UNIVERSITAS INDONESIA

PERBANDINGAN KECEPATAN SISTEM PENCARIAN KATA PADA DATABASE SIMPLE-O

SKRIPSI

RIZKA HAIFA

0706276085

FAKULTAS TEKNIK UNIVERSITAS INDONESIA

PROGRAM STUDI TEKNIK KOMPUTER

DEPOK

JUNI 2011

UNIVERSITAS INDONESIA

PERBANDINGAN KECEPATAN SISTEM PENCARIAN KATA PADA DATABASE SIMPLE-O

SKRIPSI

Diajukan sebagai salah satu syarat untuk memperoleh gelar Sarjana Teknik

RIZKA HAIFA

0706276085

FAKULTAS TEKNIK UNIVERSITAS INDONESIA

PROGRAM STUDI TEKNIK KOMPUTER

DEPOK

JUNI 2011

HALAMAN PERNYATAAN ORISINALITAS

Skripsi ini adalah hasil karya saya sendiri,
dan semua sumber baik yang dikutip maupun dirujuk
telah saya nyatakan dengan benar.

Nama	: Rizka Haifa
NPM	: 0706276085
Tanda Tangan	:
Tanggal	: 10 Juni 2011

HALAMAN PENGESAHAN

Skripsi ini diajukan oleh	:	
Nama	:	Rizka Haifa
NPM	:	0706276085
Program Studi	:	Teknik Komputer
Judul Skripsi	:	Perbandingan Kecepatan Sistem Pencarian Kata Pada Database SIMPLE-O

Telah berhasil dipertahankan dihadapan Dewan Penguji dan diterima sebagai bagian persyaratan yang diperlukan untuk memperoleh gelar Sarjanah Teknik pada Program Studi Teknik Komputer, Fakultas Teknik, Universitas Indonesia.

DEWAN PENGUJI

Pembimbing : Dr. Ir. Anak Agung Putri Ratna M.Eng (

Penguji : Ir. Endang Sriningsih MT, Si (

Penguji : Prima Dewi Purnamasari ST., MT., MSc. (

Ditetapkan di : Depok

Tanggal : 10 Juni 2010

UCAPAN TERIMA KASIH

Puji syukur saya panjatkan kehadirat Allah SWT, karena atas segala rahmat dan hidayat-Nya saya dapat menyelesaikan skripsi ini. Saya menyadari bahwa skripsi ini tidak akan terselesaikan tanpa bantuan dari berbagai pihak. Oleh karena itu, saya mengucapkan terima kasih kepada :

1. Ibu Dr. Ir. Anak Agung Putri Ratna, M.Eng, selaku pembimbing skripsi saya, terima kasih atas arahan serta koreksi skripsi saya ini dengan sabar, dan telah memberikan banyak waktu untuk mengarahkan saya hingga selesai.
2. Para peneliti sebelum ini yang juga memberikan sumber bacaan yang banyak bagi saya.
3. Senior saya Boma yang telah mengajarkan dan memberikan banyak masukan dan juga Meirisal yang telah membantu menjawab pertanyaan-pertanyaan saya.
4. Orang tua dan keluarga saya yang telah memberikan bantuan dukungan material dan moral kepada saya.
5. Sahabat, teman satu kelompok (Vanessa, Tania, Junda)serta semua yang telah memberi semangat saya dalam menyelesaikan skripsi ini tepat waktu.
6. Dan seluruh Sivitas akademik Departemen Teknik Elektro yang tidak dapat saya sebutkan satu persatu.

Akhir kata, semoga Allah SWT berkenan membalas kebaikan semua pihak yang telah membantu. Semoga skripsi ini bermanfaat bagi perkembangan ilmu pengetahuan.

Depok, 10 Juni 2011

Rizka Haifa

HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI TUGAS AKHIR UNTUK KEPENTINGAN AKADEMIS

Sebagai sivitas akademika Universitas Indonesia, saya bertanda tangan di bawah ini :

Nama : Rizka Haifa

NPM : 0706276085

Program studi : Teknik Komputer

Departemen : Teknik Elektro

Fakultas : Teknik

Jenis karya : Skripsi

demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada Universitas Indonesia **Hak Bebas Royalti Nonoksklusif (*Non-exclusive Royalty Free Right*)** atas karya ilmiah saya yang berjudul :

PERBANDINGAN KECEPATAN SISTEM PENCARIAN KATA PADA DATABASE SIMPLE-O

beserta perangkat yang ada (jika diperlukan). Dengan Hak Bebas Royalti Non Eksklusif ini Universitas Indonesia berhak menyimpan, mengalihmedia/format-kan, mengelola dalam bentuk pangkalan data (*database*), merawat, dan mempublikasikan tugas akhir saya selama tetap mencantumkan nama saya sebagai penulis/pencipta sebagai pemegang Hak Cipta.

Demikian pernyataan ini saya buat dengan sebenarnya.

Dibuat di : Depok

Pada tanggal : 10 Juni 2010

Yang menyatakan

Rizka Haifa

ABSTRAK

Nama : Rizka Haifa
Program studi : Teknik Komputer
Judul : Perbandingan Kecepatan Pencarian Kata Pada Database SIMPLE-O

Skripsi ini membahas perbandingan kecepatan sistem pencarian kata pada database SIMPLE-O. Terdapat empat sistem yang dibandingkan yaitu 2 sistem yang merupakan sistem yang telah diimplementasikan (SIMPLE V1 dan SIMPLE V2) dan 2 sistem lagi merupakan sistem yang sedang dikembangkan (SIMPLE V3 dan SIMPLE V4). SIMPLE V1 adalah sistem pencarian kata yang semua proses pencariannya terletak di PHP. SIMPLE V2 meletakkan sebagian proses pencarian kata dalam database MySQL dan membagi tabel database menjadi 5 bagian berdasarkan kata depan sehingga proses terbagi dua yaitu pemotongan kata depan yang terletak di PHP dan pencarian kata yang terletak di MySQL. SIMPLE V3 merupakan pencarian kata yang menyerahkan proses pencarian sepenuhnya pada MySQL. Sedangkan SIMPLE V4 menggunakan algoritma *Rabin-Karp* yang diletakkan di *stored procedure* MySQL. Dari keempat sistem yang dibandingkan, sistem pencarian kata SIMPLE V3 merupakan sistem yang paling cepat diantara sistem lainnya. Kecepatan SIMPLE V3 62 kali lebih cepat dari SIMPLE V1 pada pencarian 1 kata dan 106 kali lebih cepat pada pencarian banyak kata. Sedangkan kecepatan SIMPLE V4 pada pencarian 1 kata tidak stabil sehingga tidak dapat dibandingkan dengan sistem lainnya dan pada pencarian banyak kata kecepatan SIMPLE V4 hanya 1.07 kali lebih cepat dari pada SIMPLE V1. Pada pencarían kata dalam kondisi jaringan sibuk (10-30 user mengakses sistem secara bersama-sama), kecepatan SIMPLE V3 hanya turun sedikit sebesar 0.0000193947 detik yaitu sekitar 1.17 kali lebih lambat dibandingkan ketika jaringan tidak sibuk dan hanya diakses oleh 1 *user*. Sedangkan kecepatan SIMPLE V4 mengalami penurunan 0.041978478 detik yaitu sekitar 6.55 kali lebih lambat. Dari situ dapat terlihat bahwa SIMPLE V3 merupakan sistem yang paling cepat dan stabil.

Kata kunci : Pencarian Kata, Simple-O, *Rabin-Karp*, PHP, MySQL

ABSTRACT

Name : Rizka Haifa
Study Program: Computer Engineering
Title : Comparison The Speed Of Word-Matching Systems On SIMPLE-O Database

This thesis focuses on comparison the speed of word-matching systems on SIMPLE-O database. There are four systems that compare, the 2 systems which is a system that has been implemented (SIMPLE SIMPLE V1 and V2) and 2 system is a system that is being developed (SIMPLE SIMPLE V3 and V4). SIMPLE V1 is the system searches that all of the search process lies in PHP. SIMPLE V2 which put some of the search process in a MySQL database and database tables divide into 5 sections based on the preposition that the process of cutting is two prepositions which lies in the PHP and the search for a word that is located in MySQL. SIMPLE V3 is a word search that submitted the search process entirely on MySQL. While SIMPLE V4 using Rabin-Karp algorithm, which is placed on MySQL stored procedures. With Compared the four systems, a word search system SIMPLE V3 is the fastest system among other systems. V3 SIMPLE speed 62 times faster than SIMPLE V1 on a word search. While on the search for many words, the speed SIMPLE V3 106 times faster than SIMPLE V1. The speed SIMPLE V4 on a word search is unstable so can not be compared with other systems and on many search words SIMPLE speed V4 only 1.07 times faster than SIMPLE V1. In the search word in the busy network conditions (10-30 users accessing the system together), SIMPLE speed V3 is only down slightly by 0.0000193947 seconds which is about 1:17 slower than the times when the network is not busy and only accessible by a user. While speed has decreased SIMPLE V4 0.041978478 seconds which is about 6:55 slower times. From there it can be seen that the SIMPLE V3 system is the most rapid and stable.

Key Word : Word-Matching,, Simple-O, *Rabin-Karp*, PHP, MySQL

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB 1
PENDAHULUAN

1.1. Latar Belakang

Ujian merupakan suatu cara untuk mengukur sejauh mana kemampuan siswa dalam memahami pelajaran yang mereka pelajari. Ada beberapa macam bentuk ujian dan salah satunya adalah ujian *essay*. Diantara beberapa bentuk ujian, hanya ujian *essay* yang dapat secara efektif mengukur kemampuan siswa karena ujian essay menilai keseluruhan tingkatan dari *Bloom's Taxonomy* (*Knowledge, Comprehension, Application, Analysis, Synthesis and Evaluation*) [1]. Namun, terdapat satu kendala besar dalam ujian *essay* yaitu lamanya waktu yang dibutuhkan oleh pengajar untuk mengoreksi terlebih lagi jika soal yang diberikan dan siswa yang mengikuti ujian banyak.

Seiring dengan perkembangan teknologi, banyak hal yang diciptakan untuk memudahkan tugas manusia dan salah satunya adalah ujian *essay* dalam bentuk *online* atau disebut dengan *online essay grading*. Online *essay grading* dilakukan secara *online*, mulai dari menjawab soal ujian, hingga pemberian penilaian sehingga hal ini memberikan keuntungan bagi pengajar maupun siswa karena pelaksanaan ujian menjadi lebih efektif dan efisien [2]. Adapun keuntungan yang didapat siswa adalah siswa langsung dapat melihat nilai yang dicapai dan penilaian bersifat objektif karena sudah ada jawaban yang menjadi kata kunci dalam penilaian sehingga nilai siswa tergantung pada berapa banyak jawaban yang sama dengan kata kunci. Sedangkan kemudahan yang didapat pengajar adalah mereka tidak perlu lagi memeriksa begitu banyak *essay* yang dikerjakan karena semuanya sudah dikerjakan oleh komputer dan juga waktu yang dibutuhkan untuk memeriksa relatif singkat. Pada dasarnya ujian *online* hampir tidak ada bedanya dengan ujian biasa, perbedaannya hanya pada ujian *online* harus menggunakan komputer dan nilai ujian langsung dapat dilihat oleh siswa.

Saat ini telah dikembangkan juga suatu ujian *online* dalam bentuk *essay* yang dinamakan SIMPLE-O yang merupakan sistem penilaian esei otomatis dengan menggunakan metode *Latent Semantic Analysis* (LSA) berbasis bahasa

Indonesia dan menambah pembobotan pada kata-kata yang dianggap penting dari kata kunci yang dipilih [3]. SIMPLE-O merupakan ujian esei *online* berbasis *web* yang dibangun dengan bahasa *scripting* PHP dan HTML serta dengan menggunakan database MySQL. Pada SIMPLE-O, database MySQL digunakan untuk menyimpan data-data yang digunakan yaitu data soal, data jawaban , data nilai dan kata-kata yang akan digunakan untuk membentuk matriks persamaan antara jawaban siswa atau mahasiswa dengan kunci jawaban sebenarnya [4].

Permasalahan yang muncul dalam SIMPLE-O adalah begitu banyaknya data yang tersimpan pada database yang menyebabkan proses pencarian kata menjadi lambat sehingga hal tersebut dapat mengganggu atau memperlambat kinerja perhitungan sistem. Masalah lain yang muncul adalah proses pencarian kata yang terletak di PHP dirasa penulis kurang efektif karena sistem akan bekerja tiga kali dengan mengambil data yang ada di MySQL, melakukan pencarian di PHP dan mengembalikan data lagi ke MySQL.

Permasalahan tersebut menuntut dikembangkannya sistem pencarian kata yang lebih cepat. Sampai saat ini, telah ada dua sistem yang digunakan dalam pencarian kata. Sistem 1 atau disebut SIMPLE V1 adalah sistem pencarian kata yang semua proses pencariannya terletak di PHP. SIMPLE V1 ini terbukti tidak efektif karena proses pencarian kata masih terlalu lama. Oleh karena itu, dikembangkan sistem 2 atau disebut SIMPLE V2 yang meletakkan sebagian proses pencarian kata dalam database MySQL dan membagi tabel database menjadi 5 bagian berdasarkan kata depan. SIMPLE V2 terbukti lebih efektif karena dapat menaikkan kecepatan pencarian kata hingga 7.21 kali lebih cepat dengan peningkatan waktu mencapai 4,185549 detik dibandingkan algoritma 1. Namun, seiring akan bertambahnya variasi kata dalam database menyebabkan masalah kecepatan pencarian kata akan menjadi kendala lagi dikemudian hari. Oleh karena itu, penulis mencoba membuat dua sistem pencarian kata yang dirasa dapat meningkatkan proses pencarian kata agar dikemudian hari tidak terkendala oleh meningkatnya jumlah kata dalam database. Untuk membedakan dengan sistem yang sebelumnya telah dibuat, kedua sistem yang akan dibuat dinamakan SIMPLE V3 dan SIMPLE V4. SIMPLE V3 merupakan pencarian kata yang menyerahkan proses pencarian sepenuhnya pada MySQL. Sedangkan SIMPLE

V4 menggunakan algoritma *Rabin-Karp* yang diletakkan di *stored procedure* MySQL. Kedua sistem ini kemudian akan dibandingkan dengan kedua sistem yang telah ada sebelumnya untuk membuktikan apakah algoritma yang dibuat penulis terbukti lebih cepat atau bahkan lebih lambat.

Pada skripsi ini penulis akan membandingkan keempat sistem pencarian kata untuk mencari sistem yang paling cepat dan efektif untuk digunakan dalam sistem SIMPLE-O.

