2. TEORI DASAR

Dalam pembuatan sistem informasi administrasi diperlukan teori-teori penunjang tentang *database*, konsep perancangan *database*, dan pengenalan terhadap bahasa pemrograman yang digunakan, untuk itu pada bab ini akan dijelaskan tentang hal-hal tersebut.

2.1. *Database*

Istilah *database* secara umum dapat diartikan sebagai suatu kumpulan data yang saling berhubungan. Istilah data bisa dipakai baik untuk tunggal maupun jamak, karena sudah umum dalam dunia *database.* Dalam bahasa Inggris sehari-hari, data hanya dipakai untuk jamak, *datum* untuk tunggal.

Database mempunyai beberapa karakteristik antara lain adalah sebagai berikut:

a. *Database* dipakai untuk mempresentasikan aspek-aspek dari dunia nyata.
b. *Database* memiliki sekumpulan data yang teratur dan memiliki arti jelas, data sembarang tidak boleh disebut *database*.
c. *Database* didesain, dibuat dan diisi dengan data untuk suatu tujuan tertentu dan pemakai tertentu.

Suatu *database* bisa berskala kecil maupun sangat besar dengan kompleksitas yang bervariasi.

2.1.1. *Database Management System* (DBMS)

Database Management System (DBMS) adalah suatu kumpulan *database* yang saling berelasi satu sama lain dan suatu bentuk program untuk mengakses *database* tersebut. DBMS terdiri dari beberapa *database* dan satu atau lebih program pengelola yang berguna pengolahan data yang ada seperti untuk menambah data, menghapus data, mengambil data dan memperbarui data-data yang ada.

Database sendiri merupakan sekelompok kumpulan data yang saling berelasi, yang mana dapat disimpan, diproses, dimanipulasi dan digunakan oleh pihak-pihak yang berkepentingan. Relasi tersebut biasanya ditujukan dengan bagian-bagian penting dari tiap-tiap *file* (kumpulan data) yang ada. Dalam setiap perancangan dan pembuatan suatu sistem *database* yang baik diperlukan suatu prinsip mendasar yang sangat penting, yaitu: efisiensi dan ketepatan. Suatu *database* dibuat untuk menangani kumpulan data yang bersifat kompleks supaya menjadi lebih ringkas tanpa menghilangkan karakteristik data.

Karakter suatu *Database Management System* adalah:

a. Mampu untuk mengatur atau mengurus data dalam jumlah yang sangat banyak.
b. Memiliki fasilitas untuk mengakses data yang sangat banyak tersebut secara efisien.
c. Memiliki fasilitas untuk mengakses data yang sangat banyak tersebut secara *concurrent.*
d. Memiliki fasilitas untuk mengakses data yang sangat banyak tersebut secara aman.

2.1.2. Desain *Database*

Proses desain dari *database* adalah bagian penting dalam suatu pengembangan sistem informasi. Dalam suatu sistem informasi, *database* adalah komponen paling berpengaruh pada unjuk kerja (*performance*) dari sistem. Tahap-tahap pengembangan suatu sistem :

a. *Problem definition*
b. *Feasibility study*
c. *Analysis*
d. *Design*
e. *Implementation*
f. *Maintenance*

Beberapa istilah penting yang terkait dengan pembuatan *database* antara lain:

a. *Table*

Table adalah *file* yang digunakan untuk menyimpan data yang berhubungan. Suatu *table* memiliki beberapa *record* (baris), dimana tiap *record* dibagi menjadi beberapa *field* (kolom). Tiap baris memiliki jumlah kolom yang sama.

b. *Record*

Record dikenal juga dengan istilah *row* atau *tuple*. Suatu *record* berisi informasi tentang *item* tertentu.

c. *Field*

Field dikenal juga dengan istilah *column* atau *attributes*. Suatu *field* berisi sebagian informasi dari suatu *record*.

d. *Domain*

Domain dipakai untuk mempresentasikan isi dari *field* yang diijinkan. Misal: *field* jenis kelamin dalam *table* memiliki dua *domain* yaitu laki-laki dan perempuan. Setiap *field* dalam suatu *table* pasti memiliki *domain*. Banyak *field* yang bisa memiliki *domain* yang sama, sistem *database* bisa mendeteksi pengisian data yang salah pada saat mencari atau mengisi data.

e. *Primary Key*

Primary Key dipakai untuk mengidentifikasikan suatu *record* ke dalam suatu *table*.

f. *Foreign keys*

Foreign keys adalah jika suatu *primary key* dari suatu *table* dipakai sebagai *non-primary key field* dari *table* lain.

