BAB II

TINJAUAN PUSTAKA DAN LANDASAN TEORI

2.1 Tinjauan Pustaka

Tinjauan pustaka dilakukan berdasarkan pada penelitian terdahulu, yang dari itu dapat dipaparkan sebagai berikut :

Leon Andretti Abdillah (2006) dengan judul "Perancangan Basis Data Sistem Informasi Penggajian (Studi Kasus pada Universitas 'XYZ')". Peneliti ini merancang *database* yang berfungsi untuk Membantu bagian pengembangan perangkat lunak dalam membuat Sistem Informasi Penggajian. Tujuan dilakukannya penelitian adalah untuk membuat skema perancangan *database* (basis data) untuk Sistem Informasi Penggajian pada Universitas 'XYZ'. Metode perancangan basis data yang digunakan adalah *conceptual scheme design* yang merupakan perancangan *Entity Relationship Diagram* (ERD) , *logical design* yang merupakan perancangan *Relational database* dan *phisycal design* yang merupakan perancangan struktur struktur penyimpanan pada *file-file database*.

Efri Darwis (2012) dengan judul "Implementasi Basis Data Terdistribusi Menggunakan *MySQL* Pada PT Thamrin Brothers Palembang". Peneliti merancang dan mengimplementasi *database* yang berfungsi untuk membantu perusahaan PT Thamrin Brothers Palembang dalam menciptakan basis data yang mampu menutupi kekurangan dari sistem yang saat ini digunakan. Tujuan dilakukannya penelitian ini adalah untuk merancang dan mengimplementasikan

basis data terdistribusi stok dan penjualan pada PT Thamrin Brothers Palembang. Metode perancangan basis data yang digunakan adalah terdistribusi. Faktor-faktor yang terdapat dalam basis data relasional terdistribusi yang harus diperhitungkan dalam perancangan basis data adalah fragmentasi, relasi dibagi kedalam sejumlah sub relasi diantaranya adalah, fragmentasi *horizontal* yaitu relasi menjadi fragmentasi-fragmentasi berupa atribut-atribut tupel dari suatu relasi, fragmentasi *vertical* yaitu relasi menjadi fragmentasi-fragmentasi berupa atribut-atribut dari relasi itu.

Ita Rosita Wati, et al. (2013) dengan judul 'Rancangan dan Implementasi Basis Data Perpustakaan (Studi Kasus SMK Panggali Nusantara Palembang)". Peneliti ini merancang dan mengimplementasi *database* pada perpustakaan SMK Panggali Nusantara yang berfungsi untuk mengelola data pada perpustakaan agar data tersebut dapat di kelola dengan baik. Metode basis data yang digunakan adalah *Database Lifecycle* (*DBLC*). *Database* ini terdiri dari beberapa tahap yaitu, *Database planning* (Perencanaan Basis Data) untuk mempelajari dan melakukan penelitian agar dapat mengidentifikasi rencana pada sistem, *System definition* (definisi sistem) merupakan alur dari sistem yang mencakup data-data dimana definisi sistem yang ada saling berkaitan dengan data, *Requirement Collection and Analysis* (Pengumpulan dan Analisa Data) untuk mengumpulkan data-data yang diperlukan dalam proses pembuatan basis data yatu dengan melakukan analisis terhadap sistem yang telah berjalan, Perancangan *Database* Konseptual merupakan suatu bentukan model yang berasal dari informasi yang digunakan dalam suatu perusahaan yang bersifat independen kemudian model data tersebut dibangun

dengan menggunakan informasi dalam spesifikasi kebutuhan *user*, Perancangan *database* Logikal adalah proses pembentukan model namun independen terhadap DBMS tertentu dan aspek fisik relasional dan konseptual yang menjelaskan tentang proses untuk menghasilkan gambaran dari implementasi *database* pada tempat penyimpanan berdasarkan relasi dan indeks, Perancangan *database* Fisikal merupakan tahapan untuk mengimplementasikan hasil perancangan *database* secara logis menjadi tersimpan secara fisik pada media proses yang menghasilkan deskripsi implementasi *database* pada penyimpanan sekunder juga dapat dikatakan desain fisikal merupakan cara pembuatan menuju DBMS.

