OPTIMASI *QUERY* PADA SISTEM BASIS DATA TERDISTRIBUSI MENGGUNAKAN METODE *SEMIJOIN BASED APPROACH* (SBA)

SKRIPSI

Sebagai salah satu syarat untuk memperoleh gelar

Sarjana dalam bidang Ilmu Komputer

Disusun oleh :

MOHAMAD SYAUQI REZA

NIM. 0910963023

PROGRAM STUDI TEKNIK INFORMATIKA

PROGRAM TEKNOLOGI INFORMASI DAN ILMU KOMPUTER

UNIVERSITAS BRAWIJAYA

MALANG

2013

LEMBAR PERSETUJUAN

OPTIMASI *QUERY* PADA SISTEM BASIS DATA TERDISTRIBUSI MENGGUNAKAN METODE *SEMIJOIN BASED APPROACH* (SBA)

SKRIPSI

Disusun oleh :

MOHAMAD SYAUQI REZA

NIM. 0910963023

Telah diperiksa dan disetujui oleh :

Pembimbing I,	Pembimbing II,
Yusi Tyroni Mursityo, S.Kom., MS	**Satrio Agung W., S.Kom., M.Kom**
NIP. 198002282006041001	**NIP. 86052106110114**

LEMBAR PENGESAHAN

OPTIMASI *QUERY* PADA SISTEM BASIS DATA TERDISTRIBUSI MENGGUNAKAN METODE *SEMIJOIN BASED APPROACH* (SBA)

SKRIPSI

Sebagai salah satu syarat untuk memperoleh gelar
Sarjana dalam bidang Ilmu Komputer

Disusun oleh :

MOHAMAD SYAUQI REZA
NIM. 0910963023

Skripsi ini telah diuji dan dinyatakan lulus pada tanggal 19 April 2013

Penguji I	Penguji II
Edy Santoso, S.Si, M.Kom	**Aditya Rachmadi, S.ST,. M.TI**
NIP. 197404142003121004	**NIK. 86042116110426**

Penguji III

Fitra A. Bachtiar, ST, M.Eng
NIK. 84062816110427

Mengetahui
Ketua Program Studi Teknik Informatika

Drs. Marji, M.T.
NIP. 19670801 199203 1 001

PERNYATAAN ORISINALITAS SKRIPSI

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah SKRIPSI ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis dikutip dalam naskah ini dan disebutkan dalam sumber kutipan dan daftar pustaka.

Apabila ternyata didalam naskah SKRIPSI ini dapat dibuktikan terdapat unsur-unsur PLAGIASI, saya bersedia SKRIPSI ini digugurkan dan gelar akademik yang telah saya peroleh (SARJANA) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku. (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, April 2013

Mohamad Syauqi Reza

NIM 0910963023

OPTIMASI *QUERY* PADA SISTEM BASIS DATA TERDISTRIBUSI MENGGUNAKAN METODE *SEMIJOIN BASED APPROACH* (SBA)

ABSTRAK

Basis data terdistribusi adalah basis data yang secara fisik terbagi menjadi beberapa situs, namun secara logis adalah satu basis data dan digunakan bersama-sama dalam suatu jaringan komputer. Proses *query* dalam basis data terdistribusi berbeda dengan basis data tersentralisasi, ada tambahan biaya komunikasi untuk menghasilkan laporan yang sama saat mengakses situs lain yang tentunya akan membuat *query* lebih lambat. Penelitian ini mengimplementasikan metode *semijoin based approach* (SBA) untuk memperoleh *query* yang cepat. SBA bekerja dengan cara memetakan semua model akses yang mungkin sesuai relasi dari tabel kemudian dihitung masing-masing biayanya dan diambil waktu eksekusi paling kecil. Subyek dari penelitian ini adalah basis data sistem informasi akademik universitas dengan data sebanyak 1393907 data. Optimasi dilakukan pada 3 *query* dan didapatkan peningkatan kecepatan sebesar 1259% untuk *query* 1 dan peningkatan sebesar 816% untuk *query* 2 dengan total data yang dieksekusi sebesar 1393735 data. Dan peningkatan sebesar 648% untuk *query* 3 dengan total data yang dieksekusi sebesar 3393 data. Hal ini menunjukkan bahwa metode SBA bisa digunakan untuk *data* besar maupun kecil.

