



BAB II

DASAR TEORI

BAB II

TINJAUAN PUSTAKA DAN LANDASAN TEORI

2.1 Tinjauan Pustaka

Penelitian serupa pernah dilakukan dengan judul Perancangan Sistem Informasi Pengelolaan Jasa Percetakan Agna Advertising Berbasis Web. Dalam penyusunan Tugas Akhir ini penulis menggunakan komponen-komponen UML seperti *activity diagram*, *use case diagram*, *sequence diagram*, dan *deployment diagram* serta analisa data menggunakan *entity relationship diagram*. Sedangkan untuk rancangan metode penulis menggunakan metode SDLC *Waterfall*. Dengan dikembangkannya sistem penjualan jasa yang terkomputerisasi diharapkan dapat membantu mengurangi kesalahan pencatatan dan perhitungan yang sering dilakukan oleh pengguna[5].

Penelitian lainnya dengan judul Aplikasi Penjualan Pada Percetakan Zahira Print Palembang. Adapun pemrograman yang digunakan dalam penulisan penelitian ini adalah pemrograman website menggunakan sistem Operasi Windows 10, serta Notepad++ untuk pembuatan script php. Dari hasil pengujian terhadap sistem yang diharapkan dapat membantu dalam pengolahan transaksi penjualan dan laporan keuangan[6].

Adapun penelitian serupa lainnya yang telah dilakukan sebelumnya yaitu tentang Rancang Bangun Sistem Informasi Penjualan Jasa Percetakan Berbasis Website Studi Kasus CV Prima Framedia. Sistem informasi ini berfungsi untuk membantu memperkenalkan dan memasarkan produk dengan jangkauan pasar yang lebih luas sehingga mampu meningkatkan omset penjualan. Sistem yang dibangun adalah sistem berbasis web, menggunakan metode *Waterfall* serta perancangan program pada sistem yang tersusun dengan mengaplikasikan bahasa PHP[7].

Penelitian yang akan dilakukan oleh peneliti tentu akan menjadi penelitian pertama karena sistem informasi data penjualan di Level Digital Printing menjadi sistem pertama yang akan dibuat oleh peneliti, acuan peneliti untuk membuat sistem ini adalah dengan melakukan penelitian dari berbagai sistem berbeda yang telah dilakukan penelitian sebelumnya. Penelitian Sistem Informasi data penjualan di Politeknik Negeri Cilacap sebelumnya yaitu dari pembangunan Sistem Informasi ini

memungkinkan data-data dapat diakses sehingga dalam penyajian informasi tentang produk, pemesanan produk, bukti pembayaran, dan laporan yang dibutuhkan dapat diperoleh dengan lebih mudah. Metode pengembangan sistem yang digunakan peneliti adalah metode *Waterfall* di mana pengembangan sistem dapat diselesaikan dalam waktu yang lebih cepat dengan biaya yang murah. Adapun tahapan pembuatannya adalah, pertama membuat prosedur yang ada dengan Document Flow Diagram. Kedua menganalisa permasalahan yang ada menggunakan Fishbone Diagram. Ketiga merancang proses menggunakan Unified Modelling Language. Keempat membuat desain database yang berupa Conceptual Data Model dan Physical Data Model. Kelima membuat desain menggunakan Graphical User Interface. Keenam melakukan pemodelan tiap-tiap proses menggunakan Flowchart Diagram. Ketujuh membangun database menggunakan MySQL. Kedelapan Membangun sistem menggunakan Bahasa pemrograman PHP. Terakhir melakukan uji coba dengan metode Blackbox Testing.

2.2 Landasan Teori

Landasan teori berisi hal-hal atau teori-teori yang berkaitan dengan permasalahan dan ruang lingkup permasalahan sebagai landasan dalam pembuatan laporan ini.

