BAB II
LANDASAN TEORI

2.1 Tinjauan Pustaka

Tinjauan pustaka yang berupa penelitian-penelitian terdahulu yang menjadi dasar penelitian ini, antara lain sebagai berikut :

Penelitan dengan judul Sistem Informasi Kartu Rencana Studi Dan Kartu Hasil Studi *Online* membahas proses pengisian KRS dan pada akhir semester melakukan proses pengisian KHS dan Transkrip dan membuat laporan KHS dan Transkrip yang diberikan kepada mahasiswa. Proses pengisian KRS, KHS dan Transkrip masih semi manual sehingga menyebabkan proses pengisian KRS, KHS dan Transkrip masih lambat, sehingga dibuatlah sebuah sistem informasi KRS dan KHS *online*, menggunakan PHP dan MySQL (Fahrudin & Purwanto, 2012).

Penelitian dengan judul Pembuatan Sistem Informasi Kartu Rencana Studi (KRS) Dan Kartu Hasil Studi (KHS) pada Program Studi Informatika Universitas Surakarta membahas dalam pengisian KRS masih bersifat konvensional, diantaranya mahasiswa harus datang ke kampus untuk mengambil formulir KRS dan mengisinya secara manual (tulis tangan), maka dari itu dibuatlah suatu sistem informasi KRS dan KHS yang dibangun dengan menggunakan bahasa pemrograman PHP dan menggunakan basis data MySQL (Nuraini, Purnama, & T, 2012).

Penelitian dengan judul Sistem Informasi Akademik Dengan Framework *Codeigniter* (Studi Kasus : SMP Negeri 1 Teras Boyolali) membahas informasi, pada SMP Negeri 1 Teras Boyolali yang disampaikan lambat dan kurang efektif sehingga memerlukan sebuah sistem informasi akademik guna memberikan kontribusi yang bermanfaat bagi semua pihak khususnya dalam mendukung kemajuan dibidang akademik (Rohmat, 2016).

2.2 Konsep Dasar Sistem

Pengertian sistem menurut para ahli adalah sebagai berikut :

Sistem memiliki arti kumpulan dari berbagai komponen yang memiliki suatu keterkaitan antara satu dengan lainnya (Indrajit, 2001).

Sistem adalah kumpulan dari berbagai elemen yang melakukan interaksi untuk mencapai tujuan tertentu, hal ini menggambarkan kejadian-kejadian dan kesatuan dan objek yang nyata, seperti tempat, benda, dan orang-orang yang benar-benar ada dan terjadi (Jogiyanto, 2005).

Suatu sistem adalah seperangkat elemen yang membentuk suatu kumpulan atau berbagai prosedur/bagan-bagan pengolahan yang bertujuan untuk mencapai sesuatu (Murdick & dkk, 1991).

Kesimpulan dapat ditarik berdasarkan definisi-definisi tentang pengertian dari sistem menurut para ahli di atas bahwa sebuah sistem adalah hubungan-hubungan atau jaringan kerja dari berbagai prosedur atau objek-objek tertentu yang berkumpul bersama untuk mencapai suatu tujuan tertentu (Hutahaean, 2015).

2.3 *Web Server* Apache

World Wide Web (WWW), atau *web* adalah sebuah layanan internet terdiri atas kumpulan dokumen elektronik dari berbagai sumber. Setiap dokumen elektronik dalam *web* disebut halaman *web*. Halaman *web* dapat menyimpan data berupa teks, gambar, audio maupun video. Halaman *web* biasanya tersambung satu dengan lainnya dengan sarana *link*. Halaman *web* disimpan dalam suatu situs *web* dengan sarana *web server*. *Web server* adalah seperangkat komputer yang melayani permintaan halaman *web* dan mengirimkan ke *web browser* yang melakukan permintaan sebuah halaman *web* (Vermaat, Shelly, & Cashman, 2007).

