BAB II
TINJAUAN PUSTAKA DAN LANDASAN TEORI

2.1 Tinjauan Pustaka

Menurut penelitian yang dilakukan oleh Susi Susilowati pada Baitul Maal Bogor, sebuah perusahaan yang bergerak dalam bidang pengelolaan zakat yang pengelolaanya masih dilakukan secara manual. Oleh sebab itu dikembangkan suatu sistem informasi pengelolaan zakat berbasis komputer, menggunakan metode *waterfall* sebagai alur dari pengembangan sistem yang bertujuan untuk mempermudah dalam mengelola data penerimaan maupun penyaluran ziswah agar lebih cepat dan efisien, sehingga dapat menghemat waktu dan mengurangi banyaknya kesalahan [2].

Penelitian lainya dilakukan oleh Ika Afriani, Supriyanto, Nuril Anwar, Universitas Ahmad Dahlan, Yogyakarta dalam Seminar Nasional Hasil Pengabdian Kepada Masyarakat [3]. Menurut mereka pengelolaan aspek transparansi dan akuntabilitas bagi Badan Amil Zakat Nasional (BAZNAS) sebagai pengelolaan dana publik yang utama adalah mendapatkan kepercayaan dari masyarakat luas. Mereka merancang aplikasi *website* manajemen masjid, aplikasi ini juga dapat dibuka melalui *smartphone*, aplikasi yang dibuat dapat di manfaatkan oleh para pengurus LAZ dan BAZNAS untuk mencatat manajemen sesuai kebutuhan masing-masing.

Penelitian berikitnya dilakukan oleh Kartika Handayani, Nurmala Sari, Anna, Latifah Program Studi Sistem Informasi Kota Pontianak Universitas Bina Saraca Informatika [4]. Penelitian ini dilakukan karena melihat beberapa lembaga pengelolaan ZISWAF khususnya di Pontianak belum memiliki sistem informasi yang dapat membantu pengelolaan ZISWAF, dengan metode yang dalam penelitian ini menggunakan metode *waterfall*.

Perbedaan dari penelitian yang dilakukan dengan penelitian terdahulu yakni terdapat pada pengelolaan dan juga pemanfaatan sistem yang terdapat pada pengelolaan data donatur, pemasukan, pengeluaran keuangan, informasi laporan serta fitur transfer untuk membantu donatur untuk melakukan infak. Tahap pengembangan sistem informasi ini menggunakan metode SDLC (*System Development Life Cycle*) model *Waterfall* yang dibangun menggunakan bahasa pemrograman PHP

(*Hypertext Preprocessor*) dan *MySQL* sebagai aplikasi *database* dengan penambahan fitur SMS *Gateway*

2.2.3 Pengertian Infak

Secara bahasa infak berasal dari kata "Anfaqa" yang berarti memberikan sesuatu kepada orang lain. Dalam terminologi syariat, infak adalah mengeluarkan atau memberikan sebagian pendapatan untuk kepentingan yang diperintahkan oleh agama islam. Infak tidak ditentukan jumlah pengeluaranya dan tidak ditentukan untuk sasaran pengunaanya Infak sangat luas sasarannya untuk semua kepentingan pembangunan umat. (QS. Ali-Imron:134 ; Aththalaq:7) dan (QS.Al-Baqarah:215) [5].

2.2 Landasan Teori

2.2.1 Pengertian Aplikasi

Aplikasi secara umum adalah alat terapan yang difungsikan secara khusus dan terpadu sesuai kemampuan yang dimilikinya, aplikasi merupakan suatu perangkat komputer yang siap pakai bagi *user* [6].