2.2.1 Sistem Informasi

Secara umum sistem dapat diartikan sebagai suatu kumpulan dari berbagai komponen yang saling berinteraksi, saling berhubungan dan bekerja sama dalam mencapai tujuan tertentu . Pengertian sistem menurut beberapa ahli seperti dibawah ini:

Sistem adalah sekelompok elemen-elemen yang terintegrasi dengan tujuan yang sama untuk mencapai tujuan. Sistem juga dapat diartikan sebagai suatu jaringan kerja dari prosedur-prosedur yang saling berhubungan, terkumpul bersama-sama untuk melakukan suatu kegiatan atau untuk tujuan tertentu[8].

Informasi adalah data yang diolah menjadi bentuk lebih berguna dan lebih berguna bagi menerimanya. Informasi juga disebut data yang diproses atau data yang memiliki arti[8].

Informasi adalah data yang telah diolah menjadi bentuk yang lebih berarti bagi penerimanya. Data terdiri dari fakta-fakta dan angka-angka yang relative tidak berarti bagi pemakai[9].

Dari pengertian informasi di atas dapat disimpulkan bahwa informasi adalah pengolahan data yang berdasarkan fakta dan dapat memberikan arti dan manfaat bagi penerimanya.

Sistem Informasi (*Information System*) merupakan kombinasi teratur dari orang-orang, perangkat keras (*hardware*), perangkat lunak (*software*), jaringan komunikasi, dan sumber daya data yang mengumpulkan, mengubah, dan menyebarkan informasi dalam sebuah organisasi. Sistem informasi juga merupakan suatu kumpulan dari komponen-komponen dalam organisasi yang berhubungan dengan proses penciptaan dan aliran informasi[9].

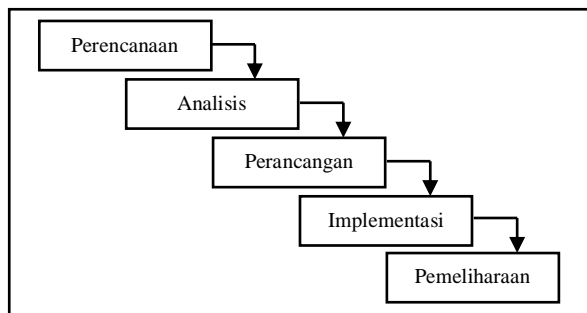
2.2.2 Rekayasa Web

Rekayasa Web merupakan pembangunan dengan menggunakan prinsip atau konsep rekayasa dengan tujuan menghasilkan perangkat lunak yang bernilai ekonomi yang dipercaya dan bekerja secara efisien menggunakan mesin.

Berikut adalah metode dan *tools* yang digunakan :

1. Metode[10]

Metode dapat diartikan sebagai menyediakan cara bagaimana secara teknis membangun perangkat lunak yang harus berada pada sebuah komitmen dasar menuju kualitas. Metode yang digunakan pada penelitian ini adalah metode *waterfall*. Metode *waterfall* adalah model klasik yang bersifat sistematis, berurutan dalam membangun software. Model ini sering disebut juga dengan *classic life cycle* atau metode *waterfall*. Disebut dengan *waterfall* karena tahap yang dilalui harus menunggu selesainya tahap sebelumnya dan berjalan berurutan.



Gambar 2. 1 Pemodelan *Waterfall*

Adapun tahap-tahapnya yaitu :

