BAB II

LANDASAN TEORI

1.1 Sistem

Suatu sistem terdiri dari sejumlah komponen yang saling berinteraksi, yang artinya saling bekerja sama membentuk satu kesatuan. Komponen-komponen sistem atau elemen-elemen sistem dapat berupa suatu subsistem atau bagian-bagian dari sistem. Sistem merupakan suatu jaringan kerja dari prosedur-prosedur yang saling berhubungan, berkumpul bersama-sama untuk melakukan suatu kegiatan atau untuk menyelesaikan suatu sasaran yang tertentu. Susanto mengemukakan bahwa sistem adalah kumpulan atau grup dari bagian atau komponen apapun baik fisik ataupun non fisik yang saling berhubungan satu sama lain dan bekerja sama secara harmoni suntuk mencapai satu tujuan tertentu.(Suherman, 2017)

1.2 Informasi

Informasi adalah data yang diolah menjadi bentuk lebih berguna dan lebih berarti bagi yang menerimanya. Informasi juga disebut data yang diproses atau data yang memiliki arti. Informasi merupakan data yang telah diproses sedemikian rupa sehingga meningkatkan pengetahuan seseorang yang menggunakan. Para pembuat keputusan memahami bahwa informasi menjadi faktor kritis dalam menentukan kesuksesan atau kegagalan dalam suatu bidang usaha. Sistem apapun tanpa ada informasi tidak akan berguna, karena sistem tersebut akan mengalami kemacetan dan akhirnya berhenti. Informasi dapat berupa data mentah, data tersusun, kapasitas sebuah saluran informasi, dan sebagainya.(Firman et al., 2016)

1.3 Sistem Informasi

Sistem informasi adalah suatu sistem di dalam suatu organisasi yang mempertemukan kebutuhan pengolahan data transaksi harian, mendukung operasi, bersifat manajerial dan kegiatan strategi dari suatu organisasi serta menyediakan pihak luar tertentu dengan laporan-laporan yang diperlukan. Sistem informasi juga

dapat di definisikan sebagai suatu sistem yang dibuat oleh manusia yang terdiri dari komponenkomponen dalam organisasi untuk menyajikan informasi. Sistem informasi merupakan sistem pembangkit informasi, kemudian dengan integrasi yang dimiliki antar sub sistem, maka sistem informasi akan mampu menyediakan informasi yang berkualitas, tepat, cepat dan akurat sesuai dengan manajemen yang membutuhkannya.

Pada lingkungan berbasis komputer, sistem informasi menggunakan perangkat keras dan lunak komputer, jaringan telekomunikasi, manajemen basis data, dan berbagai bentuk teknologi informasi yang lain dengan tujuan untuk mengubah sumber data menjadi berbagai macam informasi yang dibutuhkan oleh pemakai.(Firman et al., 2016)

1.4 Sistem Berbasis *Web*

Sistem berbasis *web* adalah aplikasi atau layanan yang berada dalam *server* dan dapat diakses dengan menggunakan penjelajah *web* dan karenanya dapat diakses dari mana saja melaui *internet*. Satu-satunya peranti lunak sisi klien yang dibutuhkan untuk mengakses dan menjalankan aplikasi berbasis *web* adalah lingkungan penjelajah *web,* dan berbagai aplikasi tersebut harus sesuai dengan *protocol internet*-nya.(Susena et al., 2019)

1.5 Pendataan

Pendataan merupakan suatu proses pencatatan keterangan yang benar dan nyata tentang sesuatu, baik manusia, benda, lingkungan, maupun kejadian tertentu. Pencatatan ini dimaksudkan sebagai suatu dokumentasi atau arsip yang dapat digunakan untuk suatu keperluan di masa depan. Adapun keperluan utama yang lazim menjadi penggagas suatu pendataan adalah pembuatan laporan. Pembuatan laporan dimaksudkan sebagai dasar atau bahan pertimbangan bagi pemimpin organisasi/perusahaan untuk mengambil suatu keputusan. (Magdalena et al., 2021)

1.6 PHP (Hypertext Preprocessor)

PHP atau kependekan dari *Hypertext Preprocessor* adalah salah satu bahasa pemrograman *open source* yang sangat cocok atau dikhususkan untuk pengembangan web dan dapat ditanamkan pada sebuah skripsi HTML. Bahasa PHP dapat dikatakan menggambarkan beberapa bahasa pemrograman seperti C, Java, dan Perl serta mudah untuk dipelajari.

PHP merupakan bahasa *scripting server – side*, dimana pemrosesan datanya dilakukan pada sisi server. Sederhananya, serverlah yang akan menerjemahkan skrip program, baru kemudian hasilnya akan dikirim kepada client yang melakukan permintaan. Adapun pengertian lain PHP adalah akronim dari *Hypertext Preprocessor*, yaitu suatu bahasa pemrograman berbasiskan kode – kode (script) yang digunakan untuk mengolah suatu data dan mengirimkannya kembali ke web browser menjadi kode HTML. PHP (atau resminya PHP: *Hypertext Preprocessor*) adalah skrip bersifat server – side yang ditambahkan ke dalam HTM. (Mirza & Putra, 2019)