Apache adalah *web server* yang dapat dijalankan di berbagai sistem operasi (Unix, BSD, Linux, Microsoft Windows dan Novell Netware serta platform lainnya) yang berguna untuk membuat layanan komputer berupa *web*. Protokol yang digunakan untuk melayani fasilitas *web* ini adalah *Hyper Text Transfer Protocol* (HTTP). Apache dikembangkan oleh *The Apache Software Foundation* dengan nama Apache HTTP *Server Project.*

Apache HTTP *Server Project* adalah sebuah pengembangan perangkat lunak secara kolaboratif yang ditujukan untuk implementasi kode sumber HTTP (*Web*)

yang kuat, komersial, penuh fitur, dan tersedia secara bebas. Proyek ini dikelola bersama oleh sekelompok sukarelawan yang berada di seluruh dunia, menggunakan Internet dan *Web* untuk berkomunikasi, merencanakan, mengembangkan sistem dan dokumentasi yang terkait. Proyek ini merupakan bagian dari Apache *Software Foundation*. Apache adalah komponen *server web* dari paket perangkat lunak LAMP (Linux, Apache, MySQL, PHP/Perl/bahasa pemrograman Python) (The Apache Software Foundation., 2017).

2.4 Basis Data Mysql

Basis data adalah kumpulan catatan atau data terstruktur yang tersimpan dalam sistem komputer dan diatur sedemikian rupa sehingga dapat dengan cepat dicari dan diproses sebagai informasi. MySQL singkatan dari *Structured Query Language*. SQL berdasarkan bahasa Inggris dan juga digunakan dalam basis data lain seperti Oracle dan Microsoft SQL *Server*. SQL didesain untuk dapat mengirimkan permintaan data dari basis data menggunakan *commands* seperti :

SELECT title FROM publications WHERE author = 'Charles Dickens';

Sebuah basis data MySQL berisi satu tabel atau lebih, yang masing-masing berisi *record* atau *row*. R*ows* disebut sebagai kolom atau *fields* yang berisi data itu sendiri. Salah satu contoh tabel dalam basis data MySQL yang menunjukkan data publikasi yang berisi data *author, title, type,* dan tahun publikasi, ditunjukkan pada Tabel 2.1.

Tabel 2.1. Contoh Basis Data Sederhana

Author	**Title**	**Type**	**Year**
Mark Twain	The Adventures of Tom Sawyer	Fiction	1876
Jane Austen	Pride and Prejudice	Fiction	1811
Charles Darwin	The Origin of Species	Nonfiction	1856
Charles Dickens	The Old Curiosity Shop	Fiction	1841
William Shakespeare	Romeo and Juliet	Play	1594

Setiap baris pada Tabel 2.1. adalah sama dengan baris pada basis data MySQL. Setiap elemen pada sebuah baris adalah sama dengan *field* pada basis data MySQL (Nixon, 2014).

2.4.1 Desain Basis Data

Seorang desainer sistem membuat desain sebuah basis data dengan tepat sebelum membuatnnya merupakan hal yang sangat penting untuk dilakukan. Jika hal tersebut tidak dilakukan, maka pada akhirnya banyak terjadi revisi basis data seperti memisahkan tabel, menggabungkan tabel dan hal-hal lain agar tercapai hubungan basis data yang optimal. Seorang desainer sistem membuat sebuah basis data dapat dimulai dengan menuliskan *query* apa saja yang sering digunakan oleh pengguna, sebagai contoh basis data toko buku *online*, beberapa pertanyaan yang bisa dijadikan sebagai acuan adalah sebagai berikut :

1. Berapa banyak penulis, buku, dan pelanggan yang ada di basis data?
2. Penulis mana yang menulis buku tertentu?
3. Buku mana yang ditulis oleh penulis tertentu?
4. Buku apa yang paling mahal?
5. Apa buku yang paling laris dijual?
6. Buku mana yang belum terjual tahun ini?
7. Buku mana yang dibeli pelanggan tertentu?
8. Buku mana yang telah dibeli bersama dengan buku-buku lain yang sama?

Contoh *query-query* yang lain dapat dibuat pada basis data semacam itu, meskipun hanya dengan contoh kecil tersebut, akan dapat memberikan wawasan tentang bagaimana menyusun tabel-tabel pada basis data. Buku dan ISBN mungkin bisa digabungkan menjadi satu tabel, karena mempunyai keterkaitan yang erat, misalnya. Buku dan pelanggan harus berada dalam tabel terpisah, karena memiliki hubungan yang sangat longgar, sebaliknya. Seorang pelanggan bisa membeli buku apa pun, dan bahkan lebih dari satu buku yang sama, namun buku bisa jadi dibeli oleh banyak pelanggan, dan sebagainya (Nixon, 2014).