- a. Analisa Kebutuhan Perangkat Lunak
Proses pengumpulan kebutuhan dilakukan secara intensif untuk mespesifikasikan kebutuhan perangkat lunak agar dapat dipahami perangkat lunak seperti apa yang dibutuhkan oleh *user*.
- b. Desain atau Perancangan Sistem
Proses perancangan sistem membagi persyaratan dalam perancangan sistem perangkat keras dan perangkat lunak. Pada tahap ini, hasil analisa kebutuhan perangkat lunak pada sistem ini akan dideskripsikan ke dalam beberapa diagram, antara lain *use case diagram*, *sequence diagram*, dan ERD.
- c. Penulisan Kode Program
Desain atau perancangan sistem harus ditranslasikan ke dalam program perangkat lunak. Pada tahap ini, hasil desain diimplementasikan ke dalam kode atau bahasa yang dimengerti oleh komputer agar program komputer sesuai dengan hasil desain yang telah dibuat.
- d. Pengujian
Pengujian fokus pada perangkat lunak secara dari segi *logic* dan fungsional dan memastikan bahwa semua bagian sudah diuji. Hal ini dilakukan untuk meminimalisir kesalahan (*error*) dan memastikan keluaran yang dihasilkan sesuai dengan yang diinginkan.
- e. Pemeliharaan
Pemeliharaan mencakup koreksi dari tahap sebelumnya, perbaikan, atau implementasi dari persyaratan-persyaratan baru yang ditambahkan, tetapi tidak untuk membuat program perangkat lunak yang baru.

Terkait dengan pengujian program, pengujian yang akan digunakan adalah pengujian *black-box*. Pengujian *black-box* berfokus pada persyaratan fungsional perangkat lunak. Dengan demikian, pengujian *black-box* memungkinkan rekayasa perangkat lunak mendapatkan serangkaian kondisi *input* yang sepenuhnya menggunakan semua persyaratan fungsional untuk suatu program. Pengujian *black-box* bukan merupakan alternatif dari teknik *white-box*, tetapi merupakan pendekatan komplementer yang memungkinkan besar mampu mengungkap kelas

kesalahan daripada metode *white-box*. Pengujian *black-box* berusaha menemukan kesalahan dalam kategori sebagai berikut[11] :

1. Fungsi-fungsi yang tidak benar atau hilang,
2. Kesalahan *interface*,
3. Kesalahan dalam struktur data atau akses *database* eksternal,
4. Kesalahan kinerja,
5. Inisialisasi dan kesalahan terminal.

Tidak seperti pengujian *white-box*, yang dilakukan pada saat awal proses pengujian, pengujian *black-box* cenderung diaplikasikan selama tahap akhir pengujian. Karena pengujian *black-box* memperhatikan struktur kontrol, maka perhatian berfokus pada domain informasi.

Pengujian di desain untuk menjawab pertanyaan-pertanyaan berikut[12].

1. Bagaimana validitas fungsional diuji?
2. Kelas *input* apa yang akan *test case* menjadi baik?
3. Apakah sistem sangat sensitif terhadap harga *input* tertentu?
4. Bagaimana batasan dari suatu data diisolasi?
5. Kecepatan data apa dan volume data apa yang dapat ditolelir oleh sistem?
6. Apa pengaruh kombinasi tertentu dari data terhadap operasi sistem.

Pengujian *black-box* ini terdapat beberapa proses. Proses-proses yang ada dalam pengujian ini diantaranya :

- a. Menganalisa kebutuhan dan spesifikasi dari perangkat lunak.
 - b. Pemilihan jenis *input* yang memungkinkan menghasilkan *output* dengan benar serta jenis *input* yang memungkinkan *output* salah pada perangkat lunak yang sedang diuji.
 - c. Menentukan *output* untuk satu jenis *input*.
 - d. Pengujian dilakukan dengan *input-input* yang telah benar-benar diseleksi.
 - e. Melakukan pengujian
 - f. Perbandingan *output* yang dihasilkan dengan *output* yang diharapkan.
- f. Pemeliharaan (*maintenance*)
Tidak menutup kemungkinan sebuah perangkat lunak mengalami perubahan ketika sudah dikirimkan ke *user*. Pada tahap ini dapat

mengulangi proses pengembangan mulai dari analisis spesifikasi untuk perubahan perangkat lunak yang sudah ada, tapi tidak untuk membuat perangkat lunak baru.

