2. LANDASAN TEORI

2.1.Pengertian Sistem Informasi

Sistem adalah sekelompok elemen-elemen yang saling terintegrasi dengan maksud yang sama untuk mencapai suatu tujuan. Informasi adalah data yang telah diproses, atau data yang memiliki manfaat/kegunaan. Sistem informasi adalah kombinasi antara prosedur kerja, informasi, orang dan teknologi informasi yang diorganisasikan untuk mencapai tujuan dalam sebuah organisasi. Sebuah sistem informasi berfungsi untuk mengumpulkan, memproses, menyimpan, menganalisis, dan menyebarkan informasi untuk tujuan yang spesifik (Mc Leod Jr.,1993).

2.2. *Holt's Exponential Smoothing Method*

Holt's exponential smoothing method adalah sebuah metode yang memungkinkan untuk menyusun persamaan *trend* local pada sebuah rentetan waktu dan dapat digunakan untuk membuat sebuah peramalan (Hanke, Wichern, 2005)

Teknik Holt's ini memperhalus *level* dan *slope* secara langsung dengan menggunakan konstanta *smoothing* yang berbeda – beda. Konstanta *smoothing* inilah yang menyediakan estimasi *level* dan *slope* yang beradaptasi setiap kali pada saat pengamatan baru berlaku. Salah satu dari keuntungan dari teknik Holt's ini adalah memungkinkan fleksibilitas yang besar dalam memilih tingkatan – tingkatan ketika sebuah *level* dan *trend* terlacak.

3 persamaan yang digunakan pada metode Holt :

- Estimasi level saat ini :

$$L_t = \alpha Y_t + (1 - \alpha)(L_{t-1} + T_{t-1})$$

- Estimasi tren :

$$T_t = \beta (L_t - L_{t-1}) + (1 - \beta) T_{t-1}$$

- Meramalkan p periode kedepan :

$$\hat{Y}_{t+p} = (L_t + p\ T_t)$$

dimana,

L_t = nilai *smoothed* baru (estimasi dari *level* saat ini)

α = konstanta *smoothing* untuk *level* ($0 < \alpha < 1$)

Y_t = pengamatan baru atau nilai sesungguhnya pada serentetan waktu pada sebuah periode *t*

β = konstanta *smoothing* untuk estimasi *trend* ($0 < \beta < 1$)

T_t = estimasi *trend*

p = periode yang akan diramalkan pada masa depan

$\hat{Y}_{t+p}$ = meramalkan untuk *p* periode kedepan

Variabel *alpha-beta* yang dipakai pada fungsi peramalan untuk tiap produknya adalah variabel *alpha-beta* yang memiliki nilai *MSE* (*Minimum Squared Error*) yang terkecil. *MSE* adalah metode yang digunakan untuk mengevaluasi teknik peramalan. Setiap *error* atau *residual* dikuadratkan, kemudian dibagi oleh jumlah pengamatan.

Rumus *MSE* :

$$MSE = \frac{1}{n}\sum_{t=1}^{n} (Y_t - \hat{Y}_t)^2$$

2.3. *Data Flow Diagram (DFD)*

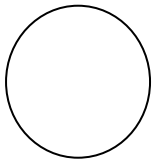
Data Flow Diagram yang nantinya akan di singkat dengan *DFD* adalah representasi dari sebuah sistem secara grafis yang digambarkan dengan sejumlah simbol tertentu untuk menunjukkan perpindahan data dalam proses-proses suatu sistem (Jogiyanto,1995).

DFD menunjukkan perpindahan dan perubahan data dalam suatu sistem dari *input* ke *output*, yang lebih dikenal dengan sebutan arus data. Meskipun demikian, penekanan pada *DFD* lebih pada prosesnya. Informasi dan perubahan dalam *DFD* ditunjukkan dengan cara hirarki dalam bentuk diagram *level*. *Context diagram* berisi entiti-entiti luar dari proses tunggal suatu sistem dengan *input* dan *output* data yang

ditunjukkan dengan arah anak panah kedalam dan keluar. Diagram yang lebih detil lagi dari sistem tersebut dapat dibentuk dengan membagi proses pada *context diagram* menjadi *DFD level* 0, 1, 2, dan seterusnya, sampai arus data yang hendak digambarkan terlihat dengan jelas.

