BAB III

LANDASAN TEORI

3.1 Konsep Dasar Sistem Informasi

Konsep dasar dari Sistem Informasi terbagi atas dua pengertian. Yang pertama adalah sistem, dan yang kedua adalah sistem informasi itu sendiri.

3.1.1 Sistem

Definisi sistem dapat dibagi menjadi dua pendekatan, yaitu pendekatan secara prosedur dan pendekatan secara komponen. Berdasarkan pendekatan prosedur, sistem didefinisikan sebagai kumpulan dari beberapa prosedur yang mempunyai tujuan tertentu. Sedangkan berdasarkan pendekatan komponen, sistem merupakan kumpulan dari komponen-komponen yang saling berkaitan untuk mencapai tujuan tertentu.

Dalam perkembangan sistem yang ada, sistem dibedakan menjadi dua jenis, yaitu sistem terbuka dan sistem tertutup. Sistem terbuka merupakan sistem yang dihubungkan dengan arus sumber daya luar dan tidak mempunyai elemen pengendali. Sedangkan sistem tertutup tidak mempunyai elemen pengontrol dan dihubungkan pada lingkungan sekitarnya. (Herlambang, 2005:116)

3.1.2 Sistem informasi

Data adalah fakta-fakta atau kejadian-kejadian yang dapat berupa angka-angka atau kode-kode tertentu. Data masih belum mempunyai arti bagi penggunanya. Untuk dapat mempunyai arti data diolah sedemikian rupa sehingga

dapat digunakan oleh penggunanya. Hasil pengolahan data inilah yang disebut sebagai informasi. Secara ringkas, Informasi adalah data yang telah diolah dan mempunyai arti bagi penggunanya. Sehingga sistem informasi dapat didefinisikan sebagai prosedur-prosedur yang digunakan untuk mengolah data sehingga dapat digunakan oleh penggunanya. (Herlambang, 2005:121)

3.1.3 Analisa dan perancangan sistem

Analisis sistem dilakukan dengan tujuan untuk dapat mengidentifikasi dan mengevaluasi permasalahan yang terjadi dan kebutuhan yang diharapkan, sehingga dapat diusulkan perbaikannya.

Perancangan sistem merupakan penguraian suatu sistem informasi yang utuh ke dalam bagian komputerisasi yang dimaksud, mengidentifikasi dan mengevaluasi permasalahan, menentukan kriteria, menghitung konsistensi terhadap kriteria yang ada, serta mendapatkan hasil atau tujuan dari masalah tersebut serta mengimplementasikan seluruh kebutuhan operasional dalam membangun sebuah aplikasi

Analisa dan perancangan sistem dipergunakan untuk menganalisis, merancang, dan mengimplementasikan peningkatan-peningkatan fungsi bisnis yang dapat dicapai melalui penggunaan sistem informasi terkomputerisasi. Berikut ini adalah proses dalam analisis dan perancangan sistem:

1. *Entity Relationship Diagram*

   Entity Relationship Diagram (ERD) adalah gambaran pada sistem dimana di dalamnya terdapat hubungan antara *entity* beserta relasinya. *Entity* merupakan sesuatu yang ada dan terdefinisikan di dalam suatu organisasi,

dapat abstrak dan nyata. Untuk setiap *entity* biasanya mempunyai *atribute* yang merupakan ciri *entity* tersebut. Relasi adalah hubungan antar *entity* yang berfungsi sebagai hubungan yang mewujudkan pemetaan antar *entity*. (Kendall, 2003:7)

Atribute adalah kolom di sebuah relasi. Macam-macam *atribute* yaitu:

a. Simple Atribute

Atribute ini merupakan *atribute* yang unik dan tidak dimiliki oleh *atribute* lainnya, misalnya *entity* mahasiswa yang *atribute*-nya NIM.

b. Composite Atribute

Composite Atribute adalah *atribute* yan memiliki dua nilai harga, misalnya nama besar (nama keluarga) dan nama kecil (nama asli).

c. *Single Value Atribute*

Atribute yang hanya memiliki satu nilai harga, misalnya *entity* mahasiswa dengan atribute-nya Umur (tangal lahir).

d. Multi Value Atribute

Multi Value Atribute adalah *atribute* yang banyak memiliki nilai harga, misalnya *entity* mahasiswa dengan *atribute*-nya pendidikan (SD, SMP, SMA).

e. *Null Value Atribute*

Null Value Atribute adalah *atribute* yang tidak memiliki nilai harga, misalnya *entity* tukang becak dengan *atribute*-nya pendidikan (tanpa memiliki ijazah).