2. Tools/ Alat Bantu Penelitian

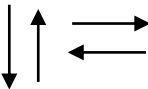


A. Flowchart

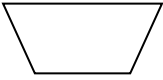



Flowchart adalah bagan (*chart*) yang menunjukkan alir (*flow*) di dalam program atau prosedur sistem secara logika. Bagan alir (*flowchart*) digunakan terutama untuk alat bantu komunikasi dan untuk dokumentasi.[13] Ada beberapa jenis-jenis *flowchart* diantaranya :

1. Bagan alir sistem (*system flowchart*)
2. Bagan alir dokumen (*document flowchart*)
3. Bagan alir skematik (*schematic flowchart*)
4. Bagan alir program (*program flowchart*)
5. Bagan alir proses (*process flowchart*)

Simbol-simbol dalam *flowchart* dapat dilihat pada Tabel 2.1.

Tabel 2. 1 Simbol *Flowchart*

| No | Simbol | Keterangan |
|----|---|---|
| 1 |  <p>Flow Direction Symbol</p> | Yaitu simbol yang digunakan untuk menghubungkan antara simbol yang satu dengan simbol yang lain. Simbol ini disebut juga <i>connecting line</i> . |
| 2 |  <p>Terminator Symbol</p> | Yaitu simbol untuk permulaan (<i>start</i>) atau akhir (<i>end</i>) dari suatu kegiatan. |
| 3 |  <p>Processing Symbol</p> | Yaitu simbol yang menunjukkan pengolahan yang dilakukan oleh komputer. |

| | | |
|---|---|--|
| 4 |  <p>Manual Operation Symbol</p> | Yaitu simbol yang menunjukkan pengolahan data yang tidak dilakukan oleh komputer. |
| 5 |  <p>Decision Symbol</p> | Yaitu simbol untuk pemilihan proses berdasarkan kondisi yang ada. |
| 5 |  <p>Input-Output Symbol</p> | Yaitu simbol yang menyatakan proses input dan output tanpa tergantung dari jenis peralatannya. |
| 6 |  <p>Document Symbol</p> | Yaitu simbol yang menyatakan input berasal dari dokumen dalam bentuk kertas atau output dicetak ke kertas. |

B. *Unified Modeling Language (UML)*

Unified Modeling Language (UML) adalah hasil kerja dari konsorium berbagai organisasi yang berhasil dijadikan sebagai standar baku dalam *Object Oriented Analysis* dan *Design (OOAD)*. Macam-macam dari *Unified Modeling Language (UML)* antara lain [11] : *use case diagram*, *sequence diagram* dan *class diagram*.







1) *Use Case*



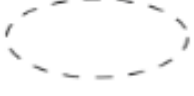

Use Case adalah deskripsi fungsi dari sebuah sistem dari perspektif pengguna. *Use Case* bekerja dengan cara mendeskripsikan tipikal interaksi antara *user* (pengguna) sebuah sistem dengan sistemnya sendiri melalui sebuah cerita bagaimana sistem dipakai. Urutan langkah-langkah yang menerangkan antara pengguna dan sistem tersebut *scenario* sedangkan pengguna disebut *actor*. *Actor* adalah sebuah peran yang biasa dimainkan oleh pengguna dalam interaksinya dengan sistem. Model *use case* adalah bagian dari model

requirement. Definisi lain *use case* adalah abstraksi dari interaksi antara sistem dan *actor*. *Use case* dibuat berdasarkan keperluan *actor*.

Berdasarkan definisi diatas maka dapat disimpulkan bahwa *Use Case* adalah kontruks untuk mendeskripsikan bagaimana sistem akan terlihat dimata pengguna potensial yang terdiri dari sekumpulan *scenario* dan *actor*. Sedangkan *use case* diagram memfasilitasi komunikasi diantara analis dan pengguna serta analis dan klien. Simbol *Use Case* dapat dilihat pada Tabel 2.2.

