programmer dapat membuat kelas-kelas di dalam program dengan perancangan diagram kelas. Susunan struktur kelas yang baik pada diagram kelas sebaiknya memiliki jenis-jenis kelas berikut :

1. Kelas main

Kelas yang memiliki fungsi awal dieksekusi ketika sistem dijalankan

1. Kelas yang menangani tampilan sistem (*view)*

Kelas yang mendefinisikan dan mengatur tampilan ke pemakai

1. Kelas yang diambil dari pendefinisian *use case* (*controller)*

Kelas yang menangani fungsi-fungsi yang harus ada diambil dari pendefinisian *use case,* kelas ini biasanya disebut dengan kelas proses yang menangani proses bisnis pada perangkat lunak.

1. Kelas yang diambil dari pendefinisian data (*model)*

Kelas yang digunakan untuk memegang atau membungkus data menjadi sebuah kesatuan yang diambil maupun akan disimpan ke basis data.

Semua tabel yang dibuat di basis data dapat dijadikan kelas, namun untuk tabel dari hasil relasi atau atribut multivalue pada ERD dapat dijadikan kelas tersendiri, dapat juga tidak asalkan pengaksesan dapat dipertangung jawabkan atau tetap ada di dalam perancangan kelas.

Dalam mendefinisikan metode yang ada di dalam kelas perlu memperhatikan apa yang disebut dengan *cohesion* dan *coupling. Cohesion* adalah ukuran seberapa dekat keterkaitan instruksi di dalam sebuah metode terkait satu nama lain sedangkan *coupling* adalah ukuran seberapa dekat keterkaitan instruksi antara metode yang satu dengan metode yang lain dalam sebuah kelas. Sebagai aturan secara umum maka sebuah metode yang dibuat harus memiliki kadar *cohesion* yang kuat dan kadar *coupling* yang lemah. (M.Shalahuddin : 2013).

Tabel 2.4 Simbol Diagram *Class*

|  |  |
| --- | --- |
| **Simbol** | **Deskripsi** |
| Kelas   |  | | --- | | **nama\_kelas** | | +atribut | | +operasi() | | Kelas pada struktur sistem |
| Antarmuka / Interface  **nama\_interface** | Sama dengan konsep *interface* dalam pemrograman berorientasi objek |
| Asosiasi / *association* | Relasi antar kelas dengan makna umum, asosiasi biasanya juga disertai dengan *multiplicty* |
| Asosiasi berarah / *directed association* | Relasi antar kelas dengan makna kelas yang satu digunakan oleh kelas yang lain, asosiasi biasanya juga disertai dengan *multiplicty* |
| Generalisasi | Relasi antar kelas dengan makna generalisasi-spesialisasi (umum khusus) |
| Kebergantungan/ *dependency* | Kebergantungan antar kelas |
| Agregrasi / *aggregation* | Relasi antar kelas dengan makna semua-bagian (*whole-part)* |

**2.8 Analisa Berorientasi Objek**

Analisa berorientasi objek (Object Oriented Analysis) adalah analisa yang mengacu pada objek-objek yang akan di teliti dan mempelajari objek-objek yang ada untuk mengetahui apakah objek tersebut dapat digunakan berulang kali atau dapat disesuiakan untuk keperluan yang baru.

Menggabarkan objek yang baru atau memodifikasi objek, yang akan di kombinasi dengan obejk-objek yang sudah ada ke dalam sebuah aplikasi bisnis computer yang bermanfaat.

**2.8.1 Perancangan Berorientasi Objek**

Perancangan berorintasi objek merupakan tahap lanjutan setelah analisa berorientasi objek.Perancangan berorintasi objek adalah suatu pendekatan yang digunakan untuk menspesifikasi kebutuhan-kebutuhan system dengan mengkolaborasikan objek-objek, atribut-atribut, dan method-method yang ada. (Jeffery L. Whitten et al : 2004-2006).

**2.8.2 Analisa Dan Perancangan Berorientasi Objek**

Analisa dan perancangan berorientasi objek adalah suatu koleksi atau kumpulan dari peralatan dan teknik untuk pengembangan sebuah system yang menggunakan teknologi objek untuk membangun suatu system serta perangkat lunaknya.Tetapi disaat yang bersamaan alat dan teknik tersetruktur tetaplah penting.Dalam mendisain database. Sebagai contoh, masih sering di rancang oleh para analis system dengan menggunakan alat bantu tersetruktur. (Jeffery L. Whitten et al : 2004-2006).