2.2.2 Pengertian Sistem Informasi

Sistem Informasi adalah suatu sistem di dalam suatu organisasi yang mempertemukan kebutuhan pengolahan transaksi harian, mendukung operasi, bersifat manajerial dan kegiatan strategi dari suatu organisasi dan menyediakan pihak luar tertentu dengan laporan-laporan yang diperlukan. Dengan berkembangnya sistem informasi saat ini, banyak sistem informasi pada organisasi yang ingin mencapai tahap sistem informasi secara cepat, relevan dan akurat. Sistem Informasi merupakan cara yang terorganisir untuk mengumpulkan, memasukan, dan memroses data dan penyimpananya, mengelola, mengontrol dan melaporkanya sehingga dapat mendukung perusahaan atau organisasi untuk mencapai tujuan. Jadi dapat disimpulkan bahwa sistem informasi adalah sebuah rangkaian prosedur yang menggabungkan subsistem-subsistem yang memepertemukan kebutuhan organisasi dengan laporan yang diperlukan [7].

2.2.4 Rekayasa Web

Merupakan proses yang digunakan untuk menciptakan aplikasi web. Mengadaptasi rekayasa perangkat lunak dalam hal konsep dasar yang menekankan pada aktifitas teknis dan manajemen tapi dengan dan

penyesuaian yang dilakukan dalam Web Engineering dimulai dari customer communication yang memiliki bagian yaitu business analysis dan formulation seperti mendiskusikan dengan client termasuk dalam hal pengumpulan informasi-informasi mengenai kebutuhan sistem, planning seperti perencanaan pengembangan sistem diantaranya penjadwalan tugas rincian tugas-tugas yang akan dikerjakan, modelling yaitu seperti proses pembuatan use case, activity diagram serta mendesain tampilan sistem, construction yaitu seperti proses pembuatan coding program [8].

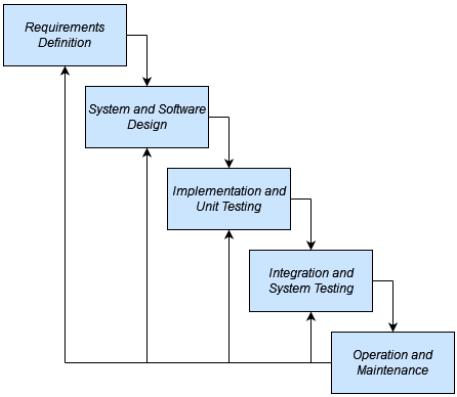
2.2.5 Rekayasa Perangkat Lunak

Menurut Janner Simarmata Rekayasa Perangkat Lunak adalah suatu bidang profesi yang mendalami cara membangun perangkat lunak termasuk pembuatan, pemeliharaan aplikasi perangkat lunak dengan menerapkan teknologi dan peraktik dari ilmu komputer, manajemen organisasi pengembangan perangkat lunak, dan bidang-bidang lainnya. Secara umum istilah Rekayasa Perangkat Lunak disepakati sebagai terjemahan dari *Software Engineering. Software Engineering* dipopulerkan pada tahun 1968 pada *Software Engineering Conference* yang diselenggarakan oleh NATO [9].

A. Metode Pengembangan Sistem

Tahapan pengembangan sistem dalam penelitian ini menggunakan SDLC (*Sytem Development Life Cycle*). Adalah proses atau mengubah suatu sistem perangkat lunak dengan mengunakan model-model dan metodologo yang digunakan untuk mengembangkan sistem perangkat lunak sebelumnya berdasarkan *best pratice* atau cara-cara yang sudah disebut baik. SDLC yang digunakan dalam pengembangan sistem menggunakan SDLC model *Waterfall* menurut lan Pressman [10].

Pengembangan ini mengunakan metode *Waterfall* menurut Pressman, adalah metode air terjun kadang dinamakan siklus hidup klasik (*classic life cyle*). Hal ini menceritakan pendekatan yang sistematis dan berurutan (sekuensial) pada Pengembangan perangkat lunak dimulai dari spesifikasi kebutuhan pengguna dan berlanjut melalui tahapan-tahapan seperti Gambar 2.1 [11].



Gambar 2. 1 Metode waterfall menurut Pressman.