Tabel 2. 2 Simbol *Use Case Diagram*

| No. | Simbol | Nama Simbol dan Keterangan |
|-----|---|---|
| 1. |  | <i>Actor</i> , Menspesifikasikan himpunan peran yang pengguna mainkan ketika berinteraksi dengan <i>use case</i> . |
| 2. |  | <i>Dependency</i> , Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri (independent) akan mempengaruhi elemen yang bergantung padanya elemen yang tidak mandiri(independent). |
| 3. |  | <i>Generalization</i> , Hubungan dimana objek anak (descendent) berbagi perilaku dan struktur data dari objek yang ada di atasnya objek induk (ancestor). |
| 4. |  | <i>Include</i> , Menspesifikasikan bahwa <i>use case</i> sumber secara eksplisit. |
| 5. |  | <i>Extend</i> , Menspesifikasikan bahwa <i>use case</i> target memperluas perilaku dari <i>use case</i> sumber pada suatu titik yang diberikan. |
| 6. |  | <i>Association</i> , yang menghubungkan |






| | | |
|-----|---|--|
| | | antara objek satu dengan objek lainnya |
| 7. |  | <i>System</i> , Menspesifikasikan paket yang menampilkan sistem secara terbatas |
| 8. |  | <i>Use Case</i> , Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu actor. |
| 9. |  | <i>Collaboration</i> , Interaksi aturan-aturan dan elemen lain yang bekerjasama untuk menyediakan perilaku yang lebih besar dari jumlah dan elemennya (sinergi). |
| 10. |  | <i>Note</i> , Elemen fisik yang eksis saat aplikasi dijalankan dan mencerminkan suatu sumber daya komputasi |

2) *Sequence Diagram*

Sequence Diagram digunakan untuk menggambarkan perilaku pada sebuah *scenario*. Diagram ini menunjukkan sejumlah obyek dan *message* yang diletakan antara obyek-obyek didalam *use case*.

Komponen utama *sequence diagram* terdiri atas objek yang dituliskan dengan kotak segiempat bernama. *Message* diwakili oleh garis dengan tanda panah dan waktu yang ditunjukkan dengan *progress vertical*. *Sequence Diagram* menambahkan dimensi waktu pada interaksi diantara obyek. Simbol-simbol yang dipakai dalam pembuatan *sequence diagram* dapat dilihat pada Tabel 2.3.

Tabel 2. 3 Simbol *Sequence Diagram*

| No | Gambar | Nama | Keterangan |
|----|---|--|--|
| 1. |  | <i>Actor</i> | Menggambarkan orang yang sedang berinteraksi dengan sistem |
| 2. |  | <i>Entity Class</i> | Menggambarkan hubungan yang akan dilakukan |
| 3. |  | <i>Boundary Class</i> | Menggambarkan sebuah gambaran dari foem |
| 4. |  | <i>Control Class</i> | Menggambarkan penghubung antara boundary dengan tabel |
| 5. | <i>A focus of Control & A Life Line</i> | Menggambarkan tempat mulai dan berakhirnya message | |
| 6. |  | <i>A message</i> | Menggambarkan pengiriman pesan |





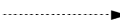

3) *Class Diagram*

Class Diagram menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. Kelas memiliki apa yang disebut atribut dan metode atau operasi. Atribut merupakan variable-variabel yang dimiliki oleh suatu kelas. Sedangkan operasi atau metode adalah fungsi-fungsi yang dimiliki oleh suatu kelas[14]. Diagram kelas dibuat agar *programmer* membuat kelas-kelas sesuai rancangan di dalam diagram kelas agar antara dokumentasi perancangan dan perangkat lunak sinkron.

