BAB 2

LANDASAN TEORI

2.1. Rekayasa Piranti Lunak

2.1.1. Definisi Piranti Lunak

Menurut Pressman (2005, p36), piranti lunak dapat diartikan sebagai berikut :

1. Perintah-perintah dalam program komputer yang ketika dieksekusi menyediakan fitur, fungsi, dan kinerja yang diinginkan.
2. Susunan data yang memungkinkan program memanipulasi data.
3. Dokumen-dokumen yang mendeskripsikan operasi dan penggunaan dari program.

Menurut Pressman (2005, p37), ada beberapa karakteristik dari piranti lunak yang membedakannya dari piranti keras, yaitu :

1. Piranti lunak dikembangkan dan direkayasa, bukan dirakit seperti piranti keras.
2. Piranti lunak tidak mengalami kerusakan secara fisik. Piranti keras mengalami penurunan kualitas dalam periode waktu tertentu. Tingkat kerusakan dapat bertambah jika piranti keras mendapat pengaruh dari lingkungan seperti debu, getaran, suhu yang tidak sesuai, dan gangguan lingkungan lainnya. Sedangkan

pada piranti lunak biasanya kerusakan terjadi akibat kesalahan pemrograman.

3. Piranti lunak dirancang ke dalam komponen-komponen yang dapat digunakan kembali. Sehingga *progammer* dapat berkonsentrasi untuk mengembangkan komponen-komponen yang baru.

2.1.2. Rekayasa Piranti Lunak

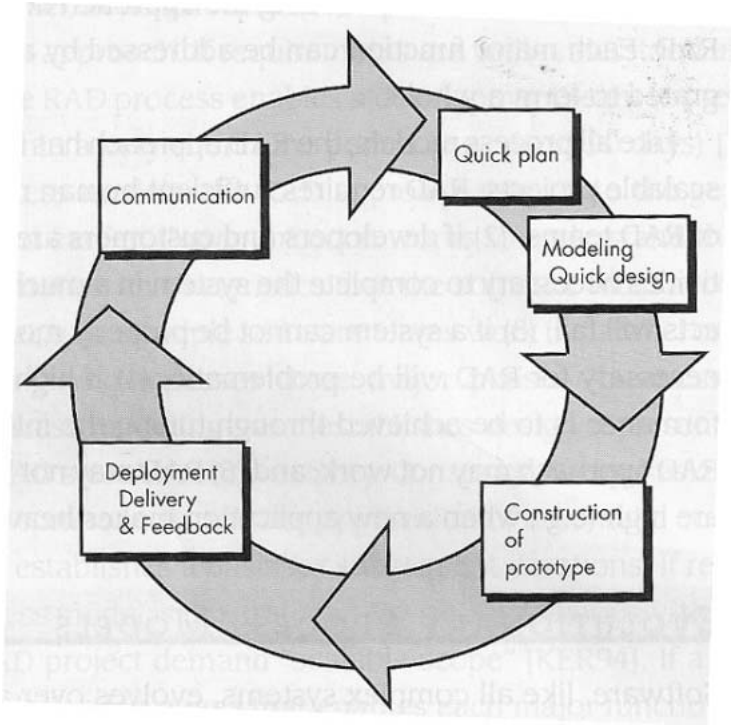
Menurut Pressman (2005, p53), rekayasa piranti lunak adalah pembentukan dan penggunaan prinsip-prinsip rekayasa untuk memperoleh perangkat lunak yang handal dan bekerja efisien pada mesin nyata secara ekonomis.

2.1.3. *Prototyping*

Menurut Pressman (2005, p83), salah satu metode pengembangan piranti lunak adalah dengan menggunakan model *prototyping*. Model ini biasanya digunakan jika *customer* hanya memberikan tujuan pembuatan piranti lunak secara umum, tidak mendefinisikan *input*, proses, dan *output* secara detail.

Prototyping dimulai dengan komunikasi antara pengembang dengan *customer* untuk mediskusikan tujuan secara keseluruhan dari piranti lunak dan mengidentifikasi kebutuhan-kebutuhan yang harus ada

pada piranti lunak. Kemudian pengembang membuat rancangan piranti lunak secara cepat dengan fokus pada aspek piranti lunak yang akan terlihat oleh pengguna. Rancangan tersebut kemudian dibangun menjadi *prototype*. *Prototype* kemudian dibawa kepada *customer* untuk dievaluasi. Hasil evaluasi dari *customer* kemudian digunakan untuk proses pengembangan selanjutnya. Proses ini terus berulang sampai piranti lunak mencapai tahap akhir.



Gambar 2.1 Prototyping Model

Sumber : Pressman (2005, p84)

2.1.4. *Project Scheduling*

Menurut Pressman (2005, p716), *project scheduling* bertujuan supaya memungkinkan *poject manager* untuk mendefinisikan tugas-tugas yang harus dikerjakan, menentukan ketergantungannya, dan mengembangkan sejumlah grafik, diagram, tabel yang bisa membantu *tracking* dan kontrol dari *software project.* Penjadwalan ini dapat ditampilkan dengan sebuah bagan yang dikenal dengan *gantt chart.*

2.2. Analisis dan Perancangan Sistem Visualisasi dan *Monitoring*

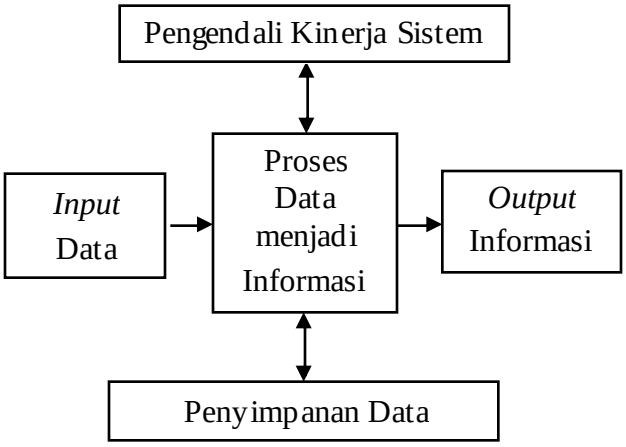
2.2.1. Sistem

Menurut O'Brien (2005, p22), sistem adalah sekelompok komponen yang saling bekerja sama untuk tujuan tertentu dengan menerima *input* dan menghasilkan *output* dalam proses transformasi yang terorganisasi.

Sistem memiliki tiga komponen dasar yaitu

- *Input,* elemen-elemen yang masuk ke dalam sistem untuk diproses.
- Proses, melibatkan proses transformasi yang mengkonversi *input* menjadi *output.*

- *Output,* mentransfer *output* melibatkan unsur-unsur yang telah dihasilkan oleh proses tranformasi dengan tujuan akhir.



Gambar 2.2 Elemen-Elemen Sistem

Sumber : O'Brien (2005, p24)

2.2.2. Analisis Sistem

Menurut Whitten et al. (2004, p176), analisis sistem adalah sebuah teknik pemecahan masalah yang menguraikan sebuah sistem menjadi bagian-bagian komponen dengan tujuan mempelajari seberapa bagus bagian-bagian komponen tersebut bekerja dan berinteraksi untuk meraih tujuan mereka. Sedangkan O'Brien (2005, p348) menjelaskan bahwa analisis sistem adalah kajian secara mendalam tentang kebutuhan informasi pengguna (*user*) yang menghasilkan persyaratan fungsional (*functional requirements*) yang digunakan sebagai dasar untuk desain sistem informasi baru.

Analisis sistem menggambarkan apa yang harus dilakukan untuk memenuhi kebutuhan informasi pengguna. Analisis sistem melibatkan beberapa hal antara lain :

- Kebutuhan informasi pengguna.
- Kegiatan, sumber daya, dan produk dari satu atau lebih dari sistem informasi yang sekarang sedang digunakan.
- Kemampuan sistem informasi yang diperlukan untuk memenuhi kebutuhan informasi pengguna, dan orang yang berkepentingan sebagai pengguna sistem.

2.2.3. Perancangan Sistem

Menurut Whitten et al. (2004, p176) mendefinisikan perancangan sistem sebagai subuah teknik pemecahan masalah yang saling melengkapi dengan analisis sistem yang merangkai kembali bagian-bagian komponen menjadi sebuah sistem yang lengkap yang lebih baik dari sebelumnya. Hal tersebut melibatkan penambahan, penghapusan, dan perubahan bagian-bagian relatif pada sistem aslinya/awalnya.

O'Brien (2005, p350) menjelaskan bahwa perancangan sistem menentukan bagaimana sistem akan mencapai tujuannya. Perancangan sistem terdiri dari kegiatan yang menghasilkan spesifikasi sistem yang memenuhi kebutuhan fungsional yang dikembangkan dalam proses analisis sistem.

2.2.4. Visualisasi

Menurut KBBI, visualisasi adalah pengungkapan suatu gagasan atau perasaan dengan menggunakan bentuk gambar, tulisan (kata dan angka), peta, grafik, dsb.

2.2.5. *Monitoring*

Menurut KBBI, *monitoring* (memantau) adalah mengawasi, mengamati atau mengecek dengan cermat, terutama untuk tujuan khusus. *Monitoring* dapat diartikan juga sebagai mengatur atau mengontrol kerja suatu mesin.

2.3. *Database*

Menurut Connolly dan Begg (2005, p15), *database* adalah kumpulan data yang terhubung secara logis, dan rincian dari data tersebut, yang dirancang untuk memenuhi kebutuhan informasi dari suatu organisasi.

2.3.1. *Database Management System* (DBMS)

Menurut Connolly dan Begg (2005, p16-17), DBMS adalah sebuah piranti lunak yang memungkinkan pengguna untuk mendefinisikan, membuat, mengelola, dan mengatur akses ke *database*.

Pada umumnya DBMS menyediakan beberapa fasilitas sebagai berikut :

1. Memungkinkan pengguna mendefinisikan *database*, biasanya melalui *Data Definition Language* (DDL). DDL memungkinkan pengguna untuk mendefinisikan tipe data dan struktur dan batasan dari data yang akan dimasukkan ke dalam *database*.
2. Memungkinkan pengguna untuk memasukkan, mengubah, menghapus, dan menerima data dari *database*, biasanya melalui *Data Manipulation Language* (DML). *Structured Query Language* (SQL) adalah bahasa *query* (*query language)* yang paling umum digunakan pada saat ini untuk DBMS relasional.
3. Menyediakan akses terkendali kepada *database*. Contohnya sistem keamanan dan sistem *recovery*.

Menurut Connolly dan Begg (2005, p18-21), ada lima komponen dari DBMS, yaitu :

1. Perangkat keras.

   Untuk dapat menjalankan DBMS dan aplikasi memerlukan perangkat keras. Sebuah DBMS memerlukan beberapa persyaratan minumum tertentu untuk perangkat keras yang akan digunakan agar DBMS dapat berjalan dengan baik.

