TUGAS AKHIR – KS 141501

IMPLEMENTASI PRINSIP HEURISTIK UNTUK REKOMENDASI INDEX MENGGUNAKAN JAVA PADA DATABASE MARIADB

PRINCIPAL HEURISTIC IMPLEMENTATION FOR INDEX RECOMMENDATION USING JAVA ON MARIADB DATABASE

RISA PERDANA SUJANAWATI
NRP 5213 100 063

Dosen Pembimbing I :
Radityo Prasetianto W. S.kom, M.Kom

Dosen Pembimbing II :
Faizal Mahananto S.kom, M.eng, Ph.D

DEPARTEMEN SISTEM INFORMASI
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2017

TUGAS AKHIR – KS 141501

IMPLEMENTASI PRINSIP HEURISTIK UNTUK REKOMENDASI INDEX MENGGUNAKAN JAVA PADA DATABASE MARIADB

RISA PERDANA SUJANAWATI
NRP 5213 100 063

Dosen Pembimbing I :
Radityo Prasetianto W. S.kom, M.Kom

Dosen Pembimbing II :
Faizal Mahananto S.kom, M.eng, Ph.D

DEPARTEMEN SISTEM INFORMASI
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2017

Halaman ini sengaja dikosongkan.

TUGAS AKHIR – KS 141501

PRINCIPAL HEURISTIC IMPLEMENTATION FOR INDEX RECOMMENDATION USING JAVA ON MARIADB DATABASE

RISA PERDANA SUJANAWATI
NRP 5213 100 063

Supervisor I :
Radityo Prasetianto W. S.kom, M.Kom

Supervisor II :
Faizal Mahananto S.kom, M.eng, Ph.D

DEPARTMENT OF INFORMATION SYSTEM
Faculty of Information Technology
Sepuluh Nopember Institute of Technology
Surabaya 2017

Halaman ini sengaja dikosongkan.

IMPLEMENTASI PRINSIP HEURISTIK UNTUK REKOMENDASI INDEX MENGGUNAKAN JAVA PADA DATABASE MARIADB

TUGAS AKHIR

Disusun Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Departemen Sistem Informasi
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember

Oleh:

RISA PERDANA SUJANAWATI
5213 100 063

Surabaya, 2017

KEPALA
DEPARTEMEN SISTEM INFORMASI

Dr. Ir. Aris Tjahyanto, M. Kom.
NIP 19650310199102001

Halaman ini sengaja dikosongkan.

LEMBAR PERSETUJUAN

IMPLEMENTASI PRINSIP HEURISTIK UNTUK REKOMENDASI INDEX MENGGUNAKAN JAVA PADA DATABASE MARIADB

TUGAS AKHIR

Disusun Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Jurusan Sistem Informasi
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Oleh :

RISA PERDANA SUJANAWATI
5213 100 063

Disetujui Tim Penguji : Tanggal Ujian : 7 Juli 2017
Periode Wisuda: September 2017

Radityo Prasetianto W. S.kom, M.Kom	**(Pembimbing I)**
Faizal Mahananto S.kom, M.eng, Ph.D	**(Pembimbing II)**
Faizal Johan Atletiko, S.Kom., M.T.	**(Penguji I)**
Nur Aini R., S.Kom., M.Sc.Eng., Ph.D	**(Penguji II)**

Halaman ini sengaja dikosongkan.

IMPLEMENTASI PRINSIP HEURISTIK UNTUK REKOMENDASI INDEX MENGGUNAKAN JAVA PADA DATABASE MARIADB

Nama Mahasiswa : Risa Perdana Sujanawati
NRP : 5213 100 063
Jurusan : Sistem Informasi FTIf-ITS
Pembimbing I : Radityo Prasetianto Wibowo S.kom, M.Kom
Pembimbing II : Faizal Mahananto S.kom, M.eng, Ph.D

ABSTRAK

Hampir seluruh bidang dalam kehidupan saat ini telah menggunakan database dan menuntut kinerja yang paling optimal sebuah database. Oleh sebab itu sebuah database harus memiliki kemampuan optimalisasi kinerja yang bisa didapatkan salah satunya dengan pemilihan index yang optima karena index memainkan peran penting dalam menentukan kinerja query yang dieksekusi terhadap system. Index yang tepat dapat meningkatkan kecepatan dalam pemrosesan data. Melakukan index secara keseluruhan juga akan memakan banyak penyimpanan padahal tidak semua index memberikan kinerja yang optimal bagi database. Menurut penelitian terdahulu Index dapat meningkatkan kinerja database yaitu jumlah transaksi permenit yang meningkat hingga tiga kali dan cost yang berkurang sekitar 50% hingga 90%. Dengan begitu pemilihan index merupakan dilema tersendiri bagi pengguna database.

Telah dilakukan beberapa penelitian pada produk terkait rekomendasi index dengan objek berbeda seperti untuk database PostgreSQL, MongoDB, dan SQL Server. Penilitian tersebut telah membuktikan dengan menerapkan metode heuristik untuk pemilihan index telah berhasil meningkatkan kinerja database.

Luaran dari penelitian berikut ini ialah sebuah perangkat lunak berbasis Java yang dapat digunakan pada Database MariaDB untuk mendapatkan rekomendasi index yang telah dipilih berdasarkan prinsip heuristik yang telah digunakan oleh peneliti sebelumnya. Perangkat lunak berikut dapat mengevaluasi kinerja database berdasarkan responsetime dan Queries per Second atau QPS. Hasil rekomendasi dari penelitian berikut dapat meningkatkan kinerja database Monitordom hingga 43,65% dan database Adventureworks2012 hingga 2.77%.

Keyword : Index, Database performance, Java, MariaDB.

PRINCIPAL HEURISTIC IMPLEMENTATION FOR INDEX RECOMMENDATION USING JAVA ON MARIADB DATABASE

Nama Mahasiswa	**: Risa Perdana Sujanawati**
NRP	**: 5213 100 063**
Jurusan	**: Sistem Informasi FTIf-ITS**
Pembimbing I	**: Radityo Prasetianto Wibowo S.kom, M.Kom**
Pembimbing II	**: Faizal Mahananto S.kom, M.eng, Ph.D**

ABSTRACT

Almost all areas of life today have used the database and require the most optimal performance of a database. Therefore a database must have a performance optimization capability that can be obtained one of them with the optimal index selection because the index plays an important role in determining the performance of queries that are executed against the system. The right index can increase the speed in data processing. Doing the whole index will also take up a lot of storage when not all the indexes provide optimal performance for the database. According to previous research, Index can improve database performance that is the number of transactions per minute which increased up to three times and cost reduced 50% to 90%. Thus the selection of index is a dilemma for database users.

There have been several studies on product-related index recommendations with different objects such as PostgreSQL, MongoDB, and SQL Server databases. The research has proven by applying the heuristic method for index selection has succeeded in improving database performance.

The output of the following research is a Java-based software that can be used in the MariaDB Database to obtain recommendation indexes that have been selected based on heuristic principles that have been used by previous researchers. The following software can evaluate database performance based on responsetime and Queries per Second or QPS. The results of the recommendations of this research can improve the performance database Monitordom up to 43.65% and database Adventureworks2012 up to 2,77%.

Keyword : Index, Database performance, Java, MariaDB.

KATA PENGANTAR

Segala puji dan syukur kepada Allah SWT yang telah melimpahkan rahmat-Nya sehingga penulis dapat menyelesaikan buku tugas akhir dengan judul "Implementasi Prinsip Heuristik untuk Rekomendasi Index menggunakan Java pada Database MariaDB " yang merupakan salah satu syarat kelulusan pada Jurusan Sistem Informasi, Fakultas Teknologi Informasi, Institut Teknologi Sepuluh Nopember Surabaya.

Penulis berharap agar buku ini dapat memberikan manfaat bagi seluruh pembaca terutama pada bidang ilmu pengetahuan yang semakin berkembang dengan cepat.

Selama pelaksanaan dan penulisan Tugas Akhir berikut ini penulis tidak akan dapat menyelesaikan tanpa adanya dukungan dari berbagai pihak. Tanpa mengurangi rasa hormat penulis ingin mengucapkan terimakasih kepada seluruh pihak yang terlibat dalam pengerjaan Tugas Akhir ini:

- Kepada Ayah dan Ibu penulis yang selalu memberikan dukungan dan doanya sehingga penulis dapat menyelesaikan pendidikan S1 dan juga kedua adik yang selalu menjadi semangat penulis.
- Bapak Radityo Prasetianto W. S.kom, M.Kom dan Bapak Faizal Mahananto S.kom, M.eng, Ph.D selaku dosen pembimbing yang telah memberikan waktunya untuk memberi masukan dan membimbing penulisan tugas akhir.
- Bapak Bekti Cahyo Hidayanto, S.Si, M.Kom selaku dosen wali penulis yang telah membimbing dan memberi nasehat semenjak penulis menjadi mahasiswa di Jurusan Sistem Informasi.
- Bapak Faizal Johan Atletiko, S.Kom., M.T. dan Ibu Nur Aini Rakhmawati, S.Kom., M.Sc.Eng, Ph.D selaku dosen penguji yang telah memberikan kritik, saran, dan masukan yang berharga sehinga dapat menyempurnakan Tugas Akhir ini

- Valliant Ferlyando, selaku rekan penulis yang selalu mendukung dan bersedia memberika data hasil Tugas Akhirnya untuk menjadi salah satu data sampel pada penelitian ini.
- Teman-teman seperjuangan, Marina, Novi, Izzur, Stezar, Haikal, Harun, Bambang, Tetha, Wisnu, dan seluruh rekan ADDI yang telah banyak berbagi ilmu dan masukan kepada penulis
- Sahabat penulis Hemas, Mira, Ofi, Lily, Amal, Dita, Nena. dan Izza yang selalu bersedia memberikan waktunya bagi penulis semenjak semester satu.
- Teman-teman BELTRANIS dan seluruh rekan penulis yang selalu mendukung satu sama lain untuk menyelesaikan Tugas Akhir.

Penulis menyadari bahwa Tugas Akhir ini masih belum sempurna dan memiliki banyak kekurangan di dalamnya. Oleh karena itu, penulis merima saran dan kritik yang membangun guna kesempurnaan Tugas Akhir ini. Semoga Tugas Akhir ini bermanfaat bagi ilmu pengetahuan dan semua pihak.

Surabaya, 2017

Penulis

DAFTAR ISI

Halaman ini sengaja dikosongkan.

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR FORMULA

Halaman ini sengaja dikosongkan.

DAFTAR KODE

BAB I
PENDAHULUAN

Pada bab pendahuluan akan menjelaskan mengenai latar belakang penelitian, kemudian rumusan masalah dan batasan masalah yang menjelaskan ruang lingkup penelitian. Setelah memaparkan tujuan penelitian dan manfaat penelitian diharapkan dapat memberikan gambaran awal mengenai penelitian.

1.1 Latar Belakang

Database sekarang merupakan bagian integral dari kehidupan kita sehari-hari yang kita gunakan dengan tidak sadar [1]. Kinerja database sering menjadi pengukuran penting yang terkait dengan kepuasan pengguna. Kinerja sebuah database tergantung pada struktur logis, struktur fisik dan desain database tersebut. Sehingga semakin baik struktur logis, struktur fisik dan desain database akan semakin baik pula kinerja database tersebut [2]. Hampir seluruh bidang dalam kehidupan saat ini telah menggunakan database dan menuntut kinerja yang paling optimal sebuah database. Oleh sebab itu sebuah database harus memiliki kemampuan optimalisasi kinerja yang bisa didapatkan salah satunya dengan pemilihan index yang optimal.

Index adalah sebuah susunan teratur yang secara logis digunakan untuk mengurutkan baris di dalam sebuah tabel. [1] index memainkan peran penting dalam menentukan kinerja *query* yang dieksekusi terhadap sistem. [3]. Index yang tepat dapat meningkatkan kecepatan dalam pemrosesan data [4]. Sering terjadi salah pemahaman semakin banyak index akan semakin baik, dimana hal tersebut dapat menambah pemeliharaan database misalnya penyimpaan dan juga tidak memberikan hasil kinerja database yang optimal [5]. Semakin banyak index akan semakin banyak penyimpanan yang digunakan dan hal tersebut akan berdampak tidak baik baik pada kinerja database. Solusi untuk mengatasi hal tersebut ialah melakukan pemilihan index yang optimal seperti yang telah

dibuktikan pada sebuah kinerja database yang diukur dari kejadian transaksi per menit, dapat meningkat hingga tiga kali lipat dan dapat mengurangi *cost* dari sebuah *workload* 50% hingga 90% [6] [7], sehingga pemilihan index dengan sebuah metode tertentu perlu dilakukan untuk mendapatkan peningkatan kinerja yang optimal.

Terdapat beberapa metode pemilihan index menurut penelitian terdahulu salah satunya ialah dengan menggunakan algoritma yang dikembangkan berdasarkan metode heuristik dan juga penggunaan metode probabilitas naïve bayes didalam algoritma tersebut [8] [9] [3]. Hampir semua penelitian untuk rekomendasi index ialah menggunakan prinsip metode heuristik dengan mengembangkan algoritma yang berorientasi pada hasil optimal.

Beberapa DBMS telah menyediakan fitur bawaan rekomendasi index diantaranya SQL server [3]. Berbagai penelitian mengenai optimasi index dengan beberapa studikasus DBMS yang lain juga telah dikembangkan seperti AISIO untuk PostgreSQL 8.4 [6], penelitian index rekomendasi lainnya menjadikan MySQL 5.5 sebagai uji coba dan implementasi [10]. Bahkan tidak terkecuali untuk database NoSQL seperti MongoDB telah menjadi objek penelitian untuk rekomendasi index menggunakan algoritma *mining* [4]. Namun *open source* seperti Maria DB belum memiliki fitur bawaan rekomendasi index dan belum banyak penelitian mengenai rekomendasi dengan studi kasus MariaDB. Salah satu cara untuk pengatasi permasalahan pemilihan index pada MariaDB dapat dikembangkan dengan melakukan implementasi prinsip heuristik dengan menggunakan Java.

Java adalah Bahasa pemrograman yang berbasis objek [11]. Java juga bersifat *multiplatform* yang dapat diinterpretasikan untuk berbagai sistem operasi sehingga java akan digunakan oleh pengguna sistem berbagai operasi. Java mempunyai metode *pattern matcher* seperti *Regular Expression* yang dapat melakukan filter untuk SQL yang ingin dioptimasi.

Penelitian ini dimaksudkan untuk membuat sebuah perangkat lunak sebagai rekomendasi index berbasis java dengan mengimplementasikan metode heuristik yang memiliki algoritma pemilihan index seperti yang telah dilakukan peneliti sebelumnya [3] namun yang berbeda dalam penelitian berikut ini ialah dalam proses pemilihan index dengan memanfaatkan *General log* ataupun dapat berupa *query* sebagai input dan memberikan luaran berupa nama kolom yang direkomendasikan sebagai index sebagai hasil dari proses seleksi yang dilakukan pada database Maria DB.

1.2 Rumusan Masalah

Berdasarkan uraian latar belakang maka didapatkan fokus rumusan masalah dalam penelitian berikut yaitu:

1. Bagaimana pemilihan index yang optimal bagi sebuah database dengan menggunakan prinsip heuristik?
2. Bagaimana implementasi sistem pemilihan index yang optimal menggunakan java pada database MariaDB ?

1.3 Batasan Masalah

Adapun penelitian berikuti ini terbatas pada:

1. Pengukuran kinerja yang digunakan dalam penelitian ini terbatas pada server database MariaDB versi 10.1.19
2. Algoritma pemilihan index yang digunakan ialah algoritma Heuristik dengan menyertakan metode Naïve Bayes sebagai pemilihan *selectifity factor*.
3. Implementasi yang dilakukan menggunakan *General Log* dan *Query* sebagai input.
4. Evaluasi dan uji coba menggunakan penilaian berupa QPS dan *responsetime*.

5. Dalam pengujian tidak ada perubahan pada skema database maupun perubahan pada spesifikasi perangkat lunak yang digunakan.

6. Index yang direkomendasikan berjenis *non-clustered* dikarenakan *storage engine* dari MariaDB yang secara otomatis membuat *clustered* index baik itu pada Primary key atau secara kasat mata.

1.4 Tujuan Penelitian

Berdasarkan perumusan masalah yang ada maka tujuan penelitian berikut ini ialah:

1. Melakukan pemilihan index sebuah database dengan menggunakan prinsip heuristik.

2. Mengimplementasikan sistem pemilihan index yang optimal menggunakan java pada database MariaDB.

1.5 Manfaat Penelitian

Tugas Akhir berikut ini diharapkan dapat memberikan beberapa manfaat setidaknya untuk dua bidang yaitu akademik dan masyarakat umum yang menggunakan database MariaDB versi 10.1.16 sebagai berikut:

Akademik

Penelitian ini bermanfaat bagi akademisi yang ingin mendapatkan pengetahuan mengenai optimasi kinerja database. Penelitian ini juga dapat dikembangkan untuk database lain yang belum memiliki metode pemilihan index.

Masyarakat umum

Sedangkan untuk masyarakat umum pengguna database MariaDB, luaran dari penelitian ini dapat digunakan dalam membantu fase *physical design* sebuah database. Index yang direkomendasikan dapat membantu meningkatkan kinerja database.

1.6 Realevansi Penelitian

Penelitian yang termasuk dalam kategori database *performance* ini relevan dengan semakin banyaknya kebutuhan data yang tersimpan dalam database dan membuat kinerja sebuah database perlu diperhatikan, salah satunya dengan pemilihan index yang optimal. Penelitian ini juga relevan dengan laboratorium Akuisisi Data dan Diseminasi Informasi Jurusan Sistem Informasi Institut Teknologi Sepuluh Nopember.

Halaman ini sengaja dikosongkan.

BAB II
TINJAUAN PUSTAKA

Pada bab tinjauan pustaka akan menjelaskan beberapa penelitian terdahulu yang terkait dengan tugas akhir berikut ini dan beberapa teori terkait dengan tugas akhir berikut.

2.1 Penilitian Sebelumnya

Bagian ini akan menjelaskan penelitian sebelumnya yang terkait dengan Tugas Akhir berikut ini dan hasil yang diperoleh penulis dari penelitian tersebut yang dijelaskan pada tabel 2.1 penelitian sebelumnya berikut ini.

Tabel 2.1. Penelitian sebelumnya

No.1	
Judul	A Tool for Automatic Index Selection in Database Management System
Pengarang Tahun	Wendel Goes Pedrozo, Maria Salete Marcon Gomez Vaz, 2014 [6]
Metode yang digunakan	Metode pemilihan index menggunakan metode Heuristik
Hasil yang diperoleh	Penelitian ini membuat sebuah *tools* yang mengimplementasikan Heuristik yang digunakan dengan DBMS yang bernama Automatic Index Selection Integrated (AISIO) dengan arsitektur aplikasi sebagai berikut: 1. **Workload:** berupa pernyataan SQL dari input user. 2. **Statistic Collection:** pada tahap ini ialah *optimizer* mengambil data statistic dari workload. 3. **Heuristik:** pada tahap ini ialah pemilihan atribut sehingga menjadi kandidat index yang menguntungkan. Tahap tersebut memiliki beberapa langkah yaitu dengan algoritma sebagai berikut : i) *Scanning of the SQL statement to identify single-column candidate indxes that satisfy the following criteria:* • *Columns involving equal predicate;*

	• *Columns involved in clauses ORDER BY, GROUP BY, or join predicate;*. • *Columns appearing in interval restrictions;* • *Columns appearing in other predicat (for example, like);and* • *Other column composing the SQL statement.* ii) *Composition of single-column candidate* index*es. This composition exponentially reproduces the* index *combinations;* iii) *Formation of multiple column candidate* index*es;* iv) *Composition of the final list of candidate* index*es to be used. This is done through the simple union of multiple column and single-column* index*es;* 4. **Creation of hypotical indexes:** kandidat dari proses algoritma heuristik akan dihubungkan kepada parameter sehingga index berada pada skema database. 5. **Optimation Process:** memulai proses optimasi seperti pada prosedur optimasi pada DBMS 6. **Listing selected indexes and statictical data:** untuk mendata ulang index yang terpilih dan beberapa data statistic yang memungkinkan didata. 7. **Index recommendation and performance measure:** menampilkan rekomendasi index dan data statistik yang ada sebagai informasi untuk *user*. Pada implementasinya AISIO diintegrasikan dengan PostgreSQL 8.4, dan dievaluasi dengan perbandingan dari index dari database tersebut, index dari DBT-2 Toolkit, dan index dari AISIO. Dalam evaluasi menunjukkan dengan menggunakan index dari AISO jumlah transaksi permenit lebih banyak dari pada menggunakan index lainnya.
No.2	
Judul	An Efficient, Cost-Driven Index Selection Tool for Microsoft SQL Server
Pengarang Tahun	Surajit Chauduri, Vivek Narasayya, 1997 [9]

Metode yang digunakan	Metode yang digunakan dalam pemilihan index ialah menggunakan pendekatan *greedy* yang menggunakan metode Naïve *Enumeration algorithm.*
Hasil yang diperoleh	Tahapan untuk memilih index sendiri terdiri dari tiga tahap utama yaitu : **-candidate index selection** Pemilihan kandidat index dari *workload* yang ada sehingga mendapatkan set konfigurasi index yang diinginkan dengan jumlah multi index per tabel ialah 2. Setelahnya dilakukan evaluasi cost untuk memeilih kandidat index terbaik. **-configuration enumeration** Selanjutnya ialah melakukan pencacahan dengan pendekatan algoritma greedy dengan metode *naïve enumeration algorithm* **-multi-level-index generation** Selanjutnya ialah kembali dilakukan percobaan kombinasi index untuk mencari hasil yang lebih optimal dengan kemungkinan multi-level-index yang muncul ialah sebanyak k! dimana k ialah jumlah kolom yang muncul hingga memenuhi m dimana m ialah *storage*. Dan pencarian akan berhenti hingga memenuhi m.
No.3	
Judul	Index Selection for Databases: A Hardness Study and a Principle Heuristik Solution.
Pengarang Tahun	Surajit Chauduri, Mayur Datar, Vivek Narasayya, 2004 [3]
Metode yang digunakan	Penelitian ini menggunakan pengembangan metode Heuristik dengan menerapkan Linier programming dan knapsack problem.
Hasil yang diperoleh	Penelitian ini merupakan perkembangan dari penelitian tahun 1997 [9], dimana perkembangan ditekankan pada *Hardness Study* yaitu penentuan dua batasan yang ditentukan berupa:1) Storage Constraint. 2) Clustered index <=1. dalam implementasinya merupakan pengembangan dari penelitian sebelumnya dengan perbedaan pemilihan *clustered* dan *non-clustered* index. Namun dikarenakan objek penelitian yang berbeda dengan penelitian berikut, maka langkah

	pemilihan jenis *clustered non-clustered* berikut tidak dilakukan.
No.4	
Judul	On a Self-Tuning Index Recommendation Approach for Database
Pengarang Tahun	Prinaz Ameri, 2016 [12]
Metode yang digunakan	Pendekatan yang digunakan dalam penelitian ini yaitu sebuah framework yang bisa digunakan untuk membuat rekomendasi index berdasarkan workloads
Hasil yang diperoleh	Pada penelitian ini penulis merekomendasikan indeks dengan menggunakan *query* optimizer untuk *query* yang paling sering digunakan saja. • *Workload* akan dianalisis dengan menggunakan *Pattern recognizer* dan *syntax analyser* dan dipilih kandidat index dari *query* yang paling sering digunakan. Langkah berikutnya yaitu menentukan prioritas dari kandidat yang akan direkomendasikan sebagai index dengan berbagai factor pemilihan yaitu salah satunya dengan mencari atribut yang paling sering muncul dalam *workload*.

2.2 Dasar Teori

Pada bagian ini memaparkan mengenai teori-teori yang berkaitan dengan Tugas Akhir yang dilakukan oleh penulis.

2.2.1 Database

Sebuah database ialah sekumpulan data yang terhubung secara logis beserta deskripsinya, yang dirancang untuk memenuhi kebutuhan informasi dari sebuah organisasi [1]. Sedangkan untuk melakukan kontrol terhadap database pengguna dapat menggunakan DBMS yaitu Database Management System. [1]. Menurut Conolly, Database memiliki tiga fase desain yaitu:

a) Conceptual

Pada fase berikut ini ialah penentuan model data yang digunakan, target DBMS, program aplikasi yang digunakan, Bahasa pemrograman yang digunakan,

hingga hardware yang akan digunakan dan aspek fisik lain yang terkait.

b) Logical
Pada fase ini ialah fase pemetaan dari *conceptual design* menuju *physical design* salah satu contohnya ialah pembuatan model relational.

c) Physical
Sedangkan pada fase ini ialah dimana proses implementasi dari database tersebut termasuk pengaturan file dalam DBMS, hingga index yang digunakan. Pada fase berikut ini dimana desain dimaksudkan untuk meningkatkan kinerja dari database. Sehingga tools dari penelitian ini berada pada lingkup fase *Physical Design* dimana luaran dimaksudkan dapat membantu proses *physical design* database.

2.2.1.1 Database Performance

Pengukuran kinerja dari database dapat dilaukan dengan menilai beberapa aspek menurut winand yaitu sebagai berikut [5]:

a) Data Volume
Kinerja dari sebuah database dapat dipengaruhi oleh volume data yang ada, karene *query* akan semakin lambat dijalankan ketika volume data dari database tersebut meningkat.

b) System Load
System Load yang dimaksud ialah ketika terjadi terjadi Load(jumlah *query* yang dapat dijalankan) dalam respon time tertentu.

c) Response Time and Throughput
Tujuan dari melakukan index ialah mengurangi *response time* dari database namun, hal ini tidak bisa hanya diukur dari *response time* karena dengan

mengukur kombinasi dari *response time* dan *throughput* diharapkan dapat menjadi ukuran kinerja database yang baik. Sehingga dalam penelitian ini lebih ditekankan pada penilaian kinerja database menurut *response time* dan *throughput* dari contoh data sebagai validasi solusi yang ditawarkan. *Throughput* sendiri ialah jumlah *query* yang dibaca atau diubah dalam interval waktu tertentu misalnya dalam detik atau menit beberapa *tools* evaluasi kinerja database sendiri menggunakan QPS atau *query per second* sebagai *Throughput*.