Class diagram membantu dalam visualisasi struktur kelas – kelas dari suatu sistem dan merupakan tipe diagram yang paling banyak. Class diagram memperlihatkan hubungan antar kelas dan penjelasan detail tiap – tiap kelas di dalam model desain (dalam logical view) dari suatu sistem. Selama proses analisis, class diagram memperlihatkan aturan – aturan dan tanggung jawab entitas yang menentukan perilaku sistem. Selama proses analisis, *class diagram* memperlihatkan aturan – aturan dan tanggung jawab entitas yang menentukan perilaku sistem. Selama tahap decian, class diagram berperan dalam menangkap struktur dari semua kelas yang membentuk arsitektur sistem yang dibuat[15].

Berikut beberapa simbol dari *class diagram* :

Tabel 2. 4 Simbol *Class Diagram*

| No. | Simbol | Nama | Keterangan |
|-----|---|--|--|
| 1. |  | Kelas | Kelas pada struktur sistem. |
| 2. |  | Asosiasi/ <i>Association</i> | Relasi antar kelas dengan makna umum, asosiasi biasanya juga disertai dengan <i>multiplicity</i> . |
| 3. |  | Asosiasi berarah/ <i>Directed association</i> | Relasi antar kelas dengan makna kelas yang satu digunakan oleh kelas yang lain. |
| 4. |  | Generalisasi | Relasi antarkelas dengan makna generalisasi-spesialisasi (umum khusus). |
| 5. |  | Kebergantungan <i>Dependency</i> | Relasi antarkelas dengan makna kebergantungan antarkelas. |
| 6. |  | Agregasi / <i>Aggregation</i> | Relasi antarkelas dengan makna semua-bagian (<i>whole-part</i>). |

2.2.3 Pemrograman Berorientasi Objek

Metodologi berorientasi objek adalah suatu strategi pembangunan perangkat lunak yang mengorganisasikan perangkat lunak sebagai kumpulan projek yang berisi data dan operasi yang diperlakukan terhadapnya. Metodologi berorientasi objek merupakan suatu cara bagaimana sistem perangkat lunak dibangun melalui pendekatan objek secara sistematis[14].

Sistem berorientasi objek merupakan sebuah sistem yang dibangun dengan berdasarkan metode berorientasi objek adalah sebuah sistem yang komponennya dibungkus menjadi kelompok data dan fungsi. Setiap komponen dalam sistem tersebut dapat mewarisi atribut, sifat, komponen lainnya, dan dapat berinteraksi satu sama lain.

2.2.4 Basis Data

Basis data adalah kumpulan file-file yang mempunyai kaitan antara satu file dengan file lain sehingga membentuk satu bangun data untuk menginformasikan suatu perusahaan instansi, dalam bahasan tertentu.

MySQL adalah sebuah program database server yang mampu menerima dan mengirimkan datanya dengan sangat cepat, multi *user* serta menggunakan perintah standar *Structured Query Language* (SQL). MySQL memiliki dua bentuk lisensi, yaitu *Free Software* dan *Shareware*. MySQL yang biasa kita gunakan adalah *MySQL Free Software I* yang berada dibawah Lisensi *General Public License* (GPL)[16].

MySQL juga dapat didefinisikan sebagai sebuah database server, dapat juga berpersion sebagai *client* sehingga sering disebut *database client / server* yang *open source* dengan kemampuan dapat berjalan baik di Operasi Sistem maupun dengan *Platform Windows* maupun *Linux*.

MySQL dikembangkan oleh sebuah perusahaan Swedia bernama MySQL AB, yang pada saat itu bernama TcX Data Konsult AB sekitar tahun 1994-1995. MySQL sudah ada sejak 1979. MySQL termasuk jenis *Relation Database Management System* (RDBMS) digunakan oleh banyak portal-portal internet sebagai basis data dari informasi yang ditampilkan pada situs *web*. Kepopuleran MySQL dimungkinkan karena kemudahannya untuk digunakan, cepat secara kinerja *query*, dan mencukupi untuk kebutuhan basis data perusahaan-perusahaan skala

menengah dan kecil. Istilah seperti tabel, baris, dan kolom tetap digunakan dalam MySQL. Sebuah basis data yang terdapat pada MySQL mengandung satu atau beberapa tabel yang terdiri dari sejumlah baris dan kolom.