2. Perangkat lunak.

   Termasuk di dalamnya DBMS itu sendiri, aplikasi, sistem operasi, dan perangkat lunak jaringan jika DBMS digunakan melalui sebuah jaringan.

3. Data.

   Bagian terpenting dari suatu *database*, khususnya dari sisi pengguna. Data berfungsi sebagai jembatan antara komponen manusia dan komponen mesin.

4. Prosedur.

   Instruksi dan aturan yang mengatur desain dan kegunaan dari *database*. Pengguna sistem dan karyawan yang mengatur *database* memerlukan prosedur yang terdokumentasi bagaimana menggunakan atau menjalankan sistem.

5. Manusia.

   Orang-orang yang terlibat dalam sistem tersebut.

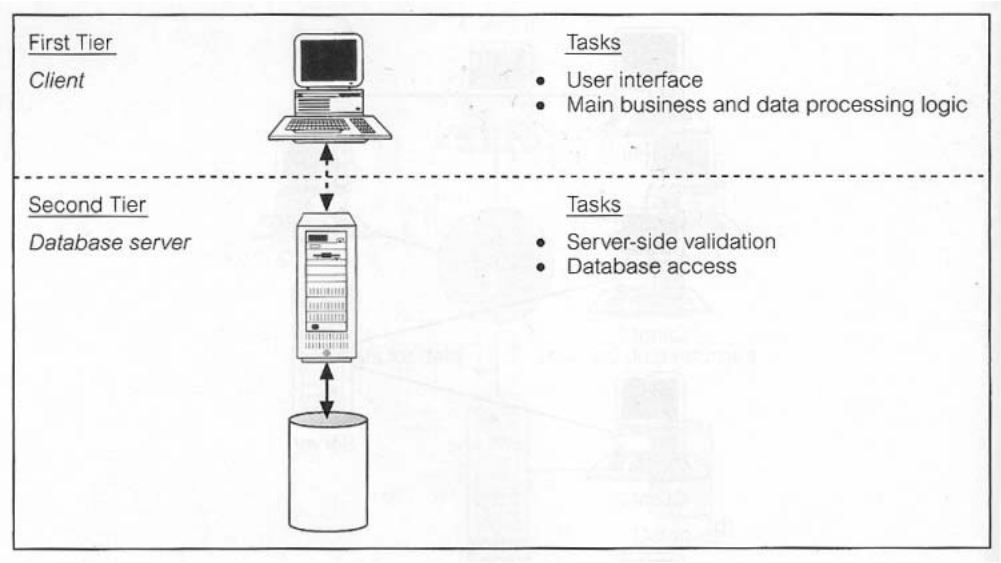
Menurut Connolly dan Begg (2005, p56), arsitektur *database* untuk *multi-user* secara umum dibagi menjadi tiga jenis, yaitu :

1. *One-Tier Client-Server Architecture.*

   Pengguna dan *database* menggunakan satu mesin yang sama.

2. *Two-Tier Client-Server Architecture.*

   Terdiri dari pengguna sebagai *first tier* dan *database server* sebagai *second tier*.

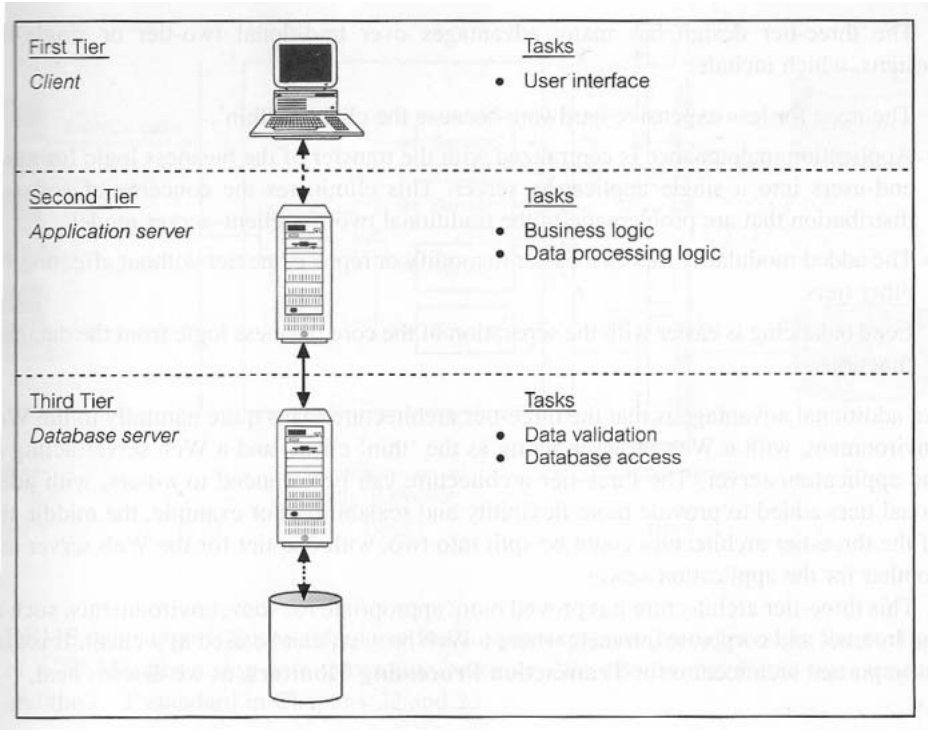


Gambar 2.3 *Two-Tier Client-Server Architecture*

(Sumber : Connolly dan Begg, 2005, p60)

3. *Three-Tier Client-Server Architecture.*

   Terdiri dari pengguna sebagai *first tier*, *application server* sebagai *second tier*, dan *database server* sebagai *third tier*.



Gambar 2.4 *Three-Tier Client-Server Architecture*

(Sumber : Connolly dan Begg, 2005, p61)

(Bennett et al., 2006, p505) Sejak mulai tahun 70-an, sistem yang digunakan oleh beberapa perusahaan telah menggunakan *database* sebagai tempat penyimpanan data. *Database Management System* (DBMS) adalah *tool* untuk mengatur tugas-tugas yang berkaitan dengan penyimpanan, penyediaan dan pengaksesan data.

(Bennett et al., 2006, p507) DBMS tidak hanya sekedar menyediakan tempat untuk penyimpanan data, tetapi juga dapat diakses oleh berbagai aplikasi yang berbeda. Fitur-fitur yang dapat digunakan untuk mengatur data dalam DBMS antara lain :

- *Data definition language* (DDL). DDL digunakan untuk mendefinisikan data (membuat *database*, membuat tabel) yang berada di dalam *database* dan struktur data yang digunakan di dalam suatu *database*.
- *Data manipulation language* (DML). DML digunakan untuk memanipulasi data (menambahkan, mengubahkan, menghapus data) yang berada di dalam *database*.
- *Integrity constraints*. *Constraints* digunakan untuk memberikan suatu aturan pada data yang berada di dalam *database*.
- *Transaction management*. Dalam memanipulasi data, dapat diikuti dengan proses *commit* (data yang telah dimanipulasi, tidak dapat dikembalikan pada posisi *check point*) atau *roll back* (jika terdapat kegagalan dalam proses manipulasi data, dapat dilakukan *roll back*, mengembalikan pada posisi *check point*).
- *Concurrency*. Pengguna dapat menggunakan *database* dan memanipulasi / mengubah isi dari *database* tersebut secara berkesinambungan.
- *Security*. Dalam suatu *database*, terdapat pemberian hak akses pada pengguna untuk mengakses data di dalam suatu *database*. Pemberian hak akses tersebut antara lain *select, insert, update* dan *delete*.

- *Tuning of storage*. *Tool* yang digunakan untuk memonitor jalur pengaksesan data dan meningkatkan efesiensi pengaksesan struktur *schema* yang berada di dalam *database*. *Tuning* dapat dilakukan tanpa adanya pengaruh terhadap program aplikasi yang digunakan oleh pengguna.

Bagi sistem yang besar, dibutuhkan *Database Administrator* (DBA) untuk mengatur dan memastikan bahwa *database* berjalan dengan efesien. DBA akan bertanggung jawab untuk mengendalikan *data dictionary* (*schema* konseptual *database*), mengendalikan untuk siapa saja yang berhak mengakses data, dan untuk *tuning* performa *database*. Bagi sistem yang lebih kecil DBA tidak dibutuhkan, tetapi seseorang akan bertanggung jawab dalam pengaturan *database*.

Dalam *database*, terdapat tiga tipe *database* antara lain

- *Relational databases*. Berawal pada tahun 70an dan berkembang terus, sebelum tahun 1986 American National Standards Institue (ANSI) meluncurkan *Structured Query Language* (SQL). Sekarang, SQL telah menjadi bahasa standar yang menyediakan DDL dan DML bagi *relational databases*. Inti dari RDBMS adalah untuk mengeliminasi / menghilangkan *redundancy* data (data yang sama/*duplicate*) dan untuk menyediakan hasil representasi data secara logis dan sederhana. Hasil representasi data ditampilkan dengan tabel 2 dimensi. Setiap tabel terdiri dari baris-baris (*record*). Setiap baris memiliki nilai atribut

yang berada di setiap kolom (*field*). Isi dari data setiap memiliki tipe atribut yang sama. Setiap baris memiliki satu atau lebih atribut yang unik untuk mengidentifikasi/membedakan antar setiap baris. *Relational* DBMS adalah tipe DBMS yang paling sering digunakan karena telah terbukti kehandalannya yaitu efesien dan fleksibel untuk berbagai jenis data. RDBMS yang sering dikenal antara lain Oracle, MS SQL *Server*, MySQL.

- *Object databases*. *Object Database Management System* (ODBMS) dikembangkan untuk menampung objek-objek yang kompleks. Objek tersebut antara lain objek *multimedia* seperti suara, gambar, dan *video clips* tidak mudah direpresetasikan dalam tabel-tabel. ODBMS menyediakan kemampuan untuk menampung objek-objek yang kompleks. Contoh ODBMS antara lain Gemstone/S, Jasmine dan ObjectStore.
- *Object-relational databases* (*hybrid*). *Object-relational databases* merupakan gabungan antara efesiensi dan kesederhanaan *relational databases* dengan kemampuan penyimpanan objek yang kompleks dan hubungan antar *class* yang dimiliki oleh *object databases*. Contoh dari *object-relational databases* adalah PostgreSQL.

2.3.2. *Relational Database Management System*

Menurut Connolly dan Begg (2005, p69), *Relational Database Management System* saat ini telah dipakai oleh sebagian besar piranti lunak . Dalam model relasional semua data terstruktur dalam tabel-tabel. Setiap tabel memiliki nama dan terdiri dari kolom-kolom data. Setiap baris mengandung satu nilai tiap kolom. Keuntungan dari model relasional adalah kesederhanaan logika ini.

