BAB 2

TINJAUAN PUSTAKA

2.1. Profil SMA Negeri 2 Purwakarta

2.1.1. Sejarah SMA Negeri 2 Purwakarta

SMA Negeri 2 Purwakarta berdiri pada tanggal 3 September 1979 berdasarkan SK pendirian Menteri Pendidikan dan Kebudayaan RI dengan No Statistik Sekolah (NISS) : 3010220001002, dalam kurun waktu 29 tahun SMAN 2 Purwakarta telah dipimin oleh 11 orang Kepala Sekolah, yaitu

1. Bpk. M. Masri (Alm) (1979-1980)
2. Bpk DRS. Akhmad Subarkah (1980-1985)
3. Bpk Drs. H. Machmud Radi (1985-1988)
4. Bpk. H. Waryo Sukanda, BA (1988-1990)
5. Ibu Dra. Hj. R. Atty Ruchyati (1990-1993)
6. Bpk. Drs. Iri Setiadi, (1993-1996)
7. Bpk Drs. Jumdiat Marzuki, MM (1996-2000)
8. Ibu Dra. Iis Sri Sugiharti (2000-2005)
9. Bpk. Drs Asep Saepudin (2005-2008)
10. Plt. Kepala Bpk. Drs. Rudiana, MM (2008-2009)
11. Bpk. Drs. Marseno (2009-2011)
12. Bpk. Drs. H. Gaos Suherman, M.Pd (2011 - Sekarang)

SMAN 2 Purwakarta memiliki tujuan, yaitu dalam mewujudkan iklim belajar yang kondusif, selalu berusaha melengkapi sarana prasarana khususnya sarana prasarana pengembangan ruangan belajar,meningkatkan partisipasi orang tua untuk terlaksananya program sekolah, melaksanakan pembelajaran *team teaching* untuk meningkatkan mutu pembelajaran, membekali keterampilan komputer bagi siswa untuk menunjang mata pelajaran dan menghadapi perkemabangan IPTEK, dan mengembangkan pembelajaran yang berorientasi kepada KTSP (Kurikulum Tingkat Satuan Pendidikan).

2.1.2. Visi dan Misi SMA Negeri 2 Purwakarta

Visi

Membangun generasi kreatif, inovatif dan tangguh yang dilandasi oleh keimanan dan ketaqwaan.

Misi

1. Meningkatkan kepribadian dan kemadirian yang dilandasi iman dan taqwa.
2. Meningkatkan prestasi akademik dan non akademik agar tercapai mutu lulusan yang berkualitas.
3. Meningkatkan profesionalisme Guru dan TU, membina semangat kerjasama yang dilandasi kekeluargaan.
4. Menciptakan suasanan yang kondusif bagi terwujudnya harmonisasi civitas akademika.

5. Meningkatkan Wawasan Wiyata Mandala untuk mendukung kegiatan belajar mengajar.

2.1.3. Logo SMA Negeri 2 Purwakarta

Seperti hal nya badan instansi lainnya, SMA Negeri Purwakarta juga memiliki logo / lambang yang membedakan mereka dangan sekolah lainnya. Logo tersebut menjadi ciri khas sekolah tersebut.

Adapun logo SMA Negeri 2 Purwakarta adalah sebagi berikut :



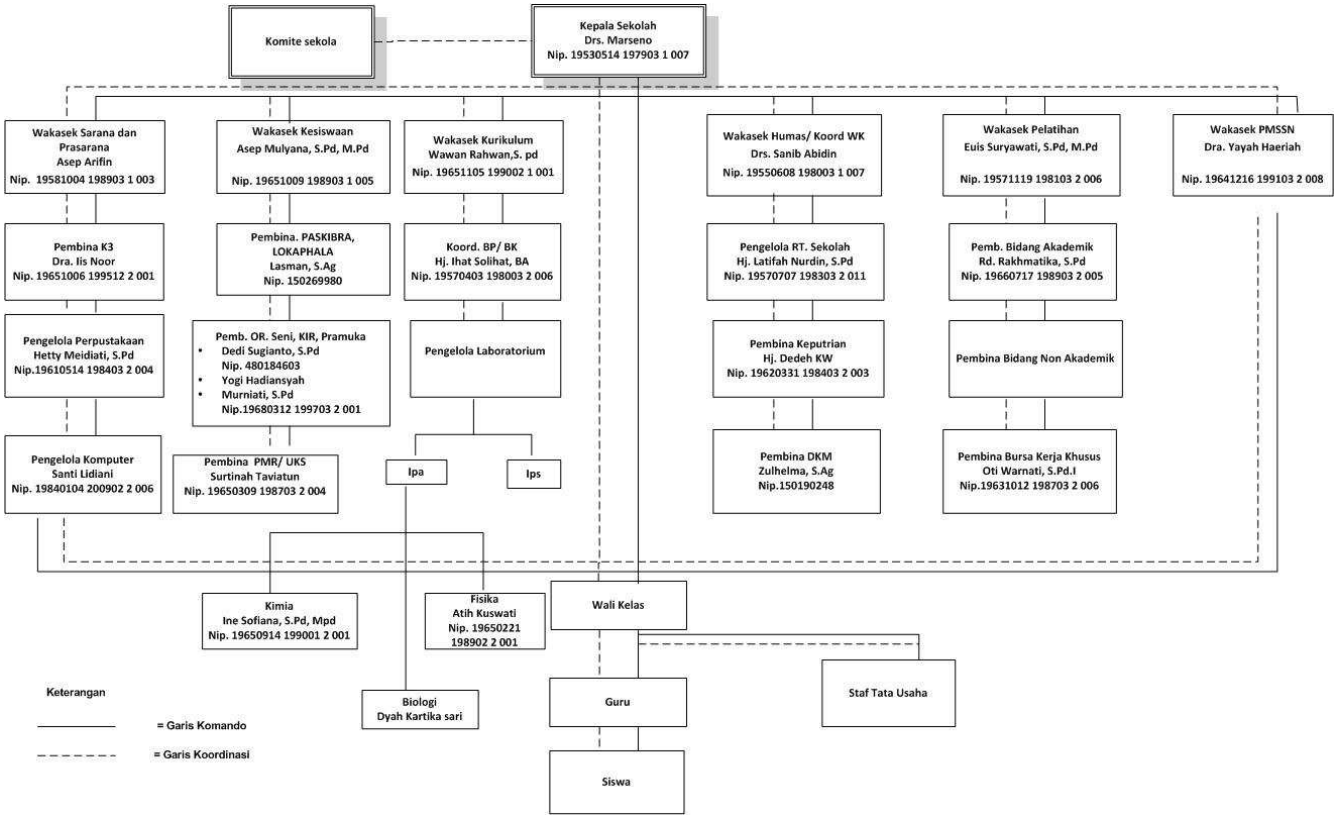
Gambar 2. 1 : Logo SMA Negeri 2 Purwakarta