1.2. Perumusan Masalah

Pada skripsi ini, penulis tidak akan menyimpang dari judul yang ditetapkan serta sesuai dengan maksud yang disampaikan. Berikut ini adalah identifikasi masalah dalam skripsi ini :

1. Membuat algoritma yang mempercepat proses pencarian kata dan implementasinya pada sistem SIMPLE-O
2. Membandingkan kecepatan pencarian kata algoritma yang baru dibuat dengan yang telah ada

1.3. Tujuan Penelitian

Tujuan dari skripsi ini adalah

1. Mempercepat pencarian kata dengan menggunakan algoritma *Rabin-Karp* dan fungsi "*SELECT*" pada MySQL.
2. Membandingkan kecepatan pencarian kata antara 2 algoritma baru dan 2 algoritma lama.
3. Menentukan algoritma pencarian kata yang paling cepat untuk digunakan oleh SIMPLE-O

1.4. Batasan Masalah

Pada skripsi ini, penulis menetapkan batasan masalah agar tidak terjadi penyimpangan dalam pembuatan skripsi ini, yaitu

1. Menerapkan fungsi "*SELECT*" dalam pencarian kata
2. Membuat algoritma *Rabin-Karp* untuk pencarian kata

1.5. Metodologi Penelitian

Metode penelitian yang digunakan dalam penulisan skripsi ini antara lain :

a. Studi literatur

Mengumpulkan data pencocokan *string* dan *stored procedure* dari jurnal, *paper* dan bacaan-bacaan yang ada kaitannya dengan judul penelitian.

b. Wawancara

Pengumpulan data melalui tatap muka dan tanya jawab langsung dengan sumber data atau pihak-pihak yang berhubungan dengan penelitian.

c. Perancangan Sistem

Perancangan sistem dibagi menjadi dua yaitu perancangan algoritma pencarian kata dan perancangan uji coba pencarian kata.

d. Implementasi

Menaruh rancangan sistem pencarian kata dalam bentuk store procedure pada SIMPLE-O dan menyesuaikannya dengan program

1.6. Sistematika Penulisan

Penulisan penelitian ini terdiri dari 4 bab, yaitu :

a. Bab 1 Pendahuluan

Menjelaskan tentang latar belakang masalah, tujuan penelitian, identifikasi masalah, batasan masalah, metodologi penelitian dan sistematika penulisan.

b. Bab 2 *Essay Grading*, SIMPLE-O, Database dan Algoritma Pencarian Kata

Menjelaskan tentang landasan atau dasar teori yang digunakan dalam penulisan penelitian ini yang berhubungan dengan permasalah yang diambil penulis. Teori-teori tersebut diambil dari literatur, observasi serta wawancara dengan pihak yang terkait.

c. Bab 3 Perancangan Sistem Pencarian Kata

Menjabarkan rancangan sistem dalam penelitian ini.

d. Bab 4 Implementasi, Ujicoba dan Analisa

Menempatkan, menjalankan dan menguji algoritma pencarian kata serta menganalisa perbandingan keempat algoritma untuk ditentukan mana yang paling cepat

e. Bab 5 Kesimpulan dan Saran

Kesimpulan yang didapatkan dari penelitian ini.

BAB 2
ESSAY GRADING, SIMPLE-O, DATABASE DAN ALGORITMA PENCARIAN KATA

2.1. Sistem Essay Grading

Penelitian mengenai *essay grading* menggunakan komputer telah ada sejak sebuah artikel mengenai topik itu muncul dalam Kappan hampir 30 tahun yang lalu [5]. Pada saat itu semua orang terkejut ketika mengetahui bahwa komputer dapat menilai sebaik penilaian yang diberikan manusia[6]. Dilanjutkannya pengembangan penelitian ini dikarenakan *essay* dianggap banyak peneliti sebagai ujian yang paling berguna untuk menilai hasil belajar, menyatakan kemampuan mengingat, mengatur dan menggabungkan ide serta kemampuan untuk mengekspresikan diri dalam bentuk tulisan memiliki kendala. Kendala yang dihadapi oleh penilaian essay adalah subjektifitas dalam proses penilaian yang dirasa oleh siswa sebagai bentuk ketidakadilan karena untuk soal yang sama, nilai yang diberikan antara penilai yang satu dengan penilai yang lain dapat berbeda. Selain itu, kegiatan *essay grading* dinilai memakan waktu. Sekitar 30% waktu pengajar di Inggris digunakan untuk memeriksa *essay* [7] sehingga dengan adanya *essay grading* otomatis diharapkan penilaian *essay* akan lebih konsisten dan dapat menghemat waktu dan biaya.

Essay grading otomatis saat ini tersedia baik sebagai sistem komersil ataupun sebagai hasil penelitian dalam bidang ini, seperti *Project Essay Grade (PEG), Intelligent Essay Assessor (IEA), Educational Testing service I, Electronic Essay Rater (E-Rater), C-Rater, BETSY, Intelligent Essay Marking System* dan masih banyak lagi. Masing-masing sistem memiliki metode yang berbeda-beda dalam menilai *essay* yang memiliki kekurangan dan kelebihan masing-masing. Namun, tujuan dari penggunaan metode tersebut tetaplah sama yaitu agar didapat penilaian yang mendekati penilaian yang diberikan oleh manusia.

Penerapan penilaian *Essay Grading* yang disebutkan diatas merupakan sistem yang dibuat dengan basis bahasa Inggris. Bahasa basis yang digunakan disini sangat mempengaruhih proses dan hasil yang dilakukan pada proses penilaian terhadap jawaban siswa karena karakteristik dari bahasa yang berbeda

[2]. Untuk menunjang penilaian esai di Indonesia, dikembangkanlah suatu sistem penilaian otomatis berbasis bahasa Indonesia yang dinamakan SIMPLE-O. Perkembangan SIMPLE-O terjadi melalui beberapa tahapan yang masing-masing tahapan mengembangkan bagian masing-masing dengan tujuan untuk meningkatkan kinerja dan keakuratan penilaian.

2.2. SIMPLE-O

SIMPLE-O yang merupakan kependekan dari sistem penilaian *essay* otomatis adalah salah satu essay grading yang sedang dikembangkan di Indonesia. Pada proses penilaian, SIMPLE-O menggunakan suatu metode *Latent Schematic Analysis* (LSA) dan teknik analisa *Singular Value Decomposition* (SVD). Penilaian esai otomatis ini juga menambahkan fitur pembobotan sebagai teknik penilaian yaitu menambahkan nilai bobot pada kata-kata yang dianggap penting dengan mengalikan nilai tersebut dengan bobot dua. Hasil yang diperoleh mencapai 82.56-96.42% *aggreement* dengan *human rater* karena menggunakan satu atau dua level pembobotan [2]. Sistem berbasis web yang dibangun dalam bahasa Indonesia ini terdiri dari 3 modul :

1. Modul Login
2. Modul Dosen
3. Modul Mahasiswa

Setiap modul memiliki peranan masing-masing dalam sistem yang menunjang terciptanya sistem penilaian.

2.3. *Latent Semantic Analysis (*LSA) Dengan Analisis *Singular Value Decomposition* (SVD)

Latent Semantic Analysis (LSA) merupakan salah satu metode untuk mengekstrak dan merepresentasikan kalimat dengan perhitungan matematis atau statistik dari teks dengan jumlah besar [8]. Perhitungan matematis dilakukan dengan memetakan ada atau tidak adanya kata dari kelompok kata pada matriks. Pada pemrosesan LSA, terdapat analisa SVD (*Singular Value Decomposition*) yang mengkompresi informasi yang berkaitan dalam jumlah besar ke dalam ruang yang lebih kecil tetapi mewakili arti yang sebenarnya.

Teknik SVD digunakan untuk melakukan estimasi atau perkiraan struktur dalam penggunaan kata dalam dokumen-dokumen. Pada dasarnya SVD merupakan teknik untuk melakukan estimasi *rank* dari matriks. Jika diketahui matriks A dengan dimensi m x n, dimana nilai $m \geq n$ dan rank (A) = r maka *singular value decomposition* dari A, dinotasikan sebagai SVD (A), didefinisikan melalui persamaan

$$A=U\Sigma V^T \quad \text{……………………………………………………………}(2.1)$$

dimana

$$U^TU=V^TV=I_n \quad \text{...……………………………………………………}(2.2)$$

dan memenuhi kondisi

$$\Sigma=\text{diag}(\sigma_1,\ldots,\sigma_2) \quad \text{…………………………………………………}(2.3)$$

dimana

$\sigma_i > 0$ untuk $1 \leq i \leq r$

$\sigma_j > 0$ untuk $j \leq r + 1$

Kolom r pertama dari matriks U dan V mendefinisikan vektor *eigen orthonormal* yang bersesuaian dengan r nilai vektor *eigen* tidak-nol dari matriks AA^T dan A^TA berturut-turut. Kolom dari matriks U dan V berisi vektor, masing-masing disebut vektor singular kiri dan kanan. Nilai singular dari A merupakan elemen diagonal dari matriks Σ, dimana nilai singular didapat dari akar pangkat dua dari nilai absolut dari sejumlah n nilai *eigen* dari AA^T [4].

2.4. Database Management Systems (DBMS)

DBMS (*Database Management systems*) adalah kumpulan program yang mengkoordinasikan semua kegiatan yang berhubungan dengan database. Dengan adanya berbagai tingkatan pandangan dalam suatu database maka untuk mengakomodasikan masing-masing pengguna dalam piranti lunak manajemen *database* biasanya terdapat bahasa-bahasa tertentu yang disebut *Data Sub language* [9].

Data sub language adalah subset bahasa yang dipakai untuk operasi manajemen *database*. Pada penggunaan biasanya dapat ditempelkan (*embedded*)

pada bahasa tuan rumah (Cobol, PL/1, dsb). Secara umum maka setiap pengguna *database* memerlukan bahasa yang dipakai sesuai tugas dan fungsinya.

Dalam database secara umum dikenal dua data sub language :

1. *Data Definition Language (DDL)*

   Bahasa yang digunakan dalam mendefinisikan struktur atau kerangka dari *database*, di dalamnya termasuk *record*, elemen data, kunci elemen, dan relasinya

2. *Data Manipulation Language (DML)*

   Bahasa yang digunakan untuk menjabarkan pemrosesan dari *database*, fasilitas ini diperlukan untuk memasukkan, mengambil, mengubah data. DML dipakai untuk operasi terhadap isi *database*

   Ada dua jenis DML :

   1. *Procedural DML*

      Digunakan untuk mendefinisikan data yang diolah dan perintah yang akan dilaksanakan.

   2. *Non Procedural*

      Digunakan untuk menjabarkan data yang diinginkan tanpa menyebutkan bagaimana cara pengambilannya.

Secara khusus pengguna menggunakan berbagai bahasa :

Programmer aplikasi menggunakan bahasa-bahasa seperti Cobol, Informix, dll (*host language*) yang ditempelkan dengan bahasa yang dipakai dalam DBMS. Pemakai terminal menggunakan bahasa Query (misal SQL) atau menggunakan program aplikasi (yang dirancang oleh programmer). Sedangkan pengelola database atau DBA (*Database Administrator*) lebih banyak menggunakan bahasa DDL dan DML yang tersedia dalam DBMS [9].

2.4.1. MySQL

Merupakan software sistem manajemen basis data SQL (bahasa Inggris: database management system) atau DBMS yang *multithread* dan *multi-user*. MySQL AB membuat MySQL tersedia sebagai *software* gratis dibawah lisensi GNU *General Public License* (GPL), tetapi mereka juga menjual dibawah lisensi

komersial untuk kasus-kasus dimana penggunaannya tidak cocok dengan penggunaan GPL. MySQL dimiliki dan disponsori oleh sebuah perusahaan komersial Swedia MySQL AB, dimana memegang hak cipta hampir atas semua kode sumbernya. Kedua orang Swedia dan satu orang Finlandia yang mendirikan MySQL AB adalah: David Axmark, Allan Larsson, dan Michael "Monty" Widenius [10].

2.4.2. MySQL *Stored Program*

Stored program atau dikenal juga sebagai *stored module* atau *stored routine* adalah kumpulan instruksi SQL yang disimpan dan dijalankan di database server. *Stored program* baru diimplementasikan pada MySQL versi 5.0. MySQL *stored program* terdiri dari tiga jenis utama [11] :

1. *Stored procedures*

   Stored procedures merupakan tipe yang paling umum dari *stored program*. *Stored procedure* adalah bagian program umum yang dieksekusi sesuai permintaan dan dapat menerima berbagai parameter input dan output.

2. *Stored functions*

   Stored functions sama dengan *stored procedures*, tetapi hasil eksekusinya mengembalikan nilai tunggal. Lebih penting lagi, *stored function* dapat digunakan bersama dengan standar SQL *statement*, mengizinkan *programmer* untuk secara efektif memperluas kemampuan bahasa SQL.

3. *Trigger*

   Trigger merupakan *stored programs* yang di aktifkan sebagai respon atau dipicu oleh aktifitas didalam database. Biasanya, *trigger* akan dilibatkan pada sebagai respon DML operation (insert, update, delete) atau dengan kata lain trigger akan dijalankan secara otomatis pada saat atau sesudah modifikasi data. *Trigger* dapat digunakan untuk validasi data.

Keuntungan-keuntungan dari menggunakan *stored program*, yaitu

1. Mengurangi beban *client* yang mungkin akan kelebihan beban.

2. Mengurangi *network traffic* karena pengoperasian *stored program* berada pada sisi server.
3. Meningkatkan keamanan karena akses *user* ke database dibatasi.

2.5. Algoritma Pencarian Kata

Algoritma pencarian kata atau yang lebih dikenal dengan algoritma pencocokan *string* merupakan algoritma yang dikebangkan untuk mencocokkan kata. Menurut cara pembacaan teks, algoritma pencocokan *string* dapat dibedakan atas dua cara pembacaan [12] :

1. Dari kiri ke kanan

   Algoritma pencarian dengan teknik ini sangat banyak. Hampir sebagian besar algoritma pencarian menggunakan cara pembacaan teks dari kiri ke kanan.

2. Dari kanan ke kiri

   Dalam Algoritma ini, terdapat algoritma *Boyer-Moore* yang dianggap merupakan salah satu algoritma yang utama dan algoritma standar dalam pencocokan *string*.

Sampai saat ini terdapat banyak algoritma pencocokan *string*, baik merupakan algoritma yang diciptakan dari awal maupun berupa pengembangan dari algoritma yang sudah ada. Untuk kompleksitas waktu beberapa algoritma yang akan dibahas, dapat dilihat pada Tabel 2.1 [12].

Secara umum, istilah yang terdapat dalam pencocokan *string* antara lain teks dan *pattern*. Teks (*text*) adalah (*long*) *string* yang panjangnya n. *Pattern* yaitu *string* dengan panjang m karakter (m < n) yang akan dicari di dalam teks.

Tabel 2.1 Perbandingan Kompleksitas Beberapa Algoritma Pencocokan String

Algorithm	**Fase Preprocessing**	**Fase Pencarian**
Brute Force	0 (tidak ada)	O(mn)
Rabin-Karp	O(m)	O(mn)
Knuth-Morris-Pratt	O(m)	O(n+m)
Boyer–Moore	O(m + σ)	O(n/m), O(n)

2.5.1. Algoritma *Brute Force*

Algoritma *Brute Force* merupakan algoritma yang ditulis tanpa memikirkan peningkatan performa. Karakteristik algoritma *Brute Force* :

- Tidak perlu fase *prepocessing* (tahap sebelum melakukan search atau pencocokan *string*).
- Selalu berpindah tepat 1 langkah ke kanan.
- Perbandingan dapat dilakukan pada urutan apa saja.
- Fase pencarian memiliki kompleksitas O(mn)
- perbandingan karakter yang terjadi diharapkan 2n

Algoritma *Brute Force* melakukan pencarian pada setiap posisi di dalam teks antara 0 dan n-m, tidak peduli apakah terjadi pengulangan pola atau tidak. Kemudian, setelah setiap percobaan, *pattern* di geser tepat 1 posisi ke kanan.

2.5.2. Algoritma *Rabin-Karp*

Algoritma *Karp-Rabin* diciptakan oleh Michael O. Rabin dan Richard M. Karp pada tahun 1987 dengan menggunakan fungsi *hashing* untuk menemukan pattern [13]. Pada algoritma ini, terdapat fase preprocessing yang merupakan fase dimana kata/*pattern* yang dicari diubah terlebih dahulu dengan fungsi *hashing* menjadi nilai numerik. Diubahnya kata menjadi nilai numerik dengan tujuan untuk mempermudah pencarian karena secara umum lebih cepat mencari nilai numerik yang memiliki 10 kemungkinan (digit 0-9) dibandingkan huruf abjad

dengan 26 kemungkinan (karakter a-z). Proses yang terjadi pada algoritma *Rabin-Karp* akan dijelaskan sebagai berikut [14] :

1. Asumsikan *string* persamaan kata adalah string t yang panjangnnya m dan *pattern* (*string* yang dicari yaitu string jawaban siswa) P panjangnya n.
2. Asumsikan Si menyatakan sebuah *string* dengan panjang n.
3. Ide utama dari penggunaan algoritma ini adalah memanfaatkan fungsi *hash* untuk memetakan Si yaitu *string* yang akan dicari ke dalam himpunan. Fungsi *hash* digunakan untuk menempatkan suatu *record* yang mempunyai nilai kunci k. Fungsi *hash* yang paling umum berbentuk ;

   H (k) = k mod m..... (3.1)

   Di dalam algoritma ini k berupa *string* P atau *string* Si
4. Sketsa Algoritma
   a. Pertama hitung nilai fungsi *hash* dari *string* P.