2.1.3. Konsep *Database*

Database adalah kumpulan dari komputer data yang diintegrasikan, diorganisir dan disimpan dengan cara yang mudah untuk diakses. Untuk ini digunakan *direct access storage*. Integrasi secara *logical* dari *record* di beberapa *file* disebut konsep *database*. Untuk mengenal *database* secara lebih detail, perlu diketahui komponen-komponen dasar pembentuk *database*, yaitu :

1. *Entity*

Entity adalah suatu bentuk dari orang, tempat, kejadian atau konsep dari suatu informasi yang dijadikan suatu data bagi basis data yang digunakan.

2. *Atribut*

   Atribut adalah sebutan untuk mewakili suatu *entity*.

3. *Data value*

   Data value adalah *data actual* atau informasi yang disimpan dalam tiap-tiap data elemen atau *atribut*.

4. *Record/tuple*

   Record/tuple adalah kumpulan elemen-elemen yang saling berkaitan yang berguna untuk menginformasikan tentang suatu *Entity* secara lengkap.

5. *File*

   File adalah sekumpulan *record-record* sejenis yang mempunyai panjang elemen sama, *atribut* yang sama namun berbeda data *value*-nya.

2.2. Perancangan *Database*

Dalam perancangan *database* dikenal dua macam cara, diantaranya adalah sebagai berikut :

a. Teknik Normalisasi

   Cara ini dimulai dari dokumen dasar yang sudah ada pada sistem atau sudah dipakai sistem tersebut, data-data pada dokumen dasar tersebut dipisah-pisah menjadi *file-file* yang tiap *field* pada *file* tersebut bergantung penuh pada kunci utama (*field* kunci) yang biasanya dikenal dengan bentuk normal ketiga. Kemudian setiap *file* dalam *database* tersebut ditentukan hubungannya dengan *file-file* yang lain dengan cara memasang *field* tamu pada *file-file* anak atau *file* konektor.

   - *First Normal Form (1NF)*

     Menghapus pengulangan elemen yang ada pada entitas. Pengulangan ini akan menimbulkan sebuah entitas baru dengan *composite key*. *Composite key* merupakan atribut dengan banyak entitas.

   - *Second Normal Form (2NF)*

     Memastikan bahwa atribut pendeskripsian bergantung pada seluruh *composite key* untuk identifikasi.

- *Third Normal Form (3NF)*

  Memastikan bahwa nilai atribut tidak sama dengan nilai atribut yang lain dalam satu entitas.

b. Teknik *Entity Relationship*

Langkah ini sering digunakan pada perancangan sistem, dimulai dengan pembuatan diagram arus data yang menghasilkan kamus data yang merupakan daftar semua elemen/*field* yang dibutuhkan dalam sistem terebut. Dari *field-field* tersebut dipilih *field* kunci yang bersifat unik artinya keseluruhan *record* dapat dicari dari *record* tersebut, kemudian baru dibuat *file-file* berdasar kunci *record* tersebut yang mana elemen/*field* dalam *field* tersebut bergantung penuh dengan *field* kunci tersebut. Setelah membuat tabel baru ditentukan relasi dari tiap tabel tersebut seperti halnya teknik normalisasi.

Adapun desain dari table *database* yang dibuat nantinya akan memenuhi teori normalisasi.

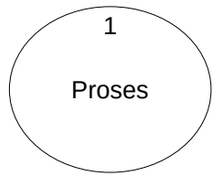
2.2.1. *Data Flow Diagram* (DFD)

Data Flow Diagram atau yang dapat disingkat dengan DFD adalah representasi dari sebuah sistem secara grafis yang digambarkan dengan sejumlah simbol tertentu untuk menunjukkan perpindahan data dalam proses-proses suatu sistem.

Dalam hal ini DFD menunjukkan perpindahan dan perubahan data dalam suatu sistem. Meskipun diberi nama DFD, namun penekanan pada DFD lebih pada prosesnya, bahkan DFD merupakan salah satu alat pemodelan proses dari sistem yang paling sering digunakan.

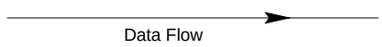
Simbol-simbol yang digunakan dalam DFD hanya terdiri dari empat macam yaitu proses, arus data, simpanan data, dan kesatuan luar (*eksternal entity*). Berikut uraian singkat mengenai empat macam simbol yang ada, yaitu:

a. **Proses** adalah simbol yang mengubah suatu data dari suatu bentuk menjadi bentuk yang lain. Atau dengan kata lain, proses menerima *input* data dan mengeluarkan *output* data lain yang telah diproses. Simbol dari proses dalam DFD ada beberapa model dan dapat dilihat pada gambar 2.1.