Kata kunci: *Join*, *Semijoin*, Basis data terdistribusi, *Semijoin Based Approach*, Optimasi

QUERY OPTIMIZATION ON DISTRIBUTED DATABASE SYSTEM USING SEMIJOIN BASED APPROACH (SBA)

ABSTRACT

Distributed database is a database that is physically divided into a number of sites, but it logically is one database and used together in a computer network. The process of query in a distributed database is different from centralized database, there is an additional communication cost to generate the same report when accessing other sites that will surely make the query slower. This study implements the semijoin method based approach (SBA) to gain a quick query. SBA works by mapping all possible access model appropriate relation from the table and then calculated each cost and take the smallest execution time. The subject of this research is databases of university academic information system with 1393907 data. Optimizations performed on the 3 queries and obtained an increase in speed by 1259% for first query and an increase of 816% for the second query is executed with a total of 1,393,735 data data. And an increase of 648% for third query with a total query execution data for 3393 data. This suggests that the SBA method can be used for large and small record.

Keys: Join, Semijoin, Distributed Database, Semijoin based approach, Optimization

KATA PENGANTAR

Puji syukur penulis panjatkan kehadirat TuhanYang Maha Esa yang telah melimpahkan segala kasih sayang dan rahmat-Nya sehingga penulis dapat menyelesaikan tugas akhir dengan judul "**Optimasi *Query* pada Sistem Basis Data Terdistribusi menggunakan Metode *Semijoin Based Approach* (SBA)**".

Skripsi ini diajukan sebagai salah satu syarat untuk memperoleh gelar Sarjana Komputer Strata Satu (S-1) Program Studi Teknik Informatika, Program Teknologi Informasi & Ilmu Komputer Universitas Brawijaya, Malang.Penulis menyadari bahwa tugas akhir ini tidak dapat terealisasikan tanpa bantuan dari berbagai pihak, untuk itu penulis menyampaikan penghargaan dan ucapan terima kasih yang sedalam-dalamnya kepada :

1. Yusi Tyroni Mursityo, S.Kom., M.S selaku pembimbing utama yang telah meluangkan waktu memberikan pengarahan dan bimbingan kepada penulis.
2. Satrio Agung Wicaksono, S.Kom., M.Kom selaku pembimbing pendamping yang telah meluangkan waktu memberikan pengarahan dan bimbingan kepada penulis.
3. Drs. Marji, MT., selaku Ketua Program Studi Teknik Informatika, Program Teknologi Informasi & Ilmu Komputer Universitas Brawijaya.
4. Candra Dewi, S.Kom, M.Sc selaku dosen pembimbing akademik, atas nasehat dan bimbingan akademik yang telah diberikan.
5. Segenap bapak dan ibu dosen yang telah mendidik dan mengajarkan ilmunya kepada penulis selama menempuh pendidikan di Program Teknologi Informasi & Ilmu Komputer Universitas Brawijaya.
6. Segenap staf, karyawan dan civitas di Program Teknologi Informasi & Ilmu Komputer Universitas Brawijaya yang telah banyak membantu penulis dalam pelaksanaan penyusunan tugas akhir ini.
7. Kedua orang tua yang penulis cintai, yang telah memberikan dukungan baik materi maupun non-materi selama penulis menyelesaikan tugas akhir ini.
8. Rekan-rekan mahasiswa Program Studi Teknik Informatika Universitas Brawijaya yang telah memberikan dukungan, semangat dan kebersamaan.

9. Semua pihak yang telah membantu terselesaikannya penyusunan tugas akhir ini yang tidak dapat disebutkan satu per satu.

Penulis menyadari bahwa tugas akhir ini masih jauh dari kesempurnaan, untuk itu dengan segala kerendahan hati penulis mengharapkan saran dan kritik yang membangun demi kesempurnaan penulisan selanjutnya. Semoga laporan tugas akhir ini dapat bermanfaat untuk semua pihak.