2.2.2 Index

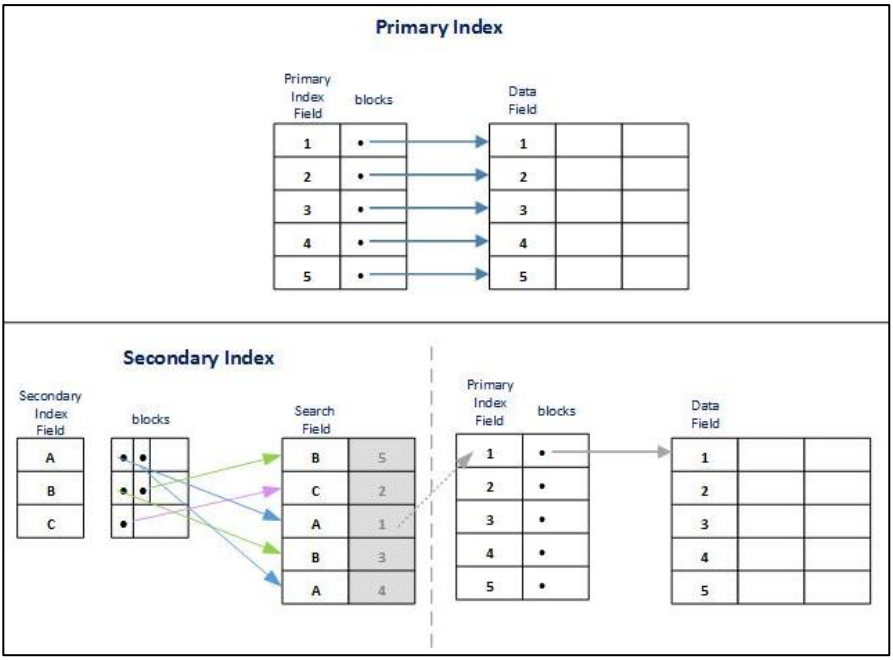
Index adalah sebuah struktur data yang mengatur record data pada penyimpanan untuk mengoptimalkan beberapa jenis operasi pengambilan atau *retrieval* tertentu [13]. Index yang tepat dapat meningkatkan kecepatan dalam pemrosesan data [4]. Berikut ini beberapa penjelasan mengenai index menurut Ramakhirsman dan Gehrke [13]:

- Urutan data pada index dapat dibedakan menjadi dua jenis index yaitu ***clustered* dan *non-clustered* index**, dimana *clustered* index ialah index yang mempunyai urutan entri data yang sama atau hampir sama dengan urutan record data. Sedangkan *non-clustered* index ialah index yang memiliki entri data yang tidak tergantung pada urutan dari record data.

- Struktur index juga dapat dibedakan menjadi **Single-level-index dan Multi-level-index.** Level yang dimaksudkan ialah level pencarian dalam sebuah index. Sebuah single-level-index akan mencari hanya pada satu kolom index karena hanya berada pada satu level yaitu level pertama, sedangkan multi-level-index ialah index dengan level pencarian lebih dari satu. Kolom index yang merupakan level kedua dalam jenis multi-level index akan merujuk kepada level pertama ketika pencarian pada level ke dua sudah ditemukan. Oleh karena itu sebuah urutan nama

kolom dalam multi-level-index sangat penting dan tidak dapat diabaikan.

- Terdapat dua tipe index yaitu **index primer dan index sekunder**. Index Primer ialah Sebuah index yang melibatkan field primary key merupakan index primer. Sedangkan index lain yang tidak melibatkan primary key merupakan index Sekunder. Dalam sebuah tabel, pencarian dari index sekunder digamabrkan seperti pada gambar 2.1 [5]. Sehingga risiko penggunaan index sekunder semakin banyak ialah semakin lama pencarian karena pencarian akan berjalan dari index sekunder ke index primer baru kemudian menuju data yang diinginkan.



Gambar 2.1. Primary dan Secondary index

2.2.3 MariaDB

MariaDB ialah sebuah database *open source* yang diciptakan oleh Monty Widenius pada tahun 2005 yang merupakan salah satu pencipta MySQL. Hal itu dikarenakan Oracle mengakuisisi *storage engine* InnoDB dari MySQL dan pada tahun 2009 Oracle mengakuisisi keseluruhan MySQL. Pada dasarnya MariaDB hampir sama dengan MySQL terlepas bahwa MariaDB dapat digunakan dengan gratis [14]. Pada psenelitian berikut ini alat optimasi index ialah spesifik untuk pengguna MariaDB versi 10.1.19. Perbedaan antara MariaDB dan MySQL sendiri tidak terlalu signifikan dan berikut ialah perbedaan antara MariaDB dan MySQL dalam beberapa aspek ialah [15]:

Storage Engine: Pada MariaDB memiliki lebih banyak *storage engine* dari pada MySQL yaitu Aria, XtraDB (sebagai pengganti InnoDB), FederatedX, SphinxSE, Cassandra, CONNECT, SEQUENCE, Spider, dan ColumnStore.

Speed Improvement: Dalam segi kecepatan MariaDB lebih mumpuni dibandingkan MySQL dengan beberapa keunggulan seperti:

- *Setup* untuk Replikasi dan pembaruan menjadi 2 kali lebih cepat.
- Indeks untuk MEMORY (HEAP) mesin lebih cepat.
- Dapat mempercepat tabel MyISAM dengan hingga 4 kali lebih cepat.
- CHECKSUM TABLE lebih cepat.

Fewer Warning and Bugs: Pada MariaDB telah dilakukan perbaikan pada *Bugs* dan *warning* tanpa menambah sebuah peringatan untuk *bug* dan *warning* baru yang berarti masalah tersebut telah terselesaikan.

Truly Opensource : MariaDB tidak memiliki modul tertutup seperti pada MySQL Enterprise Edition. Bahkan, semua fitur

sumber tertutup di MySQL 5.5 Enterprise Edition ditemukan di MariaDB versi open source. Bahan seperti *library* JDBC dan ODBC.

Beberapa kemudahan dan keterbukaan yang didapat pada MariaDB tersebut membuat MariaDB layak dijadikan objek penelitian ini.

2.2.4 Metode Pemilihan Index

Pemilihan index yang sering terjadi ialah berdasarkan prinsip heuristik. Heuristik yang berkaitan dengan pemecahan masalah adalah cara menujukan pemikiran seseorang dalam melakukan proses pemecahan sampai masalah tersebut berhasil dipecahkan [16]. Dalam penelitian ini metode heuristik ialah berperan dalam algoritma pemilihan index dari mulai input hingga menjadi rekomendasi index yang dipakai seperti pada penelitian sebelumnya [9] [3] [6] [8]. Algoritma heuristik pada penilitian ini mungkin akan sedikit dikembangkan atau berbeda dengan algoritma heuristik untuk pemilihan index yang telah diemukan oleh peneliti sebelumnya yaitu dalam melakukan proses pencacahan. Dasar dari algoritma dalam penelitian berikut ialah prinsip heuristik AISIO [6] dimana dalam pemilihan indexnya memiliki empat proses utama :

i) *Scanning of the SQL statement to identify single-column candidate indxes that satisfy the following criteria:*
   - *Columns involving equal predicate;*
   - *Columns involved in clauses ORDER BY, GROUP BY, or join predicate;.*
   - *Columns appearing in interval restrictions;*
   - *Columns appearing in other predicat (for example, like);and*
   - *Other column composing the SQL statement.*

ii) *Composition of single-column candidate* index*es. This composition exponentially reproduces the* index *combinations;*

iii) *Formation of multiple column candidate* index*es;*

iv) *Composition of the final list of candidate* index*es to be used. This is done through the simple union of multiple column and single-column* index*es;*

Alur algoritma heuristik diatas menjadi referensi untama dalam melakukan penelitian ini namun terdapat beberapa detail yang masih belum terlalu jelas seperti pada saat pembuatan kandidat single dan multi-level-index sehingga dapat menggunakan beberapa referensi penelitian lain seperti proses pemilihan index pada SQL server [9] yaitu dengan cara melakukan permutasi untuk nama kolom, dan mendapatkan heuristik yang sesuai dengan mariaDB dan diterapkan dalam penelitian berikut:

i) Parsing dan Filter Query.
Untuk mendapatkan awal kandidat index berupa nama kolom-kolom yang terdapat pada input seperti yang dilakukan semua penelitian mengenai rekomendasi index sebelumnya.

ii) Retrieving previous indexes.
Berikut ini merupakan tambahan dari penelitian ini karena dalam penelitian sebelumnya index yang sudah ada jarang diperhatikan padahal index tersebut bisa jadi merupakan index yang sudah optimal.

iii) Identifying index Configuration.
Setelah mendapat kandidat index berupa nama kolom selanjutnya ialah dilakukan pembuatan konfigurasi index atau berupa satu set index berdasarkan kombinasi nama kolom sehingga akan muncul multi-level-index seperti heuristik utama dan pembuatan konfigurasi ini menganut detail pembuatan set konfigurasi pada penelitian pada sql server [9] yang menemukan bahwa kombinasi optimal ialah dengan nilai j=2, kombinasi tersebut mementingkan urutan sehingga dilakukan dengan cara permutasi.

iv) Configuration cost evaluation
Cost evalution merupakan kegiatan evaluasi dengan pengukuran berupa QPS atau Responsetime yang dilakukan untuk mencari kinerja terbaik dari semua konfigurasi index sebelum dilakukan pencacahan seperti pada penelitian sebelumnya [9].

v) Configuration enumeration.
Setelah mendapat satu set konfigurasi index dengan kinerja terbaik maka dilakukan pencacahan seperti pada penelitian untuk sql server [9], namun pemilihan untuk kandidat hasil pencacahan terbaik bukan berdasarkan *storage* yang paling *fit* namun menggunakan faktor pemilihan yaitu dengan memilih kandidat yang paling sering muncul dalam *workload* berdasarkan penelitian lain [12]. Faktor pemilihan ini dalam referensi lain dianggap sebagai *selectivity factor* yang dihitung menggunakan probabilitas naïve bayes [8]. Hal ini disebabkan kurang relevannya sebuah *storage* dijadikan sebuah pengukuran karena sifatnya yang tidak statis.

vi) Generate recommended index query
Setelah mendapatkan hasil index yang terpilih maka rekomendasi tersebut akan di-*generate* untuk
Dijadikan query create index yang siap diimplementasikan pada database.

2.2.5 Naïve bayes

Naïve bayes adalah salah satu metode klasifikasi berdasarkan probabilitas yang menggunaka asumsi yang sangat kuat pada independensi antara alat prediksi [17]. Algoritma probabilitas dari naïve bayes dapat digunakan dalam mempertimbangkan kemunculan index seperti yang telah diteliti oleh Ameri [4] bahwa sebuah kolom yang sering muncul dan digunakan dalam sebuah *workload* perlu dipertimbangkan sebagai index. Naïve bayes *probabilities* dapat menghitung secara matematik hubungan antara kemunculan kolom dalam sebuah *workload*

dengan mempertimbangkan fakta pernyataan SELECT dan kemunculan nama kolom lain dimodelkan sebagai berikut :

$$\boldsymbol{Sf(Ij|S) = \frac{P(S|Ij)P(Ij)}{P(S)}}$$ ***Formula 2.1 Contoh Naïve Bayes***

Implementasi probabilitas dari naïve bayes pada pemilihan index sendiri telah digunakan dalam menentukan *selectifity factor* dari sebuah index [8], dan juga bisa sebagai *cost evaluation* dari sebuah kandidat index [9]. Dalam contoh penerapannya naïve bayes ialah untuk memperkirakan presentase terjadinya hujan dengan mempertimbangkan faktor terjadinya hujan yaitu mendung. Dengan adanya fakta atau data hujan dan mendung maka dalam penerapan naïve bayes *probability* ialah sebagai berikut [18]:

- **Sf(Ij|S)** ialah nilai probabilitas hipotesis kemunculan nama kolom kandidat terhadap pernyataan SELECT.
- **P(S|Ij)** ialah nilai probabilitas munculnya nama kolom yang menjadi kandidat di dalam jumlah pernyataan SELECT yang ada.
- **P(Ij)** nilai propbabilitas munculnya nama kolom yang menjadi kandidat diantara nama kolom lain tanpa memperhatikan fakta selain nama kolom.
- **P(S)** nilai probabilitas munculnya SELECT dimana dalam pemilihan index yang diperhatikan adalah SELECT dan tentunya kemunculannya bernilai pasti sehingga probabilitas kemunculan SELECT ialah 1.

Sedangkan padapenilitian berikut probabilitas dari kandidat index yang dihitung menggunakan naïve bayes menjadi *selectivity factor* pada saat proses pencacahan dari sebuah konfigurasi index yang terpilih yaitu probabilitas kemunculan dari sebuah kandidat index terhadap bukti statement yang ada.

2.2.6 Data sampel

Data sampel yang digunakan dalam penelitian berikut ini ialah berupa data sampel yang dimiliki Microsoft yaitu data mengenai perusahaan Adventure works yang tersedia secara terbuka pada website Microsoft. Untuk menggunakan data sampel berikut peneliti berpacu pada alur penggunaan data dalam database tersebut yang bisa didapatkan pada *Adventure Works Cycles Business Scenarios* [20] dengan empat skenario utama dan salah satu skema tabelnya seperti pada gambar 2.2.Proses bisnis dari Adventureworks sendiri digambarkan oleh empat skenario utama beserta contoh *query* yang dijelaskan pada website resmi Microsoft sebagai pencipta perusahaan fiksi tersebut, dan berikut ialah penjelasan skenario yang dimaksud [20]:

- **Sales and Marketing**
  Dalam skenario ini dijelaskan mengenai penjualan dan pemasaran dari Adventureworks dan penggunaan datanya mulai dari melihat data *customer*, melihat alamat dari *customer*, melihat *customer* baik retail maupun *wholesaler*, melihat kontak dari tiap toko, melihat penjualan per toko, dan melihat toko berdasarkan lokasi.

- **Product**
  Pada skenario produk yang dipaparkan dari skenario ialah bagaimana melihat produk per kategori, sub kategori, dan berdasarkan modelnya. Selain itu dapat melihat deskripsi produk berdasarkan model produk, dan melihat daftar single-level *bill of material* dari sebuah produk induk(yang dimaksud produk induk ialah misalnya sebuah sepeda dengan model tertentu).

- **Purchasing and Vendor**
  Skenario pembelian dan vendor menjelaskan mengenai kegiatan pengambilan data dalam kegiatan pembelian dan pemasokan barang pada Adventureworks sehingga beberapa kegiatan yang bisa dilakukan ialah seperti

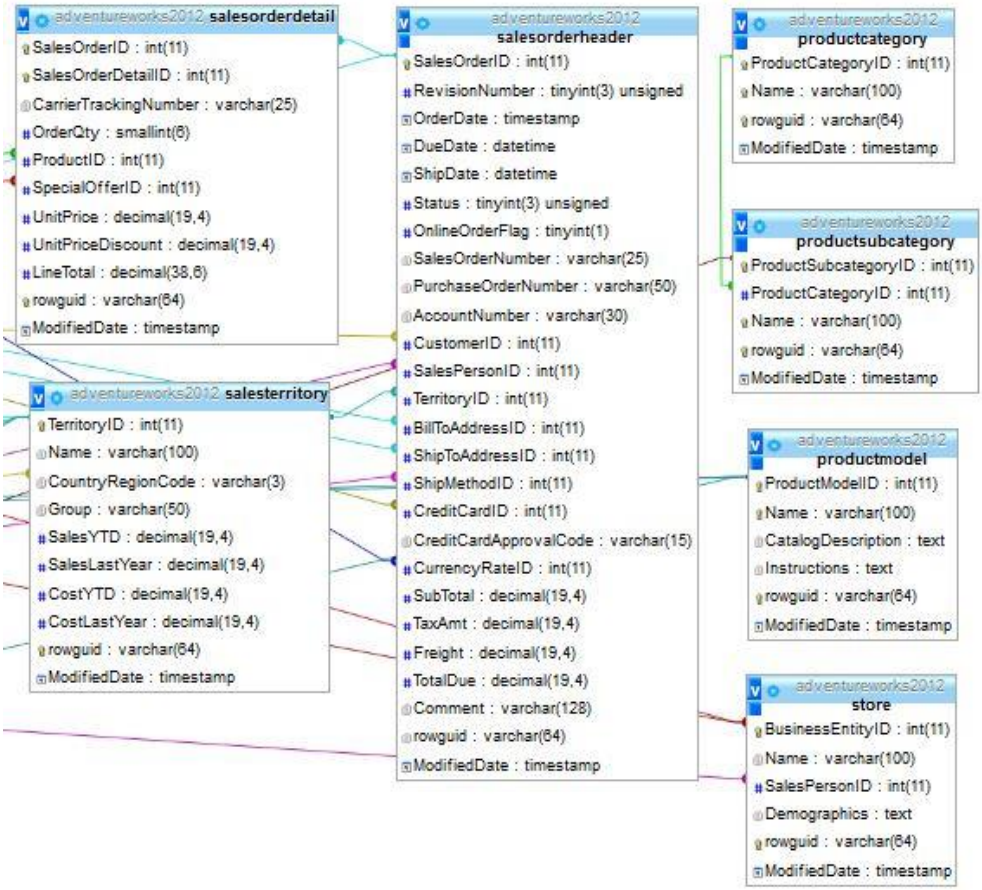
melihat vendor berdasarkan lokasi, melihat produk-produk yang dipasok berdasarkan vendor yang memasoknya, melihat kontak vendor, dan melihat pembelian dari Adventureworks terhadap vendor tertentu.

- **Manufacturing**
  Pada skenario manufaktur menjelaskan contoh dari kegiatan-kegiatan yang terjadi dalam proses manufaktur produk seperti misalnya melihat daftar multi-level *bill of material* dari sebuah produk induk, melihat *inventory* dari sebuah produk, dan melihat order yang sedang dikerjakan untuk sebuah produk tersebut.

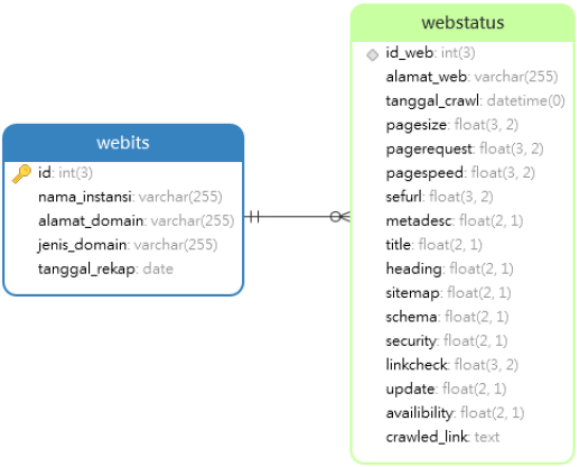
Contoh *query* menurut skenario penjualan yaitu apabila ingin melihat penjualan pertoko ialah sebagai berikut:

```
SELECT Name, SalesOrderNumber, OrderDate, TotalDue
FROM Sales.Store AS S
    JOIN    Sales.SalesOrderHeader    AS    SO    ON
S.CustomerID = SO.CustomerID
ORDER BY Name, OrderDate ;
```

Selain Adventureworks, digunakan lagi sebuah data berupa data sampel aplikasi yang dihasilkan dari penelitian terdahulu mengenai evaluasi kinerja dari web yaitu pantau-web.its.ac.id [21]. Pantau-web.its.ac.id menggunakan dua database dalam melakukan proses operasionalnya yaitu ‘simdomits; dan ‘Monitordom’. Proses operasional pantau-web.its.ac.id sendiri banyak menggunakan query select pada database Monitordom sehingga yang digunakan sebagai data sampel ialah database Monitordom dengan struktur database pada gambar 2.3. Pertimbangan untuk menggunakan data sampel berkut ialah karena banyaknya transaksi yang harus dilakukan aplikasi berbasis *query* untuk melakukan penilaian dari metriks yang dimiliki oleh web dan penialaian tersebut dapat memakan waktu hingga 2,5 jam sehingga dirasa data sampel ini bisa digunakan sebagai objek uji coba penelitian rekomendasi index.



Gambar 2.2. Skema database *sampel sebagian dari Sales and marketing*

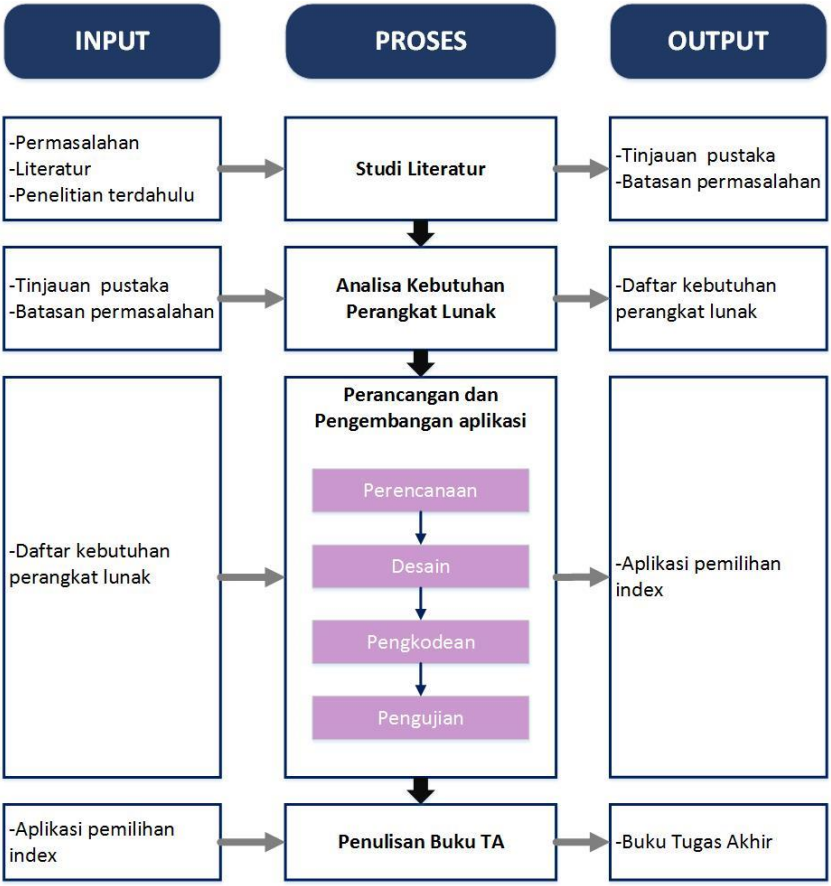


Gambar 2.3. Skema model database *sampel*

Dengan menggunakan dua jenis data sampel tersbut diharapkan dapat mencakup beberapa kemungkinan yang ada pada jenis data dan proses yang terjadi pada database.

BAB III
METODOLOGI PENELITIAN

Pada bab metodologi penelitian akan menjelaskan alur dari penelitian Berikut ini ialah diagram alur dari pengerjaan penelitian pada gambar 3.1.



Gambar 3.1. Alur metodologi penelitian

3.1 Deksripsi metodologi penelitian

3.1.1 Studi Literatur

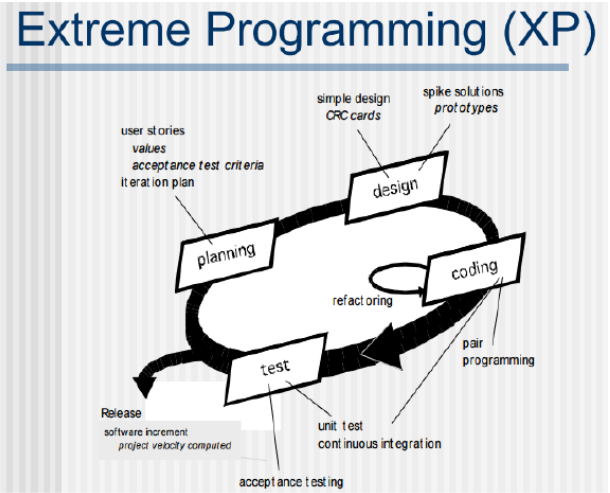
Berdasarkan penelitian terdahulu, literatur dan permasalahan yang ada yaitu pemilihan index yang optimal tahap awal yang dilakukan pada penelitian berikut ini ialah melakukan studi literatur menganai Database *performance*, Index *Selection Problem*, Java, dan penelitian-penelitian terdahulu serta dasar teori untuk mendapatkan tinjauan pustaka dan batasan permasalahan dalam penelitian berikut.

3.1.2 Analisa Kebutuhan Perangkat Lunak

Setelah mendapat tinjauan pustaka dan batasan permasalahan untuk penelitian berikut ini selanjutnya ialah menganalisis kebutuhan perangkat lunak yang akan dibuat agar pembuatan perangkat lunak sesuai dengan kebutuhan yang diinginkan. Luaran dari tahap ini ialah daftar kebutuhan fungsional dari perangkat lunak yang akan dikembangkan. Analisa kebutuhan perangkat lunak sendiri dilakukan dengan melakukan observasi terhadap beberapa penelitian terdahulu mengenai rekomendasi atau pemilihan index.