Menurut Connolly dan Begg (2005, p72-84, p356), terdapat beberapa istilah penting dalam relational model, antara lain :

1. Relasi : sebuah tabel dengan kolom dan baris.
2. Atribut : sebuah kolom yang diberi nama dari sebuah relasi.
3. *Domain* : sekumpulan nilai yang dapat digunakan untuk satu atau lebih atribut.
4. *Tuple* : sebuah baris dalam relasi.
5. *Relational database* : sekumpulan relasi yang telah ternormalisasi dengan nama relasi yang berbeda-beda.
6. *Relationship* : sekumpulan hubungan yang memiliki arti diantara jenis-jenis entitas.
7. *Primary key* : *candidate key* yang terpilih untuk mengidentifikasi baris secara unik dalam suatu relasi.
8. *Foreign key* : adalah sebuah atau sekumpulan atribut dalam suatu relasi yang sesuai dengan *candidate key* dari satu atau beberapa relasi.

9. *Multiplicity* : jumlah kejadian yang mungkin terjadi dari satu jenis entitas yang berhubungan dengan kejadian tunggal dari jenis entitas lain yang berhubungan melalui suatu *relationship*.

Menurut Connolly dan Begg (2005, p362), batasan-batasan *multiplicity* dapat diringkas seperti pada Tabel 2.1.

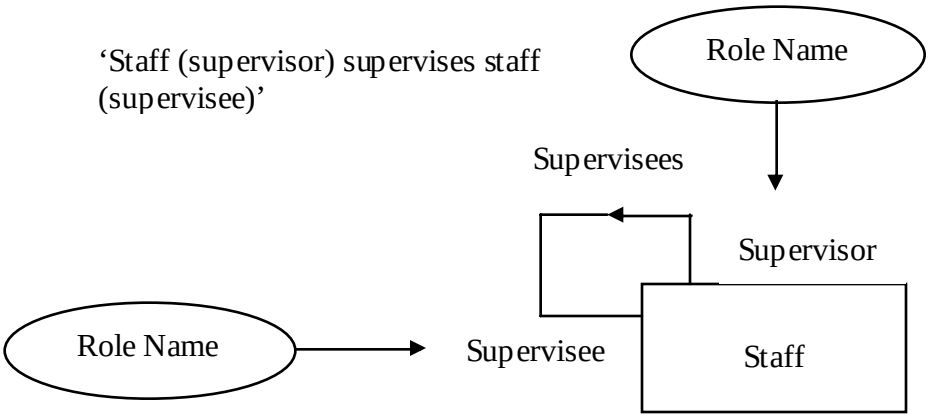
Tabel 2.1 Batasan *Multiplicity*

(Sumber : Connolly dan Begg, 2005, p362)

Batasan *Multiplicity*	**Arti**
0..1	Kejadian entitas nol atau satu.
1..1 (atau hanya 1)	Kejadian entitas tepat satu.
0..*	Kejadian entitas nol atau lebih.
1..*	Kejadian entitas satu atau lebih.
5..10	Kejadian entitas minimum lima sampai maksimum sepuluh.
0, 3, 6-8	Kejadian entitas nol atau tiga atau atau enam atau tujuh atau delapan.

2.3.3. *Recursive Relationship*

Menurut Connolly dan Begg (2005, p349), *recursive relationship* adalah hubungan tipe di mana tipe entitas yang sama terlibat lebih dari satu kali dalam peranan yang berbeda.



Gambar 2.5 Contoh *Recursive Relationship*

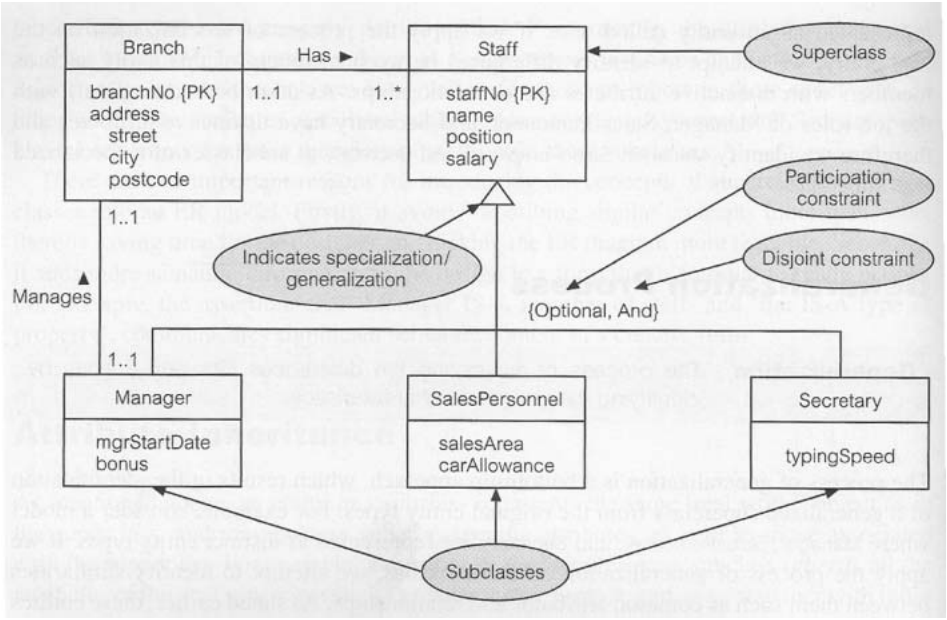
(Sumber : Connolly dan Begg, 2005, p349)

2.3.4. *Entity Relationship Modeling* (ERM)

Menurut Connolly dan Begg (2005, p291-292), ada dua macam pendekatan terhadap perancangan *database*, yaitu *bottom-up* dan *top-down*. Pendekatan *bottom-up* dimulai dengan mendefinisikan atribut-atribut mendasar yang berdasarkan hubungan antar atribut dikelompokan ke dalam relasi yang mewakili jenis entitas dan hubungan antar entitas.

Pendekatan ini melaui proses yang dinamakan dengan normalisasi. Sebaliknya pendekatan *top-down* dimulai dengan medefinisikan entitas-entitas yang berhubungan dengan organisasi dan hubungan antar entitas yang kemudian dijelaskan secara lebih rinci. Pendekatan ini dapat diilustrasikan menggunakan model *Entity-Relationship Diagram* (ERD).

Menurut Connolly dan Begg (2005, p342), *Entity Relationship Modeling* adalah pendekatan *top-down* dalam perancangan *database* yang dimulai dengan mengidentifikasi objek-objek penting yang dinamakan entitas dan *relationships* atau hubungan-hubungan antar objek yang harus ditampilkan dalam bentuk model. Selanjutnya ditambahkan rincian berupa informasi-informasi yang diperlukan dari entitas dan *relationships*.



Gambar 2.6 Contoh ERD

(Sumber : Connolly dan Begg, 2005, p376)

2.4. Infrastruktur *Database* dan Aplikasi

Menurut KBBI, infrastruktur adalah segala sesuatu yang merupakan penunjang utama terselenggaranya suatu proses (usaha, pembangunan, proyek, dsb): jalan dan angkutan merupakan-penting bagi pembangunan suatu daerah. Jadi dalam infrastruktur *database* dan aplikasi yang dimaksud dengan segala sesuatu adalah *database,* aplikasi dan jaringan yang ada.

2.5. Teknik Penemuan Fakta

Bennett et al. (2006, p132) menjelaskan terdapat lima teknik untuk menemukan fakta penelitian yang dilakukan oleh penganalisa. Lima teknik tersebut adalah membaca latar belakang (*background reading*), wawancara (*interview*), pengamatan langsung (*observation*), *document sampling*, kuesioner (*questionnaires*).

2.5.1. Wawancara (*Interview*)

(Bennett et al., 2006, p133) Wawancara adalah teknik penemuan fakta dengan cara melakukan pertemuan antara pewawancara dengan orang yang akan diwawancarai (anggota / *staff* dari organisasi tertentu yang akan dianalisa). Proses wawancara, dilakukan dengan memberikan sejumlah pertanyaan. Sejumlah pertanyaan tersebut dapat bersifat tetap ataupun fleksibel. Fleksibel artinya pertanyaan yang telah dipersiapkan

sebelumnya, dapat mengalami penambahan atau penggurangan sesuai dengan respon dari orang yang diwawancarai.

(Bennett et al., 2006, p135) Teknik wawancara memiliki kelebihan antara lain

- Si pewawancara dapat melakukan interaksi langsung dengan orang yang di wawancarai. Jika terdapat pertanyaan yang kurang dimengerti oleh orang yang diwawancarai, pewawancara dapat menjelaskan langsung maksud / arti dari pertanyaan tersebut sehingga teknik ini menghasilkan keakuratan informasi yang tinggi.
- Pewawancara dapat melakukan pendekatan langsung pada orang yang diwawancarai untuk memperoleh informasi yang bersifat internal.
- Jika diperlukan, pewawancara dapat memotong pembicaraan.

Teknik wawancara juga memiliki beberapa kekurangan antara lain

- Membutuhkan waktu dan biaya yang lebih besar.
- Pewawancara harus merangkum kembali dari hasil wawancara yang telah dilakukan (hasil rekaman, catatan hasil wawancara).
- Wawancara membutuhkan orang yang dapat mewawancarai dengan baik, memiliki pemikiran yang luas

sehingga proses wawancara menghasilkan data yang akurat.

2.5.2. Kuesioner (*Questionnaires*)

(Bennett et al., 2006, p138) Kuesioner adalah suatu cara / metode yang digunakan untuk suatu penelitian. Kuesioner terdiri dari beberapa pertanyaan. Pertanyaan tersebut ditulis dengan bahasa yang mudah dimengerti oleh responden dan dilengkapi dengan pilihan-pilihan jawaban yang memudahkan responden dalam memilih jawaban.

Kuesioner paling ideal digunakan untuk jumlah responden yang banyak dan tersebar di berbagai wilayah yang berbeda.

(Bennett et al., 2006, p138) Teknik kuesioner memiliki beberapa kelebihan antara lain :

- Cara paling ekonomis untuk memperoleh data dari jumlah responden yang banyak.
- Memperoleh hasil data yang mudah dianalisa oleh komputer, jika pertanyaan didesain dengan baik.

Teknik kuesioner juga memiliki beberapa kekurangan antara lain

- Sulit untuk membuat pertanyaan kuesioner yang baik.
- Tidak ada mekanisme yang lebih dalam untuk menindak lanjuti hasil jawaban dari responden. Jika menggunakan metode wawancara, dapat ditindak lanjuti lebih dalam lagi.