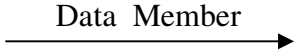
DFD menggunakan 4 macam simbol yaitu proses, arus data, simpanan data, dan kesatuan luar (*external entity*).

- Proses ialah simbol yang mengubah suatu data dari suatu bentuk menjadi bentuk yang lain. Proses menerima *input* data dan mengeluarkan *output* data lain yang telah diproses. Simbol dari proses dapat dilihat pada Gambar 2.1.



Gambar 2.1. Proses

- Arus data atau *data flow* ialah aliran data yang menunjukkan perpindahan data dari satu bagian ke bagian yang lain dalam sebuah sistem. Data *flow* dalam *DFD* disimbolkan dengan tanda panah dan diberi nama atau keterangan di sampingnya yang menunjukkan data apa yang mengalir. Simbol dari arus data dapat dilihat pada Gambar 2.2.



Gambar 2.2. Arus Data

- Simpanan data atau *data store* ialah tempat penyimpanan data dalam suatu sistem, baik secara manual maupun secara elektronik. Simpanan data digunakan jika suatu proses perlu menggunakan data tersebut lagi kemudian. Simbol dari simpanan data dalam *DFD* ada 2 macam, seperti pada Gambar 2.3.

Gambar 2.3. Simpanan Data

- Kesatuan luar (*external entity*) ialah seseorang, sekelompok orang, sebuah departemen didalam maupun diluar organisasi, atau sebuah sistem yang lain yang memberikan *input* untuk sistem yang ada atau menerima *output* dari sistem yang ada. Dalam *DFD* kesatuan luar disimbolkan dengan sebuah kotak persegi panjang seperti pada Gambar 2.4.

Gambar 2.4. *External Entity*

2.4. *Entity Relationship Diagram / ERD*

Entity Relationship Diagram adalah suatu model yang menggambarkan hubungan antar tabel dalam *database*, yang akan menjelaskan secara keseluruhan dari sistem yang ada di dalam perusahaan (Jogiyanto,1995).

Istilah –istilah dan simbol-simbol yang digunakan dalam *ERD*, antara lain:

- *Entity*

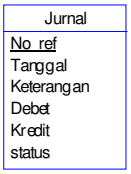
  merupakan sebuah objek yang dapat dibedakan dari objek yang lainnya. *Entity* mempunyai atribut dimana atribut tersebut dapat menjelaskan karakteristik dari *entity*. Contohnya: orang, tempat, kejadian, nama perusahaan.

  Simbol *entity* dapat dilihat pada Gambar 2.5.

Gambar 2.5. *Entity*

- *Attribute*

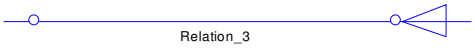
  Merupakan bagian dari suatu *entity*, yang berisi keterangan mengenai *field-field* apa saja yang dimiliki oleh suatu *entity*, dimana setiap *attribute* biasanya berada pada kotak *entity* beserta penjelasan ukuran *size field* tersebut.



Gambar 2.6. *Attribute*

- *Relationship*

  merupakan relasi atau hubungan antara *entity* yang satu dengan *entity* yang lainnya. Simbol *relationship* dapat dilihat pada Gambar 2.7.



Gambar 2.7. *Relationship*

- *Key*

  Merupakan suatu pengenal yang unik dari suatu *entity* antara satu dengan yang lainnya berbeda, biasanya ditulis dengan digaris bawahi dalam suatu *ERD*. Ada beberapa macam *key*, yaitu : *Primary key*, *Foreign key*, *Partial key*, dan *Composite key*.