      Catatan : Ketika proses pencarian berlangsung, tidak perlu lagi menghitung nilai fungsi *hash* dari *string* Si karena fungsi Si sebelumnya telah dihitung dan hasilnya diletakkan pada tabel persamaan kata yang disimpan oleh database.
   b. Lakukan penelusuran terhadap Si, jika h(Si) = h(P), maka lakukan pencocokan antara *string* Si dengan *string* p secara *Brute Force*. Pada tahap ini, pencocokan dengan *Brute Force* akan diganti dengan pencocokan kata yang telah disediakan oleh MySQL.
   c. Jika h(Si) =! h(P) , tidak perlu dilakukan pencocokan *string* dan penelusuran dilanjutkan kembali terhadap *string* yang berikutnya sampai ditemukan atau sampai *string* terakhir.

Karakterikstik Algoritma *Rabin-Karp* :

- Menggunakan sebuah fungsi *hashing*
- Fase *prepocessing* menggunakan kompleksitas waktu O(m)
- Untuk fase pencarian kompleksitasnya : O(mn)
- Waktu yang diperlukan O(n+m)

2.5.3. Algoritma *Knuth-Morris-Pratt*

Algoritma *Knuth-Morris-Pratt* (KMP) [12] bergerak dari kiri ke kanan seperti algortima *Brute Force* tetapi memilik kemampuan yang lebih baik dalam hal melakukan pergeseran *pattern*. Rancangan algoritma *Knuth-Morris-Pratt* mengikuti analisis dari algoritma *Morris* dan *Pratt* yang sebelumnya telah ditemukan terlebih dahulu oleh J.H Morris (jr) dan V.R Pratt pada tahun 1970. Bersama dengan D.E Knuth, algoritma ini menjadi *Knuth-Morris-Pratt* dengan berbagai perbaikan dari algoritma sebelumnya. Karakteristik utama algorima KMP :

- Kompleksitas ruang dan waktu untuk fase *prepocessing* adalah O(m)
- Kompleksitas waktu untuk fase pencarian : O(n+m)

Pada algoritma KMP dikenal adanya fungsi pinggiran (*Border Function*) yang didefinisikan sebagai ukuran terpanjang dari *pattern* yang juga akhiran dari *pattern* tersebut.

2.5.4. Algoritma Boyer-Moore

Algoritma *Boyer-Moore* dianggap sebagai algoritma pencocokan *string* yang paling efisien pada aplikasi umum [15]. Berbagai versi algoritma ini digunakan dalam teks *editor* untuk perintah pencarian dan pergantian (*find and replace*). Algoritma pencocokan *string Boyer-Moore* didasarkan atas dua teknik :

1. Teknik *looking-glass*, menemukan *pattern* di dalam teks dengan menggerakan pattern mundur dimulai dari akhir teks.
2. Teknik *character-jump*, pergeseran karakter yang dilakukan saat terjadi ketidak cocokan

Karakteristik utama algoritma *Boyer-Moore* :

- Melakukan perbandingan dari kanan ke kiri
- Fase persiapan / *prepocessing* membutuhkan kompleksitas waktu O(m + σ)
- Fase pencarian : kompleksitas waktunya O(mn)
- Pada kasus terburuk, sebanyak 3n karakter teks yang dibandingkan untuk *pattern* yang tak berulang.
- Kasus terbaik O(n/m)

2.5.5. Algoritma DFA (*Determined Finite Automata)*

DFA merupakan salah satu algoritma pencarian kata yang dikenal sebagai suatu metode yang digunakan untuk menerima suatu *pattern* atau tidak menerimanya. Pada optimasi algoritma DFA, sistem tidak lagi membaca teks yang lebih kecil dari karakter yang diperlukan yaitu dengan menambahkan kondisi ((T.length() – i) >= (5 – Status)) pada while [16]. Dengan sedikit penyesuaian, penulis akan mencoba menaruh metode DFA pada awal proses pencarian. Penyesuaian yang dilakukan adalah dengan menambahkan kondisi yang membuat sistem akan melewatkan atau tidak membaca kata jika (panjang kata pada database =! Panjang kata yang dicari) sehingga sistem tidak perlu lagi memeriksa kata yang tidak mendekati atau tidak cocok dengan kata yang dicari.

BAB 3
PERANCANGAN SISTEM PENCARIAN KATA

Pada bab ini akan dibahas tentang perancangan pencarían kata pada SIMPLE-O. Perancangan ini terbagi menjadi 2 yaitu perancangan sistem dan peracangan uji coba. Pada perancangan sistem, terdapat dua sistem pencarían kata yang akan dirancang yaitu SIMPLE V3 dan SIMPLE V4. Perancangan uji coba merancang proses uji coba yang akan dilakukan dengan berbagai kondisi.

3.1. Perancangan SIMPLE V3

SIMPLE V3 merupakan pencarian kata yang proses pencariannya diserahkan sepenuhnya pada database MySQL. Proses pencarian ini tidak menggunakan database abstraction layer AdobDB. Dengan demikian, *overhead* database *abstraction layer* AdobDB dapat dieliminasi sehingga diharapkan akan mempercepat proses pencarian kata.

Pada pencarian kata SIMPLE V3 digunakan fungsi pencarian yang ada pada database MySQL yaitu menggunakan fungsi *"SELECT"* untuk mengambil kode persamaan kata pada table persamaan kata jika kata pada database sama dengan kata yang dicari yaitu jawaban mahasiswa ataupun jawaban dosen. Source code pencarian kata SIMPLE V3 ditunjukkan pada Gambar 3.1.

```
SELECT persamaan FROM tb_perskata WHERE kata ='$kata'
```

Gambar 3. 1 Source Code Pencarian Kata SIMPLE V3

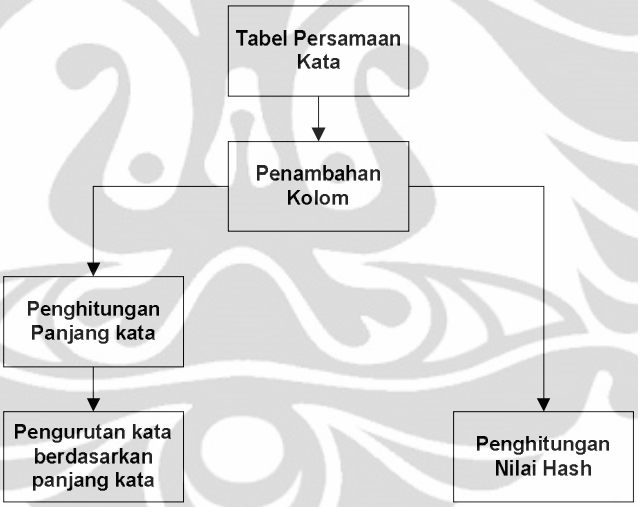
3.2. Perancangan SIMPLE V4

SIMPLE V4 merupakan sistem pencarian kata dengan menggunakan algoritma *Rabin-Karp*. Dipilihnya algoritma *Rabin*-Karp sebagai algoritma untuk pencarian kata karena algoritma ini dianggap paling cocok untuk pencarian kata dalam jumlah besar. Konsep perancangan pencarian kata dengan algoritma Rabin-Karb terbagi menjadi 2, yaitu :

1. Pengoptimalan perancangan tabel persamaan kata
2. Perancangan sistem pencarian kata SIMPLE V4

3.2.1. Pengoptimalan Perancangan Tabel Persamaan Kata

Tabel persamaan kata merupakan tabel yang akan diakses oleh proses pencarian kata untuk mencari kata yang memiliki arti yang sama. Pada tabel persamaan kata terdapat beberapa kolom yaitu kolom no, kolom kata, kolom kata dasar, kolom kode kata dan kolom kode persamaan. Pada perancangan tabel persamaan kata yang ditunjukkan pada Gambar 3.2, penulis mencoba untuk menambahkan kolom panjang *string* yang merupakan kolom yang menunjukkan banyaknya karakter yang terdapat pada sebuah *string* dan juga mengurutkan *string* berdasarkan panjang karakternya. Hal ini dilakukan untuk mempermudah proses pencarian kata yang akan menggunakan salah satu metode yang terdapat pada optimasi algoritma DFA (*Determined Finite Automata*) yang penulis telah sesuaikan dengan keadaan sistem. Selain itu, penulis juga akan menambahkan kolom nilai *hash* yang nanti akan digunakan dalam pencarian kata.



Gambar 3.2 Activity diagram pengoptimalan database

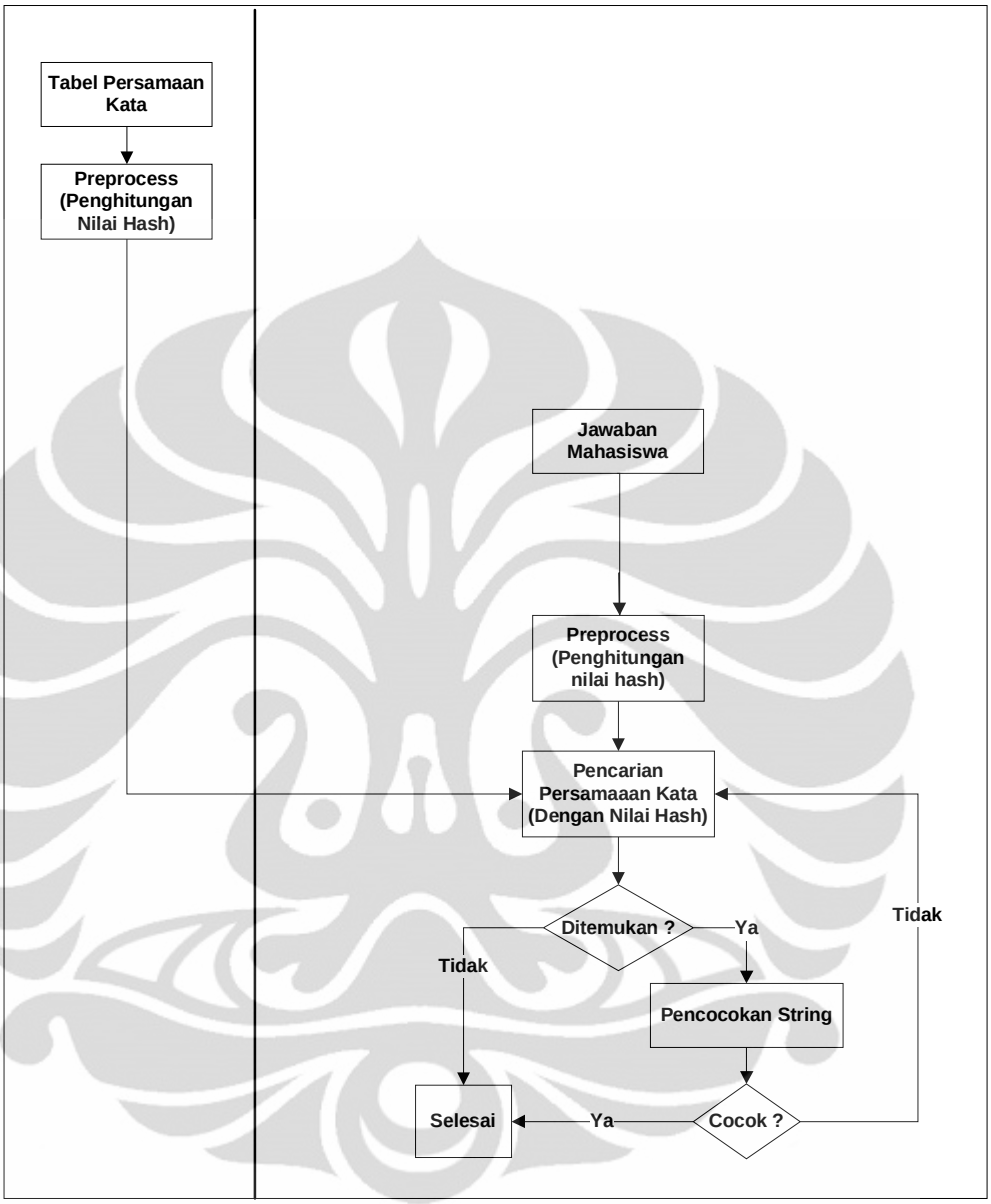
3.2.2. Perancangan Sistem Pencarian Kata SIMPLE V4

Dalam perancangan sistem pencarian kata SIMPLE V4 yang akan ditempatkan pada database MySQL, digunakan algoritma *Rabin_Karp* dengan fungsi *Hash* untuk pencocokan kata yang diperkirakan dapat mempercepat proses tersebut. Gambar 3.3 memperlihatkan *pseudocode* standar algoritma *Rabin-Karp*. Selain itu, dilakukan percobaan untuk menggabungkan salah satu metode pada optimasi algoritma DFA (*Determined Finite Automata*) untuk kemudian

disisipkan pada awal pencarian kata, sebelum algoritma *Rabin_Karp* di jalankan. Pada dasarnya, proses dari pencarian kata adalah mencocokkan kata yang memiliki karakter yang sama dengan katakter kata yang dicari. Jadi pada skripsi ini, akan sering digunakan istilah pencocokan kata, pencocokan *string*, pencarian *string* yang memiliki arti yang sama dengan pencarian kata. Diagram pencarian kata akan dijelaskan pada Gambar 3.4.

```
function RabinKarpSet(string s[1..n], set of
string subs, m) {
     set hsubs := emptySet
     for each sub in subs
         insert hash(sub[1..m]) into hsubs
     hs := hash(s[1..m])
     for i from 1 to n
         if hs € hsubs
             if s[i..i+m-1] = a substring
with hash hs
                 return i
         hs := hash(s[i+1..i+m])
     return not found
 {
```