Dari ketiga penelitian di atas tahapan yang digunakan dalam perancangan *database* adalah *Database Planning* (Perencanaan Basis data), *System definition* (definisi sistem), *Requirement Collection and Analysis* (Pengumpulan dan Analisa Data), Perancangan *Database* Konseptual, Perancangan *Database* Logikal, Perancangan *Database* Fisikal. Sedangkan dalam penelitian ini tahapan yang digunakan dalam perancangan *database* adalah *Requirement Collection and Analysis* (Pengumpulan dan Analisa Data), Perancangan *Database* Konseptual (Perancangan *Entity Relationship Diagram*), Perancangan *Database* Logikal (Perancangan *Relational Database*), Perancangan *Database* Fisikal (struktur struktur penyimpanan pada *file-file database*). Selain meracang *Database* penelitian ini juga melakukan pengujian anomali agar dapat mengetahui adanya anomali data atau tidak dan pengujian *view check* sebagai validasi data. Dengan dilakukan pengujian sehingga dapat diperoleh *database* dengan data yang bagus dan efisien.

2.2 Landasan Teori

2.2.1 *Database*

Menurut (Gordon C. Everest, 1986) *Database* merupakan sebuah koleksi atau kumpulan dari data yang bersifat mekanis, terbagi, terdefinisi secara formal serta terkontrol. Basis Data pertama kali dikembangan dalam bentuk *flat file*. *Flat File* merupakan data atau *record* yang disimpan tanpa memperhatikan struktur hubungan antar *record*. Kemudian dilakukan pengembangan terhadap *Hierarchical* Data Model. Dalam model ini setiap tabel memiliki hubungan dengan tabel lain menurut konsep hirarki *parent-child*. Masing-masing tabel *child* memiliki satu tabel *parent*. Pengembangan selanjutnya dilakukan oleh *Network Data Model*. Perbedaannya dengan model hirarki adalah dimana model ini setiap tabel dapat memiliki lebih dari satu pemilik atau *parent* sehinga hubungan antar tabel tidak dibatasi oleh *level* tabel dalam struktur data. Pengembangan berikutnya yang hingga sekarang masih dipakai adalah *Relational Database Mangement System* (RDBMS).

Sifat kontrol sistem *database* adalah terpusat, sehingga biasanya dimiliki dan juga dipegang oleh suatu organisasi. Basis data dapat pula diartikan sebagai penyimpanan data yang dirancang untuk mendukung efisiensi penyimpanan, pemulihan dan pemeliharaan data. Basis data dapat berupa penyimpanan file biner, dokumen, gambar, video, data relasional, data multidimensional, data transaksional, data analitik mapun geografis.

Konsep basis data dimulai dari komponen terkecil hingga membentuk kesatuan basis data adalah kolom,baris (*record*), tabel hingga basis data itu sendiri.

Kolom merupakan salah satu unit informasi yang dapat berupa angka atau karakter. Sementara itu, baris atau *record* merupakan sekumpulan kolom yang saling berhubungan. Contohnya apabila suatu *record* dari karyawan disimpan, maka *record* tersebut terdiri atas beberapa informasi tunggal seperti nama depan, nama belakang, nomor induk karyawan, usia, alamat dan lain-lain. Sementara tabel adalah kumpulan dari banyak *record* yang memiliki kesamaan struktur informasi. Misalnya dalam suatu Perusahaan memilik banyak karyawan, maka dapat terbentuk tabel karyawan yang berisi *record* dari masing-masing karyawan.

Basis data merupakan kumpulan dari tabel yang berhubungan. Misalkan saja dalam suatu Perusahaan terdapat banyak jenis *record* misalnya tabel karyawan dan tabel inventaris.

