BAB II

TINJAUAN PUSTAKA

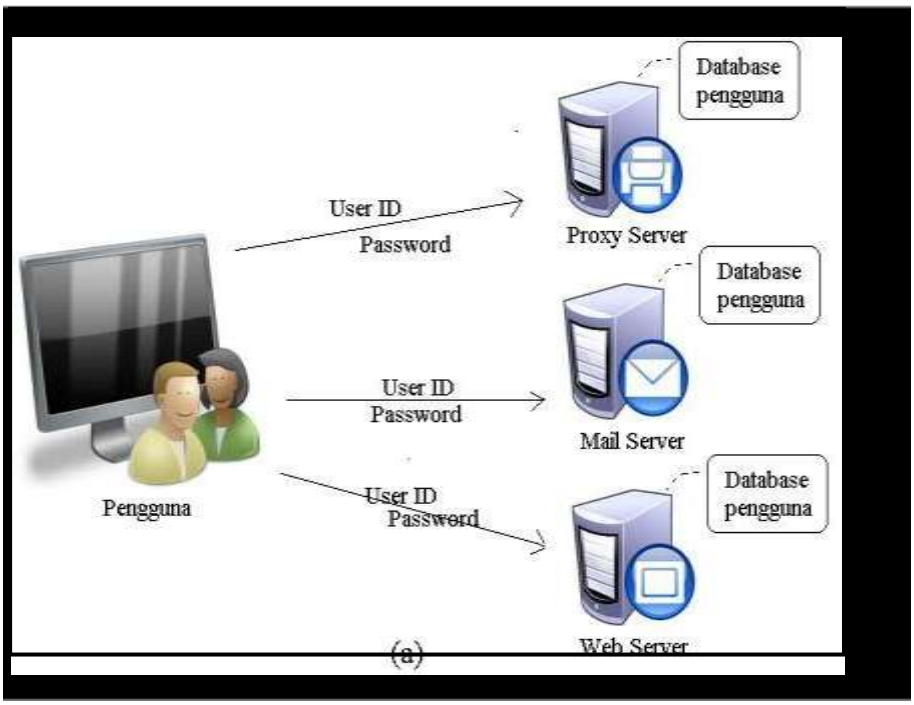
II.1. *Single Sign On* (SSO)

SSO merupakan teknologi yang memiliki kemampuan untuk memasukkan *id* dan *password* yang sama untuk *login* ke beberapa aplikasi dalam suatu perusahaan. Seperti *password* adalah mekanisme otentikasi paling aman, *SSO* kini telah dikenal sebagai *reduced sign on* (*RSO*) sejak lebih dari satu jenis mekanisme otentikasi yang digunakan sesuai dengan model risiko perusahaan(Rudy ; 2009 : 12).

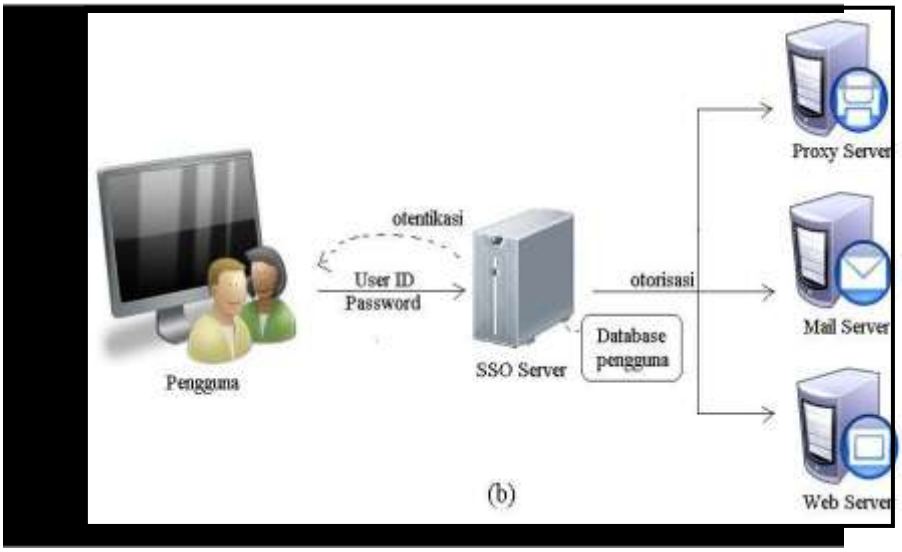
Untuk jaringan yang sangat besar dan bersifat heterogen, dimana pengguna diminta untuk mengisi informasi dirinya pada setiap aplikasi yang hendak di akses dibutuhkan *SSO*. Sistem *SSO* tidak memerlukan interaksi yang manual, untuk mengakses seluruh layanan aplikasi tanpa harus melakukan *login* dan mengetikan *password*-nya berulang kali.