**2.9 Basis Data**

Basis Data adalah kumpulan data (elementer), yang secara logik berkaitan dalam merepresentasikan fenomena/fakta secara terstruktur dalam domain tertentu untuk mendukung aplikasi pada sistem tertentu.

Pangkalan data atau basis data ([bahasa Inggris](http://id.wikipedia.org/wiki/Bahasa_Inggris): *database*), atau sering pula dieja basisdata, adalah kumpulan [informasi](http://id.wikipedia.org/wiki/Informasi) yang disimpan di dalam [komputer](http://id.wikipedia.org/wiki/Komputer) secara sistematik sehingga dapat diperiksa menggunakan suatu [program komputer](http://id.wikipedia.org/wiki/Program_komputer) untuk memperoleh informasi dari basis data tersebut. [Perangkat lunak](http://id.wikipedia.org/wiki/Perangkat_lunak) yang digunakan untuk mengelola dan memanggil [kueri](http://id.wikipedia.org/wiki/Kueri) (*query*) basis data disebut [sistem manajemen basis data](http://id.wikipedia.org/wiki/Sistem_manajemen_basis_data) (*database management system,* DBMS). Sistem basis data dipelajari dalam [ilmu informasi](http://id.wikipedia.org/wiki/Ilmu_informasi).

Istilah "basis data" berawal dari ilmu komputer.Meskipun kemudian artinya semakin luas, memasukkan hal-hal di luar bidang elektronika, artikel ini mengenai basis data komputer.Catatan yang mirip dengan basis data sebenarnya sudah ada sebelum revolusi industri yaitu dalam bentuk buku besar, kuitansi dan kumpulan data yang berhubungan dengan bisnis.

Konsep dasar dari basis data adalah kumpulan dari catatan-catatan, atau potongan dari pengetahuan. Sebuah basis data memiliki penjelasan terstruktur dari jenis fakta yang tersimpan di dalamnya: penjelasan ini disebut [skema](http://id.wikipedia.org/w/index.php?title=Skema&action=edit&redlink=1). Skema menggambarkan obyek yang diwakili suatu basis data, dan hubungan di antara obyek tersebut. Ada banyak cara untuk mengorganisasi skema, atau memodelkan struktur basis data: ini dikenal sebagai [model basis data](http://id.wikipedia.org/w/index.php?title=Model_basis_data&action=edit&redlink=1) atau model data.

Istilah basis data mengacu pada koleksi dari data-data yang saling berhubungan, dan perangkat lunaknya seharusnya mengacu sebagai sistem manajemen basis data (database management system/DBMS).Jika konteksnya sudah jelas, banyak administrator dan programer menggunakan istilah basis data untuk kedua arti tersebut. (Bambang Hariyanto: 2004)

**2.9.1 ERD (*Entity Relationship Diagram*)**

ERD merupakan suatu model jaringan yang menggunakan susunan data yang disimpan pada sistem secara abstrak. ERD juga menggambarkan hubungan antara satu entitas yang memiliki sejumlah atibut dengan entitas yang lain dalam susatu sistem yang terintegrasi. ERD digunakan oleh perancang sistem untuk memodelkan data yang nantinya akan dikembangkan menjadi basis data (*database*). Model data ini juga akan membantu pada saat melakukan analisis dan perancangan basis data, karena model data ini akan menunjukkan bermacam-macam data yang dibutuhkan dan hubungan antar data. (Yakub : 2008)

Tabel 2.5 Simbol ERD (*Entity Relation Diagram)*

|  |  |
| --- | --- |
| **Simbol** | **Keterangan** |
|  | Entitas, yaitu kumpulan dari objek yang dapat diidentifikasikan secara unik |
|  | Relasi, yaitu hubungan yang terjadi antara satu atau lebih entitas. Jenis hubungan antara lain; satu ke satu, satu ke banyak, dan banyak ke banyak |
|  | Atribut, yaitu karakteristik dari entity atau relasi yang merupakan penjelasan detail tentang entitas |
|  | Hubungan antara entity dengan atributnya dan himpunan entitas dengan himpunan relasinya |

ERD terbagi atas tiga komponen, yaitu entitas (*entity*), atribut (*attribute*) dan relasi atau hubungan (*relation*).