2.2.2 *Relational Database*

Menurut (Florescu D dan Fourny G, 2013) *Relational Database* dikembangkan Edgar Codd dari IBM pada tahun 1969. Penggunaan *Relational Database* sebagai basis data model saat ini telah mendominasi model lain seperti *hierarchical*, *network* ataupun *object-model*. *Relational Database* dapat diartikan sebagai sebuah basis data yang merepresentasikan data sebagai suatu kumpulan tabel dimana hubungan antar tabel juga direpresentasikan menurut *value* yang ada pada tabel-tabel tersebut. Struktur basis data yang baik tersusun atas relasi antar entitas yang ada dalam suatu data yang terorganisasi sehingga informasi yang tersimpan dapat diakses. Relasi *database* menyimpan datanya di dalam tabel yang terdiri atas baris dan kolom. Suatu baris dalam tabel dapat juga disebut sebagai *record* atau *tuple*, sementara kolom disebut *field* atau *atribut*.

Terdapat tiga jenis atribut yang digunakan pada *database* model relasional yaitu atribut *Primary Key* (PK), *Foreign Key* (FK), dan atribut bukan PK maupun FK. *Primary Key* adalah atribut atau kombinasi beberapa atribut yang memiliki nilai yang unik sehingga digunakan sebagai *identifier* dari baris di dalam sebuah tabel. Pada aturan model basis data relasional, *Primary Key* (PK) tidak boleh bernilai nol (null).

Untuk lebih jelas memahami implementasi *Primary Key* dalam sebuah tabel dapat dilihat pada contoh 2.1

Tabel 2.1: diberikan tabel STUDENT dan baris-baris yang ada dibawah ini. Diasumsikan bahwa atribut ID merupakan *key* dari tabel ini :

Tabel 2. 1 Data Student

ID	STUDENT_NAME	SEMESTER
0011	Budi	6
0012	Doni	8
0014	Eko	6

Kemudian akan dimasukkan *record* baru pada tabel STUDENT dengan data sebagai berikut:

0011	ANGGA	7
	ARDI	5
0015	BOBI	8

0011	ANGGA	7	←	Baris ini tidak dapat disisipkan karena tidak sesuai aturan. ID dengan nilai 0011 sudah ada di dalam tabel
	ARDI	5	←	Baris ini tidak dapat disisipkan karena tidak sesuai aturan. *PK* tidak boleh bernilai NULL
0015	BOBI	8	←	Baris ini dapat disisipkan karena memenuhi aturan *database* relasional.

Sementara itu *Foreign Key* (FK) adalah atribut atau kombinasi dari beberapa atribut yang ada pada sebuah tabel yang juga muncul sebagai atribut *Primary Key* di tabel yang lain. Konsep *Foreign Key* memungkinkan RDBMS untuk menjaga konsistensi di sepanjang baris dari dua relasi atau antar baris dari relasi yang sama. Berbeda dengan PK, FK boleh bernilai nol. Jika nilai dari FK tidak sama dengan nol, maka nilainya harus sama dengan nilai PK di tabel lain dalam satu hubungan antar tabel. Ada pula atribut-atribut yang bukan merupakan *Primary Key* maupun *Foreign Key* yang menyimpan informasi penting mengenai entitasnya dan mendeskripsikan nilai objek pada *Primary Key* secara praktik, konsep FK memastikan bahwa relasi *record* R^1 yang mengacu ke relasi *record* R^2 harus mengacu ke pada *record* R^2 yang sudah ada.

Untuk lebih jelas memahami implementasi *Foreign Key* pada suatu relasi antar tabel maka diberikan contoh 2.2 sebagai berikut.

Tabel 2.2: Diberikan dua buah tabel dibawah yaitu tabel STUDENT dan COURSE. Diasumsikan bahwa atribut STUDENT_ID adalah FK dari tabel COURSE yang mereferensikan atribut ID dari tabel STUDENT.