SSO meng-otentikasi pengguna pada semua aplikasi yang telah di- *authorized* untuk diakses. Ini menghilangkan permintaan *authentication* lagi ketika pengguna mengganti aplikasi selama *session* berlaku. *SSO* juga memperkenankan informasi autentikasi dan mengidentifikasi subjek secara ketat guna menghindari *login* ganda pada sistem atau kelompok sistem yang terpercaya. Sistem *SSO* juga dapat memusatkan penglolaan dari parameter sistem yang relevan pada saat bersamaan dan meningkatkan penggunaan secara keseluruhan. Pengguna layanan bisa lebih menyukai sistem *SSO* dari pada sistem

sign-on biasa. Gambaran perbedaan sistem *SSO* dengan sistem *sign-on* dapat dilihat pada gambar II.1. berikut.



Gambar II.1 Sistem *Sign On*
(Sumber : E-book Universitas Sumatera)



Gambar II.2. Gambaran Sistem *SSO*
(Sumber : E-book Universitas Sumatera)

Arsitektur Sistem *SSO* memiliki dua bagian utama yaitu *agent* yang berada di *web server* / layanan aplikasi dan sebuah *server SSO* berdedikasi yang akan dijelaskan sebagai berikut:

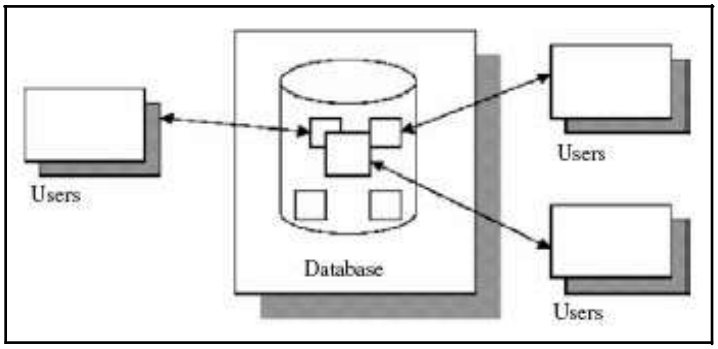
1. *Agent* : permintaan setiap *HTTP* yang masuk ke *web server* akan diterjemahkan oleh *agent*. Di tiap-tiap *web server* ada satu *agent* sebagai *host* dari layanan aplikasi. *Agent* ini akan berinteraksi pada *server SSO* dari sisi lain aplikasi dan berinteraksi dengan *web browser* dari sisi pengguna.
2. *SSO server* : Dalam menyediakan fungsi manajemen sesi *cookies* temporer (sementara) menggunakan *server SSO*. *User-id, session creation time, session expiration time* dan lain sebagainya adalah informasi ada pada *cookies.*

Produk-produk sistem *SSO* yang berbasis *open source* yang umum digunakan pada saat ini adalah *CAS, OpenAM* (*Open Access Manager*), dan *JOSSO* (*Java Open Single Sign-On*).