Entity Relationship Diagram ini diperlukan agar dapat menggambarkan hubungan antar *entity* dengan jelas, dapa menggambarkan batasan jumlah

entity dan partisipasi antar *entity*, mudah dimengerti pemakai dan mudah disajikan oleh perancang *database*. Untuk itu *Entity Relationship Diagram* dibagi menjadi dua jenis model, yaitu:

a. *Conceptual Data Model*

Conceptual Data Model (CDM) adalah jenis model data yang mengambarkan hubungan antar tabel secara konseptual.

b. *Physical Data Model*

Physical Data Model (PDM) adalah jenis model data yang mengambarkan hubungan antar tabel secara fisikal. (Marlinda, 2004 : 28)

2. Data Flow Diagram

Pada tahap ini, penggunaan notasi dapat dapat membantu komunikasi dengan pemakai sistem untuk memahami sistem tersebut secara logika. Diagram yang menggunakan notasi-notasi untuk menggambarkan arus dari data sistem ini dikenal dengan nama Diagram Arus Data (*Data Flow Diagram*). DFD berfungsi untuk mengambarkan proses aliran data yang terjadi di dalam sistem dari tingkat yang tertinggi sampai yang terendah, yang memungkinkan untuk melakukan dekomposisi, membagi sistem kedalam bagian-bagian yang lebih kecil dan yang lebih sederhana.

DFD fokus pada aliran data dari dan ke dalam sistem serta memproses data tersebut (Kendall, 2003 : 241).

Simbol-simbol dasar dalam DFD antara lain:

a. *Eksternal Entity*

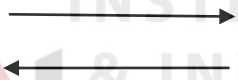
Suatu *Eksternal Entity* atau entitas merupakan orang, kelompok, departemen, atau sistem lain di luar sistem yang dibuat dapat menerima atau memberikan

informasi atau data ke dalam sistem yang dibuat. Gambar 3.1 merupakan simbol entitas dalam DFD dalam model Gene dan Sarson.

Gambar 3.1 Simbol *Eksternal Entity*

b. *Data Flow*

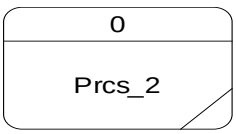
Data Flow atau aliran data disimbolkan dengan tanda panah. *Data Flow* menunjukan arus data atau aliran data yang menghubungkan dua proses atau entitas dengan proses. Gambar 3.2 merupakan simbol *Data Flow*.



Gambar 3.2 Simbol *Data Flow*

c. *Process*

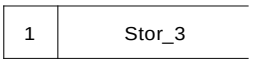
Suatu Proses dimana beberapa tindakan atau sekelompok tindakan dijalankan. Gambar 3.3 merupakan simbol *Process*.



Gambar 3.3 Simbol *Process*

d. *Data Store*

Data Store adalah simbol yang digunakan untuk melambangkan proses penyimpanan data. Gambar 3.4 merupakan simbol file penyimpanan/*Data Store*.



Gambar 3.4 Simbol *Data Store*

3.2 Konsep Dasar Basis Data

Konsep dasar dari basis data terbagi atas tiga bagian, yaitu sistem basis data, *database*, dan *database management system*.

3.2.1 Sistem basis data

Sistem basis data adalah suatu sistem menyusun dan mengelola *records* mengunakan komputer untuk menyimpan atau merekam serta memelihara dan operasional lengkap sebuah organisasi/perusahaan sehinga mampu menyediakan informasi optimal yang diperlukan pemakai untuk proses mengambil keputusan.

Pada sebuah sistem basis data terdapat komponen-komponen utama yaitu perangkat keras (*hardware*), sistem operasi (*operating system*), basis data (*database*), sistem (aplikasi atau perangkat lunak) pengelola basis data (DBMS), pemakai (*user*), dan aplikasi (perangkat lunak) lain (bersifat operasional).

Keuntungan sistem basis data adalah:

1. Mengurangi kerangkapan data, yaitu data yang sama disimpan dalam berkas data yang berbeda-beda sehingga *update* dilakukan berulang-ulang.

2. Mencegah ketidak konsistenan.
3. Keamanan data dapat terjaga, yaitu data dapat dilindungi dari pemakai yang tidak berwenang.
4. Integritas dapat dipertahankan.
5. Data dapat digunakan bersama-sama.
6. Menyediakan *recovery*.
7. Memudahkan penerapan standartisasi.
8. Data bersifat mandiri.
9. Keterpaduan data terjaga, memelihara keterpaduan data berarti data harus akurat.

