OPTIMASI PENGOLAHAN DATA NILAI SISWA PADA SMK NEGERI 2 PRABUMULIH MENGGUNAKAN METODE COST BASED

PROPOSAL PENELITIAN

Diajukan guna melakukan penelitian skripsi

OLEH:

ALKAT IPRIONO

08142013

PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS ILMU KOMPUTER

UNIVERSITAS BINA DARMA

PALEMBANG

TAHUN 2013

LEMBAR PENGESAHAN

OPTIMASI DATA NILAI SISWA PADA SMK NEGERI 2 PRABUMULIH MENGGUNAKAN METODE COST BASED

OLEH :

ALKAT IPRIONO
08142013

PROPOSAL PENELITIAN

Disusun sebagai salah satu syarat untuk melakukan penelitian skripsi

Disetujui,

Palembang, April 2013
Program Studi Teknik Informatika
Fakultas Ilmu Komputer
Universitas Bina Darma Palembang
Ketua

Pembimbing I

(M. Izman Herdiansyah. ST., M.M., Ph.D) **(Syahril Rizal, ST., M.M., M.Kom)**

Pembimbing II

(A. Mutakin Bakti, M.M., M.Kom)

UNIVERSITAS BINA DARMA
Sk. Menteri Pendidikan Nasional Republik Indonesia No.112/D/O/2002
Jln. Jendral Ahmad Yani No. 12 Palembang 30264
Telp. (0711) 515679 fax. (0711) 515679
Website : www.binadarma.ac.id E-mail : bidar@binadarma.ac.id

LEMBAR KONSULTASI PROPOSAL

Nama : Alkat Ipriono
Nim : 08142013
Program Studi : Teknik Informatika
Judul : Optimasi Data Nilai Siswa Pada SMK Negeri 2 Prabumulih Menggunakan Metode Cost Based

Dosen Pembimbing II : A. Mutakin Bakti, M.M., M.Kom

No	Tanggal	Keterangan	Paraf
	5/4 2013	Latar Belakang	
		~~Latar~~ Tinjauan pustaka	
	8/4 2013	Latar belakang dan tujuan	
		Query, RDB	
	10/4 2013		
	11/4/2013	Proposal	
		Lengkapi	
		ACC proposal	

UNIVERSITAS BINA DARMA
Sk. Menteri Pendidikan Nasional Republik Indonesia No.112/D/O/2002
Jln. Jendral Ahmad Yani No. 12 Palembang 30264
Telp. (0711) 515679 fax. (0711) 515679
Website : www.binadarma.ac.id E-mail : bidar@binadarma.ac.id

LEMBAR KONSULTASI PROPOSAL

Nama : *Alkat Ipriono*
Nim : 08142013
Program Studi : Teknik Informatika
Judul : Optimasi Data Nilai Siswa Pada SMK Negeri 2 Prabumulih Menggunakan Metode Cost Based
Dosen Pembimbing I : M. Izman Herdiansyah. ST., M.M., Ph.D

No	Tanggal	Keterangan	Paraf
	16/4 13	Bawa jurnal - Print Daftar pustaka	
	23/4	judul ditambahi: Optimasi Pengelolaan - - - - - -	
		Acc proposal & ujian	

PROPOSAL PENELITIAN

OPTIMASI PENGOLAHAN DATA NILAI SISWA PADA SMK NEGERI 2 PRABUMULIH MENGGUNAKAN METODE COST BASED

I. PENDAHULUAN

1.1 Latar Belakang

Kemajuan zaman dan teknologi saat ini telah merubah pola dalam perusahaan ataupun organisasi-organisasi. Data yang besar dan telah terkumpul dalam waktu yang lama dapat diolah menjadi sumber informasi yang dapat membantu menganalisis eksistensi dari sebuah perusahaan atau organisasi. Analisa otomatis dari data yang berjumlah besar atau kompleks yang terbentuk sebagai data mining, dengan tujuan untuk menemukan pola atau kecenderungan yang penting yang biasanya tidak disadari keberadaannya saat ini telah banyak di lakukan.

Database atau sering juga disebut basis data adalah sekumpulan informasi yang disimpan dalam komputer secara sistematik dan merupakan sumber informasi yang dapat diperiksa menggunakan suatua program komputer *database* berfungsi untuk menyimpan informasi atau data. Untuk mengelolah *database* diperlukan *software* yang sering disebut dengan *DBMS* (*database management system*). Dengan *DBMS* pengguna atau user dapat membuat, mengelolah, mengontrol dan mengakses *database* dengan mudah, praktis dan efisien. *Database* terdiri dari tabel yang didalamnya terdapat *field-field* , dan sebuah *database* bisa terdiri dari beberapa tabel.