3.1.3 Perencanaan dan pengembangan aplikasi

Pada tahap berikut ini ialah tahap perencanaan dan pengembangan yang merupakan implementasi dari analisa kebutuhan perangkat lunak dengan menggunakan metode pengembangan *Agile Software Development* khususnya ialah mengacu pada langkah-langkah dari *Extreme Programming* yaitu perencanaan, desain, pengkodean, dan pengujian seperti pada gambar 3.2.



Gambar 3.2. Extreme Programming

3.1.3.1 Perencanaan

Pada tahap ini dilakukan perencanaan awal mengenai bagaimana aplikasi yang diinginkan dan perancangan umum mengenai aplikasi seperti gambaran mengenai arsitektur sistem, data sampel yang digunakan, dan menyiapkan *query* ataupun *log* yang digunakan untuk pengujian dan perencanaan *timeline* pengerjaan dalam penelitian.

Arsitektur sistem dalam penelitian ini dipaparkan oleh gambar 3.3. Dimana aplikasi java yang akan dibangun akan terhubung dengan server MariaDB dan menerima input dari user berupa *general log* atau berupa langsung *query* yang digunakan oleh admin database.

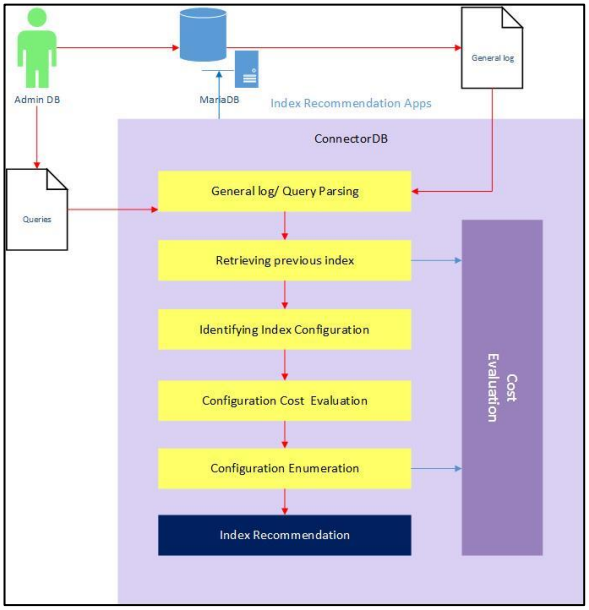
Pemilihan index akan dilakukan berdasarkan algoritma heuristik. Pertama akan dilakukan *Parsing* untuk mengambil frasa yang dibutuhkan saja. Selain itu aplikasi akan mendapatan index yang sudah ada apabila database sudah memiliki index sebelumnya. Aplikasi akan membuat konfigurasi index dan melakukan evaluasi terhadap *query* yang ada. Setelah mendapat konfigurasi yang paling baik kinerjanya, dilakukan pencacahan

konfigurasi untuk mendapat atribut atau nama kolom secara atomik kemudian dipilih berdasarkan *selectivity factor*.. Sehingga admin akan mendapatkan luaran berupa *query* pembuatan index yang direkomendasikan oleh aplikasi.

3.1.3.2 Desain

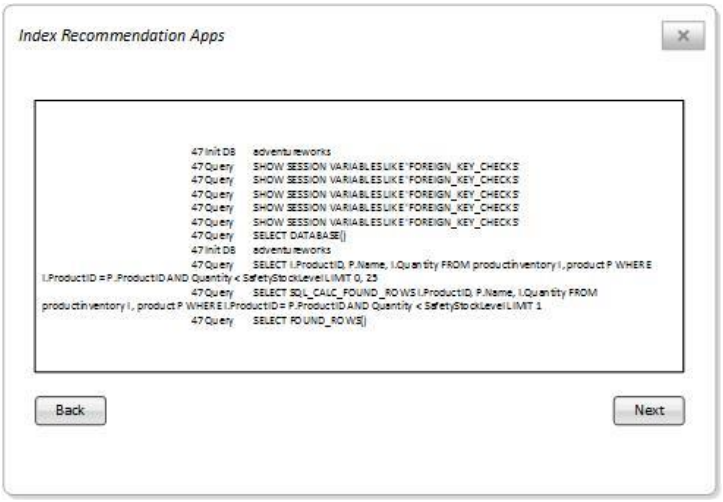
Pada tahap ini akan dilakukan desain *user interface* dari aplikasi untuk memenuhi kebutuhan fungsional dari hasil analisis kebutuhan fungsional fase sebelumnya.

- Desain 1 memenuhi kebutuhan fungsional a, dimana admin database dapat melakukan pengaturan database dan memlih jenis input yang digunakan yaitu antara *general log* atau *query*.
- *Parsing* dan pemanggilan kembali index yang sudah ada sebelumnya digambarkan desain 2 dan desain 3 yanitu pemanggilan kembali memenuhi kebutuhan fungsional b dan c.
- Pembuatan konfigurasi index dan cost evaluation memenuhi kebutuhan fungsional d dan f digambarkan pada usulan desain 4.

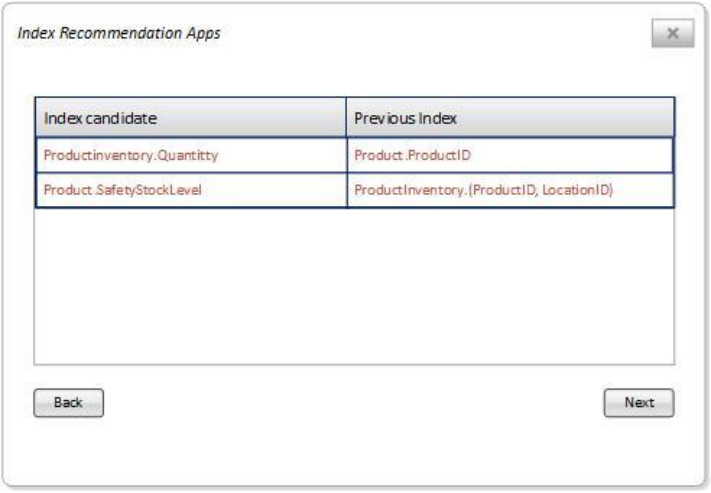


Gambar 3.3. Arsitektur aplikasi

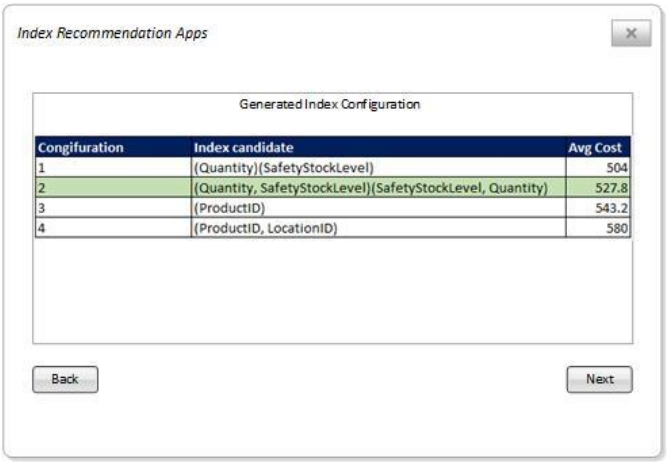
Gambar 3.4. Desain 1



Gambar 3.5. Desain 3

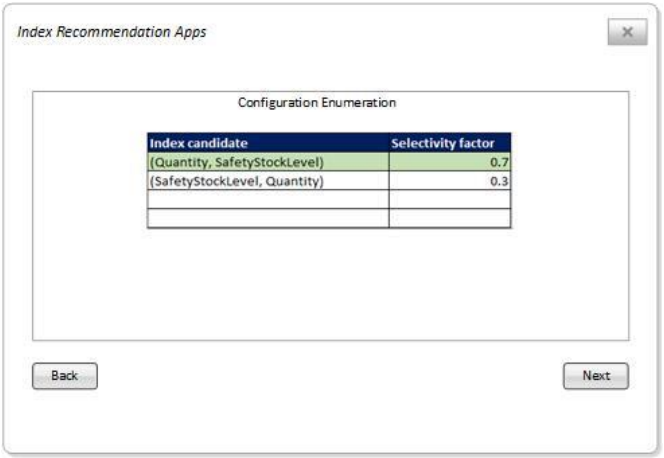


Gambar 3.6. Desain 2



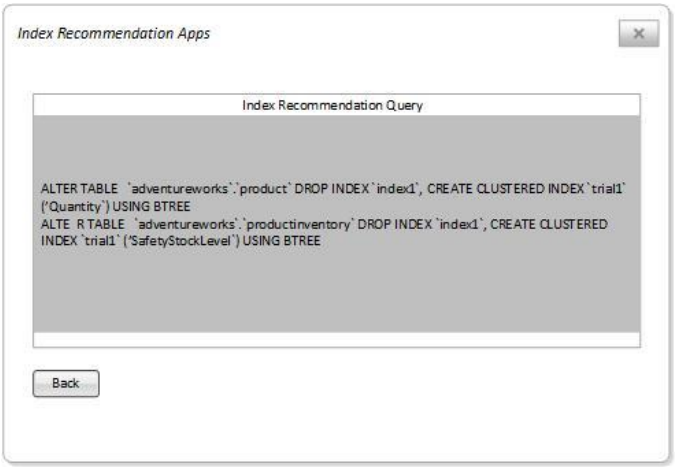
Gambar 3.7. Desain 4

- Pencacahan konfigurasi index, penghitungan *selectivity factor* untuk dipilih yang tertinggi. Gambaran desain 5 berikut memenuhi fungsionalitas g dan h.



Gambar 3.8. Desain 5

- Pembuatan *query* index yang direkomendasikan memenuhi fungsionalitas j yang merupakan akhir dari aplikasi ini digambarkan pada Desain 6.



Gambar 3. 9. Desain 6

3.1.3.3 Pengkodean

Pada proses pengkodean dilakukan dengan menggunakan Java dan dilakukan secara bertahap. Proses implementasi algoritma akan dilakukan berdasarkan desain antarmuka yang diharapkan dapat memenuhi kebutuhan fungsionalitas dengan tahap pengkodean berikut:

- Setting
- *Parsing* dan pengambilan index sebelumnya
- Pembuatan konfigurasi index dan pemilihan kinerja dari konfigurasi terbaik.
- Pencacahan konfigurasi
- Pembuatan query rekomendasi index

3.1.3.4 Pengujian

Pada tahap pengujian dibagi menjadi dua tahap yaitu verifikasi perangkat lunak dan validasi perangkat lunak yang dijelaskan sebagai berikut:

Verifikasi perngkat lunak

Verifikasi dilakukan untuk memastikan perangkat lunak tidak mengalami *error* atau fungsionalitas yang tidak berjalan. Cara memastikannya ialah dengan melakukan uji coba penggunaan aplikasi hingga mendapatkan output.

Validasi perangkat lunak

Setelah aplikasi rekomendasi index berbasis java telah dibuat selanjutnya ialah melakukan validasi apakah algoritma pada aplikasi tersebut berjalan sesuai dengan tujuan dengan cara melakukan perbandingan evaluasi berdasarkan data sampel yang ada. Apabila hasil kinerja database yang menggunakan hasil rekomendasi index lebih optimal diandingkan dengan hasil kinerja database yang tidak menggunakan index, maka algoritma dalam aplikasi telah valid.

3.1.4 Penulisan Buku Tugas Akhir

Tahapan terakhir dari penelitian ini ialah melakukan penulisan dokumentasi penelitian dari seluruh proses kerja dan hasil dari penelitian sehingga menghasilkan sebuah buku penelitian Tugas Akhir yang dimaksudkan agar dapat diulas kembali dan dapat dikembangkan bagi peneliti selanjutnya.

Halaman ini sengaja dikosongkan.

BAB IV
PERANCANGAN

Pada bab ini akan menjelaskan mengenai detail perancangan untuk menjalakan implementasi yang mencakup data sample dan desain.

4.1 Pengambilan Data Sample

Dalam penelitian berikut ini data sampel yang digunakan dari dua database yaitu AdventureWorks2012 dan Monitordom. AdventureWorks2012 yang merupakan database sampel yang disediakan oleh Microsoft dan Monitordom merupakan database yang digunakan dalam Pantau-web.its.ac.id. pengambilan data sendiri dengan cara observasi.

4.1.1 Database

Untuk AdventureWorks2012 data yang ada telah tersedia terbuka dalam website resmi Microsoft namun masih berupa format untuk SQL server sehingga harus dilakukan migrasi dengan menggunakan bantuan *tools* MySQL Workbench Migration Tools. Sedangkan untuk Monitordom didapatkan dari peneliti terdahulu secara langsung melalui observasi.

4.1.2 General Log dan Query

Sebagai bahan *input* untuk evaluasi kinerja dalam pemilihan index, dua data yang digunakan mewakili *input* yang berbeda. Dalam AdventureWorks menggunakan input berupa Query yang didapatkan dari skenario bisnis dari web Microsoft dan diimplementasikan dalam kasus studi untuk kegiatan *Custemer Relationship Management* dengan asumsi penggunaan metode *Recency, Monitery, and Frequency* (RFM) untuk melakukan klasterisasi. Cara klasterisasi berikut ini telah diterapkan oleh perusahaan untuk melakukan segmentasi pelanggan [22] guna mengetahui pelanggan yang paling menguntungkan.

Berdasarkan analisa kebutuhan data untuk pengujian pada database Adventureworks, maka *query* yang diadaptasi dari skenario bisnis Adventurworks untuk memenuhi data yang diperlukan untuk CRM dengan metode RFM akan dilampirkan.

Sedangkan untuk Monitordom menggunakan *input* berupa General Log yang tercatat pada file .log ketika aplikasi berjalan dan log tersebut berisi semua *query* yang digunakan dalam melakukan sekali penilaian yang telah di proses ulang untuk disesuaikan dengan database pengujian. Pantau-web.its.ac.id menggunakan dua database dalam kegiatan penilaiannya sehingga dalam pengujan dipilih database yang paling sering digunakan untuk query SELECT yaitu monitordom dan menghilangkan query lain yang tidak dieksekusi dalam database Monitordom.

Input untuk penelitian ini memiliki pola yang berbeda dimana Monitordom memiliki banyak query namun tidak mengandung banyak elemen yang berpengaruh untuk pencarian index dan Adventureworks2012 memiliki sedikit *Query* namun lebih kompleks dimana seluruh *Query*-nya memiliki elemen yang memiliki pengaruh terhadap index. Tabel 4.1 menjelaskan mengenai statistic dari input yang digunakan dalam penelitian berikut ini dengan pengukuran dari elemen yang berpengaruh pada pencarian index menurut Winand yaitu GROUP BY, ORDER BY, AND, LIKE, dan BETWEEN.

Tabel 4.1. Statistik Input

	Monitordom	Adventureworks2012
ALL QUERIES	2183	6
SELECT	680	6
GROUP BY	0	6
ORDER BY	679	6
AND	4	0
LIKE	0	0
BETWEEN	0	0

Monitordom meskipun memiliki jumlah *query* SELECT sejumlah 680 namun pada kenyataannya hanya memiliki tiga pola *query* utama sebagai beirkut:

- `SELECT `crawled_link` FROM `webstatus` WHERE `alamat_web`='iwwsmssc.its.ac.id' ORDER BY `webstatus`.`tanggal_crawl` DESC limit 1`
- `SELECT COUNT(DISTINCT DATE(`tanggal_crawl`)) as `nilai` FROM `webstatus``
- `SELECT DISTINCT DATE(`tanggal crawl`) as `tanggal` FROM `webstatus` ORDER BY DATE(`tanggal_crawl`) ASC LIMIT 1`

Pola *query* dari Monitordom tersebut akan digunakan sebagai bahan pengujian kinerja per *query* untuk mengetahui *query* yang paling banyak mendapatkan peningkatan kinerja setelah menggunakan rekomendasi dari perangkat lunak berikut.

4.2 Analisa Kebutuhan Perangkat Lunak

Kebutuhan perangkat lunak didapatkan dengan cara observasi pada penelitian mengenai pemilihan atau rekomendasi index yang telah dilakukan peneliti sebelumnya yang telah dicantumpak pada bagian penelitian terdahulu. Setelah didapatkan kesimpulan maka kebutuhan fungsional aplikasi pada penelitian ini ialah:

a. Dapat melakukan *setting* terhadap database tujuan.
b. Dapat melakukan *parsing* dari input berupa *general log* ataupun *query*.
c. Apabila sudah ada index dalam database, aplikasi dapat mencatat index sebelumnya.
d. Dapat membuat konfigurasi index.
e. Dapat melakukan pengukuran kinerja dengan menggunakan *response time* atau *qps.*
f. Dapat memilih konfigurasi index dengan kinerja paling baik.
g. Dapat melakukan pencacahan konfigurasi index.
h. Dapat memilih hasil pencacahan konfigurasi index dengan *selectivity factor* paling tinggi.
i. Dapat menghasilkan *query* pembuatan index dari hasil index yang direkomendasikan.

4.3 Desain Sistem

Sistem berikut ini berupa aplikasi java yang akan dibangun akan terhubung dengan server MariaDB dan menerima input dari user berupa *general log* atau berupa *query* kemudian diproses ketika sudah terhubung melalui log in pada database yang dijelaskan oleh gambar 4.1. Setelah input berhasil di-*parsing* hasil nama kolom dan tabel disimpan pada 'temptable' dan akan dijadikan sebagai bahan melakukan permutasi yaitu proses Generate Configuration. Hasil configurasi akan disimpan pada tabel 'tableconf'.

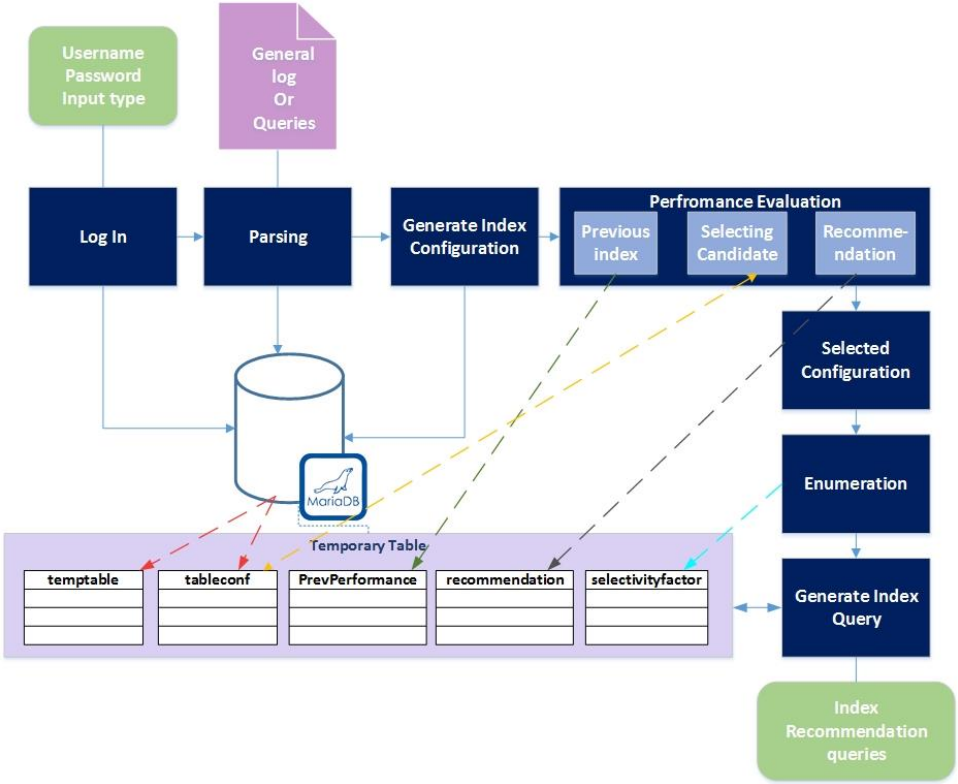
Proses berikutnya yaitu penilaian kinerja yang terdiri dari beberapa sub proses yaitu prev performance yang merupakan hasil kinerja tanpa index rekomendasi dan penilaian disimpan pada tabel 'prevperformance', penilaian setiap kandidat yang berupa konfigurasi disimpan dalam tabel 'tableconf', dan penilaian hasil dari penerapan kandidat terbaik tersebut jika merupakan multi-level-index akan melewati proses Enumeration dan dinilai *seletivityfactor*-nya untuk disimpan pada table 'Selectivityfactor' dan dipilih yang terbaik, terakhir penilaian dengan kondisi database telah menerapkan dari hasil seluruh kandidat yang terpilih akan dievaluasi dan disimpan pada tabel 'recommendation'.

Penyimpanan kinerja pada tabel-tabel sementara tersebut dimaksudkan untuk memberikan informasi mengenai perubahan kinerja yang didapatkan apabila menerapkan index hasil rekomendasi dari sistem. Karena bersifat rekomendasi, pada akhir proses sistem hanya akan mengimplementasikan langsung hasil rekomendasi index apabila pengguna dapat memilih untuk mengimplementasikan index tersebut atau tidak mengimplementasikannya dengan pertimbangan perubahan kinerja yang ditampilkan.

4.4 Desain Tabel Sementara

Dalam proses penilaian kinerja dan memberikan rekomendasi index yang telah dibahas pada bagian desain sistem, perangkat lunak membuat lima tabel sementara dalam database dan akan di hapus ketika aplikasi sudah berhasil memberikan rekomendasi. Tabel tersebut ialah Temptable, Tableconf, Prevperformance, Recommendation dan Selectifityfactor seperti pada tabel 4.2.

Pembuatan tabel dilakukan pada proses ketika tabel tersebut akan digunakan agar terlalu memakan banyak penyimpanan sementara apabila tabel tersebut sama sekali tidak digunakan. Seperti ‘Selectivityfactor’ yang akan dibuat pada saat apabila dari seluruh kandidat yang terbaik terdapat kandidat terpilih yang merupakan multi-level-index.



Gambar 4.1. Desain sistem

Tabel 4.2. Data Dictionary tabel sementara dalam perangkat lunak

Entity	Attribute	Description	Data Type & Length	Null
Temptable	Table_name	Menyimpan nama tabel kandidat dari hasil parsing	VARCHAR (50)	*Not Null*
	Column_name	Menyimpan nama kolom kandidat dari hasil parsing	VARCHAR (50)	*Not Null*
Tableconf	Table_name	Menyimpan nama tabel konfigurasi dari hasil permutasi	VARCHAR (50)	*Not Null*
	Index1	Menyimpan nama kolom index kolom pertama dari hasil permutasi	VARCHAR (50)	*Not Null*
	Index2	Menyimpan nama kolom index kolom kedua dari hasil permutasi untuk jenis index multi-level-index	VARCHAR (50)	*Null*
	QPS	Menyimpan nilai QPS dari hasil evaluasi konfigurasi	FLOAT (50)	*Null*
	Responsetime	Menyimpan nilai Responsetime dari hasil evaluasi konfigurasi	FLOAT (50)	*Null*
Prevperformance	Table_name	Menyimpan nama tabel untuk hasil evaluasi kinerja dengan index yang sudah dimiliki oleh database	VARCHAR (50)	*Not Null*
	QPS	Menyimpan nilai QPS untuk hasil evaluasi kinerja dengan index yang sudah dimiliki oleh database	FLOAT (50)	*Null*
	Responsetime	Menyimpan nilai Responsetime untuk hasil evaluasi kinerja dengan index yang sudah dimiliki oleh database	FLOAT (50)	*Null*

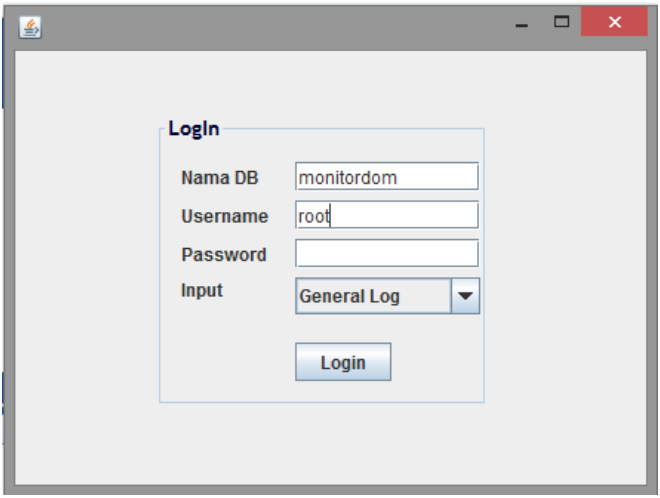
Entity	Attribute	Description	Data Type & Length	Null
Recommendation	Table_name	Menyimpan nama tabel untuk hasil evaluasi kinerja dengan kondisi database menerapkan index hasil rekomendasi	VARCHAR (50)	*Not Null*
	QPS	Menyimpan nilai QPS untuk hasil evaluasi kinerja dengan kondisi database menerapkan index hasil rekomendasi	FLOAT (50)	*Null*
	Responsetime	Menyimpan nilai Responsetime untuk hasil evaluasi kinerja dengan kondisi database menerapkan index hasil rekomendasi	FLOAT (50)	*Null*
Selectivityfactor	Table_name	Menyimpan nama tabel dari kolom yang melewati proses pencacahan	VARCHAR (50)	*Not Null*
	Column_name	Menyimpan nama tiap kolom yang telah melakukan penilaian selectivityfactor	VARCHAR (50)	*Not Null*
	Selectivity_factor	Menyimpan nilai probabilitas kemunculan yang telah dinilai dengan metode selectivity factor	FLOAT (50)	*Not Null*

4.5 Desain Antarmuka

Desain antarmuka berikut ini ialah sesuai dengan Jframe atau tampilan yang akan digunakan dalam pengembangan perangkat lunak yang terdiri dari beberapa frame yaitu log in, parsing, generate configuration, selected configuration, selectivity factor and index generation.