1. Entitas (*Entity*)

Entitas (*entity*) menunjukkan obyek-obyek dasar yang terkait dalam sistem. Obyek dasar dapat berupa orang, benda atau hal lain yang keterangannya perlu disimpan dalam basis data. Untuk menggambarkan entitas dilakukan dengan mengikuti aturan-aturan sebagai berikut :

1. Entitas dinyatakan dengan simbol persegi panjang.
2. Nama entitas berupa kata benda tunggal.
3. Nama entitas sedapat mungkin menggunakan nama yang mudah dipahami dan menyatakan maknanya dengan jelas.
4. Atribut (*Attribute*)

Atribut sering juga disebut dengan properti (*property*), merupakan keterangan-keterangan yang terkait pada sebuah entitas yang perlu disimpan sebagai basis data. Atribut berfungsi sebagai penjelas sebuah entitas untuk menggambarkan atribut yang dilakukan dengan mengikuti aturan sebagai berikut:

1. Atribut dinyatakan dengan simbol elipps
2. Nama atribut dituliskan dalam simbol elipps
3. Nama atribut berupa kata benda tunggal
4. Nama atribut sedapat mungkin menggunakan nama yang mudah dipahami dan dapat menyatakan maknannya dengan jelas
5. Atribut dihubungkan dengan entitas yang bersesuaian dengan menggunakan garis.
6. Relasi (*Relation*)

Relasi atau hubungan adalah kejadian atau transaksi yang terjadi di antara dua entitas yang keterangannya perlu deisimpan dalam basis data. Aturan penggambarana relasi antar *entity* adalah :

1. Relasi dinyatakan dengan simbol belah ketupat
2. Nama relasi dituliskan di dalam simbol belah ketupat
3. Relasi menghubungkan dua entitas
4. Nama relasi menggunakan kata kerja aktif (diawali awalan me) tunggal
5. Nama relasi sedapat mungkin menggunakan nama yang mudah dipahami dan dapat menyakan maknanya dengan jelas.

**2.9.2 Derajat Relasi (Kardinalitas)**

Kardinalitas relasi menunjukkan maksimum entitas yang dapat berelasi dengan entitas pada himpunan entitas yang lain. Kardinalitas yang terjadi di antara dua himpunan entitas (misalkan A dan B) dapat berupa satu ke satu (*one by one*), satu ke banyak (*one to many*), banyak ke satu (*many to one*) dan banyak ke banyak (*many to many*), berikut adalah penjelasan dari kardinalitas relasi. (Fatansyah : 2012).

1. Satu ke Satu (*One to One*)

Berarti setiap entitas pada himpunan entitas A berhubungan dengan paling banyak dengan satu entitas pada himpunan entitas B, dan begitu juga sebalikanya setiap entitas pada himpunan entitas B berhubungan dengan paling banyak dengan satu entitas himpunan entitas B

A B

Entitas 1

Entitas 1

Entitas 2

Entitas 2

Entitas 3

Entitas 4

Entitas 3

Entitas 4

1. Kardinalitas Satu ke Satu (*One to One*)
2. Satu ke Banyak (*One to Many*)

Berarti setiap entitas pda himpunan entitas A dan dapat berhubungan dengan banyak entitas pada himpunan entitas B, tetapi tidak sebaliknya, dimana setiap entitas pada himpunan entitas berhubungan dengan paling banyak dengan satu entitas pada himpunan entitas A.

A B

Entitas 1

Entitas 2

Entitas 3

Entitas 2

Entitas 5

Entitas 4

Entitas 3

Entitas 1

Gambar 2.2 Kardinalitas Satu ke Banyak (One to Many)

1. Banyak ke Satu (*Many to One)*

Berarti setiap entitas pada himpunan entitas A berhubungan dengan paling banyak dengan satu entitas pada himpunan B, tetapi tidak sebaliknya, dimana setiap entitas pada himpunan entitas A berhubungan dengan paling banyak satu entitas pada himpunan entitas B.