Malang, April 2013

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

1.1 Latar Belakang

Universitas adalah sebuah lembaga tinggi yang bergerak dibidang pendidikan. Universitas sendiri memiliki fakultas dengan masing-masing prodi dibawahnya. Tiap universitas pasti memiliki basis data untuk menyimpan informasi penting universitas. Basis data yang digunakan bisa merupakan basis data tersentralisasi ataupun basis data terdistribusi. Basis data tersentralisasi adalah basis data yang secara fisik terdapat dalam satu situs. Sedangkan basis data terdistribusi adalah basis data dimana secara fisik tersebar ke dalam beberapa situs namun secara logis terhubung menjadi satu kesatuan basis data dan digunakan bersama-sama dalam suatu jaringan komputer [VAL-11]. Pemilihan model basis data tersentralisasi maupun terdistribusi dilakukan sesuai keadaan dan kebutuhan universitas. Untuk universitas dengan jumlah mahasiswa sedikit, basis data tersentralisasi cukup untuk memenuhi *current connection* yang terjadi. Namun untuk universitas dengan jumlah mahasiswa yang relatif banyak, maka basis data terdistribusi sangat cocok untuk dipilih. Hal ini dikarenakan dalam basis data terdistribusi kapasitas *current connection* yang mungkin adalah sebanding dengan jumlah situs yang ada.

Prakteknya, universitas pasti membutuhkan laporan tentang data yang ada di basis data. Dalam basis data tersentralisasi, hal ini cukup mudah dilakukan karena semua data tersimpan dalam satu situs sehingga untuk membuat laporan yang dibutuhkan, maka petugas tinggal menggunakan *query* untuk menghasilkan laporan yang dibutuhkan didalam situs tersebut. Namun dalam basis data terdistribusi, untuk membuat laporan yang sama, mungkin saja membutuh merelasikan beberapa tabel yang terdapat dalam situs yang berbeda dengan menggunakan *join query* untuk mendapatkan laporan yang dibutuhkan. Dengan adanya akses antar situs, tidak bisa dipungkiri adanya biaya tambahan yaitu biaya komunikasi. Untuk *Wide Area Network*, biaya komunikasi adalah 20:1 dengan lokal. Sedangkan untuk *ethernet* 10Mbps biaya komunikasi adalah 1,6:1 dengan

lokal [VAL-11]. Masalah yang timbul dengan besarnya biaya komunikasi ini adalah proses *query* yang lambat dan mungkin tidak sesuai dengan harapan. Operasi *join query* sendiri adalah operasi yang membutuhkan biaya cukup tinggi serta teknik khusus didalamnya. Selain itu, dalam basis data terdistribusi ada tiga hal penting yang perlu kita perhatikan. *Join order*, model akses, dan jumlah *data* basis data. *Join order* merupakan aspek penting dalam *join query*. Melakukan *join* dalam level basis data terdistribusi antar fragmen pastinya akan menambah biaya komunikasi. Dua hal dasar yang bisa dilakukan untuk meminimalkan biaya komunikasi adalah dengan mengoptimalkan *join*, cara lain adalah me*replace join* dengan *semijoin* [VAL-11]. Pengetahuan tentang model akses dan jumlah *data* basis data juga merupakan hal yang penting sebagai dasar untuk melakukan *query*. Karena berbeda model akses sangat besar kemungkinan untuk beda biaya saat melakukan eksekusi *query*. Model akses juga berpengaruh pada model *query* yang akan di*order*.

Pada penelitian sebelumnya, untuk optimasi *query* dalam basis data terdistribusi menggunakan metode *Fragment and Replicate Strategy* (FRS) oleh Alom,B.M.M. Prinsip kerja metode ini adalah melokalkan fragmen yang ada dengan metode replikasi sehingga biaya komunikasi sewaktu melakukan *query* bisa diminimumkan. Sehingga *query* bisa optimal. Namun FRS dalam implementasinya membutuhkan storage yang besar [ALO-09]. Oleh karena itu berdasarkan kebutuhan ketepatan hasil *query* dengan biaya yang minimum, maka penulis akan melakukan analisa untuk mendapatkan *query* yang optimal dengan metode *Semijoin based approach* (SBA) berdasarkan model akses yang mungkin dilakukan. Dengan metode pendekatan ini, diharapkan bisa mendapatkan hasil yang optimal di lingkungan basis data terdistribusi dikarenakan *join order* yang dilakukan bisa lebih spesifik dengan model akses yang optimal.