4.5.1 Log in

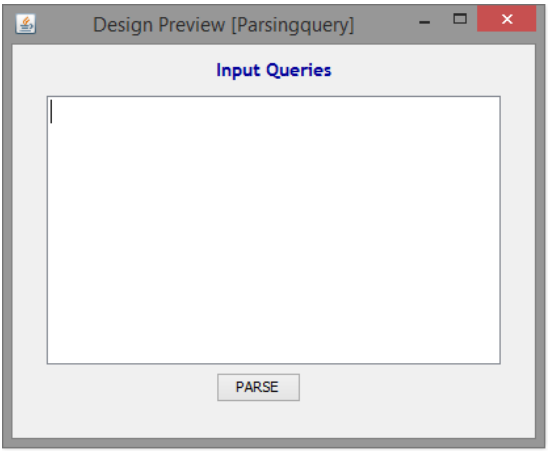
Prototype login seperti pada gambar 4.2 ialah halaman untuk mengisi data mengenai database tujuan dan jenis input yang akan digunakan.



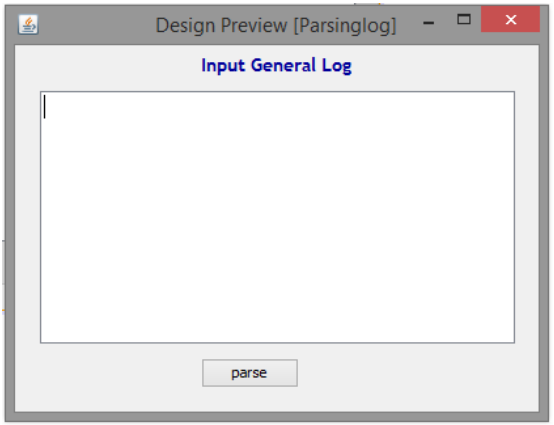
Gambar 4.2. Prototype Login

4.5.2 Parsing

Apabila log in berhasil maka akan menuju dua tipe halaman berikut ini yaitu input query atau input general log seperti pada gambar 4.3 dan gambar 4.4.



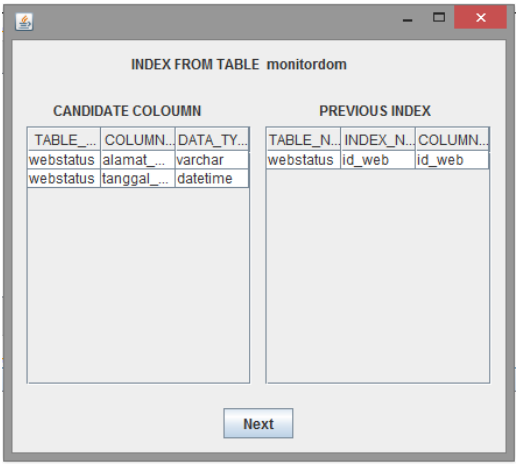
Gambar 4.3. Prototype Input Query



Gambar 4.4. Prototype Input General Log

4.5.3 Retrieve previous index

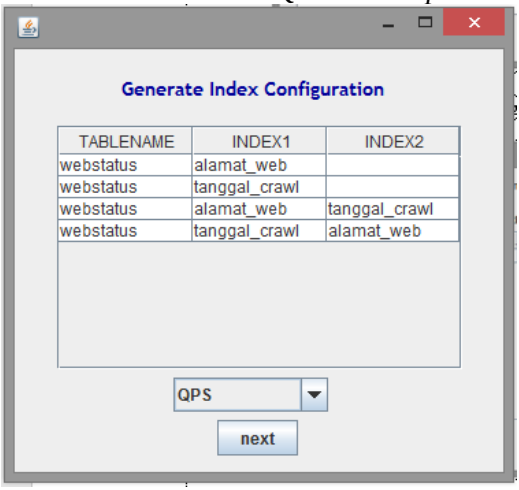
Setelah hasil parsing dari input didapatkan akan ditampilkan dalam tabel candidate column dan untuk mengetahui index apa yang sudah ada pada database tersebut ditampilkan pada tabel previous index seperti pada gambar 4.5.



Gambar 4. 5. Prototype Retrieve previous index

4.5.4 Generate index configuration

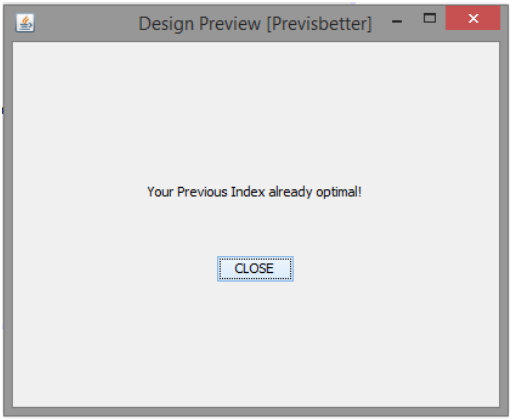
Setelah mempunyai candidate column yang memungkinkan mejadi kandidat index yang direkomendasikan, dilakukan permutasi hingga n = 2, sehingga akan mendapatkan hasil permutasi index1 dan index2 yang mungkin terjadi untuk setiap tabelnya seperti pada gambar 4.6. Dalam halaman ini juga ditentukan evaluasi berdasarkan QPS atau *responsetime*.



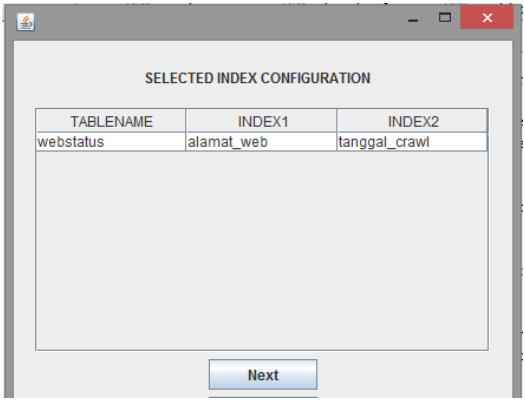
Gambar 4. 6. Protoype Generate Index *Configuration*

4.5.5 Selected Configuration

Kondisi database dengan index yang sudah ada akan dievaluasi pula, kemudian dibandingkan dengan rata-rata hasil dari evaluasi konfigurasi. Apabila hasil evaluasi kondisi database dengan index sebelumnya lebih baik maka akan muncul ke halaman pemberitahuan bahwa index yang dimiliki database tersebut sudah optimal dan tidak akan melakukan proses evaluasi selanjutnya seperti pada gambar 4.7. Apabila hasil evaluasi dari konfigurasi lebih baik maka akan menampilkan halaman konfigurasi dengan hasil terbaik seperti gambar 4.8.



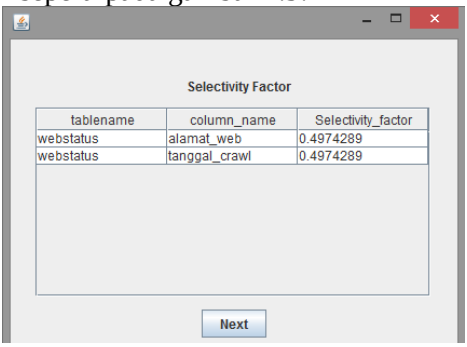
Gambar 4.7. Prototype notifikasi previous index *already optimal*



Gambar 4.8. Prototype Selected index configuration

4.5.6 Selectivity factor

Dimungkinkan pada saat konfigurasi yang terpilih ialah configurasi dengan komposisi multi-level-index atau yang terdiri dari dua kolom. Apabila yang terpilih dalam sebuah tabel ialah konfigurasi hanya dengan index primer maka tidak akan dilakukan proses perhitungan *selectivity factor*, proses *selectivity factor* hanya untuk konfigurasi multi-level-index yang terpilih seperti pada gambar 4.9.



Gambar 4.9. Prototype Selectivity factor

4.5.7 Generate Query

Setelah dilakukan penilaian dengan *selectivity factor* maka dipilih probabilitas paling tinggi dari kombinasi index yang yang akan dijadikan index, apabila dari kedua kolom memiliki probabilitas yang sama maka akan tetap menjadi rekomendasi multi-level-index untuk tabel tersebut.

Gambar 4.10. Prototype generate and implement index *query*

4.6 Desain Pengkodean

Pada proses pengkodean dilakukan dengan menggunakan Java dan dilakukan secara bertahap beberapa tahap seperti pada uraian berikut:

- **Setting**
  Pada saat membuat setelan awal pengkodean dilakukan dengan java dan menggunakan *library* berupa keneksi yang dimiliki MariaDB yang bisa didapatkan pada website MariaDB [22]. Selanjutnya ialah menyiapkan halaman untuk input *log* dan *query* karena admin *database* hanya bisa memilih salah satu input tersebut. Tujuan utama pada pengkodan ini ialah agar bisa melakukan koneksi dengan *database* dan menentukan langkah *parsing* yang akan dilakukan dan mulai dari sini ialah penerapan algoritma heuristik [9] dari penelitian sebelumnya hingga proses pemilihan *selectivity factor*.

- ***Parsing* dan pengambilan index sebelumnya**
  Proses pengambilan nama kolom dan tabel pada saat parsing dilakukan dengan menggunakan *library* JSqlParser yang didapatkan secara terbuka pada forum github [24]. Sedangkan untuk menampilkan index sebelumnya dilakukan *query* yang memanggil index dari informatics.schema dari database.

- **Pembuatan konfigurasi index dan pemilihan kinerja dari konfigurasi terbaik.**
  Cara pembuatan konfigurasi ialah dengan mengkombinasikan beberapa nama kolom yang muncul dan kemungkinan index yang terjadi tersebut per konfigurasi dapat dihitung dengan rumus sebagai berikut [13]:

$$\sum_{i=1}^{n} \frac{n!}{(n-j)!}$$ *Formula 4.1. Kemungkinan index yang terjadi*

Dimana *n* ialah jumlah kolom yang merupakan kandidat index dan j ialah jumlah kolom berbeda yang ingin diindex. Dikarenakan kombinasi dari konfigurasi index yang bisa menjadi sangat banyak, seperti pada penelitian chauduri [9] bahwa jumlah paling efektif untuk kombinasi kolom dalam sebuah index ialah 2 sehingga j = 2. Setelah mendapatkan satu set index yang memiliki *cost* paling kecil berdasarkan *rows* yang dibaca ketika terjadi eksekusi *query*. *Cost Evaluation* sendiri dihitung berdasarkan *Response time* dan *throughput* yang berupa *Queries per Second*.

- **Pencacahan konfigurasi**
  Pencacahan konfigurasi yaitu membagi kandidat index dari konfigurasi index menjadi satuan dan memilih satuan kandidat menggunakan metode naïve bayes ialah sebagai berikut:

$$\boldsymbol{Sf(Ij|S)} = \frac{\boldsymbol{P(S|Ij)P(Ij)}}{\boldsymbol{P(S)}}$$ *Formula 4.2. Selectivity factor*

Dimana S ialah jumlah *statement query* yang terdapat dalam *workload* dan *Ij* ialah probabilitas kandidat index yang muncul dalam *workload* terhadap kandidat index yang lain. Kandidat index dengan *Sf* tertinggi akan menjadi index yang akan direkomendasikan [12]. Namun sebelumnya ditentukan terlebih dahulu jenis *cluster* ataupun *non-clustered* nya.

- **Pembuatan query rekomendasi index**
  Terakhir ialah menampilkan rekomendasi index yang berupa *query* yang telah dibuat berdasarkan hasil rekomendasi index. Selain menampilkan index yang direkomendasikan, pada bagian ini akan menunjukkan hasil peningkatan presentase performa dengan perhitungan pada formula 4.3 dan formula 4.4. dua penilaian peningkatan tersebut berbeda karena untuk QPS semakin banyak akan

semakin baik dan Responsetime semakin sedikit dianggap semakin baik.

$$\mathbf{QPS} = \left(\frac{\mathbf{QPSakhir - QPSawal}}{\mathbf{QPSawal}}\right)\mathbf{100}\%$$

Formula 4.3. Presentase kenaikan QPS

$$\mathbf{Responsetime} = \left(\frac{\mathbf{RTawal - RTakhir}}{\mathbf{QPSawal}}\right)\mathbf{100}\%$$

Formula 4.4. Presentase kenaikan responsetime

BAB V
IMPLEMENTASI

Pada bab implementasi akan menjelaskan mengenai lingkungan implementasi dari penelitian berikut beserta implementasi pembuatan aplikasi.

5.1 Lingkungan Implementasi

Dalam proses pengembangan dan pengujian perangkat lunak rekomendasi index berikut, lingkungan implementasi berupa perangkat keras yang digunakan yaitu pada tabel 5.1 dan perangkat lunak yang digunakan terdapat pada tabel 5.2.

Tabel 5.1 Perangkat keras

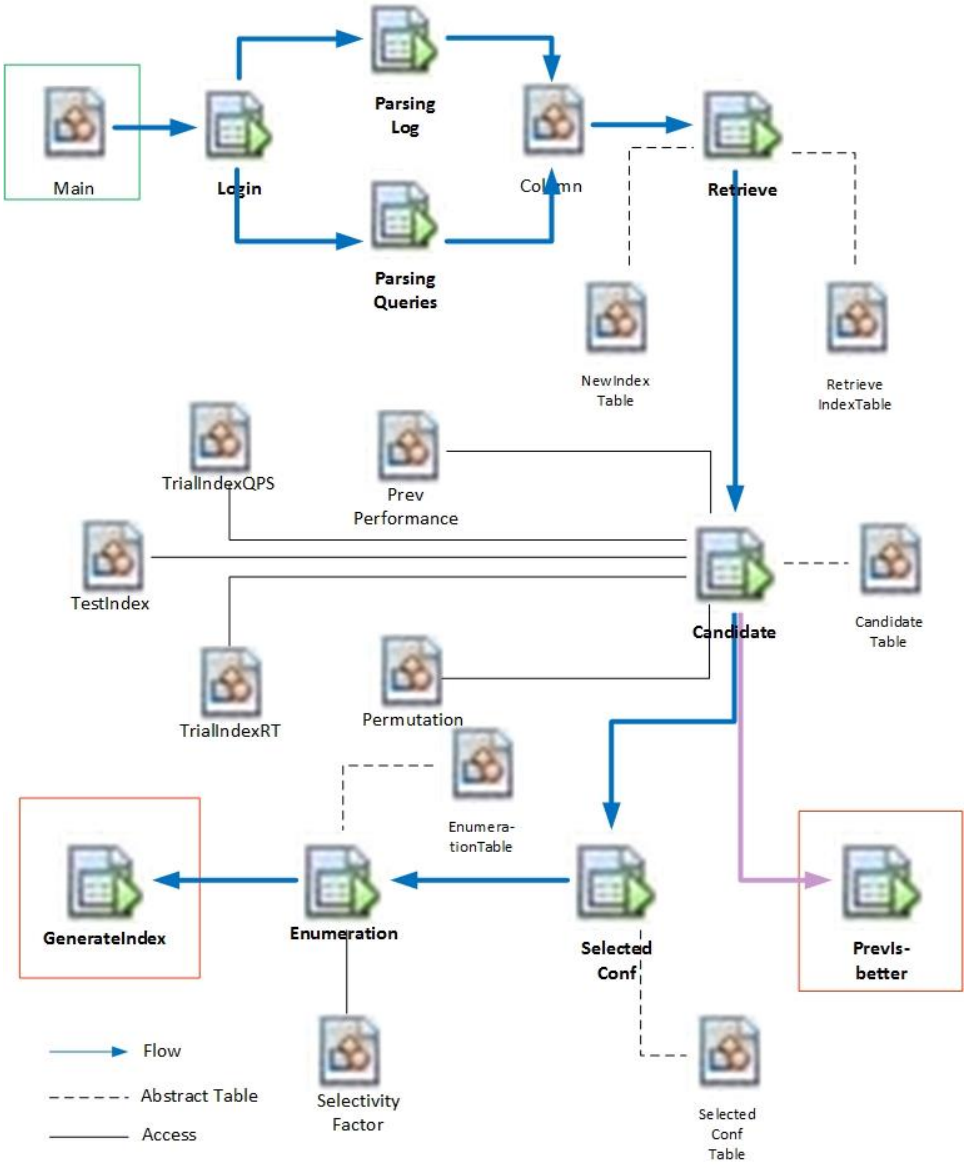
Perangkat	Spesifikasi
Jenis	Sony SVF14319SGB
Processor	Intel Core i5
RAM	8GB
Hard Disk Drive	1000GB

Tabel 5.2 Perangkat Lunak

Nama Perangkat Lunak	Kegunaan dan Implementasi
MariaDB 10.1.19	Database
NetBeans IDE 8.2	Pengembangan aplikasi dengan java

5.2 Pembuatan Aplikasi

Pembuatan applikasi menggunakan Bahasa pemrograman java berikut ini terbagi menjadi dua bagian penjelasan yaitu pembuatan Class yang termasuk pula beberapa Class untuk *abstact table* dan Frame dengan alur eksekusi pada gambar 5.1, dimana dimulai dari class Main dan dapat berakhir di frame Previsbetter jika kinerja database dengan index yang sebelumnya lebih baik atau sebaliknya akan berakhir di Frame Generateindex.



Gambar 5.1. Alur eksekusi class dan frame

5.2.1 Class:

Class adalah *prototype*, atau *blueprint*, atau rancangan yang mendefinisikan variable dan method-methode pada seluruh objek tertentu [25]. Pada implementasi pengembangan perangkat lunak untuk rekomendasi index berikut terdiri dari beberapa class yaitu Main, Connect, Column, Permutation, Responsetime, QPS, Prevperformance, TrialIndexQPS, TrialIndexRT. Terdapat pula Class yang berfungsi sebagai implementasi dari *Abstract Table* yang berada pada frame yaitu Class RetrieveIndexTable, NewIndexTable, CandidateTable, dan SelectedConfTable.

Main

Pada class Main hanya sebagai awal dari mulainya aplikasi yaitu menuju Frame Login.

```
public static void main(String[] args) {
         java.awt.EventQueue.invokeLater(new Runnable()
{
             public void run() {
                 new Login().setVisible(true);
             }});
    }
```

Kode 5.1. Class Main

Connect

Class Connect digunakan pada seluruh Class lain karena di sinilah koneksi antara aplikasi dan database tujuan dieksekusi.

```
public void init() {
        try {
             Class.forName("com.mysql.jdbc.Driver");
             Connect =
DriverManager.getConnection("jdbc:mysql://localhost/" +
db, user, pass);
             JOptionPane.showMessageDialog(null,
"Connected");
         } catch (Exception e) {
             JOptionPane.showMessageDialog(null,
"Connection Failed");
             e.printStackTrace();
         }  }
```

Kode 5.2. Class Connect

Column

Class Column melakukan proses parsing dari input dari Frame Parsing Log atau Parsing Queries dengan menggunakan *library* JSQLParser untuk mengambil nama kolom dan nama tabel yang ada dari pola *query* input yang kemudian disimpan dalam bentuk array.

```
public void init() {
        inputpisah = inputawal.split("\n");
        for (String inputSatu : inputpisah) {
            Pattern patternselect =
Pattern.compile("^(SELECT) | ^(select)");
            Matcher input1 =
patternselect.matcher(inputSatu);
            Statement stmt;
            if (input1.find()) {
                try {
                    stmt =
CCJSqlParserUtil.parse(inputSatu);
                    Select selectStatement = (Select)
stmt;
                    CobacobaFinder columnNamesFinder =
new CobacobaFinder();
                    List<String> columnList =
columnNamesFinder.getColumnList(selectStatement);
                    output += columnList.toString();

                    TablesNamesFinder tabelfinder = new
TablesNamesFinder();
                    List<String> tableList =
tabelfinder.getTableList(selectStatement);
                    outputtable += tableList.toString();

                } catch (JSQLParserException ex) {

Logger.getLogger(Parsingquery.class.getName()).log(Level.
SEVERE, null, ex);
                }

            } else {

            }
```

Kode 5.3. Class Column

RetrieveIndexTable

RetrieveIndexTable ialah bentuk tabel abstrak dari salah satu tabel di Frame Retrieve yang berfungsi untuk mengambil nilai dari *Query* yang dieksekusi berupa pengambilan Index yang sudah ada di database.

```
public class RetrieveIndexTable extends
AbstractTableModel {
    private int colnum1=3;
    private int rownum1;
    private String[] colNames1={"TABLE_NAME",
"INDEX_NAME", "COLUMN_NAME"};
    private  ArrayList<String[]> ResultSets1;
    public RetrieveIndexTable(ResultSet rs1)
        ResultSets1=new ArrayList<String[]>();
      try{
        while(rs1.next()){
                  String[] row={

rs1.getString("TABLE_NAME"),rs1.getString("INDEX_NAME")
,rs1.getString("COLUMN_NAME")
            };
            ResultSets1.add(row);
         }
      }
      catch(Exception e){
          System.out.println("Exception in
CarTableModel");
            }
    }
```

*Kode 5.4. Class Retrieve*Index*Table*

NewIndexTable

Sedangkan untuk NewIndexTable ialah abstrak tabel yang berfungsi untuk mengambil nilai dari kolom dan tabel dan tipe data yang telah didapatkan dari proses parsing. Tipe data perlu diperhatikan karena ada batasan tipe data berupa *text* yang tidak bisa dijadikan sebagai index dalam MariaDB.

```
public class NewIndexTable extends AbstractTableModel {
    private int colnum2=3;
    private int rownum2;
    private String[]
colNames2={"TABLE_NAME","COLUMN_NAME","DATA_TYPE"};
    private  ArrayList<String[]> ResultSets2;
    public NewIndexTable(ResultSet rs2) {
      ResultSets2=new ArrayList<String[]>();
      try{
        while(rs2.next()){
              String[] row={

rs2.getString("TABLE_NAME"),rs2.getString("COLUMN_NAME"
),rs2.getString("DATA_TYPE")
            };
            ResultSets2.add(row);
        }
      }
      catch(Exception e){
          System.out.println("Exception in
CarTableModel");
            }    }
```

*Kode 5.5. Class*New*Index*Table*

Permutation

Proses permutasi berikut ini dilakukan dengan menggunakan dimensional array yang akan memberikan hasil permutasi dari maksimal dua factorial. Sehingga kandidat index yang disarankan hanya sampai pada multi-level-index maksimal dua kolom seperti pada Kode 5.6.