2.4.2 *Primary Key*

Seorang desainer sistem dapat mendefinisikan informasi dari setiap *author*, buku, dan pelanggan dalam satu tempat dengan menggunakan fungsi utama basis data relasional. Hal-hal yang menjadi perhatian utama adalah hubungan di antaranya, seperti siapa yang menulis buku tertentu dan siapa yang membeli, kita dapat meletakkan informasi tersebut dengan membuat hubungan di antara tiga tabel tersebut (Nixon, 2014).

Setiap tabel harus diberikan sebuah *primary key* yang unik terhadap setiap isi tabel, tidak boleh memilih *key* yang mungkin memiliki nilai yang sama ke objek yang lain, sebagai contoh kita bisa menggunakan *field* ISBN untuk tabel buku, karena *field* ISBN tidak mungkin sama antara buku satu dengan lainnya. Entitas buku memiliki ISBN, maka dari itu dibuatlah *arbitrary primary key* dengan fitur *auto_increment* (Nixon, 2014).

2.4.3 Normalisasi

Proses memisahkan data ke dalam tabel dan membuat *primary key* disebut normalisasi. Tujuan utama proses normalisasi adalah memastikan setiap informasi muncul dalam basis data hanya sekali. Data duplikat membuat basis data tidak efisien, karena akan menjadikan basis data lebih besar dari yang memang diperlukan, dan akan memperlambat akses. Data duplikat meningkatkan risiko bahwa hanya akan memperbarui satu baris data duplikat, yang menyebabkan terjadinya ketidakkonsistensi dalam basis data dan berpotensi menyebabkan kesalahan serius. Seorang desainer sistem dalam membagi basis data menjadi beberapa tabel, penting untuk terlalu jauh dan membuat lebih banyak tabel daripada yang diperlukan, yang juga akan desain tidak efisien dan akses data yang lebih lambat (Nixon, 2014).

E. F. Codd, sang penemu model relasional, menganalisis konsep normalisasi dan membaginya menjadi tiga skema terpisah yang disebut bentuk normal pertama, bentuk normal kedua dan bentuk normal ketiga, jika membuat suatu basis data dan memenuhi kriteria setiap bentuk normal secara urut, maka basis

data akan teroptimasi maksimal untuk akses cepat, penggunaan *memory* minimal dan tempat penyimpanan.

Seorang sistem desainer dapat mengetahui bagaimana proses normalisasi dilakukan, proses pertama bisa dilihat pada Tabel 2.2 yang menunjukkan desain basis data yang kurang efisien. Desain basis data ini merupakan satu tabel yang berisi semua data yaitu nama *author,* judul buku dan detail data *customer*. Desain ini dapat dianggap sebagai langkah awal membuat sebuah tabel yang bertujuan untuk mencatat tiap *customer* yang telah membeli buku. Hal ini merupakan desain yang tidak efisien, karena data terduplikasi di semua tempat (duplikasi data) (Nixon, 2014).

2.4.4 Bentuk Normal Pertama

Basis data yang memenuhi syarat bentuk normal pertama, maka basis data harus memenuhi tiga persyaratan :

1. tidak boleh ada kolom yang berulang yang berisi data yang sama,
2. semua kolom harus berisi nilai tunggal dan,
3. terdapat *primary key* untuk mengidentifikasi tiap baris data.

Kolom *author* 1 dan *author* 2 berisi data dengan tipe data yang sama, berdasarkan persyaratan di atas secara urut. Kedua kolom tersebut dapat dipisah ke tabel yang berbeda, karena tidak sesuai dengan syarat nomor 1. Hal yang kedua, terdapat tiga *author* pada buku terakhir dengan judul *Programming PHP, m*aka dari itu detail *author* dapat dipisahkan ke dalam tabel yang berbeda. Syarat ketiga sudah terpenuhi, karena sudah terdapat kolom ISBN yang dapat digunakan sebagai *primary key,* dapat dilihat pada Tabel 2.3 dan Tabel 2.4

Sebuah *key,* juga disebut dengan *index*, mempunyai beberapa fungsi pada MySQL. Sebuah *key* berfungsi untuk membuat operasi pencarian lebih cepat, secara fundamental. *Key* juga berfungsi untuk memberikan identitas unik pada setiap data, disebut juga sebagai *primary key, d*an sebagai *foreign key* yang berfungsi untuk penghubung data pada tabel ke tabel yang lain (Nixon, 2014).