Gambar 3. 3 Standar Pseudocode Algoritma Rabin-Karp



Gambar 3.4 Activity diagram pencarian kata dengan *Rabin-Karp* pada Simple-O

3.2.3. Fungsi *Hash*

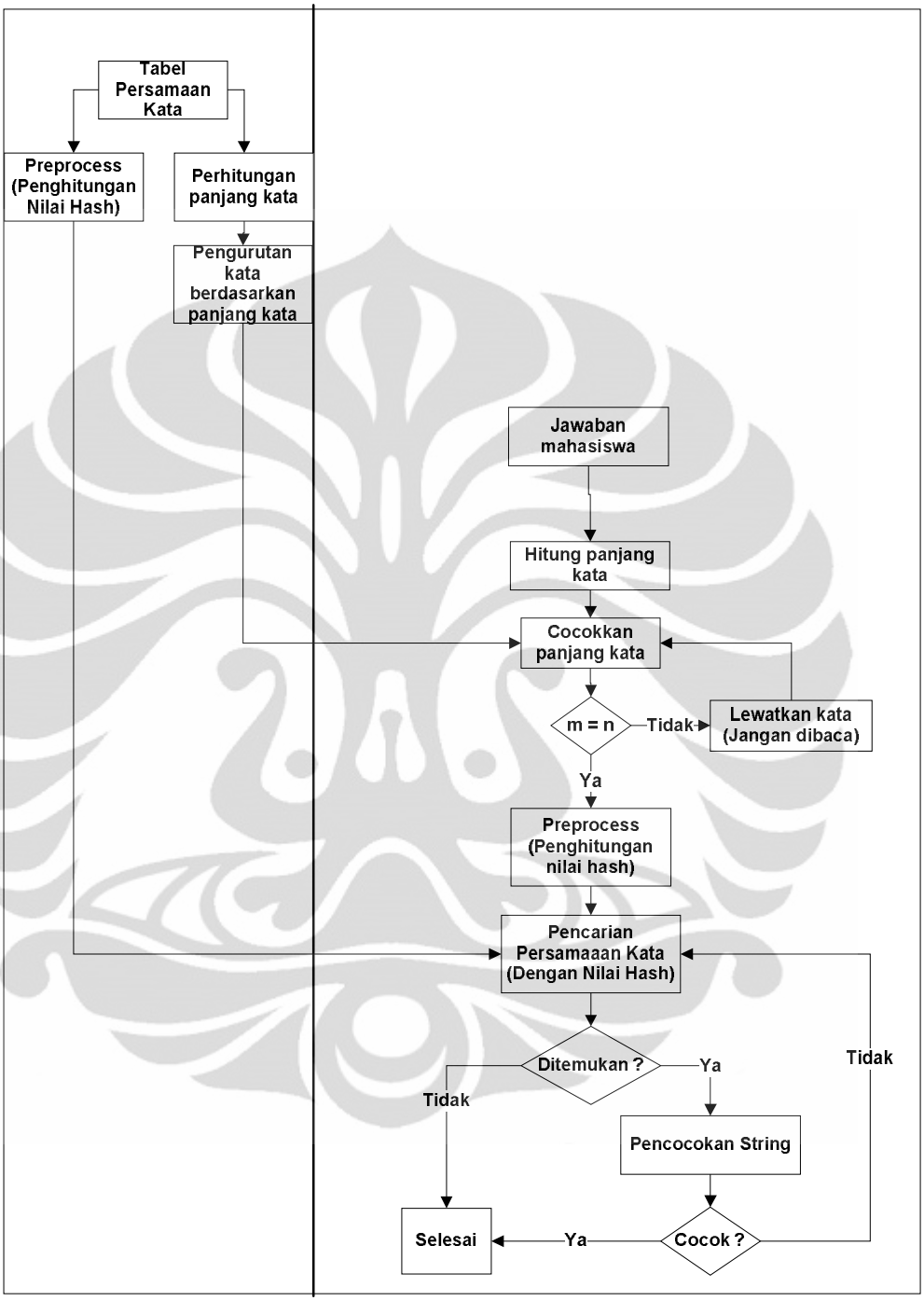
Fungsi *hash* merupakan sebuah fungsi yang mengubah *string* menjadi nilai numerik, yang disebut nilai *hash*. Contohnya adalah *Hash*(”Hello”) = 5. Faktanya, jika terdapat dua buah *string* yang sama, maka nilai *hash* mereka pasti sama sehingga fakta inilah yang mendasari Algoritma *Rabin-Karp*. Terdapat beberapa kendala dalam pencocokan dengan fungsi ini yaitu banyaknya *string* yang berbeda

yang memiliki nilai hash yang sama. Penyelesaian masalah pertama, menyebabkan masalah kedua yaitu belum tentu *string* yang memiliki nilai *hash* yang sama adalah sama sehingga untuk menyelesaikan masalah ini dilakukan pencocokan *string* dengan fungsi MySQL.

Kunci agar algoritma *Rabin-Karp* berjalan baik, terdapat pada pemilihan nilai *hash* nya. Salah satu cara yang terkenal dan efektif adalah memperlakukan setiap *substring* sebagai suatu bilangan dengan basis tertentu, biasanya yang dijadikan basis adalah bilangan prima berukuran besar. Sebagai contoh, *substring* "hi" dengan basis 101, akan mempunyai *hash value* sebesar 104*1011 + 105*1010 = 10609 (nilai ASCII untuk 'h' = 104 dan 'i' = 105).

3.2.4. Perancangan Gabungan

Ini merupakan perancangan akhir dari sistem pencarian kata SIMPLE V4 yang akan dikembangkan. Pada perancangan ini akan dijelaskan penggabungan antara metode DFA dengan algoritma *Rabin-Karp* dengan fungsi *hans*. Gambar 3.5 menjelaskan keseluruhan proses pencarian kata.



Gambar 3.5 Activity diagram proses pencarian kata SIMPLE-O

Pada Gambar 3.5, terlihat adanya garis melintang yang memisahkan proses yang terjadi antara tabel persamaan (bagian kiri) dan *input user* (bagian kanan). Pemisah tersebut untuk menegaskan bahwa proses yang terjadi pada tabel

persamaan kata yang kemudian ditambahkan nilai *hash* dan dihitung panjang kata atau karakter lalu diurutkan merupakan proses yang sebelumnya telah tersimpan di database sehingga tidak perlu diproses kembali kecuali ada penambahan kata dalam tabel persamaan kata. Pada pencarian persamaan kata, *input* dari jawaban mahasiswa dihitung panjang karakternya. Setelah itu, pada pencarian pertama panjang karakter *input* akan dicari pada kolom panjang karakter di tabel persamaan kata. Jika terdapat panjang karakter yang tidak sama, maka *input* hanya akan melewatkan tanpa memeriksa lebih lanjut. *Input* baru akan memeriksa jika panjang *input* yang dicari (n) sama dengan panjang kata pada tabel persamaan kata (m). Setelah ditemukan panjang kata yang sama, maka pada *input* akan dilakukan perhitungan nilai *hash*. Berikutnya, nilai *hash input* akan dicocokkan dengan nilai *hash* pada kata-kata yang terdapat pada tabel persamaan kata yang tadi memiliki panjang karakter yang sama. Jika ditemukan nilai *hash* yang sama maka kata pada input akan dicocokkan pada kata tersebut dengan pencocokan kata pada database MySQL . Jika cocok, maka kata telah ditemukan dan akan diambil kode persamaan kata untuk dilakukan pencocokan dengan kata kunci. Jika tidak cocok maka dilakukan pemeriksaan kembali persamaan kata dengan nilai *hash* sampai cocok atau sampai kata terakhir dalam tabel yang berarti kata *input* tidak ada dalam tabel persamaan kata.

3.3. Perancangan Uji Coba

Perancangan uji coba ini merancang kondisi-kondisi uji coba yang akan dijalankan sistem untuk membandingkan kecepatan pencarian kata. Proses pengujian dilakukan dengan memvariasikan banyaknya kata yang akan dicari. Kata yang akan dicari disesuaikan dengan keadaan sistem yang pada kenyataannya akan mencari kata pada jawaban mahasiswa sehingga variasi yang diterapkan adalah dengan mempertimbangkan banyaknya mahasiswa yang mengakses pada waktu yang bersamaan dan jumlah soal yang dikerjakan mahasiswa dalam satu ujian. Data yang diuji berupa dokumen teks yang diambil dari jawaban ujian mahasiswa yang sebelumnya telah di kumpulkan. Dokumen yang akan diuji disesuaikan dengan kondisi yang akan ditetapkan dalam pengujian untuk mengukur algoritma pencarian kata yang paling cepat. Untuk keseluruhan

pengujian, program akan dijalankan sebanyak sepuluh kali untuk masing-masing pengujian. Pengujian terdiri dari tiga kondisi, yaitu

1. Pengujian 1

   Pengujian dengan mengambil sampel dari pencarian satu kata dan menghitung banyaknya jumlah kata yang harus dilewati sistem untuk menemukan kata yang dicari. Tujuan dari pengujian ini :

   - Membandingkan kecepatan masing-masing sistem dalam mencari sebuah kata
   - Menganalisa pengaruh banyaknya jumlah kata yang dilalui terhadap kecepatan algoritma

2. Pengujian 2

   Mengambil sampel dari jawaban ujian, dimana akan diambil data pertahap dengan melihat banyaknya jumlah kata yang akan dicari. Pada hal ini, akan diambil sampel dari penggunaan satu buah jawaban soal hingga delapan buah jawaban soal. Tujuan dari pengujian ini adalah melihat pengaruh banyaknya jumlah kata yang dicari terhadap kecepatan masing-masing sistem.

3. Pengujian 3

   Mengambil sample dari banyaknya user yang mengakses simple-o secara bersamaan, terdiri dari 10, 20 dan 30 user.

   a. 10 client dengan @ 8 soal
   b. 20 client dengan @ 8 soal
   c. 30 client dengan @ 8 soal

   Tujuan dari pengujian ini adalah membandingkan pengaruh kecepatan dari banyaknya user yang mengakses sistem secara bersama-sama terhadap masing-masing sistem.

BAB 4
IMPLEMENTASI, PENGUJIAN DAN ANALISA

4.1. Implementasi Perangkat

Tahap implementasi ini akan menerapkan apa yang telah dirancang pada perancangan sistem. Sebelum penerapan perancangan dijabarkan, ada beberapa kondisi yang harus dipenuhi oleh suatu sistem berbasis web yaitu mengenai spesifikasi alat untuk menjalankan sistem tersebut. Berikut ini merupakan spesifikasi hardware dan software untuk menjalankan sistem :

4.1.1. Hardware

Berikut ini adalah spesifikasi hardware yang digunakan :

- Motherboard : Gigabyte GA
- Prosesor : Intel Core 2 Quad Q8300 2.33 Ghz
- Memory (RAM) : 4 GB

4.1.2. Software

Berikut ini adalah spesifikasi software yang digunakan :

- Sistem Operasi : Ubuntu 10.4.2
- Web Server : Apache 2.2.14
- Sistem Basis Data : MySQL 5.1.37
- PHP Engine : PHP 5.3.2

4.2. Implementasi Sistem

Ini merupakan tahap implementasi 4 sistem pencarian kata untuk dibandingkan dalam mencari sistem yang tercepat. Seperti yang telah dijelaskan dalam bab pendahuluan, bahwa sistem yang akan dibandingkan adalah sistem yang telah dan sedang digunakan yaitu SIMPLE V1 dan SIMPLE V2 dengan sistem yang sedang dirancang yaitu SIMPLE V3 dan SIMPLE V4. Berikut ini akan dijabarkan satu persatu mengenai sistem pencarian kata yang akan dibandingkan.

4.2.1. SIMPLE V1

SIMPLE V1 merupakan sistem pencarian kata tahap pertama atau versi pertama yang dibangun untuk menunjang pencarian kata pada database. Sistem ini juga sistem yang pertama kali digunakan pada SIMPLE-O. Pada sistem ini, semua proses pencarian kata dipindahkan ke PHP. Proses pencarian kata pada sistem ini adalah sistem akan memindahkan semua kata dan kode persamaan pada tabel persamaan kata untuk dimasukkan sebagai *array* di PHP. Lalu proses pencarian katanya dilakukan oleh fungsi *array_search()* yang ada di PHP. Dari proses tersebut diketahui bahwa seluruh proses pencarian kata terletak di PHP.

4.2.2. SIMPLE V2

SIMPLE V2 merupakan sistem pencarian kata versi kedua yang melakukan proses pencarian kata dengan memecah tabel persamaan kata menjadi 5 buah tabel tergantung dari banyaknya kemungkinan kata yang memiliki persamaan karakter/huruf. Pada pencarian kata ini terdapat dua proses yaitu proses untuk memecah imbuhan sebuah kata dan mencarinya kedalam database. Proses memecah imbuhan sebuah kata dilakukan pada PHP. Sedangkan proses pencarian kata akan dilakukan di database MySQL. Jadi dapat dikatakan bahwa SIMPLE V2 memanfaatkan fungsi pada PHP dan MySQL untuk melakukan pencarian kata.

4.2.3. SIMPLE V3

Implementasi SIMPLE V3 sangatlah sederhana, karena seluruh proses pencarian kata diserahkan ke MySQL sehingga sistem hanya menggunakan perintah ”*SELECT*” pada database MySQL untuk pencarian kata.

4.2.4. SIMPLE V4

SIMPLE V4 merupakan sistem pencarian kata yang sedang dikembangkan. Sistem ini seperti yang telah dijelaskan pada bab perancangan, menggunakan algoritma *Rabin-Karp* sebagai algoritma pencocokan kata. Proses implementasi pada SIMPLE V4 terbagi menjadi beberapa bagian, yaitu :

1. Pembentukan nilai *hash*

   Tahap pembentukan nilai *hash* dapat dikatakan juga sebagai fase *preprocessing* yaitu fase yang terjadi sebelum proses pencarian kata

dilakukan. Fase ini mengubah kata yang dicari dan kata yang berada didalam database menjadi nilai *hash*. Nilai hash dibentuk dengan mengubah kata menjadi bilangan ASCII dan membaginya dengan bilangan prima. Pembetukan algoritma perubahan nilai *hash* menggunakan *stored function*. Digunakannya *stored function* karena pada akhirnya fungsi ini akan diletakkan pada proses pencarian kata sehingga dibutuhkan fungsi yang dapat digunakan bersamaan dengan perintah SQL. Gambar 4.1 menunjukkan cuplikan kode pembentukan nilai hash.

```
CREATE FUNCTION `ascii_string_v01`(in_string
VARCHAR(20)) RETURNS varchar(256)
DETERMINISTIC
BEGIN

  DECLARE i INT DEFAULT 1;
  ....

  SET string_len=LENGTH(in_string);
  ....

  WHILE (i<string_len) DO
     ....
     SET out_string=
ASCII(SUBSTR(in_string,i,1))*pow(10,y);
     ....

  END WHILE;
  ....
  RETURN (jumlah_mod);

END
```

Gambar 4. 1 Cuplikan Kode Pembentukan Hash

2. Proses Pencarian Kata

Sistem pencarian kata ini menggunakan algoritma *Rabin-Karp* yang di buat di dalam *stored procedure*. Didalam sistem pencarian kata ini, terdapat perubahan kata menjadi nilai *hash* sehingga digunakan kode nilai *hash* dalam algoritma ini. Gambar 4.2 menunjukkan cuplikan kode pencarian kata.

```
CREATE PROCEDURE `rabinkarp_v01`(jawaban_mhs
varchar(100))
luar: BEGIN

  DECLARE ketemu VARCHAR(50) DEFAULT 0;
  ....
  DECLARE db_cursor1 CURSOR FOR

  ....

  db_loop: WHILE(ketemu=0) DO
  ....
     IF
ascii_string_v01(jawaban_mhs)=l_nilai_hash
THEN
              ...
             END IF;
     END IF;
  END WHILE db_loop;
  ...

END luar
```

Gambar 4. 2 Cuplikan Kode *Rabin-Karp*

3. Peletakan pencarían kata dalam PHP

Tahap implementasi terakhir adalah penggabungan sistem pencarían kata kedalam keseluruhan sistem SIMPLE-O yang diletakkan di PHP. Untuk memanggil sistem pencarían kata yang menggunakan *stored procedure*, digunakan fungsi "*call*". Pada akhirnya, *output* dari sistem pencarían kata ini adalah kode persamaan kata. Gambar 4.3 menunjukkan cuplikan kode fungsi pencarían kata.

```
function carikata($kata)
{
....
    }
    $sql = "CALL rabinkarp_v01(?)";
    $stmt = $mysqli->prepare($sql);
    if ($mysqli->errno) {die($mysqli->errno."::
".$mysqli->error);}

    $jawaban_mhs = $kata;
    $stmt->bind_param("s", $jawaban_mhs);
    $stmt->execute(  );
    ....
    while ($stmt->fetch(  )) {
            }
    $stmt->close();
}
```

Gambar 4. 3 Cuplikan Fungsi Pencarian Kata

4.3. Scenario Uji Coba

Pada scenario uji coba ini, akan dijelaskan proses pencarian kata yang berjalan dalam mencocokkan kata antara jawaban mahasiswa atau jawaban dosen dengan kata pada database. Masing-masing sistem memiliki proses yang berbeda dalam pencarian kata, tergantung bagaimana struktur penyimpanan kata pada database.