CandidateTable

Candidate table ialah abstrak tabel dari frame Candidate yang berfungsi menampilkan hasil dari permutasi yang telah disimpan dalam tabel temporary bernama tableconf dalam database tujuan seperti pada kode 5.7.

```
private static String[][] permutation(String[]
inputperm) {
        int pnjng = factorial(inputperm.length) /
factorial(inputperm.length - 2);
        String[][] dmns = new String[pnjng][2];
        int temp = 0;
        for (int i = 0; i < inputperm.length; i++) {
            for (int j = 0; j < inputperm.length; j++)
{
                if (inputperm[i] != inputperm[j]) {
                    dmns[temp][0] = inputperm[i];
                    dmns[temp][1] = inputperm[j];
                    temp++;
                }
            }}
        return dmns;
    }

    public static int factorial(int n) {
        int fact = 1; // this  will be the result
        for (int i = 1; i <= n; i++) {
            fact *= i;
        }
        return fact;
    }
```

Kode 5. 6. Class Permutation

```
public class CandidateTable extends AbstractTableModel{
    private int colnum1=3;
    private int rownum1;
    private String[]
colNames1={"TABLENAME","INDEX1","INDEX2"};
    private  ArrayList<String[]> ResultSets;
    public CandidateTable(ResultSet rsconf) {
      ResultSets=new ArrayList<String[]>();
      try{
        while(rsconf.next()){
              String[] row={
rsconf.getString("tablename"),rsconf.getString("index1"
),rsconf.getString("index2")};
            ResultSets.add(row);
         }   }
      catch(Exception e){
          System.out.println("Exception in
CarTableModel");            }
    }
```

Kode 5. 7. Class Candidate Table

PrevPerformance

Sebelum melakukan simulasi penerapan kandidat index, kinerja database dengan menggunakan index sebelumnya perlu dievaluasi sehingga dalam class PervPerformance terdapat dua tipe evaluasi kinerja yaitu berdasarkan responsetime yang akan mengakses method **initresponsetime(),** atau yang menggunakan QPS yang akan mengakses **initqps()**. Evaluasi dilakukan pertabel sesuai dengan cara evaluasi untuk kandidat agar dapat dibandingkan dengan kinerja dari simulasi kandidat. Hasil dari evaluasi class ini sendiri akan disimpan pada tabel sementara yaitu prevperformance.

```
public void initresponsetime() throws SQLException {
  ArrayList<String> namatabel = new ArrayList<>();
ResultSet indexconf = stmt.executeQuery("SELECT TABLENAME FROM
tableconf");
   while (indexconf.next()) {
   namatabel.add(indexconf.getString("tablename"));}
      for (int i = 0; i < namatabel.size(); i++) {
           rt = Responsetime.getInstance();
           float rtime = rt.getAvgrt();
           rt.setAvgrt(0);
           rt.calculateresponsetime(namatabel.get(i));
           rtime = rt.getAvgrt();
           System.out.println("avgrt = " + rtime);
           int insertprev = stmt.executeUpdate("INSERT INTO
prevperformance(tablename, responsetime) VALUES ('" +
namatabel.get(i) + "','" + rtime + "')");
        } }
public void initqps() throws SQLException {
  ArrayList<String> namatabel = new ArrayList<>();
ResultSet indexconf = stmt.executeQuery("SELECT TABLENAME FROM
tableconf");
    while (indexconf.next()) {
    namatabel.add(indexconf.getString("tablename")); }
      for (int i = 0; i < namatabel.size(); i++) {
           qp = Qps.getInstance();
           float qps = qp.getQps();
           qp.setQps(0);
           qp.calculateqps(namatabel.get(i));
           qps = qp.getQps();
           System.out.println("qps = " + qps);
           insertprev = stmt.executeUpdate("INSERT INTO
prevperformance(tablename, qps) VALUES ('" + namatabel.get(i) +
"','" + qps + "')");
        }   }
```

Kode 5. 8. Class PrevPerformance

Qps

Class Qps ialah dasar penilaian menggunakan pengukuran berupa QPS untuk semua penilaian yang memanggil Class ini. Penilaian QPS dilakukan dengan cara melakukan iterasi untuk setiap *query* yang sudah dipisah berdasarkan tabel dengan menggunakan *Pattern Matcher*, sehingga *query* yang tidak ada kaitannya dengan tabel dan kandidat index pada tabel tersebut tidak akan dieksekusi. Eksekusi iterasi dilakukan selama waktu 60 detik untuk setiap query, dan dicatat yang kemudian akan dibagi 60 untuk mendapatkan hasil QPS.

```
for (int i = 0; i < query.length; i++) {

            float loopindex = 0;
            for (long stop = System.nanoTime() +
TimeUnit.SECONDS.toNanos(2); stop > System.nanoTime();)
{
                ResultSet a =
stmt.executeQuery(query[i]);
                loopindex++;
            }
            float queryloop = loopindex / 2;
            totalqueryloop += queryloop;
            qps = totalqueryloop / query.length;
            this.setQps(qps);
        }
```

Kode 5. 9. Class QPS

Responsetime

Prinsip kerja pada Class responsetime hampir sama dengan Class QPS, hanya perbedaan pada penilaian responsetime mengukur nilai waktu yang dibutuhkan oleh sebuah query dengan menggunakan iterasi selama 50 kali yang kemudian akan diambil nilai responsetime rata-rata dari tiap query dan kemudian untuk seluruh query dalam workload diambil rata-rata lagi yang akan menjadi nilai responsetime kandidat tersebut.

```
for (int i = 0; i < query.length; i++) {
            Date start = new Date();
            long start1 = start.getTime();
            for (int j = 0; j < 1; j++) {
                a = stmt.executeQuery(query[i]);
            }
            Date end = new Date();
            long end1 = end.getTime();
            long responsetime = (end1 - start1);
            float totalrt = 0;
            totalrt += responsetime;
            float totalrtquery = totalrt / 1;
            avgrt = totalrtquery / query.length;
            this.setAvgrt(avgrt);
            a.close();
        }
```

Kode 5.10. Class Responsetime

TrialIndexQps

TrialIndexQps ialah class yang akan melakukan simulasi untuk pembuatan index yang memanggil satu persatu kandidat dari tableconf dan kemudian melakukan penilaian QPS untuk setiap kandidat index. Setelah melakukan penilaian maka candidate index yang disimulasikan tadi akan di-*drop* dari database.

```
for (int i = 0; i < namatabel.size(); i++) {
String querycreateidx = null;
if (indexconfarr2.get(i) == null) {
   querycreateidx = "CREATE INDEX index" + i+ " ON " +
namatabel.get(i) + "(" + indexconfarr1.get(i) + ")";
int buatindex = stmt.executeUpdate(querycreateidx);
qp = Qps.getInstance();
float qps = qp.getQps();
qp.getQps();
qp.calculateqps(namatabel.get(i));
qps = qp.getQps();
int rtquery = stmt.executeUpdate("UPDATE tableconf set
qps = " + qps + " WHERE tablename = '" +
namatabel.get(i) + "' AND index1 = '" +
indexconfarr1.get(i) + "' AND index2 IS NULL");
int dropidx = stmt.executeUpdate("DROP INDEX index" + i
+ " ON " + namatabel.get(i));
```

```
} else {
querycreateidx = "CREATE INDEX index" + i + " ON " +
namatabel.get(i) + "(" + indexconfarr1.get(i) + "," +
indexconfarr2.get(i) + ")";
 int buatindex = stmt.executeUpdate(querycreateidx);
 qp = Qps.getInstance();
 float qps = qp.getQps();
 qp.getQps();
 qp.calculateqps(namatabel.get(i));
 qps = qp.getQps();

int rtquery = stmt.executeUpdate("UPDATE tableconf set
qps = " + qps + " WHERE tablename = '" +
namatabel.get(i) + "' AND index1 = '" +
indexconfarr1.get(i) + "' AND index2 = '" +
indexconfarr2.get(i) + "'");

int dropidx = stmt.executeUpdate("DROP INDEX index" +
i + " ON " + namatabel.get(i));
            }
```

*Kode 5.0.11. Class Trial*Index*QPS*

TrialIndexRT

Cara kerja Class TrialIndexRT ialah hampir sama dengan TrialIndexQPS hanya saja penilaian yang dilakukan yaitu dengan menggunakan penilaian dari responsetime yang dipanggil dari Class Responsetime.

```
for (int i = 0; i < namatabel.size(); i++) {
String querycreateidx = null;
if (indexconfarr2.get(i) == null) {
   querycreateidx = "CREATE INDEX index" + i+ " ON " +
namatabel.get(i) + "(" + indexconfarr1.get(i) + ")";
int buatindex = stmt.executeUpdate(querycreateidx);
float rtime = rt.getAvgrt();
rt.setAvgrt(0);
rt.calculateresponsetime(namatabel.get(i));
rtime = rt.getAvgrt();
int rtquery = stmt.executeUpdate("UPDATE tableconf set
responsetime  = " + rtime + " WHERE tablename = '" +
namatabel.get(i) + "' AND index1 = '" +
indexconfarr1.get(i) + "' AND index2 IS NULL");
int dropidx = stmt.executeUpdate("DROP INDEX index" + i
+ " ON " + namatabel.get(i));
```

```
} else {
querycreateidx = "CREATE INDEX index" + i + " ON " +
namatabel.get(i) + "(" + indexconfarr1.get(i) + "," +
indexconfarr2.get(i) + ")";
int buatindex = stmt.executeUpdate(querycreateidx);
float rtime = rt.getAvgrt();
rt.setAvgrt(0);
rt.calculateresponsetime(namatabel.get(i));
rtime = rt.getAvgrt();
int rtquery = stmt.executeUpdate("UPDATE tableconf set
responsetime = " + rtime + " WHERE tablename = '" +
namatabel.get(i) + "' AND index1 = '" +
indexconfarr1.get(i) + "' AND index2 = '" +
indexconfarr2.get(i) + "'");
int dropidx = stmt.executeUpdate("DROP INDEX index" +
i + " ON " + namatabel.get(i));
            }
```

*Kode 5.12. Class Trial*Index*RT*

TestIndex

TestIndex merupakan class yang berfungsi untuk mengimplementasikan hasil index yang sudah terpilih dari tableconf agar diterapkan menjadi kesatuan penerapan untuk evaluasi hasil rekomendasi yang diberikan memiliki kenaikan berapa persen. Class berikut hanya sebagai implement namun penilaian sama seperti cara lain dan hasil penilaian akan melalui query *inserti* ke tabel 'recommendation'.

```
String[] implement = cetak.split("\n");

                for (String implement1 : implement) {
                    try {
                        int implementidx =
stmt.executeUpdate(implement1);
                    } catch (SQLException ex) {
                        System.out.println("cannot
implement");
                    }
```

*Kode 5.13. Class Trial*Index*RT*

SelectedConfTable

Class SelectedConfTable merupakan abstrak tabel dari Frame SelectedConf yang menampilkan kandidat yang terpilih.

```
public class SelectedConfTable extends
AbstractTableModel {
    private int colnum1 = 3;
    private int rownum1;
    private String[] colNames1 =
{"TABLENAME","INDEX1","INDEX2",};
    private ArrayList<String[]> ResultSets;
    public SelectedConfTable(ResultSet rs) {
        ResultSets = new ArrayList<String[]>();
        try {
            while (rs.next()) {
                String[] row = {

rs.getString("tablename"),rs.getString("index1"),rs.ge
tString("index2")
                };
                ResultSets.add(row);
            }
        } catch (Exception e) {        }
```

Kode 5.14. Class SelectedConfTable

SelectivityFactor

Pada selectivity factor ialah proses perhitungan probabilitas dari kemunculan kandidat index yang akan dinilai terhadap *statement* yang ada.

Perhitungan dilakukan dengan menghitung jumlah statement Select yang ada, kemudian menghitung jumlah seluruh kolom yang muncul. Setelah itu dihitung Probabilitas kandidat Index terhadap statement sehingga nilai kemunculan kolom kandidat index dibagi dengan jumlah statement yang ada sebagai P(S|Ij).

jumlah kemunculan kandidat kolom dibagi dengan seluruh kolom yang ada pada workload sebagai P(Ij).

```
public void init(String temp) throws SQLException {

        String queries = kol.getInputawal();
         System.out.println("queries = \n" + queries);
         String a = "\n";

        Pattern pstatement = Pattern.compile(a);
         Matcher mstatement = pstatement.matcher(queries);
         float jmlhstatement = 0;

        while (mstatement.find()) {
             jmlhstatement++;
         }
         ArrayList<String> index = new ArrayList<String>();
         ResultSet r = null;
         r = stmt.executeQuery("select distinct * from
temptable");

         while (r.next()) {
             index.add(r.getString("COLUMN_NAME"));
         }
         float jmlhkandidat = 0;
         for (int i = 0; i < index.size(); i++) {
             Pattern pidx = Pattern.compile(index.get(i));
             Matcher midx = pidx.matcher(queries);
             while (midx.find()) {
                 jmlhkandidat++;
             }
     }
         ResultSet rs = null;

         rs = stmt.executeQuery(querysql);
         ArrayList<String> index1 = new ArrayList<String>();
         ArrayList<String> index2 = new ArrayList<String>();
         ArrayList<String> tabel = new ArrayList<String>();

 while (rs.next()) {
             index1.add(rs.getString("index1"));
             index2.add(rs.getString("index2"));
             tabel.add(rs.getString("tablename"));        }
```

```
        ResultSet rs = null;
        rs = stmt.executeQuery(querysql);
        ArrayList<String> index1 = new
ArrayList<String>();
        ArrayList<String> index2 = new
ArrayList<String>();
        ArrayList<String> tabel = new
ArrayList<String>();
        while (rs.next()) {
            index1.add(rs.getString("index1"));
            index2.add(rs.getString("index2"));
            tabel.add(rs.getString("tablename"));
        }
        for (int i = 0; i < index1.size(); i++) {
            if (index2.get(i) == null) {

            } else {
                Pattern p =
Pattern.compile(index1.get(i));
                Matcher m = p.matcher(queries);
                float jmlhkolom = 0;
                while (m.find()) {
                    jmlhkolom++; }
                float probsi1 = jmlhkolom /
jmlhstatement;
                float ij = jmlhkolom / jmlhkandidat;
                sf1 = probsi1 * ij;
                int insert = stmt.executeUpdate("INSERT
INTO Selectivityfactor(tablename, column_name,
selectivity_factor) VALUES ('" + tabel.get(i) + "','" +
index1.get(i) + "'," + sf1 + ")");
                Pattern p2 =
Pattern.compile(index2.get(i));
                Matcher m2 = p.matcher(queries);
                float jmlhkolom2 = 0;
                while (m2.find()) {
                    jmlhkolom2++;  }
                float probsi2 = jmlhkolom2 /
jmlhstatement;
                float ij2 = jmlhkolom2 / jmlhkandidat;
                sf2 = probsi2 * ij2;
                int insert2 =
stmt.executeUpdate("INSERT INTO
Selectivityfactor(tablename, column name,
selectivity_factor) VALUES ('" + tabel.get(i) + "','" +
index2.get(i) + "'," + sf2 + ")");
            }
```

Kode 5.15. Class SelectivityFactor

EnumerationTable

Class berikut ialah bentuk abstrak table dari Frame Enumeration yang menampilkan hasil nilai selectivity factor perkolom.

```
public class EnumerationTable extends
AbstractTableModel {
private int colnum1=3;
    private int rownum1;
    private String[]
colNames1={"tablename","column_name","Selectivity_fact
or"};
    private  ArrayList<String[]> ResultSets;
    public EnumerationTable(ResultSet rs) {
    ResultSets=new ArrayList<String[]>();
    try{
        while(rs.next()){
              String[]
row={rs.getString("tablename"),rs.getString("column_na
me"),rs.getString("Selectivity_factor")};
            ResultSets.add(row);
       }
      }
      catch(Exception e){
          System.out.println("Exception in
CarTableModel");   }   }
```

Kode 5.16. Class EnumerationTable

5.2.2 Frame:

Login

Pada Frame Login akan didapatkan input berupa nama Database tujuan, beserta user dan password untuk melakukan login dan membuat koneksi. Dimana input tersebut akan diproses dalam Class Connect. Sedangkan untuk ComboBox akan menentukan Frame Parsing yang akan ditampilkan sesuai dengan pilihan yaitu General Log atau Queries. Pada frame ini juga salah satu temporary tabel akan dibuat yaitu tabel temptable.

```
        String inputdb = txt_namadb.getText().toString();
         String inputuser = txt_user.getText().toString();
         String inputpass = txt_pass.getText().toString();
         conn = Connect.getInstance();
         conn.setDB(inputdb);
         conn.setUser(inputuser);
         conn.setPass(inputpass);
         conn.init();
         String pilihan =
cmb_input.getSelectedItem().toString();
         if (pilihan.equals("General Log")) {
             this.setVisible(false);
             Parsinglog l = null;
             try {
                 l = new Parsinglog();
             } catch (SQLException ex) {

Logger.getLogger(Login.class.getName()).log(Level.SEVERE,
null, ex);
             }
             l.setVisible(true);
         } else {
             this.setVisible(false);
             Parsingquery p = null;
             try {
                 p = new Parsingquery();
             } catch (SQLException ex) {

Logger.getLogger(Login.class.getName()).log(Level.SEVERE,
null, ex);
             }
             p.setVisible(true);
         }
         try {
             ResultSet tables = meta.getTables(null, null,
"temptable", null);
             if (!tables.next()) {
                 int i = stmt1.executeUpdate("CREATE TABLE
temptable(TABLE NAME varchar(200) NOT NULL, COLUMN_NAME
varchar(200) NOT NULL, DATA_TYPE varchar(200))");
             }
         } catch (SQLException ex) {

Logger.getLogger(Login.class.getName()).log(Level.SEVERE,
null, ex);
         }
```

Kode 5.17. Frame Login

Retrieve

Pada Frame Retrieve akan menampilkan abstact table dari kandidat kolom hasil parsing yang dicocokkan dengan data yang ada pada informatics.schema pada database dan kemudian disimpan dalam temptable. Selain itu juga menampilkan index yang sudah ada pada database yang terdapat pada tabel yang ada input.

```
    public Retrieve() throws SQLException {
        conn = Connect.getInstance();

        Connection conn1 = conn.getConnect();
        stmt = conn1.createStatement();
        kol = Column.getInstance();

        int deleterow = stmt.executeUpdate("DELETE FROM temptable
WHERE DATA_TYPE = 'text'");
        int previdx = stmt.executeUpdate("CREATE TABLE
previdx(TABLE NAME VARCHAR(50), INDEX_NAME VARCHAR(50),
COLUMN_NAME VARCHAR(50))");
        int nonindx = stmt.executeUpdate("create table
noneindex(TABLE_NAME VARCHAR(50), qps FLOAT(50), responsetime
FLOAT(50))");
        initComponents();
    }
    public ResultSet getResultFromCars() {
        ResultSet rs1 = null;
        String sqldb = conn.getDb();
        String[] cobatabel = kol.getOutputTable();
        String temptabel = "";
        for (int i = 0; i < cobatabel.length; i++) {
            if (i != 0) {
                temptabel += ",";
            }
            String string = cobatabel[i].trim();
            temptabel += "'" + string + "'";
        }
        String temptabel2 = temptabel.replace("`", "");
        String queryprev = "Select TABLE_NAME, INDEX_NAME,
COLUMN_NAME from information_schema.statistics where
table_schema='" + sqldb + "' AND TABLE_NAME IN (" + temptabel2 +
")";
        try {
            rs1 = stmt.executeQuery(queryprev);
        } catch (SQLException e) {
        }
        return rs1;
    }
```

```
    public ResultSet getResultFromCars2() {
        ResultSet rs2 = null;
        try {
            rs2 = stmt.executeQuery("SELECT * FROM
temptable");

        } catch (SQLException ex) {

Logger.getLogger(Retrieve.class.getName()).log(Level.SEVER
E, null, ex);
        }
        return rs2;
    }
private void
jButton1ActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event_jButton1ActionPerformed

        try {
            int tableconf = stmt.executeUpdate("CREATE
TABLE tableconf(tablename VARCHAR(50), index1 VARCHAR(50)
NOT NULL, index2 VARCHAR(50), qps FLOAT(50), responsetime
FLOAT(50))");
        } catch (SQLException ex) {

Logger.getLogger(Retrieve.class.getName()).log(Level.SEVER
E, null, ex);
        }
        this.setVisible(false);
        Candidate a = null;

        try {
            a = new Candidate();
        } catch (SQLException ex) {

Logger.getLogger(Retrieve.class.getName()).log(Level.SEVER
E, null, ex);
        } catch (IOException ex) {

Logger.getLogger(Retrieve.class.getName()).log(Level.SEVER
E, null, ex);
        }
        a.setVisible(true);
}
```

Kode 5.18. Frame Retrieve

Candidate

Pada frame candidate hal yang dijalankan pertamkali ialah Permutasi dari Class Permutation yang kemudian hasilnya akan disimpan dalam tabel temporary yaitu tableconf dan ditampilkan pada abstract tabel. Pada frame ini pula kegiatan evaluasi berdasarkan pilihan ComboBox berupa QPS atau Responsetime. Evaluasi dimulai dengan kondisi database dengan index yang kini sudah ada dan kemudia hasil evaluasi disimpan pada tabel Prevperformance. Kemudian dilanjutkan dengan melakukan simulasi index untuk setiap kandidat yang ada pada tableconf. Simulasi tersebut dilakukan dengan memanggil class TrialIndexQPS atau TrialIndexRT. Setelah mempunyai hasil evaluasi dari kondisi sebelum dan sesudah simulasi, maka akan dibandingkan kinerja yang terbaik. Apabila kinerja database dengan index yang sebelumnya sudah ada lebih baik dari hasil kinerja dari kandidat configurasi pada tableconf, maka frame yang muncul setelah ini ialah Previsbetter. Namun apabila configurasi memiliki kinerja yang lebih baik maka akan menuju Frame SelectedConf.

```
public Candidate() throws SQLException, IOException {
        conn = Connect.getInstance();
        Connection conn1 = conn.getConnect();
        stmt = conn1.createStatement();
        kol = Column.getInstance();
        ti = TrialIndexQPS.getInstance();
        tirt = TrialIndexRT.getInstance();
        initComponents();
        ni = NoneIndex.getInstance();

               perm = Permutation.getInstance();
        perm.permutasi(); //melakukan proses permutasi

    }
public ResultSet getResultFromCars() {
        ResultSet rsconf = null;
        try {
            rsconf = stmt.executeQuery("Select
TABLENAME, INDEX1, INDEX2 from tableconf");
        } catch (SQLException ex) {

Logger.getLogger(Candidate.class.getName()).log(Level.SE
VERE, null, ex);
```

```
try {
int createprev = stmt.executeUpdate("create table
prevperformance(tablename VARCHAR(50),"
+ "qps float(50),responsetime float(50))");
} catch (SQLException ex) {
Logger.getLogger(SelectedConf.class.getName()).log(Leve
l.SEVERE, null, ex);
        }
//evaluasi previous index perfromance
  String cmb = cmb_ev.getSelectedItem().toString();
if (cmb.equals("QPS")) {
   try {
      pp = PrevPerformance.getInstance();
      pp.initqps();
   } catch (SQLException ex) {
Logger.getLogger(Candidate.class.getName()).log(Level.S
EVERE, null, ex);
  } catch (IOException ex) {
Logger.getLogger(Candidate.class.getName()).log(Level.S
EVERE, null, ex); }
 //proeses evaluasi performa semua kandidat
    try {
        ti.init();
       tix.init("QPS");
       ti.sf();
       pp.initqpsnew();
       tix.dropcandidate("QPS");
   } catch (SQLException ex) {
Logger.getLogger(Candidate.class.getName()).log(Level.S
EVERE, null, ex);
            }
try {
ArrayList<String> avgqpsprev = new ArrayList<>();
ResultSet tempprevqps;
tempprevqps = stmt.executeQuery("select avg(qps) as
avgqps from prevperformance");
while (tempprevqps.next()) {
avgqpsprev.add(tempprevqps.getString("avgqps"));}
String qpsprev = "";

for (int i = 0; i < avgqpsprev.size(); i++) {
qpsprev += avgqpsprev.get(i).toString();}
float prevqps = Float.valueOf(qpsprev);
ArrayList<String> avgqpscan = new ArrayList<>();
ResultSet tempcanqps;
tempcanqps = stmt.executeQuery("select avg(qps) as
avgqps from tableconf");
```

```
for (int i = 0; i < avgqpscan.size(); i++) {
    qpscan += avgqpscan.get(i).toString();
           }
    float canqps = Float.valueOf(qpscan);
//kalau prevresponsetime lebih kecil
   float presentaseqps = (canqps / prevqps) * 100;
   this.SetPersen(presentaseqps);

 if (prevqps >= canqps) {
     Previsbetter pib = new Previsbetter();
     pib.setVisible(true)
  } else {
  SelectedConf sc;
    try {
    sc = new SelectedConf("QPS");
    sc.setPrevMethod("QPS");
    sc.setVisible(true);
    } catch (SQLException ex) {
Logger.getLogger(Candidate.class.getName()).log(Level.S
EVERE, null, ex);
    } catch (IOException ex) {
Logger.getLogger(Candidate.class.getName()).log(Level.S
EVERE, null, ex);
                    }
                }
   } catch (SQLException ex) {
Logger.getLogger(Candidate.class.getName()).log(Level.S
EVERE, null, ex);
            }
```

Kode 5.19. Frame Candidate

Previsbetter

Apabila kinerja dengan index sebelumnya paling optimal maka sistem tidak akan melanjutkan pemilihan index dan melakukan DROP tabel sementara yang telah dibuat pada proses sebelumnya.

```
int a = stmt.executeUpdate("DROP TABLE temptable");
int b = stmt.executeUpdate("DROP TABLE tableconf");
int c = stmt.executeUpdate("DROP TABLE
prevperformance");
int d = stmt.executeUpdate("DROP TABLE previdx");

System.exit(0);
```

Kode 5.20. Frame Previsbetter

SelectedConf

SelectedConf ialah Frame yang menampilkan kandidat untuk tiap tabel dengan kinerja paling baik saat simulasi. Pada Frame ini juga akan diidentifikasi apabila ada Configurasi terpilih yang memiliki index2, atau masih berupa multi-level-index maka akan dilakukan proses perhitungan Selectivity factor pada candidate itu saja. Kemudian hasilnya akan disimpan pada table selectivityfactor.

```
public ResultSet getResultFromCars() {
        String sqlquery;
        System.out.println(getPrevMethod());
        if (getPrevMethod().equals("QPS")) {
            sqlquery = qpsquery;
        } else {
            sqlquery = rtquery;         }
        try {
            rs = stmt.executeQuery(sqlquery);
        } catch (SQLException ex) {
Logger.getLogger(Candidate.class.getName()).log(Level.S
EVERE, null, ex);
        }         return rs;
    }
try {
while (temp.next()) {
tempindex.add(temp.getString("index2"));
                }
for (int i = 0; i < tempindex.size(); i++) {
     if (tempindex.get(i) == null) {
    System.out.println("do nothing in selectivity
factor");
     } else {
     sf.init("QPS");
     }     }
  } catch (SQLException ex) {
Logger.getLogger(SelectedConf.class.getName()).log(Leve
l.SEVERE, null, ex); }
    Enumeration enumer;
     try {
     enumer = new Enumeration("QPS");
     enumer.setVisible(true);
     this.setVisible(false);
     } catch (SQLException ex) {
Logger.getLogger(SelectedConf.class.getName()).log(Leve
l.SEVERE, null, ex);
```

Kode 5.21. Frame SelectedCandidate

Enumeration

Proses pencacahan pada Frame enumeration sendiri dipermudah dengan penyimpanan pada table sementara yang sudah terpisah pisah bagi setiap kandidat kolom. Sehingga untuk melakuan pencacahan dari index yang berjenis multi-level-index ¸ proses yang dilakukan ialah memanggil kandidat yang telah dihitung selectivity factor dari tiap kolom pada tiap tabel untuk dipilih menajadi rekomendasi. Namun apabila dalam satu tabel masing-masin gkolomnya memiliki nilai selectivity factor yang sama maka tetap akan menjadi multi-level-index.

```
public ResultSet getResultFromCars() {
        ResultSet rs = null;
        try {
            rs = stmt.executeQuery("select * from
selectivityfactor");
        } catch (SQLException ex) {

Logger.getLogger(Candidate.class.getName()).log(Level.
SEVERE, null, ex);
        }
        return rs;
    }
```

Kode 5.22. Frame Enumeration

GenerateIndex

Frame Genetrate Index merupakan Frame terakhir pada sistem yang akan menjadikan kandidat index yang terpilih menjadi sebuah Query Create Index yang direkomendasikan dan dapat diimplementasikan langsung pada database. Pada frame ini pula dapat diketahui presentase peningkatan kinerja berdasarkan penilaian yang telah dipilih yaitu QPS atau Responsetime apabila index yang telah direkomendasikan tersebut diimplementasikan. Ketika sistem ditutup maka akan *Drop* tabel-tabel sementara yang telah dibuat dalam proses untuk merekomendasikan index.

```
String tempqps = "select * from tableconf where
(tablename, qps) in ( select tablename, max(qps) from
tableconf group by tablename) group by tablename";
        String temprt = "select * from tableconf where
(tablename, responsetime) in ( select tablename,
min(responsetime) from tableconf group by tablename)
group by tablename";

String sqlquery = "";

if (getPrevMethod().equals("QPS")) {
  sqlquery += tempqps;
  try {
  ArrayList<String> avgqpsprev = new ArrayList<>();
  ResultSet temprevqps = stmt.executeQuery("select
avg(qps) as avgqps from prevperformance");

while (temprevqps.next()) {
avgqpsprev.add(temprevqps.getString("avgqps"));}

String qpsprev = "";

for (int i = 0; i < avgqpsprev.size(); i++) {
qpsprev += avgqpsprev.get(i).toString();
float prevqps = Float.valueOf(qpsprev);
ArrayList<String> avgqpscan = new ArrayList<>();
ResultSet tempcanqps = stmt.executeQuery("select
avg(qps) as avgqps from tableconf");

while (tempcanqps.next()) {
avgqpscan.add(tempcanqps.getString("avgqps"));}
String qpscan = "";

for (int i = 0; i < avgqpscan.size(); i++) {
   qpscan += avgqpscan.get(i).toString();   }
   float canqps = Float.valueOf(qpscan);
   float presentaseqps=((canqps-prevqps)/prevqps)*100;

 String settext = Float.toString(presentaseqps);
 txt_persen.setText("Performance increasing " +
settext + " %");

 } catch (SQLException ex) {
Logger.getLogger(GenerateIndex.class.getName()).log(Le
vel.SEVERE, null, ex);
```

*Kode 5.23. Frame Generate*Index

Halaman ini sengaja dikosongkan.