2.1.4. Struktur Organisasi dan Job Description SMA Negeri 2 Purwakarta

Di setiap perusahaan atau instansi, pasti memiliki yang namanya struktur organisasi. Struktur organisasi berfungsi mengatur tugas dan kewenangan masing-masing anggota didalam organisasi tersebut dengan tujuan agar masing-masing individu memiliki deskripsi kerja (*job desc*) sehingga terjadi pembagian kerja yang jelas yang akan menyebabkan pencapaian tujuan secara efektif dan efisien. Begitu juga SMA Negeri 2 Purwakarta.

Mereka juga memiliki struktur organisasi yang berfungsi mengatur tugas dan kewenangan masing-masing pegawai.

Adapun struktur organisasi SMA Negeri 2 Purwakarta adalah sebagai berikut :



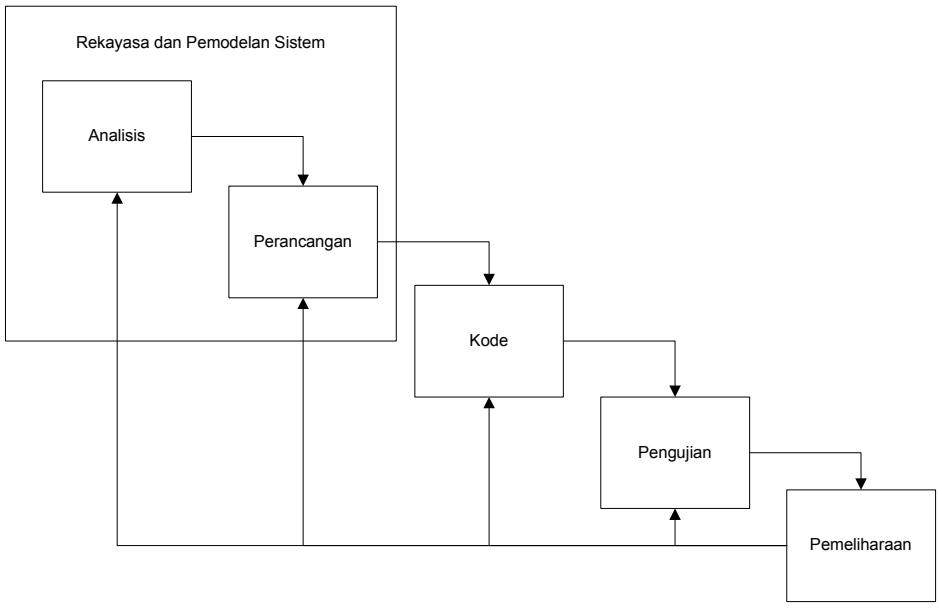
Gambar 2. 2 : Struktur Organisasi

2.2. Landasan Teori

2.2.1. Rekayasa perangkat lunak

Rekayasa perangkat lunak merupakan suatu teknologi pembangunan perangkat lunak yang mencakup proses, metode, dan *tools*. Rekayasa perangkat lunak merangkak kedekade keempat sejak keberadaannya, mendapatkan banyak kekuatan dan kelemahan yang dialami oleh manusia dari abad yang sama. Sifat inosen serta antusiasmenya yang ada di tahun-tahun awal kemunculannya, telah diganti dengan harapan yang lebih masuk akal yang mendukung tahun-tahun pengalamannya. Sekarang, rekayasa perangkat lunak dikenal sebagai disiplin yang sah, layak mendapatkan penelitian yang serius, studi yang sungguh-sungguh, dan diskusi yang matang. Salah satu model dalam pembangunan perangkat lunak adalah dengan model sekuensial linear atau biasa disebut model *waterfall.* [(Pressman, 2002)]

Model tersebut terdiri dari beberapa tahapan, yaitu :



Gambar 2. 3 : *Model Waterfall* [(Pressman, 2002)]

Keterangan Tahapan [(Pressman, 2002)] :

a. *Rekayasa dan pemodelan sisem/informasi*

   Rekayasa dan analisis sistem menyangkut pengumpulan kebutuhtan pada tingkat sistem dengan sejumlah kecil analisis serta desain tingkat puncak. Rekayasa informasi mencakup juga pengumpulan kebutuhan pada tingkat bisnis strategis dan tingkat area bisnis.

b. *Analisis kebutuhan perangkat lunak*

   Proses pengumpulan kebutuhan diintensifkan dan difokuskan, khususnya pada perangkat lunak. Untuk memahami sifat program yang dibangun, seorang analis harus memahami domain informasi, tingkah laku, unjuk kerja, dan antar muka yang diperlukan. Kebutuhan baik untuk sistem maupun perangkat lunak didokumentasikan dan dilihat lagi dengan pelanggan.