2.6. *Unified Modeling Language* (UML)

Menurut Whitten et al. (2004, p408), UML adalah satu kumpulan konvensi pemodelan yang digunakan untuk menentukan atau menggambarkan sebuah sistem piranti lunak yang terkait dengan objek. Whitten et al. (2004, p417) UML menawarkan diagram-diagram untuk memodelkan suatu sistem, seperti satu set *blueprint*. Sedangkan menurut Mathiassen et al. (2000, p330), UML adalah sekumpulan diagram yang merepresentasikan sistem dengan bahasa tekstual dan grafikal melalui simbol-simbol tertentu. Jadi UML adalah bahasa permodelan visual yang standar untuk memgambarkan sebuah sistem.

2.6.1. *Rich Picture*

(Matthiassen et al., 2000, p335) *Rich picture* digunakan untuk menggambarkan alur proses dalam perkembangan sistem secara keseluruhan. *Rich picture* terdiri dari orang, alur proses dan sistem permasalahannya. Pada dasarnya, *rich picture* tidak memiliki notasi khusus untuk mendeskripsikannya, tetapi perlu adanya kesepakatan khusus untuk menetapkan simbol atau notasi tertentu sesuai dengan apa yang dideskripsikan.

2.6.2. *Use Case Diagram*

(Whitten et al., 2004, p417) *Use case diagram* adalah proses pemodelan fungsi-fungsi sistem dalam konteks peristiwa-peristiwa bisnis, siapa yang mengawalinya dan bagaimana sistem itu merespons hal tersebut.

(Mathiassen et al., 2000, p119) *Use case diagram* digunakan untuk memaparkan interaksi hubungan antara pengguna sistem (*actor*) dengan *use case*. *Actor* didefinisikan sebagai suatu abstraksi dari pengguna atau sistem lain yang berinteraksi dengan target sistem. *Use case* adalah pola interaksi antara sistemdengan *actor* dalam *application domain*.

(Schneider et al., 2001, p27-47) Bagian-bagian yang penting dalam mendokumentasikan *use case diagram* antara lain :

1. *Actor*

   Actor adalah segala sesuatu yang berinteraksi dengan sistem dan memiliki peranan tertentu.

2. *Precondition*

   Precondition menunjukan apa yang terjadi sebelum *use case* dimulai dan menyatakan kondisi sistem pada saat *use case* dimulai.

3. *Post condition*

   Post condition menunjukan apa yang terjadi sesudah *use case* dan menyatakan kondisi sistem pada saat *use case* berakhir.

4. *Flow of events*

   Flow of events adalah serangkaian pernyataan deklaratif yang menyatakan langkah-langkah *use case* dari sudut pandang *actor*.

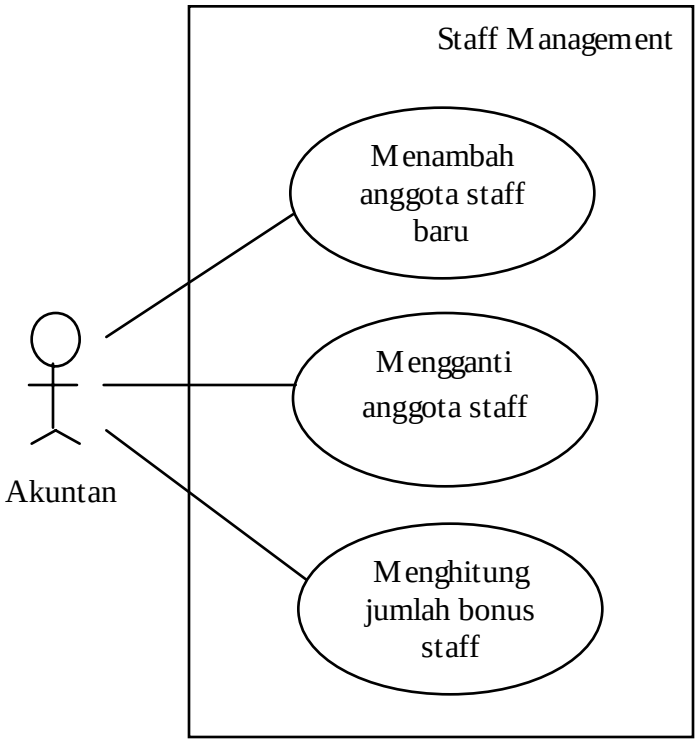
5. *Basic path*

   Basic path terjadi jika suatu kondisi di mana semua berjalan benar

6. *Alternative path*

   Alternative path menunjukan adanya urutan kejadian yang berbeda dengan urutan kejadian pada *basic path* (adanya kesalahan).

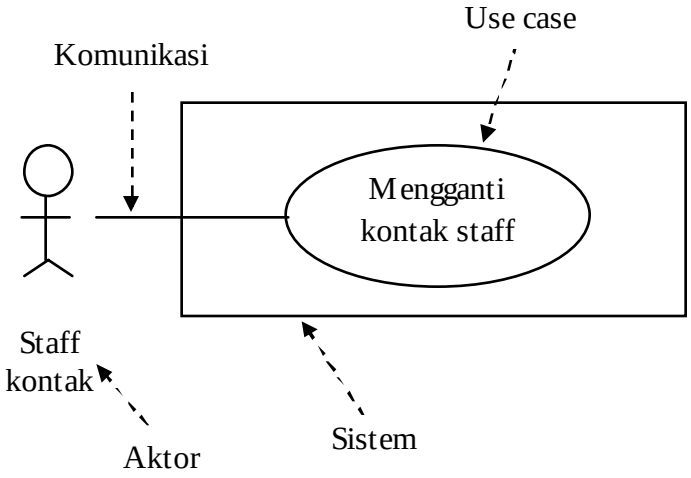
(Bennett et al., 2006, p145) *Use case diagram* adalah deskripsi dari fungsionalitas antara sistem dengan pengguna sistem. *Use case diagram* digunakan untuk menunjukkan dan mendeskripsikan pengguna berkomunikasi dengan sistem sesuai dengan fungsionalitasnya.



Gambar 2.7 Contoh *Use Case Diagram*

Sumber : Bennett et al. (2006, p145)

(Bennett et al., 2006, p146) *Use case diagram* dikembangkan oleh Jacobson pada tahun 1992 (*Use Case Driven Approach*). *Use case diagram* terdiri dari 3 bagian utama yaitu *actor*, *use case* (fungsi), dan sistem yang terkait.

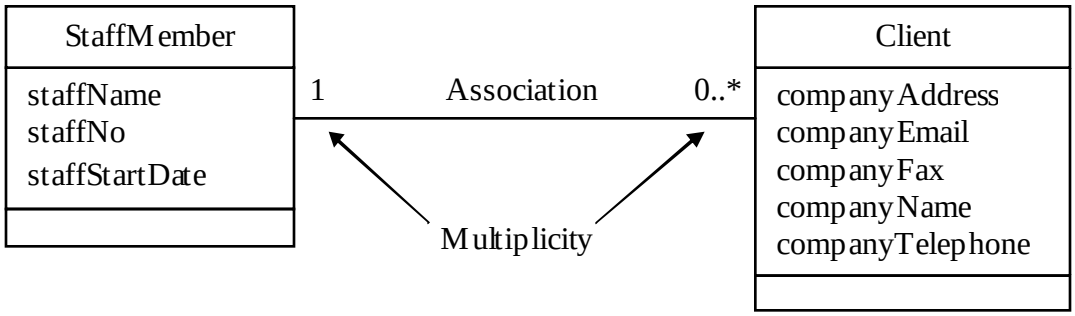


Gambar 2.8 Notasi *Use Case Diagram*

Sumber : Bennett et al. (2006, p146)

2.6.3. *Class Diagram*

(Mathiassen et al., 2000, p336) *Class diagram* menggambarkan kumpulan kelas dan hubungan strukturalnya. *Class* didefinisikan sebagai suatu deskripsi dari kumpulan objek yang memiliki struktur, perilaku, pola perilaku dan atribut yang serupa.



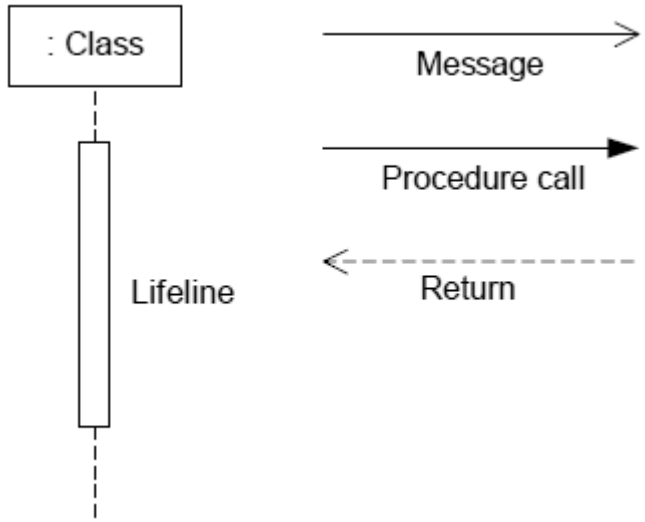
Gambar 2.9 Notasi *Class Diagram*

Sumber : Bennett et al. (2006, p185)

2.6.4. *Sequence Diagram*

(Mathiassen et al., 2000, p340-341) *Sequence diagram* menggambarkan interaksi di antara beberapa objek dari waktu ke waktu. Hal ini berbeda dengan *class diagram* yang bersifat statis. *Sequence diagram* menggambarkan situasi yang lebih kompleks dan dinamis.

(Bennett et al., 2006, p253) *Sequence diagram* digunakan untuk menunjukkan interaksi antar objek yang tersusun dalam urutan waktu. *Sequence diagram* dapat digambarkan dengan beberapa tingkatan berbeda dengan tujuan berbeda pada setiap siklus sistem. Secara umum, *sequence diagram* mempresentasikan interakasi setiap objek dengan detail yang memicu jalannya operasi lainnya.



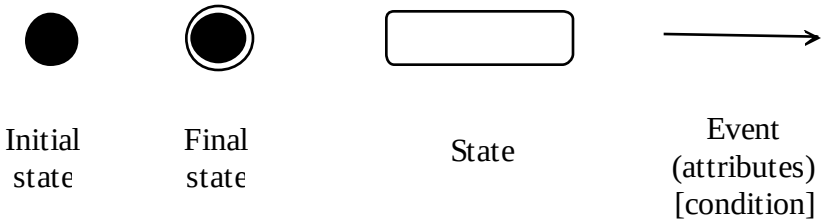
Gambar 2.10 Notasi *Sequence Diagram*

Sumber : Mathiassen et al. (2000,p340)

2.6.5. *Statechart Diagram*

(Whitten et al., 2004, p419) *Statechart diagram* digunakan untuk memodelkan *behavior* objek khusus yang dinamis. Diagram ini mengilustrasikan siklus hidup objek, dengan berbagai keadaan yang dapat diasumsikan oleh objek dan *event* yang menyebabkan objek beralih dari satu state ke state lain.

(Mathiassen et al., 2000, p341) *Statechart diagram* menggambarkan perilaku umum dari semua objek di *class* khusus, berisi *state* dan transisi antar objek.