1. *Requirements Definition*
   Tahapan ini penulis melakukan proses pencarian kebutuhan yang difokuskan pada *software*. Dengan tujuan untuk mengetahui sifat dari program yang akan dibuat, maka para *software engineer* harus mengerti informasi tentang *domain* dari *software*, misalnya fungsi yang dibutuhkan *user interface*. Dari aktifitas tersebut harus didokumentasikan dan ditunjukan terhadap pelanggan.
2. *System and Software Design*
   Dari hasil kebutuhan terhadap sistem yang dibuat, proses ini digunakan untuk Ubah kebutuhan diatas kedalam bentuk "*blueprint*" *software* sebelum masuk ke tahap penulisan program (*coding*). Desain harus mengimplementasikan kebutuhan yang telah disebutkan ditahap sebelumya dan didokumentasikan sebagai konfigurasi dari *software.*
3. *Implementation and Unit Testing*
   Penulisan kode program bertujuan untuk mengartikan bahasa manusia untuk nantinya bisa dikenali oleh komputer sehingga akan menghasilkan sistem yang diinginkan oleh penguna melalui proses *coding*, tahap ini merupakan implementasi dari tahap *design* yang secara teknis nantinya dikerjakan oleh programmer.
4. *Integration and System Testing*
   Setelah melakukan penulisan kode pemrograman haruslah diuiji cobakan. Demikian juga dengan *software* harus diuji cobakan

untuk mengetahui fungsional *input* ataupun *output* dari sistem yang dibuat agar bebas dari *eror* dan selanjutnya dapat digunakan oleh penguna, pada tahap ini bisa dikatakan sebagai hasil akhir dalam pembuatan sitem.

5. *Operation and Maintenance*
   Sistem yang telah selesai dalam pengujian selanjunya akan digunakan oleh penguna, dalam penggunaan sistem ini dibutuhkan pemeliharaan suatu *software*, termasuk didalamnya adalah pengembangan, karena *software* yang dibuat tidak selamanya hanya seperti itu, ketika di jalankan mungkin saja mengalami perubahan yang disebabkan oleh kesalahan perangkat lunak yang tidak ditemukan sebelumnya, dan penambahan fitur yang belum ada pada pada software tersebut karena harus menyesuaikan dengan apa yang dibutuhkan oleh pengguna.

A. Metode Pengujian Sistem

Metode pengujian sistem yang digunakan adalah metode *Black Box Testing*. *Black Box Testing* adalah metode pengujian perangkat lunak yang berfokus pada fungsionalitas perangkat lunak yang memiliki tujuan untuk menemukan kesalahan antarmuka, struktur data, perfomansi, kesalahan inisialisasi [12].

Ciri-ciri *Black Box Testing* [13]:

1. *Black Box Testing* berfokus pada kebutuhan fungsional *software* berdasarkan spesifikasi kebutuhan dari *softwere.*
2. *Black Box Testing* bukan teknik alternatif dari *White Box Testing*. Melainkan pendekatan pelengkapan dalam mencakup *error* dengan kelas yang berbeda dari metode *White Box Testing.*
3. *Black Box Testing* melakukan pengujian tanpa pengetahuan detail struktur internal dari sistem yang diuji.

Kategori *error* yang diketahui melalui *Black Box Testing* [14]:

1. Fungsi-fungsi yang tidak benar atau hilang.
2. Kesalahan *interface*.
3. Kesalahan dalam struktur data.
4. Keslahan inisialisasi dan terminasi.
5. Kesalahan validitas fungsional.