Permasalahan dalam mengelolah *database* saat ini tidak cepat dan efisen penggunaan *query* pada suatu aplikasi meskipun menggunakan RDBMS yang berbeda-beda. Penggunaan *query* yang tidak teroptimasi juga memperlambat informasi yang ingin di tampilkan, karena terjadi *script time out* yang dilakukan oleh aplikasi yang akan mengakibatkan proses penampilan informasi berhenti.

Optimisasi *query* mencoba memberikan suatu pemecahan untuk menangani masalah tersebut dengan cara menggabungkan sejumlah besar teknik-teknik dan strategi, yang meliputi transformasi-transformasi logika dari *query-query* untuk mengoptimisasi jalan akses dan penyimpanan data pada sistem *file* terutama pada data yang besar dan lama tersimpan. Setelah ditransformasikan, sebuah *query* harus dipetakan ke dalam sebuah urut-urutan operasi untuk menghasilkan data yang diminta. Hal ini dapat membantu suatu organisasi dalam mengolah informasi dari data mining yang terkumpul dengan pengaksesan data yang optimal

Metode *cost based* merupakan optimasi data, teknik ini mengoptimasikan *cost* yang dipergunakan dari beberapa alternatif untuk kemudian dipilih salah satu yang menjadi *cost* terendah. Teknik ini mengoptimalkan urutan join terbalik yang dimungkinkan pada relasi-relasi. Untuk melihat kinerja sistem basis data dalam pencarian data yang akan digunakan, yaitu *cross product*. Cross product atau disebut juga cross join digunakan saat mengkombinasikan data pada dua tabel atau lebih. *Query* dengan *cross product* melakukan *select* sejumlah kolum dari dua tabel atau lebih dimana pengkondisian masing-masing tabel di *join*.

Alasan menggunakan algoritma *cross product* pada penelitian ini agar hasil pencarian data nilai siswa ditampilkan cepat dan menajemen baik. Permasalahan tersebut diatas menjadi alasan penulis untuk melakukan penelitian dengan judul **"Optimasi Pengolahan Data Nilai Siswa Pada SMK Negeri 2 Prabumulih Menggunakan Metode Cost Based".**

1.2 Perumusan Masalah

Dari uraian permasalahan diatas, maka penulis merumuskan masalah dalam penelitian ini adalah *"*Bagaimana optimasi pengolahan data nilai siswa pada SMK Negeri 2 Prabumulih menggunakan metode *cost based* ? ".

1.3 Batasan Masalah

Peneliti membatasi masalah yang akan diteliti hanya pada optimasi pengolahan data nilai siswa pada SMK Negeri 2 Prabumulih menggunakan metode *cost based* menggunakan data siswa, data pelajaran dan data nilai. Maka metode yang digunakan berdasarkan latar belakang diatas yaitu, metode *cost product* dan basis data menggunakan *MySQL*.

1.4 Tujuan dan Manfaat Penelitian

1.4.1. Tujuan Penelitian

Tujuan dari penelitian ini adalah :

1. Menganalisa dan merancang penggunaan *query SQL* dalam pengolahan basis data siswa, guru dan nilai.
2. Mengoptimasi penggunaan *query SQL* dalam basis data manajemen sistem.

1.4.2. Manfaat Penelitian

Manfaat dari penelitian ini adalah :

1. Bagi Sekolah
   a. Membantu pegawai pendataan sistem akademik khususnya pada pendataan guru, siswa dan penilaian siswa.
   b. Membantu pendataan sistem akademik yang dibutuhkan secara optimal dengan metode *cost product*.
2. Bagi Penulis
   a. Menjadi sumber pembelajaran untuk mengembangkan ilmu pengetahuan dan teknologi di bidang komputer yang telah diterima selama mengikuti perkuliahan di Universitas Bina Darma Palembang.
   b. Mempelajari dan memahami operasi-operasi dasar yang digunakan untuk mengeksekusi *query*.
   c. Menganalisis proses *query* optimasi pada basis data.

II. TINJAUAN PUSTAKA

Dalam sub bab berikut akan dijelaskan teori-teori yang berhubungan dengan penelitian, tinjauan obyek penelitian dan alat bantu yang digunakan dalam penelitian.