A B

Entitas 2

Entitas 5

Entitas 4

Entitas 3

Entitas 1

Entitas 1

Entitas 3

Entitas 2

Gambar 2.3Kardinalitas Banyak ke Satu (Many to One)

1. Banyak ke Banyak (*Many to Many)*

Berarti setiap entitas pada himpunan entitas A dapat berhubungan dengan banyak entitas pada himpunan entitas B, demikian juga sebaliknya, dimana setiap entitas pada himpunan B dapat berhubungan dengan banyak entitas pada himpunan entitas

A B

Entitas 1

Entitas 2

Entitas 3

Entitas 4

Entitas 1

Entitas 2

Entitas 3

Entitas 4

Gambar 2.4 Kardinalitas Banyak ke Banyak (*Many to Many*)

**2.9.3 LRS (Logical Record Structured)**

Setelah ERD ditransformasikan ke bentuk LRS, maka hasil akhir dari proses transformasi tersebut adalah sebuah diagram yang sudah dapat menggambarkan basis data yang akan digunakan. LRS terdiri dari tipe *record*, yang berupa sebuah persegi dengan *field* yang dibutuhkan di dalamnya.LRS terdiri juga dari hubungan antara tipe *record* tersebut (Nugraha: 2011). LRS *(Logical Record Structure)* adalah representasi dari struktur *record – record* pada tabel – tabel yang terbentuk dari hasil antar himpunan entitas.

**2.9.4 Tipe Data**

MySQL memiliki cukup banyak tipe data untuk field (kolom) tabel. Tipe field (kolom) ini menentukan besar kecilnya ukuran suatu tabel. Tipe field di MySQL setidaknya terbagi menjadi beberapa kelompok, yaitu numerik, string, date and time, dan kelompok himpunan (set dan enum). Masing – masing tipe field memiliki batasan lebar dan ukurannya, berikut penjelasannya:

1. Tipe Numerik

Tipe data numerik digunakan untuk menyimpan data numerik (angka).Ciri utama data numerik adalah suatu data yang memungkinkan untuk dikenai operasi aritmatika seperti penambahan, pengurangan, perkalian, dan pembagian. Berikut ini tipe *field* (kolom) di MySQL yang termasuk ke dalam kelompok tipe numerik:

1. Tinyint

Penggunaan : digunakan untuk menyimpan data bilangan bulat positif dan negatif

Jangkauan : -128 s/d 127

Ukuran : 1 byte (8 bit)

1. Smallint

Penggunaan : digunakan untuk menyimpan data bilangan bulat positif dan negatif

Jangkauan : -32.768 s/d 32.767

Ukuran : 2 byte (16 bit)

1. Mediumint

Penggunaan : digunakan untuk menyimpan data bilangan bulat positif dan negatif

Jangkauan : -8.388.608 s/d 8.388.607

Ukuran : 3 byte (24 bit)

1. Integer

Penggunaan : digunakan untuk menyimpan data bilangan bulat positif dan negatif

Jangkauan : -2.147.483.648 s/d 2.147.483.647

Ukuran : 4 byte (32 bit)

1. Bigint

Penggunaan : digunakan untuk menyimpan data bilangan bulat positif dan negatif

Jangkauan : ± 9,22 x 1018

Ukuran : 8 byte (64 bit)

1. Float

Penggunaan : digunakan untuk menyimpan data bilangan pecahan positif dan negatif presisi tunggal

Jangkauan : -3.402823466E+38 s/d -1.175494351E-38, 0, dan 1.175494351E-38 s/d 3.402823466E+38

Ukuran : 4 byte (32 bit)

1. Double

Penggunaan : digunakan untuk menyimpan data bilangan pecahan positif dan negatif presisi ganda

Jangkauan : -1.79....E+308 s/d -2.22....E-308, 0, dan 2.22....E-308 s/d 1.79....E+308

Ukuran : 8 byte (64 bit)

1. Real: Merupakan sinonim dari *double*
2. Decimal

Penggunaan : digunakan untuk menyimpan data bilangan pecahan positif dan negatif

Jangkauan : -1.79....E+308 s/d -2.22....E-308, 0, dan 2.22....E-308 s/d 1.79....E+308

Ukuran : 8 byte (64 bit)

1. Numeric

Merupakan sinonim dari decimal

1. Tipe String

Tipe data string digunakan untuk menyimpan data string (text). Ciri utama data string adalah suatu data yang memungkinkan untuk dikenai operasi aritmatika seperti penambahan, pengurangan, perkalian, dan pembagian. Berikut ini tipe *field* (kolom) di MySQL yang termasuk ke dalam kelompok tipe *string*:

1. Char

Penggunaan : digunakan untuk menyimpan data string ukuran tetap

Jangkauan : 0 s/d 255 karakter

1. Varchar

Penggunaan : digunakan untuk menyimpan data string ukuran dinamis

Jangkauan : 0 s/d 255 karakter (versi 4.1), 0 s/d 65.535 (versi 5.0.3)

1. Tinytext

Penggunaan : digunakan untuk menyimpan data text

Jangkauan : 0 s/d 255 karakter (versi 4.1), 0 s/d 65.535 (versi 5.0.3)

1. Text

Penggunaan : digunakan untuk menyimpan data text

Jangkauan : 0 s/d 65.535 (216-1) karakter

1. Mediumtext

Penggunaan : digunakan untuk menyimpan data text

Jangkauan : 0 s/d 224-1 karakter

1. Longtext

Penggunaan : digunakan untuk menyimpan data text

Jangkauan : 0 s/d 232-1 karakter

1. Tipe Date and Time

Tipe data *date and time* digunakan untuk menyimpan data tanggal dan waktu. Berikut ini tipe *field* (kolom) di MySQL yang termasuk ke dalam kelompok tipe *date and time* :

1. Date

Penggunaan : digunakan untuk menyimpan data tanggal

Jangkauan : 1000-01-01 s/d 9999-12-31 (YYYY-MM-DD)

Ukuran : 3 byte (24 bit)

1. Time

Penggunaan : digunakan untuk menyimpan data waktu

Jangkauan : -838:59:59 s/d +838:59:59 (HH-MM-SS)

Ukuran : 3 byte (24 bit)

1. Datetime

Penggunaan : digunakan untuk menyimpan data tanggal dan waktu

Jangkauan : ‘1000-01-01 00:00:00’ s/d ‘9999-12-31 23:59:59’

Ukuran : 8 byte (64 bit)

1. Year

Penggunaan : digunakan untuk menyimpan data tahun dari gambar

Jangkauan : 1900 s/d 2155

Ukuran : 1 byte (8 bit)

1. Tipe Data yang Lain

Selain tipe data di atas, MySQL juga menyediakan tipe data yang lain. Tipe data di MySQL mungkin akan terus bertambah seiring dengan perkembanngan versi MySQL. Berikut ini beberapa tipe data tambahan MySQL:

1. Set

Penggunaan : combination (himpunan data)

Jangkauan : s/d 255 string anggota

1. Enum

Penggunaan : enumerasi (kumpulan data)

Jangkauan : s/d 65.535 string

**2.10 Software Pendukung**

**2.10.1 Netbeans**

*Netbeans* adalah sebuah IDE (*Integrated Development Environment) open source* yang sering kali diasosiasikan dengan *Java.* Akan tetapi bila diperhatikan IDE yang satu ini tidak hanya dapat digunakan sebagai IDE untuk membuat proyek-proyek *Java* saja, melainkan juga proyek seperti *web service* seperti *php, python, ruby* dan lain-lain. IDE ini tidak kalah canggihnya dengan IDE lainnya seperti *Eclipse*.

*Netbeans* merupakan salah satu proyek *open source* yang disponsori oleh *Sun Microsystem*. Proyek ini berdiri pada tahun 2000 dan telah menghasilkan 2 produk, yaitu *Netbeans IDE dan Netbeans Platform. Netbeans IDE* merupakanproduk yang digunakan untuk melakukan pemrograman baik menulis kode, meng-*compile,* mencari kesalahan dan mendistribusikan program. (Komputer : 2010).

**2.10.2 MySQL**

MySQL adalah sebuah software *database*. MySQL merupakan tipe data relasional yang artinya MySQL menyimpan datanya ddalam bentuk table-tabel yang saling berhubungan. Keuntungan menyimpan data di *database* adalah kemudahannya dalam penyimpanan dan menampilkan data karena dalam bentuk table. SQL singkatan dari *Structured Query Language.* PHP menggunakan SQL untuk berkomunikasi dengan *database* dan melakukan pengolahan data. (Edy Winarno, ST: 2014).

## Xampp

Xampp (X(Windows/Linux) Apache MySQL PHP dan Perl) merupakan paker *server web* PHP dan *database* MySQL yang paling populer di kalangan pengembang *web* dengan menggunakan PHP dan MySQL sebagai databasenya. Paket XAMPP, sesuai dengan kepanjangannya, X yang berarti Windows atau Linux, pengguna bisa memilih paket yang diinginkan untuk Windows atau Linux.