BAB VI
HASIL DAN PEMBAHASAN

Pada bab hasil dan pembahasan akan menjelaskan mengenai hasil penelitian dan pembahasan dari hasil penelitian yang telah dilakukan.

6.1 Hasil

Bagian ini merupakan hasil inplementasi yang dilakukan dari pengujian menggunakan dua database pengujian Monitordom, dan Adventureworks2012. Pada penerapannya, database Monitordom menjadi bahan pengujian penelitian dengan input berupa General Log dan pengukuran menggunakan r*esponsetime*. Sedangkan Adventureworks2012 menjadi bahan pengujian penelitian dengan input berupa kumpulan *query* dan pengukuran menggunakan QPS.

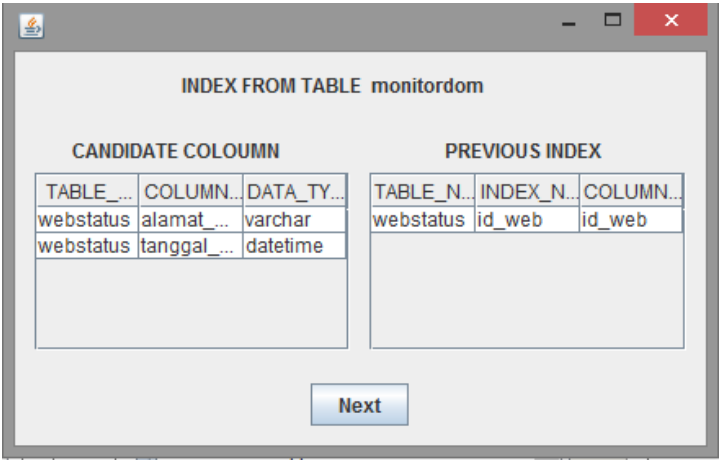
6.1.1 Monitordom

Pada saat menajalankan sistem rekomendasi index menggunakan *responsetime*, setiap *query* yang dieksekusi diiterasi selama 50 kali untuk mendapatkan kestabilan rata-rata *responsetime*. Kestabilan diuji dengan melakukan pengujian beberapa kali dengan kondisi lingkungan yang sama. Dengan iterasi 50 kali, sistem dapat memberikan rekomendasi yang konsisten.

Pada saat parsing yang didapatkan ialah beberapa kandidat kolom pada gambar 6.1 dan index sebelumnya yang sudah ada. hasil dari kandidat tersebut pun akan disimpan pada temptable untuk selanjutnya menjadi input proses permutasi yang dilakukan untuk mendapatkan konfigurasi Index. Hasil parsing tersebut hanya ada satu jenis tabel dengan dua nama kolom yang memiliki tipe data varchar dan decimal.

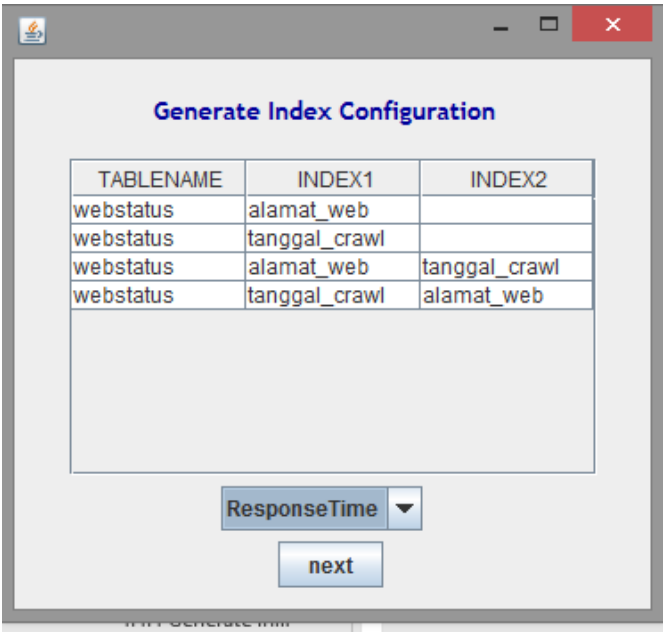
Ketika dilakukan parsing, sebenarnya ada satu kolom kandidat yang telah tereliminasi terdahulu sebelum ditampilkan menjadi kandidat yaitu kolom ‘crawled_link’ yang memiliki tipe data *text.* Pertimbangan untuk melakukan eliminasi pada kolom bertipe data *text* ialah karena pada database MariaDB tidak

mendukung Index dengan tipe data *text* sehingga kandidat tersebut nantinya tidak akan bias melewati proses evaluasi yang dilakukan dengan cara mensimulasikan satu persatu kandidat konfigurasi index pada database.



Gambar 6.1 . Hasil parsing Monitordom

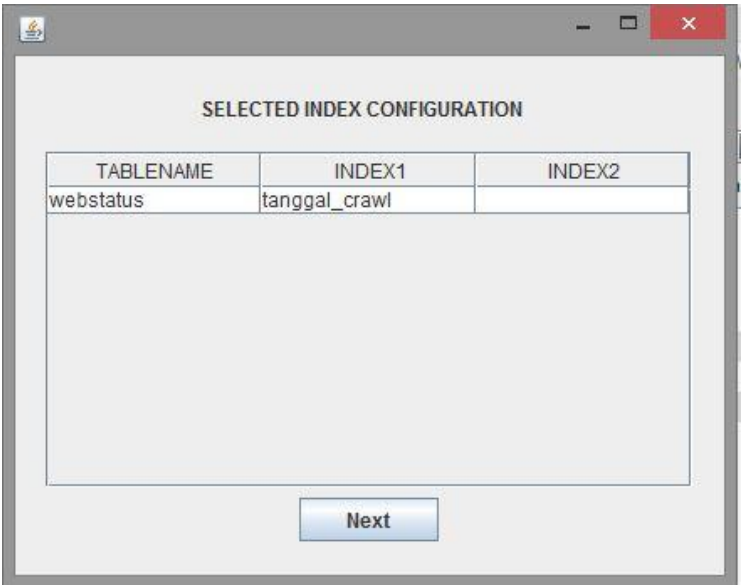
Setelah dilakukan permutasi dan hasilnya akan disimpan pada tableconf sebagai kandidat Konfigurasi Index yang akan direkomendasikan setelah mendapatkan hasil evaluasi *responsetime* seperti pada gambar 6.2. Namun sebelum ditentukan kandidat yang terbaik dari Konfigurasi, terlebih dahulu *responsetime* yang ada pada tableconf akan dibandingkan dengan *responsetime* pada tabel prevperformance.



Gambar 6.2. Konfigurasi Index *Monitordom*

Selanjutnya ialah mendapatkan kandidat Konfigurasi Index dengan *responsetime* paling sedikit untuk setiap tabel. Pada percobaan berikut kandidat konfigurasi yang terpilih ialah pada baris kedua yaitu 'tanggal_crawl'. Data dari hasil pemilihan index berikut tersedia pada lampiran C bagian Monitordom percobaan 1.

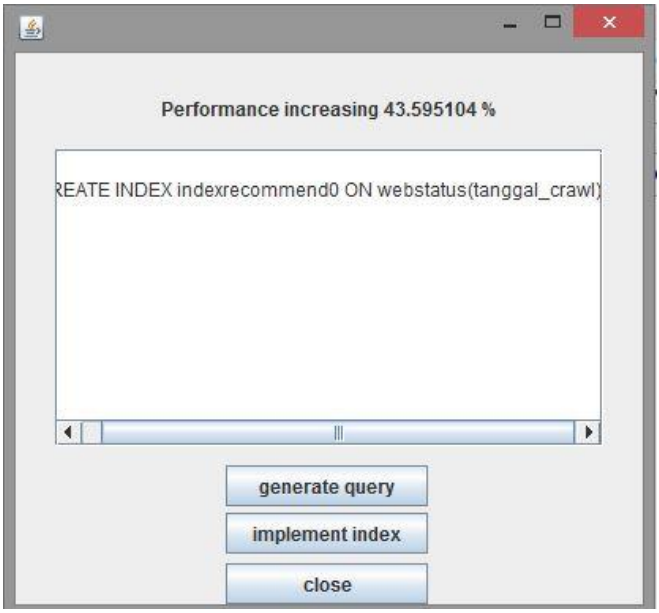
Konfigurasi yang terpilih ditampilkan seperti pada gambar 6.3 dan karena hasil Konfigurasi Index yang memiliki responsetime paling sedikit bukan multi-level-index maka proses penilaian *Selectivity factor* dan pencacahan tidak akan dilakukan. Sedangkan peningkatan kinerja sendiri akan dibahas pada sub bab Pembahasan.



Gambar 6.3. Konfigurasi Index *Monitordom yang terpilih*

Karena tidak melakukan pencacahan maka pada halaman tabel *selectivity factor* akan kosong dan tidak melakukan perhitungan.

Setelah melewati pencacahan namun tidak melakukan eksekusi apapun, maka menuju proses dimana rekomendasi index akan ditampilkan sebagai *query* yang dapat diimplementasikan langsung pada database Monitordom. Apabila database tersebut menerapkan index yang direkomendasikan maka *responsetime*-nya akan lebih baik sebesar sebesar **42,5595104%** seperti pada gambar 6.4.



Gambar 6.4. Hasil rekomendasi Index *Monitordom*

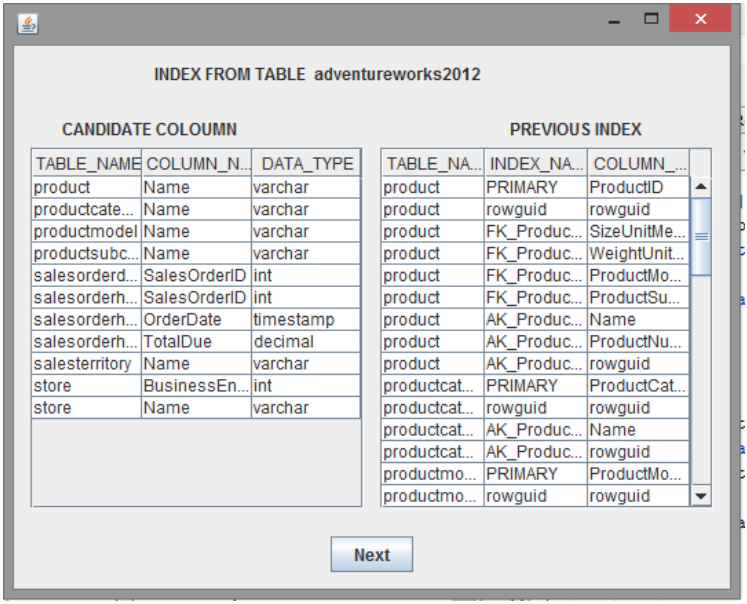
6.1.2 Adventureworks2012

Pada pengujian yang dilakukan pada database Adventurewroks2012 yang menggunakan pengukuran QPS, waktu pengujian yang dilakukan untuk menguji sebuah statement ialah 60 detik untuk mendapatkan kestabilan perhitungan QPS. Waktu 60 detik tersebut didapatkan dari pengujian setiap *query* yang menjadi input dan mencari waktu yang lebih besar dari pada waktu yang dibutuhkan untuk eksekusi *query* tersebut dimana waktu maksimal dari *query* tersbut ialah sekitar 22 detik.

Hasil yang didapatkan pada saat menjalankan aplikasi rekomendasi index untuk database Adventureworks2012 dengan melakukan lima kali percobaan ialah hasil rekomendasi peningkatan yang signifikan yaitu tidak ada yang melebihi 3% dari kinerja Index yang memang sudah ada pada database AdventureWorks2012 tersebut.

Salah satu hasil percobaan menggunakan database AdventureWorks2012 yang memberikan rekomendasi tersebut ialah seperti pada gambar 28 yang merupakan hasil *parsing* dari *query* yang diinput dan kolom dari hasil *parsing* tersbut diproses menjadi konfigurasi index pada gambar 6.5 yang kemudian konfigurasi index tersebut akan dievaluasi satu-persatu berdasarakan pengukuran berupa QPS dan akan dipilih konfigurasi terbaik dari tiap tabel.

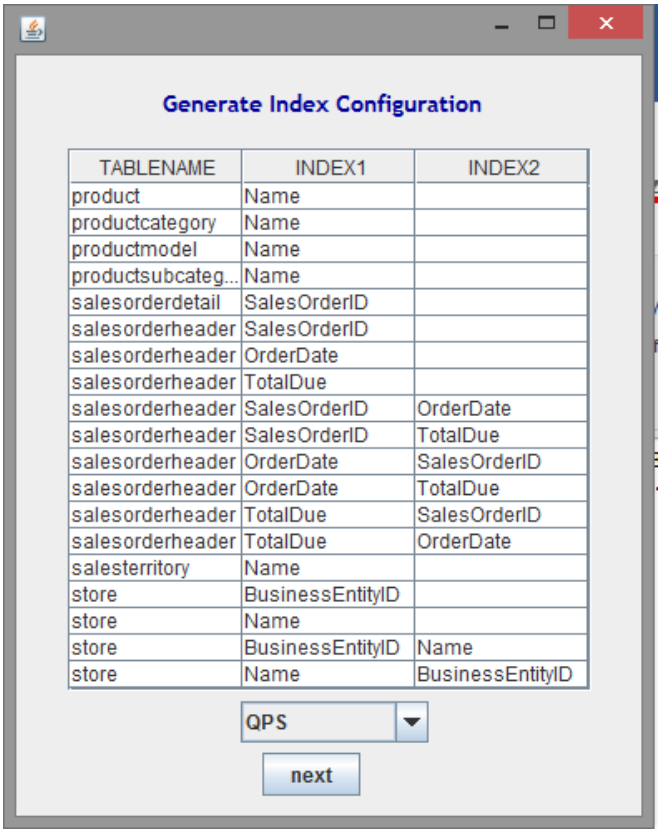
Hasil parsing tersebut berupa tujuh tabel yang berbeda dengan jumlah nama kolom sebanyak 11 kolom dengan tipe data berupa variasi antara varchar, integer , timestamp, dan decimal.



Gambar 6.5. Hasil parsing AdventureWorks

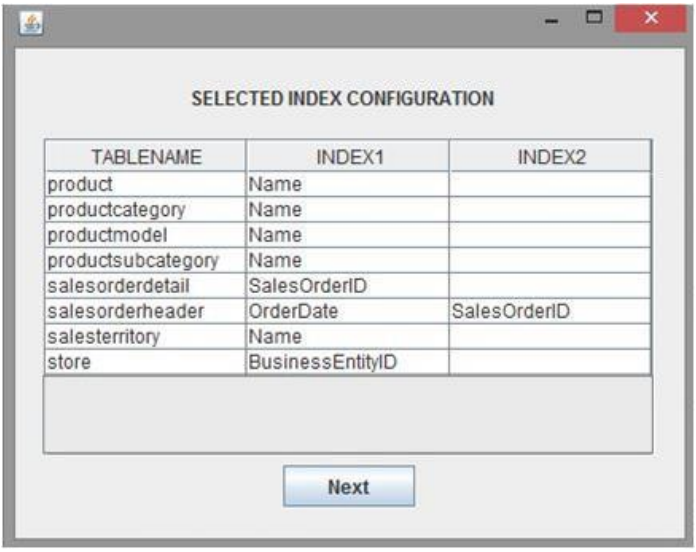
Hasil Parsing pada gambar 6.6 dilakukan pembuatan kandidat konfigurasi index dari hasil proses permutasi berdasarkan masing-masing tabel sehingga didapatkan 19 kandidat konfigurasi seperti pada gambar 6.7. Setelah menjadi sebuah konfigurasi, maka evaluasi kinerja pada database

AdventureWorks2012 dilakukan dengan penilaian berdasarkan QPS dan dipilih untuk kandidat konfigurasi dengan QPS yang terbaik dari masing-masing tabel.

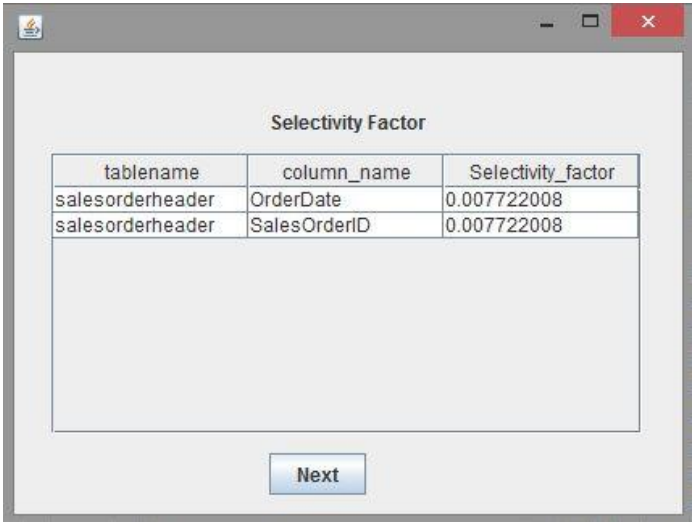


Gambar 6.6. Konfigurasi Index *AdventureWorks*

Dari seluruh konfigurasi yang terbaik pada tiap tabel seperti gambar 30, pada tabel salesorderheader mempunyai konfigurasi multi-level-index yang berarti akan melakukan proses pencacahan dengan *selectivity factor* seperti pada gambar 6.8.



Gambar 6.7. Konfigurasi yang terpilih pada AdventureWorks



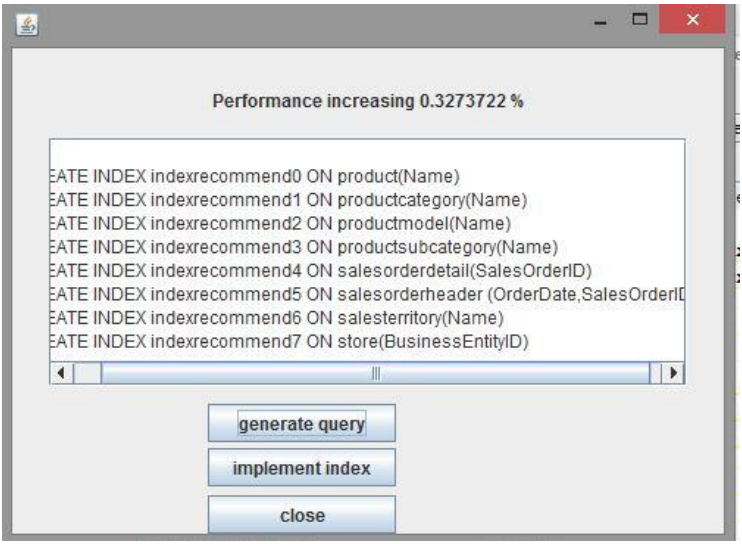
Gambar 6.8. Proses SelectivityFactor pada AdventureWorks

Hasil pencacahan menunjukkan bahwa probablitas kemunculan kolom OrderDate dan SalesOrderID terhadap *statement* yang dalam kasus berikut berupa *query* yang menjadi input aplikasi adalah sama yaitu masing-masing 0.007722008.

Kesamaan probabilitas tersebut menjadikan kedua kolom tersebut berarti bahwa banyaknya kemunculan kolom OrderDate dan SalesOrderID memiliki bobot yang sama dan rekomendasi untuk index pada salesorderheader akan tetap menjadi multi-level-index.

Kinerja dari database AdventureWorks2012 akan meningkat sebanyak **0.032%** apabila menggunakan hasil rekomendasi seperti pada gambar 6.9.

Hasil tersebut tidak terlalu signifikan bila diterapkan karena pada dasarnya database AdventureWorks2012 telah memiliki index yang tepat untuk kebutuhan data mereka.



Gambar 6.9. Hasil rekomendasi index *AdventureWorks*

6.2. Pembahasan

Pada bagian berikut ini akan membahas mengenai analisa dari hasil pengujian pada aplikasi yang telah dilakukan dengan menggunakan dua database sampel. Pembahasan dilakukan dengan menggunakan pendekatan verifikasi dan validasi.

6.2.1 Pengujian Monitordom

Verifikasi
Untuk memastikan bahwa fungsionalitas dari aplikasi telah berjalan maka dilakukan uji coba penggunaan aplikasi pada database Monitordom. Setelah dilakukan ujicoba seluruh fungsi dari aplikasi dapat berjalan sesuai dengan kebutuhan fungsional mulai dari melakukan login yang hingga hasil rekomendasi index yang dapat di implementasikan langsung pada database. Hasil uji coba berupa gambar aplikasi ketika dijalankan tersedia pada lampiran B bagian Monitordom.

Validasi
Pembahasan pada data hasil pengujian pada penelitian ini dilakukan sekaligus untuk melaukan validasi. Validasi dilakukan dengan menganalisa data pengujian yang dilakukan selama lima kali untuk tiap database, untuk memastikan bahwa aplikasi telah memberikan rekomendasi yang benar.

Rekomendasi sendiri dapat dikatakan benar apabila yang terpilih memang hasil dengan kinerja yang paling baik. Berdasarkan data selama lima kali pengujian, hasil rekomendasi index untuk database Monitordom telah konsisten yaitu untuk membuat index dengan kolom ‘tanggal_crawl’ pada tabel ‘webstatus’ dengan kenaikan kinerja yang dapat dilihat pada tabel 6.1 yang berupa ringkasan hasil keseluruhan pengujian dari data pengujian yang terdapat pada Lampiran C bagian Monitordom.

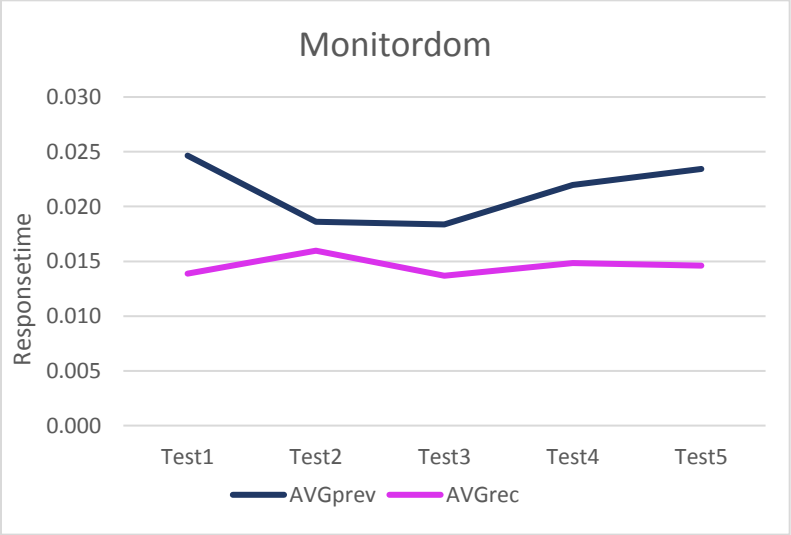
AVGprev ialah rata-rata *responsetime* tanpa index yang direkomendasikan dan AVGrec ialah rata-rata *responsetime* yang didapatkan setelah mensimulasikan penggunaan index yang akan direkomendasikan. Rata-rata peningkatan kinerja

berdasarkan *responsetime* pada database Monitordom selama lima kali percobaan yaitu meningkat sebanyak **30.712%.**

Tabel 6.1. Hasil pengujian pada Monitordom

	AVGprev	AVGrec	increasing
Test1	0.025	0.014	43.652
Test2	0.019	0.016	14.219
Test3	0.018	0.014	25.500
Test4	0.022	0.015	32.475
Test5	0.023	0.015	37.716
AVG	**0.021**	**0.015**	**30.712**

Keseluruhan pengujian untuk Monitordom mengalami peningkatan dari lima percobaan yang ada seperti pada grafik yang ditunjukkan pada gambar 6.10.