Gambar 2.11 Notasi *Statechart Diagram*

Sumber : Mathiassen et al. (2000,p341)

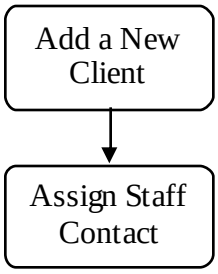
2.6.6. *Activity Diagram*

(Bennett et al., 2006, p113) *Activity diagram* sangat bermanfaat untuk merepresentasikan model kegiatan bisnis. Secara singkat, *activitiy diagram* digunakan dengan tujuan :

- Untuk menggambarkan model atau alur dari suatu proses.
- Untuk mendeskripsikan suatu fungsi sistem.

- Untuk mendeskripsikan logika dari sebuah operasi.

(Bennett et al., 2006, p114) Notasi *activity diagram* digambarkan dengan sederhana. Satu *activity* dihubungkan dengan *activity* lainnya dengan garis panah lurus (*activity edges*). Setiap proses digambarkan dengan segi empat oval dan nama dari proses ditulis di dalamnya.



Gambar 2.12

Contoh dua *activites* dihubungkan dengan *control flow*

Sumber : Bennett et al. (2006, p114)

2.7. Perancangan Layar (*User Inter*face)

Menurut Mathiassen et al (2000, p151), *interface* adalah fasilitas yang membuat model dari suatu sistem dan fungsi-fungsinya tersedia untuk pengguna.

Menurut Shneiderman (1998, p74), ada delapan aturan emas (*Eight Golden Rules*) yang perlu diperhatikan dalam merancang *user interface,* yaitu :

1. Usahakan konsistensi. Dalam merancang tampilan usahakan untuk selalu konsisten, baik dalam penggunaan *terminology*, menu, warna, huruf, *shortcut*, dan sebagainya.
2. Menyediakan *shortcut* untuk *frequent user*. Aksi-aksi yang sering digunakan sebaiknya dibuatkan *shortcut* agar mempermudah dan mempercepat interaksi.
3. Memberikan umpan balik yang informatif. Untuk setiap aksi dari pengguna sebaiknya ada umpan balik dari sistem. Untuk aksi-aksi yang sering terjadi dan berdampak kecil sebaiknya gunakan respon yang sederhana. Untuk aksi-aksi yang jarang terjadi dan berdampak besar sebaiknya digunakan respon yang lebih tegas.
4. Merancang dialog yang memberikan penutupan. Umpan balik berupa dialog konfirmasi yang dimunculkan sistem atas aksi yang dilakukan pengguna sebelum aksi tersebut dieksekusi. Misalnya pada saat menutup suatu program maka akan muncul dialog konfirmasi penutupan.
5. Menyediakan pencegahan kesalahan fatal dan penanganan kesalahan sederhana. Sebisa mungkin rancang sistem agar pengguna tidak dapat membuat kesalahan fatal. Sebagai contoh, gunakan pemilihan menu dibanding dengan pengisian *form* dan tidak menginjinkan penggunaan karakter alfabet dalam pengisian *numeric form fields*. Jika pengguna melakukan kesalahan, sistem harus dapat mendeteksi kesalahan tersebut dan menawarkan instruksi yang sederhana, membangun, dan spesifik untuk melakukan *recovery*. Sebagai contoh, pengguna tidak harus mengetik ulang

seluruh *command* yang salah, namun hanya memperbaiki bagian yang salah dari *command* tersebut.

6. Memungkinkan pembalikan aksi dengan mudah. Sebisa mungkin, setiap aksi sebaiknya bersifat *reversible*. Fasilitas ini dapat membuat pengguna lebih berani untuk mengeksplorasi pilihan-pilihan yang tidak dikenal.
7. Mendukung pusat kendali internal. Pengguna yang berpengalaman menginginkan pengontrolan atas sistem dan dapat merespon setiap tindakannya.
8. Mengurangi beban ingatan jangka pendek. Keterbatasan kemampuan mengolah informasi pada ingatan jangka pendek manusia mengharuskan agar tampilan dibuat sesederhana mungkin.

2.8. Rich Internet Application (RIA)

2.8.1. Definisi RIA

(Fain et al., 2004, p2) *Rich Internet Application* (RIA) adalah aplikasi berbasis *web* yang memiliki hampir semua kemampuan dari aplikasi *desktop*. RIA menggabungkan keuntungan menggunakan *web* sebagai model *deployment* yang efisien dengan kemampuan interaksi dengan pengguna sebaik menggunakan aplikasi *desktop*.

2.8.2. Keuntungan RIA

[http1] *Rich Internet Application* (RIA) menawarkan fitur-fitur yang sudah terbukti, biaya dan cara yang efektif untuk menyampaikan aplikasi modern dengan manfaat bisnis yang nyata :

- Menawarkan fitur-fitur yang lebih kaya dan meningkatkan kepuasan pengguna.
- Memenuhi sesuai harapan pengguna.
- Meningkatkan loyalitas pelanggan dan menghasilkan keuntungan yang lebih tinggi.
- Meningkatkan personil, proses, dan infrastruktur yang ada.

2.9. Adobe Flex

2.9.1. Definisi Adobe Flex

[http2] Flex adalah *free open source framework* untuk membangun dan memelihara aplikasi *web* yang ekspresif, menyebarkan secara konsisten di semua *web browser*, *desktop*, dan sistem operasi. Aplikasi Flex hanya dapat dibangun dengan menggunakan *free open source framework*, pengembang dapat menggunakan Adobe ® Flex ® Builder ™ software untuk mempercepat pembangunan.

[http3] Flex berbasis bahasa pemrograman standar dan model *programming* yang mendukung pola desain umum. MXML, bahasa

deklaratif yang berbasis XML, yang digunakan untuk menggambarkan tata letak dan perilaku UI (*User Interface*), dan ActionScript ™ 3, bahasa pemrograman yang berorientasi objek, digunakan untuk membuat *client logic*. Flex juga mencakup *rich component library* dengan lebih dari 100 yang telah terbukti.

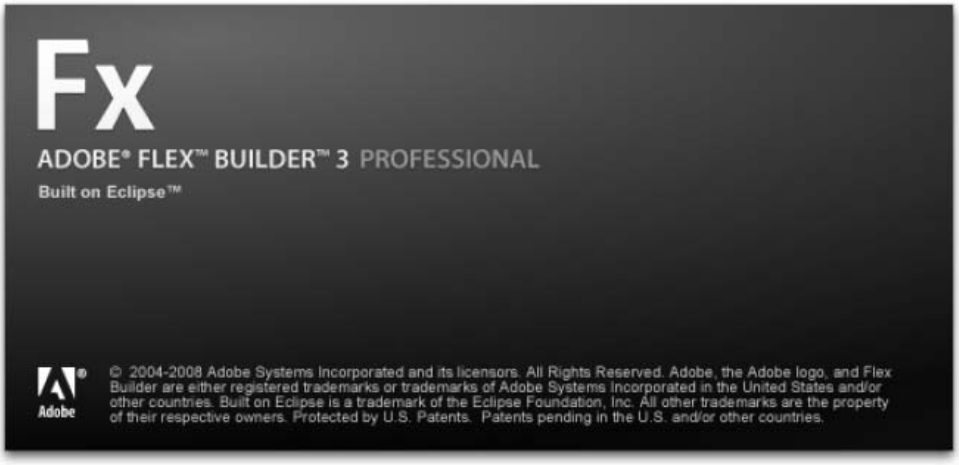
[http3] RIA dibuat dengan Flex dapat berjalan di *browser* dengan menggunakan Adobe Flash ® Player atau pada *desktop* dengan Adobe AIR ™. Hal ini memungkinkan untuk menjalankan aplikasi Flex secara konsisten di semua *browser* dan pada *desktop* dengan menggunakan AIR. Aplikasi Flex dapat mengakses *access local* dan *system resources* pada *desktop*. Flash Player dan Adobe AIR dapat diunduh secara gratis di www.adobe.com.

2.9.2. Adobe Flex Builder 3

[http3] Adobe Flex Builder 3 tersedia dalam Standard dan Professional Editions. Kedua edisi tersebut dapat memungkinkan *import assets* dari Adobe Creative Suite ® 3 software sehingga memudahkan bagi *designers* dan *developers* bekerja secara bersamaan. Flex Builder 3 Profesional memiliki kemampuan visualisasi data yang kuat, Advanced Datagrid yang baru, dan mendukung untuk *functional testing* secara otomatis untuk mengembangkan aplikasi bisnis yang penting.

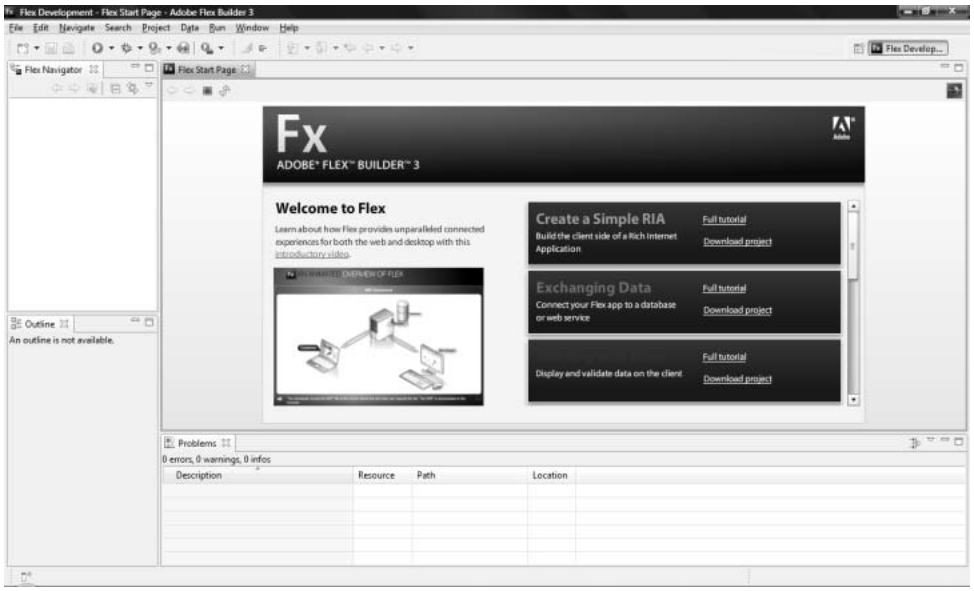
2.9.3. Adobe Flex 3.4 *Software Development Kit* (SDK)

[http3] Flex *framework* dapat diunduh secara terpisah dari Flex Builder 3 dengan mengunduh Free Adobe Flex 3.4 SDK. *Developers* dapat membuat RIA secara lengkap hanya dengan menggunakan Free Adobe Flex 3,4 SDK.