Kerugian sistem basis data adalah:

1. Diperlukan tempat penyimpanan yang besar.
2. Diperlukan tenaga yang terampil dalam mengolah data.
3. Perangkat lunaknya mahal.
4. Kerusakan sistem basis data dapat mempengaruhi departemen yang terkait. (Marlinda, 2004:1)

3.2.2 *Database*

Database merupakan sekumpulan data yang berisi informasi yang saling berhubungan. Pengertian ini sangat berbeda antara *database* Relasional dan Non Relasional. Pada *database* Non Relasional, sebuah *database* hanya merupakan sebuah *file*. (Yuswanto, 2005:2)

Database adalah suatu kumpulan data operasional lengkap dari suatu organisasi perusahaan yang dikelola dan disimpan secara terintegerasi dengan

menggunakan metode tertentu menggunakan komputer sehingga mampu menyediakan informasi optimal yang diperlukan pemakainya.

Penyusunan suatu *database* digunakan untuk mengatasi masalah-masalah pada penyusunan data yaitu redundansi dan inkosistensi data, kesulitan pengaksesan data, isolasi data untuk standarisasi, *multiple user*, masalah keamanan, masalah integerasi, dan masalah *data independence*. (Marlinda, 2004:1)

3.2.3 *Database* Management System

Database Management System (DBMS) merupakan kumpulan *file* yang saling berkaitan dan program untuk pengelolanya. Basis Data adalah kumpulan datanya, sedang program pengelolanya berdiri sendiri dalam suatu paket program yang komersial untuk membaca data, menghapus data, dan melaporkan data dalam basis data.

Bahasa-bahasa yang terdapat dalam DBMS adalah:

1. *Data Definition Language*

   Pola skema basis data dispesifikasikan dengan satu set definisi yang diekspresikan dengan satu bahasa khusus yang disebut DDL. Hasil kompilasi perintah DDL adalah satu set tabel yang disimpan di dalam file khusus yang disebut *data dictionary/directory*.

2. *Data Manipulation Language*

   Bahasa yang memperbolehkan pemakai mengakses atau memanipulasi data sebagai yang diorganisasikan sebelumnya model data yang tepat.

3. *Query*

Pernyataan yang diajukan untuk mengambil informasi. Merupakan bagian DML yang digunakan untuk pengambilan informasi.

DBMS memiliki fungsi sebagai berikut:

a. Data Definition

DBMS harus dapat mengolah pendefinisian data.

b. Data Manipulation

DBMS harus dapat menangani permintaan-permintaan dari pemakai untuk mengakses data.

c. Data *Security* dan *Integrity*

DBMS dapat memeriksa *security* dan *integrity* data yang didefinisikan oleh DBA.

4. *Data Recovery dan Concurrency*

a. DBMS harus dapat menangani kegagalan-kegagalan pengaksesan basis data yang dapat disebabkan oleh kesalahan sistem, kerusakan *disk*, dan sebagainya.

b. DBMS harus dapat mengontrol pengaksesan data yang konkuren yaitu bila satu data diakses secara bersama-sama oleh lebih dari satu pemakai pada saat yang bersamaan.

5. *Data Dictionary*

DBMS harus menyediakan *data dictionary*. (Marlinda, 2004:6)

3.3 Interaksi Manusia dan Komputer

Interaksi Manusia dan Komputer (IMK) adalah sebuah disiplin ilmu yang mempelajari desain, evaluasi, implementasi dari sistem komputer interaktif untuk dipakai oleh manusia, beserta studi tentang faktor-faktor utama dalam lingkungan interaksinya.

Deskripsi lain dari IMK adalah suatu ilmu yang mempelajari perencanaan dan desain tentang cara manusia dan komputer saling bekerja sama, sehingga manusia dapat merasa puas dengan cara yang paling efektif. Dikatakan juga bahwa sebuah desain antar muka yang ideal adalah yang mampu memberikan kepuasan terhadap manusia sebagai pengguna dengan faktor kapabilitas serta keterbatasan yang terdapat dalam sistem.