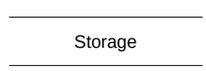
Gambar 2.1. Proses

b. **Arus data** atau *data flow* adalah aliran 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 disampingnya yang menunjukkan data apa yang mengalir. Gambar arus data dapat dilihat pada gambar 2.2.



Gambar 2.2. Arus Data

c. **Simpanan data** adalah 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 juga beberapa model seperti pada gambar 2.3.



Gambar 2.3. Simpanan Data

d. **Kesatuan luar** (*external entity*) adalah seseorang, sekelompok orang, sebuah departemen di dalam maupun di luar organisasi atau sebuah sistem yang lain yang memberikan *input* untuk sistem yang ada atau menerima *output* dari sistem yang ada. *External entity* juga disebut *terminator*, karena merupakan batas dari sebuah sistem. Dalam DFD, external entity memiliki beberapa macam simbol seperti pada gambar 2.4.

Gambar 2.4. *External Entity*

Context diagram adalah DFD yang menunjukkan batas-batas dari sebuah sistem informasi, yaitu *top level view* dari sebuah sistem. Untuk menggambar *Context diagram*, hanya dibuat satu proses saja yang menggambarkan keseluruhan proses dari sistem tersebut dan beberapa kesatuan luar di sekelilingnya yang berhubungan. Contoh *context diagram* untuk DFD di atas adalah sebagai berikut :

a. Dari DFD *level* 0, kadang-kadang ada proses yang dapat dijabarkan lebih *detail* lagi
b. Dari DFD *level* 0 dapat dibuat sejumlah DFD *level* 1, demikian pula dari setiap *level* 1 tersebut jika masih dapat dijabarkan lagi menjadi DFD *level* 2, 3 dan seterusnya. Sebuah DFD minimal digambarkan sampai *level* 0 saja, namun dalam kenyataan sebuah DFD umumnya pasti terdiri dari beberapa *level*.

Hal hal yang perlu diperhatikan dalam penyusunan DFD adalah sebagai berikut:

a. Dalam DFD tidak dibedakan antara data dan informasi, semua dianggap data.
b. Nama proses dalam *context diagram* harus sama dengan nama sistem tersebut.
c. Setiap DFD harus muat dalam satu halaman.
d. Setiap simbol harus diberi nama yang unik, namun harus konstan atau tidak boleh berubah jika berpindah ke *level* berikutnya.
e. Hindari garis yang berpotongan jika mungkin.
f. DFD pasti selalu mempunyai *output*.
g. Setiap proses dalam DFD harus mempunyai *input* dan *output*.
h. Suatu kesatuan luar hanya boleh mempunyai *input* atau *output*, tapi tidak boleh kedua-duanya.
i. Setiap simpanan data hanya boleh menerima *input* dari proses dan juga memberikan *output* ke proses saja.

2.2.2. *Entity Relationship Diagram* (ERD)

Metode ini dipakai pada jumlah atribut yang banyak. Metode ini menggambarkan hubungan antar tabel yang ada dalam suatu *database*. Pertama kali yang dilakukan adalah membagi *database* menjadi beberapa tabel yang disebut *object* atau *entity*. Masing-masing *entity* ini mempunyai beberapa atribut, yang merupakan bagian dari *entity* tersebut. Antar beberapa *entity* tersebut dapat digambarkan suatu hubungan (*relationship*) yang biasanya dipakai dalam ERD. ERD digunakan untuk menggambarkan hubungan antar *Entity*. Untuk menghubungkan satu *entity* dengan *entity* yang lain digunakan *entity key*, yaitu suatu atribut tertentu atau sekelompok atribut tertentu yang bersifat unik, sehingga dapat digunakan untuk membedakan suatu anggota *entity* dengan anggota yang lain pada e*ntity* yang sama. Selain itu harus ada pula yang disebut *relationship key*, yaitu suatu *relationship* yang digunakan untuk menyatakan hubungan antar *entity key*.