Berdasarkan latar belakang yang telah dikemukakan, maka judul yang akan diambil dalam skripsi ini adalah **"Optimasi *Query* pada Sistem Basis Data Terdistribusi menggunakan Metode *Semijoin Based Approach*"**.

1.2 Rumusan Masalah

Dengan adanya latar belakang di atas, maka dapat dirumuskan permasalahan yang akan dijadikan obyek penelitian ini, yaitu :

1. Bagaimana biaya komunikasi bisa diminimalkan dalam basis data terdistribusi.
2. Bagaimana mengoptimalkan *join order* dengan menggunakan metode *Semijoin based approach*.
3. Bagaimana model akses basis data yang paling efektif.
4. Seberapa optimal jika optimasi dilakukan dalam basis data yang memiliki data yang besar.

1.3 Batasan Masalah

Penelitian ini memiliki batasan masalah sebagai berikut:

1. Optimasi dalam sistem basis data terdistribusi dilakukan dalam basis data yang difragmen secara horizontal.
2. Basis data yang digunakan adalah basis data SIAKAD studi kasus Universitas Brawijaya.
3. Data yang digunakan berupa data *dummy* dengan jumlah sesuai data asli.
4. Kecepatan jaringan yang diterapkan antar situs adalah sama, yaitu 1 Mbps.

1.4 Tujuan

Tujuan yang ingin dicapai dalam penelitian ini, antara lain :

1. Menghasilkan eksekusi *query* dengan waktu y ang relatif cepat.
2. Menghasilkan formulasi *query* dengan biaya komunikasi minimum dengan mengimplementasikan metode *Semijoin based approach*.
3. Menghasilkan model akses paling efektif dalam basis data terdistribusi.
4. Mengukur tingkat keoptimalan metode *Semijoin based approach* untuk optimasi *query* dalam sistem basis data terdistribusi.

1.5 Manfaat

Manfaat penelitian ini untuk memberikan wawasan tentang analisis optimasi *query*. Diharapkan dengan adanya penelitian ini, bisa menjadi acuan bagi

programmer dalam menentukan formula *query* yang optimal sehingga data bisa didapatkan lebih cepat.

1.6 Sistematika Pembahasan

1. BAB I PENDAHULUAN

Bab ini berisi latar belakang penelitian, rumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, dan sistematika pembahasan.

2. BAB II KAJIAN PUSTAKA

Bab ini berisi teori dari berbagai pustaka yang menunjang dalam penelitian ini. Teori yang terdapat pada bab ini antara lain mengenai terdistribusi, arsitusktur terdistribusi, konsep aljabar relasional, dan optimasi *query*.

3. BAB III METODOLOGI DAN PERANCANGAN

Bab ini berisi mengenai perancangan yang akan digunakan untuk penelitian ini meliputi analisis data, rancangan basis data, contoh perhitungan manual dan rancangan uji coba.

4. BAB IV IMPLEMENTASI

Bab ini berisi mengenai implementasi rancangan yang akan digunakan meliputi pembuatan basis data, pembuatan sistem basis data terdistribusi dan melakukan implementasi metode SBA dengan model akses yang telah dirancang pada bab III.

5. BAB V HASIL DAN PEMBAHASAN

Bab ini berisi hasil dari perhitungan manual optimasi formula *query* dan implementasi optimasi *query* dalam terdistribusi menggunakan DBMS yang digunakan untuk mendukung perhitungan secara manual. Serta pembahasan analisis hasil uji coba dan evaluasi uji coba.