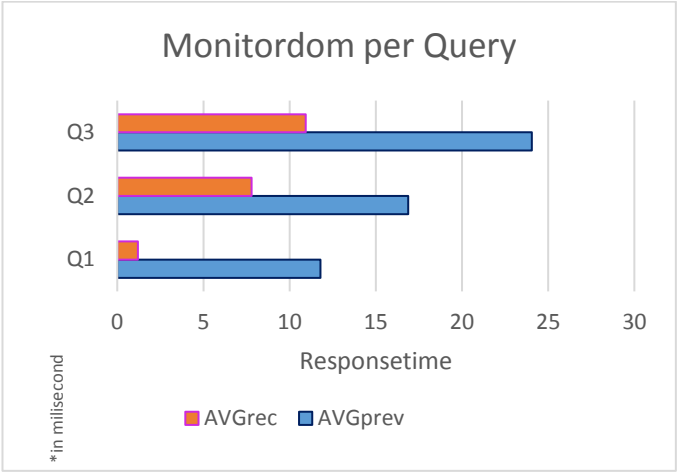


Gambar 6.10. Grafik kinerja Monitordom

Peningkatan kinerja hingga 43,65% pada keseluruhan input dai database Monitordom akan dibahas lebih dalam menggunakan peningkatan kinerja per *query* yang diwakilkan oleh tiga pola *query* SELECT yang ada pada input Monitordom. Hasil uji coba kinerja pada query tersebut dapat dilihat pada Tabel 6.2. dimana AVGprev merupakan rata-rata kinerja pada saat belum menerapkan index hasil rekomendasi dan AVGrec merupakan rata-rata hasil kinerja setelah menggunakan index dari rekomendasi perangkat lunak. Pengukuran tersebut diukur menggunakan satuan waktu millisecond dan dapat dilihat pada Gambar 6.11 bahwa peningkatan tertinggi pada Q3 yang merupakan query pola ketiga yaitu *query* dengan kolom 'tanggal-crawl' yang menjadi hasil rekomendasi index dengan penjelasan beberapa keterangan menurut *query* berdasarkan operator yang ada yaitu ORDER BY, GROUP BY, dan COUNT.

Tabel 6.2 Pengujian Query Monitordom

	Tabel	Jumlah Kolom	Keyword	Operator	AVGprev	AVGrec
Q1	webstatus	1	ORDER BY, DESC, LIMIT	-	11.78	1.20
Q2	webstatus	1	COUNT, DISTINCT	-	16.88	7.80
Q3	webstatus	1	DISTINTC, ORDER BY, ASC, LIMIT	-	24.06	10.94



Gambar 6.11. Grafik kinerja per Query Monitordom

Dengan menggunakan hasil rekomendasi index dari aplikasi yang telah menerapkan prisip heuristik pada java, kinerja dari database Monitordom dapat meningkat hingga **43,65%** dengan waktu untuk memberikan rekomendasi sekitar 59 menit Monitordom ialah database dengan kondisi index hanya sebagai *foreign key* dan belum memiliki index untuk mendukung operasional database.

6.2.2 Pengujian Adventureworks

Verifikasi

Pada database AdventureWorks juga dilakukan verifikasi seperti pada Monitordom, untuk memastikan seluruh fungsionalitas dari aplikasi berjalan dengan benar. Aplikasi dapat dikatakan berjalan dengan benar apabila dapat melakukan proses mulai dari login hingga memberikan hasil rekomendasi.

Pada AdventureWokrs aplikasi telah berjalan dengan benar dari lima kali pengujian menunjukkan kenaikan yang tidak signifikan tapi telah melalui seluruh proses yang dimiliki aplikasi. Dokumentasi dari salah satu pengujian tersebut tersedia pada Lampiran B bagian AdventureWorks.

Validasi

Analisa pada AdventureWorks juga dilakukan sekaligus untuk melakukan validasi apakah rekomendasi yang diberikan sudah sesuai seperti pada percobaan Monitordom.

Hasil percobaan selama lima kali dari rekomendasi berikut memberikan hasil yang sedikit berbeda dari Monitordom dari segi kenaikan kinerja. Pengujian yang dilakukan dengan menggunakan pengukuran QPS memberikan hasil rekomendasi dengan peningkatan kinerja tidak lebih dari 3% dari kinerja Index awal yang dimiliki AdventureWorks seperti pada tabel 6.3 yang merupakan hasil ringkasan data selama lima kali pengujian, bahkan pada percobaan ke empat mengalami penurunan kinerja jika menerapkan index yang direkomendasikan.

AVGprev sebagai rata-rata kenaikan kinerja dengan menggunakan index yang dimiliki AdventureWorks sebelumnya dan AVGrec merupakan kinerja dengan menggunakan tambahan index rekomendasi dan increase merupakan presentase kenaikan setelah adanya rekomendasi. Hasil kenaikan rata-rata seluruh percobaan memberikan angka **1.286 %** kenaikan dari hasil penggunaan rekomendasi index.

Tabel 6.3. Hasil pengujian pada AdventureWorks

	AVGprev	AVGrec	Increase
Test1	2.406	2.450	1.841
Test2	2.485	2.554	2.777
Test3	2.437	2.453	0.666
Test4	2.432	2.420	-0.481
Test5	2.531	2.571	1.588
AVG	**2.458**	**2.490**	**1.286**

Kenaikan yang tidak signifikan tersebut dikarenakan AdventureWorks merupakan database yang memiliki index yang sudah sesuai dengan kebutuhannya. Hal tersebut juga

dibuktikan pada dokumentasi *Data Dictionary* AdventureWorks2012 [26] mengenai index apa saja yang telah dimiliki.

Hasil rekomendasi yang diberikan telah banyak yang sudah menjadi Index dari AdventureWokrs seperti contoh pada tabel Product yang direkomendasikan ialah kolom Name, sedangkan AdventureWorks tabel Product sudah mempuyai index bernama AK_Product_Name yang merupakan index non-*cluster* untuk kolom Name. Index dengan kolom yang sama tersebut menyebabkan kinerja menurun dikareakan index baru yang dibuat tidak malah membantu pencarian namun membebani kerja database seperti yang dibuktikan pada grafik pada gambar 6.12 yang menunjukkan bahwa kinerja dari hasil saat mengimplementasikan rekomendasi index tidak mempunyai peningkatan yang signifikan untuk QPS bahkan dapat mengalami penurunan seperti pada pengujian ke empat. Penurunan kinerja tersebut dapat disebabkan oleh adanya index yang direkomendasikan semakin membebani kinerja dari database dikarenakan database Adventureworks2012 sudah memiliki index dengan nama kolom seperti yang direkomendasikan oleh perangkat lunak berikut.

Gambar 6.12 Grafik kinerja AdventurWorks2012

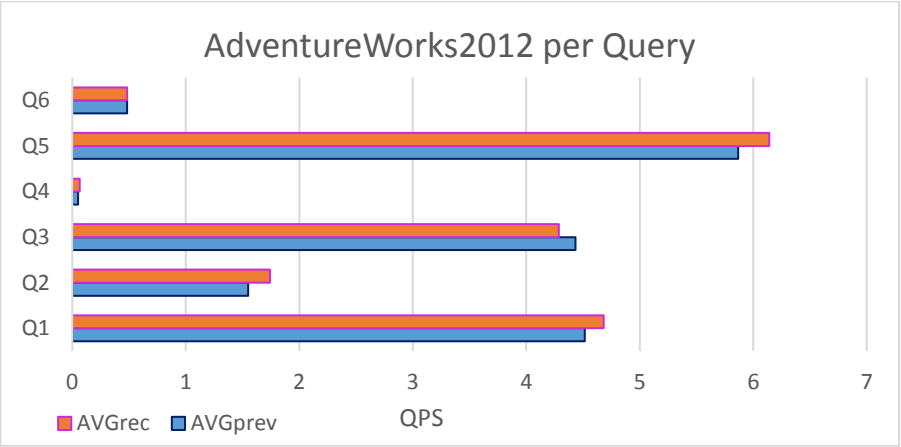
Pengujian juga dilakukan pada setiap *query* input dari Adventureworks2012. Pengujian dilakukan dengan mengamati setiap keningkatan pada setiap query yang ada seperti pada Tabel 6.4 dimana Q1 ialah nomer dari urutan *query* seperti pada Lampiran A, sedangkan AVGprev ialah rata-rata QPS pada query tersebut dan AVGrec merupakan rata-rata QPS setelah menerapkan rekomendasi dari perangkat lunak. Hasil tersebut menunjukkan bahwa tidak ada terlalu banyak peningkatan kinerja pada setiap *query* bahkan ada yang menurun pada Q3 dan tidak terjadi perubahan pada Q6.

Pembuktian peningkatan kinerja yang tidak terlalu signifikan untuk setiap *query* tersebut dapat dilihat pada Gambar 6.13 dimana kinerja setelah menggunakan rekomendasi bahkan hampir sejajar dengan kinerja sebelum menggunakan rekomendasi. Keterangan dari setiap *query* menurut ORDER BY, GROUP BY, dan COUNT.

Tabel 6.4. Pengujian Query Adventureworks2012

	Tabel	Jumlah Kolom	Keywords	Operator	AVGprev	AVGrec
Q1	store, salesorderheader,	2	GROUP BY, ORDER BY, DESC.	-	4.517	4.683
Q2	store, salesorderheader, salesterritorry	4	GROUP BY, ORDER BY, DESC.	-	1.550	1.741
Q3	store, salesorderheader, salesterritorry	3	GROUP BY, ORDER BY, DESC.	-	4.433	4.286
Q4	product, productmodel, productsubcategory, productcategory, salesorderdetail,	5	GROUP BY, ORDER BY, ASC	-	0.050	0.067

	salesorderheader, store					
Q5	store, salesorderheader	2	GROUP BY, ORDER BY , ASC	-	5.867	6.142
Q6	store, salesorderheader,	2	COUNT, ORDER BY, GROUP BY, DESC	-	0.483	0.483



Gambar 6.13 Grafik kinerja per Query Adventureworks2012

Kesimpulan untuk hasil uji coba menggunakan database AdventureWorks2012 kenaikan kinerja yaitu **1.285%** dengan waktu untuk memberikan rekomendasi sekitar 349 menit sehingga rekomendasi Index tersebut tidak layak diimplementasikan pada database. Hal tersebut dikarenakan karena AdventureWorks telah memiliki index yang dibutuhkan untuk kegiatan operasional database bahkan hampir seluruh hasil rekomendasi index yang ada telah dimiliki oleh AdventureWorks.

Halaman ini sengaja dikosongan.

BAB VII
KESIMPULAN DAN SARAN

Pada bab kesimpulan dan saran yang merupakan bab terakhir dari penelitian ini akan menjelaskan mengenai hasil kesimpulan yang telah didapatkan dari penelitian berikut dan saran yang bisa diberikan untuk penelitian selanjutnya.

7.1 Kesimpulan

Setelah melakukan implementasi dan pengujian pada penelitian "Implementasi Prinsip Heuristik untuk Rekomendasi Index menggunakan Java pada MariaDB", dapat disimpulkan beberapa hal sebagai berikut:

1. Kinerja sebuah database menurut QPS (menggunakan waktu iterasi selama 60 detik) dan *responsetime* menggunakan index hasil rekomendasi dari penelitian berikut dapat meningkat sekitar 2.777% hingga 43,652%.

2. Dari dua contoh database yang digunakan untuk ujicoba, dapat disimpulkan bahwa pemilihan index dengan prinsip heuristik berikut bekerja lebih baik pada database yang belum memiliki index pendukung kegiatan operasional.

3. Dalam penelitian ini disimpulkan bahwa pengukuran nilai *responsetime* sebuah *query* yang dieksekusi memiliki nilai yang terbaik selama 50 kali. Hasil nilai tersebut didapatan dari hasil percobaan selama beberapa kali untuk mendapatkan rekomendasi yang konsisten, sehingga dengan rekomendasi yang konsisten dapat dikatakan bahwa rata-rata *responsetime* telah stabil.

7.2 Saran

Saran penulis yang dapat dilakukan untuk penelitian selanjutnya ialah:

1. Mengembangkan aplikasi dengan menambahkan fitur seperti menggabungkan penilaian QPS dan *responsetime* misalnya dengan menggunakan persamaan regresi linear atau penilaian metriks lain.
2. Memperhatikan struktur database seperti *storage engine* yang digunakan agar dapat menentukan rekomendasi lebih detail seperti tipe index *cluster* atau non-*cluster*.
3. Mengimplementasikan algoritma lain dalam pemilihan rekomendasi index seperti Genetic algorithm untuk mendapatkan kinerja yang lebih baik.
4. Melakukan impelementasi algoritma pemilihan index pada bahasa pemrograman lain atau objek database lain.
5. Melakukan penerapan konfigurasi tabel tidak hanya konfigurasi nama kolom pada pemilihan rekomendasi index.
6. Melakukan penilaian kinerja berbasis database atau tidak pertabel untuk mengetahui apakah rekomendasi dengan pemilihan index secara keseluruhan dan pertabel memiliki perbedaan yang signifikan.

Halaman ini sengaja dikosongkan.

DAFTAR PUSTAKA

[1] T. Connolly och C. Begg, Database Systems A practic approach to design, implementation, and management.

[2] R. Oshman och J. W. Knottenbelt, ”Database syste performance evaluation models: A survey,” *Performanc Evaluation*, vol. 69, p. 471, 2012.

[3] S. Chauduri, M. Datar och V. Narasayya, ”Index Selectic for Databases: A Hardness Study and a Principle Heurist Solution,” *IEEE*, vol. 16, nr TRANSACTIONS O KNOWLEDGE AND DATA ENGINEERING, p. 131 2004.

[4] P. Ameri, J. Meyer och A. Streit, ”On a New Approach t the Index Selection Problem using Mining Algorithms *IEEE*, nr International Conference on Big Data, pp. 280 2810, 2015.

[5] M. Winand, SQL Performance Explained, Wien: Marku Winand, 2012.

[6] W. G. Pedrozo och M. S. M. G. Vaz, ”A Tool for Automat Index Selection in Database Management System,” *IEE* nr International Symposium on Computer. Consumer ar Control, p. 1061, 2014.

[7] G. Valentin, M. Zuliani, D. C. Zilio, G. Lohman och A Skelley, ”DB2 Advisor : An Optimizer Smart Enough t Recommended Its Own Indexes,” 2005.

[8] S. Choenni, H. Blanken och T. Chang, ”Index Selectio relational Database,” *IEEE,* pp. 491-496, 1993.

[9] S. Chauduri och V. Narasayya, ”An Efficient, Cost-Drive Index Selection Tool for Microsoft SQL Server,” *VLDB* vol. 23, p. 146, 1997.

[10] D. Somayajulu och S. V. Vobugari, ”Index Tuning Throug Query Evauation Mechanism Based On Indirect DOma Knowledge,” *IEEE,* nr Internation Conference on Modelli and Simulation, pp. 630-635, 2012.

[11] B. Hermawan, Menguasai Java 2 & Object Oriente Programming, Yogyakarta: ANDI, 2004.

[12] P. Ameri, ”On self-Tuning Index Recommendatio Approach for Database,” *ICDE 2016 Workshop,* pp. 201 205, 2016.

[13] R. Ramakhrisnan och J. Gehrke, Sistem Manajeme Database Edisi 3, Yogyakarta: ANDI and McGraw-Hi Education, 2004.

[14] F. Razzoli, Mastering MariaDB, Birmingham: Pac Publishing, 2014.

[15] ”MariaDB versus MySQL - Features,” MariaDB, [Online
Available: https://mariadb.com/kb/en/mariadb/mariadb-v
mysql-features/. [Använd 26 February 2017].

[16] ”Heuristik,” Wikipedia, 6 June 2016. [Online]. Availabl
https://id.wikipedia.org/wiki/Heuristik. [Använd 20 Janua
2017].

[17] L. Torgo, Data Mining with R : Learning with case studie
Taylor and Francis Group, LLC, 2011.

[18] *INGREDIENTS OF MACHINE LEARNING (Sistem Cerd
Materi 3).* [Performance]. Jurusan Sistem Informasi , 201

[19] ”Adventure Works Cycles Business Scenarios,” Microso
TechNet, [Online]. Availabl
https://technet.microsoft.com/en-
us/library/ms124825(v=sql.100).aspx. [Använd 16 februa
2017].

[20] V. Ferlyando, ”RANCANG BANGUN PERANGKA
LUNAK UNTUK MONITORING DAN PENILAIA
KINERJA WEBSITE PADA DOMAN ITS.AC.ID
Surabaya, 2017.

[21] d. chen, l. s. sain och k. guo, ”Data mining for the onli
retail industry: A case study of RFM model-based custom
segmentation using data mining,” *Database Marketing*

Customer Strategy Management, vol. 19, pp. 197-20 2012.

[22] ”MariaDB Connector/J 2.0.3 Stable,” MariaDB, [Online Available: https://downloads.mariadb.org/connecto java/2.0.3/. [Använd 20 January 2017].

[23] ”GitHub,” 17 Maret 2016. [Online]. Availabl https://github.com/JSQLParser/JSqlParser. [Använd Apı 2017].

[24] H. M. S. Informasi, ”BINUS UNIVESITY,” 14 Juli 201 [Online]. Availabl http://scdc.binus.ac.id/himsisfo/2016/07/pengertian-methode-class-dan-objek-dalam-oop/. [Använd 13 Ju 2017].

[25] ”SQL Data Dictionary,” 25 July 2016. [Online]. Availabl http://www.sqldatadictionary.com/AdventureWorks2012. df. [Använd 2 June 2017].

[26] ”Adventureworks Database for MySQL,” sourceforge, 2 September 2009. [Online]. Availabl https://sourceforge.net/projects/awmysql/. [Använd 1 January 2017].

Halaman ini sengaja dikosongkan.

LAMPIRAN A
DATA INPUT

A.1 General Log

Berikut ini ialah contoh potongan input general log

```
170407 17:00:59     174 Query SELECT `crawled_link` FROM
`webstatus` WHERE `alamat_web`='interior.its.ac.id' ORDER
BY `webstatus`.`tanggal_crawl` DESC limit 1
170407 17:01:44     174 Query SELECT `crawled_link` FROM
`webstatus` WHERE `alamat_web`='interior.its.ac.id' ORDER
BY `webstatus`.`tanggal_crawl` DESC limit 1
170407 17:02:25     174 Query SELECT `crawled_link` FROM
`webstatus` WHERE `alamat_web`='electra.ee.its.ac.id' ORDER
BY `webstatus`.`tanggal_crawl` DESC limit 1
170407 17:02:29     174 Query SELECT `crawled_link` FROM
`webstatus` WHERE `alamat_web`='kerjasama.its.ac.id' ORDER
BY `webstatus`.`tanggal_crawl` DESC limit 1
170407 17:02:35     174 Query SELECT `crawled_link` FROM
`webstatus` WHERE `alamat_web`='sa.its.ac.id' ORDER BY
`webstatus`.`tanggal_crawl` DESC limit 1
170407 17:02:44     174 Query SELECT `crawled_link` FROM
`webstatus` WHERE `alamat_web`='io.its.ac.id' ORDER BY
`webstatus`.`tanggal_crawl` DESC limit 1
170407 17:02:52     174 Query SELECT `crawled_link` FROM
`webstatus` WHERE `alamat_web`='io.its.ac.id' ORDER BY
`webstatus`.`tanggal_crawl` DESC limit 1
170407 17:03:03     174 Query SELECT `crawled_link` FROM
`webstatus` WHERE `alamat_web`='spektronics.its.ac.id'
ORDER BY `webstatus`.`tanggal_crawl` DESC limit 1
170407 17:03:11     174 Query SELECT `crawled_link` FROM
`webstatus` WHERE `alamat_web`='kopma.lmb.its.ac.id' ORDER
BY `webstatus`.`tanggal_crawl` DESC limit 1
170407 17:04:17     174 Query SELECT `crawled_link` FROM
`webstatus` WHERE `alamat_web`='kse.lakone.ep.its.ac.id'
ORDER BY `webstatus`.`tanggal_crawl` DESC limit 1
170407 17:04:18     174 Query SELECT `crawled_link` FROM
`webstatus` WHERE `alamat_web`='hima.interior.its.ac.id'
```

Kode A.1. potongan general Log Monitordom

A.2 Query

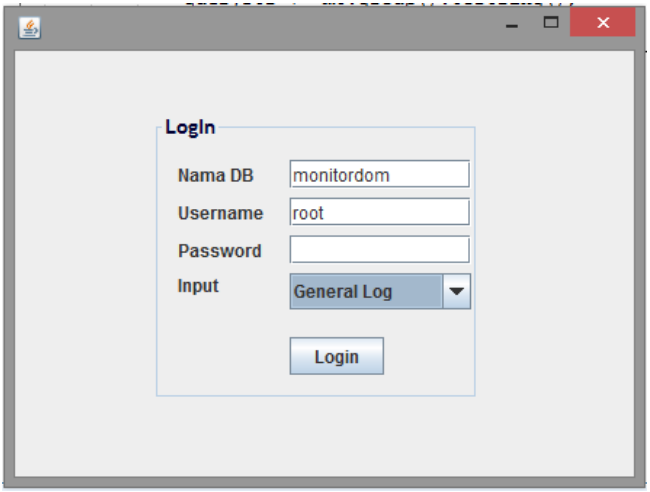
```
--Melihat keuntungan berdasarkan per toko
SELECT businessentityID, TotalDue FROM store AS S JOIN salesorderheader
AS SO ON S.SalesPersonID = SO.SalesPersonID GROUP BY businessentityID
ORDER BY TotalDue DESC;

--Melihat keuntungan top toko per negara
SELECT S.Name, ST.Name, businessentityID, TotalDue FROM store AS S JOIN
salesorderheader AS SO ON S.SalesPersonID = SO.SalesPersonID JOIN
salesterritory AS ST ON St.TerritoryID = SO.TerritoryID GROUP BY ST.Name
ORDER BY TotalDue DESC;

--Melihat negara yang paling menguntungkan penjualannya
SELECT ST.Name, businessentityID, TotalDue FROM store AS S JOIN
salesorderheader AS SO ON S.SalesPersonID = SO.SalesPersonID JOIN
salesterritory AS ST ON St.TerritoryID = SO.TerritoryID GROUP BY ST.Name
ORDER BY TotalDue DESC;

--Melihat Produk yang paling menguntungkan dan yang kurang menguntungkan
SELECT S.Name AS Store , SOH.TotalDue, PC.Name AS Category,  PM.Name AS
Model, P.Name AS Product FROM Product AS P JOIN productmodel AS PM ON
PM.ProductModelID = P.ProductModelID JOIN productsubcategory AS PSC ON
PSC.ProductSubcategoryID = P.ProductSubcategoryID JOIN productcategory AS
PC ON PC.ProductCategoryID = PSC.ProductCategoryID JOIN salesorderdetail
AS SOD ON SOD.ProductID = P.ProductID JOIN salesorderheader AS SOH ON
SOH.SalesOrderID = SOD.SalesOrderID JOIN store AS S ON SOH.SalesPersonID
= S.SalesPersonID GROUP BY S.Name ORDER BY 2 ASC;

--Melihat tanggal order terakhir per toko
SELECT businessentityID, OrderDate FROM store AS S JOIN salesorderheader
AS SO ON S.SalesPersonID = SO.SalesPersonID GROUP BY businessentityID
ORDER BY OrderDate DESC ;

--Melihat frekuensi pembelian store
SELECT businessentityID, COUNT(SOD.SalesOrderID) AS frekuensi FROM store
AS S JOIN salesorderheader AS SO ON S.SalesPersonID = SO.SalesPersonID
JOIN salesorderdetail AS SOD ON SO.SalesOrderID = SOD.SalesOrderID GROUP
BY BusinessEntityID ORDER BY frekuensi DESC
```