6. Kesalahan sistem dalam nilai *input*.
7. Batasan dari suatu data.

2.2.6 Pemrograman Berorientasi Objek

Metodologi berorientasi objek adalah strategi pembangunan perangkat lunak sebagai kumpulan objek yang berisi data dan informasi yang diberlakukan terhadapnya. Metodologi berorientasi objek merupakan cara bagaimana sistem perangkat lunak dibangun melalui pendekatan objek secara sistematis melalui rangkaian aktivitas analisis orentasi objek, pemrograman berorientasi objek, dan pengujian berorientasi objek. Ada beberapa konsep dasar yang harus dipahami dalam pemrograman berorientasi objek diantaranya [15]:

1. Kelas (*class*)
   Kelas adalah kumpulan objek-objek dengan karakteristik yang sama. Kelas merupakan definisi statik dan himpunan objek yang sama yang lahir atau diciptakan dari kelas tersebut. Sebuah kelas akan mempunyai sifat (atribut), kelakuan (operasi/metode), hubungan (*relationship*) dan arti suatu kelas dapat diturunkan dari kelas yang lain.
2. Objek (*objek*)
   Objek adalah abstraksi dan suatu yang mewakili dunia nyata seperti benda, manusia, organisasi, tempat, kejadian, struktur, status, atau hal lain yang bersifat abstrak. Objek merupakan suatu entitas yang mampu menyimpan informasi (status) dan mempunyai operasi (kelakuan) yang dapat diterapkan dan berpengaruh pada status objeknya. Objek memiliki siklus hidup yaitu diciptakan, dimanipulasi, dan dihancurkan.
3. Metode (*method*)
   Operasi atau metode atau *method* pada sebuah kelas hampir sama dengan fungsi atau prosedur pada metodologi setruktural. Sebuah kelas boleh memiliki lebih dari satu metode atau operasi. Metode atau operasi yang berfungsi untuk memanipulasi objek itu sendiri. Operasi atau metode merupakan fungsi atau transformasi yang dapat dilakukan terhadap objek atau dilakukan oleh objek.
4. Atribut (*attribute*)
   Atribut dari sebuah kelas adalah variabel global yang dimiliki. Atribut dapat berupa nilai atau elemen data yang dimiliki oleh objek dalam kelas objek. Atribut dipunyai secara individual oleh

sebuah objek, misalnya berat, jenis, nama, dan lain sebagainya. Atribut sebaliknya bersifat prifat untuk menjaga konsep enkapsulasi.

5. Abstraksi (*abstraction*)
   Prinsip untuk mempresentasikan dunia nyata yang kompleks menjadi satu bentuk model yang sederhana dengan mengabaikan aspek-aspek lain yang tidak sesuai dengan permasalahan.
6. Enkapsulasi (*encapsulation*)
   Pembungkusan atribut data dan layanan (operasi-operasi) yang dipunyai objek untuk menyembunyikan implementasi dan objek sehingga objek lain tidak mengetahui cara kerjanya.
7. Pewarisan (*inheritance*)
   Mekanisme yang memungkinkan satu objek mewarisi sebagian atau seluruh definisi dan objek lain sebagai bagian dan dirinya.
8. Antarmuka (*interface*)
   Antarmuka atau *interface* sangat mirip dengan kelas, tapi tanpa atribut kelas dan memiliki metode yang dideklarasikan tanpa isi. Deklarsi metode pada sebuah *interface* dapat diimplementasikan oleh kelas lain.
9. Reusability
   Pemanfaatan kembali objek yang sudah didefinisikan untuk suatu permasalahan lainnya yang melibatkan objek tersebut.
10. Generalisasi dan spesifikasi
    Menunjukan hubungan antara kelas dan objek yang umum dengan kelas dan objek yang khusus.
11. Komunikasi antar objek
    Komunikasi antar objek dilakukan lewat pesan (*message*) yang dikirimkan dan satu objek ke objek lainya.
12. Polimorfisme (*polymorphism*)
    Kemampuan suatu objek yang digunakan dalam tujuan yang berbeda dengan nama yang sama sehingga menghemat baris program.
13. *Package*
    Package adalah sebuah kontainer atau kemasan yang dapat digunakan untuk mengkelompokan kelas-kelas sehingga bernama sama disimpan dalam *package* yang berbeda