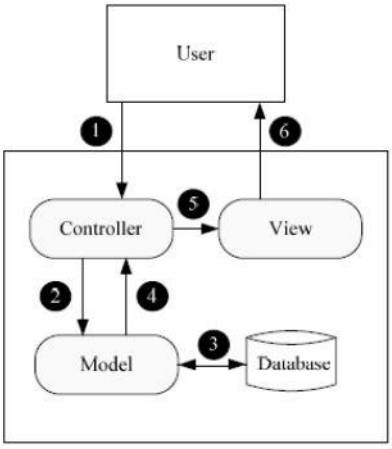
1.6.1 Codeingniter

CodeIgniter adalah framework aplikasi web yang open source untuk bahasa pemrograman PHP. CodeIgniter memiliki banyak fitur yang membuatnya berbeda dengan framework lainnya. Tidak seperti beberapa framework PHP lainnya, dokumentasi untuk framework ini sangat lengkap,yang mencakup seluruh aspek dalam framework. CodeIgniter juga mampu bekerja pada lingkungan shared hosting karena memiliki ukuran yang sangat kecil, namun memiliki kinerja yang sangat luar biasa (Griffiths, 2010). *Codeigniter* dikembangkan oleh Rick Ellis, dengan versi awal yang dirilis pada 28 Februari 2006. Dari tahun itulah hingga sekarang, telah muncul versi CodeIgniter yang terus berkembang dengan penambahan fitur baru dari versi sebelumnya. Untuk versi terbaru dari *CodeIgniter* adalah versi 2.1.4.(Burrakhman et al., 2016)

1.6.2 Pengertian Konsep MVC Pada Web

Model-View-Controller (MVC) adalah pola arsitektur yang memisahkan aplikasi dalam tiga komponen utama Logis: model, *view* dan *controller*. Masing-masing komponen ini dibangun untuk menangani aspek-aspek tertentu pembangunan aplikasi. MVC adalah salah satu kerangka pembangunan *web* standar industri paling sering digunakan untuk menciptakan proyek yang terukur yang besar dan *extensible*.

Dengan menggunakan prinsip MVC suatu aplikasi dapat dikembangkan sesuai dengan kemampuan developernya, yaitu *progammer* yang menangani bagian model dan *controller*. Sedangkan desainer yang menangani bagian *view*, sehingga penggunaan arsitektur MVC dapat meningkatkan *maintanability* dan organisasi kode. (Pasaribu, 2017)



Gambar 1.1 Skema Model-View-Controller (MVC)

Pengertian MVC adalah sebuah bentuk pemrograman yang memisahkan berdasarkan logika penanganan tampilan, logika pengontrolan dan logika *Model*. MVC bertujuan supaya pada pengembangan perangkat lunak yang besar mudah untuk dilakukan maintenance. Bagian - bagian dari MVC adalah:

a. Model

   Model adalah bagian kode program yang menangani *query* atau *database*. Jadi isi zari model merupakan bagian (fungsi-fungsi) yang berhubungan langsung

dengan *database* untuk memanipulasi data seperti memasukkan data, pembaruan data, hapus data, dan lain-lain, namun tidak dapat berhubungan langsung dengan bagian *view*.

b. *View*

View adalah bagian kode program yang mengatur tampilan *website*. Pada aplikasi *web* bagian *View* biasanya berupa *file* template HTML, yang diatur oleh *controller*. Bagian ini tidak memiliki akses langsung terhadap bagian model namun berhubungan langsung dengan *controller*. *View* berfungsi untuk menerima dan merepresentasikan data kepada pengguna. Jadi bisa dikatakan bahwa *View* merupakan halaman *web.*

c. *Controller*

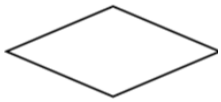
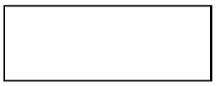
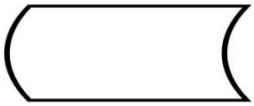
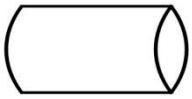
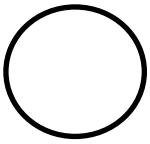
Controller merupakan bagian yang menjembatani *model* dan *view*. *Controller* berisi perintah-perintah yang berfungsi untuk memproses suatu data dan mengirimkannya ke halaman *web*. *Controller* berfungsi untuk menerima *request* dan data dari *user* kemudian menentukan apa yang akan diproses oleh aplikasi.

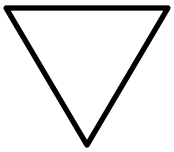
1.7 Flowmap

Flowmap atau bagan alir adalah bagan yang menunjukan aliran di dalam program atau prosedur sistem secara logika. *Flowmap* ini berfungsi untuk memodelkan masukan, keluaran, proses maupun transaksi dengan menggunakan simbol-simbol tertentu. Pembuatan *Flowmap* ini harus dapat memudahkan pemakai dalam memahami alur dari sistem atau transaksi. Berikut simbol-simbol yang digunakan dalam membuat *flowmap* menurut.(Lisnawanty, 2014)

Tabel 1.1 Simbol dan Deskripsi Flowmap

Simbol	Deskripsi
	***Terminator,*** untuk menyatakan permulaan atau akhir program.

	***Decision,*** perbandingan, pernyataan, penyeleksian data yang memberikan pilihan untuk langkah selanjutnya.
	**Garis Alir,** menunjukkan arah aliran program.
	**Proses Komputerisasi,** menunjukan proses penghitung atau proses pengolahan data.
	**Dokumen,** menunjukan proses *input* atau *output* berupa dokumen.
	***Manual Storage***, berfungsi sebagai tempat penyimpanan manual.
	***Storage Data,*** menunjukan akses langsung perangkat penyimpanan data ke dalam *Harddisk*.
	***Manual Operation,*** menunjukan proses yang dilakukan secara manual.
	***Input Manual,*** menunjukan proses input menggunakan *keyboard*.
	***Konektor***, menunjukkan penghubung ke halaman yang masih sama atau ke halaman lain.