4.3.1. Scenario SIMPLE V1

Database SIMPLE V1, hanya menggunakan 1 tabel kata yang terdiri dari 1141 *rows* untuk pencarian kata. Letak kata pada tabel berurut sesuai abjad sehingga dalam pencariaannya kata yang terletak paling bawah akan memiliki waktu pencarian yang relatif lebih lama dibandingkan kata yang berada diatasnya. Waktu pencarian lebih lama disebabkan oleh jumlah kata yang harus dilalui selama pencarian.

Pada pencarian kata yang dilakukan oleh SIMPLE V1, pertama-tama sistem akan memindahkan seluruh kata dan persamaan kata pada tabel persamaan kata ke dalam *array* PHP. Urutan kata yang dipindahkan sesuai dengan urutan kata yang berada pada tabel database. Pada proses pencariannya, sistem akan

mulai melakukan pencarian dari kata teratas atau dari array ke 0. Jadi saat melakukan pencarian kata ”*broadcast*” yang terletak pada array ke 143, sistem harus melewati 143 kata sampai menemukan kata yang dicari.

4.3.2. Scenario SIMPLE V2

Database SIMPLE V2 memiliki 5 buah tabel untuk proses pencarian kata. Kelima tabel tersebut dibagi berdasarkan kata depan yaitu tabel ber, tabel di, tabel me, tabel per dan tabel kata. Masing-masing tabel memiliki jumlah kata yang berbeda namun pengurutan kata pada setiap tabel tetaplah sama yaitu berdasarkan urutan abjad.

Proses pencarian kata pada SIMPLE V2 adalah pertama-tama kata yang dicari dipisahkan kata depannya. Misalnya, kata yang dicari adalah diimplementasikan. Setelah kata diimplementasikan dipisahkan kata depannya, lalu akan dicocokkan kata depan dengan nama tabel. Dari proses tersebut didapatkan bahwa kata depan diimplementasikan adalah ”di” sehingga pencarian kata diimplementasikan akan dilakukan pada tabel di. Kata diimplementasikan terletak pada urutan ke 248 pada tabel di sehingga sistem harus melewati 247 kata sebelum menemukan kata yang dicari.

4.3.3. Scenario SIMPLE V3

Database SIMPLE V3 sama dengan database pada SIMPLE V1. Perbedaannya hanya proses pencarian pada SIMPLE V3 langsung mencari pada databasenya. Jadi saat melakukan pencarian kata ”*broadcast*” yang terletak pada kolom ke 144, sistem harus melewati 143 kata sampai menemukan kata yang dicari.

4.3.4. Scenario SIMPLE V4

Pada dasarnya Database SIMPLE V4 sama dengan database SIMPLE V1 dan SIMPLE V3. Perbedaannya hanyalah pada SIMPLE V4 terdapat sedikit penyesuaian pada fungsi pencariannya yaitu hanya mencocokkan kata yang memiliki panjang karakter sama dengan kata yang dicari. Jadi saat mencari kata “*broadcast”* yang mengandung 9 karakter, sistem hanya akan mencari pada kata yang mengandung 9 karakter juga. Jumlah kata yang mengandung 9 katakter

sebanyak 29 kata dan kata "*broadcast"* berada pada kolom ke 10. Oleh karena itu, untuk dapat menemukan kata "*broadcast"* sistem harus melewati 9 kata diatasnya.

4.4. Analisa Hasil Uji Coba Sistem

Hasil uji coba sistem ini merupakan hasil yang didapatkan setelah dilakukan uji coba terhadap sistem. Karena skripsi ini hanya untuk melihat perbandingan kecepatan proses pencarian kata, sehingga uji coba hanya akan melibatkan proses pencarian kata itu sendiri dan tidak melibatkan proses keseluruhan sistem SIMPLE-O Hasil-hasil dari uji coba akan berupa perbandingan-perbandingan antara satu sistem dengan sistem lainnya.

4.4.1. Analisa Pengujian 1

Setelah melakukan pengujian dengan mengambil sampel dari pencarian satu kata sebanyak 10 kali, didapatkan hasilnya untuk masing-masing sistem sebagai berikut :

Tabel 4. 1 Hasil Pengujian 1

No	Kata	Waktu (detik)			
		SIMPLE V1	SIMPLE V2	SIMPLE V3	SIMPLE V4
1	Bertambah	0.007313	0.0007307	0.000153	0.0119119
2	Broadcast	0.006967	0.0007011	0.000109	0.0023740
3	Diimplementasikan	0.006923	0.0006870	0.000111	0.0025423
4	Domain	0.006875	0.0007000	0.000110	0.0041508
5	Jaringan	0.007573	0.0007082	0.000109	0.0046018
6	Mekanisme	0.006932	0.0006919	0.000112	0.0136913
7	Multimedia	0.006812	0.0006902	0.000109	0.0045043
8	Pemograman	0.006953	0.0006856	0.000108	0.0125997
9	Protocol	0.007549	0.0007014	0.000110	0.0074656
10	Wireless	0.006853	0.0006935	0.000109	0.0116938

Dari hasil uji coba pada Tabel 4.1 yang mengambil sampel pencarian satu kata, terlihat bahwa terdapat perbedaan kecepatan pencarian kata pada setiap sistem. Jika ditinjau pengaruh kata yang dicari dengan kecepatan sistem maka diketahui bahwa dalam setiap sistem terdapat variasi kecepatan tergantung dari

kata yang dicari. Perbedaan kecepatan pada pencarian kata yang berbeda dipengaruhi oleh dua hal, yaitu

1. Letak kata yang berbeda pada database
2. Metode pencarian yang digunakan.

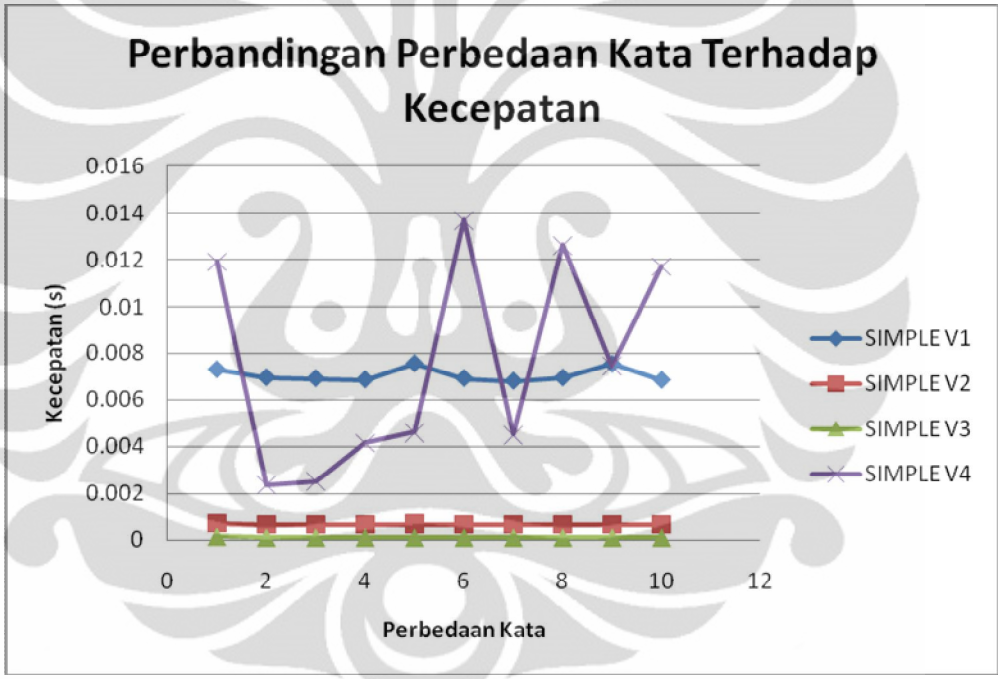
Ditinjau dari letak, kata yang lebih cepat pengaksesannya dikarenakan letak kata tersebut yang berada lebih atas dari kata yang memiliki kecepatan yang lebih lambat sehingga ketika proses pencarian berlangsung, kata-kata yang harus dilewati sistem selama proses pencarian cenderung lebih sedikit. Berikut ini merupakan tabel yang akan menunjukkan jumlah kata yang harus dilewati sistem dalam pencarian kata-kata yang diujikan :

Tabel 4. 2 Banyak Kata Dalam Pengujian Satu

NO	Kata	Banyak Kata Yang Dilalui			
		SIMPLE V1	SIMPLE V2	SIMPLE V3	SIMPLE V4
1	Bertambah	123	32	123	81
2	Broadcast	143	108	143	9
3	Diimplementasikan	247	26	247	6
4	Domain	301	188	301	34
5	Jaringan	424	311	424	31
6	Mekanisme	590	3	590	106
7	Multimedia	735	487	735	25
8	Pemograman	795	17	-	-
9	Protocol	885	556	885	58
10	Wireless	1139	811	1139	98

Jika dilihat dari banyaknya kata yang harus dilalui, SIMPLE V4 seharusnya menjadi sistem yang paling cepat diantara sistem lainnya, disusul oleh SIMPLE V2 dan SIMPLE V1 serta SIMPLE V3. Namun yang terjadi sebaliknya, SIMPLE V3 yang merupakan sistem dengan kata yang dilalui terbanyak, menghasilkan kecepatan yang jauh lebih cepat dibandingkan dibandingkan ketiga sistem lainnya dan disusul oleh SIMPLE V2 dan SIMPLE V4 serta SIMPLE V1. Walaupun banyaknya data yang harus dilalui SIMPLE V1 sama dengan SIMPLE

V3, namun ternyata SIMPLE V1 tidak sama cepat dengan SIMPLE V3 dan dapat dikatakan bahwa SIMPLE V1 merupakan sistem yang paling lambat. Jika SIMPLE V1 dibandingkan dengan SIMPLE V4, sebenarnya rata-rata kecepatan sistem mencari kata hampir sama. Padahal perbedaan banyaknya kata yang harus dilewati antara SIMPLE V1 dengan SIMPLE V4 sangatlah jauh. Dari sini dapat diketahui bahwa dalam pencarian kata, banyaknya kata yang harus dilalui sepanjang pencarian bukanlah satu-satunya faktor yang menentukan kecepatan sistem. Kecepatan pencarian juga ditentukan oleh metode pencarian kata yang digunakan. Walaupun banyaknya kata yang dilewatkan sama, jika metode yang digunakan berbeda maka kecepatan pun akan berbeda.



Gambar 4. 4 Grafik Perbandingan Perbedaan Kata Terhadap Kecepatan

Dari Tabel 4.1, dapat dibuat grafik pada Gambar 4.4 untuk meninjau metode atau sistem yang digunakan. Dari situ dapat terlihat bahwa terdapat variasi kecepatan yang terjadi pada ketiga sistem yaitu SIMPLE V1, SIMPLE V2 dan SIMPLE V3 pada pencarian kata yang berbeda. Namun, variasinya kecepatannya tidaklah terlalu jauh antara satu kata dengan kata lainnya, bahkan variasi pada SIMPLE V2 dan SIMPLE V3 hampir tidak terlihat dikarenakan perrubahan yang begitu kecil. Namun, hal yang demikian tidaklah terjadi pada SIMPLE V4 yang

kecepatan pencarian setiap kata yang berbeda sangatlah jauh variasinya sehingga dapat dikatakan bahwa kecepatan pencarian SIMPLE V4 benar-benar terpengaruh oleh letak kata yang dicari. Sistem lainnya walaupun terpengaruh, namun pengaruhnya tidak terlalu terlihat.

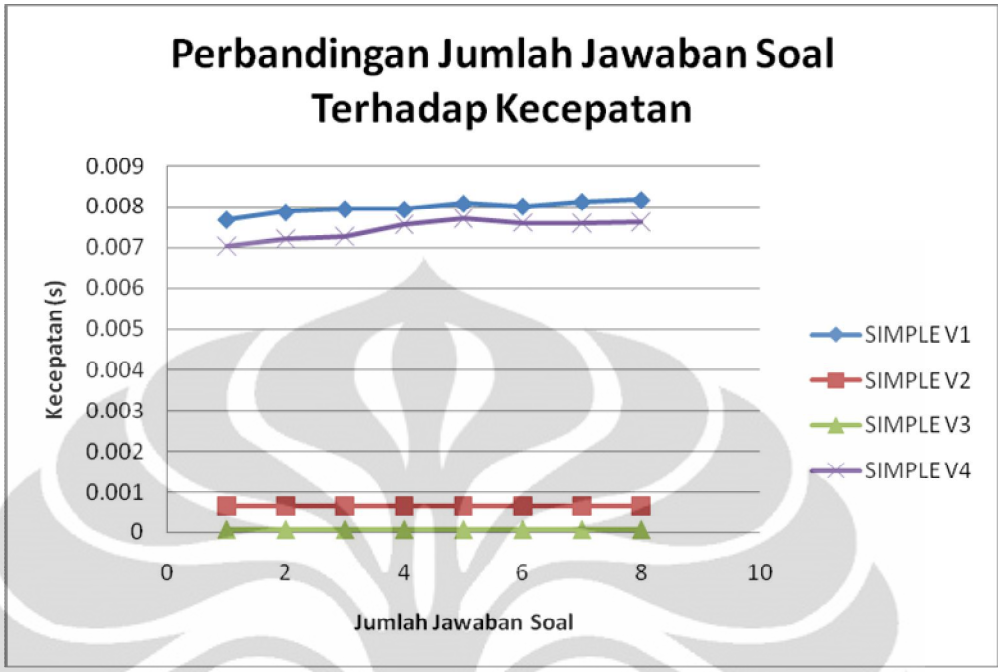
Ditinjau dari perbandingan kecepatan kata masing-masing sistem, SIMPLE V3 melakukan pencarian kata yang paling cepat diantara sistem lainnya. Kecepatan SIMPLE V3 sekitar 62 kali lebih cepat dibandingkan dengan SIMPLE V1. Sedangkan kecepatan SIMPLE V2 hanya sekitar 9 kali lebih cepat jika dibandingkan dengan SIMPLE V1. Kecepatan SIMPLE V4 tidaklah konstan, Sehingga ada saat dimana SIMPLE 4 lebih cepat dibandingkan SIMPLE V1 dan juga ada saat sebaliknya. Kecilnya kecepatan SIMPLE V4 dikarenakan tahap *preprocessing* yang lama sehingga walaupun kata yang harus dilewatkan dalam pencarian kata sedikit, total waktu pencarian menjadi lama.

4.4.2. Analisa Pengujian 2

Berikut ini adalah hasil uji coba pengujian 2 yang menguji untuk mengukur kecepatan sistem berdasarkan jumlah soal.