2.1 Landasan Teori

2.1.1 *Optimasi Query*

Optimasi *query* adalah suatu proses untuk menganalisa *query* untuk menentukan sumber-sumber apa saja yang digunakan oleh *query* tersebut dan

apakah penggunaan dari sumber tersebut dapat dikurangi tanpa merubah output. Atau bisa juga dikatakan bahwa optimasi *query* adalah sebuah prosedur untuk meningkatkan strategi evaluasi dari suatu *query* untuk membuat evaluasi tersebut menjadi lebih efektif. Optimasi *query* mencakup beberapa teknik seperti transformasi query ke dalam bentuk logika yang sama, memilih jalan akses yang optimal dan mengoptimumkan penyimpanan data. (Siallagan, 2008:1).

Optimasi *query* menemukan jalan akses yang termurah untuk meminimumkan total waktu pada saat proses sebuah *query*. Untuk mencapai tujuan tersebut, maka diperlukan optimizer untuk melakukan analisa *query* dan untuk melakukan pencarian jalan akses. Teknik optimasi dapat terdapat 2 pendekatan optimasi sebagaimana diungkapkan oleh (Chanowich, 2001:2), yakni:

1. Heuristik atau *rule-based*

   Teknik ini mengaplikasikan aturan heuristik untuk mempercepat proses *query*. Optimasi jenis ini mentransformasikan *query* dengan sejumlah aturan yang akan meningkatkan kinerja eksekusi.

b. *Cost-based*

   Teknik ini mengoptimasikan *cost* yang dipergunakan dari beberapa alternatif untuk kemudian dipilih salah satu yang menjadi *cost* terendah. Teknik ini mengoptimalkan urutan *join* terbalik yang dimungkinkan pada relasi-relasi r1 → r2 → ... rn. Teknik ini dipergunakan untuk mendapatkan pohon *left-deep* join yang akan menghasilkan sebuah relasi sebenarnya pada node sebelah kanan yang bukan hasil dari sebuah *intermediate join*.

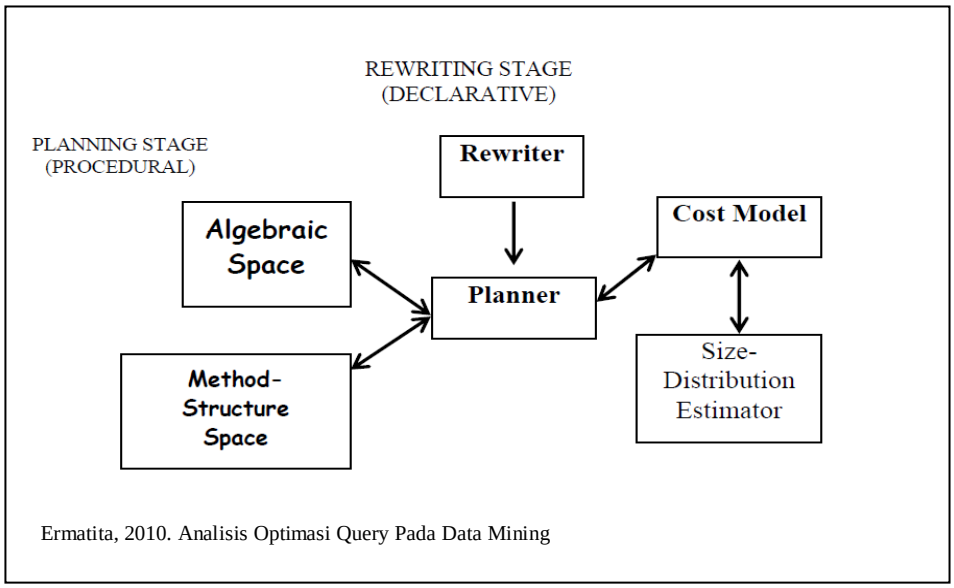
Berdasarkan uraian diatas maka kesimpulan dari optimasi *query* adalah mengoptimasikan *cost* yang dipergunakan dari beberapa alternatif untuk kemudian dipilih salah satu yang menjadi *cost* terendah.

2.1.1.1 *Query Optimizer*

Query optimizer adalah bagian dari DBMS yang berfungsi mengoptimasi query. Proses yang biasanya terjadi dalam *optimizer* adalah *optimizer* memeriksa semua ekspresi-ekspresi aljabar yang sama yang diberikan query dan memilih salah satunya yang memiliki harga taksiran paling rendah. Tugas dari *optimizer* adalah untuk mentransformasikan inisial ekspresi *query* ke dalam sebuah rencana evaluasi yang menghasilkan *record* yang sama. (Ermatita, 2010:3).