c. *Perancangan / Design*

   Perancangan perangkat lunak sebenarnya adalah proses multi langkah yang berfokus pada empat atribut sebuah program yang bebeda; struktur data, arsitektur perangkat lunak, representasi *interface,* dan detail *(algoritma)* prosedural. Proses perancangan menerjemahkan syarat / kebutuhan ke dalam sebuah representasi perangkat lunak yang dapat diperkirakan demi kualitas sebelum dimulai pemunculan kode. Sebagaimana persyaratan, perancangan didokumentasikan dan menjadi bagian da konfigurasi perangkat lunak.

d. Generasi kode / Coding

Perancangan harus diterjemahkan ke dalam bentuk mesin yang bisa dibaca. Langkah pembuat kode (programer) melakukan tugas ini. Jika perancangan dilakukan dengan cara yang lengkap, pembuatan kode dapan diselesaikan secara mekanis.

e. Pengujian

Proses pengujian berfokus pada logika internal perangkat lunak, memastikan bahwa semua pernyataan sudah diuji, dan pada eksternal fungsionalitas, yaitu mengarahkan pengujian untuk menemukan kesalahan-kesalahan dan memastikan bahwa input yang dibatasi akan memberikan hasil aktual yang sesui dengan hasil yang dibutuhkan.

f. Pemeliharaan / Maintenance

Pemeliharaan perangkat lunak mengaplikasikan lagi setiap fase program sebelumnya dan tidak membuat yang baru lagi.

2.2.2. *Unfied Modelling Language* (UML)

Unfied Modelling Language (UML) adalah sebuah "bahasa" yang menjadi standar dalam industri untuk visualisasi, merancang dan mendokumentasikan sistem piranti lunak. UML menawarkan sebuah standar untuk merancang sebuah sistem. [(Dharwiyanti & Wahono, 2003)].

Dengan menggunakan UML kita dapat membuat model untuk semua jenis aplikasi piranti lunak, dimana aplikasi tersebut dapat berjalan pada piranti keras, sistem operasi dan jaringan apapun, serta ditulis dalam bahasa

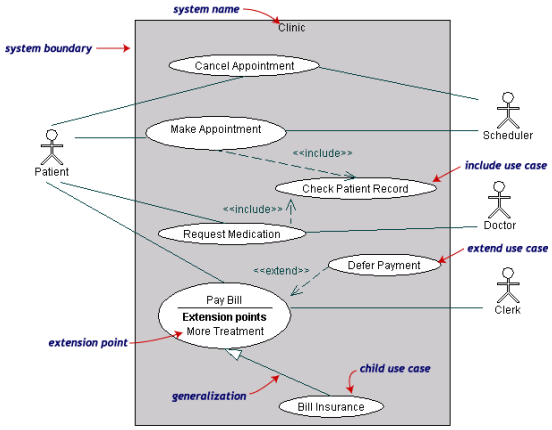
pemrograman apapun. Tetapi karena UML juga menggunakan class dan operation dalam konsep dasarnya, maka ia lebih cocok untuk penulisan piranti lunak dalam bahasa-bahasa berorientasi objek seperti C++, Java, C# ata VB.NET. Walaupun demikian, UML tetap digunakan untuk modeling aplikasi prosedural dalam VB atau C. [(Dharwiyanti & Wahono, 2003)]

Pada UML dikenal beberapa diagram, diantaranya sebagai berikut :

2.2.2.1.*Use Case Diagram*

Use Case Diagram menggambarkan fungsionalitas yang diharapkan dari sebuah sistem. Yang ditekankan adalah “apa” yang diperbuat sistem, dan bukan “bagaimana”. Sebuah use case merepresentasikan sebuah interaksi antara aktor dengan sistem. Use case merupakan sebuah pekerjaan tertentu, misalnya login ke sistem, meng-*create* sebuah daftar belanja, dan sebagaimanya. Seorang / sebuah aktor adalah sebuah entitas manusia atau mesin yang berinteraksi dengan sistem untuk melakukan pekerjaan-pekerjaan tertentu. [(Dharwiyanti & Wahono, 2003)]

Contoh use case diagram :



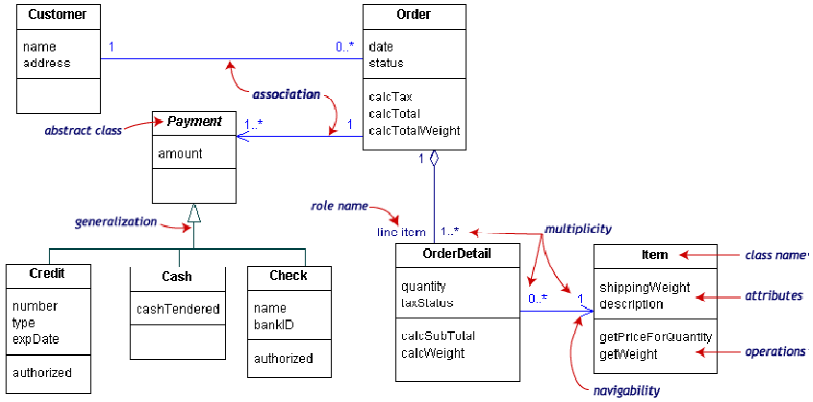
Gambar 2. 4 : Use Case Diagram [(Dharwiyanti & Wahono, 2003)]

2.2.2.2.*Class Diagram*

Class adalah sebuah spesifikasi yang jika diinstanisasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek. *Class* menggambarkan keadaan (atribut/properti) suatu sistem, sekaligus menawarkan layanan untuk manipulasi keadaan tersebut (metode/fungsi).