Macam-macam hubungan (*relationship*) antar *entity* :

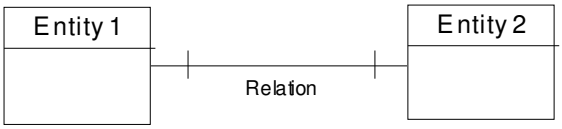
- *Zero entity*

  Merupakan *entity* yang tidak mempunyai hubungan dengan *entity* lainnya

- *One to one relationship*

  merupakan suatu hubungan dimana satu anggota *entity* mempunyai hubungan dengan satu anggota *entity* pada *entity* yang berbeda. Ada 2 macam hubungan, yaitu *obligatory* dan *non obligatory*. *Obligatory* adalah semua anggota dari suatu

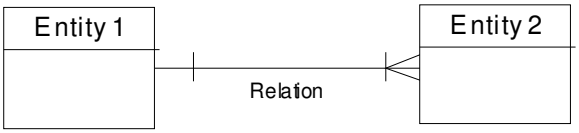
entity harus berpartisipasi atau mempunyai hubungan dengan *entity* yang lain. *Non obligatory* adalah tidak semua anggota harus mempunyai *entity* yang lain. Contoh satu orang mahasiswa hanya mempunyai satu nomor induk mahasiswa (NRP) dan satu nomor induk mahasiswa (NRP) hanya dimiliki oleh satu orang mahasiswa. Gambar *one to one relationship* dapat dilihat pada Gambar 2.8.



Gambar 2.8. *One to One Relationship*

- *One to many relationship*

  merupakan suatu hubungan antara suatu anggota e*ntity* yang satu dengan beberapa anggota e*ntity* pada e*ntity* yang berbeda. Hubungan ini juga bisa dua macam yaitu *obligatory* dan *non obligatory*. Contoh seorang dosen wali dapat menjadi wali beberapa mahasiswa tetapi seorang mahasiswa hanya dapat memiliki seorang dosen wali. Gambar *one to many relationship* dapat dilihat pada Gambar 2.9.

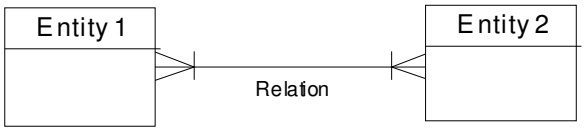


Gambar 2.9. *One to Many Relationship*

- *Many to many relationship*

  merupakan hubungan antara beberapa anggota *entity* yang satu dengan beberapa anggota *entity* pada *entity* yang lain. Jadi kedua belah pihak bisa mempunyai hubungan lebih dari satu dengan beberapa anggota *entity*. Hubungan *entity* juga bisa dua macam yaitu *obligatory* dan *non obligatory*. Contoh satu orang mahasiswa dapat mengikuti mata kuliah lebih dari satu dan mata kuliah – mata kuliah tersebut dapat diikuti oleh lebih dari satu orang

mahasiswa. Gambar *many to many relationship* dapat dilihat pada Gambar 2.10.



Gambar 2.10. *Many to Many Relationship*

2.5. *Database*

Database adalah sekumpulan data dan informasi yang terstruktur dalam suatu tabel dan saling berelasi satu sama lain, serta dapat disimpan, diproses, dimanipulasi dan digunakan oleh pihak-pihak yang berkepentingan. *Database* dipakai dari aplikasi yang sederhana sampai aplikasi yang rumit dan melibatkan beberapa *user* sekaligus.