Tabel 2.2. Desain Basis Data yang Kurang Efisien

***Author* 1**	David Sklar	Danny Goodman	Hugh E Williams	David Sklar	Rasmus Lerdorf
***Author* 2**	Adam Trachtenberg		David Lane	Adam Trachtenberg	Kevin Tatroe & Peter MacIntyre
Title	PHP Cookbook	Dynamic HTML	PHP And MySQL	PHP Cookbook	Programming PHP
ISBN	596101015	596527403	596005436	596101015	596006815
Price	44.99	59.99	44.95	44.99	39.99
***Customer* Name**	Emma Brown	Darren Ryder	Earl B.Thurston	Darren Ryder	David Miller
***Customer* Address**	1565 Rainbow Road, Los Angeles, CA 90014	4758 Emily Drive, Richmond, VA 23219	862 Gregory Lane, Frankfort, KY 40601	4758 Emily Drive, Richmond, VA 23219	3647 Cedar Lane, Waltham, MA 02154
***Purchase* Date**	Mar 03 2009	Dec 19 2008	Jun 22 2009	Dec 19 2008	Jan 16 2009

Tabel 2.3. Tabel Hasil dari Memisahkan Kolom *Author* dari Tabel 2.2.

Title	**ISBN**	**Price**	***Customer* Name**	***Customer* Address**	***Purchase* Date**
PHP Cookbook	596101015	44.99	Emma Brown	1565 Rainbow Road, Los Angeles, CA 90014	Mar 03 2009
Dynamic HTML	596527403	59.99	Darren Ryder	4758 Emily Drive, Richmond, VA 23219	Dec 19 2008
PHP and MySQL	596005436	44.95	Earl B. Thurston	862 Gregory Lane, Frankfort, KY 40601	Jun 22 2009
PHP Cookbook	596101015	44.99	Darren Ryder	4758 Emily Drive, Richmond, VA 23219	Dec 19 2008
Programming PHP	596006815	39.99	David Miller	3647 Cedar Lane, Waltham, MA 02154	Jan 16 2009

Tabel 2.4. Tabel *Author*

ISBN	***Author***
596101015	David Sklar
596101015	Adam Trachtenberg
596527403	Danny Goodman
596005436	Hugh E Williams
596005436	David Lane
596006815	Rasmus Lerdorf
596006815	Kevin Tatroe
596006815	Peter MacIntyre

2.4.5 Bentuk Normal Kedua

Bentuk normal pertama berfungsi untuk mengatasi data redundan pada banyak kolom. Bentuk normal kedua berfungsi untuk menghilangkan data redundan pada berbagai baris. Basis data dapat mencapai bentuk normal kedua harus sudah berbentuk normal pertama terlebih dahulu. Bentuk normal kedua dapat dicapai dengan membuat dengan mengidentifikasi kolom yang mempunyai data yang berulang di tempat lain dan membuatkan tabel baru (Nixon, 2014).

Tabel 2.3 menunjukkan bahwa Darren Ryder membeli dua buku dan menyebabkan redundansi pada detail pembelian. Hal ini menjelaskan tabel tersebut bisa dipisahkan menjadi tabel tersendiri. Hasil pemisahan tersebut ditunjukkan pada Tabel 2.5 (Nixon, 2014).

Tabel 2.5. Tabel Baru Dengan Nama Judul Buku

ISBN	**Title**	**Price**
0596101015	PHP Cookbook	44.99
0596527403	Dynamic HTML	59.99
0596005436	PHP and MySQL	44.95
0596006815	Programming PHP	39.99

Tabel 2.5 menunjukkan pemisahan tabel, sehingga yang tersisa hanyalah kolom ISBN, judul, dan kolom harga untuk setiap *entry* data buku. Kolom *customer*, setelah dipisahkan, seperti pada Tabel 2.5 terlihat bahwa masih ada

proses normalisasi yang dapat dilakukan, karena detail dari Derren Ryder masih terduplikasi, dan dapat juga dikatakan bahwa aturan pertama bentuk normal pertama belum diterapkan, karena alamat *customer* seharusnya dipisahkan ke dalam tabel tersendiri.