Class diagram menggambarkan struktur dan deskripsi *class, package* dan objek beserta hubungan satu sama lain seperti containment, pewarisan, asosiasi, dan lain-lain. [(Dharwiyanti & Wahono, 2003)]

Contoh class diagram :



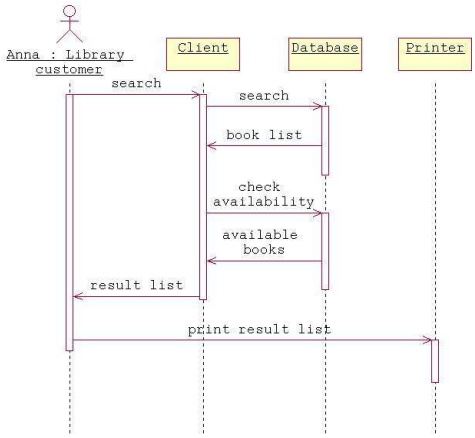
Gambar 2. 5 : Class Diagram [(Dharwiyanti & Wahono, 2003)]

2.2.2.3.*Sequence Diagram*

Sequence Diagram menggambarkan interaksi antar objek di dalam dan di sekitar sistem (termasuk pengguna, display, dan sebagainya) berupa *message* yang digambarkan terhadap waktu. Sequence diagram terdiri atas dimensi *vertical* (waktu) dan dimensi horizontal (objek-objek yang terkait).

Sequence diagram biasa digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan sebagai respon dari sebuah event untuk menghasilkan output tertentu. Diawali dari apa yang men-trigger aktivitas tersebut, proses dan perubahan apa saja yang terjadi secara internal dan output apa saja yang dihasilkan. [(Dharwiyanti & Wahono, 2003)]

Contoh sequence diagram :



Gambar 2. 6 : Sequence Diagram [(Dharwiyanti & Wahono, 2003)]

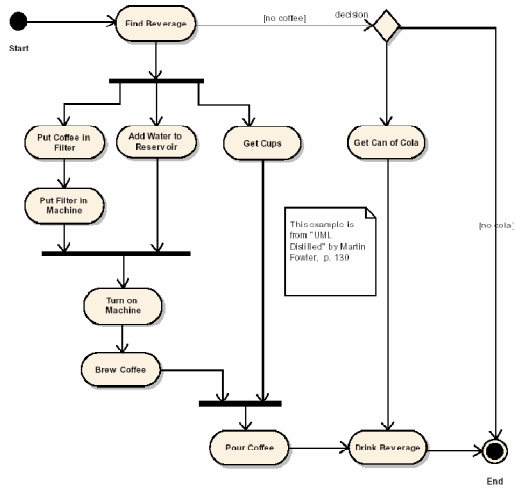
2.2.2.4.*Activity Diagram*

Activity diagram menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, *decision* yang mungkin terjadi, dan bagiamana mereka berakhir. *Activity diagram* juga dapat menggambarkan proses parallel yang mungkin terjadi pada beberapa eksekusi.

Activity diagram merupakan *state diagram* khusus, dimana sebagian besar *state* adalah *action* dan sebagian besar transisi di-triger oleh selesainya *state* sebelumnya (internal processing). Oleh karena itu *activity diagram* tidak menggambarkan *behavior internal* sebuah sistem (dan interaksi antar subsistem) secara eksak, tetapi lebih menggambarkan proses-proses dan

jalur-jalur aktivitas dari level atas secara umum. [(Dharwiyanti & Wahono, 2003)]

Contoh activity diagram :



Gambar 2. 7 : Activity Diagram [(Dharwiyanti & Wahono, 2003)]

2.2.3. Basisdata / *database relasional*

Seiring dengan meningkatnya kemampuan komputer dan jumlah data yang diolah, yang semakin hari semakin banyak, maka masalah pengolahana data menjadi semakin kompleks. Oleh sebab itu kemudian diciptakan suatu teknik pengolahan data yang disebut *Managemen Database Relasional*. Dengan manajemen *database relasional* kita dapat meningkatkan efisiensi akses, kecepatan pengolahan, pengembangan maupun kemudahan dalam pemeliharaan data. [(Rickyanto, 2004)]

Dalam manajemen database relasional dikenal komponen – komponen penting penanganan data. Berikut ini adalah komponen utama dalam konsep database relasional : [(Dharwiyanti & Wahono, 2003)]

1. Tabel

   Tabel merupakan entitas yang merupakan suatu kesatuan unit dari kumpulan *row* / *record* dengan atribut-atributnya (column).

2. Baris

   Baris merupakan elemen dari tabel dan sering disebut *record* atau *row*. Baris merupakan elemen data dari tabel yang memiliki atribut-atribut atau sering disebut *field* atau kolom. Baris atau *row* dalam tabel mewakili kesatuan data yang lebih kecil dari tabel yang biasanya menyimpan suatu masukan objek data, misalnya data pegawai yang terdiri data atribut seperti nama, alamat, kota di mana seorang pegawai tinggal.

3. Kolom

   Kolom atau column sering disebut juga atribut atau field dari baris / *row*. Column adalah pembentuk suatu baris dan mewakili atribut terdiri dari suatu *record.*

4. Index

   Index diperlukan untuk meningkatkan performa, khususnya dalam mendapatkan record dalam database. Index dapat menggunakan kolom tertentu sebagai *key* index atau gabungan dari beberapa kolom sebagai *composite key.*

5. *Primary key*

   Primary key merupakan elemen penting untuk mempertahankan integritas data. Dengan adanya *primary key* maka kita dapat mencegah terjadinya redudansi atau data ganda yang dapat berpengaruh pada integritas data dan jalannya program pengolahan data. Oleh karena itu *primary key* diperlukan untuk mendapatkan keunikan data atau record.