II.2. Basis Data (Database)

Menurut Esakkirajan dan Sumathi (2007), suatu basis data adalah suatu koleksi data yang terorganisir dengan baik yang terkait dengan cara yang penuh arti, yang dapat diakses di permintaan *logical* yang berbeda. Sistem basis data adalah sistem yang interpretasi dan penyimpanan informasi merupakan hal yang utama. Secara umum, karakteristik sebuah sistem basis data yaitu basis data

seharusnya memuat semua data yang diperlukan oleh organisasi sebagai hasil, volume data yang sangat besar, kebutuhan untuk penyimpanan data dalam waktu yang lama, dan akses data oleh sejumlah besar *user.* Gambar sistem basis data sederhana terlihat di gambar II.3. Dari gambar tersebut, terlihat jelas banyak *user* bisa mengakses data di dalam sebuah organisasi yang integrasi datanya seharusnya tetap dipelihara. Sebuah basis data terintegrasi ketika informasi yang sama tidak terekam di dua tempat.



Gambar II.3. Sistem Basis Data Sederhana
(Sumber : Esakkirajan dan Sumathi, 2007)

Sistem basis data juga dapat diartikan sebagai suatu sistem penyimpanan *record* terkomputerisasi dengan tujuan utama adalah perawatan dan penyediaan informasi pada saat yang dibutuhkan. Sistem basis data dirancang untuk mengelola informasi, pengelolaan informasi dan mekanisme untuk manipulasi informasi (Prayitno ; 2007 : 14). basis data terdiri dari file fisik yang disimpan pada komputer ketika menginstal *software* basis data.

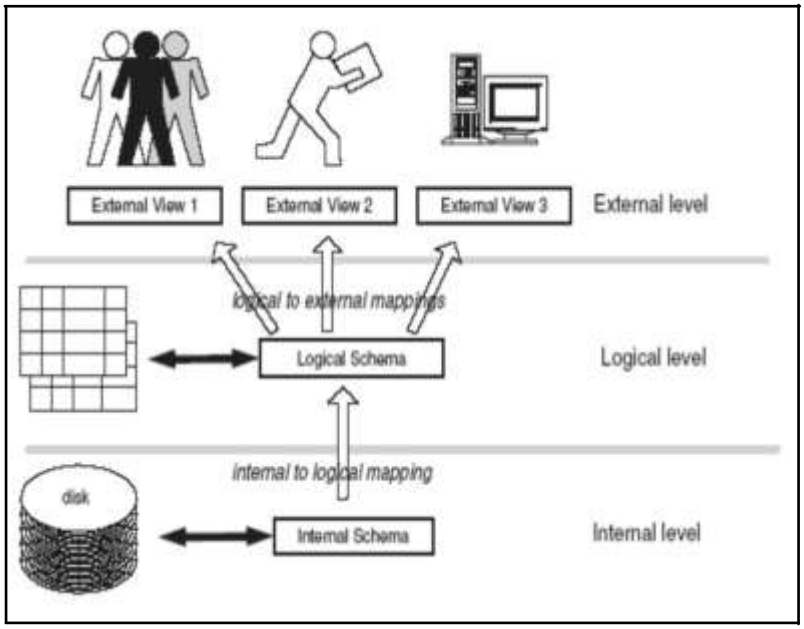
Basis data merupakan objek yang terstruktur. Objek tersebut disusun oleh data dan metadata. Data merupakan informasi deskriptif yang disimpan, sebagai contoh apabila sebuah basis data menyimpan data pelanggan maka data dapat berupa nama dan alamat pelanggan. Metadata menjelaskan struktur yang

digunakan basis data baik berupa kolom yang akan disediakan, panjang tiap kolom, dan tipe data.

Secara fisik, pengelolaan basis data tidak dilakukan langsung oleh pengguna namun ditangani oleh suatu sistem perangkat lunak. *basis data Management System* (DBMS) merupakan perangkat lunak yang memberikan layanan kepada *user* dalam mendefinisikan, membuat, memelihara *basis data*, dan menyediakan akses terkontrol terhadap data sehingga memudahkan *user* untuk mengorganisasi, menyimpan, mengubah dan mengambil kembali data.