Tabel 4. 3 Hasil Pengujian 2

NO	Jumlah Jawaban	Waktu (detik)			
		SIMPLE V1	SIMPLE V2	SIMPLE V3	SIMPLE V4
1	1 Soal	0.007680264	0.000653094	7.42462E-05	0.007030016
2	2 Soal	0.007870633	0.000659575	7.42386E-05	0.0072094
3	3 Soal	0.007950325	0.00066217	7.46727E-05	0.007273374
4	4 Soal	0.007938678	0.00066238	7.48132E-05	0.007553402
5	5 Soal	0.008085352	0.000660441	7.51757E-05	0.007708349
6	6 Soal	0.008015893	0.000667851	7.48E-05	0.007602231
7	7 Soal	0.008129802	0.000667526	7.49662E-05	0.007578944
8	8 Soal	0.008176844	0.000667543	7.48561E-05	0.00762163



Gambar 4. 5 Grafik Perbandingan Jumlah Jawaban Soal Terhadap Kecepatan

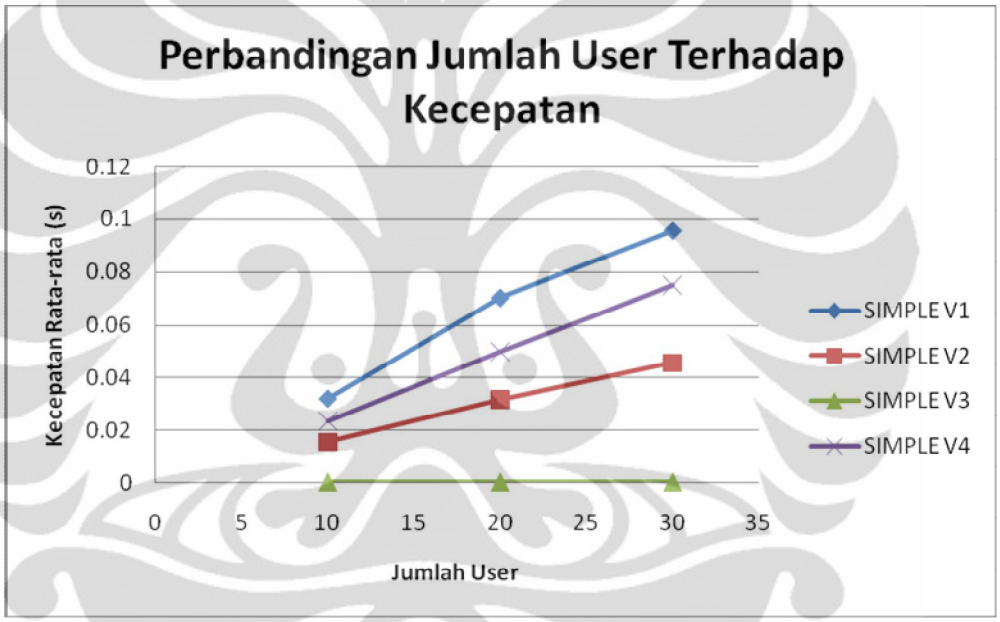
Dari hasil pengujian yang ditunjukkan pada Tabel 4.3 dan Gambar 4.5, terlihat bahwa banyaknya jawaban atau jumlah kata yang harus dicari berbanding lurus dengan kecepatan sistem. Jadi semakin banyak jawaban soal yang dicari maka semakin lama kecepatan sistem.

Jika dilakukan perbandingan satu persatu maka akan terlihat bahwa SIMPLE V3 merupakan sistem yang paling cepat diantara sistem lainnya. Untuk memperjelas perbandingan, maka SIMPLE V1 yang merupakan sistem pencarian kata yang pertama kali digunakan akan dijadikan sebagai pembanding. Kecepatan semua sistem lebih cepat dari SIMPLE V1 yaitu SIMPLE V2 12 kali lebih cepat , sedangkan SIMPLE V3 106 kali lebih cepat dan yang terakhir adalah SIMPLE V4 1.07 kali lebih cepat. Melalui pengujian dengan variasi soal ini dapat terlihat bahwa SIMPLE V3 merupakan sistem yang tercepat dalam pencarian kata.

4.4.3. Analisa Pengujian 3

Pengujian 3 menguji 3 kondisi dari banyaknya user yang mengakses sistem. Hasil dari pengujian berupa tabel-tabel yang saling terpisah antara setiap sistem. Tabel ini terdiri dari 6 kolom yaitu kolom *time*,sistem, *user, dispatch,start dan timestamp*. Keenam kolom ini menjelaskan bagiannya masing-masing. Kolom

time menjelaskan waktu yang dibutuhkan oleh suatu sistem untuk mencari kata. Selanjutnya kolom sistem merupakan kolom yang menyatakan sistem yang diujikan. Lalu kolom *user* menjelaskan *user* terlibat dalam proses pencarian kata. Kemudian kolom *dispatch* menyatakan waktu yang ditetapkan untuk menjalankan sistem. Kolom *start* merupakan waktu sistem dieksekusi sehingga dari keseluruhan kolom yang ditampilkan pada pengujian tiga, hanya kolom time yang akan digunakan dalam menganalisa kecepatan sistem dalam mencari kata. Hasil dari pengujian 3 akan dilampirkan pada Lampiran 1.



Gambar 4. 6 Grafik Perbandingan Jumlah User Terhadap Kecepatan

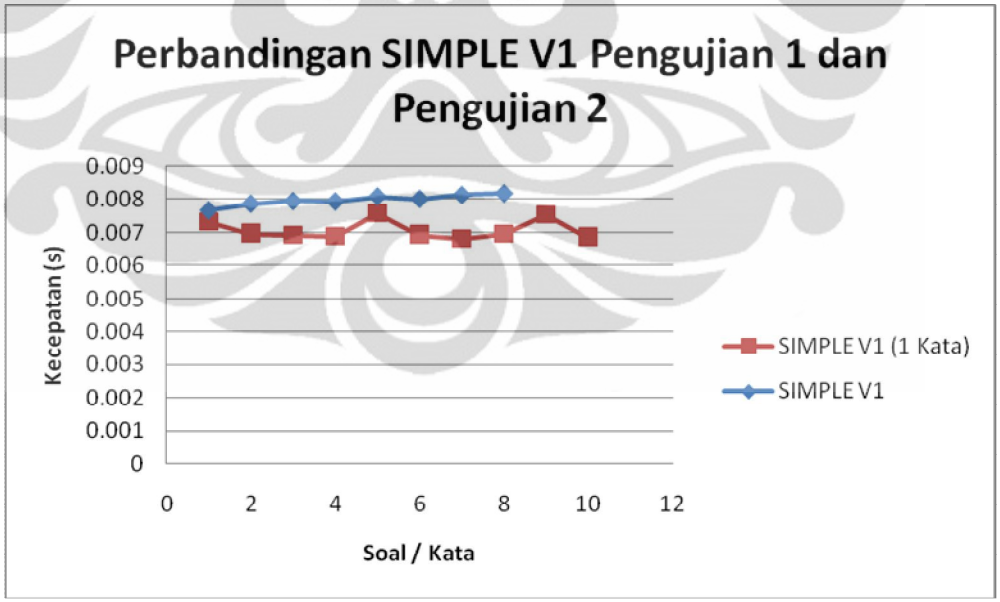
Dari grafik pada Gambar 4.6 dapat dianalisa bahwa pada SIMPLE V1, SIMPLE V2 dan SIMPLE V4 semakin banyak user yang mengakses sistem secara bersamaan maka kecepatan sistem akan semakin menurun. Pada dasarnya, semakin banyak user yang mengakses sistem juga mempengaruhi kinerja kecepatan pada SIMPLE V3. Namun, terlihat stabilnya SIMPLE V3 pada grafik disebabkan oleh perubahan kecepatan sistem yang hanya sedikit yang yaitu sekitar 0.0000253014584944 detik sehingga tidak terbaca jelas oleh grafik. Namun, walaupun kecepatan sistem menurun, SIMPLE V3 tetap menjadi yang tercepat. Jika dianalisa lebih dekat, sistem diakses secara bersamaan. Lalu, sistem mulai mengeksekusi juga hampir secara bersamaan (ada yang berbeda tetapi sedikit,

tergantung kondisi). Letak kata dalam database juga ikut mempengaruhi kecepatan eksekusi. Karena kecepatan pemrosesan user berbeda-beda sehingga sistem selesai dengan waktu yang berbeda.

4.4.4. Analisa Perbandingan Pengujian

Dalam analisa perbandingan pengujian ini, akan dibandingkan keterkaitan pengaruh masing-masing pengujian terhadap kecepatan sistem. Pada dasarnya, pengujian 1 menguji pengaruh perbedaan kata yang dicari terhadap kecepatan sistem. Sedangkan pengujian 2 menguji pengaruh banyaknya jumlah jawaban soal yang dicari terhadap kecepatan sistem. Terakhir yaitu pengujian 3 menguji pengaruh padatnya jaringan terhadap kecepatan sistem sehingga dalam analisa perbandingan ini yang akan dibandingkan, yaitu

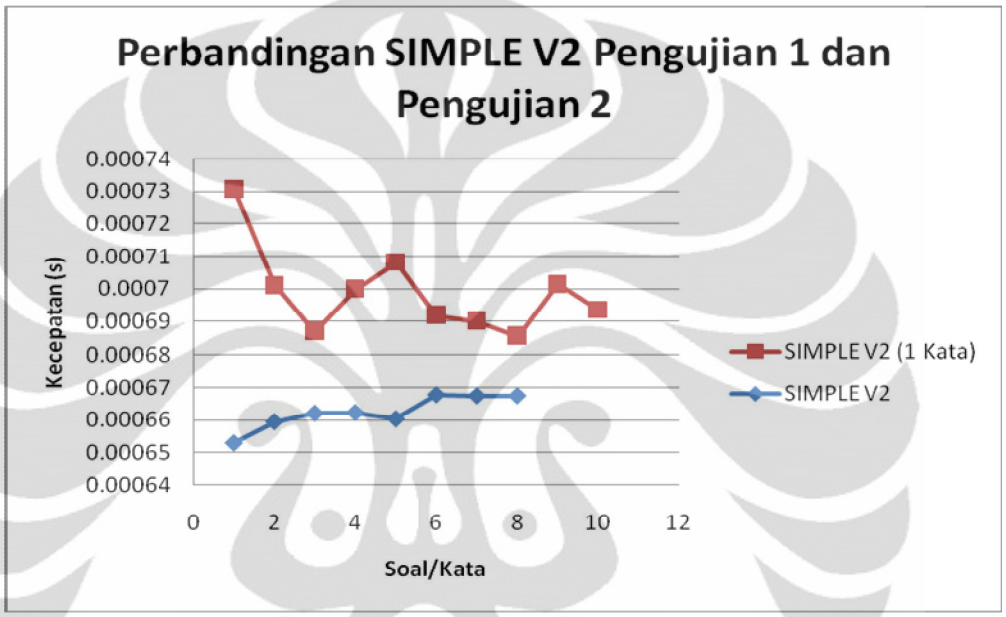
1. Membandingkan kecepatan sistem dalam pencarian 1 kata dengan pencarian 1 jawaban soal. Pada perbandingan ini, akan dilihat pengaruh perbandingan pada setiap sistem. Berikut adalah grafik perbandingan pada setiap sistem.



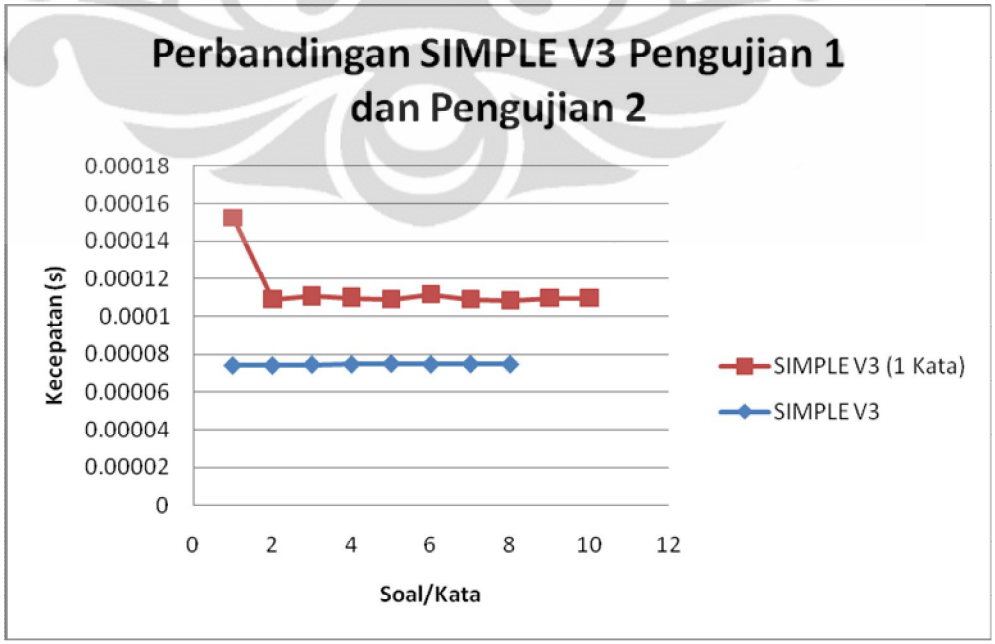
Gambar 4. 7 Grafik Perbandingan SIMPLE V1 Pengujian 1 dan Pengujian 2

Dari grafik pada Gambar 4. 7 terlihat bahwa pada SIMPLE V1, pengujian dengan 1 kata lebih cepat dibandingkan pengujian soal jawaban yang terdiri dari banyak kata. Kecepatan pengujian dengan 1 kata 1.12 kali lebih cepat

dibandingkan dengan kecepatan pengujian dengan banyak kata. Hal ini disebabkan oleh proses pencarian pada SIMPLE V1 yang harus memindahkan semua kata dan kode persamaan pada tabel persamaan kata untuk dimasukkan sebagai array di PHP. Semakin sedikit kata yang dipindahkan maka semakin cepat sistem bekerja. Jadi dapat dikatakan bahwa SIMPLE V1 tidak optimal untuk pencarian kata dalam jumlah banyak.