Keuntungan dari *optimizer* adalah dapat mengakses semua informasi statistik dari sebuah basis data. Selain itu optimizer juga dapat dengan mudah untuk melakukan optimisasi kembali apabila informasi statistik sebuah basis data berubah dan *optimizer* dapat menangani strategi yang berbeda-beda dalam jumlah besar yang tidak mungkin dilakukan oleh manusia.

Input dari *optimizer* adalah sebuah *tree* yang sudah mengalami proses parsing di dalam *query* parser. *Tree* tersebut biasanya disebut dengan parse tree. Sedangkan *output* dari *optimizer* adalah berupa rencana eksekusi (*execution* plan) yang siap untuk dikirimkan ke dalam *query* kode generator dan *query processor* untuk diproses untuk mendapatkan hasil akhir dari *query* tersebut.



Gambar 1 Arsitektur *Query Optimizer*

Proses optimisasi *query* dapat dianggap mempunyai dua tingkatan. Dua tingkatan tersebut adalah : *rewriting* dan *planning*. Hanya ada satu modul pada tingkat pertama yaitu *Rewriter*, dimana semua modul-modul lainnya berada pada tingkat kedua. Tahap penulisan dapat disebut sebagai level *declarative*, sedangkan tahap perencanaan dapat juga disebut sebagai level *procedural*.

Optimisasi *query* adalah suatu proses untuk menganalisis *query* untuk menentukan sumber-sumber apa saja yang digunakan oleh *query* tersebut dan apakah penggunaan dari sumber tersebut dapat dikurangi tanpa merubah keluaran. Atau dengan kata lain bahwa optimisasi *query* adalah sebuah prosedur untuk meningkatkan strategi evaluasi dari suatu query untuk membuat evaluasi tersebut menjadi lebih efektif (Sudrajat, 2007:16).

Berdasarkan uraian diatas maka kesimpulan dari *query optimizer* adalah bagian dari DBMS yang berfungsi mengoptimasi *query*. Proses yang biasanya terjadi dalam *optimizer* adalah optimizer memeriksa semua ekspresi-ekspresi

aljabar yang sama yang diberikan query dan memilih salah satunya yang memiliki harga taksiran paling rendah.

2.1.2 *SQL*

SQL adalah bahasa standar yang digunakan untuk mengakses data di dalam *database relasional*. Setiap *server database relational* atau *relational database manajemen system* (*RDBMS*) mendukung *SQL* utuk mengatur dan mengelolah datanya. *SQL* lahir pada tahun 1970, yang berawal dari artikel yang berisi tentang ide pembentukan *database relational* oleh seorang peneliti bernama *edgae f. Codd* di perusahaan IBM. Dalam artikel tersebut dibahas juga tentang kemungkinan pembentukan suatu standar untuk mengakses data di dalam *database relational* bersangkutan. Bahasa tersebut kemudian diberia nama *SEQUEL* (*Structured English Query Language*), yang akhirnya diganti nama menjadi *SQL* (*Structured Query Language*). Hal ini disebabkan oleh permasalahan hukum. Nama *SEQUEL* ternyata sudah menjadi trademark dari suatu perusahaan penerbangan bernama UK-based hawker siddeley. (Raharjo:2011:45).

Structured Query Language (*SQL*) adalah bahasa *database relational* yang dikembangkan oleh IBM. Bahasa ini memungkinkan pengguna mengajukan pertanyaan-pertanyaan dalam bahasa yang mendekati bahasa Inggris untuk melihat isi *database* dengan berbagai cara. (Sudarmo, 2006:428)

Berdasarkan dua pengertian di atas penulis menyimpulkan bahwa *Structured Query Language* (*SQL*) merupakan bahasa standar yang digunakan untuk mengakses data di dalam *database relasional*. Setiap *server database*

relational atau *relational database manajemen system* (*RDBMS*) mendukung *SQL* utuk mengatur dan mengelolah datanya.

2.1.3 Basis Data

Basis data atau *database* adalah representasi kumpulan fakta yang saling berhubungan disimpulkan secara bersama sedemikan rupa dan tanpa pengulangan (redudansi) yang tidak perlu, untuk memenuhi berbagai kebutuhan. Data perlu disimpan dalam basis data untuk keperluan penyediaan informasi lebih lanjut. Data di dalam basis data perlu diorganisasikan sedemikan rupa supaya informasi yang dihasilkan berkualitas. Organisasi basis data yang baik juga berguna untuk efisiensi kapasitas penyimpanannya. Dalam maksud yang sama, bisa juga diartikan sebagai sekumpulan informasi yang disusun sedemikan rupa untuk dapat diakses oleh sebuah *software* tertentu. Basis data tersusun atas bagian yang disebut *field* dan *record* yang tersimpan dalam sebuah *file*. (Febrian, 2007:133).