Pada implementasiannya, IMK dipengaruhi berbagai macam faktor antara lain organisasi, lingkungan, kesehatan, pengguna, kenyamanan, antar muka, kendala dan produktifitas. (Rizky, 2006:4)

3.4 Microsoft *Visual Basic .NET* 2005

Visual Basic .NET 2005 adalah salah satu bahasa pemrograman yang ada di dalam *Visual Basic .NET* 2005 mulai dari tampilan kontrol, mendukung penuh OOP (*Object Oriented Programming),* tersedianya fasilitas GUI *(Graphic Universal Interface)* sampai dengan cara melakukan koneksi *database* yang lebih sempurna dari pendahulunya. Pada pemrograman *database*, *Visual Basic .NET* 2005 sangat tepat jika disandingkan dengan *Microsoft SQL Server 2005.*

Tidak berlebihan jika para pemakai program *Visual Basic* harus bermigrasi ke *Visual Basic .NET 2005,* karena beberapa alasan berikut :

1. Adanya fasilitas penanganan kesalahan (*bug*) yang *real time background compiler* sehingga *developer Visual* C# dapat mengetahui kesalahan kode secara *up-to-date.*
2. *Visual Basic .NET 2005* menyediakan model pemrograman data akses *ActiveX Data Object (*ADO*),* ditambah dengan XML baru berbasis *Microsoft ADO.NET.*
3. *Visual Basic .NET 2005* menghasilkan *Visual Basic .NET 2005* untuk *web.*
4. Mendukung pembuatan aplikasi *client-server,* terdistribusi, serta aplikasi yang berbasis *Windows* maupun *web.*
5. *Net Framework Com* memungkinkan pemakai dapat berinteraksi dengan sistem yang sudah ada, dengan menggunakan *XML web service.*
6. *Net Framework* mendukung integrasi lebih dari 20 bahasa pemrograman.
7. Penyebaran program yang mudah, baik untuk aplikasi *Windows* maupun aplikasi *web* karena sudah tersedia *wizard* secara khusus dengan fasilitas tambahan yang menarik. (Yuswanto, 2008)

3.5 *Testing* dan Implementasi Sistem

Menurut standart ANSI/IEEE 1059, *Testing* adalah proses menganalisa suatu entitas *software* untuk mendeteksi perbedaan antara kondisi yang ada dengan kondisi yang diinginkan (*defects*/*Error*/*bugs*) dan mengevaluasi fitur-fitur dari entitas *software*.

Testing software adalah proses mengopersikan *software* dalam suatu kondisi yang dikendalikan untuk:

1. Verifikasi

   Apakah telah berlaku sebagaimana yang di tetapkan (menurut spesifikasi)?

2. Mendeteksi *Error*

   Apakah telah bebas dari kesalahan *software*?

3. Validasi

   Apakah spesifikasi yang di tetapkan telah memenuhi keinginan atau kebutuhan pengguna yang sebenarnya? (Romeo, 2003:3)

Test Case merupakan tes yang dilakukan berdasarkan pada suatu inisialisasi, masukan kondisi ataupun hasil yang telah ditentukan sebelumnya. (Romeo, 2003:33). Metode *Testing* ini dibagi menjadi dua, yaitu:

1. White box *Testing*

White Box Testing atau *glass box Testing* atau *clear box Testing* adalah suatu metode *test case* yang menggunakan struktur kendali dari desain prosedural. Metode desain test case ini dapat menjamin:

1. Semua jalur (path) yang independen/terpisah dapat dites setidaknya sekali tes.

2. Semua logika keputusan dapat ditesdengan jalur yang salah atau jalur yang benar.
3. Semua *loop* dapat dites terhadap batasannya dan ikatan operasional.
4. Semua struktur internal data dapat dites untuk memastikan validasinya.

2. Black box *Testing*

Black box Testing atau *behavioral Testing* atau *specification-based Testing*, *input/output Testing* atau *functional Testing* dilakukan tanpa sepengetahuan detil struktur internal dari sistem atau komponen yang dites. *Black* box *Testing* berfokus pada kebutuhan fungsional pada *software*, berdasarkan spesifikasi kebutuhan dari *software*.

Menggunakan *black box Testing*, perekayasa *software* dapat menggunakan sekumpulan kondisi masukan yang dapat secara penuh memeriksa keseluruhan kebutuhan fungsional pada suatu program. Kategori *Error* dapat diketahui melalui *black box Testing*, antara lain:

1. Fungsi yang hilang atau tidak benar.
2. *Error* dari antar muka.
3. *Error* dari struktur data atau akses eksternal *database*.
4. *Error* dari kinerja atau tingkah laku.
5. *Error* dari inialisasi dan terminasi.