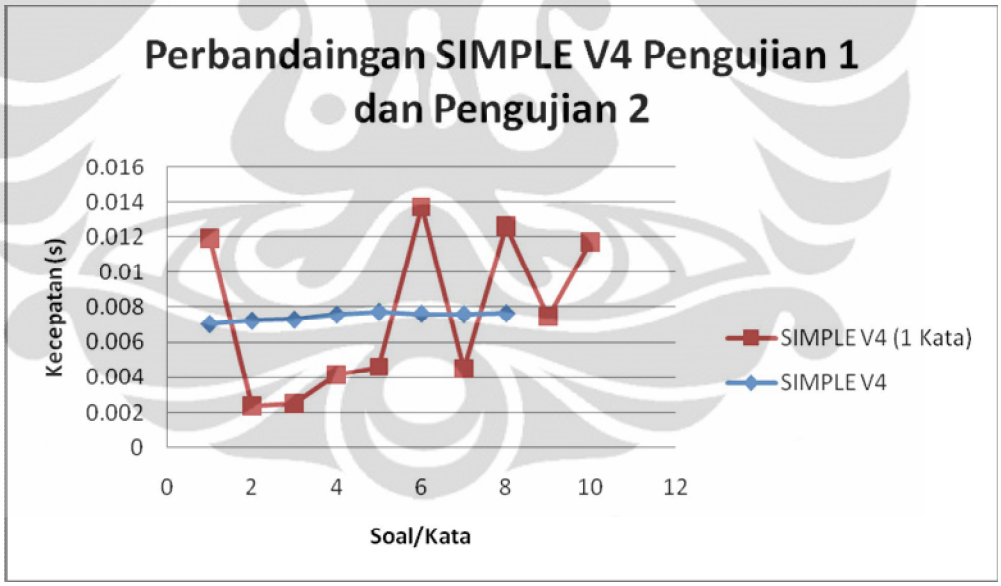


Gambar 4. 8 Grafik Perbandingan SIMPLE V2 Pengujian 1 dan Pengujian 2



Gambar 4. 9 Grafik Perbandingan SIMPLE V3 Pengujian 1 dan Pengujian 2

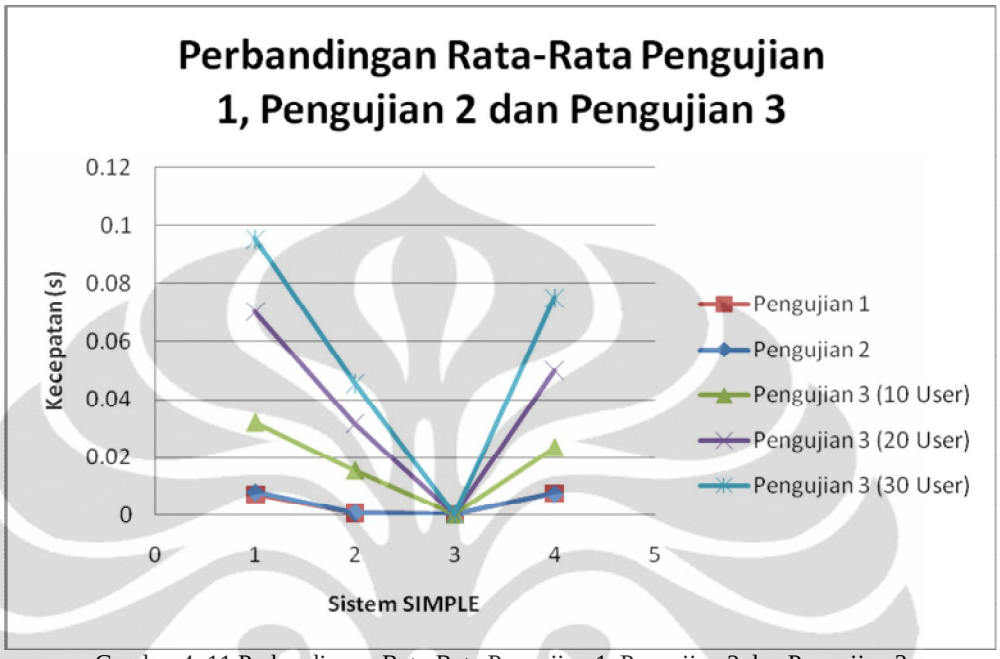
Hal yang berbeda terlihat pada Gambar 4.8 dan Gambar 4.9, dari grafik tergambar bahwa pengujian dengan banyak kata pada SIMPLE V2 dan SIMPLE V3 lebih cepat dibandingkan dengan pengujian yang hanya dengan 1 kata. Kecepatan SIMPLE V2 dalam pencarian banyak kata 1.05 kali lebih cepat jika dibandingkan dengan SIMPLE V2 pada pencarian 1 kata sedangkan SIMPLE V3 pada pencarian banyak kata 1.52 kali lebih cepat dibandingkan dengan pencarian 1 kata. Walaupun pada SIMPLE V2 seiring banyaknya jumlah semakin menurun kecepatannya, namun penurunan yang terjadi tidaklah signifikan dan kecepatannya pun masih diatas pengujian dengan 1 kata. SIMPLE V3 terlihat stabil dalam grafik. Dilihat dari Table 4.9 bahwa penurunan kecepatan yang terjadi sangat kecil sekali dan bahkan ada titik dimana semakin banyak kata, kecepatannya meningkat. Oleh karena itu, perubahan yang sangat kecil pada SIMPLE V3 tidak terlihat digrafik.



Gambar 4. 10 Grafik Perbandingan SIMPLE V4 Pengujian 1 dan Pengujian 2

Perbandingan SIMPLE V4 pada Gambar 4. 10, grafik yang tergambar tidak dapat menunjukkan perbandingan pengujian 1 dan pengujian 2 karena sistem ini sangat terpengaruh oleh letak kata pada database sehingga hasil dari pengujian 1 tidak dapat untuk membandingkan pengaruh kecepatan pada pencarian 1 kata dengan banyak kata pada pengujian 2.

2. Membandingkan kestabilan ketiga pengujian.



Gambar 4. 11 Perbandingan Rata-Rata Pengujian 1, Pengujian 2 dan Pengujian 3

Dari grafik pada Gambar 4. 11 terlihat bahwa sistem SIMPLE V3 merupakan sistem yang paling stabil dan paling cepat. Hal itu terlihat dari perbandingan setiap pengujian. Pada pengujian 3 dengan 10-30 user memadati jaringan atau manjalankan sistem secara bersama-sama, kecepatan sistem SIMPLE V3 tetaplah stabil dan cepat dan tidak mengalami penurunan kecepatan yang jauh. Berbeda dengan sistem lainnya yang kecepatannya akan menurun seiring dengan banyaknya orang yang mengakses sistem secara bersama-sama. Jika dihitung, penurunan kecepatan SIMPLE V4 antara pengujian 2 dan pengujian 3 hanya 0.0000193947 detik yaitu sekitar 1.17 kali lebih lambat. Sedangkan SIMPLE V1 penurunan sebesar 0.058900529 detik yaitu sekitar 9.32 kali lebih lambat. SIMPLE V2 mengalami penurunan yang paling tajam sebesar 0.030238901 detik yaitu sekitar 44.26 kali lebih lambat. SIMPLE V4 mengalami penurunan 0.041978478 detik yaitu sekitar 6.55 kali lebih lambat.

BAB 5
KESIMPULAN DAN SARAN

5.1. Kesimpulan

Dari perancangan yang telah dibuat, dapat ditarik beberapa kesimpulan :

1. Sistem pencarían kata yang baru dibuat yaitu SIMPLE V3 dan SIMPLE V4 berhasil diimplementasikan.
2. Dari perbandingan pencarían kata yang dilakukan, didapatkan bahwa SIMPLE V3 berhasil mempercepat sistem SIMPLE-O dan merupakan sistem yang paling cepat dan stabil diantara sistem SIMPLE lainnya. Sedangkan SIMPLE V1 merupakan sistem yang paling lambat.
3. SIMPLE V4 hanya berhasil mempercepat sistem melebihi SIMPLE V1, namun masih kalah cepat jika dibandingkan dengan SIMPLE V2 dan SIMPLE V3.
4. Pada pengujian 1, jika kecepatan keseluruhan sistem dibandingkan dengan SIMPLE V1 maka kecepatan SIMPLE V2 yaitu 9 kali lebih cepat, kecepatan SIMPLE V3 yaitu 62 kali lebih cepat. Kecepatan SIMPLE V4 tidak dapat dibandingkan karena SIMPLE V4 tidak stabil dalam pencarian 1 kata.
5. Pada pengujian 2, jika kecepatan keseluruhan sistem dibandingkan dengan SIMPLE V1 maka Kecepatan SIMPLE V2 yaitu 12 kali lebih cepat, Kecepatan SIMPLE V3 yaitu 106 kali lebih cepat dan kecepatan SIMPLE V4 yaitu 1.07 kali lebih cepat.
6. Pada SIMPLE V1 sistem akan lebih cepat mencari 1 kata dibandingkan banyak kata yaitu sebesar 1.12 kali lebih cepat. Sedangkan SIMPLE V2 dan SIMPLE V3 akan lebih cepat dalam pencarian banyak kata dibandingkan pencarian 1 kata yaitu sebesar 1.05 kali lebih cepat pada SIMPLE V2 dan 1.52 kali lebih cepat pada SIMPLE V3.

5.2. Saran

Hindari tahap preprocessing dalam proses pencarian kata, bahkan lebih baik tidak ada tahap preprocessing karena tahap preprocessing hanya akan memotong waktu pencarian kata yang tidak mempengaruhi performa pencarian sistem.

REFERENSI

[1] http://teachingacademy.wisc.edu/archive/Assistance/course/essay.htm, Akses pada 10 November 2010

[2] Anak Agung Putri Ratna, M Salman, Bagio Budiardjo, Djoko Hartando dan Seinosuke Narita. "SIMPLE: Sistem Penilaian Esei Otomatis Berbasis WEB dengan Metoda Latent Semantic Analysis Yang Digunakan Pada Bahasa Indonesia Dengan Penambahan Kata Bobot" Departemen Elektro, Fakultas Teknik, Universitas Indonesia, Depok, Indonesia.

[3] Anak Agung Putri Ratna, Bagio Budiardjo dan Djoko Hartanto, "SIMPLE: Sistem Penilaian Esei Otomatis untuk Menilai Ujian dalam bahasa Indonesia", Jurnal Makara Seri Teknologi, volume 11, April 2007, ISSN : 1693-6698

[4] Meirisal Dwi Waldi, "Pengembangan Sistem Pencarian Kata Pada SIMPLE-O", Skripsi, Depertemen Teknik Elektro, Fakultas Teknik, Universitas Indonesia, Depok, Juli 2010.

[5] Ellis B. Page, "The Imminence of Grading Essays by Computer," Phi Delta Kappan, January 1966, pp. 238-43.

[6] Page, Ellis B., and Nancy S. Petersen. "The computer moves into essay grading: updating the ancient test." *Phi Delta Kappan* 76.7 (1995): 561+. *Gale Arts, Humanities and Education Standard Package*. Web. 12 Dec. 2010.

[7] Valenti, Salvatore, Francesca Neri, and Alessandro Cucchiarelli. "An overview of current research on automated essay grading.(Report)." *Journal of Information Technology Education* 2 (2003): 319+. *Gale Arts, Humanities and Education Standard Package*. Web. 12 Dec. 2010.

[8] Landauer, T. K., Foltz, P. W., & Laham, D. (1998). Introduction to Latent Semantic Analysis. *Discourse Processes,* **25**, 259-284.

[9] diah.staff.gunadarma.ac.id/.../files/.../LINGKUNGAN+DATABASE.doc

[10] Arianto and Hidayat. Rahmat, "Seri 01 : Belajar Database MySQL"

[11] Feuerstein,Steven.,& Harrison,Guy. (2006). *MySQL Stored Procedure Programming.* O'Reilly Media, Inc.

[12] Fernando, Hary. "Perbandingan dan Pengujian Beberapa Algoritma Pencocokan String". Program Studi Teknik Informatika, Institut Teknologi Bandung.

[13]http://en.wikipedia.org/wiki/Rabin%E2%80%93Karp_string_searchalgorithm akses pada 27 Desember 2010

[14] Baedlowi, Najib. Adam, Deka Aditia. "String Matching Menggunakan Algoritma Rabin Karp". Departemen Teknik Informatika, Institut Teknologi Bandung

[15] Lecroq, Thierry Charras, Christian. 2001. Handbook of Exact String Matching Algorithm.

[16] Masido, Nurio Juliandatu. "Pengembangan DFA Dalam Algoritma Pencarian String", Sekolah Teknik Elektro dan Informatika ITB

Hasil Pengujian 3

- **10 User**

Tabel L1. 1 Hasil Pengujian 3 SIMPLE V1 dengan 10 User

Time	Sistem	User	Dispatch	Start	Timestamp
0.030125386	SIMPLE V1	5	1307375045	1307375045	22:44:15
0.03031624	SIMPLE V1	9	1307375045	1307375045	22:44:15
0.03097515	SIMPLE V1	3	1307375045	1307375045	22:44:15
0.032761693	SIMPLE V1	0	1307375045	1307375045	22:44:16
0.032447321	SIMPLE V1	6	1307375045	1307375045	22:44:16
0.032495027	SIMPLE V1	7	1307375045	1307375045	22:44:16
0.03281731	SIMPLE V1	2	1307375045	1307375045	22:44:16
0.032849789	SIMPLE V1	4	1307375045	1307375045	22:44:16
0.033304874	SIMPLE V1	1	1307375045	1307375045	22:44:16
0.03336644	SIMPLE V1	8	1307375045	1307375045	22:44:16

Tabel L1. 2 Hasil Pengujian 3 SIMPLE V2 dengan 10 User

Time	Sistem	User	Dispatch	Start	Timestamp
0.014298382	SIMPLE V2	3	1307374925	1307374925	22:42:09
0.014422674	SIMPLE V2	1	1307374925	1307374925	22:42:09
0.015522208	SIMPLE V2	2	1307374925	1307374925	22:42:10
0.015332872	SIMPLE V2	7	1307374925	1307374925	22:42:10
0.015721558	SIMPLE V2	0	1307374925	1307374925	22:42:10
0.015362288	SIMPLE V2	8	1307374925	1307374925	22:42:10
0.016061247	SIMPLE V2	5	1307374925	1307374925	22:42:10
0.016038373	SIMPLE V2	6	1307374925	1307374925	22:42:10
0.015907881	SIMPLE V2	9	1307374925	1307374925	22:42:10
0.016541539	SIMPLE V2	4	1307374925	1307374925	22:42:10

Tabel L1. 3 Hasil Pengujian 3 SIMPLE V3 dengan 10 User

Time	Sistem	User	Dispatch	Start	Timestamp
9.9222445e-05	SIMPLE V3	1	1307374985	1307374985	22:43:05
0.000108693	SIMPLE V3	0	1307374985	1307374985	22:43:05
9.1207271e-05	SIMPLE V3	3	1307374985	1307374985	22:43:05

0.000142291	SIMPLE V3	2	1307374985	1307374985	22:43:05
8.6153214e-05	SIMPLE V3	4	1307374985	1307374985	22:43:05
0.000121908	SIMPLE V3	6	1307374985	1307374985	22:43:05
0.000102174	SIMPLE V3	7	1307374985	1307374985	22:43:05
0.000165062	SIMPLE V3	5	1307374985	1307374985	22:43:05
7.0007480e-05	SIMPLE V3	9	1307374985	1307374985	22:43:05
0.000135883	SIMPLE V3	8	1307374985	1307374985	22:43:05

Tabel L1. 4 Hasil Pengujian 3 SIMPLE V4 dengan 10 User

Time	Sistem	User	Dispatch	Start	Timestamp
0.023066453	SIMPLE V4	3	1307374865	1307374875	22:41:22
0.023381408	SIMPLE V4	5	1307374865	1307374875	22:41:22
0.023443866	SIMPLE V4	4	1307374865	1307374875	22:41:22
0.023455923	SIMPLE V4	2	1307374865	1307374875	22:41:22
0.02355975	SIMPLE V4	7	1307374865	1307374875	22:41:23
0.023595641	SIMPLE V4	6	1307374865	1307374875	22:41:23
0.023757893	SIMPLE V4	9	1307374865	1307374875	22:41:23
0.023759015	SIMPLE V4	1	1307374865	1307374875	22:41:23
0.023942732	SIMPLE V4	0	1307374865	1307374875	22:41:23
0.023994751	SIMPLE V4	8	1307374865	1307374875	22:41:23

- **20 User**

Tabel L1. 5 Hasil Pengujian 3 SIMPLE V1 dengan 20 User

Time	Sistem	User	Dispatch	Start	Timestamp
0.069510924	SIMPLE V1	6	1307376036	1307376036	23:00:59
0.069518826	SIMPLE V1	0	1307376036	1307376036	23:00:59
0.06991362	SIMPLE V1	1	1307376036	1307376036	23:00:59
0.070050598	SIMPLE V1	4	1307376036	1307376036	23:00:59
0.070363611	SIMPLE V1	3	1307376036	1307376036	23:00:59
0.070314699	SIMPLE V1	10	1307376036	1307376036	23:01:00
0.070522039	SIMPLE V1	8	1307376036	1307376036	23:01:00
0.070615895	SIMPLE V1	7	1307376036	1307376036	23:01:00
0.070926414	SIMPLE V1	5	1307376036	1307376036	23:01:00
0.069913784	SIMPLE V1	15	1307376036	1307376036	23:01:00
0.070323161	SIMPLE V1	11	1307376036	1307376036	23:01:00
0.070244544	SIMPLE V1	13	1307376036	1307376036	23:01:00

0.070790343	SIMPLE V1	9	1307376036	1307376036	23:01:00
0.069888371	SIMPLE V1	17	1307376036	1307376036	23:01:00
0.069980651	SIMPLE V1	18	1307376036	1307376036	23:01:00
0.069788589	SIMPLE V1	19	1307376036	1307376036	23:01:00
0.070275992	SIMPLE V1	14	1307376036	1307376036	23:01:00
0.071398739	SIMPLE V1	2	1307376036	1307376036	23:01:00
0.070334047	SIMPLE V1	16	1307376036	1307376036	23:01:00
0.070716512	SIMPLE V1	12	1307376036	1307376036	23:01:00