Tabel 2.6. Tabel Detail *Customer* Dari Tabel 2.3.

ISBN	***Customer* Name**	***Customer* Address**	***Purchase* Date**
0596101015	Emma Brown	1565 Rainbow Road, Los Angeles, CA 90014	Mar 03 2009
0596527403	Darren Ryder	4758 Emily Drive, Richmond, VA 23219	Dec 19 2008
0596005436	Earl B. Thurston	862 Gregory Lane, Frankfort, KY 40601	Jun 22 2009
0596101015	Darren Ryder	4758 Emily Drive, Richmond, VA 23219	Dec 19 2008
0596006815	David Miller	3647 Cedar Lane, Waltham, MA 02154	Jan 16 2009

Tabel 2.7 adalah hasil dari normalisasi tabel *customer* ke dalam bentuk normal pertama dan kedia. Setiap *customer* sekarang memiliki kode unik yang disebut dengan *CustNo* yang menjadi *primary key* yang dapat dibuat dengan *auto_increment.* Semua elemen dari alamat *customer* telah dipisahkan ke dalam kolom yang berbeda yang bisa dicari dan diupdate dengan cepat

Tabel 2.7. Tabel Baru Dengan Nama Tabel *Customer*

CustNo	**Name**	**Address**	**City**	**State**	**Zip**
1	Emma Brown	1565 Rainbow Road	Los Angeles	CA	90014
2	Darren Ryder	4758 Emily Drive	Richmond	VA	23219
3	Earl B. Thurston	862 Gregory Lane	Frankfort	KY	40601
4	David Miller	3647 Cedar Lane	Waltham	MA	02154

Tabel 2.6 agar dapat dilakukan normalisasi, maka informasi dari buku apa yang dibeli oleh *customer* harus dihapus, karena kalau tidak, akan terjadi kejadian perulangan informasi detail *customer*. Data pembelian, dengan alasan di atas, dipisahkan ke tabel baru dengan nama tabel *purchase* yang ditunjukkan pada Tabel 2.8.

Tabel 2.8. Tabel Baru Dengan Nama Tabel *Purchase*

CustNo	ISBN	Date
1	0596101015	Mar 03 2009
2	0596527403	Dec 19 2008
2	0596101015	Dec 19 2008
3	0596005436	Jun 22 2009
4	0596006815	Jan 16 2009

Tabel 2.8 menunjukkan kolom *CustNo* dari tabel Tabel 2.7 digunakan kembali sebagai *key* untuk *customer* dan data pembelian. Kolom ISBN juga berulang pada tabel ini, maka tabel ini dapat disambungkan dengan tabel *author* atau tabel judul buku.

Kolom *CustNo* dapat menjadi *key* yang berguna pada tabel *purchase*, tetapi bukan merupakan *primary key*. Seorang *customer* dapat membeli buku lebih dari satu. Jika *customer* membeli dua buku pada hari yang sama, dapat dilakukan dengan menjadikan kolom *CustNo* dan *ISBN* sebagai *key*, tetapi bukan sebagai *primary key* (Nixon, 2014).

2.4.6 Bentuk Normal Ketiga

Basis data jika sudah mencapai bentuk normal pertama dan kedua, maka basis data sudah dalam bentuk yang baik, dan mungkin tidak perlu dimodifikasi lebih lanjut. Bentuk normal ketiga dapat diterapkan, jika ingin sangat ketat dalam manajemen basis data, yang mana data yang tidak bergantung secara langsung dengan *primary key* tetapi bergantung pada elemen lain di dalam tabel harus dipindahkan ke tabel terpisah tergantung dengan dependensinya (Nixon, 2014).

Tabel 2.7 menunjukkan bahwa kolom *state, city*, dan *zip code*, tidak bergantung pada masing-masing *customer* karena banyak *customer* lain memiliki detail alamat yang sama, tetap, kolom-kolom tersebut mempunyai hubungan langsung satu sama lainnya, alamat jalan bergantung dengan kota, dan kota bergantung pada negara. Tabel 2.7 dapat dipisahkan ke dalam Tabel 2.10, Tabel 2.11, dan Tabel 2.12 untuk dapat memenuhi bentuk normal ketiga.