Basis data adalah secara umum berarti koleksi data yang jelas identifikasinya, misalnya daftar nomor-nomor telepon, daftar buku-buku di sebuah perpustakaan, dsb. Secara teoritis, sebuah basis data harus memuat semua informasi dalam satu *file* yang terdiri dari sejumlah catatan. (Sudarmo,2006:108).

Menurut (Teguh:2011:1), tipe *database* yaitu :

1. *Operational database*

   Database ini menyimpan data rinci yang diperlukan untuk mendukung operasi dari seluruh organisasi. Mereka juga disebut subject-area databases (SADB), transaksi *database*, dan produksi *database*. Contoh: *database* pelanggan, database pribadi, *database* inventaris, akuntansi *database*.

2. *Analytical database*

   Database ini menyimpan data dan informasi yang diambil dari operasional yang dipilih dan eksternal *database*. Mereka terdiri dari data dan informasi yang dirangkum paling dibutuhkan oleh sebuah organisasi manajemen dan *End-user* lainnya. Beberapa orang menyebut analitis multidimensi database sebagai database, manajemen database, atau informasi *database*.

3. Data *warehouse*

   Sebuah data *warehouse* menyimpan data dari saat ini dan tahun-tahun sebelumnya - data yang diambil dari berbagai database operasional dari sebuah organisasi. Data *warehouse* menjadi sumber utama data yang telah diperiksa, diedit, standar dan terintegrasi sehingga dapat digunakan oleh para manajer dan pengguna akhir lainnya di seluruh organisasi profesional. Perkembangan terakhir dari data *warehouse* adalah dipergunakan sebagai *Shared nothing architecture* untuk memfasilitasi *ekstrem scaling*.

4. *Distributed database*

   Ini adalah *database* kelompok kerja lokal dan departemen di kantor regional, kantor cabang, pabrik-pabrik dan lokasi kerja lainnya. *Database* ini dapat mencakup kedua segmen yaitu operasional dan user *database*, serta data yang dihasilkan dan digunakan hanya pada pengguna situs sendiri.

5. *End-user database*

   Database ini terdiri dari berbagai *file* data yang dikembangkan oleh *end-user* di *workstation* mereka. Contoh dari ini adalah koleksi dokumen dalam *spreadsheet*, *word processing* dan bahkan *download file*.

6. *External database*

   Database ini menyediakan akses ke eksternal, data milik pribadi *online* - tersedia untuk biaya kepada pengguna akhir dan organisasi dari layanan komersial. Akses ke kekayaan informasi dari database eksternal yang tersedia untuk biaya dari layanan online komersial dan dengan atau tanpa biaya dari banyak sumber di *Internet*.

7. *Hypermedia databases on the web*

   Ini adalah kumpulan dari halaman-halaman multimedia yang saling berhubungan di sebuah situs *web*. Mereka terdiri dari *home page* dan halaman *hyperlink* lain dari multimedia atau campuran media seperti teks, grafik, gambar foto, klip video, audio dll.

8. *Navigational database*

   Dalam navigasi *database*, *queries* menemukan benda terutama dengan mengikuti referensi dari objek lain.

9. *In-memory databases*

   Database di memori terutama bergantung pada memori utama untuk penyimpanan data komputer. Ini berbeda dengan sistem manajemen database yang menggunakan disk berbasis mekanisme penyimpanan. *Database* memori utama lebih cepat daripada dioptimalkan disk database sejak Optimasi algoritma internal menjadi lebih sederhana dan lebih sedikit CPU mengeksekusi instruksi. Mengakses data dalam menyediakan memori lebih cepat dan lebih dapat diprediksi kinerja dari disk. Dalam aplikasi di mana waktu respon sangat penting, seperti peralatan jaringan telekomunikasi yang

mengoperasikan sistem darurat, database memori utama yang sering digunakan.

10. *Document-oriented databases*

    Document-oriented databases merupakan program komputer yang dirancang untuk aplikasi berorientasi dokumen. Sistem ini bisa diimplementasikan sebagai lapisan di atas sebuah database relasional atau objek *database*. Sebagai lawan dari database relasional, dokumen berbasis database tidak menyimpan data dalam tabel dengan ukuran seragam kolom untuk setiap record. Sebaliknya, mereka menyimpan setiap catatan sebagai dokumen yang memiliki karakteristik tertentu. Sejumlah bidang panjang apapun dapat ditambahkan ke dokumen. Bidang yang dapat juga berisi beberapa bagian data.