6. *Foreign key*

   Selain *primary key, foreign key* diperlukan untuk relasi antartabel. Dengan *foreign key* kita dapat meningkatkan efisiensi penyimpanan data maupun dalam mendapatkan data.

2.2.4. Konsep berorientasi objek

Teknologi objek oriented memandang *software* sebagai sebuah interaksi antarbagian atau subsistem dalam sebuah sistem, dan menggambarkan satu bagian tersebut dalam satu objek.

OOP (*object oriented programming*) adalah metode pemrograman yang mengikuti konsep-konsep berorientasi objek, yakni *encapsulation* (pengapsulan), *inheritance* (pewarisan), *polimorfisme* (keragaman/bentuk lain). [(Huda, Subiyanto, & Nugroho, 2008)]

Dalam konsep berorientasi objek dikenal beberapa istilah sebagai berikut:

a. *Class*

   Class adalah cetak biru (rancangan) dari objek. Ini berarti bisa dibuat banyak objek dari satu macam class. *Class* mendefiniskan sebuah tipe dari

objek. Di dalam class dapat dideklarasikan variabel dan diciptakan *object* (instansiasi). Sebuah *class* mempunyai anggota (member) yang terdiri atas atribut dan method.

b. Object

Object (objek) secara lugas dapat diartkan sebagai insatansiasi atau hasil ciptaan dari suatu class yang telah dibuat sebelumnya. Dalam pengembangan program orientasi objek lebih lanjut, sebuah objek dapat dimungkinkan terdiri atas objek-objek lain. Seperti halnya objek mobil terdiri atas mesin, ban, kerangka mobil, pintu, karoseri dan lain-lain. Atau, bisa jadi sebuah objek merupakan turunan dari objek lain sehingga mewarisi sifat-sifat induknya.

c. Atribut

Atribut menunjuk pada elemen data dari sebuah object. Atribut menyimpan informasi tentang object. Dikenal juga sebagai member data, variabel instance, properti atau sebuah field data.

d. Method

Method dikenal juga sebagai suatu *function* dan *procedure*. Dalam OOP, *method* digunakan untuk memodularisasi program melalui pemisahan tugas dalam suatu class. Pemanggilan method menspesifikasikan nama method dan menyediakan informasi (parameter) yang diperlukan untuk melaksanakan tugasnya.

e. Constructor

Constructor adalah tipe khusus *metho*d yang digunakan untuk menginstansiasi atau menciptakan sebuah objek. Nama *constructor* adalah sama dengan nama kelasnya. Selain itu, constructor tidak bisa mengembalikan suatu nilai (not return value) bahkan void sekalipun.

f. Package

Package menunjuk pada pengelompokkan *class* dan/atau *subpackages*. Strukturnya dapat disamakan dengan direktorinya.

2.2.5. Arsitektur *Client-Server*

Client-server adalah arsitektur komputasi yang memisahkan sisi *client* dari *server* dan paling banyak diimplementasikan melalui jaringan komputer. Setiap *client* atau *server* dikoneksikan ke jaringan sebagai sebuah node. Tipe dasar arsitektur *client server* hanya terdiri dari 2 node: *client* dan *server*. Tipe ini sering disebut dengan *two-tier*.

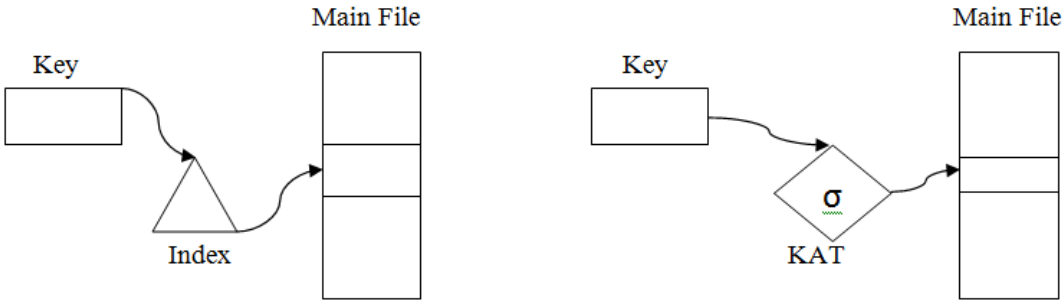
Setiap perangkat lunak *client* dapat mengirim *request* data kepada satu atau lebih server yang terhubung. Sebaliknya, *server* dapat menerima *request in*, memprosesnya, dan mengembalikan informasi yang diminta kepada *client*. Contoh *client* yaitu web browser atau aplikasi database. *Server* dapat meliputi *web server*, *database server,* dan *mail server*.

Pendekatan arsitektur *client-server* memperkenalkan sebuah *server database* untuk menggantikan *file server*. Menggunakan *Relational Database Management System* (RDBMS), query-query dari *user* dapat dijawab secara langsung. Dalam arsitektur c*lient-server, statement Remote Procedure Calls* (RPCs) atau SQL digunakan untuk berkomunikasi antara *client* dan *server*.

2.2.6. Teknik Hashing (*Hash*)

Pada file hash, pengaksesannya dilakukan secara langsung (direct access) seperti file berindeks tapi, file hash memakai fungsi dari nilai key yang disebut fungsi hash atau KAT (Key-to-address Transformation). [(J & Imbar, 2008)]

Gambar berikut ini menunjukkan perbedaan mendasar antara file berindeks dan file hash



Gambar 2. 8 : File Berindeks vs File Hash [(J & Imbar, 2008)]

Pada file berindeks satu record diambil melalui alamat indeks, sedangkan pada *file hash* yang menentukan posisi *record* adalah *key* nya sendiri.