Tabel L1. 6 Hasil Pengujian 3 SIMPLE V2 dengan 20 User

Time	Sistem	User	Dispatch	Start	Timestamp
0.031387581	SIMPLE V2	0	1307375915	1307375915	22:58:45
0.031426081	SIMPLE V2	4	1307375915	1307375915	22:58:45
0.031591139	SIMPLE V2	5	1307375915	1307375915	22:58:45
0.031735065	SIMPLE V2	3	1307375915	1307375915	22:58:45
0.032077365	SIMPLE V2	1	1307375915	1307375915	22:58:46
0.031644997	SIMPLE V2	11	1307375915	1307375915	22:58:46
0.032007516	SIMPLE V2	7	1307375915	1307375915	22:58:46
0.031783619	SIMPLE V2	9	1307375915	1307375915	22:58:46
0.032235351	SIMPLE V2	2	1307375915	1307375915	22:58:46
0.032124441	SIMPLE V2	6	1307375915	1307375915	22:58:46
0.031924942	SIMPLE V2	8	1307375915	1307375915	22:58:46
0.03172305	SIMPLE V2	12	1307375915	1307375915	22:58:46
0.031807054	SIMPLE V2	10	1307375915	1307375915	22:58:46
0.031529581	SIMPLE V2	14	1307375915	1307375915	22:58:46
0.031502469	SIMPLE V2	15	1307375915	1307375915	22:58:46
0.031272248	SIMPLE V2	17	1307375915	1307375916	22:58:46
0.031456331	SIMPLE V2	16	1307375915	1307375915	22:58:46
0.031829856	SIMPLE V2	13	1307375915	1307375915	22:58:46
0.03122652	SIMPLE V2	18	1307375915	1307375916	22:58:46
0.031176701	SIMPLE V2	19	1307375915	1307375916	22:58:46

Tabel L1. 7 Hasil Pengujian 3 SIMPLE V3 dengan 20 User

Time	Sistem	User	Dispatch	Start	Timestamp
9.916614e-05	SIMPLE V3	1	1307375976	1307375976	22:59:35
0.000138231	SIMPLE V3	2	1307375976	1307375976	22:59:35
0.000166814	SIMPLE V3	0	1307375976	1307375976	22:59:35
0.000170935	SIMPLE V3	3	1307375976	1307375976	22:59:35

9.9935615e-05	SIMPLE V3	4	1307375976	1307375976	22:59:35
8.409850e-05	SIMPLE V3	6	1307375976	1307375976	22:59:35
8.7507265e-05	SIMPLE V3	7	1307375976	1307375976	22:59:35
9.5915864e-05	SIMPLE V3	8	1307375976	1307375976	22:59:35
0.000156978	SIMPLE V3	5	1307375976	1307375976	22:59:35
8.7751244e-05	SIMPLE V3	9	1307375976	1307375976	22:59:35
9.3813193e-05	SIMPLE V3	11	1307375976	1307375976	22:59:35
0.000118509	SIMPLE V3	10	1307375976	1307375976	22:59:35
9.923774e-05	SIMPLE V3	12	1307375976	1307375976	22:59:35
9.4602129e-05	SIMPLE V3	13	1307375976	1307375976	22:59:35
7.6278653e-05	SIMPLE V3	14	1307375976	1307375976	22:59:35
7.3229954e-05	SIMPLE V3	15	1307375976	1307375976	22:59:35
0.000119378	SIMPLE V3	16	1307375976	1307375976	22:59:35
8.0246967e-05	SIMPLE V3	17	1307375976	1307375976	22:59:35
7.2591854e-05	SIMPLE V3	18	1307375976	1307375976	22:59:35
7.0828391e-05	SIMPLE V3	19	1307375976	1307375976	22:59:35

Tabel L1. 8 Hasil Pengujian 3 SIMPLE V4 dengan 20 User

Time	Sistem	User	Dispatch	Start	Timestamp
0.048977658	SIMPLE V4	9	1307375855	1307375865	22:58:01
0.049044478	SIMPLE V4	10	1307375855	1307375865	22:58:01
0.049349129	SIMPLE V4	14	1307375855	1307375865	22:58:01
0.049571213	SIMPLE V4	15	1307375855	1307375865	22:58:01
0.049716364	SIMPLE V4	12	1307375855	1307375865	22:58:01
0.049883079	SIMPLE V4	2	1307375855	1307375865	22:58:02
0.049987226	SIMPLE V4	0	1307375855	1307375865	22:58:02
0.050019797	SIMPLE V4	3	1307375855	1307375865	22:58:02
0.05002685	SIMPLE V4	11	1307375855	1307375865	22:58:02
0.050049217	SIMPLE V4	7	1307375855	1307375865	22:58:02
0.050058381	SIMPLE V4	17	1307375855	1307375865	22:58:02
0.050079757	SIMPLE V4	18	1307375855	1307375865	22:58:02
0.050094728	SIMPLE V4	4	1307375855	1307375865	22:58:02
0.050097633	SIMPLE V4	13	1307375855	1307375865	22:58:02
0.050164236	SIMPLE V4	1	1307375855	1307375865	22:58:02
0.050211323	SIMPLE V4	6	1307375855	1307375865	22:58:02
0.050247196	SIMPLE V4	8	1307375855	1307375865	22:58:02
0.050255013	SIMPLE V4	19	1307375855	1307375865	22:58:02
0.050263499	SIMPLE V4	5	1307375855	1307375865	22:58:02
0.050294065	SIMPLE V4	16	1307375855	1307375865	22:58:02

- **30 User**

Tabel L1. 9 Hasil Pengujian 3 SIMPLE V1 dengan 30 User

Time	Sistem	User	Dispatch	Start	Timestamp
0.091050587	SIMPLE V1	8	1307376480	1307376480	23:08:30
0.091636112	SIMPLE V1	4	1307376480	1307376480	23:08:30
0.091259771	SIMPLE V1	11	1307376480	1307376480	23:08:31
0.091644362	SIMPLE V1	13	1307376480	1307376480	23:08:31
0.092654271	SIMPLE V1	6	1307376480	1307376480	23:08:31
0.091810694	SIMPLE V1	18	1307376480	1307376480	23:08:31
0.093607756	SIMPLE V1	3	1307376480	1307376480	23:08:31
0.090794346	SIMPLE V1	27	1307376480	1307376481	23:08:31
0.093106077	SIMPLE V1	10	1307376480	1307376480	23:08:31
0.092280685	SIMPLE V1	21	1307376480	1307376480	23:08:31
0.093610185	SIMPLE V1	14	1307376480	1307376480	23:08:32
0.093217562	SIMPLE V1	22	1307376480	1307376480	23:08:32
0.093975384	SIMPLE V1	17	1307376480	1307376480	23:08:32
0.093233866	SIMPLE V1	23	1307376480	1307376480	23:08:32
0.098428805	SIMPLE V1	1	1307376480	1307376480	23:08:33
0.098549424	SIMPLE V1	2	1307376480	1307376480	23:08:33
0.09857	SIMPLE V1	5	1307376480	1307376480	23:08:33
0.098983485	SIMPLE V1	0	1307376480	1307376480	23:08:33
0.098499951	SIMPLE V1	9	1307376480	1307376480	23:08:33
0.098866845	SIMPLE V1	12	1307376480	1307376480	23:08:33
0.097930538	SIMPLE V1	20	1307376480	1307376480	23:08:33
0.097136329	SIMPLE V1	26	1307376480	1307376480	23:08:33
0.099508612	SIMPLE V1	7	1307376480	1307376480	23:08:33
0.097162429	SIMPLE V1	25	1307376480	1307376480	23:08:33
0.098262862	SIMPLE V1	19	1307376480	1307376480	23:08:33
0.098765126	SIMPLE V1	15	1307376480	1307376480	23:08:33
0.098637682	SIMPLE V1	16	1307376480	1307376480	23:08:33
0.097079489	SIMPLE V1	28	1307376480	1307376481	23:08:33
0.097841061	SIMPLE V1	24	1307376480	1307376480	23:08:33
0.097235798	SIMPLE V1	29	1307376480	1307376481	23:08:33

Tabel L1. 10 Hasil Pengujian 3 SIMPLE V2 dengan 30 User

Time	Sistem	User	Dispatch	Start	Timestamp
0.043367195	SIMPLE V2	4	1307376358	1307376358	23:06:13
0.043862239	SIMPLE V2	0	1307376358	1307376358	23:06:13
0.043873223	SIMPLE V2	3	1307376358	1307376358	23:06:13
0.043834107	SIMPLE V2	6	1307376358	1307376358	23:06:13
0.04396422	SIMPLE V2	9	1307376358	1307376358	23:06:13
0.043887907	SIMPLE V2	11	1307376358	1307376358	23:06:13
0.044208369	SIMPLE V2	12	1307376358	1307376358	23:06:13
0.04423453	SIMPLE V2	15	1307376358	1307376359	23:06:13
0.044413697	SIMPLE V2	14	1307376358	1307376359	23:06:13
0.04399281	SIMPLE V2	19	1307376359	1307376359	23:06:13
0.044105322	SIMPLE V2	20	1307376359	1307376359	23:06:13
0.043632612	SIMPLE V2	24	1307376359	1307376359	23:06:13
0.044185298	SIMPLE V2	22	1307376359	1307376359	23:06:14
0.044174062	SIMPLE V2	23	1307376359	1307376359	23:06:14
0.047406555	SIMPLE V2	1	1307376358	1307376358	23:06:14
0.04752289	SIMPLE V2	7	1307376358	1307376358	23:06:14
0.047473129	SIMPLE V2	8	1307376358	1307376358	23:06:14
0.047832467	SIMPLE V2	5	1307376358	1307376358	23:06:14
0.047990259	SIMPLE V2	2	1307376358	1307376358	23:06:14
0.047702689	SIMPLE V2	10	1307376358	1307376358	23:06:14
0.047516542	SIMPLE V2	13	1307376358	1307376359	23:06:14
0.047259252	SIMPLE V2	16	1307376358	1307376359	23:06:14
0.046681141	SIMPLE V2	21	1307376359	1307376359	23:06:14
0.045768225	SIMPLE V2	28	1307376359	1307376359	23:06:14
0.045980654	SIMPLE V2	27	1307376359	1307376359	23:06:14
0.047308292	SIMPLE V2	18	1307376359	1307376359	23:06:14
0.047469449	SIMPLE V2	17	1307376359	1307376359	23:06:14
0.045871518	SIMPLE V2	29	1307376359	1307376359	23:06:14
0.046565919	SIMPLE V2	25	1307376359	1307376359	23:06:14
0.046502425	SIMPLE V2	26	1307376359	1307376359	23:06:15

Tabel L1. 11 Hasil Pengujian 3 SIMPLE V3 dengan 30 User

Time	Sistem	User	Dispatch	Start	Timestamp
0.000121122	SIMPLE V3	0	1307376419	1307376419	23:06:59
0.00011923	SIMPLE V3	2	1307376419	1307376419	23:06:59
0.000129209	SIMPLE V3	3	1307376419	1307376419	23:06:59
0.000167325	SIMPLE V3	1	1307376419	1307376419	23:06:59
0.000120302	SIMPLE V3	4	1307376419	1307376419	23:06:59
0.000114725	SIMPLE V3	5	1307376419	1307376419	23:06:59
7.938852e-05	SIMPLE V3	7	1307376419	1307376419	23:06:59
0.000109865	SIMPLE V3	6	1307376419	1307376419	23:06:59
0.000117883	SIMPLE V3	8	1307376419	1307376419	23:06:59
8.925613e-05	SIMPLE V3	10	1307376419	1307376419	23:06:59
0.000140998	SIMPLE V3	9	1307376419	1307376419	23:06:59
0.000146876	SIMPLE V3	11	1307376419	1307376419	23:06:59
0.000110632	SIMPLE V3	12	1307376419	1307376419	23:06:59
7.530899e-05	SIMPLE V3	13	1307376419	1307376419	23:06:59
7.600270e-05	SIMPLE V3	15	1307376419	1307376419	23:06:59
7.34336e-05	SIMPLE V3	14	1307376419	1307376419	23:06:59
7.252582e-05	SIMPLE V3	16	1307376419	1307376419	23:06:59
7.463127e-05	SIMPLE V3	17	1307376419	1307376419	23:06:59
8.829620e-05	SIMPLE V3	18	1307376419	1307376419	23:06:59
9.999400e-05	SIMPLE V3	19	1307376419	1307376419	23:06:59
9.456807e-05	SIMPLE V3	20	1307376419	1307376419	23:06:59
7.296095e-05	SIMPLE V3	22	1307376419	1307376419	23:06:59
7.903055e-05	SIMPLE V3	21	1307376419	1307376419	23:06:59
8.307115e-05	SIMPLE V3	24	1307376419	1307376419	23:06:59
8.37676e-05	SIMPLE V3	23	1307376419	1307376419	23:06:59
0.000105864	SIMPLE V3	25	1307376419	1307376419	23:06:59
8.056254e-05	SIMPLE V3	26	1307376419	1307376420	23:06:59
8.717153e-05	SIMPLE V3	29	1307376420	1307376420	23:06:59
7.824717e-05	SIMPLE V3	27	1307376419	1307376420	23:06:59
7.444221e-05	SIMPLE V3	28	1307376419	1307376420	23:06:59

Tabel L1. 12 Hasil Pengujian 3 SIMPLE V4 dengan 30 User

Time	Sistem	User	Dispatch	Start	Timestamp
0.073888778	SIMPLE V4	17	1307376298	1307376308	23:05:33
0.074212818	SIMPLE V4	8	1307376298	1307376308	23:05:33
0.074412039	SIMPLE V4	23	1307376298	1307376308	23:05:33
0.074529637	SIMPLE V4	28	1307376298	1307376308	23:05:33
0.074643124	SIMPLE V4	22	1307376298	1307376308	23:05:33
0.074680082	SIMPLE V4	0	1307376298	1307376308	23:05:33
0.074711091	SIMPLE V4	16	1307376298	1307376308	23:05:33
0.074880067	SIMPLE V4	4	1307376298	1307376308	23:05:33
0.074855345	SIMPLE V4	10	1307376298	1307376308	23:05:33
0.074957553	SIMPLE V4	6	1307376298	1307376308	23:05:33
0.074989861	SIMPLE V4	5	1307376298	1307376308	23:05:33
0.075012033	SIMPLE V4	14	1307376298	1307376308	23:05:33
0.0750433	SIMPLE V4	12	1307376298	1307376308	23:05:33
0.075061751	SIMPLE V4	18	1307376298	1307376308	23:05:33
0.075141185	SIMPLE V4	11	1307376298	1307376308	23:05:33
0.075192139	SIMPLE V4	2	1307376298	1307376308	23:05:33
0.075218236	SIMPLE V4	29	1307376298	1307376308	23:05:33
0.075254524	SIMPLE V4	27	1307376298	1307376308	23:05:33
0.075316762	SIMPLE V4	7	1307376298	1307376308	23:05:33
0.075327182	SIMPLE V4	24	1307376298	1307376308	23:05:33
0.075376313	SIMPLE V4	25	1307376298	1307376308	23:05:33
0.075383865	SIMPLE V4	21	1307376298	1307376308	23:05:33
0.075469049	SIMPLE V4	9	1307376298	1307376308	23:05:33
0.075481787	SIMPLE V4	15	1307376298	1307376308	23:05:33
0.075524537	SIMPLE V4	26	1307376298	1307376308	23:05:33
0.07549914	SIMPLE V4	20	1307376298	1307376308	23:05:33
0.075538145	SIMPLE V4	13	1307376298	1307376308	23:05:33
0.075586283	SIMPLE V4	3	1307376298	1307376308	23:05:33
0.075595192	SIMPLE V4	19	1307376298	1307376308	23:05:33
0.075642235	SIMPLE V4	1	1307376298	1307376308	23:05:33