6. BAB VI PENUTUP

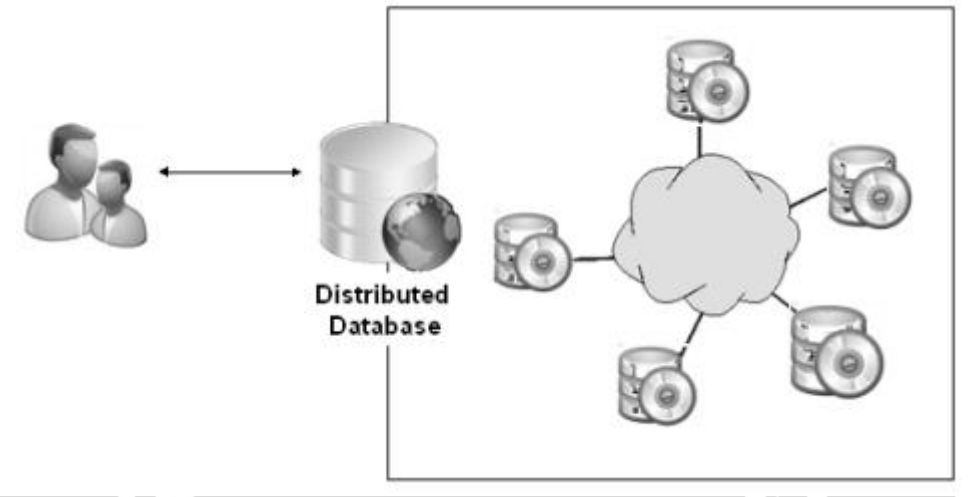
Bab ini berisi kesimpulan dari hasil penelitian dan saran-saran yang bermanfaat untuk pengembangan penelitian ini selanjutnya.

BAB II
KAJIAN PUSTAKA

2.1. Basis Data Terdistribusi

Secara pengelolaan, sistem basis data terbagi menjadi dua yaitu pengelolaan secara terpusat atau basis data terpusat dan basis data terdistribusi. Perbedaan antara keduanya yang utama adalah pada lokasi, jika pada basis data terpusat maka basis data secara fisik diletakkan pada satu tempat dimana *user* hanya akan mengakses dan mendapatkan akses dari lokasi basis data tersebut. Dalam basis data terdistribusi, data disimpan kedalam beberapa komputer dimana antar komputer saling berkomunikasi dengan berbagai media seperti jaringan atau kabel telepon. Didalamnya tidak ada *share main memory*. Dan walaupun disebar pada beberapa tempat yang berbeda namun tetap berelasi, kemudian dengan metode tertentu situs-situs tersebut dijadikan satu kesatuan [SIL-04].

Untuk mengelola sistem basis data terdistribusi memerlukan suatu sistem manajemen. Menurut Valduriez, sistem manajemen basis data terdistribusi adalah sistem perangkat lunak yang mengijinkan memanajemen basis data terdistribusi dan membuat proses distribusi menjadi transparan untuk pengguna. Kadang sistem basis data terdistribusi digunakan untuk menggabungkan basis data terdistribusi dan sistem manajemen basis data terdistribusi [VAL-11].

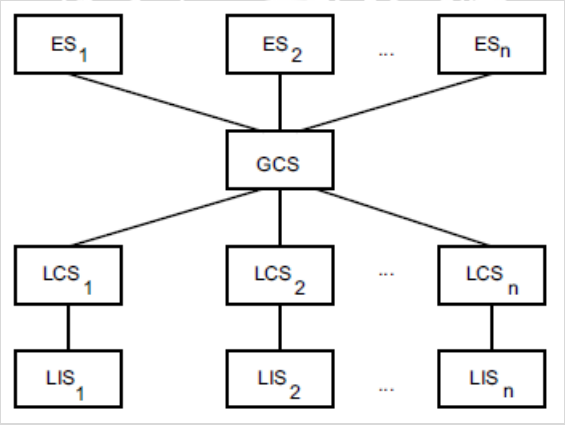


Gambar 2. 1 Lingkungan sistem basis data terdistribusi.

Sumber: Silberschatz (2004).

2.1.1. Arsitektur Sistem Basis Data Terdistribusi

Pada sistem basis data terdistribusi, arsitusktur suatu basis data terdistribusi adalah seperti berikut



Gambar 2. 2 Arsitektur Basis Data Terdistribusi. Sumber: Valduriez (2011)

Arsitektur ini adalah arsitektur secara *peer to peer*. Arsitektur ini dinilai paling cocok untuk sistem basis data terdistribusi. Dalam arsitektur ini, terdapat 4 bagian yaitu LIS, LCS, GCS, dan ES. LIS adalah *Local Internal Schema*, yaitu skema internal yang didefinisikan pada tiap situs nya. LCS adalah *Local*

Conceptual Schema, yaitu bagian untuk meng*handle* data fragmentasi dan replikasi. GCS adalah *Global Conceptual Schema*, yaitu bagian yang mendeskripsikan struktur logis dari data pada semua situs. GCS bisa jadi merupakan *union* dari LCS. Sedangkan ES adalah *external schema*, yaitu bagian yang berkenaan dengan aplikasi atau pengguna.