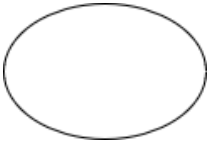
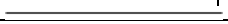
2.2.7 UML (Unified Modelling Language)

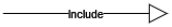
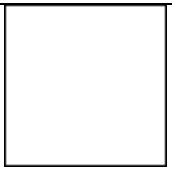
Merupakan bahasa pemodelan visual digunakan dalam menentukan, memvisualisasi, membangun, dan mendokumentasikan perangkat lunak, lebih dari 14 jenis diagram umum yang dapat dikategorikan pengambaran informasi struktural dan diagram yang menjelaskan informasi pelaku [16].

1. *Use Case* Diagram

Adalah proses merepresentasikan hal-hal yang dapat dilakukan oleh aktor dalam menyelesaikan pekerjaan, *Use case* diagram juga menjelaskan sebuah interaksi antara aktor dengan sistem informasi yang akan dibuat. Berikut Tabel 2.1 merupakan simbol-simbol yang ada pada *use case* diagram [17]:

Tabel 2. 1 Simbol-simbol *Use Case* Diagram

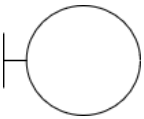
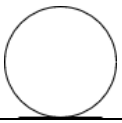
No	Simbol	Nama	Keterangan
1.		*Use case*	Fungsionalitas yang disediakan sistem sebagai unit atau aktor yang saling bertukar pesan dinyatakan dengan mengunakan kata kerja.
2.		*Actor*	Peran yang berinteraksi dengan sistem informasi yang dibuat, simbol aktor berupa gambar orang tetapi aktor belum tentu merupakan orang, biasanya dinyatakan dengan kata benda.
3.		*Association*	Komunikasi antara

			aktor dan *use case*, menghubungan objek satu dengan yang lainya.
4.		*Include*	Relasi *use case* tambahan ke sebuah *use case* untuk menjalankan fungsinya atau sebagai syarat dijalankan *use case* ini.
5.		*Extend*	Relasi *use case* tambahan ke sebuah *use case* yang dapat berdiri sendiri walau tanpa *use case* tambahan.
7.		*System*	Paket yang menampilkan sistem secara terbatas.

2. *Sequence* Diagram

Sequence diagram adalah gambar diagram kolaborasi dinamis antara sejumlah objek yang berfungsi untuk menunjukkan rangkaian pesan yang dikirim, dan mengambarkan interaksi objek dalam urutan waktu. Berikut tabel 2.2 adalah simbol-simbol yang terdapat pada *sequence* diagram [18].

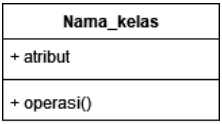
Tabel 2. 2 Simbol-simbol *Sequence* Diagram.

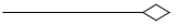
No	Simbol	Nama	Keterangan
1.		*Lifeline*	Objek *entity*, yang saling berinteraksi.
2.		*Actor*	Mengambarkan *user* atau penguna.
3.		*Message*	Spesifikasi dari komunikasi antar objek yang memuat informasi-informasi tentang aktivitas yang terjadi.
4.		*Boundary*	Mengambarkan sebuah *form.*
5.		*Control Class*	Menghubungkan *boundary* dengan tabel.
6.		*Entity Class*	Mengambarkan hubungan kegiatan yang akan dilakukan.

3. *Class* diagram

Class diagram adalah model dari gambar struktur dan pendefinisian *class* yang dapat terhubung antara *class* yang lain yang akan dibuat untuk membangun sistem [19]. Berikut Tabel 2.3 adalah simbol-simbol yang terdapat pada *class* diagram [20]:

Tabel 2. 3 Simbol-simbol Class Diagram.