MySQL sebenarnya merupakan turunan salah satu konsep utama dalam database sejak lama, yaitu SQL (*Structured Query Language*). SQL adalah suatu bahasa (*language*) yang digunakan untuk mengakses data di dalam sebuah database relasional. SQL sering juga disebut dengan istilah query, dan bahasa SQL secara praktiknya digunakan sebagai bahasa standar untuk manajemen database relasional. Hingga saat ini hampir seluruh server database atau software database mengenal dan mengerti bahasa SQL. Dalam penggunaan SQL terdapat beberapa perintah yang berguna untuk mengakses dan memanajemen data yang terdapat dalam database. Secara garis besar, SQL Server mempunyai 3 (Tiga) jenis perintah SQL yaitu :

1.) **Data Definition Language (DDL)**

DDL adalah sub perintah dari bahasa SQL yang digunakan untuk membangun kerangka sebuah database, dalam hal ini database dan Tabel. Terdapat tiga perintah penting dalam DDL, yaitu :

- a.) CREATE: perintah ini digunakan untuk membuat, termasuk di dalamnya membuat database baru, tabel baru view baru, dan kolom baru. Contoh: CREATE DATABASE nama_database.
- b.) ALTER: perintah ALTER berfungsi untuk mengubah struktur tabel yang telah dibuat. Mencakup di dalamnya mengubah nama tabel, menambah kolom, mengubah kolom, menghapus kolom, dan memberikan atribut pada kolom. Contoh: ALTER Tabel nama_tabel ADD nama_kolom datatype.
- c.) DROP: perintah DROP berfungsi untuk menghapus database atau tabel. Contoh: DROP DATABASE nama_database.

2) **Data Manipulation Language (DML)**

DML adalah sub perintah dari bahasa SQL yang digunakan untuk memanipulasi data dalam database yang telah dibuat. Terdapat 4 (Empat) perintah penting dalam DML, yaitu :

- a.) INSERT: perintah ini digunakan untuk memasukkan data baru ke dalam sebuah tabel. Perintah ini tentu saja bisa dijalankan ketika database dan tabel sudah dibuat. Contoh: INSERT INTO nama_tabel VALUES (data1, data2, dst...);

- b.) **SELECT:** perintah ini digunakan untuk mengambil dan menampilkan data dari tabel atau bahkan dari beberapa tabel dengan penggunaan relasi. Contoh: `SELECT nama_kolom1, nama_kolom2 FROM nama_tabel;`
- c.) **UPDATE:** perintah update digunakan untuk memperbaharui data pada sebuah tabel. Contoh: `UPDATE nama_tabel SET kolom1=data1, kolom2=data2,... WHERE kolom=data;`
- d.) **DELETE:** perintah delete digunakan untuk menghapus data dari sebuah tabel. Contoh: `DELETE FROM nama_tabel WHERE kolom=data;`

3.) **Data Control Language (DCL)**

DCL adalah sub bahasa SQL yang berfungsi untuk melakukan pengontrolan data dan server databasenya, seperti manipulasi user dan hak akses (priviledges). Yang termasuk perintah dalam DCL ada 2 (Dua), yaitu :

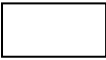

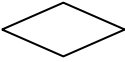
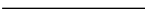
- a.) **GRANT:** perintah ini digunakan untuk memberikan hak akses oleh admin ke salah satu user atau pengguna. Hak akses tersebut bisa berupa hak membuat (**CREATE**), mengambil data (**SELECT**), menghapus data (**DELETE**), mengubah data (**UPDATE**), dan hak khusus lainnya yang berhubungan dengan sistem database.
- b.) **REVOKE:** perintah ini digunakan untuk mencabut hak akses yang telah diberikan kepada user. Dalam ini merupakan kebalikan dari perintah **GRANT**. [14]