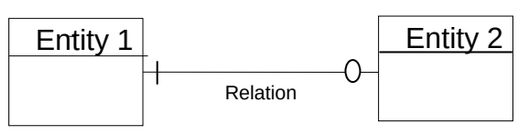
ERD ini menggambarkan relasi atau hubungan antar entitas yang ada, dimana pada setiap macam hubungan terdapat pembagian 2 jenis hubungan, yaitu:

a. *Obligatory*

   Bila semua anggota dari suatu *entity* harus berpartisipasi atau memiliki hubungan dengan *entity* yang lain. *Foreign key* di suatu tabel yang diambil dari tabel lain harus diberi nilai atau tidak boleh *null* (kosong). Untuk lebih jelas maka dapat dilihat pada gambar 2.5.

b. *Non obligatory*

   Bila tidak semua anggota dari suatu *entity* harus berpartisipasi atau memiliki hubungan dengan *entity* yang lain. *Foreign key* di suatu tabel yang diambil dari tabel yang lain boleh tidak diberi nilai atau *null* (kosong). Untuk lebih jelas maka dapat dilihat pada gambar 2.5.



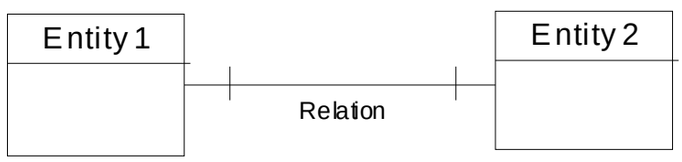
Gambar 2.5. *Non Obligatory* dan *Obligatory*

Entity 1 merupakan *non obligatory* yang tidak semua anggota *entity* ini berpartisipasi dalam *entity* 2 dan *entity* 2 merupakan *obligatory* semua anggota *entity* harus ikut berpartisipasi pada *entity* 1.

Ada beberapa macam hubungan antar *entity* diantaranya adalah sebagai berikut:

a. One to one relationship

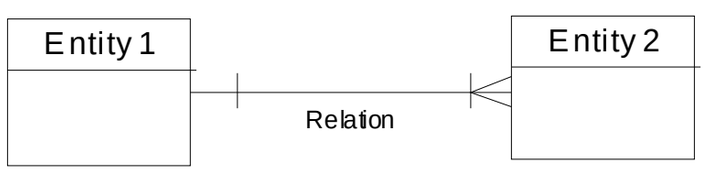
Merupakan suatu hubungan dimana satu anggota *entity* mempunyai hubungan dengan satu anggota *entity* pada *entity* yang berbeda. Pada *one to one relationship* juga bisa untuk dua jenis hubungan, yaitu *obligatory* dan *non obligatory*. Contoh, satu pelanggan hanya mempunyai satu nomor kode pelanggan dan satu nomor kode pelanggan hanya dimiliki oleh satu pelanggan.



Gambar 2.6. *One to One Relationship*

b. One to many relationship

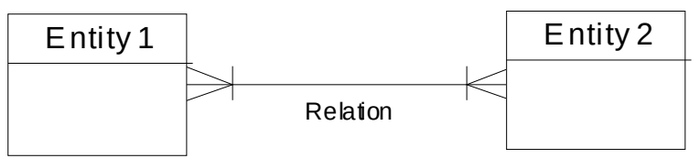
Merupakan suatu hubungan antara suatu anggota e*ntity* yang satu dengan beberapa anggota e*ntity* pada e*ntity* yang berbeda. Hubungan ini juga bisa dua jenis yaitu untuk *obligatory* dan *non obligatory*. Contoh sebuah perusahaan dapat mempunyai beberapa nomor telepon tetapi setiap nomor telepon hanya dapat dipunyai oleh sebuah perusahaan.



Gambar 2.7. *One to Many Relationship*

c. 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 *many to many relationship* ini juga bisa untuk dua jenis yaitu *obligatory* dan *non obligatory*. Contoh seseorang mempunyai perusahaan yang lebih dari satu dan perusahaan dapat dimiliki oleh lebih dari satu orang.



Gambar 2.8. *Many to Many Relationship*

2.3. *Database* Firebird

Firebird adalah *database server* yang dikembangkan secara *open source* oleh sebuah tim yang terdiri dari para *developer* sukarela. Pengembangan ini berdasarkan dari *software open source* yang di keluarkan oleh Borland Corp. yaitu InterBase Public License V.1.0. Firebird pertama kali diluncurkan pada tanggal 25 Juli 2000.

Pengembangan dari *source code* Firebird yang kedua dimulai pada awal tahap pengembangan Firebird yang pertama. Firebird 1.5 adalah *release* pertama yang dihasilkan pada saat pengembangan Firebird yang kedua. Walaupun hal tersebut merupakan suatu tonggak sejarah yang penting bagi para *developer* dan bagi keseluruhan proyek Firebird, namun ini bukanlah akhir dari proyek Firebird. Ketika Firebird 1.5. diluncurkan, pengembangan proyek ini terus berjalan menuju pada tahap berikutnya dimana Firebird yang kedua diluncurkan.