Gambar 2.13 *Startup Splash* Adobe Flex Builder 3 Professional

Sumber : Herrington et al. (2008, p2)



Gambar 2.14 *Start Page* Adobe Flex Builder 3 Professional

Sumber : Herrington et al. (2008, p3)

2.9.4. Flex *Framework*

[http4] *Framework* Flex menyediakan bahasa deklaratif, *application services*, *component*, dan *data connectivity* sehingga pengembang untuk secara cepat membangun *rich internet application* (RIA) untuk *browser* atau *desktop*.

2.9.4.1. *Languages*

MXML adalah bahasa yang digunakan untuk menentukan tata letak, penampilan, dan *behaviors* dari aplikasi Flex. ActionScript ™ 3 adalah bahasa berorientasi objek yang

mendefinisikan logika aplikasi di sisi klien. MXML dan ActionScript yang disusun bersama-sama ke dalam satu berkas SWF yang membuat aplikasi Flex tersebut.

2.9.4.2. *Component*

Component Library menyediakan semua kontrol antarmuka pengguna antara lain tombol sederhana, *check box*, dan *radion button*, *data grid*, *combo box*, dan *rich text editor*.

2.9.5. *System Requirement* [http5]

- **Flex Builder 3 for Windows**
  - Intel® Pentium® 4 processor
  - Microsoft® Windows® XP Service Pack 2 atau Windows Vista® Home (Premium or Basic), Business, atau Ultimate
  - 1GB of RAM (disarankan 2GB)
  - 500MB kapasitas *hard-disk* yang tersedia (tambahan 500MB dibutuhkan untuk pengaturan *plug-in*)
  - Java™ Virtual Machine: Sun™ JRE 1.4.2, Sun JRE 1.5 (included), IBM® JRE 1.5, atau Sun JRE 1.6
  - Adobe® Flash® Player 9
- **Flex 3.4 SDK**

- Windows 2000, Windows XP, atau Windows *Server*® 2003, Java 1.4 (Sun, IBM, or BEA) atau 1.5 (Sun)
- Mac OS X v10.4.x, Java 1.5 (seperti yang disertakan dari Apple) pada PowerPC atau Intel *processor*
- Red Hat® Enterprise Linux® 3 atau 4, SUSE® 10, Java 1.4 (Sun, IBM, atau BEA) atau 1.5 (Sun)
- Solaris™ 9, 10, Java 1.4 atau 1.5 (Sun) Compilers
- 512MB RAM (disarankan 1GB)
- 200MB kapasitas *hard-disk* yang tersedia

2.10. *eXtensible Markup Language* (XML)

2.10.1. Definisi XML

Menurut Allamaraju et al (2000 , p218), sulit untuk mendefinisikan XML karena XML bukanlah sebuah teknologi tunggal namun adalah kelompok dari teknologi-teknologi yang berhubungan yang secara terus - menerus menambah anggota baru. Yang bisa didefinisikan adalah XML *Documents*, yaitu sebuah unit informasi yang bisa dipandang dengan 2 cara, sebagai urutan karakter linear yang mengandung data karakter dan *markup*, atau sebagai sebuah struktur data abstrak yang terdiri dari sebuah *nodes tree*.

Menurut O'Brien (2005, p126), XML menjelaskan isi dari halaman *web* (termasuk dokumen-dokumen bisnis yang didesain untuk

digunakan dalam *web*) dengan menerapkan pengenalan *tags* atau *contextual labels* pada data di dokumen-dokumen *web*.

Menurut Marchal (2000 , p6), XML dikembangkan oleh World Wide *Web* Consortium (W3C) untuk mengatasi kelemahan dari *Hypertext Markup Language* (HTML), salah satunya adalah banyaknya jumlah *tag* HTML yang seringkali menyebabkan *human error* saat pembuatan dokumennya. XML tidak memiliki *predefined tag*. Karena itulah dinamakan *extensible* karena XML tidak memiliki *predefined tags* melainkan penulis / *developer* akan menciptakan *tags* mereka sendiri yang dibutuhkan untuk aplikasi mereka.

Menurut Allamaraju et al (2000 , p27), XML mempengaruhi bagaimana cara kita memandang, memproses, memindahkan dan mengatur data.

2.10.2. Kegunaan XML

Menurut Allamaraju et al (2000 , p222), kegunaan utama XML adalah :

- Menciptakan dokumen untuk diterbitkan / dipublish pada *web* atau tempat lain.
- Pengkodean data, termasuk data yang diambil dari *relational databases*.
- Mengkonfigurasi program – program komputer.

- Mengembangkan kata-kata berdomain spesifik dan DTD (*Document Type Definition*). DTD adalah *grammar* atau kata yang ditentukan atau diciptakan sendiri oleh *programmer* untuk isi dari XML.

2.10.3. Kelebihan XML

Menurut Simon (2001 , p2), kelebihan XML adalah bahwa XML mengizinkan *developer* untuk mengembangkan *markup tag* yang digunakan dalam dokumen atau aplikasi. *Developer* dapat memodifikasi *tags* yang sudah ada, mendefinisikan arti baru bahkan menciptakan *tag* yang benar - benar baru.

XML adalah dasar dari B2B *e-commerce* dan tulang punggung dari *Wireless Application Protocol* (WAP). XML akan menyederhanakan *Electronic Data Ingterchange* (EDI). XML akan terus menjadi pendukung di balik cara - cara baru untuk berbagi informasi, memindahkan data, dan pengiriman komunikasi ke banyak jenis *device*.

2.10.4. *Document Type Definition* (DTD)

Menurut Simon (2001 , p6), DTD adalah inti dari XML. DTD mendefinisikan bagaimana *browser* harus menangani text atau elemen lain dari dokumen XML. DTD didefinisikan melalui tag dan action. User memasukkan tag-tag ke dalam dokumen sebagai tempat bagi action yang diinginkan berlangsung. Contohnya user ingin membuat tag yang

menampung salary karyawan, maka user dapat memasukkan tag <salary> yang berisi data salary.

2.11. *Server Side Scripting*

2.11.1. Java *Server Pages* (JSP)

2.11.1.1. JSP *Overview*

Menurut Allamaraju et al (2000 , p475), Java *Server Pages* (JSP) bukanlah sebuah produk melainkan seperti aplikasi Java API lainnya, JSP adalah sebuah spesifikasi yang disediakan oleh Sun Microsystems kepada *vendor* untuk mengimplementasikannya. Spesifikasi JSP dibangun di atas fungsionalitas yang dimiliki aplikasi *servlet*. JSP lebih menyerupai model J2EE dengan memisahkan konten statik dan dinamik – memisahkan presentasi dan logika – pada sebuah aplikasi *web*.

Contoh penggunaan JSP :

```
<%@page import=”java.util.Date”%>
<html>
<body>
The Current Time is <%= new Date().toString() %>
</body>
</html>
```

Penggalan program di atas akan menampilkan waktu komputer saat ini. Untuk menampilkan hasilnya, file JSP harus dimasukkan dalam

web server yang digunakan (misalkan Apache Tomcat) dan dibuka melalui *browser*.

2.11.1.2. Aturan Dasar JSP

Tags pada JSP terbagi menjadi 3 kategori :

- *Directives* : Mempengaruhi struktur keseluruhan *servlet* yang dihasilkan dari translasi.
- *Scripting Elements* : Mengijinkan *programmer* memasukkan kode Java dalam halaman JSP.
- *Actions* : *Tags* spesial yang dapat mempengaruhi *runtime* dari JSP.

Tags JSP bersifat *case sensitive* yaitu penggunaan huruf kapital dan huruf biasa akan berpengaruh secara detail.

2.11.1.3. JSP *Directives*

Menurut Allamaraju et al (2000 , p481), JSP *directives* menjadi pesan yang dikirimkan ke *container* JSP yang digunakan untuk menetapkan nilai global seperti deklarasi *class, method* yang akan diimplementasikan, tipe konten *output*, dan lain – lain, tetapi tidak mengembalikan *output* apapun kepada *user*. *Directives* dimulai dengan

tanda ‘<%@’ dan diakhiri dengan tanda ‘%>’. Secara umum, perintah yang digunakan adalah :

<%@ directivename attribute=”value” attribute=”value” %>.

Ada 3 *directives* utama yang bisa digunakan pada JSP :

- *Page Directives*

  Allamaraju et al (2000 , p481) menjelaskan bahwa *page directives* digunakan untuk mendefinisikan dan memanipulasi atribut –atribut penting yang akan mempengaruhi seluruh JSP, dan mengkomunikasikan atribut – atribut tersebut ke *container* JSP. Satu halaman JSP bisa memiliki lebih dari satu page *directives*, urutannya tidak berpengaruh, namun hanya boleh ada satu pengesetan atribut. Mereka akan diambil saat *translation* dan akan dijalankan bersama-sama dengan halaman tersebut. Secara umum, perintah yang digunakan adalah :

  <%@ page ATTRIBUTES %>

- *Include Directives*

  (Allamaraju et al, 2000 , p484) *Include directives* menginstruksikan kepada JSP *container* untuk memasukkan konten dari sumber halaman JSP, dengan memasukkan file tersebut pada *directives*. Tentu saja file tersebut harus bisa diakses oleh JSP *container*. Aksi *include* digunakan untuk memasukkan sumber saat *runtime*. Kebanyakan JSP *container* biasanya menyimpan file yang sudah disertakan dan akan meng-*compile* ulang saat ada perubahan. Perintah dari *include directive* adalah :

<%@ include file=”filename” %>

- *Taglib Directives*

  Menurut Allamaraju et al (2000 , p486), *taglib directives* memungkinkan page untuk menggunakan *tags* tambahan. *Taglib directives* menamakan *tag library* yang menyimpan kode Java yang telah di-*compile* yang mendefinisikan *tags* tambahan tersebut. Perintah dari *taglib directives* :

  <%@ taglib url=”taglibraryURL” prefix=”tagPrefix” %>

2.11.1.4. *Scripting Elements*

Menurut Allamaraju et al (2000 , p486), JSP s*cripting* se*ements* mengizinkan kode Java – deklarasi variabel dan *method, scriplets* dan *expression* – untuk dimasukkan ke dalam halaman JSP.