2.1.1.1.Fragmentasi Horizontal

Fragmentasi adalah strategi pemecahan data dalam suatu basis data dan menempatkannya pada situs yang berbeda. Tujuan dari fragmentasi adalah untuk transparansi pada pengguna. Fragmentasi sendiri terdapat 3 jenis, horizontal, vertikal, dan *hybrid*. Pada fragmentasi horizontal, data dipartisi secara horizontal. Data dipartisi menjadi beberapa subset. Contoh fragmentasi horizontal adalah berikut:

Tabel 2. 1 Tabel Sebelum Difragmen

Fakultas	Nama
PTIIK	Andi
MIPA	Bandi
MIPA	Candi
PTIIK	Dandi

Maka setelah difragment secara horizontal akan menjadi seperti berikut

Tabel 2. 2 Tabel setelah difragmen

Fakultas	Nama
PTIIK	Andi
PTIIK	Dandi

Fakultas	Nama
MIPA	Bandi
MIPA	Candi

2.1.2. Jaringan Komputer

Dalam sistem basis data terdistribusi, jaringan komputer adalah hal yang penting karena proses distribusi data bergantung pada jaringan yang digunakan. Untuk *Wide Area Network*, besar biaya komunikasi adalah 20:1 dengan lokal.

Sedangkan untuk *ethernet* 10Mbps besar biaya komunikasi adalah 1,6:1 dengan lokal [VAL-11]. Salah satu topologi jaringan yang bagus adalah *star*. Topologi ini Paling fleksibel dibandingkan dengan topologi yang lain. Pemasangan workstation yang baru sangat mudah, dan tidak mengganggu kerja dari komputer yang lain. Kontrol terpusat, sehingga memudahkan pengecekan kesalahan jaringan. Dan mudah mendeteksi kesalahan pada jaringan, karena adanya kontrol terpusat dan satu kabel untuk satu komputer. [SUG-10].

2.2. Aljabar Relasional

Aljabar relasional adalah operasi dasar untuk model relasional. Operasi ini memungkinkan pengguna secara spesifik mengembalikan nilai *query* atau request. Kembaliannya adalah berupa relasi baru yang mungkin saja dihasilkan dari satu atau lebih relasi. Operasi aljabar ini juga bisa menghasilkan relasi baru, dari hasil manipulasi menggunakan operasi aljabar pula. Untuk beberapa alasan, aljabar relasional sangat penting sebagai fondasi awal model relasional dan sebagai dasar untuk implementasi mengoptimalkan *query* dalam sistem manajemen basis data relasional.

Aljabar relasional dibagi menjadi dua grup. Grup pertama termasuk didalamnya adalah operasi matematika yang *applicable*, hal ini dikarenakan tiap relasi didefinisikan sebagai *tuple* dalam model relasi formal operasi himpunan. Yang termasuk didalamnya antara lain *UNION*, *INTERSECTION*, *SET DIFFERENCE*, dan *CARTESIAN PRODUCT*. Grup kedua terdiri dari operasi yang memang ditujukan untuk membangun sebuah relasi basis data seperti *SELECT*, *PROJECT*, dan *JOIN* [ELM-10].

2.2.1. Operasi Unary Relasional

2.2.1.1.*Select*

Operasi *select* dilambangkan dengan σ (sigma), operasi ini digunakan untuk memilih subset dari *tuple* dari relasi yang menggunakan kondisi seleksi. Klausa untuk operasi *select* adalah

$$\sigma < selection\ condition > (R) \qquad (2.1)$$

Dimana *selection condition* adalah kondisi yang harus dipenuhi dalam melakukan operasi *select*. Sedangkan R adalah representasi dari relasi aljabar, atau biasanya berupa nama suatu tabel dalam basis data [ELM-10].