Dalam perkembangannya, perusahaan pengembang DBMS menetapkan standar sendiri, sehingga DBMS yang tersedia berbeda format dan model. Akan tetapi pada tahun 1978 ditetapkan arsitektur dasar DBMS yang dikeluarkan oleh SPARC (*Standards Planning and Requierement Committee*) (Esakkirajan dan Sumathi ; 2007 : 10). Pada arsitektur ini sistem basis data dibagi menjadi tiga tingkat seperti gambar II.4. yaitu:

1. Tingkat *physical* atau *internal* yang terfokus pada cara data disimpan dalam *hardware* secara fisik.
2. Tingkat *logical* atau *conceptual* yang berisi definisi basis data, berupa model data dan skema diagram basis data.
3. Tingkat *view* atau *external* yang terfokus kepada pengguna. Pada tingkatan ini, pengguna dibagi menjadi beberapa tingkatan, masing-masing tingkatan memiliki hak akses yang berbeda terhadap basis data.



Gambar II.4. Tingkatan Sistem Basis Data
(Sumber : Esakkirajan dan Sumathi, 2007)

II.3. PHP

Menurut kamus komputer, PHP adalah bahasa pemrograman untuk dijalankan melalui halaman web, umumnya digunakan untuk mengolah informasi di internet. Sedangkan dalam pengertian lain, PHP adalah singkatan dari PHP *Hypertext Preprocessor* yaitu bahasa pemrograman web *server-side* yang bersifat *open source* atau gratis. PHP merupakan *script* yang menyatu dengan HTML dan berada pada server (*server side HTML embedded scripting*) (Rulianto Kurniawan; 2010: 2).

II.4. MySQL

MySQL adalah salah satu jenis database server yang sangat terkenal. MySQL termasuk jenis RDBMS (*Relational Database Management System*). MySQL ini mendukung bahasa pemrograman PHP. MySQL juga mempunyai *query* atau bahasa SQL (*Structured Query Language*) yang simpel dan menggunakan *escape character* yang sama dengan PHP (Rulianto Kurniawan; 2010: 16).

II.5. UML (Unified Modelling Language)

UML (*Unified Modeling Language*) adalah sebuah ”bahasa” yang telah menjadi standar dalam industry untuk visualisasi, merancang dan mendokumentasikan sistem piranti lunak. UML menawarkan sebuah standar untuk merancang model sebuah sistem. Seperti bahasa-bahasa lainnya, UML mendefenisikan notasi dan *sintax/semantic*. Notasi UML merupakan sekumpulan bentuk khusus untuk menggambarkan berbagai diagram piranti lunak. Setiap bentuk memiliki makna tertentu, dan UML *sintax* mendefinisikan bagaimana bentuk – bentuk tersebut dapat dikombinasikan.

Unified Modeling Language biasa digunakan untuk :

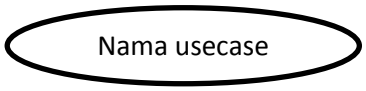
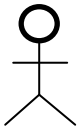
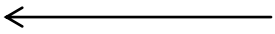
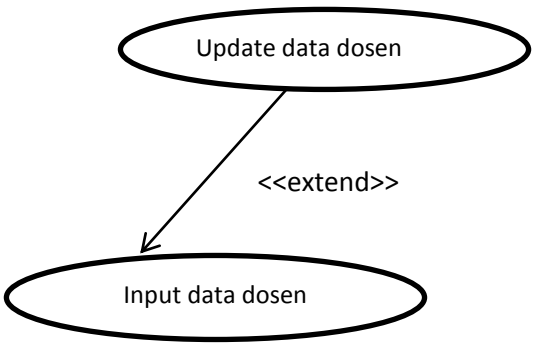
1. Menggambarkan batasan sistem dan fungsi-fungsi sistem secara umum, di buat dengan *use case* dan *actor*.
2. Menggambarkan kegiatan atau proses bisnis yang di laksanakan secara umum, di buat dengan *interaction diagrams*.