- *Declaration*

  (Allamaraju et al, 2000 , p487) *Declaration* adalah blok kode Java dalam JSP yang digunakan untuk mendefinisikan variabel dan *method*. *Declaration* akan diinisialisasikan saat halaman JSP diinisialisasikan. Blok *declaration* dibatasi dengan ‘<%!’ dan ‘%>’ Perintah yang digunakan secara umum :

  <%! Deklarasi Java Variable dan method %>

- *Scriptlets*

Allamaraju et al (2000 , p481) mengatakan bahwa *scriptlet* adalah blok kode Java yang dieksekusi saat *request* diproses dan ditutup antara tag '<%' dan '%>'. *Scriptlets* bisa menghasilkan *output* untuk *client*. Perintah dari *scriptlets* adalah :

<% Kode Java yang valid %>

- *Expressions*

  Menurut Allamaraju et al (2000 , p481), *expressions* adalah notasi singkat dari *scriptlet* yang mengirimkan nilai dari ekspresi Java ke Client. *Expression* dieksekusi saat proses *request* HTTP dan hasilnya akan diubah ke dalam bentuk *string* dan akan ditampilkan. *Expression* ditutup dalam tag '<%=' dan '%>'. Perintahnya adalah :

  <%= Java Expression %>

2.11.2. *Web Server*

Menurut Laurie et al (1999, p1-3), *web server* adalah program komputer yang berfungsi untuk mengartikan sebuah URL menjadi sebuah nama file, kemudian mengirim kembali file tersebut melalui *internet*, atau ke dalam sebuah program, dan kemudian menjalankan program tersebut dan mengirim hasilnya kembali.

Ketika *client* berusaha mengunjungi *home page* melalui URL, contohnya http://www.binusmaya.binus.ac.id/, maka *client* mengirim pesan melalui *internet* kepada mesin pada alamat tersebut. Mesin tersebut, diharapkan, dalam keadaan menyala dan sedang berjalan,

koneksi *internet* tersambung dengan baik, dan siap menerima dan merespon pesan *client.*

URL pada contoh di atas terdiri dari tiga bagian : *<method>://<host>/<absolute path (apURL)>*. Maka pada contoh di atas, <method> adalah http, artinya bahwa *browser* harus menggunakan HTTP (Hypertext Transfer Protocol); *<host>* adalah www.binusmaya.binus.ac.id; dan *<apURL>* adalah "/", artinya adalah direktori paling atas dari *host*. Menggunakan HTTP/1.1, *browser* mengirim pesan sebagai berikut :

GET / HTTP/1.1

Host : www.binusmaya.binus.ac.id

Pesan tersebut tiba pada *port* 80 (*port* default untuk HTTP) pada *host* www.binusmaya.binus.ac.id. Pesan ini terdiri dari tiga bagian : sebuah *method* (HTTP *method*, bukan URL *method*), pada hal ini adalah GET, namun dapat juga berupa PUT, POST, DELETE, atau CONNECT; Uniform Resource Identifier (URI) "/"; dan versi *protocol* yang kita gunakan. Kemudian pesan ini ditindaklanjuti oleh *web server*.

Web server diharapkan memiliki fitur-fitur sebagai berikut :

1. Cepat, sehingga dapat melayani banyak *inquiries* dengan menggunakan perangkat keras minimum.
2. *Multitasking*, sehingga dapat melayani lebih dari satu *inquiry* pada saat yang sama.

3. *Multitasking*, sehingga orang yang menjalankan dapat melakukan *maintenance* data tanpa harus mematikan *service*.
4. Mengenali *inquirers* : beberapa mungkin memiliki hak akses lebih terhadap beberapa *service* dibanding yang lain.
5. *Error messages* yang memberikan informasi yang jelas tentang kesalahan yang terjadi.
6. Menyesuaikan dengan bahasa *inquirer*. *Inquirer* dapat memilih sendiri bahasa yang ingin digunakan.
7. Menyediakan format-format yang berbeda. Pengguna mungkin menginginkan format gambar JPEG dibandingkan dengna GIF.
8. Berperan sebagai *proxy server*. *Proxy server* menerima permintaan dari *clients*, meneruskannya kepada *server* yang sebenarnya, dan kemudian mengirim respon dari *server* yang sebenarnya kepada *clients*.
9. Aman. Tujuan dari *server* yang baik adalah mencegah terjadinya akses dari pihak-pihak yang tidak diinginkan.

2.11.3. Apache Tomcat

[http6] Apache Tomcat adalah *open source web server* yang merupakan implementasi dari teknologi Java Servlet dan Java *Server* Pages. Apache Tomcat merupakan produk dari The Apache Software

Foundation. Versi yang terbaru pada saat penulisan ini adalah Apache Tomcat 6.0.20.

[http7] Direktori utama instalasi Apache Tomcat disebut $CATALINA_HOME. Di dalam $CATALINA_HOME terdapat beberapa *folder* utama, yaitu :

- /bin – *Startup*, *shutdown*, dan *scripts* lainnya.
- /conf – File-file konfigurasi dan yang berhubungan dengan DTD. File terpenting adalah *server*.xml sebagai file konfigurasi utama.
- /logs – File-file log disimpan di sini secara default.
- /*web*apps – Folder untuk menyimpan aplikasi berbasis *web*.

Untuk mengatur Apache Tomcat digunakan Tomcat Manager. Untuk mengakses Tomcat Manager, pastikan *service* dari Apache Tomcat dalam keadaan *up*, kemudian gunakan *browser* untuk mengakases default Tomcat *home page* pada URL http://localhost:8080/, 8080 adalah *default port* saat instalasi Tomcat. Kemudian masuk ke dalam Tomcat Manager dengan *username* dan *password* admin yang didefinisikan saat instalasi.

2.12. Oracle

2.12.1. Oracle Corporation

Menurut Nugroho (2008, p8-9), pada tahun 1977, Lawrence J. Ellison, Robert N. Miner, dan Edward Oates mendirikan System Development Laboratories untuk mengembangkan perangkat lunak *database* untuk komputer *mainframe*. Pada tahun 1979, System Development Laboratories mengembangkan sistem *database* relasional yang diberi nama Oracle RDBMS (Relational *Database* Management System), produk sistem *database* relasional pertama yang menggunakan SQL.

Pada tahun 1982, System Development Laboratories berubah nama menjadi Oracle Corporation. Oracle Corporation selanjutnya mulai bergerak mengembangkan perangkat keras dan perangkat lunak untuk menangani *database* video, suara, dan teks berukuran besar, untuk aplikasi-aplikasi yang berjalan lewat jaringan global (*internet*).

2.12.2. Oracle Database 10g

Menurut Lowenthal dan Dyke (2004, p18), Oracle Database adalah *database* pertama yang didesain untuk *enterprise grid computing*, cara yang paling fleksibel dan efisien untuk mengatur informasi dan aplikasi. Oracle *database* tersedia dalam tiga edisi, yaitu Enterprise, Standard, dan Personal.

(Dawes, 2005, p3) Oracle 10g dirilis pada tahun 2004 dengan versi 10.1.0.2. Produk *database* dari Oracle 10g menunjukan fitur-fitur baru. Ada 3 hal yang menonjol antara lain kemudahan dalam *management*, peningkatan dalam *scalability*, dan peningkatan dalaam kinerja *management.*

Kemudahan dalam *management* antara lain pengalokasian ke tempat penyimpanan *database* dengan *management* yang otomatis, peningkatan dalam sistem *monitoring*, tools *monitoring* berbasis *web* dan pengaturan keseluruhan arsitektur Oracle.

Peningkatan *scalability* dan kinerja di Oracle 10g berbasis model *grid computing*.

2.12.3. *System Requirements*

(Lowenthal dan Dyke, 2004, p27) Spesifikasi minimum perangkat keras yang dibutuhkan antara lain :

- 512 MB RAM
- 400 MB kapasitas *hard-disk* untuk direktori sementara
- 1.5 GB kapasitas *hard-disk* untuk Oracle *software*
- 1.5 GB kapasitas *hard-disk* untuk *preconfigured database*

2.12.4. Oracle *Database Architecture*

Lowenthal dan Dyke (2004, p53) Oracle Database terdiri dari beberapa file yang dikelompokan sebagai berikut:

- *Control files*

  Control files terdiri dari data tentang *database* itu sendiri, yang disebut dengan *metadata*. Semua file tersebut bagian yang *critical* dalam *database*. Jika file tersebut tidak ada, maka tidak dapat membuka *data files* untuk mengakses data dalam *database*.

- *Data files*

  Data files terdiri dari data-data yang ada di dalam *database*.

- *Online redo log files*

  Online redo log files dapat digunakan untuk *recovery database*. Jika terjadi *crash database*, maka data ataupun file dapat di *recovery* melalui *online redo log files*.

Adapun file-file lainnya yang bukan merupakan bagian utama dari *database*, tetapi memiliki peranan penting dalam menjalankan *database* antara lain:

- *Parameter file*

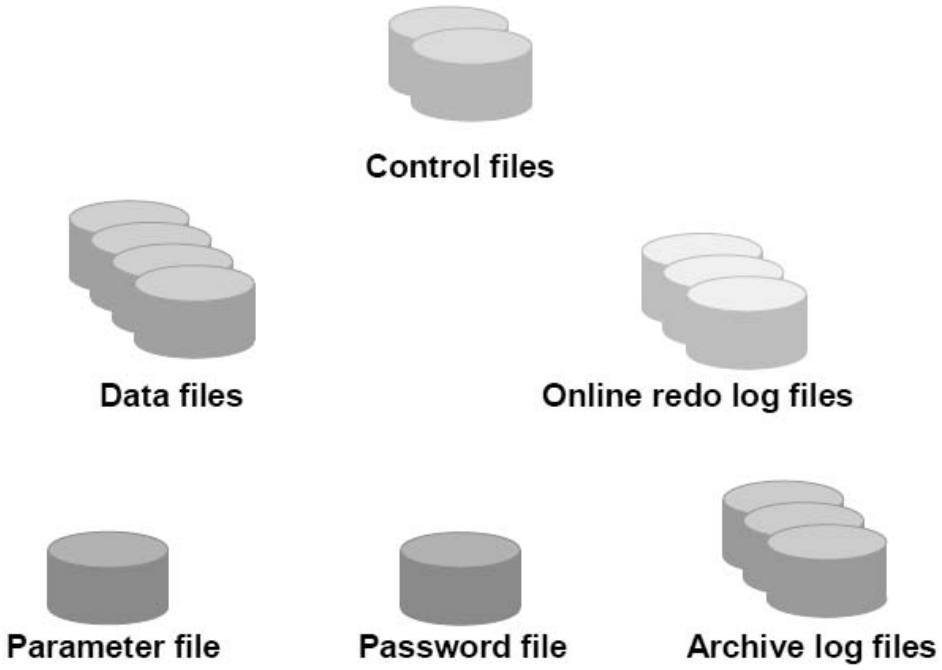
Parameter file digunakan untuk mendefinisikan konfigurasi-konfigurasi *database* pada saat dijalankan.

- *Password file*

  Password file memungkinkan pengguna untuk melakukan koneksi ke *database* dan melakukan *tasks* tertentu.