Firebird 1.0.x terus berkembang secara aktif dengan perbaikan-perbaikan dan peningkatan-peningkatan yang secara tidak langsung diambil dari basis Firebird versi 1.5.

2.4. Borland Delphi 6.0

Delphi merupakan program aplikasi berbasis *Windows*. Delphi memiliki beberapa keunggulan, antara lain mendukung penggunaan bahasa *SQL* yang merupakan bahasa *query* yang handal.

Borland Delphi 6.0 memiliki kelebihan jika dibanding dengan program aplikasi lainnya. Dengan menggunakan *Borland Delphi* 6.0, maka telah ditemukan cara yang paling sederhana untuk membangun aplikasi pada 2 basis sistem operasi yang berbeda yaitu berbasis *Windows* dan *linux*. Produktifitas dari pengembangan perangkat lunak dibagi menjadi lima *atribute* penting, yaitu:

a. Kualitas dari lingkungan pengembangan *visual*.
b. Kecepatan *compiler* dibandingkan dengan kompleksitasnya.
c. Kekuatan dari bahasa pemrograman dibandingkan dengan kompleksitasnya.
d. Fleksibilitas dari arsitektur basis data.
e. Pola desain dan pemakaian yang diwujudkan oleh *framework*-nya.

Komponen-komponen yang umum digunakan dalam pembuatan aplikasi *database*, antara lain:

a. Data Module

   Data module merupakan sebuah kelas yang pada umumnya digunakan untuk menampung komponen-komponen *non visual*, seperti komponen-komponen yang berhubungan dengan *database*.

b. Komponen Label

   Komponen label digunakan untuk menjelaskan objek lain dalam sebuah aplikasi, label juga dapat membantu navigasi dan menyediakan informasi tambahan bagi pemakai.

c. Komponen Edit

   Komponen ini dipakai untuk menerima *input* dari pemakai atau memberikan *output* bagi pemakai.

d. Komponen Combo Box

   Berfungsi untuk menampilkan daftar pilihan. Pilihan yang ditampilkan adalah nilai *default* atau nilai yang telah dipilih oleh pemakai.

e. Komponen BitBtn

Komponen ini serupa dengan komponen button, hanya saja pada komponen ini bisa diletakkan suatu gambar.

f. Komponen DBGrid

Komponen DBGrid digunakan untuk menampilkan data dalam semua *field* untuk sebuah tabel. Tetapi juga dapat diatur *field-field* mana saja yang akan ditampilkan dengan menggunakan fasilitas *fields editor*.

2.5. Koneksi Antara Borland Firebird dan Borland Delphi 6.0

Dalam pembuatan program *database* pada Borland Delphi 6.0, dapat digunakan beberapa program aplikasi sebagai penyedia fungsi *database*nya, diantaranya adalah Firebird, Oracle, Paradox, dBase, Interbase dan masih banyak lagi lainnya.

Borland Delphi 6.0 digunakan sebagai aplikasi penyedia layanan *interface*-nya (penghubung antara *user* dan *database*), sedangkan penyedia layanan aplikasi *database*nya digunakan Firebird untuk menyimpan *database*.

Komponen-komponen utama yang akan digunakan antara lain sebagai berikut:

a. Komponen-komponen Konektivitas

Komponen TSQLConnection dipakai untuk membangun hubungan dengan sebuah *data store* DBExpress. Dapat digunakan untuk menghubungkan beberapa komponen *dataset* dan perintah DBExpress ke dalam sebuah komponen TSQLConnection untuk berbagai koneksi dalam menjalankan perintah.

b. Komponen Akses DBExpress

Terdiri atas TSQLClientDataSet. Komponen TSQLClientDataSet adalah komponen utama untuk mencari dan beroperasi pada data DBExpress. Komponen ini mempunyai kemampuan untuk memanipulasi tabel-tabel dan menjalankan *query-query* SQL, dan dapat langsung berhubungan dengan sebuah data store atau koneksi melalui komponen TSQLConnection. TSQLClientDataSet mempunyai fungsi-fungsi dari TSQLTable, TSQLQuery, dan TSQLStoredProc.

c. Komponen Kompatibilitas

TSQLTable, TSQLQuery, dan TSQLStoredProc dianggap sebagai komponen-komponen kompatibilitas dengan komponen Table, Query, dan StoredProc.