3. Menggambarkan representasi struktur *static* sebuah sistem dalam bentuk *class diagram*s.
4. Membuat model behavior "yang menggambarkan kebiasaan atau sifat sebuah sistem" dengan *state transition diagrams*.
5. Menyatakan arsitektur implementasi fisik menggunakan *component and development diagrams*.
6. Menyampaikan atau memperluas *fungsionality* dengan *stereotypes*.

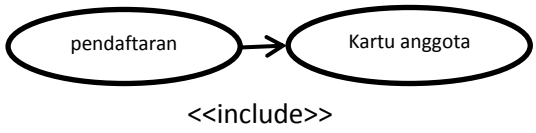
II.5.1. Use Case Diagram

Use case diagrams merupakan pemodelan untuk menggambarkan kelakuan (*behavior*) sistem yang akan dibuat. Diagram *use case* mendeskripsikan sebuah interaksi antara satu atau lebih *actor* dengan sistem yang akan dibuat. Dengan pengertian yang cepat, diagram *use case* digunakan untuk mengetahui fungsi apa saja yang ada didalam sebuah sistem dan siapa saja yang berhak menggunakan fungsi – fungsi tersebut. Terdapat beberapa simbol dalam menggambarkan diagram *use case*, yaitu *use case*, *actor* dan relasi. Berikut adalah sismbol – simbol yang ada pada diagram *use case* :

Tabel II.1 Simbol-Simbol Pada *Use Case*

Simbol	Deskripsi
Use case 	Fungsionalitas yang disediakan sistem sebagai unit – unit yang saling bertukar pesan antar unit atau *actor*; biasanya ditanyakan dengan menggunakan kata kerja di awal frase nama *use case*.
Aktor  nama aktor	Orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari *acto*r adalah gambar orang, tapi *actor* belum tentu merupakan orang; biasanya dinyatakan menggunakan kata benda diawal frase nama *actor*.
Asosiasi/ *association* 	Komunikasi antara actor dan use case yang berpartisipasi pada use case atau use case memiliki interaksi dengan kator.
Extend  <<extend>>	Relasi use case tambahan ke sebuah use case dimana use case yang ditambahkan dapat berdiri sendiri walau tanpa use case tambahan itu; mirip dengan prinsip inheritance pada pemrograman berorientasi objek; biasanya use case tambahan memiliki nama depan yang sama dengan use case yang ditambahkan, arah panah menunjuk pada use case yang dituju. Contoh : 

Tabel II.1 Simbol-Simbol pada *Use Case* (Lanjutan)

Simbol	Deskprisi
Include <<include>> 	Relasi use case tambahan kesebuah use case dimana use case yang yang ditambahkan memerlukan use case ini untuk menjalankan fungsinya atau sebagai syarat dijalankan use case ini. Ada dua sudut pandang yang cukup besar mengenai include di use case, *include* berarti use case yang ditambahkan akan selalu dipanggil saat use case tambahan dijalankan, contoh : 

Sumber: (Yuni Sugiarti; 2013,42)

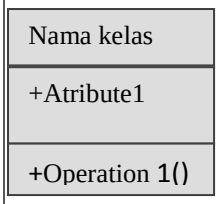
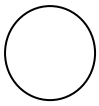
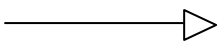
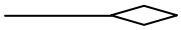
II.5.2. Class Diagram

Diagram kelas atau class diagram menggambarkan struktur sistem dari segi pendefenisian kelas-kelas yang akan di buat untuk membangun sistem. Kelas memiliki apa yang di sebut atribut dan metode atau operasi.

1. Atribut merupakan variabel- variabel yang di miliki oleh suatu kelas.
2. Atribut mendeskripsikan properti dengan sebaris teks di dalam kotak kelas tersebut.
3. Operasi atau metode adalah fungsi-fungsi yang di miliki oleh suatu kelas.