Prinsip – prinsip dasar dalam perancangan dan pembuatan sistem *database* yang baik yaitu: efisiensi dan ketepatan, karena sebuah *database* dibuat untuk menangani kumpulan data yang bersifat kompleks supaya menjadi lebih ringkas tanpa menghilangkan karakteristik data. Hal – hal yang paling utama dan yang paling sering ditemukan di dalam sebuah *database* adalah *entry* data, pencarian data, penghapusan data yang tidak diperlukan, dan laporan-laporan data. Untuk mengurangi kompleksitas dalam suatu *database* maka harus dilakukan pengelompokan data – data yang memiliki karakteristik yang sama ke dalam satu kelas. Kemudian data yang telah dikelompokkan tersebut disimpan dalam *file*. Tujuan dari pengelompokan data yang ada ke dalam beberapa *file* adalah untuk mengantisipasi terjadinya pengulangan data, menghemat kapasitas tempat penyimpanan data, dan membangun suatu struktur *database* yang baik dan teratur sehingga memudahkan dalam proses pencarian data dan efisiensi waktu.

2.5.1. Pengertian Tabel pada *Database*

Tabel adalah alat bantu untuk mengatur atau mengelompokkan data mengenai subyek yang sama dan mengandung informasi dari kolom dan baris. Tabel- tabel

saling berhubungan dengan *database* pada saat dibutuhkan, tabel juga mempunyai tipe-tipe data dan data yang aktual. Hal-hal penting yang harus ada pada tabel antara lain:

- Nama tabel; nama ini harus unik sehingga dapat dibedakan dengan tabel lain.
- Deskripsi kolom; (kolom kadang disebut juga atribut, *field* atau data-*item*) nama kolom, *domain* kolom (menyangkut jenis data tergantung *database* yang digunakan), panjang, dan *range* yang diperbolehkan.
- *Referential Integrity Constraint*
  a. Definisi apakah kolom tersebut termasuk *primary key*.
  b. Hubungan *foreign key* pada tabel dengan *primary key* dari tabel lain.

2.6. Pengertian *MySQL*

MySQL adalah *Relational Database Management System* (*RDBMS*) yang didistribusikan secara gratis. *MySQL* merupakan *database open source* yang saat ini cukup banyak digunakan pada berbagai aplikasi. Keandalannya dalam mengolah *database* ditunjang kecepatannya dalam mengakses perintah *query* serta banyaknya fitur-fitur yang dimiliki menjadikannya sebagai *database* idola saat ini. Dengan menggunakan konsep *client-server*, kelebihan dari *MySQL* adalah cepat, kuat, serta mudah digunakan, sehingga dapat dengan mudah menyimpan, mengubah, dan mengakses data. Beberapa tipe tabel *MySQL*, antara lain:

- *ISAM*

  ISAM digunakan untuk menangani tabel non-transaksi. *ISAM* sudah diganti dengan *MyISAM* dan tidak digunakan lagi.

- *MyISAM*

  Merupakan pengembangan dari ISAM. Beberapa karakteristik dari *MyISAM*:
  - Semua nilai data disimpan dengan bit yang kecil
  - Sistem operasi mendukung file yang berukuran besar
  - Nilai *null* diperbolehkan dalam kolom *index*
  - Penanganan internal satu kolom *auto-increment* per tabel
  - Penggunaan ruang dalam *index tree* semakin ditingkatkan.

- *BDB* (*BerkeleyDB*)

  BDB menyediakan tabel untuk transaksi. Beberapa karakteristik *BDB*:

  - Dapat memiliki lebih dari 31 *index* per tabel, 16 kolom per *index*, dan ukuran maksimum *key* 1024 bit
  - Tiap tabel membutuhkan *primary key*
  - *Primary key* lebih cepat dari *index* manapun
  - Dapat melakukan penambahan baris baru di tengah *index tree*.

- *InnoDB*

  InnoDB menyediakan *MySQL* tempat penyimpanan untuk *transaction-safe*.

  Beberapa karakteristik *InnoDB*:

  - Dapat digabungkan dengan tipe tabel yang lain, bahkan dengan tabel yang memiliki *query* yang sama
  - Dirancang untuk hasil maksimum disaat memproses data dalam jumlah yang besar
  - Digunakan dalam situs *database* berskala besar yang membutuhkan performa tinggi.