	**Arsip,** Menunjukkan pengarispan *file* tanpa menggunakan komputer.

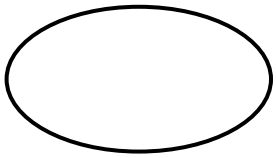
1.8 UML (Unified Modelling Languange)

Unified Modelling Language (UML) adalah suatu alat untuk memvisualisasikan dan mendokumentasikan hasil analisa dan desain yang berisi sintak dalam memodelkan sistem secara visual. Juga merupakan satu kumpulan konvensi pemodelan yang digunakan untuk menentukan atau menggambarkan sebuah sistem *software* yang terkait dengan objek.(Haviluddin, 2011)

1.8.1 Use Case Diagram

Use case diagram merupakan pemodelan untuk kelakukan (behavior) sistem informasi yang akan dibuat. *Use case diagram* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat. Berikut simbol-simbol yang ada pada *use case diagram*.(Sangadah & Kartawidjaja, 2020)

Tabel 1.2 Simbol dan Deskripsi Use Case Diagram

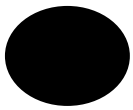
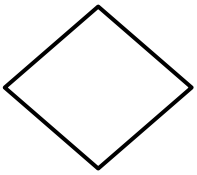
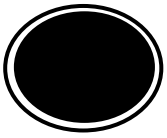
Simbol	Deskripsi
 Use Case	Use case merupakan fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antara unit atau aktor.

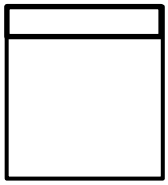
 Aktor	Aktor merupakan orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat diluar sistem informasi yang akan dibuat itu sendiri. Walaupun simbol dari aktor adalah gambar orang, tapi aktor tidak selamanya mendeskripsi sebagai orang.
 Asosiasi	Asosiasi adalah komunikasi antara aktor dan use case yang berpartisipasi pada use case atau use case memiliki interaksi dengan aktor.
 Include	Include diartikan sebagai relasi use case tambahan ke sebuah use case di mana use case yang ditambahkan memerlukan use case ini untuk menjalankan fungsinya atau sebagai syarat dijalankan use case ini.
 *Sistem*	*Sistem* merupakan menspesifikasikan paket yang menampilkan sistem secara terbatas.

1.8.2 Diagram Aktivitas (*Activity Dagram*)

Activity diagram atau diagram aktivitas menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis atau menu yang ada pada perangkat lunak. Dimana diagram aktivitas menggambarkan aktivitas sistem bukan apa yang dilakukan aktor, jadi aktivitas yang dapat dilakukan oleh sistem. Berikut adalah simbol-simbol yang ada pada *activity diagram*.(Mirza & Putra, 2019)

Tabel 1.3 Simbol dan Deskripsi Diagram Aktivitas (Activity Diagram)

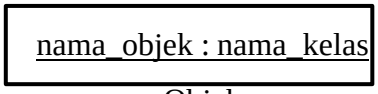
Simbol	Deskripsi
 Start/Status awal	Menunjukan dimana aliran kerja dimulai
 Aktivitas	Langkah-langkah dalam sebuah activity diagram. Aktivitas bisa terjadi saat memasuki activity diagram, meninggalkan activity diagram, atau pada event yang spesifik
 Percabangan	Menunjukan suatu keputusan yang mempunyai satu atau lebih transisi dan dua atau lebih transisi sesuai dengan suatu kondisi.
 Penggabungan	Dimana lebih dari satu aktivitas digabungkan menjadi satu.
 End/Status akhir	Menunjukan dimana aliran kerja diakhiri.

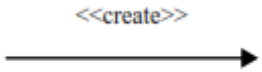
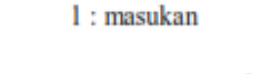
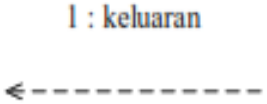
 Swimlane	Menunjukan siapa yang bertanggung jawab dalam melakukan aktivitas dalam suatu diagram.

1.8.3 Diagram Urutan (*Sequence*)

Sequence diagram menggambarkan kelakuan objek pada *use case* dan message yang dikirim dan diterima oleh objek. Berikut simbol-simbol yang ada pada *sequence* diagram.(Achmad et al., 2019)

Tabel 1.4 Simbol dan Deskripsi Diagram Urutan (Sequence)

Simbol	Deskripsi
 Aktor	Orang, proses atau sistem lain yang berinteraksi dengan sistem informasi dan mendapat manfaat dari sistem.
 Garis hidup/*lifeline*	Menyatakan kehidupan suatu objek.
 Objek	Menyatakan objek yang berinteraksi pesan.

 Waktu aktif	Menyatakan objek dalam keadaan aktif dan berinteraksi.
 Pesan tipe *create*	Menyatakan suatu objek membuat objek yang lain, arah panah mengarahkan pada objek yang dibuat.
 Pesan tipe *call*	Menyatakan suatu objek memanggil operasi/metode yang ada pada objek lain atau dirinya sendiri.
 Pesan tipe *send*	Menyatakan bahwa suatu objek mengirimkan data/masukan/informasi ke objek lainnya, arah panah mengarah pada objek yang dikirim.
 Pesan tipe *return*	Menyatakan bahwa suatu objek yang telah menjalankan suatu operasi atau metode menghasilkan suatu kembalian ke objek tertentu.