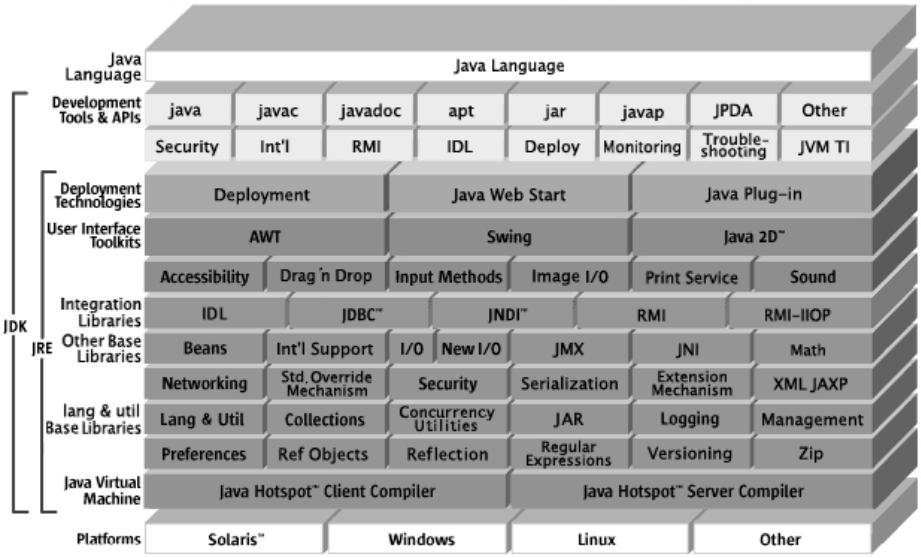
Fungsi hash (*f(hash))* adalah penentuan alamat *record* pada *file* data dengan melakukan perhitungan nilai kunci (*key*) dari *record.*

Nilai *key* → f (hash) → alamat relatif.

Tujuan dari hashing adalah untuk menemukan fungsi yang memetakan setiap nilai *key* ke dalam nilai lokasi/alamat kosong. [(J & Imbar, 2008)].

2.2.7. Java

Java dibuat dan diperkenalkan pertama kali oleh sebuah tim *Sun Microsystem* yang dipimpin oleh Patrick Naughton dan James Gosling pada tahun 1991 dengan code nama Oak. Tahun 1995 Sun mengubah nama Oak tersebut menjadi Java. Ide pertama kali kenapa java dibuat adalah karena adanya motivasi untuk membuat sebuah bahasa pemrograman yang bersifat *portable* dan *platform independent* (tidak tergantung mesin dan sistem operasi) yang dapat digunakan untuk membuat piranti lunak yang dapat ditanamkan (*embedded*) pada berbagai macam peralatan elektronik consumer biasa, seperti microwave, remote control, telepon, card reader dan sebagainya.



Gambar 2. 9 : Arsitektur Java

Berdasarkan white paper resmi dari SUN, Java memiliki karakteristik berikut:

1. **Sederhana.** Bahasa pemrograman Java menggunakan sintaks mirip dengan C++ namun sintaks pada Java telah banyak diperbaiki terutama menghilangkan penggunaan pointer yang rumit dan multiple inheritance. Java juga menggunakan automatic memory allocation dan memory garbage collection.
2. **Berorientasi objek (Object Oriented).** Java mengunakan pemrograman berorientasi objek yang membuat program dapat dibuat secara modular dan dapat dipergunakan kembali. Pemrograman berorientasi objek memodelkan dunia nyata kedalam objek dan melakukan interaksi antar objek-objek tersebut.
3. **Dapat didistribusi dengan mudah.** Java dibuat untuk membuat aplikasi terdistribusi secara mudah dengan adanya libraries networking yang terintegrasi pada Java.

4. **Interpreter**. Program Java dijalankan menggunakan interpreter yaitu Java Virtual Machine (JVM). Hal ini menyebabkan source code Java yang telah dikompilasi menjadi Java bytecodes dapat dijalankan pada platform yang berbeda-beda.
5. **Robust.** Java mempuyai reliabilitas yang tinggi. Compiler pada Java mempunyai kemampuan mendeteksi error secara lebih teliti dibandingkan bahasa pemrograman lain. Java mempunyai runtime-Exception handling untuk membantu mengatasi error pada pemrograman.
6. **Aman.** Sebagai bahasa pemrograman untuk aplikasi internet dan terdistribusi, Java memiliki beberapa mekanisme keamanan untuk menjaga aplikasi tidak digunakan untuk merusak sistem komputer yang menjalankan aplikasi tersebut.
7. ***Architecture Neutral*.** Program Java merupakan platform independent. Program cukup mempunyai satu buah versi yang dapat dijalankan pada platform yang berbeda dengan Java Virtual Machine.
8. ***Portabel.*** Source code maupun program Java dapat dengan mudah dibawa ke platform yang berbeda-beda tanpa harus dikompilasi ulang.
9. ***Performance.*** Performance pada Java sering dikatakan kurang tinggi. Namun performance Java dapat ditingkatkan menggunakan kompilasi Java lain seperti buatan Inprise, Microsoft ataupun Symantec yang menggunakan Just In Time Compilers (JIT).

10. ***Multithreaded.*** Java mempunyai kemampuan untuk membuat suatu program yang dapat melakukan beberapa pekerjaan secara sekaligus dan simultan.
11. **Dinamis.** Java didesain untuk dapat dijalankan pada lingkungan yang dinamis. Perubahan pada suatu class dengan menambahkan properties ataupun method dapat dilakukan tanpa menggangu program yang menggunakan class tersebut.