STUDENT

Tabel 2. 2 Data Student Tabel Induk

ID	STUDENT_NAME	SEMESTER
0011	Budi	6
0012	Doni	8
0014	Eko	6

COURSE

Tabel 2. 3 Data Course Tabel Anak

COURSE _ID	COURSE_NAME	TEACHER_ID	STUDENT_ID
CR001	Mathematic	1202	0012
CR004	Physic	2310	0014
CR002	Chemistry	1328	0011

Kemudian akan dimasukkan *record* baru pada tabel COURSE dengan data sebagai berikut:

CR003	English	3210	0012
CR005	Biology	2761	0043
CR006	Indonesian	2837	NULL

CR003	English	3210	0012	← Baris ini dapat disisipkan karena tidak melanggar aturan *database* relasional.
CR005	Biology	2761	0043	← Baris ini tidak dapat disispkan karena tidak sesuai aturan. ID dengan nilai 0043 tidak ada dalam tabel.
CR006	Indonesian	2837	NULL	← Baris ini dapat disisipkan karena FK boleh berniali NULL. Nilai NULL mungkin saja mengindikasikan bahwa Course tersebut belum diambil siapapun.

Selain *key* yang telah disebutkan diatas, ada beberapa *key* yang masih berhubungan dengan PK dan FK. *Candidate Key* adalah atribut atau kumpulan atribut yang memiliki nilai unik dan dapat mengidentifikasi suatu baris pada tabel. *Candidate Key* merupakan calon *Primary Key*. Adapun *Alternative Key* merupakan *Candidate Key* yang tidak terpilih menjadi *Primary Key*. Selain itu ada pula *Super Key* yang merupakan kombinasi atribut yang mengidentifikasi suatu baris dalam tabel sehingga merepresentasikan informasi mengenai entitasnya.

2.2.3 *MySQL*

Menurut (Miftakhul Huda, 2010) MySQL adalah sebuah perangkat lunak sistem manajemen basis data SQL atau dikenal dengan DBMS (Database Management System), database ini multithread, multi-user. MySQL AB membuat MySQL tersedia sebagai perangkat lunak gratis di bawah lisensi GNU General Public Lisence (GPL), tetapi juga menjual dibawa lisensi komersial untuk kasus-kasus bersifat khusus.

MySQL adalah Relational Database Management System (RDBMS) yang didistribusikan secara gratis di bawah lisensi GPL (General Public License). Dimana setiap orang bebas untuk menggunakan MySQL, namun tidak boleh dijadikan produk turunan yang bersifat closed source atau komersial. MySQL

sebenarnya merupakan turunan salah satu konsep utama dalam database sejka lama, yaitu SQL (Structured Query Language). SQL adalah sebuah konsep pengoperasian database, terutama untuk pemilihan atau seleksi dan pemasukan data yang memungkinkan pengoperasian data dikerjakan dengan mudah secara otomatis. Sebagai database server, MySQL dapat dikatan lebih unggul dibandingkan database server lainnya, terutama dalam kecepatan. Berikut beberapa keistimewaan MySQL, antara lain:

1. ***Portability***

   MySQL dapat berjalan stabil pada berbagai sistem operasi seperti Windows, Linux, FreeBSD, Mac Os X Server, Solaris Amiga, dan masih banyak lagi.

2. ***Multiuser***

   MySQL dapat digunakan oleh beberapa user dalam waktu yang bersamaan tanpa mengalami masalah atau konflik.

3. ***Security***

   MySQL memiliki beberapa lapisan sekuritas seperti level subnetmask, nama host, dan izin akses user dengan sistem perizinan yang mendetail serta password terenkrepsi.

4. ***Scalability* dan *limits***

   MySQL mampu menangani database dalam skala besar, dengan jumlah records lebih dari 50 juta dan 60 ribu tabel serta 5 milyar baris. Selain itu batas indeks yang dapat ditampung menjapai 32 indeks pada setiap tabelnya.

2.2.4 *PHPMyAdmin*

Menurut (Umi DKK, 2011) *PHPMyAdmin* adalah suatu program *open source* yang berbasis *web*, merupakan perangkat lunak bebas yang ditulis dalam bahasa pemrograman PHP yang digunakan untuk menangani administrasi *MySQL* melalui *World Wide Web*. *PhpMyAdmin* mendukung berbagai operasi *MySQL*.