(Rosyadi & Mutaalimah, 2015).

1.8.4 Diagram Kelas (*Class Diagram*)

Class diagram adalah diagram yang digunakan untuk menampilkan beberapa kelas serta paket-paket yang ada dalam sistem atau perangkat lunak yang digunakan. *Class diagram* memberi gambaran (diagram statis) tentang sistem atau perangkat lunak dan relas-relasi yang ada didalamnya. Sebuah class memiliki tiga area pokok:

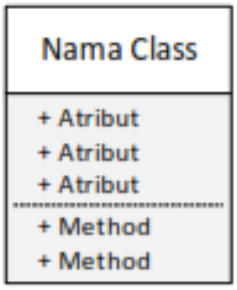
a. Nama, merupakan nama dari sebuah kelas

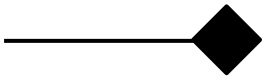
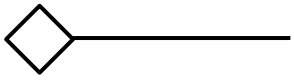
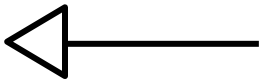
b. Atribut, merupakan peroperti dari sebuah kelas. Atribut melambangkan batas nilai yang mungkin ada pada obyek dari *class*

Operasi, adalah sesuatu yang bisa dilakukan oleh sebuah *class* atau yang dapat dilakukan oleh *class* lain terhadap sebuah *class*.
(Kuswidiardi, 2015).

Berikut adalah notasi–notasi umum yang terdapat pada *class diagram*:

Tabel 1.5 Simbol dan Deskripsi Diagram Kelas (Class Diagram)

Simbol	Deskripsi
 *Class*	Blok-blok pembangun pada pemrograman berorientasi objek. Sebuah class digambarkan sebagai sebuah kotak yang terbagi atas 3 bagian. Bagian atas adalah bagian nama dari class. Bagian tengah mendefinisikan property/atribut class. Bagian akhir mendefinisikan method-method dari sebuah class.
 *Association*	Asosiasi merupakan sebuah relationship paling umum antara 2 class dan dilambangkan oleh sebuah garis yang menghubungkan antara 2 class. Garis ini bisa melambangkan tipe-tipe relationship dan juga dapat menampilkan hukum-hukum multiplisitas pada sebuah relationship.

 *Composition*	Jika sebuah class tidak bisa berdiri sendiri dan harus merupakan bagian dari class yang lain, maka class tersebut memiliki relasi Composition terhadap class tempat bergantung tersebut. Sebuah relationship composition digambarkan sebagai garis dengan ujung berbentuk jajaran genjang berisi/solid.
 *Dependency*	Penggunaan dependency digunakan untuk menunjukkan operasi pada suatu class yang menggunakan class yang lain. Sebuah dependency dilambangkan sebagai sebuah panah bertitik-titik.
 *Aggregation*	Aggregation mengindikasikan keseluruhan bagian relationship dan biasanya disebut sebagai relasi.
 *Generalization*	Relasi generalization sepadan dengan sebuah relasi inheritance pada konsep berorientasi obyek. Sebuah generalization dilambangkan dengan sebuah panah dengan kepala panah yang tidak solid yang mengarah ke kelas "parent"-nya atau induknya.

1.9 XAMPP

XAMPP Server adalah perangkat lunak gratis yang mendukung banyak sistem operasi, merupakan kompilasi dari beberapa program untuk menjalankan fungsinya sebagai server yang berdiri sendiri, yang terdiri atas program Apache HTTP Server, MySQL database, dan penerjemah bahasa yang dirulis dengan PHP dan Perl. XAMPP adalah nama yang merupakan singkatan dari X berbagai sistem oprasi (Linux, MAC, Windows), Apache, MySQL, PHP, PERL.Program ini tersedia dalam

GNU General Public License dan bebas, merupakan web server yang mudah digunakan yang mampu melayani halaman dinamis.(Burrakhman et al., 2016).

1.10 MYSQL

MySQL (*My Structure Query Language*) adalah sebuah program pembuat database yang bersifat *open source*, artinya siapa saja boleh menggunakannya dan tidak dicekal. MySQL juga merupakan program pengakses database yang bersifat jaringan sehingga dapat digunakan untuk aplikasi multi user (banyak pengguna). Saat ini database MySQL telah digunakan hampir oleh semua program database, apalagi dalam pemrograman web. Kelebihan lain dari MySQL adalah ia menggunakan bahasa Query standar yang dimiliki SQL (*Structure Query Language*). SQL adalah suatu bahasa permintaan yang terstruktur yang telah distandarkan untuk semua program pengaksesan database seperti Oracle, Posgres, SQL, SQL Server, dan lain-lain.(Rosyadi & Mutaalimah, 2015).

1.11 Database

Database dalam bahasa Indonesianya Basis data adalah kumpulan informasi yang disimpan di dalam komputer secara sistematik sehingga dapat diperiksa menggunakan suatu program komputer untuk memperoleh informasi dari basis data tersebut. Perangkat lunak yang digunakan untuk mengelola dan memanggil kueri (query) basis data disebut sistem manajemen basis data (database management *sistem*, DBMS).(Herdiana, 2013).