Tabel 2.9. Bentuk Normal Ketiga Tabel *Customer*

CustNo	Name	Address	Zip
1	Emma Brown	1565 Rainbow Road	90014
2	Darren Ryder	4758 Emily Drive	23219
3	Earl B. Thurston	862 Gregory Lane	40601
4	David Miller	3647 Cedar Lane	02154

Tabel 2.10 Bentuk Normal Ketiga Tabel *ZipCode*

Zip	CityID
90014	1234
23219	5678
40601	4321
02154	8765

Tabel 2.11 Bentuk Normal Ketiga Tabel *Cities*

CityID	Name	StateID
1234	Los Angeles	5
5678	Richmond	46
4321	Frankfort	17

Tabel 2.12 Bentuk Normal Ketiga Tabel *States*

StateID	Name	Abbreviation
5	California	CA
46	Virginia	VA
17	Kentucky	KY
21	Massachusetts	MA

2.5 Black Box Testing

Seorang *tester* menggunakan tes perilaku (juga dikenal sebagai *black box testing*) untuk menemukan *bug* dalam operasi tingkat tinggi, seperti fitur utama, profil operasional, dan skenario pelanggan. Penguji dapat membuat tes fungsional *black box* berdasarkan apa yang harus dilakukan oleh sebuah sistem. Salah satu hal yang dapat dijadikan contoh adalah jika *SpeedyWriter* harus menyertakan fitur yang menyimpan *file* dalam format XML, maka seorang *tester* harus menguji

apakah *SpeedyWriter* dapat melakukannya. Penguji juga bisa membuat tes non-fungsional kotak hitam berdasarkan apakah sebuah sistem melakukan apa yang dilakukannya secara benar, sebagai contoh jika *DataRocket* bisa mencapai *throughput* yang efektif dari hanya 10 Mbps di dua 1-*gigabit* koneksi Ethernet bertindak sebagai *bridge*, Tes *black box* performa kinerja jaringan bisa menemukan *bug* ini.

Pengujian perilaku melibatkan pemahaman rinci cakupan aplikasi, masalah bisnis yang dipecahkan oleh sistem, dan tujuan yang harus dicapai oleh sistem. Penguji memahami desain sistem, setidaknya pada tingkat yang tinggi, mereka dapat secara efektif meningkatkan kinerja tes perilaku mereka untuk menemukan *bug* umum untuk jenis desain tersebut. Sebuah program diimplementasikan dalam bahasa seperti C dan C++, misalnya. Hal tersebut dapat menyebabkan terjadi *bug* keamanan serius terkait dengan *buffer overflows,* tergantung dari *programmer* yang membuat program tersebut.

Penguji harus mengerti teknik khusus yang paling efektif dalam menemukan sebuah *bug*. Beberapa tes perilaku dilakukan dengan melihat skenario pengguna biasa, banyak tes dilakukan secara ekstrem pada antarmuka, batas-batas sistem, dan berbagai kondisi kesalahan. *Bugs* banyak terdapat pada kondisi-kondisi seperti itu, dan pengujian perilaku juga melibatkan pencarian *bug* sebagai pengujian struktural. Penguji perilaku yang baik menggunakan skrip, persyaratan, dokumentasi, dan pengujian keterampilan untuk membantu menemukan *bug* tersebut. Seorang *tester* jika hanya menjalankan sistem saja atau mendemonstrasikan bahwa sistem berfungsi dengan baik dengan kondisi operasi strd saja kurang efektif untuk pengujian perilaku (Black, 2009).

2.6 Data Flow Diagram

Data flow diagram adalah sebuah teknik penggambaran desain sebuah sistem informasi yang menunjukkan aliran data mulai dari proses *input* sampai ke *output*. *Data flow diagram* memberikan suatu mekanisme untuk pemodelan desain dan penggambaran aliran data (Fatta, 2009).