Diagram kelas mendeskripsikan jenis-jenis objek dalam sistem dan berbagai hubungan statis yang terdapat di antara mereka. Diagram kelas juga menunjukkan properti dan operasi sebuah kelas dan batasan-batasan yang terdapat dalam hubungan-hubungan objek tersebut.

Tabel II.2 Simbol – simbol Class Diagram

Simbol	Deskripsi
Package 	Package merupakan sebuah bungkusan dari satu atau lebih kelas
Operasi 	Kelas pada struktur sistem
Antarmuka / interface  **Interface**	Sama dengan konsep *interface* dalam pemrograman berorientasi objek
Asosiasi 1 1..*	Relasi antar kelas dengan makna umum, asosiasi biasanya juga disertai dengan *multiplicity.*
Asosiasi berarah/directed asosiasi 	Relasi antar kelas dengan makna kelas yang satu di gunakan oleh kelas yang lain, asosiasi biasanya juga di sertai dengan *multiplicity*.
Generalisasi 	Relasi antar kelas dengan makna generalisasi – spesialisasi (umum khusus).
Kebergantungan / defedency	Relasi antar kelas dengan makna kebergantungan antar kelas
Agregasi 	Relasi antar kelas dengan makna semua bagian (*whole-part*)

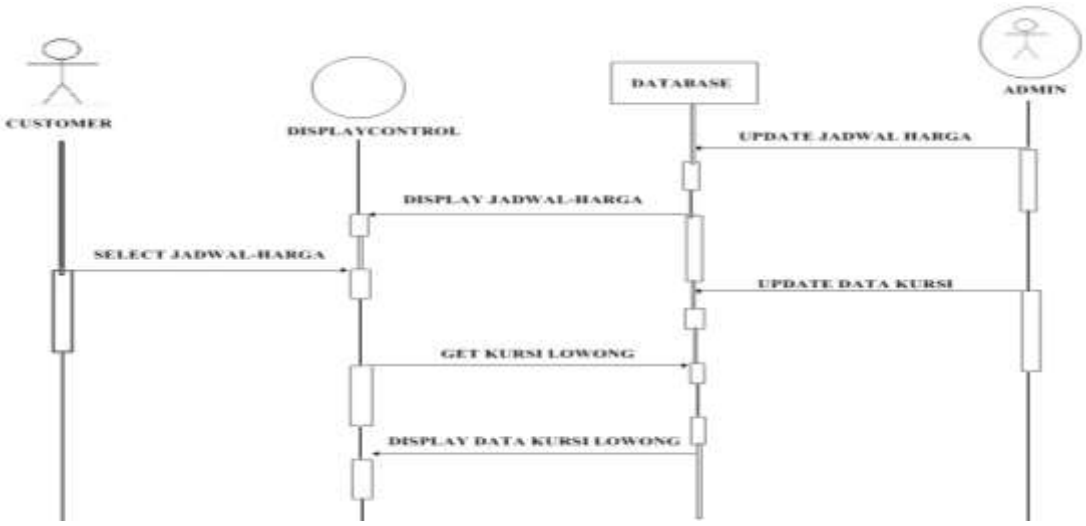
Sumber: (Yuni Sugiarti; 2013,59)

II.5.3. Sequence Diagram

Diagram sekuence menggambarkan kelakuan/ pelaku objek pada *use case* dengan mendeskripsikan waktu hidup objek dan *message* yang dikirimkan dan diterima antar objek. Oleh karena itu untuk menggambarkan diagram sequence maka harus diketahui objek-objek yang terlibat dalam sebuah *use case* beserta metode -metode yang dimiliki kelas yang diinstasiasi menjadi objek itu.