Kode A.2. Query AdventureWorks2012

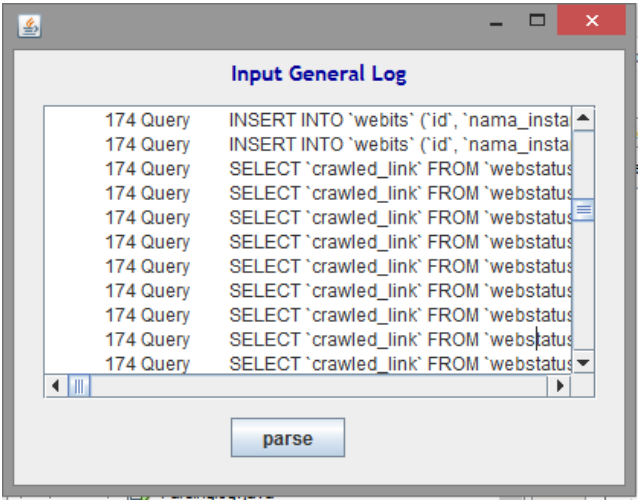
LAMPIRAN B

HASIL PENGUJIAN FUNGSIONAL

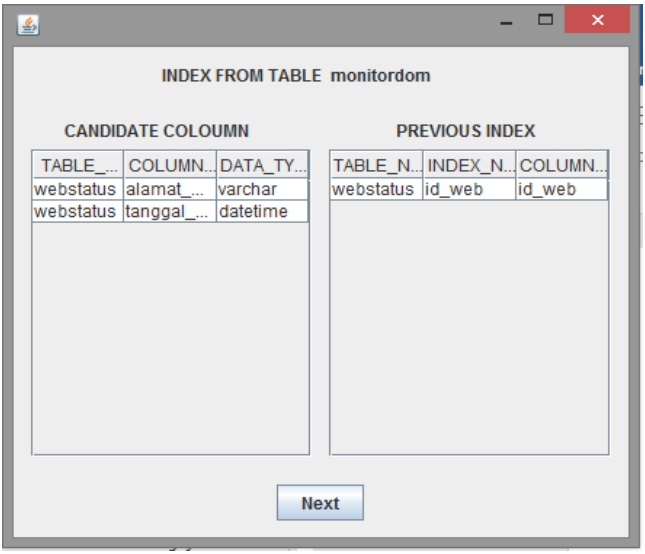
B.1 Monitordom



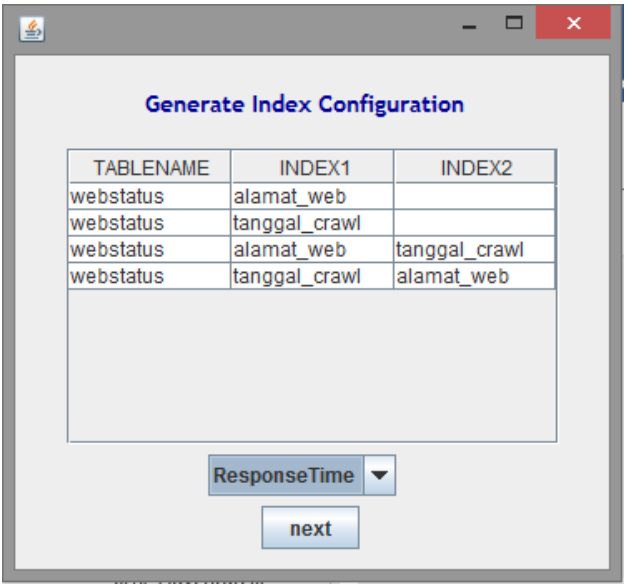
Gambar B.1 Halaman Login Monitordom



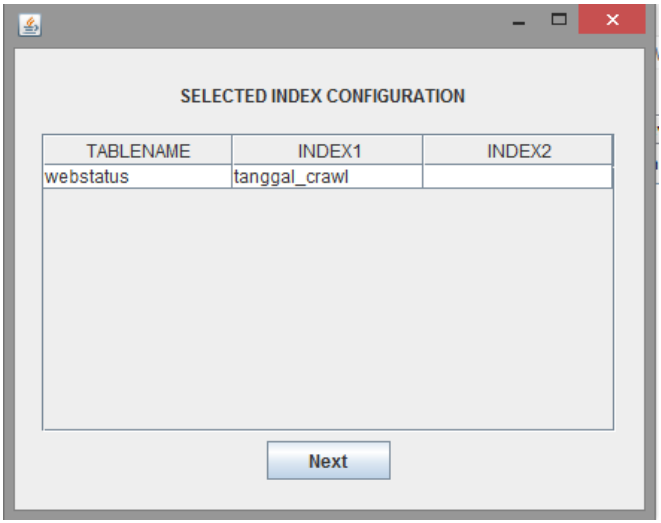
Gambar B.2 Halaman Parsing Monitordom



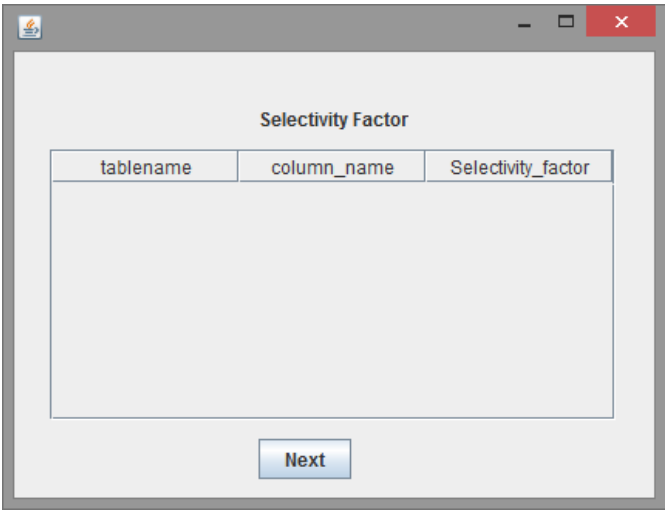
Gambar B.3 Halaman kandidat kolom dan pengambilan previous index



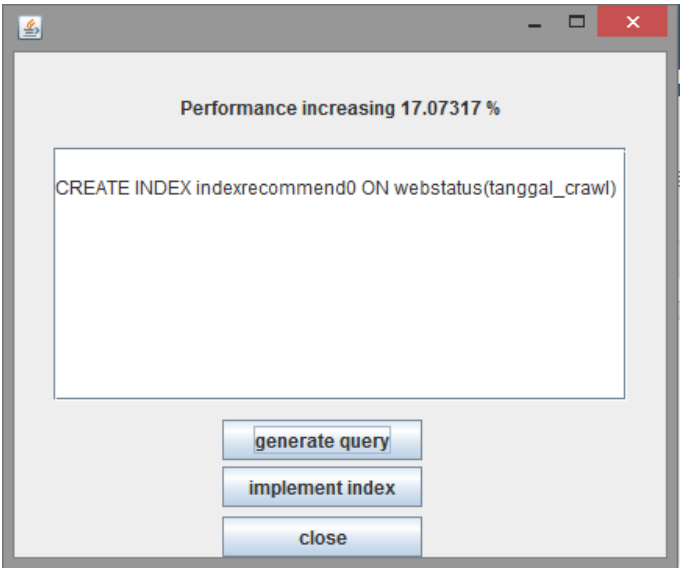
Gambar B.4. Halaman Generate Index



Gambar B.5 Halaman Selected Configuration

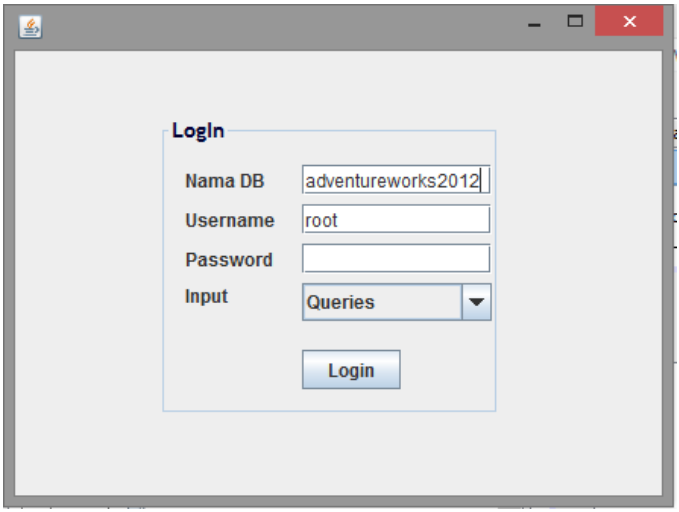


Gambar B.6. Halaman Selectivity Factor

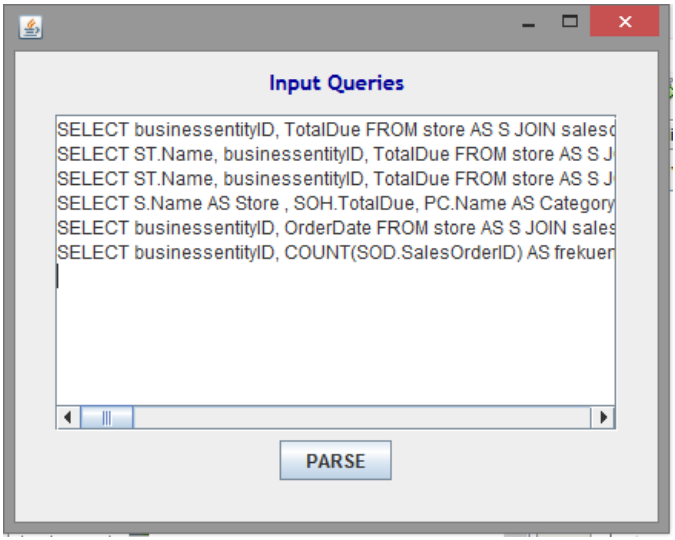


Gambar B.7 Halaman Generate Index Query

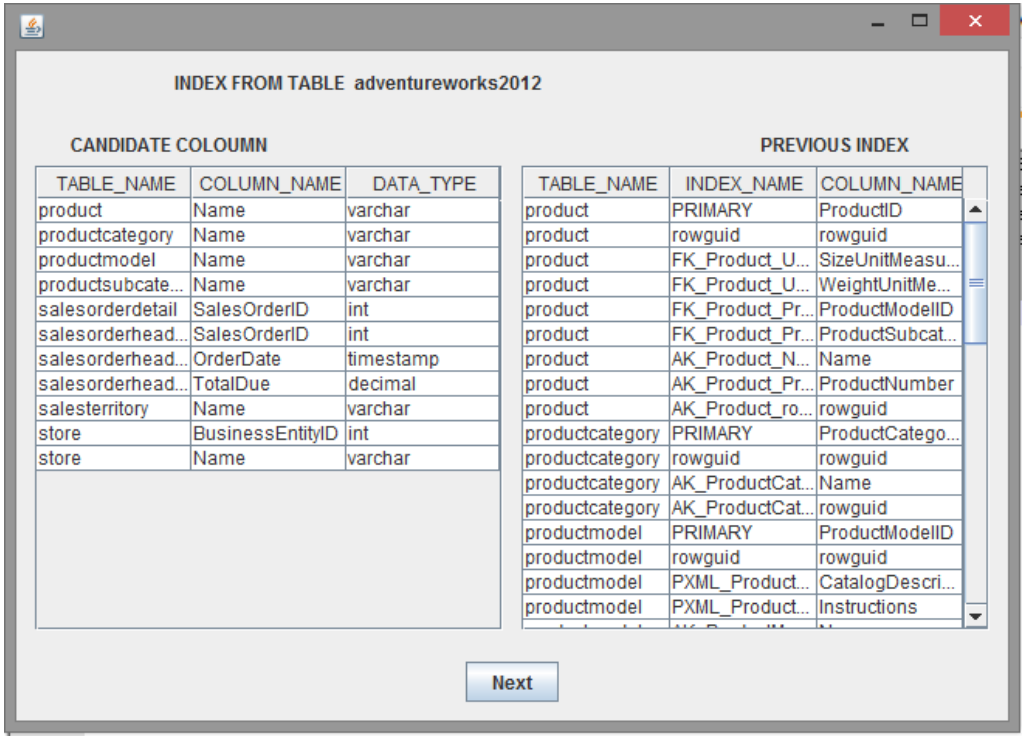
B.2 AdventureWorks



Gambar B.8. Halaman Login AdventureWorks2012



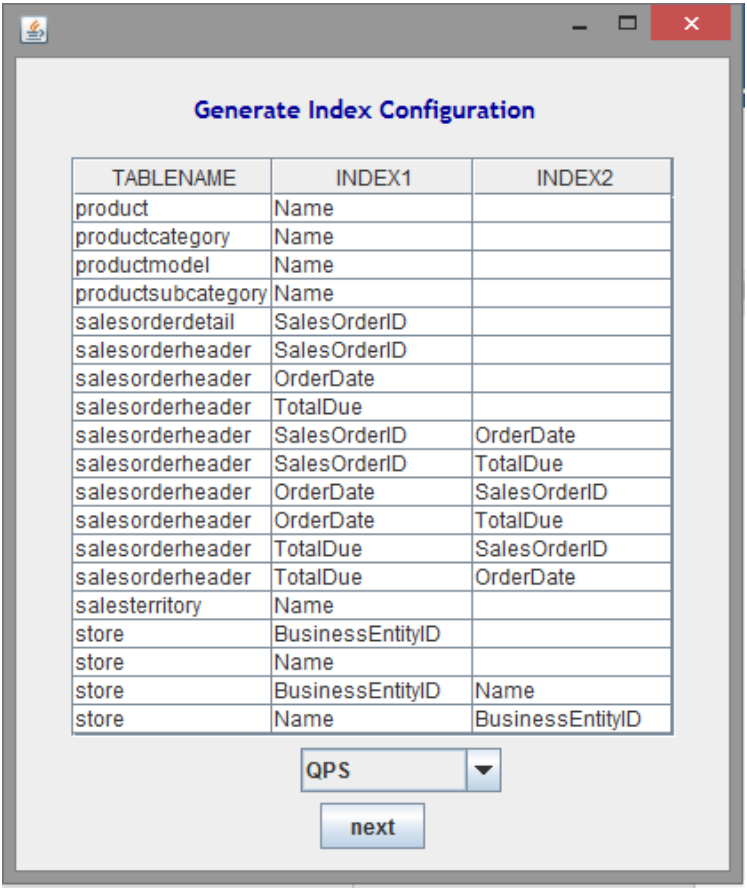
Gambar B.9 Halaman Parsing Query



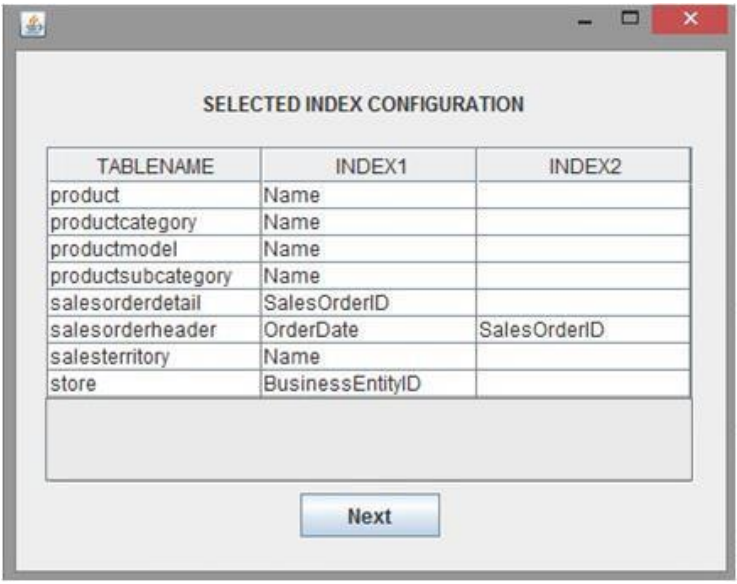
TABLE_NAME	COLUMN_NAME	DATA_TYPE
product	Name	varchar
productcategory	Name	varchar
productmodel	Name	varchar
productsubcate...	Name	varchar
salesorderdetail	SalesOrderID	int
salesorderhead...	SalesOrderID	int
salesorderhead...	OrderDate	timestamp
salesorderhead...	TotalDue	decimal
salesterritory	Name	varchar
store	BusinessEntityID	int
store	Name	varchar

TABLE_NAME	INDEX_NAME	COLUMN_NAME
product	PRIMARY	ProductID
product	rowguid	rowguid
product	FK_Product_U...	SizeUnitMeasu...
product	FK_Product_U...	WeightUnitMe...
product	FK_Product_Pr...	ProductModelID
product	FK_Product_Pr...	ProductSubcat...
product	AK_Product_N...	Name
product	AK_Product_Pr...	ProductNumber
product	AK_Product_ro...	rowguid
productcategory	PRIMARY	ProductCatego...
productcategory	rowguid	rowguid
productcategory	AK_ProductCat...	Name
productcategory	AK_ProductCat...	rowguid
productmodel	PRIMARY	ProductModelID
productmodel	rowguid	rowguid
productmodel	PXML_Product...	CatalogDescri...
productmodel	PXML_Product...	Instructions

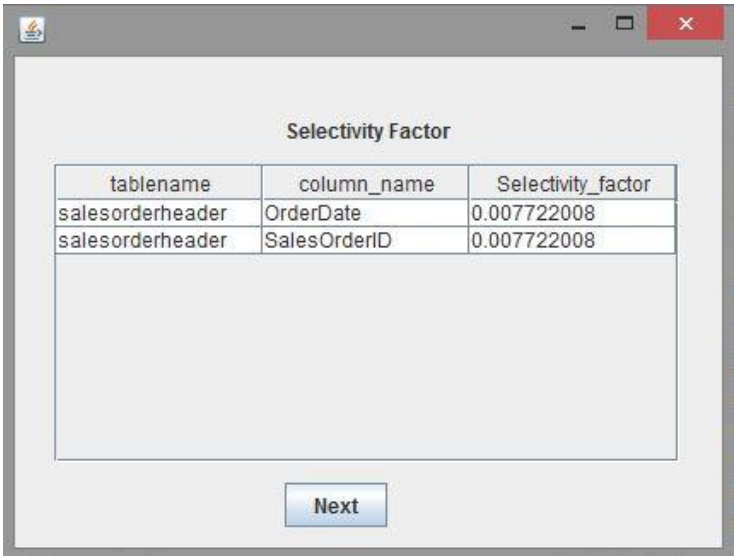
Gambar B.10 Halaman Kandidat Kolom dan Retrieve Index



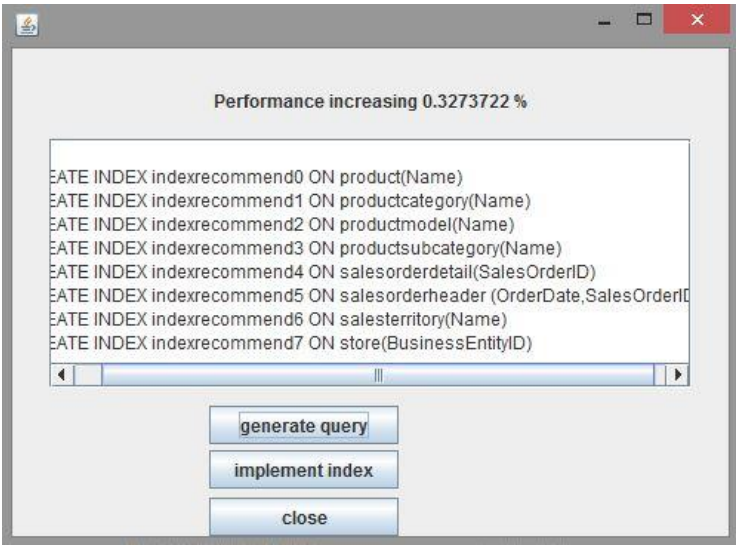
Gambar B.11 Halaman Generate Index Configuration



Gambar B.12. Halaman Index Configuration



Gambar B.13 Halaman Selectivity Factor



Gambar B.14. Halaman Generate Index Query Adventureworks

Halaman ini sengaja dikosongkan.

LAMPIRAN C

DATA HASIL PENGUJIAN

C.1 Monitordom

Tabel C.1 Hasil Pengujian 1 Monitordom

Test 1	PrevIndex	Rekomendasi	%
Webstatus	0.028599998	0.021117646	
Webstatus	0.021588236	0.011470589	
Webstatus	0.027264707	0.011500000	**43.65231**
webstatus	0.021147060	0.011470589	
Rata-rata	**0.024650000**	**0.013889706**	

Tabel C.2 Hasil Pengujian 2 Monitordom

Test 2	PrevIndex	Rekomendasi	%
webstatus	0.0142647060	0.0294111765	
webstatus	0.0192941170	0.0110294120	
webstatus	0.0216176470	0.0124117640	**14.2188**
webstatus	0.0192941170	0.0110294120	
Rata-rata	**0.0186176468**	**0.0159704411**	

Tabel C.3 Hasil Pengujian 3 Monitordom

Test 3	PrevIndex	Rekomendasi	%
webstatus	0.0225294120	0.0216470580	
webstatus	0.0147058820	0.0110294120	
webstatus	0.0147058820	0.0110294120	**25.5004**
webstatus	0.0215294120	0.0110294120	
Rata-rata	**0.0183676470**	**0.0136838235**	

Tabel C.4 Hasil Pengujian 4 Monitordom

Test 4	PrevIndex	Rekomendasi	%
webstatus	0.0151470590	0.0251176470	
webstatus	0.0221176460	0.0110294120	
webstatus	0.0234705880	0.0112942100	**32.47481**
webstatus	0.0272058830	0.0119411760	
Rata-rata	**0.0219852940**	**0.0148456113**	

Tabel C.5 Hasil Pengujian 5 Monitordom

Test 5	PrevIndex	Rekomendasi	**Tabel 0.2**
Webstatus	0.0247941180	0.0257352940	
Webstatus	0.0192941170	0.0105882350	
Webstatus	0.0239117650	0.0110294120	**37.71572**
Webstatus	0.0257352940	0.0110294120	
Rata-rata	**0.0234338235**	**0.0145955883**	

C.2 AdventureWorks

Tabel C.6 Hasil Pengujian 1 Adventureworks2012

Test 1	PrevIndex	Recommendation	%
product	0.06666667	0.06666667	1.84116255
productcategory	0.06666667	0.06666667	
productmodel	0.06666667	0.06666667	
productsubcategory	0.06666667	0.06666667	
salesorderdetail	0.26666668	0.26666668	
salesorderheader	2.8277779	3.1333332	
salesorderheader	2.8777778	3.211111	
salesorderheader	3.0777779	3.2055557	
salesorderheader	3.2555554	3.213889	
salesorderheader	3.2166665	3.2222223	
salesorderheader	3.2583332	3.2611113	
salesorderheader	3.1944447	3.2583332	
salesorderheader	3.2305555	3.2277777	
salesorderheader	3.2305555	3.2555554	
salesterritory	3.975	4	
store	3.2472222	3.2555554	
store	3.2666667	3.2527778	
store	3.2666667	3.2555559	
store	3.2555554	3.2694445	
AVG	**2.405994145**	**2.450292408**	

Tabel C.7. Hasil Pengujian 2 Adventureworks2012

Test 2	PrevIndex	Recommendation	%
Product	0.05	0.6666667	
productcategory	0.05	0.6666667	
productmodel	0.6666667	0.6666667	
productsubcategory	0.6666667	0.6666667	
salesorderdetail	0.275	0.275	
salesorderheader	3.1555555	3.3	
salesorderheader	3.2194445	3.286111	
salesorderheader	3.2166665	3.263889	
salesorderheader	2.988889	3.213889	
salesorderheader	3.0777779	3.1944447	
salesorderheader	3.261111	3.1916668	
salesorderheader	3.3	3.2722223	
salesorderheader	3.111111	3.1750002	
salesorderheader	3.0472224	2.9166667	
salesterritory	3.9916668	3.7666667	
Store	3.3055556	3.2166665	
Store	3.3305557	3.2444441	
Store	3.2611113	3.2555554	
Store	3.2361114	3.2833335	
AVG	**2.484795368**	**2.553801195**	**2.777123106**

Tabel C.8. Hasil Pengujian 3 Adventureworks2012

Test 3	PrevIndex	Recommendation	%
product	0.06666667	0.06666667	0.665988124
productcategory	0.06666667	0.06666667	
productmodel	0.06666667	0.06666667	
productsubcategory	0.06666667	0.06666667	
salesorderdetail	0.26666668	0.26666668	
salesorderheader	3.2611113	2.988889	
salesorderheader	3.263889	3.1472223	
salesorderheader	3.2555554	3.3166666	
salesorderheader	3.175	3.3500001	
salesorderheader	3.1805556	3.3277779	
salesorderheader	3.161111	3.2388887	
salesorderheader	3.2361114	3.211111	
salesorderheader	3.2083333	3.2611113	
salesorderheader	3.2083333	3.2444441	
salesterritory	3.8416667	3.9583335	
store	3.2555554	3.263889	
store	3.1916666	3.2666667	
store	3.2666664	3.238889	
store	3.2583332	3.2583334	
AVG	**2.436695893**	**2.452923998**	

Tabel C.9 Hasil Pengujian 4 Adventureworks2012

Test 4	PrevIndex	Recommendation	%
product	0.06666667	0.06666667	-0.480912892
productcategory	0.06666667	0.033333335	
productmodel	0.06666667	0.06666667	
productsubcategory	0.06666667	0.06666667	
salesorderdetail	0.275	0.275	
salesorderheader	3.222222	3.136111	
salesorderheader	3.1916666	3.1833334	
salesorderheader	3.152778	3.1111114	
salesorderheader	3.2	3.2416668	
salesorderheader	3.2	3.1972225	
salesorderheader	3.2222223	3.2166665	
salesorderheader	3.2194445	3.263889	
salesorderheader	3.216667	3.2	
salesorderheader	3.213889	3.213889	
salesterritory	3.9416666	3.8833332	
Store	3.2083333	3.216667	
Store	3.211111	3.2083333	
Store	3.2305555	3.1916666	
Store	3.2361114	3.213889	
AVG	**2.432017573**	**2.420321687**	

Tabel C.10. Hasil Pengujian 5 Adventureworks2012

Test 5	PrevIndex	Recommendation	%
product	0.6666667	0.6666667	
productcategory	0.6666667	0.6666667	
productmodel	0.6666667	0.6666667	
productsubcategory	0.6666667	0.6666667	
salesorderdetail	0.2666668	0.275	
salesorderheader	3.2	3.286111	
salesorderheader	3.1861115	3.2777777	
salesorderheader	3.1472223	3.302778	
salesorderheader	3.163889	3.2333336	
salesorderheader	3.1000001	3.3166668	
salesorderheader	3.1777775	3.261111	
salesorderheader	3.1722221	3.2777777	
salesorderheader	3.1694448	3.2972221	
salesorderheader	3.1166668	3.2833335	
salesterritory	3.9166665	3.725	
store	3.1833334	3.2500002	
store	3.2222223	2.9416666	
store	3.1777775	3.2055557	
store	3.2250001	3.2555554	
AVG	**2.531140395**	**2.571345058**	**1.588401151**

BIODATA PENULIS

Penulis lahir di Jombang pada tanggal 30 bulan Juni tahun 1996. Penulis merupakan anak pertama dari tiga bersaudara. Sebelum menepuh pendidikan di Institut Teknologi Sepuluh Nopember pada tahun 2013 dengan jalur undangan, penulis menempuh pendidikannya di SMA Negeri 2 Jombang. Penulis sangat menggemari belajar hal-hal baru seperti organisasi, kepanitiaan, olahraga, dan musik. Sehingga selagi mengemban pendidikan formal pada Jurusan Sistem Informasi penulis aktif di organisasi dan kepanitiaan, antara lain menjadi staff *Organization Social and Responsibility* di BEM FTIF, sebagai panitia SESINDO (Seminar Nasional Sistem Informasi Indonesia), *Liaison Officer* pada acara IFFA College Bowl IV, dan kegiatan lain yang selalu memberikan pelajaran dan pengalaman berharga bagi penulis. Pada masa kahir perkuliahan, penulis memiliki ketertarikan pada bidang akuisisi data dan diseminasi informasi, sehingga penulis memutuskan untuk melakukan penelitian Tugas Akhir pada Lab Akuisisi Data dan Diseminasi Informasi. Penulis dapat dihubungi melalui email : sujanawatirisa@gmail.com