11. *Real-time databases*

    Real-time Database adalah sistem pengolahan dirancang untuk menangani beban kerja negara yang dapat berubah terus-menerus. Ini berbeda dari database tradisional yang mengandung data yang terus-menerus, sebagian besar tidak terpengaruh oleh waktu. Sebagai contoh, pasar saham berubah dengan cepat dan dinamis. *Real-time processing* berarti bahwa transaksi diproses cukup cepat bagi hasil untuk kembali dan bertindak segera. Real-time database yang berguna untuk akuntansi, perbankan, hukum, catatan medis, multi-media, kontrol proses, sistem reservasi, dan analisis data ilmiah.

12. *Relational Database*

    Standar komputasi bisnis sejak tahun 2009, relational *database* adalah *database* yang paling umum digunakan saat ini. Menggunakan meja untuk informasi struktur sehingga mudah untuk mencari.

Berdasarkan dua pengertian di atas penulis menyimpulkan bahwa basis data merupakan sekumpulan data atau informasi yang teratur berdasarkan kriteria tertentu yang saling berhubungan.

2.1.4 Sistem Manajemen Basis Data

Sistem manajemen basis data adalah perangkat lunak yang didesain untuk membantu dalam hal pemeliharaan dan utilitas kumpulan data dalam jumlah besar. DBMS dapat menjadi alternative penggunaan secara khusus untuk aplikasi. (Herman, 2007:2).

Database Management System (DBMS) adalah suatu perangkat lunak yang ditujukan untuk menangani penciptaan, pemeliharaan, dan pengendalian akses data. Dengan menggunakan perangkat lunak ini pengelolaan data menjadi mudah lakukan. Salain itu perangkat lunak ini juga menyediakan berbagai peranti yang digunakan. (Kadir, 2008:17).

2.1.4.1 Fungsi DBMS

1. Data *Definition*, DBMS harus dapat mengolah pendefinisian data.
2. Data *Menipulation*, DBMS harus dapat menangani permintaan dari pemakai untuk mengakses data.
3. Data *Security* dan *Integrity*, DBMS harus dapat memeriksa security dan integrity data yang didefinisikan oleh DBA.

4. Data *Recovery* dan *Concurency*, DBMS harus dapat menangani kegagalan-kegagalan pengaksesan basis data yang dapat disebabkan oleh kesalahan sistem, kerusakan disk.
5. Data *Dictionary*, DBMS harus menyediakan data dictionary.
6. *Performance*, DBMS harus menangani unjuk kerja dari semua fungsi seefisien mungkin.

Secara umum, terdapat 2 jenis bahasa basis data, yaitu: DDL (data *definition language*) dan DML (data *manipulation language*). DDL merupakan perintah-perintah yang biasa digunakan administrator basis data untuk mendefinisikan skema dan subskema basis data (Contoh: *CREATE, ALTER, MODIFY, DROP*). Sedangkan, DML merupakan merupakan perintah-perintah yang memungkinkan pengguna melakukan akses dan manipulasi data sebagaimana yang telah diorganisasikan sebelumnya dalam model data yang tepat (Contoh: *SELECT, INSERT, UPDATE, DELETE*) .

Berdasarkan dua pengertian di atas penulis menyimpulkan bahwa *Structured Query Language* (SQL) merupakan bahasa yang banyak digunakan dalam berbagai produk basis data.

2.1.5 *MySQL*

Mysql adalah *software* sistem manajemen *database*. *Database* adalah suatu koleksi data yang terstruktur. *Database* ini bisa berupa daftar belanja sederhana sampai informasi yang sangat besar dari suatu perusahaan internasional. Untuk menambahkan, mengakses dan memproses data disimpan di komputer. Rickyanto (2002:32).

Sedangkat menurut Simarmata (2006:29), *MYSQL* adalah suatu *database* populer dengan pengembangan *web developers*. Kecepatan dan ukuran yang kecil membuatnya ideal untuk *website*. Ditambah lagi dengan fakta bahwa *MySQL* adalah *open source* yang artinya gratis.

2.1.6 *Cross-Product Query*

Cross product digunakan saat mengkombinasikan data pada dua tabel atau lebih. (Santriputri, 2010:2).

Model *cross product* diwakili oleh *query* berikut ini:

```
select [nama_kolom1],..., [nama_kolomN]
from [nama_tabel1], [nama_tabel2]
where [nama_tabel1].[nama_kolom1] = [nama_tabel2].[nama_kolom2]
```

Query tersebut melakukan select sejumlah kolum dari 2 tabel atau lebih dimana pengkondisian masing-masing tabel di *join*.