2.2.1.2.*Project*

Operasi *project* dilambangkan dengan π (pi), operasi ini digunakan untuk memilih atribut yang diinginkan dan mengabaikan atribut yang lain. Klausa untuk operasi *project* adalah

$$\pi(attribute\ list)R \tag{2.2}$$

Dimana *attribute list* adalah atribut yang diinginkan untuk ditampilkan. Sedangkan R Sedangkan R adalah representasi dari relasi aljabar, atau biasanya berupa nama suatu tabel dalam basis data [ELM-10].

2.2.2. Operasi Aljabar Relasional Teori Set

2.2.2.1.*Union* dan *Intersection*

Operasi *union* dilambangkan dengan ∪, *union* sendiri adalah statemen yang mengkombinasikan dua buah atau lebih *result* set dari multi sql statemen *SELECT* sehingga menjadi satu buah *result* set. *Union* mempunyai ketentuan sebagai berikut, pertama jumlah kolom/*field* dari setiap statemen *SELECT* harus sama, kedua tipe data kolom/*field* dari setiap statemen *SELECT* harus kompatibel. Klausa untuk operasi *union* adalah

$$R1\ (a1, a2, \dots) \cup R2\ (b1, b2, \dots) \tag{2.3}$$

R adalah representasi dari relasi aljabar, atau biasanya berupa nama tabel. A1, a2 dan b1, b2 mewakili kolom yang kompatible.

Operasi *intersection* dilambangkan dengan ∩, *intersection* adalah statement yang menghasilkan irisan dari dua relasi yang memenuhi ketentuan

union. *Intersection* akan menghasilkan *data* yang sama antara relasi yang ada [ELM-10]. Klausa untuk operasi *intersection* adalah

$$R1\ (a1, a2, \dots) \cap R2\ (b1, b2, \dots) \tag{2.4}$$

2.2.2.2.*Cross product*

Cross product atau *cartesian product* dilambangkan dengan ×, *cross product* adalah operasi untuk mengkombinasi *tuple* dari relasi yang berbeda. Untuk melakukan *cross product*, tidak perlu memenuhi ketentuan *union* [ELM-10]. Klausa untuk operasi *cross product* adalah

$$R1\ (a1, a2, \dots) \times R2\ (b1, b2, \dots) \tag{2.5}$$

2.2.3. Operasi *Relasional Binary*

2.2.3.1.*Join*

Operasi *join* dilambangkan dengan ⋈, *join* sendiri adalah operasi yang digunakan untuk menggabungkan *tuple* yang berelasi, dari dua *tuple* menjadi satu *tuple*. Operasi ini sangat penting untuk basis data relasional yang mempunyai lebih dari satu relasi, karena ini memungkinkan kita untuk memproses relasi antar relasi. Klausa untuk operasi *join* adalah

$$R3 \leftarrow R1 \bowtie x = y\ R2 \tag{2.6}$$

R1 adalah *tuple*1 sedangkan R2 adalah *tuple*2, dimana x adalah primary key dari R1 sedangkan y adalah *foreign key* dari R2. *Join* juga bisa dibentuk menggunakan operasi *cross product* diikuti oleh operasi *select* dan *projection*. Klausanya adalah seperti berikut

$$R3 \leftarrow R1 \times R2$$
$$RESULT \leftarrow \sigma\ x = y\ (R3) \tag{2.7}$$

Dimana *result* adalah *tuple* hasil. Jika menggunakan klausa *join* maka

$$RESULT \leftarrow R1 \rtimes x = y\ R2 \quad (2.8)$$

Dibandingkan dengan klausa menggunakan *cross product*, menggunakan *join* adalah relatif lebih singkat. Dua baris relasi aljabar bisa digantikan dengan satu baris menggunakan *join*. Hal ini nantinya sangat berpengaruh jika sudah diterapkan pada basis data. *Join* tidak hanya bisa menggunakan operator =, tapi *join* juga bisa menggunakan operator $<, >, \leq, \geq, \neq$. *Join* secara general juga disebut *theta join*. Jika hanya menggunakan operator = biasanya kita menyebut dengan *equijoin* [ELM-10].