Diagram *sequence* memiliki ciri yang berbeda dengan diagram interaksi pada diagram kolaborasi sebagai berikut :

1. Pada diagram *sequence* terdapat garis hidup objek. Garis hidup objek adalah garis *vertical* yang mencerminkan eksistensi sebuah objek sepanjang periode waktu. Sebagian besar objek-objek yang tercakup dalam diagram interaksi akan eksis sepanjang durasi tertentu dari interaksi, sehingga objek-objek itu diletakkan dibagian atas diagram dengan garis hidup tergambar dari atas hingga bagian bawah diagram. Suatu objek lain dapat saja diciptakan, dalam hal ini garis hidup dimulai saat pesan *destroy*, jika kasus ini terjadi, maka garis hidupnya juga berakhir.
2. Terdapat focus kendali (*Focus Of Control*), berupa empat persegi panjang ramping dan tinggi yang menampilkan aksi suatu objek secara langsung atau sepanjang sub ordinat. Puncak dari empat persegi panjang adalah permulaan aksi, bagian dasar adalah akhir dari suatu aksi. Pada diagram ini mungkin juga memperhatikan penyaringan (*nesting*) dan *focus* kendali yang disebabkan oleh proses rekursif dengan menumpuk *focus* kendali yang lain pada induknya.



Gambar II.5. Contoh *Sequence Diagram*

(Sumber: Yuni Sugiarti,2013,71)

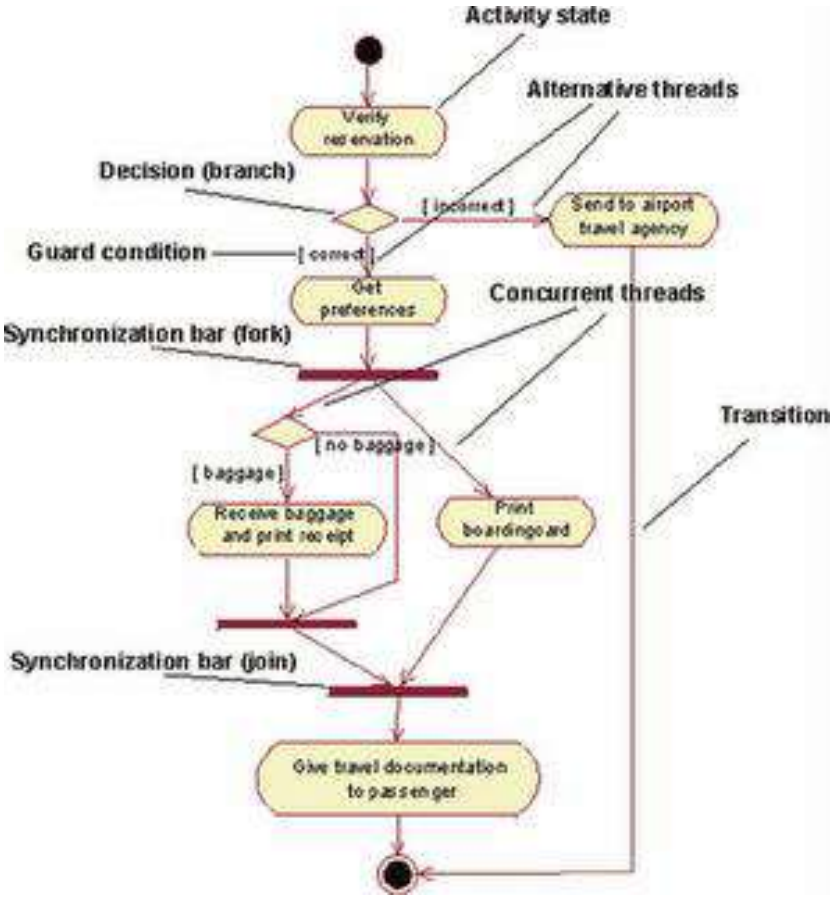
II.5.4. Activity Diagram (Aktivitas)

Diagram aktivitas atau *activity* diagram menggambarkan *workflow* (aliran kerja) atau aktifitas dari sebuah sistem atau proses bisnis.

Activity diagram merupakan *state* diagram khusus, di mana sebagian besar state adalah action dan sebagian besar transisi di-trigger oleh selesainya state sebelumnya (*internal processing*). Oleh karena itu *activity* diagram tidak menggambarkan behaviour internal sebuah sistem (dan interaksi antar subsistem) secara eksak, tetapi lebih menggambarkan proses-proses dan jalur-jalur aktivitas dari level atas secara umum.