2.1.7 *UML* (*Unified Modeling Language*)

UML (Unified Modeling Language) adalah pendekatan terstruktur memiliki *tool-tool* perancangan yang di kenal secara luas serta menjadi standar umum, seperti *DFD (Data Flow Diagram)*, *ERD (Entity Relationship Diagram)*, bagan terstruktur *(structure chart)*, diagram alir *flow chart*, (Nugroho,2005:16).

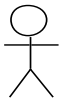
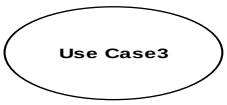
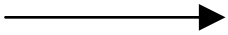
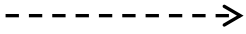
Unified Modeling Language (UML) merupakan alat merancang perangkat lunak, sarana komunikasi antara perangkat lunak dengan proses bisnis, menjabarkan sistem secara rinci untuk analisa dan mencari apa yang diperlukan

sistem, mendokumentasikan sistem yang ada, proses-proses dan organisasinya. (Herlawati,2011:6).

1. *Use Case Diagram*

Use case diagram adalah deskripsi fungsi dari sebuah sistem dari perspektif pengguna. *Use case* bekerja dengan cara mendeskripsikan tipikal interaksi antara *user* (pengguna) sebuah sistem dengan sistemnya sendiri melalui sebuah cerita bagaimana sebuah sistem dipakai.

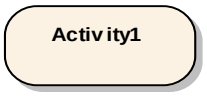
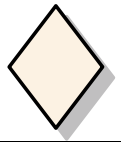
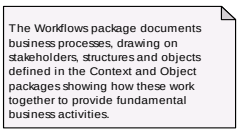
Table* 1 Simbol-simbol *Use Case Diagram

No	Simbol	Keterangan Fungsi
1	*Aktor* 	Aktor adalah sebuah entitas manusia atau mesin yang berinteraksi dengan system untuk melakukan pekerjaan-pekerjaan tertentu.
2	*Use Case* 	*Use Case* adalah deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu aktor.
3	*Asosiasi*	Asosiasi adalah apa yang menghubungkan antara objek satu dengan objek yang lainnya.
4	*Generalisasi* 	Generalisasi adalah hubungan dimana objek anak (descendent) berbagi prilaku dan struktur data dari objek yang ada diatasnya atau sebaliknya dari bawah ke atas.
5	*Dependency* 	*Dependency* (ketergantungan) adalah hubungan dimana perubahan yang terjadi pada suatu elemen Dependen (mandiri) akan mempengaruhi elemen yang bergantung padanya (Independen).

(Sumber: Nugroho, 2005:16), *Rational Rose* Untuk Permodelan Berorientasi Objek.

2. *Activity Diagram*

Table* 2 Simbol-simbol *Activity Diagram

No	Simbol	Keterangan Fungsi
1	*Start* 	Mendefinisikan suatu tindakan sebelum aktivitas dimasukkan.
2	*Activity* 	Aktivitas menggambarkan proses yang berjalan, sementara *use case* menggambarkan bagaimana aktor menggunakan sistem untuk melakukan aktivitas.
3	*Control Flow*	Mendeskripsikan kemana aliran kegiatan berlangsung.
4	*Fork/Join* 	Untuk mengilustrasikan proses-proses paralel (*fork* dan *join*) digunakan titik sinkronisasi yang dapat berupa titik, garis horizontal atau vertikal.
5	*Decision* 	Untuk menggambarkan *behaviour* pada kondisi tertentu.
6	*Annotation Things* 	*Annotation Things* merupakan bagian yang memperjelas model UML. Ia dapat berupa komentar-komentar yang menjelaskan fungsi serta ciri-ciri tiap elemen dalam model UML.
7	*Final* 	Menandakan bahwa suatu tindakan atau aktivitas telah selesai

(Sumber: Nugroho, 2005:17), *Rational Rose* Untuk Permodelan Berorientasi Objek.

2.2 Penelitian Sebelumnya

2.2.1 Optimasi *SQL query* untuk information retrieval pada aplikasi berbasis *web*.

Teroptimasi dapat mempercepat proses information retrieval pada aplikasi berbasis *web* secara signifikan dengan karakteristik yang sama meskipun menggunakan RDBMS yang berbeda-beda. Penggunaan *query* yang teroptimasi juga dapat menjaga keberlangsungan proses information retrieval agar dapat tertampil ke dalam halaman *web*, karena dapat meminimalisir terjadinya *script time out* yang dilakukan oleh *web* server yang akan mengakibatkan proses penampilan informasi berhenti. (Setiawan, 2004).