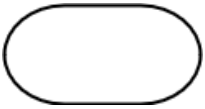
No	Simbol	Nama	Keterangan
1.		Kelas	Kelas pada struktur sistem
2.		Antarmuka/*interface*	Sama dengan konsep *interface* dalam pemrograman berorientasi objek
3.		Asosiasi/*association*	Relasi antar kelas yang biasanya disertai dengan *multiplicity*.
4.		Asosiasi berarah/*directed association*	Relasi antar kelas yang satu digunakan oleh kelas yang lain, biasanya disertai dengan *multiplicity*.
5.		Generalisasi	Relasi antar kelas dengan makna generalisasi-spesialisasi (umum khusus)
6.		Kebergantunga	Relasi antar kelas

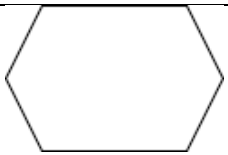
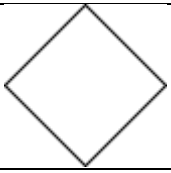
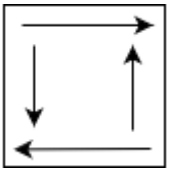
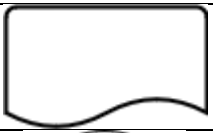
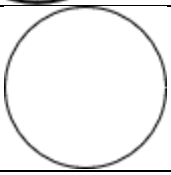
		n/*dependency*	dengan makna kebergantungan antar kelas
7.		Agregasi/*aggregation*	Relasi antar kelas dengan makna semua bagian.

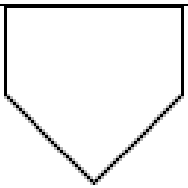
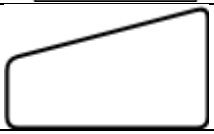
2.2.8 Flowchart

Flowchart adalah pengambaran secara grafik dari langkah-langkah dan urutan-urutan prosedur dari suatu program. Dalam pemecahan masalah seorang Programmer memerlukan segmen-segmen yang lebih kecil dalam menganalisi alternatif dalam pengoprasian. *Flowchart* membantu dalam penyelesain masalah yang perlu dipelajari dan dievaluasi lebih lanjut. *Flowchart* merupakan bentuk gambar /diagram yang mempunyai aliran satu atau dua arah secara sekuensial [21]:

Tabel 2. 4 Simbol-simbol *Flowchart*

No	Simbol	Nama	Keterangan
1.		*Terminal*	Mulai dan mengakhiri suatu program
2.		*Proses*	Proses perhitungan atau proses pengolahan data
3.		*Predefined Process (sub program)*	Permulaan sub program atau proses pengolahan data
4.		*Preparation*	Proses inisialisasi atau pemberian harga awal

			
5.		*Input-Output*	Memasukan data maupun menunjukan hasil dari suatu *process* tanpa tergantung dengan jenis peralatanya
6.		*Decision*	Memilih proses berdasarkan kondisi yang ada
7.		*Flow*	Menghubungkan antara simbol satu dengan simbol yang lain atau menyatakan jalanya arus dalam suatu proses. Simbol arus ini sering disebut dengan *connecting line*
8.		*Document*	Merupakan simbol untuk data yang terbentuk informasi
9.		*On Page Connector*	Penghubungan bagian-bagian *flowchart* yang berada pada suatu halaman
10.		*Off Page*	Penghubung

		*Connector*	bagian-bagian *flowchart* yang berada pada halaman berbeda
11.		*Manual Operation*	Menunjukan pengolahan yang tidak dilakukan oleh komputer
12.		*Manual Input*	Memasukan data secara manual *on-line keyboard*

2.2.9 Basis Data (Database)

Basis data atau database adalah sekumpulan relasi data logika, dan deskripsi dari data yang dirancang untuk memenuhi kebutuhan informasi organisasi. Basisi data atau database adalah kumpulan data yang mewakili berbagai macam entitas dan hubungannya yang dapat digunakan secara bersamaan oleh banyak pengguna dan dirancang untuk memenuhi kebutuhan informasi organisasi [22]. Basis data memungkinkan tempat penyimpanan datayangbesar dan dapat digunakan secara bersamaan oleh banyak departemen dan pengguna. Database mewakili entitas, atribut, dan hubungan logis antara entitas. Basis data terdiri dari kumpulan data yang terorganisir, relasi antar data, dan objektifnya. Objektif utama adalah kecepatan dan kemudahan berinteraksi dengan data yang dikelola atau diolah. Selain itu terdapat pengertian bahwa basis data adalah sekumpulan data persisten yang digunakan oleh aplikasi sistem dari perusahaan..