1.12 Tabel

Tabel hakekatnya merupakan susunan data dalam baris dan kolom atau bisa juga dalam struktur yang lebih kompleks. Dimana penggunaan tabel banyak diterapkan dalam berbagai bidang, diantaranya yaitu komunikasi, penelitian kuantitatif dan teknik analisis data. Tabel bisa muncul di media cetak, perangkat lunak komputer, catatan tulisan tangan, ornamen arsitektur, rambu lalu lintas, dan beragam tempat lainnya.

1.12.1 Pengertian Tabel

Tabel adalah alat bantu visual yang berfungsi untuk menjelaskan suatu fakta atau informasi secara singkat, jelas, dan lebih menarik daripada hanya disajikan secara naratif atau menggunakan kata-kata. Sajian informasi yang menggunakan tabel lebih mudah dibaca, sehingga lebih mudah pula untuk disimpulkan.

1.12.2 Pengertian Tabel Menurut Para Ahli

Adapun definisi tabel menurut para ahli, antara lain:

1. Nurhadi (2015)**,** Pengertian tabel adalah sajian data yang berupa angka-angka yang disajikan dalam bentuk baris dan kolom yang diklasifikasikan secara sistematis menurut kesatuan tertentu.

2. Database Guide (2016), Arti tabel mirip dengan lembar kerja dalam aplikasi spreadsheet. Baris tertera secara horizontal dan mewakili setiap record. Kolom tertera secara vertikal dan mewakili bidang tertentu. Baris dan kolom berpotongan, membentuk kisi. Perpotongan baris dan kolom menentukan setiap sel dalam tabel.

1.12.3 Relasi Tabel

Relasi tabel merupakan hubungan antara beberapa tabel di mana relasi tabel ini dihubungkan dengan *field* yang menjadi *primary key* dan *foreign key*. *Primary key* merupakan *key fields* yang menjadi kunci (identitas) untuk setiap baris (*record*) pada suatu tabel dan tidak bisa diisi dengan data yang sama. Dengan kata lain *primary key* menjadikan tiap *record* memiliki identitas sendiri yang membedakan satu sama lainnya (unik). Sedangkan *Foreign key* merupakan *field* kunci pada suatu tabel yang digunakan pada tabel yang lain sebagai penghubung (relasi) antar tabel. *Foreign key* berguna untuk mendefinisikan kolom-kolom pada suatu tabel yang nilainya mengacu ke tabel yang lain, jadi nilai kolom foreign key nilainya harus diambil dari nilai kolom tabel lain. (Maimunah et al., 2021).

Untuk dapat membuat relasi antar tabel minimal mempunyai dua buah tabel yang saling terkait. Relasi tabel dapat berupa relasi *one-to-one*, *one-to-many*, atau *many-to-many*.

1. Relasi *One to One*

   Relasi *one to one* merepresentasikan relasi antara entitas di mana satu data memiliki relasi dengan satu dan hanya satu data saja dalam entititas yang berelasi. Sebagai contoh, setiap mahasiswa hanya memiliki 1 NIM yang tidak bisa dimiliki oleh mahasiswa lain. Berikut gambarannya :

Gambar 1.2 Relasi One to One

2. Relasi *One to Many*

   Relasi *one to many* merupakan relasi yang paling umum digunakan. Pada relasi *one to many*, setiap data pada suatu entitas memiliki relasi dengan nol atau lebih data pada entitas. Sebagai contoh, Setiap dosen dapat mengajar beberapa atau lebih dari satu mata mata kuliah. Berikut gambarannya :

Gambar 1.3 Relasi One to Many

3. Relasi *Many to Many*

   Relasi *many to many* merupakan relasi dimana data dalam sebuah entitas memiliki relasi dengan nol atau lebih data pada entitas kedua. Dan pada saat yang bersamaan, setiap data pada entitas kedua memiliki relasi dengan nol atau

lebih data pada entitas pertama. Sebagai contoh, beberapa mahasiswa dapat mengontrak beberapa mata kuliah pada setiap semester. Berikut gambarannya:

Gambar 1.4 Relasi Many to Many

(Rosihan & Lutfi, 2018)

1.13 Normalisasi

Normalisasi adalah proses pembentukan struktur basis data sehingga sebagian besar *ambiguity* bias dihilangkan. Tahap Normalisasi dimulai dari tahap paling ringan (1NF) hingga paling ketat (5NF). Biasanya hanya sampai pada tingkat 3NF atau BCNF karena sudah cukup memadai untuk menghasilkan tabel-tabel yang berkualitas baik. Sebuah tabel dikatakan baik (efisien) atau normal jika memenuhi 3 kriteria sebagai berikut :

a. Jika ada dekomposisi (penguraian) tabel, maka dekomposisinya harus dijamin aman (*Lossless-Join Decomposition*). Artinya, setelah tabel tersebut diuraikan / didekomposisi menjadi tabel-tabel baru, tabel-tabel baru tersebut bisa menghasilkan tabel semula dengan sama persis.
b. Terpeliharanya ketergantungan fungsional pada saat perubahan data (Dependency Preservation).
c. Tidak melanggar Boyce-Code Normal Form (BCNF). Adapun bentuk-bentuk normalisasi sebagai berikut :
    a. Bentuk Normal Tahap Pertama (1st Normal Form / 1NF)
    b. Bentuk Normal Tahap Kedua (2nd Normal Form / 2NF)
    c. Bentuk Normal Tahap (3rd Normal Form / 3NF)
    d. Boyce-Code Normal Form (BCNF)
    e. Bentuk Normal Tahap (4th Normal Form / 4NF)

f. Bentuk Normal Tahap (5th Normal Form / 5NF)
g. Domain Key Normal Form (DKNF)
h. Bentuk Normal Tahap (6th Normal Form / 6NF)