Diagram konteks adalah diagram yang tersusun atas keseluruhan sistem secara garis besar. Diagram konteks merupakan level tertinggi dari DFD yang menggambarkan seluruh proses *input* sampai *output* serta batasan dari sebuah sistem (Muslihudin & Oktafianto, 2016). Diagram konteks tidak boleh terdapat proses penyimpanan. Simbol-simbol yang digunakan pada *data flow diagram* ditunjukkan pada Tabel 2.13.

Terdapat tiga level dalam *data flow diagram*, yaitu :

1. Diagram Konteks : merupakan penggambaran sistem secara keseluruhan. Diagram konteks merupakan tingkatan tertinggi dalam *data flow diagram*.
2. Diagram Level 1 : merupakan penjabaran dari diagram konteks secara lebih detail. Diagram ini memuat proses penyimpanan data.
3. Diagram Rinci : merupakan diagram yang menguraikan proses apa yang ada dalam diagram level 1. Diagram rinci dapat dibuat lebih rinci lagi jika diperlukan.

2.7 *Entity Relationship Diagram*

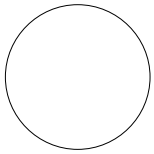
Entity Relationship Diagram digunakan untuk menggambarkan basis data ke dalam bentuk diagram, serta menyediakan sarana untuk lebih mudah memahami berbagai bagian dari sebuah desain basis data. ERD memiliki tiga komponen yaitu *entity*, *relationship* dan atribut (Kusrini, 2007).

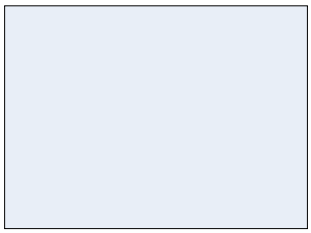
Entity dapat diartikan sebagai sebuah objek yang memiliki identitas dan dapat dibedakan dengan objek lainnya. *Entity* dilambangkan dengan bentuk persegi panjang seperti tampak pada Gambar 2.1.

Relationship adalah hubungan antar *entity*. *Relationship* set adalah kumpulan telasi yang mempunyai tipe data yang sama. *Relationship* set dapat digambarkan dengan Gambar 2.2.

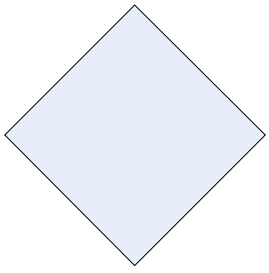
Atribut merupakan deskripsi karakteristik dari suatu entitas. Tiap *property* atau atribut mempunyai *primary key* dan *foreign key*. Atribut dapat digambarkan dengan Gambar 2.3

Tabel 2.13. Simbol-Simbol Pada *Data Flow Diagram*

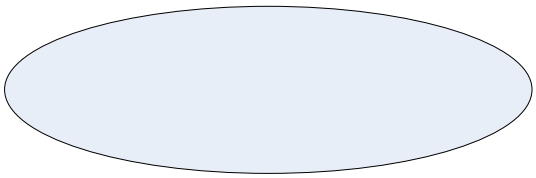
Simbol	Keterangan
	Simbol entitas eksternal berupa orang atau unsur terkait lain yang berinteraksi dalam sistem tetapi di luar sistem
	Simbol orang atau unit yang menggunakan dan melakukan transformasi data
	Simbol aliran data dengan arah khusus dari sumber ke tujuan
	Simbol penyimpanan data atau tempat data yang direferensikan oleh proses



Gambar 2.1. Lambang *Entity* Pada ERD



Gambar 2.2. Lambang *Relationship* Pada ERD



Gambar 2.3. Lambang Atribut Pada ERD

2.7.1 Kardinalitas

Kardinaltas atau pemetaan menunjukkan jumlah *entity* yang dihubungkan ke *entity* yang lain dengan *relationship* (Kusrini, 2007).. Kardinalitas meliputi :

1. Hubungan satu ke satu (*one on one,)* yaitu satu *entity* dihubungkan dengan *entity* lain dengan maksimum satu *entity*
2. Hubungan satu ke banyak (*one to many),* yaitu satu *entity* dihubungkan dengan lebih dari satu *entity*.
3. Hubungan banyak ke banyak (*many to many).* Satu *entity* dapat dihubungkan dengan lebih dari *entity* yang lain, dan *entity* yang lain dapat dihubungkan dengan banyak *entity* sebelumnya.