2.2.5 *XAMPP*

Menurut (Riyanto, 2010) XAMPP adalah sebuah software yang berfungsi untuk menjalankan website berbasis PHP dan menggunakan pengolah data MySQL yang dijalankan dikomputer secara lokal. XAMPP berperan sebagai web server pada komputer. XAMPP juga dapat disebut sebuah CPanel server virtual, yang dapat membantu Anda melakukan preview sehingga dapat memodifikasi website tanpa harus online atau terakses dengan internet. Software XAMPP bersifat open sources yang dapat diperoleh secara gratis dari situs www.apachefriends.org. XAMPP adalah perangkat lunak yang mendukung banyak sistem operasi dan merupakan komplikasi dari beberapa program. Fungsinya adalah sebagai server yang berdiri sendiri dan terdiri atas Apache, MySQL, dan bahasa pemrograman PHP.

2.2.6 *Entitiy Relationship Diagram (ERD)*

Menurut (Imbar, Radiant & Billy, 2011) *Entitiy Relationship Diagram* (ERD) adalah suatu penyajian data menggunakan *Entity* dan *Relationship*. *ERD* merupakan peralatan pembuatan model data yang paling fleksibel dan dapat diadaptasi untuk berbagai pendekatan yang mungkin diikuti perusahaan dalam

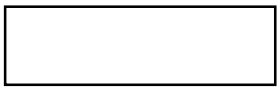
mengembangkan sistem. ERD ini menggambarkan relasi atau hubungan antar entitas yang ada.

Menurut (Mustafa, 2012) *ERD* merupakan notasi garis dalam permodelan data konseptual yang mendeskripsikan hubungan antara penyimpanan. *ERD* digunakan untuk memodelkan struktur data dan hubungan antar data, karena hal ini relatif kompleks. Dengan *ERD* menguji model dengan mengabaikan proses yang diperlukan, dan bagaimana data yang satu berhubungan dengan yang lain.

ERD menggunakan sejumlah notasi dan simbol untuk menggambarkan struktur dan hubungan antar data. Pada dasarnya ada 3 (tiga) macam simbol yang digunakan, yaitu:

1. **Entitas (*Entity*)**

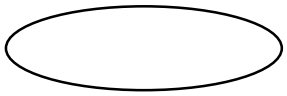
   Entitas adalah suatu objek yang dapat diidentifikasi dalam lingkungan pemakai. Entitas digambarkan menggunakan persegi empat, seperti pada **Gambar 2.1**.:



Gambar2. 1 Bentuk Entitas

2. **Atribut (*Fields*)**

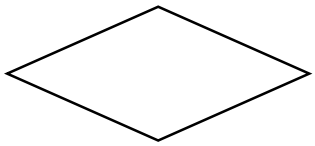
   Atribut adalah elemen yang ada pada setiap entitas, berfungsi untuk mendeskripsikan karakteristik dari entitas tersebut. Isi dari atribut mempunyai sesuatu yang dapat mengidentifikasikan isi elemen satu dengan yang lain. Atribut diwakili oleh simbol *ellips*. Seperti pada **Gambar 2.2**. :



Gambar2. 2 Bentuk Atribut

3. Hubungan atau relasi

Relasi adalah penghubung antara suatu *entity* dengan *entity* yang lain dan merupakan bagian yang sangat penting dalam mendesain *database*. Seperti pada **Gambar 2.3**. :



Gambar2. 3 Hubungan *Entity* dengan Relasi

***4. Relationship* (Hubungan)**

Relationship merupakan hubungan persekutuan antara dua *entity* atau lebih yang saling berkaitan. Hubungan antara satu entiti dengan entiti lain, meliputi:

a. Hubungan satu ke satu (*one to one*)
b. Hubungan satu ke banyak (*one to many*)
c. Hubungan banyak ke banyak (*many to many*)

5. ER (Entity Relationship) Model

Entity Relationship Model adalah permodelan data dalam *database* biasanya menggunakan *Entity Relation Model* (ER_Model). Karakter dari model adalah:

a. Untuk setiap *non weak entity* R, dibuat *relation* skema yang atributnya terdiri atas atribut E tersebut.
b. Untuk setiap *weak entity* W yang diwakili oleh *entity* E dibuat skema relasi yang atributnya terdiri dari skema atribut W dan sebagai *foreign key* adalah kunci utama dari E.