Namun dalam prakteknya dalam dunia industri bentuk normalisasi ini yang paling sering digunakan ada sekitar 5 bentuk. Sudah disebutkan bahwa secara teori, bentuk normal suatu relasi bisa sampai ke tingkat lima 5NF, yaitu 1NF – 2NF – 3NF/BCNF – 4NF – 5NF. Tetapi secara praktik dalam dunia nyata, relasi dalam suatu database sudah dibilang baik kalau sudah mencapai 3NF (bentuk normal ketiga). Untuk lebih jelasnya ciri-ciri dari bentuk-bentuk tahapan normalisasi dapat dilihat pada gambar di bawah ini :

Pengimplementasikan normalisasi ke dalam sebuah rancangan *database*

dapat diikuti langkahlangkah sebagai berikut :

Bentuk Tidak Normal

↓ Menghilangkan perulangan group

Bentuk Normal Pertama (1 NF)

↓ Menghilangkan ketergantungan parsial

Bentuk Normal Kedua (2 NF)

↓ Menghilangkan ketergantungan transitif

Bentuk Normal Ketiga (3 NF)

↓ Menghilangkan anomali-anomali hasil dari ketergantungan fungsional

Bentuk Normal *Boyce-Cood* (BCNF)

↓ Menghilangkan ketergantungan *multivalue*

Bentuk Normal Keempat (4 NF)

↓ Menghilangkan anomali-anomali yang tersisa

Bentuk Normal Kelima

1. *Unnormalization Form*

Bentuk yang tidak normal dimaksudkan suatu kumpulan data yang akan diolah yang diperoleh dari format–format yang beraneka ragam, masih terdapat duplikasi atau pengulangan data, bisa saja tidak sempurna atau tidak lengkap, dan sesuai fakta lapangan.

Bentuk ini didapat dari dokumen yang ada dilapangan atau manual dengan atribut bukan nilai sederhana. Untuk lebih jelasnya dapat dilihat pada gambar di bawah ini

Tabel Mahasiswa

nim	nama	prodi	kode_mtk	nama_mtk	id_dosen	nama_dosen	nilai
1234	Roma	TI	TI4801	Sistem Basis Data	SSD	Surya	A
			TI4815	Rekayasa Perangkat Lunak	RNW	Ronal	C
2345	Beni	SI	TI4801	Sistem Basis Data	SSD	Surya	B
			UN121	Kalkulus	WHY	Wahyu	B
			UN125	Bahasa Indonesia	SAB	Sabrina	A

Gambar 1.5 Unnormalisasi Tabel

2. Bentuk Normal Tahap Pertama (1st Normal Form /1NF)

Adapun ciri-ciri bentuk normal 1NF adalah :

a. Jika sebuah tabel tidak memiliki atribut bernilai banyak (*multivalued attribute*) denga arti harus bernilai tunggal.

b. Jika sebuah tabel tidak memiliki atribut *composite* atau kombinasinya dalam domain data yang sama. Setiap atribut dalam tabel tersebut harus bernilai *atomic* (tidak dapat dibagi-bagi lagi).

c. Jika sebuah tabel tidak memiliki atribut turunan/*derivatied value*.

d. Jika sebuah tabel tidak memiliki *record* yang bernilai ganda/ *redundancy*.

e. atribut *composite* atau kombinasinya dalam domain data yang sama.

f. Setiap atribut dalam tabel tersebut harus bernilai *atomic* (tidak dapat dibagi- bagi lagi).

Tabel dari *unnormalisasi* pada langkah pertama dapat dekomposisi menjadi gambar di bawah ini :

Tabel Mahasiswa

nim	nama	prodi	kode_mtk	nama_mtk	id_dosen	nama_dosen	nilai
1234	Roma	TI	TI4801	Sistem Basis Data	SSD	Surya	A
1234	Roma	TI	TI4815	Rekayasa Perangkat Lunak	RNW	Ronal	C
2345	Beni	SI	TI4801	Sistem Basis Data	SSD	Surya	B
2345	Beni	SI	UN121	Kalkulus	WHY	Wahyu	B
2345	Beni	SI	UN125	Bahasa Indonesia	SAB	Sabrina	A