2.2.2 Perancangan dan pembuatan aplikasi pengoptimal *SQL query*

Aplikasi Pengoptimal *SQL query* menghasilkan *SQL query* dengan waktu eksekusi sama dengan atau lebih cepat dari input *SQL query*-nya. Percepatan waktu eksekusi *SQL query* yang dihasilkan Aplikasi Pengoptimal *SQL query* bergantung salah satunya pada jumlah kolom pada *SQL query* yang dapat dioptimasi. Percepatan waktu eksekusi *SQL query* yang dihasilkan oleh Aplikasi Pengoptimal *SQL query* semakin besar secara linier pada jumlah data yang lebih besar pada basis data *MySQL*. (Pornawan, 2007).

III. METODOLOGI PENELITIAN

3.1 Waktu dan Tempat Penelitian

Waktu penelitian yang penulis lakukan pada SMK Negeri 2 Prabumulih mulai dari Maret 2013 sampai dengan Agustus 2013.

3.2 Metode Pengumpulan Data

Metode pengumpulan data yang digunakan untuk mendapatkan data dan informasi, maka metode yang digunakan dalam proses pengumpulan data sebagai berikut :

1. Metode Observasi

   Dalam hal ini yang akan dilakukan adalah melihat serta mempelajari permasalahan yang ada dilapangan yang erat kaitannya dengan objek yang diteliti.

2. Metode Studi Pustaka

   Metode yang dilakukan adalah dengan cara mancari bahan yang mendukung dalam pendefinisisan masalah melalui buku-buku, *internet*.

3.3 Metode Pengembangan Perangkat Lunak

Metode pengembangan perangkat lunak yang digunakan dalam penelitian adalah *waterfall*. Menurut Pressman (2002:36) *waterfall* adalah model klasik yang sederhana dengan aliran sistem yang linier. *Output* dari setiap tahap merupakan *input* bagi tahap berikutnya. Tahapan dari metode *waterfall* sebagai berikut :

1. **Rekayasa dan Pemodelan Perangkat Lunak**

   Pada aktivitas ini, pekerjaan dimulai dengan membangun syarat dari semua elemen sistem dan mengalokasian beberapa subset dari kebutuhan ke perangkat lunak.

2. **Analisis Kebutuhan Perangkat Lunak**

   Untuk memahami sifat program yang dibangun, analisis harus memahami domain informasi, tingkah laku, unjuk kerja, dan antaramuk yang dibutuhkan.

3. **Rancangan Perangkat Lunak**

   Proses rancangan menerjemahkan syarat/kebutuhan kedalam sebuah representasi perangkat lunak yang dapat diperkirakan demi kualitas sebelum dimulai pemunculan kode. Sebagaimana persyaratan, rancangan/desain didokumentasikan dan menjadi bagian konfigurasi perangkat lunak.

4 Pengkodean Perangkat Lunak

Dalam pembuatan perangkat lunak peneliti menggunakan *scripting php* yang cendrung mudah dipelajari dan mempunyai fasilitas yang mendukung dalam menghubungkan dengan sistem *windows*.

5 Pengujian Perangkat Lunak

Proses pengujian berfokus pada logika internal perangkat lunak, memastikan bahwa semua pernyataan sudah diuji, dan pada eksternal fungsional, yaitu mengarangkan pengujian untuk menemukan kesalahan - kesalahan.

1.5.5 Teknik Optimasi

Teknik optimasi yang digunakan adalah *cost based*. Teknik ini mengoptimasikan *cost* yang dipergunakan dari beberapa alternatif untuk

kemudian dipilih salah satu yang menjadi *cost* terendah. Teknik ini mengoptimalkan urutan *join* terbalik yang dimungkinkan pada relasi-relasi r1 → r2 → rn. Teknik ini dipergunakan untuk mendapatkan pohon *left-deep join* yang akan menghasilkan sebuah relasi sebenarnya pada node sebelah kanan yang bukan hasil dari sebuah *intermediate join*. (Setiawan, 2001:2).

Sedangkan algoritma yang digunakan adalah *cross product* atau disebut juga *cross join* digunakan saat mengkombinasikan data pada dua tabel atau lebih (Santputri, 2010:2).

Model *cross product* diwakili oleh query berikut ini:

select [nama_kolom1],..., [nama_kolomN]

from [nama_tabel1], [nama_tabel2]

where [nama_tabel1].[nama_kolom1] = [nama_tabel2].[nama_kolom2]

Query tersebut melakukan *select* sejumlah kolum dari 2 tabel atau lebih dimana pengkondisian masing-masing tabel di *join*.