XAMPP termasuk paket *server* yang paling mudah untuk digunakan sebagai paket untuk pengembangan aplikasi *web.* XAMPP termasuk paket yang paling bagus updatenya, sehingga paling baik dipilih untuk digunakan untuk *development* atau pun untuk produksi. XAMPP dapat diperoleh dari [*http://xampp.org*](http://xampp.org)atau <http://apachefriends.org>. XAMPP memiliki paket yang bisa didownload dalam bentuk:

* 1. Installer
  2. *File* ZIP
  3. *USB*

Paket dalam bentuk USB disediakan oleh XAMPP agar pengembang dapat membawa-bawa paket ini dengan dipasang di USB agar pengembang dapat dengan mudah melakukan pengembangan dikomputer mana pun. (Sidik: 2012).

**2.11 Pengujian Sistem (Testing)**

**2.11.1 Filosopi Pengujian**

Pengujian sistem merupakan proses mengeksekusi sistem perangkat lunak untuk menentukan apakah sistem perangkat lunak tersebut cocok dengan spesifikasi sistem dan berjalan sesuai dengan lingkungan yang diinginkan. Pengujian sistem sering diasosiasikan dengan pencarian bug, ketidaksempurnaan program, kesalahan pada baris program yang menyebabkan kegagalan pada eksekusi sistem perangkat lunak. Menemukan dan menghilangkan ketidaksempurnaan program disebut dedugging, yang berbeda dengan pengujian sistem yang berfokus pada pengidentifikasian adanya ketidaksempurnaan. (Fatta: 2007).

**2.11.2 Metode Pengujian Sistem**

Beberapa test case harus dilaksanakan dengan beberapa perbedaan strategi transaksi, query, atau jalur navigasi yang mewakili penggunaan sistem yang tipikal, kritis, atau abnormal.Isu kunci pada pengembangan sistem adalah pemilihan sekelompok test case yang cocok, sekecil, dan secepat mungkin, untuk meyakinkan perilaku sistem secara detail. Pengujian harus mencakup unit testing, yang mencecek validasi dari prosedur dan fungsi – fungsi secara independen dari komponen sistem yang lain. Kemudian modul testing harus menyusul dilakukan untuk mengetahui apakah penggabungan beberapa unit dalam satu modul sudah berjalan dengan baik, termasuk eksekusi dari beberapa modul yang saling berelasi apakah sudah berjalan sesuai karakteristik sistem yang diinginkan. Berikut ini kategori tes yang bisa dilakukan. (Fatta: 2007):

1. Stub Testing

Stub testing adalah pengujian yang difokuskan pada pengujian struktur kendali sebelum semua modul dituliskan. Sistem perangkat lunak secara umum terdiri dari modul yang berelasi, baik secara hierarki maupun relasional.Pengujian ini penting untuk mengecek apakah struktur kendali sudah memetakan kinerja keseluruhan modul secara tepat. Jika ada modul yang berfungsi memanggil modul yang lain maka harus dipastikan semua modul bekerjasama dengan tepat.

1. Unit Testing

Pengujian unit digunakan untuk menguji setiap modul untuk menjamin setiap modul menjalankan fungsinya dengan baik. Ada 2 metode untuk melakukan unit testing, yaitu :

1. Black Box Testing

Pada black box testing, cara pengujian hanya dilakukan dengan menjalankan atau mengeksekusi unit atau modul, kemudian diamati apakah hasil dari unit itu sesuai dengan proses bisnis yang diinginkan. Jika ada unit yang tidak sesuai outputnya maka untuk menyelesaikannya, diteruskan pada pengujian yang kedua, yaitu white box testing.