Gambar 1.6 Normalisasi Bentuk 1NF

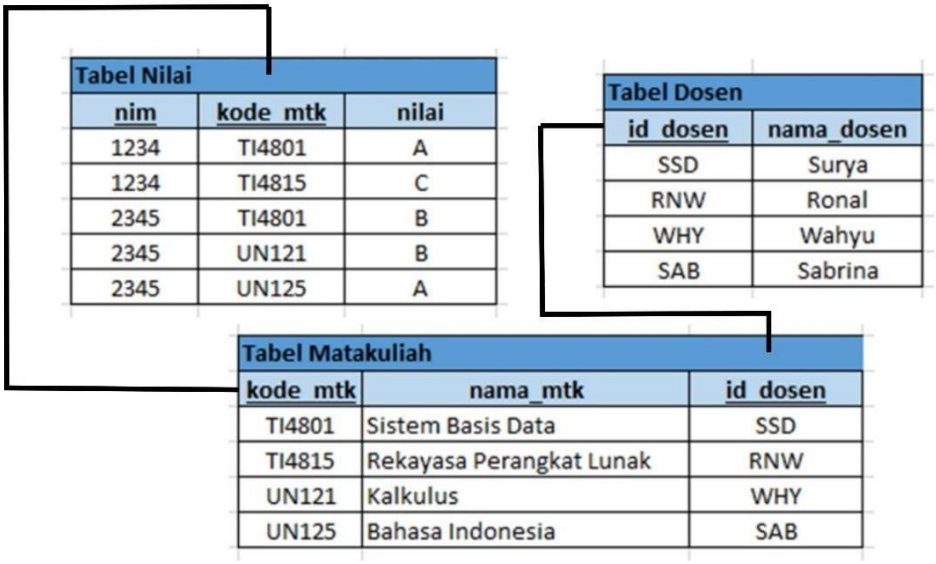
Dari gambar di atas masih terdapat atribut yang muncul secara berulang, untuk itu harus melanjutkan ke tahap normalisasi kedua.

3. Bentuk Normal Tahap Kedua (2nd Normal Form)

a. Bentuk normal 2NF terpenuhi dalam sebuah tabel jika telah memenuhi bentuk 1NF, dan semua atribut selain *primary key*, secara utuh memiliki *Functional Dependency* pada *primary key*

b. Sebuah tabel tidak memenuhi 2NF, jika ada atribut yang ketergantungannya (*Functional Dependency*) hanya bersifat parsial saja (hanya tergantung pada sebagian dari *primary key*)

c. Jika terdapat atribut yang tidak memiliki ketergantungan terhadap primary key, maka atribut tersebut harus dipindah atau dihilangkan. Hal ini dapat dilihat pada gambar di bawah ini :



Tabel Nilai

nim	kode_mtk	nilai
1234	TI4801	A
1234	TI4815	C
2345	TI4801	B
2345	UN121	B
2345	UN125	A

Tabel Dosen

id_dosen	nama_dosen
SSD	Surya
RNW	Ronal
WHY	Wahyu
SAB	Sabrina

Tabel Matakuliah

kode_mtk	nama_mtk	id_dosen
TI4801	Sistem Basis Data	SSD
TI4815	Rekayasa Perangkat Lunak	RNW
UN121	Kalkulus	WHY
UN125	Bahasa Indonesia	SAB

Gambar 1.7 Normalisasi Bentuk 2NF

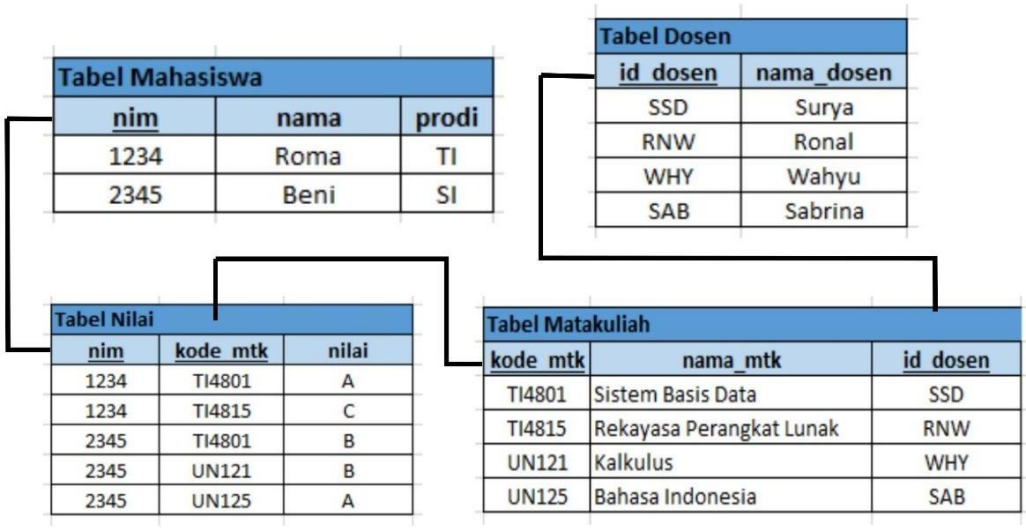
4. Bentuk Normal Tahap Ketiga (3rd Normal Form /3NF)

a. Bentuk normal 3NF terpenuhi jika telah memenuhi bentuk 2NF, dan jika **tidak ada** atribut *non primary key (biasa)* yang memiliki ketergantungan terhadap atribut *non primary key (biasa)* yanglainnya.

b. Untuk setiap *Functional Dependency* dengan notasi X -> A, maka:

- X harus menjadi *superkey* pada tabel tersebut.
- Atau A merupakan bagian dari *primary key* pada tabel tersebut.

Hal ini dapat dilihat pada gambar di bawah ini, yakni tabel mahasiswa, tabel dosen, tabel matakuliah dan tabel nilai.