c. Untuk setiap *binary relationship* 1 : 1 antara S dan T maka dipilih satu *entity* misalnya S, dibuat *relation* skema yang atributnya adalah semua atribut dalam S ditambah satu *foreign key* yaitu *key* dari T.
d. Untuk setiap *binary relationship* 1 : N antara S dan T dibuat *relation* skema dengan semua atribut T plus satu *foreign key* yaitu *key* dari S.
e. Untuk setiap *binary* N : M antara *relation entity* S dan T, dibuat *relation* yang mengandung semua *primary key* di S dan T.
f. Untuk setiap *multivalue* atribut A dari entiti E, dibuat *relation* skema dengan atributnya adalah A itu sendiri dan *primary key* dari E.
g. Untuk setiap *non binary relationship* dibuat *relation* skema baru yang atributnya *key* dari semua *relation* yang berhubungan ditambah dengan atribut yang terdapat pada relasinya.

2.2.7 *Store Procedure*

Menurut (Ampari J Ansanay, 2013) *Store Procedure* merupakan fitur utama yang paling penting di *MySQL*. *Store Procedure* merupakan suatu kumpulan perintah atau *statement* yang disimpan dan dieksekusi di server *database MySQL*. Dengan SP (*Stored Procedure*), pengembang sistem dapat menyusun program sederhana berbasis sintaks *SQL* untuk menjalankan fungsi tertentu.

Untuk membuat *store procedure*, dengan menjalankan perintah "*create procedure*" diikuti dengan *SQL script* atau melalui *Query Analyzer* pada *MS SQL* Server, atau dengan menggunakan menu *New Procedure* pada *Enterprise Manager*. Berikut adalah kerangka sederhana *Store Procedure*:

```
CREATE PROC procedure_name
   [{@parameter data_type}
   ]
AS sql_statement
```

Penggunaan *store procedure* dalam sebuah pemrograman *database* memiliki beberapa keuntungan atau kelebihan sebagai berikut:

a. *Performance*

Semua perintah *SQL*, yang di kirimkan ke *database* server melalui kumpulan *action* yang disebut dengan *execution*. Hal ini menjadikan perintah *SQL* yang dikirimkan melalui beberapa proses sebelum data dikembalikan ke *client*. *User* mengirimkan *request* untuk mengeksekusi *store procedure*. *SQL* Server kemudian mengecek apakah ada *syntax error*. Mengidentifikasi dan melakukan pengecekan alias pada *FROM clausa*. Membuat *query* plan. Meng-*compile* dan kemudian mengeksekusi *query* plan untuk kemudian mengembalikan data yang diminta.

b. *Security*

Store Procedure memberikan keuntungan yang baik dalam hal keamanan. Dengan menggunakan *store procedure*, dapat diberikan *permission* untuk *user* yang ditunjuk agar dapat mengakses data, menekan *immense coding* yang perlu di lakukan pada *Application Client*. Ini adalah cara terbaik untuk mengontrol akses ke data.

c. *Maintenance*

Jika menggunakan *store procedure* untuk mengakses *database*, setiap perubahan pada *database* dapat dipantau berdasarkan *client application*. Hal ini memungkinkan untuk mengetahui secara persis dimana data diakses, dan juga untuk mengetahui dimana harus dilakukan perubahan.

d. *Minimal processing at the client*

Dengan menggunakan *store proocedure* membantu untuk membuat *batch* perintah *SQL*, yang bisa digunakan untuk *manage transaction* dan *constraints*. Aplikasi lebih terfokus pada menampilkan data untuk keperluan *user* dan aplikasi *client* tidak mengetahui banyak mengenai *database*. Sebagai contoh jika terdapat *database* yang berisi ribuan *rows* dan ratusan *table* diperlukan beberapa perhitungan sebelum melakukan *update* pada setiap *record*. Jika diambil data secara lengkap ke *client* dan *client computer* memproses data secara lengkap, dapat dibayangkan apa yang ditimbulkan. Tetapi jika *client* bisa mengeksekusi *store procedure*.