1. DBMS (*Database Management System*)

DBMS (*Database Management System*) merupakan perantara dari *user* dengan basis data. Dalam pengunaan nya memerlukan bahasa basis data yang sudah ditetapkan dari perusahaan DBMS, terdiri dari berbagai macam intruksi yang diformulasikan sehingga dapat diperoses oleh DBMS. DBMS menyediakan beberapa fasilitas sebagai berikut [23]:

a. DDL (*Data Definition Language*) adalah sebuah metode *Query* SQL yang berguna untuk mendefinisikan data pada sebuah *database*, adapun *Query* yang dimiliki adalah :
    1. *CREATE* : Digunakan untuk pembuatan *table* dan *database*.
    2. *DROP* : Digunakan untuk penghapusan *table* dan *database.*
    3. *ALTER* : Digunakan untuk pengubahan struktur *table* yang dibuat, baik menambah *field* (*add*), mengganti nama *field* (*change*), ataupun menamakannya kembali (*rename*) serta menghapus (*drop*).
b. DML (*Data Manipulation Language*) adalah sebuah metode *Query* yang dapat digunakan apabila DDL telah terjadi, sehingga fungsi dari *Query* ini adalah untuk melakukan pemanipulasian database yang telah ada atau telah dibuat sebelumnya. Adapun *Query* yang termasuk didalamnya adalah :
    1. *INSERT* : Digunakan untuk penginputan data pada *table database*.
    2. *UPDATE* : Digunakan untuk pengubahan data pada *table database*.
    3. *DELETE* : Digunakan untuk penghapusan data pada *table database*.

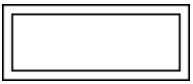
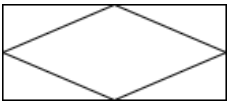
2. SQL (*Struktured Query Language*)

Structured Query Language (SQL) di bangun di laboratorium IBM-san jose california sekitar tahun 70-an, merupakan bahsa yang banyak digunakan dalam berbagai produk *database* yang pertama kali dikembangkan sebagai bahasa DB2 yang menjadi produk andalan IBM. Saat ini (ANSI) organisasi Standar America menetapkan standar bahasa SQL yaitu ANSI-92 standard[24].

3. ERD (*Entity Relationship Diagram*)

ERD (Entity Relationship Diagram) merupakan tools yang digunakan untuk memodelkan struktur data dengan menggambarkan entitas dan hubungan antar enititas (relationship) secara abstrak (konseptual). ERD berfungsi untuk memodelkan struktur data dan hubungan antar data, untuk menggambarkannya digunakan beberapa notasi dan symbol. Berikut Tabel 2.5 adalah simbol-simbol dalam ERD [25] :

Tabel 2. 5 Simbol-simbol ERD.

No	Simbol	Nama	Keterangan
1.		*Entity*	Suatu entitas yang memiliki nama benda, orang atau lokasi digambarkan dengan persegi panjang
2.		*Weak Entity*	Entitas yang tidak dapat diidentifikasi melalui atributnya sendiri. Weak entity bergantung pada enty lain (*owner entity*)
3.		*Associative*	Entitas yang digunakan pada banyak antar banyak hubungan
4.		*Attribute*	Mendeskripsikn karakter entitas
5.		*Key attribute*	Atribut yang berfungsi sebagai kunci
6.		*Link*	Penghubung antar relasi dengan entitas dan entitas dengan atribut.