2.8 Codeigniter

CodeIgniter adalah salah satu *framework open source* pemrograman aplikasi berbasis *web* dengan PHP. *CodeIgniter* memiliki banyak fitur yang menjadikannya unggul dari *framework* yang lain seperti dokumentasi yang lengkap, kemampuan untuk berjalan di lingkungan *shared hosting* dan performa yang tinggi. *CodeIgniter* juga merupakan *framework* yang kompatibel dengan PHP4 dan PHP5, yang mampu berjalan di sebagian besar *web hosting*.

Codeigniter menggunakan *pattern* pemrograman *MVC,* yang membagi aplikasi menjadi tiga bagian, yaitu : *model*, *view*, dan *controller*. *Model* merupakan lapisan abstraksi dari basis data. *View* merupakan bagian yang mengatur tampilan ke pengguna. *Controller* merupakan bagian logika dan penghubung antara view dan model. *Codeigniter* juga menekankan pada penggunaan desain *singleton*, yang membuat sebuah *instance* dari sebuah kelas yang dipanggil berkali-kali. Contoh penggunaan dari *singleton* salah satunya

adalah pada saat melakukan koneksi ke basis data, yang mana hanya memerlukan satu koneksi saja setiap kali sebuah kelas digunakan.

CodeIgniter juga memiliki implementasi *Active Record* yang berfungsi untuk memudahkan dalam penulisan *query* SQL yang kompleks dan membuat kode aplikasi lebih mudah dibaca. Fitur ini juga memungkinkan mengganti *driver* basis data dengan mudah, misalkan menulis program dengan basis data MySQL maka dapat dengan mudah diganti dengan basis data Oracle tanpa harus menulis ulang deklarasi kode-kode program dalam aplikasi. *CodeIgniter* juga dilengkapi dengan sejumlah *library* yang sangat berguna dan beberapa perangkat lainnya yang memudahkan dalam membangun aplikasi (Griffiths, 2010).

2.9 Bootstrap

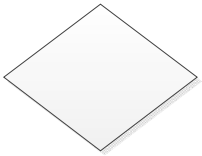
Bootstrap atau *twitter bootstrap* adalah sebuah *toolkit* untuk membantu pengembang perangkat lunak berbasis web untuk dalam mendesain tampilan. Situs yang dibuat dengan alat bantu ini tampilannya mirip dengan *twitter,* sesuai dengan namanya. *Bootstrap* dibuat dengan HTML, CSS dan *javascript.* Seorang pengembang *website* cukup memanggil fungsi-fungsi yang telah tersedia dalam *bootstrap* dalam mendesain tampilan sebuah aplikasi *web,* sehingga proses pembangunan aplikasi dapat dilakukan dengan relatif cepat (Ridha, 2013).

2.10 *Flowchart*

Flowchart adalah sebuah media penyajian sistematis tentang proses dan logika dari kegiatan pengolahan data dan informasi atau penggambaran dari tahapan dan urutan prosedur dari suatu program. *Flowchart* membantu analis dan programmer untuk memecahkan permasalah kedalam segmen-segmen yang lebih kecil dan membantu untuk menganalisis alternatif-alternatif lain dalam membuat suatu program. *System flowchart* adalah penggambaran urutan proses yang terjadi dalam sebuah sistem dengan menunjukkan alat *input, output* serta media penyimpanan dalam proses pengolahan data. *Program flowchart* adalah suatu bagan dengan simbol-simbol tertentu yang menggambarkan urutan proses secara detail dan hubungan antara suatu proses dengan proses lainnya dalam suatu

sistem. Simbol-simbol dalam *flowchart* dapat dilihat pada Tabel 2.14 (Anharku, 2009).

Tabel 2.14. Simbol-simbol *Flowchart*

Simbol	Nama	Fungsi
	Terminator	Permulaan/Akhir Program
	Garis Alir	Arah Aliran Program
	Proses	Proses Perhitungan / Pengolahan Data
	Decision	Perbandingan Pernyataan yang memberikan pilihan untuk langkah selanjutnya
	Sub Program	Permulaan sub program/proses menjalankan sub program
	Input Output Data	Proses input/output data, informasi