- *Archive log file*

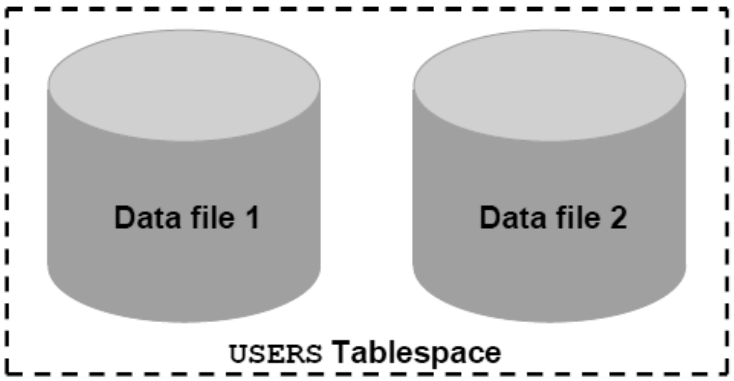
  Archive log file berisi *history redo* yang dapat digunakan untuk *recovery database*. File ini juga dapat digunakan sebagai *backup database* dan dapat mengembalikan *data file* yang hilang.



Gambar 2.15 Oracle Database *Architecture*

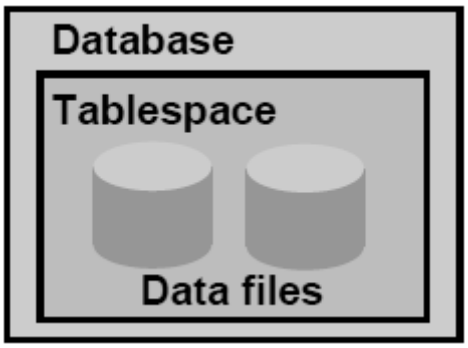
Sumber : Lowenthal dan Dyke (2004, p53)

Database secara *logical* dibagi menjadi unit-unit penyimpanan yang disebut dengan *tablespaces*, yang dapat digunakan bersama-sama berdasarkan *group related logical structures*. Setiap *database* secara *logical* dibagi ke dalam satu atau lebih *tablespaces*. Setiap *tablespace* memiliki satu atau lebih *data* file yang diciptakan secara eksplisit untuk menyimpan data dari semua struktur *logical* dalam sebuah *tablespace*.



Gambar 2.16 *Tablespace* dan *Data* file

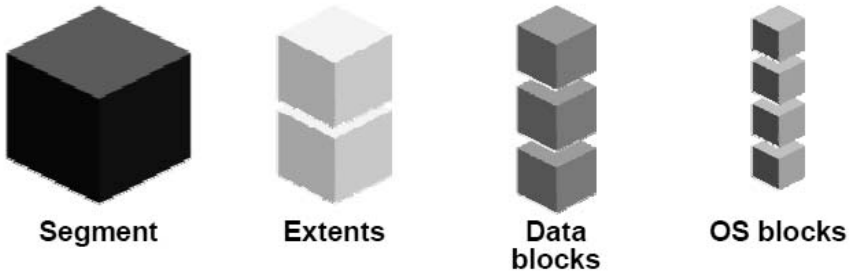
Sumber : Lowenthal dan Dyke (2004, p57)



Gambar 2.17 *Database, Tablespace* dan *Data* file

Sumber : Lowenthal dan Dyke (2004, p139)

Objek *database* seperti tabel dan indeks disimpan dalam *tablespace* sebagai *segment*. Setiap *segment* berisi satu atau lebih *extents*. Sebuah *extent* terdiri susunan *data blocks*, yang berarti bahwa setiap *extent* hanya ada dalam satu *data file*. *Data block* adalah unit terkecil dari I/O dalam *database*. Ketika *database* melakukan *request* satu set *block data* dari OS, maka OS *mapping* ke OS *blocks* pada perangkat penyimpanan.



Gambar 2.18 *Segments*, *Extends* dan *Data blocks*

Sumber : Lowenthal dan Dyke (2004, p58)

2.12.5. *Structure Query Language* (SQL)

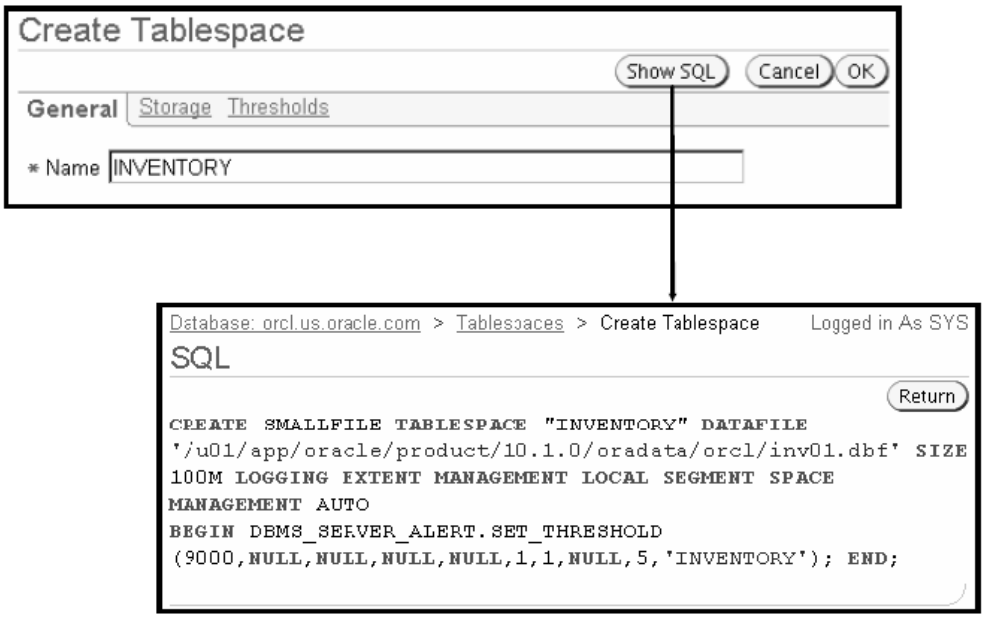
(Lowenthal dan Dyke, 2004, p83-84) *Structured query language* (SQL) adalah serangkaian *statement* yang digunakan mengakses data dalam *database* Oracle. *Statement* SQL dibagi menjadi beberapa kategori sebagai berikut:

- *Data Definition Language* (DDL), seperti *create, alter*.
- *Data Manipulation Language* (DML), seperti *insert, update, delete*.
- *Data Query Language* (DQL), seperti *select*.

- *Data Control Language* (DCL), seperti *grant, revoke.*

SQL dapat dijalankan dengan berbagai *tools* antara lain:

- Oracle SQL*Plus dan Oracle iSQL*Plus, *tools* yang menyediakan *interface* sederhana, dengan *command-line interface.*
- Oracle Enterprise Manager, *tool* berbasis grafis yang memanipulasi objek dan struktur *database.* Dengan menggunakan Enterprise Manager, *administrator* dapat berkonsentrasi untuk menyelesaikan tugas di tangan dari pada mengingat perintah SQL.



Gambar 2.19 Enterprise Manager, Show SQL

Sumber : Lowenthal dan Dyke (2004, p85)

2.12.6. SQL*Plus

SQL*Plus adalah *command-line interface* yang digunakan untuk menulis SQL*Plus, SQL dan PL/SQL.

- *Enter, edit, retrieve* dan *save* perintah SQL dan blok PL/SQL.
- Format, *calculate, store* dan mencetak hasil perintah *query.*
- Menjalankan tugas dari *database administrator*.

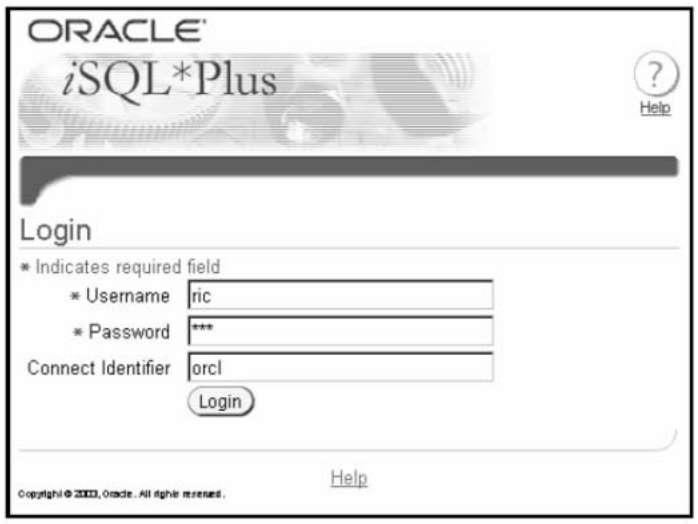
```
$ sqlplus /nolog
SQL*Plus: Release 10.1.0.2.0 - Production on Tue Feb
17 06:17:14 2004
Copyright (c) 1982, 2004, Oracle.  All rights
reserved.
SQL> connect ric
Enter password:
Connected.
SQL> SELECT * FROM dual;

D
-
X
SQL>
```

Gambar 2.20 SQL*Plus

Sumber : Lowenthal dan Dyke (2004, p86)

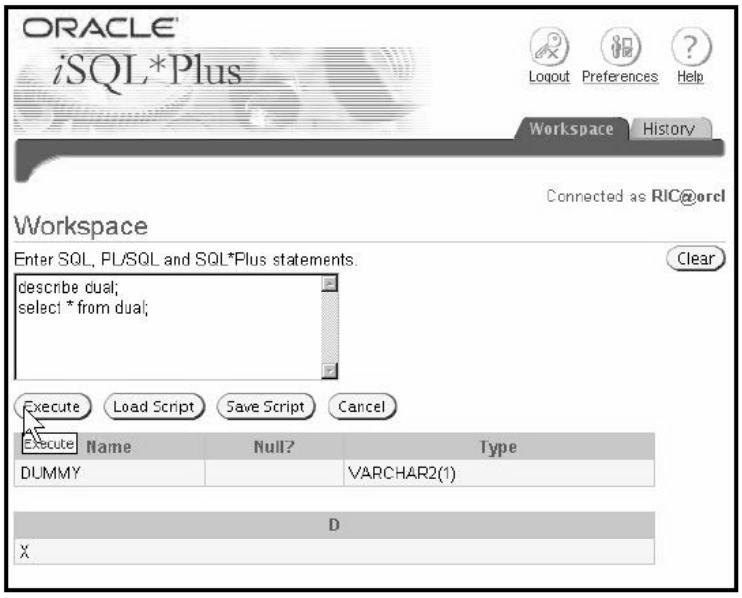
2.12.7. iSQL*Plus

Lowenthal dan Dyke (2004, p87) iSQL*Plus adalah *tool* berbasis *browse interface* dari Oracle *database.* iSQL*Plus merupakan bagian komponen dari produk SQL*Plus. iSQL * Plus memiliki proses *server-side listener* yang harus dimulai sebelum terhubung dengan *browser*.



Gambar 2.21 iSQL*Plus

Sumber : Lowenthal dan Dyke (2004, p87)



Gambar 2.22 Workspace iSQL*Plus

Sumber : Lowenthal dan Dyke (2004, p88)