Gambar 1.8 Normalisasi Bentuk 3NF

Selanjutnya **langkah kelima** dilakukan pengecekan *composite* dan *multivalue attribute* dengan cara melihat data yang mengandung tanda koma. Jika tidak ada data yang mengandung nilai koma, maka tabel yang dihasilkan tetap dan proses normalisasi selesai, dan tabel dapat diimplementasikan ke database relational. (Suryadi, 2019)

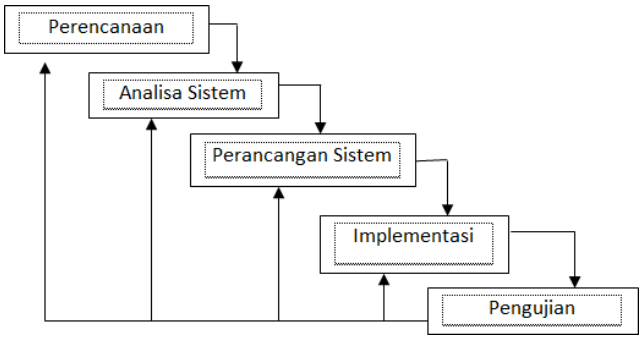
1.14 Metode Pengembangan Sistem

Metode pengembangan sistem yang digunakan dalam merancang dan membangun sistem yaitu sebagai berikut:

1.14.1 Metode Waterfall

Metode waterfall sebagai metode pengembangan sistem. Model ini mengusulkann sebuah pendekatan kepada perkembangan perangkat lunak yang sistematik dalam tingkat kemajuan system pada seluruh analisis, desain, kode, pengujian dan pemeliharaan. Model waterfall merupakan metode yang paliing banyak digunakan dalam software engineering, karena pemodelan sistem terbagi menjadi

tahapan-tahapan yang mengikuti pola teratur, seperti layaknya air terjun. Tahapan-tahapan pada model waterfall dapat dilihat pada gambar berikut:



Gambar 1.9 Waterfall Model

Berdasarkan model waterfall, garis besar penyelesaian masalah ini terdapat 5 tahapan yang meliputi:

a. Tahap Perencanaan Sistem

Pada tahap ini akan dilakukan pendefinisian seluruh kebutuhan perangkat lunak yang nantinya akan dijadikan sebagai SRS (software Requirements Spesification).

b. Tahap Analisa Sistem

Tahapan Analisis kebutuhan terdiri atas analisis kebutuhan dan analisis pemodelan Analisis kebutuhan merupakan pengidentifikasian kebutuhan yang diperlukan oleh sistem.

c. Tahap Perancangan

Proses perancangan sistem membagi persyaratan dalam system perangkat keras atau perangkat lunak.

d. Tahap Implementasi

Pada tahap ini, perancangan perangkat lunak direalisasian sebagai serangkaian program atau unit program.

e. Tahap Pengujian Sistem

Tahap pengujian adalah proses eksekusi suatu program, bila pengujian dilakukan secara sukses (sesuai dengan sasaran tersebut) maka tidak akan dprodukukan kesalahan di dalam perangkat lunak. (Muslim, 2016)

1.15 Metode Pengujian Sistem

Terdapat beberapa metode dalam pengujian perangkat lunak yaitu :

1. Pengujian *Black Box*

Pengujian *black box* bertujuan untuk menemukan kesalahan fungsi pada program. Pengujian dilakukan dengan cara memasukan input tertentu dan melihat hasil yang dapat dari input tersebut. Pengujian *black box*, yang diuji adalah masukan serta keluarannya. Metode uji coba *black box* memfokuskan pada keperluan fungsional dari *software*. Karena uji coba *black box* memungkinkan pengembangan *software* untuk membuat himpunan kondisi input yang akan melatih seluruh syarat-syarat fungsional suatu program. Uji coba *black box* bukan merupakan alternatif dari uji coba *white box*, tetapi merupakan pendekatan yang melengkapi untuk menemukan kesalahan lainnya, selain menggunakan metode *white box*.(Mustaqbal et al., 2015).

2. Pengujian *White Box*

White Box Testing adalah salah satu cara untuk menguji suatu aplikasi atau *software* dengan cara melihat modul untuk dapat meneliti dan menganalisa kode dari program yang di buat terdapat kesalahan atau tidak. Kalau modul yang telah dan sudah di hasilkan berupa output yang tidak sesuai dengan yang di harapkan maka akan dikompilasi ulang dan di cek kembali kode-kode tersebut hingga sesuai dengan yang diharapkan.

Kasus yang sering menggunakan *white box testing* akan di uji dengan beberapa tahapan yaitu:

1. Pengujian seluruh keputusan yang menggunakan logikal.
2. Pengujian keseluruh loop yang ada sesuai batasan-batasannya.
3. Pengujian pada struktur data yang sifatnya internal dan yang terjamin validitasnya.

Kelebihan *White Box Test*ing antara lain:

1. Kesalahan Logika, menggunakan sintax '*if*' dan sintax pengulangan. Langkah selanjutnya metode *white box testing* ini akan mencari dan mendeteksi segala kondisi yang di percaya tidak sesuai dan mencari kapan suatu proses perulangan di akhiri.
2. Ketidaksesuaian Asumsi, menampilkan dan memonitor beberapa asumsi yang diyakini tidak sesuai dengan yang diharapkan atau yang akan diwujudkan, untuk selanjutnya akan dianalisa kembali dan kemudian diperbaiki.
3. Kesalahan Pengetikan, mendeteksi dan mencaribahasa-bahasa pemograman yang di anggap bersifat *case sensitif*.
4. Kelemahan *White Box Testing* adalah pada perangkat lunak yang jenisnya besar, metode *white box testing* ini dianggap boros karena melibatkan banyak sumber daya untuk melakukannya,(Mustaqbal et al., 2015).