2.2.8. MySQL

MySQL adalah sebuah perangkat lunak sistem manajemen basis data SQL atau DBMS yang multithread, multi-user, dengan sekitar 6 juta instalasi di seluruh dunia. MYSQL AB membuat MySQL tersedia sebagai perangkat lunak gratis di bawah lisensi komersial untuk kasus-kasus dimana penggunaannya tidak cocok dengan penggunaan GPL. [(Solichin, 2010)].

Tidak seperti PHP atau Apache yang merupakan software yang dikembangkan oleh komunitas umum, dan hak cipta untuk kode sumber dimiliki oleh penulisnya masing-masing, MySQL dimiliki dan di sponsori oleh sebuah perusahaan komersial Swedia yaitu MySQL AB. MySQL AB memegang penuh hak cipta hamper atas semua kode sumbernya. Kedua orang Swedia dan satu orang Finlandia yang mendirikan MySQL AB adalah : David Axmark, Allan Larson, dan Michael “Monty” Widenius.

Fitur-fitur MySQL antara lain : [(Solichin, 2010)]

1. *Relational Database System*, seperti *halnya software database* lain yang ada di pasaran, MySQL termasuk DBMS.
2. Arsitektur *Client-Server*. MySQL memiliki arsitektur *client-server* dimana server database MySQL terinstal di *server*. *Client* MySQL dapat berada di computer yang sama dengan *server*, dan dapat juga di computer lain yang berkomunikasi dengan *server* melalui jaringan bahkan internet.
3. Mengenal perintah SQL standar. SQL (*Structured Query Language*) merupakan suatu bahasa standar yang berlaku di hampir semua *software database.* MySQL mendukung SQL versi SQL:2003.
4. Mendukung *sub select*. Melalui versi 4.1 MySQL telah mendukung *select* dalam *select (sub select).*
5. Mendukung views. MySQL mendukung views sejak versi 5.0
6. Mendukung Stored Prosedured (SP). MySQL mendukung SP sejak versi 5.0.
7. Mendukung Triggers. MySQL mendukung trigger pada versi 5.0 namun masih terbatas. Pengembang MySQL berjanji akan meningkatkan kemampuan trigger pada versi 5.1.
8. Mendukung *replication.*
9. Mendukung transaksi.
10. Mendukung *foreign key.*
11. Tersedia fungsi GIS.
12. Free (bebas didownload).
13. Stabil dan tangguh.

14. Fleksibel dengan berbagai pemrograman.
15. Security yang baik.
16. Dukungan dari banyak komunitas.
17. Perkembangan *software* cukup cepat.

2.2.8.1. *STORAGE ENGINE AND INDEXING*

MySQL mendukung beberapa *storage engine* yang berperan untuk menangani berbagai tipe tabel yang ada. Berikut adalah berbagai storage engine yang tersedia pada MySQL [(Santoso, Cahyadi, Ardian, Lokasasmita, & Mirza, 2006)]:

1. *MyISAM*

   Mengatur *non-transactional* tabel. Engine ini dapat melakukan penyimpanan dan pengambilan data dengan cepat. *MyISAM* adalah default storage engine pada MySQL.

   MyISAM memiliki beberapa karakteristik sebagai berikut:

   - Seperti panjang key maksimum adalah 1000 bytes, maksimum jumlah kolom per indeks adalah 16.
   - Nilai numerik disimpan pada high byte sehingga dapat melakukan kompresi secara lebih baik, dapat melakukan *AUTO_INCREMENT*, dan masih banyak yang lainnya, yang tidak akan kita bahas satu per satu di sini.
   - Dapat menaruh data file dan index file dalam direktori yang berbeda untuk meningkatkan kecepatan akses.

2. *MEMORY*

Menghasilkan *in-memory table*. *Storage engine MERGE* dapat mengatur beberapa tabel *MyISAM* sebagai sebuah tabel tunggal. *Storage engine MEMORY* dan *MERGE* khusus untuk mengatur *non-transactional* tabel. *Memory storage engine* ini dahulu dikenal dengan nama *Heap engine*. Memori membuat tabel dengan isinya disimpan di memori. Setiap tabel memori memiliki satu buah *file* yang akan disimpan dalam disk, ekstensinya adalah .frm.

MEMORY memiliki beberapa karakteristik sebagai berikut:

- Menggunakan 100% *dynamic hashing* untuk operasi insert, sehingga lebih hemat karena tidak perlu tempat tambahan untuk menyimpan *key*.
- Kita dapat memiliki *non-unique keys* di tabel *MEMORY*.
- Dapat memiliki *NULL values*, menggunakan panjang yang tetap dalam hal penyimpanan, kolom tidak bisa mengandung *BLOB* atau *TEXT*.

3. InnoDB dan BDB

Menghasilkan *transaction-safe table*. Maksud dari *transaction safe table* adalah tabel yang memiliki kemampuan *commit, rollback, crash recovery*. Namun dengan kemampuan yang tersebut, tentu saja ada overhead yang dimiliki.Kita dapat mengaktifkan atau menon-aktifkan engine bertipe ini.

4. EXAMPLE

Example adalah *engine* yang tidak melakukan apa-apa. Kita dapat membuat tabel dengan tipe ini, tetapi tidak ada data yang dapat disimpan atau dibaca dari tabel ini, sesuai dengan namanya, *engine* ini hanya sekedar memberikan contoh bagaimana kode dari MySQL digunakan.

Tujuan dari *engine* ini adalah untuk menunjukkan bagaimana cara memulai untuk menulis sebuah *storage engine.*

5. NDB cluster

   NDB cluster adalah *engine* yang digunakan MySQL untuk mengimplementasi tabel yang terpisah di lebih dari satu komputer. Saat ini NDB hanya didukung pada sistem *Linux, Solaris* dan *Mac OS X.* Namun pada versi-versi selanjutnya akan akan dikembangkan versi selanjutnya (hingga saat paper ini ditulis, MySQL terakhir adalah versi 5.1) yang akan mendukung NDB pada *Windows*, dan juga platform lainnya.