Sebuah aktivitas dapat direalisasikan oleh satu *use case* atau lebih. Aktivitas menggambarkan proses yang berjalan, sementara *use case*

menggambarkan bagaimana aktor menggunakan sistem untuk melakukan aktivitas.



Gambar II.6. Activity Diagram (Aktivitas)

(Sumber: Jurnal, Memahami Penggunaan UML,2011)

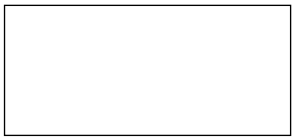
II.6. ERD (Entity Relationship Diagram)

Entity relationship diagram adalah pemodelan basis data yang fokusnya pada notasi yang umum di pergunakan oleh analisis sistem,yaitu simbol entitas,relasi dan garis penghubung. Elemen – elemen diagram hubungan entitas yaitu

1. Entitas (*Entity*)

   Entitas merupakan notasi untuk mewakili suatu objek dengan karateristik sama,yang di lengkapi dengan atribut,sehingga pada suatu lingkungan nyata setiap objek akan berbeda dengan objek lainnya.Pada umumnya,objek dapat berupa benda,pekerjaan dan orang.

   .

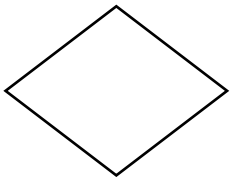


Gambar II.7. Simbol *Entitas*

(Sumber : Yudi Priadi,2013,20)

2. Relasi (*Relationship*)

   Relasi adalah notasi yang di gunakan untuk menghubungkan beberapa entitas berdasarkan fakta pada suatu lingkungan.

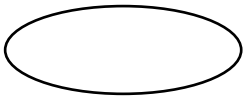


Gambar II.8. Simbol *Relationship*

(Sumber : Yudi Priadi,2013,20)

3. Atribut (*Attribute*)

   Atribut adalah notasi yang menjelaskan karateristik suatu entitas dan juga relasinya.Attribut dapat sebagai key yang bersifat unik,yaitu primary key dan foreign key.



Gambar II.9. Simbol *Attribute*

(Sumber : Yudi Priadi,2013,20)

4. Garis Penghubung adalah untuk mengkaitkan keterkaitan antara notasi notasi yang di gunakan dalam Diagram E-R, yaitu:itu entitas,relasi dan atribut.

Gambar II.10. Simbol Garis Penghubung

(Sumber : Yudi Priadi,2013,20)

II.6.1 Normalisasi

Normalisasi adalah bagian perancangan basis data. Tanpa normalisasi, sistem basis data menjadi tidak akurat, lambat, tidak efisien, serta tidak memberikan data yang diharapkan. Pada waktu menormalisasikan basis data, ada empat tujuan yang harus dicapai,.

1. Mengatur data dalam kelompok-kelompok sehingga masing-masing kelompok hanya mengenai bagian kecil sistem.
2. Meminimalkan jumlah data berulang dalam basis data.
3. Membuat basis data yang datanya diakses dan dimanipulasi secara cepat dan efisien tanpa melupakan integritas data.
4. Mengatur data sedemikian rupa sehingga ketika memodifikasi data, anda hanya mengubah pada satu tempat.

II.7. Kamus Data

Kamus data (*data dictionary*) dipergunakan untuk memperjelas aliran data yang digambarkan pada DFD. Kamus data adalah kumpulan daftar elemen data

yang mengalir pada sistem perangkat lunak sehingga masukan (*input*) dan keluaran (*ouput*) dapat dipahami secara umum (memiliki standar cara penulisan). Kamus data biasanya berisi :

1. Nama-nama dari data.
2. Digunakan pada merupakan proses-proses yang terkait data.
3. Deskripsi merupakan deskripsi data
4. Informasi tambahan, seperti tipe data, nilai data, batas nilai data, dan komponen yang membentuk data.