Dengan mengaplikasikan uji coba black box, diharapkan dapat menghasilkan sekumpulan kasus uji yang memenuhi kriteria berikut (Ayuliana: 2009):

1. Kasus uji yang berkurang, jika jumlahnya lebih dari 1, maka jumlah dari uji kasus tambahan harus didesain untuk mencapai uji coba yang cukup beralasan.
2. Kasus uji yang memberitahukan sesuatu tentang keberadaan atau tidaknya suatu jenis kesalahan, daripada kesalahan yang terhubung hanya dengan suatu uji coba yang spesifik.
3. White Box Testing

White box testing adalah cara pengujian dengan melihat ke dalam modul untuk meneliti kode – kode program yang ada, dan menganalisis apakah ada kesalahan atau tidak. Jika ada modul yang menghasilkan output yang tidak sesuai dengan proses bisnis yang dilakukan, maka baris – baris program, variabel, dan parameter yang terlibat pada unit tersebut dicek satu persatu dan diperbaiki, kemudian dicompile ulang. Dalam melakukan pengujian menggunakan metode white box ada 2 hal yang perlu dilakukan, yaitu (Ayuliana: 2009) :

1. Uji Coba Basis *Path*

Merupakan teknik uji coba white box yang diusulkan Tom McCabe. Metode ini memungkinkan perancang test case mendapatkan ukuran kekompleksan logical dari perancangan prosedural dan menggunakan ukuran ini sebagai petunjuk untuk mendefinisikan basis set dari jalur pengerjaan. Test case yang didapat digunakan untuk mengerjakan basis set yang menjamin pengerjaan setiap perintah minimal satu kali selama uji coba.

1. Pengujian *Loop*

Loop merupakan kendala yang sering muncul untuk menerapkan algoritma dengan tepat. Uji coba loop merupakan teknik pengujian white box yang fokusnya pada validitas dari loop. Kelas loop yaitu loop sederhana, loop tersarang, loop terangkai, dan loop tidak terstruktur.

## 2.12 JasperReports

Laporan adalah hal yang mutlak ada pada hampir semua aplikasi. Tanpa laporan, program komputer hampir tidak bermakna apa-apa. Percuma pengolahan data dilakukan kalau tidak ada hasil berupa laporan atau report yang dihasilkan. Biasanya laporan dibuat saat *user interface* (kebutuhan *user)* dan proses pengolahan data selesai, akan dilanjutkan dengan pembuatan laporan (*report).* (Komputer M. H: 2010).

### 2.12.1 Mengenal JasperReport

*JasperReport* adalah sebuah tool yang sangat *powerful* untuk membuat laporan dalam bentuk PDF, HTML, XLS, RTF, ODT, CSV, TXT, XML. Program ini mulai dikembangkan tahun 2001 oleh Teodor Danciu, setelah pada peluncurannya pada tahun itu sampai sekarang tool ini menjadi sangat populer. Sehingga pada *search engine* dia muncul paling awal dibanding dengan yang lainnya. Kemudian pada tahun 2005 muncul perusahaan *JasperSoft* yang perkembangannya didukung oleh beberapa pengembang *software*, dan menyediakan juga beberapa aplikasi tambahan yang bersifat *commercial.*

Setiap bulannya program *JasperReport* di *download* lebih dari 20.000 kali dan sudah dikembangkan oleh lebih dari 10.000 perusahaan *software.* Dengan waktu yang singkat, program ini mampu menjadi sebuah trend dengan kemampuannya dan *fleksibilitas*nya yang sangat tinggi, anda bisa membuat berbagai laporan berbentuk *Chart, Crosstabs, Subreport* (master detail) dan kemampuan lain yang anda butuhkan. Satu lagi, yang tidak kalah pentingnya adalah sifatnya yang *open source* membuat tool ini semakin menawan untuk menjadi pilihan bagi *programmer*maupun *developer* terkini. (Komputer M. H. : 2010).

## IReport

*Report* atau laporan merupakan hasil *output* yang berisi informasi dan berasal dari pengolahan data-data yang tersimpan di dalam *database.* Untuk keperluan tersebut, *Java* telah didukung oleh *reporting library* yang terkenal, yaitu *Jasper Report,* beserta *file* pendukung *reporting* lainnya. Dengan *Jasper Report,* aplikasi *Java* yang anda buat dapat menampilkan *report statis* maupun *report dinamis* yang dapat menampilkan data yang tersimpan di dalam *database MySQL.*

Oleh karena *Jasper Report* hanya berupa *file* *library,* diperlukan *software* pendukung lain untuk melengkapinya, yaitu *IReport. IReport* meupakan *software* yang digunakan untuk membuat dan mendesain *templatereport/*laporan dengan tampilan *GUI*, sehingga tidak perlu mengetik secara manual file *XML* untuk membuat *template report* nya. IReport ini dapat berinteraksi secara dinamis dengan banyak *software database,* seperti MySQL. (Supriyatno : 2010).