6. ARCHIVE

   Digunakan untuk menyimpan data dengan ukuran yang sangat besar tanpa memerlukan index. Engine ini mensupport operasi *INSERT* dan *SELECT,* namun tidak untuk operasi *DELETE, REPLACE*, dan *UPDATE*. Dalam engine ini juga dikenal dengan menggunakan *row-level locking.*

7. CSV

   Digunakan untuk menyimpan data dalam *text-file* dimana antara satu nilai dengan nilai yang lainnya dibatasi oleh koma. Saat kita membuat tabel CSV, selain membuat file yang berisikan *table format, storage engine* juga membuat data file yang berupa *teks file* biasa. Ketika kita menyimpan data, *storage engine* menyimpannya ke dalam data file dengan menggunakan koma untuk memisahkan nilai-nilai yang ada.

8. *BLACKHOLE*

   Menerima setiap data yang diberikan, tetapi tidak menyimpan data dan selalu mengembalikan data berupa himpunan kosong. *Blackhole* mendukung semua jenis pengindeksan., kegunaan dari *blackhole* ini antara lain:

   - verifikasi dari *syntax dump file*
   - pengukuran *overhead* yang terjadi dengan membandingkan dengan ketika menggunakan *blackhole*
   - mencari performance *bottlenecks* yang tidak terkait dengan *storage engine* itu sendiri.

9. *FEDERATED*

   Menyimpan data di *database* dengan akses terbatas (jarak jauh). Saat ini *engine* ini hanya dapat bekerja dengan menggunakan *MySQL C Client API*. Dengan menggunakan *Federated storage engine*, tidak akan ada lagi lokal data files, melainkan *remote database* menyimpan data yang umumnya akan ada di tabel. Alur informasi yang terjadi adalah sebagai berikut:

a. *SQL calls issued locally*
b. *MySQL handler API (data in handler format)*
c. *MySQL client API (data converted to SQL calls)*
d. *Remote database -> MySQL client API*
e. *Convert result sets (if any) to handler format*
f. *Handler API -> Result rows or rows-affected count to local*

2.2.8.2.*Partitioning*

Partitioning memungkinkan kita untuk mendistribusikan data-data di suatu tabel yang sama ke *filesystem* yang berbeda di tempat yang berbeda pula. MySQL mendukung horizontal partitioning yang berarti row-row yang berbeda akan didistribusikan ke partisi yang berbeda. Ada empat tipe *partition* yang didukung oleh MySQL, berikut adalah deskripsi dari keempat tipe tersebut:

1. Range Partitioning

Tabel yang dipartisi menggunakan *range* dipartisi dimana setiap partisinya memiliki *row* yang mengandung *variabel* yang berada pada rentang nilai tertentu. Rentang nilai (*range*) dapat kontinu tetapi tidak overlap dan didefinisikan melalui operator *VALUES LESS THAN*.

Partisi jenis ini akan berguna pada kondisi sebagai berikut:

- jika kita ingin menghapus data lama, terutama untuk data yang besar, kita dapat melakukan penghapusan ataupun perubahan secara efisien, contoh: DELETE FROM employees WHERE YEAR(separated) <= 1990;
- jika kita ingin menggunakan kolom berisikan tanggal atau waktu.
- jika kita akan sering menjalan query yang bergantung pada kolom yang dipartisi.

2. List Partitioning

Pada *list partitioning*, setiap partisi didefinisikan dan diklasifikasikan berdasarkan keanggotaan sebuah nilai suatu kolom terhadap suatu daftar nilai. Partisi

dilakukan oleh perintah *PARTITION BY LIST* (expr) di mana expr adalah nama kolom yang hendak dijadikan dasar partisi dan setiap partisi didefinisikan melalui perintah *VALUES IN* (*value_list*) di mana *value_list* adalah daftar bilangan bulat yang dipisahkan oleh koma.

3. Hash Partitioning

Partisi dengan *hash* dilakukan untuk menjamin pendistribusian yang adil (seragam) pada setiap data yang dimasukkan. Dengan partisi jenis ini, kita tidak perlu mendefinisikan keanggotaan setiap partisi. Kita hanya perlu menspesifikasi suatu ekspresi yang memiliki input suatu nilai kolom dan mempunyai output nomor partisi dari data tersebut. MySQL juga mendukung suatu varian dari *hash* partitioning yaitu *LINEAR HASH partitioning*.

4. Key Partitioning

Partisi dengan *key* memiliki prinsip yang hampir sama dengan *hash partitioning*. Yang menjadi perbedaan adalah fungsi hashing untuk partisi jenis ini dibuat langsung oleh server MySQL dengan fungsi MD5() pada MySQL *Clusters*. Untuk *storage engine* yang lain, *server* telah mengembangkan fungsi yang menerapkan algoritma yang hampir sama bernama *PASSWORD().*

2.2.8.3. *Indexing*

Index adalah fitur dalam database yang membuat kita dapat mengakses *row* pada tabel. Index dapat digunakan untuk optimasi proses *searching*, biasanya melalui

balanced tree. DBMS yang ada saat ini dapat melakukan *indexing* melalui R-/R+ *tree, hash, expression, partial, reverse, bitmap,* dan GiST.

Dari berbagai metode *indexing* diatas, MySQL hanya dapat mengimplementasikan B/B+ *tree* dan *hash* dimana B/B+ *tree* digunakan oleh tabel MyISAM, InnoDB dan HEAP dan hash digunakan oleh tabel HEAP.

Storage Engine	Allowable Index Types
MyISAM	BTREE
InnoDB	BTREE
Memory / Heap	Hash, BTREE