- *MERGE*

  Beberapa karakteristik *MERGE*:

  - Lebih cepat
  - Melakukan pencarian dan perbaikan yang efisien
  - Tabel *MERGE* menggunakan *file descriptor* yang lebih banyak
  - Lambat dalam membaca *key*.

- *MEMORY (HEAP)*

  Beberapa karakteristik *MEMORY*:

  - Ruang untuk tabel memory dialokasikan di dalam blok yang kecil
  - Dapat lebih dari 32 *index* per tabel, 16 kolom per *index*
  - Di-*shared* diantara semua klien
  - Membutuhkan *memory* tambahan yang cukup untuk menjaga semua tabel *MEMORY* yang digunakan pada saat yang sama.

- *FEDERATED*
    - Tidak dapat mendukung transaksi
    - Tidak mengenali beberapa perintah seperti *ALTER TABLE* dan *DROP TABLE*
    - Tidak dapat digunakan dengan *query cache*
    - Tidak dapat mengetahui bila tabel *remote* sudah diubah.

Untuk pembuatan Tugas Akhir ini, digunakan tipe tabel *InnoDB* karena tipe tabel ini mampu memproses data dalam jumlah yang besar dengan performa yang tinggi.

2.7. Pengertian *PHP*

PHP adalah singkatan dari (*Hypertext Preprocessor*), sebuah bahasa pemrograman yang lebih menitik beratkan pada Applikasi *Web*. *PHP* adalah sebuah bahasa pemrograman skrip di sisi *server*, yang berarti semua skrip dijalankan di *server* dan ditampilkan di *browser client*. Kelebihan *PHP* dari bahasa pemrograman lain :

* Bahasa pemrograman *php* adalah sebuah bahasa *script* yang tidak melakukan sebuah kompilasi dalam penggunaannya.

* *Web Server* yang mendukung *php* dapat ditemukan dimana - mana dari mulai *IIS* sampai dengan *apache*, dengan konfigurasi yang relatif mudah.

* Dalam sisi pengembangan lebih mudah, karena banyaknya milis - milis dan *developer* yang siap membantu dalam pengembangan.

* Dalam sisi pemahamanan, *php* adalah bahasa *scripting* yang paling mudah karena referensi yang banyak.

2.8.Pengertian *Javascript*

Javascript adalah bahasa pemrograman yang berbentuk kumpulan skrip yang pada fungsinya berjalan pada suatu dokumen *web*, sepanjang sejarah *internet*, bahasa ini adalah bahasa *script* pertama untuk *web* (Dr. Ir. Onno W. Purbo,1997). Bahasa ini adalah bahasa pemrograman untuk memberikan kemampuan tambahan terhadap bahasa *HTML* dengan mengijinkan pengeksekusian perintah-perintah di sisi

user/client, yang artinya di sisi *browser client* bukan di sisi *server web*. *Javascript* bergantung kepada *browser (navigator)* yang memanggil halaman *web* yang berisi skrip-skrip dari *Javascript* dan tentu saja terselip di dalam dokumen *web. Javascript* juga tidak memerlukan kompilator atau penterjemah khusus untuk menjalankannya (pada kenyataannya kompilator *Javascript* sendiri sudah termasuk di dalam *browser* tersebut).

2.9. Pengertian *Apache Web Server*

Apache Web server adalah sebuah perangkat lunak *server web* yang dapat dijalankan pada banyak sistem operasi (*Unix, BSD, Linux, Microsoft Windows* dan *Novell Netware* serta *platform* lainnya) yang berguna untuk melayani dan memfungsikan situs *web*. *Apache* juga didukung oleh sejumlah antarmuka pengguna berbasis grafik yang memungkinkan penanganan *server* menjadi mudah. *Apache* merupakan perangkat lunak sumber terbuka dikembangkan oleh komunitas terbuka yang terdiri dari pengembang-pengembang di bawah naungan *Apache Software Foundation*.