4.) **Entity Relationship Diagram (ERD)**

ERD adalah suatu rancangan atau bentuk hubungan suatu kegiatan di dalam sistem yang berkaitan langsung dan mempunyai fungsi di dalam proses tersebut. ERD adalah suatu pemodelan dari basis data relasional yang didasarkan atas persepsi di dalam dunia nyata, dunia ini senantiasa terdiri dari sekumpulan objek yang saling berhubungan antara satu dengan yang lainnya. Suatu objek disebut *entity* dan hubpeungan yang dimilikinya disebut *relationship*. Suatu *entity* bersifat unik dan memiliki atribut sebagai pembeda dengan entity lainnya. [17]

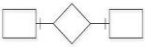
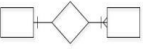
Berikut merupakan simbol-simbol dari ERD:


Tabel 2. 5 Simbol *Entity Relationship Diagram*

| No. | Nama | Simbol | Keterangan Fungsi |
|-----|---------|---|--|
| 1. | Entitas |  | Persegi panjang menyatakan himpunan entitas adalah orang, kejadian, atau berada. |
| 2. | Atribut |  | Atribut merupakan informasi yang diambil tentang sebuah entitas. |
| 3. | Relasi |  | Belah ketupat menyatakan himpunan relasi merupakan hubungan antar entitas. |
| 4. | Link |  | Garis sebagai penghubung antara himpunan, relasi, dan himpunan entitas dengan atributnya |

ERD memiliki derajat relasi atau biasa disebut kardinalitas. Kardinalitas menjelaskan batasan jumlah keterhubungan satu entity dengan entity lainnya.

Tabel 2. 6 Macam-Macam Kardinalitas

| No. | Simbol | Nama | Keterangan |
|-----|---|--|---|
| 1. |  | Relasi Satu ke Satu (<i>One to One</i>) | Relasi yang menunjukkan bahwa setiap himpunan entitas berhubungan dengan tepat satu himpunan entitas lainnya |
| 2. |  | Relasi Satu ke Banyak (<i>One to Many</i>) | Relasi yang menunjukkan bahwa hubungan antara entitas pertama dengan entitas kedua adalah satu banding banyak, begitu pula sebaliknya |

| | | | |
|----|---|--|--|
| 3. |  | Relasi Banyak ke Banyak <i>(Banyak to Many)</i> | Relasi yang menunjukkan bahwa setiap himpunan entitas boleh berhubungan dengan banyak himpunan entitas lainnya dan sebaiknya |
|----|---|--|--|

2.2.5 Percetakan

Percetakan adalah sebuah proses industri untuk memproduksi massal tulisan dan gambar, terutama dengan tinta di atas kertas menggunakan sebuah mesin cetak. Percetakan merupakan bagian penting dalam penerbitan dan percetakan transaksi. Teknik percetakan umum lainnya termasuk cetak relief, sablon, rotogravure, dan percetakan berbasis digital seperti pita jarum, inkjet, dan laser. Dikenal pula teknik cetak poly untuk pemberian kesan emas dan perak ke atas permukaan dan cetak emboss untuk memberikan kesan menonjol kepada kertas[18].

2.2.6 Jasa

Jasa adalah setiap tindakan atau perbuatan yang dapat ditawarkan oleh satu pihak kepada pihak lain yang pada dasarnya tidak berwujud *intangible* dan tidak menghasilkan kepemilikan apapun[19].

2.2.7 Pemesanan

Pemesanan adalah Proses pembelian suatu barang atau jasa yang dilakukan oleh konsumen kepada penjual sebelum konsumen mendapatkan barang. Langkah pemesanan yang paling sederhana adalah dengan melakukan kontak langsung kepada penjual kemudian konsumen memesan barang atau jasa yang diinginkan. Pemesanan barang atau jasa saat ini bisa dilakukan dengan berbagai cara, baik secara lisan maupun dengan dunia maya[5].