2.2.3.2.Natural *Join*

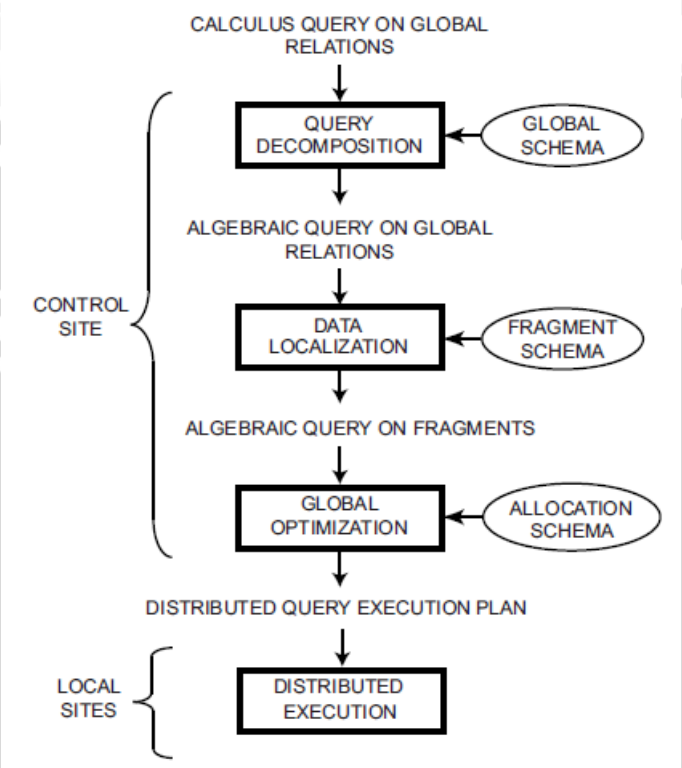
Natural *join* dilambangkan dengan ⋈, natural *join* adalah operasi yang digunakan untuk menggabungkan dua *tuple* yang berelasi dengan tidak melihat kolom yang saling berelasi terproyeksi pada satu *tuple* hasil. Natural *join* memanfaatkan *project* untuk memilih kolom mana yang diproyeksikan [ELM-10]. Klausa natural *join* adalah

$$\pi\ a, b\ (R1\ \bowtie x = y\ R2) \quad (2.9)$$

2.3. Proses *Query* Dalam Sistem Basis Data Terdistribusi

Dalam basis data terdistribusi, terdapat 4 layer dimana 3 layer terdapat pada situs utama dan 1 layer terdapat pada *local* situs. Layer pertama *query decomposition* adalah layer yang memecah *query* kalkulus menjadi *query* aljabar pada relasi global. Layer kedua adalah data *localization*, input dari layer ini adalah *query* aljabar pada relasi global. Fungsi dari layer ini adalah untuk melokalkan data hasil *query* menggunakan skema fragmen. Layer ketiga adalah *global query optimization*, input dari layer ini adalah *query* aljabar dalam fragmen. Tujuan dari *query optimization* adalah untuk menemukan strategi eksekusi yang paling mendekati optimal. Optimal disini termasuk didalamnya adalah meminimalkan *cost function*. *Cost function* disini adalah CPU *cost* dan *communication cost*.

Layer keempat adalah *distributed query execution*, layer ini dilakukan oleh semua situs yang memiliki fragmen yang terlibat dalam *query*. Setiap sub*query* mengeksekusi pada satu situs, disebut *local query*, kemudian dioptimalkan menggunakan *local schema* dari situs dan dieksekusi [VAL-11].



Gambar 2. 3 Skema layer proses *query* basis data terdistribusi.
Sumber: Valduriez (2011).

2.4. Optimasi *Query* Dalam Basis Data Terdistribusi

2.4.1. *Semijoin Base Approach*

Semijoin based approach adalah metode pendekatan berbasis biaya menggunakan pendekatan berbasis *semijoin*. Konsep *semijoin* disini adalah menerapkan *projection* dan *selection* secara tepat. Dalam metode ini akan dilakukan pemetaan berbentuk tree terhadap kemungkinan model akses yang ada. Dari beberapa kemungkinan yang ada akan dihitung dan model yang optimal adalah model dengan biaya terendah. Untuk menghitung biaya dalam proses ini, menggunakan kompleksitas basis data yaitu TDP dan TDT. TDP adalah *time data*

transmission, kompleksitas waktu dalam melakukan transfer data sesuai *query* yang dijalankan. Rumus TDT dapat dilihat pada 2.10.

$$TDT = T_0 + d/ S \quad (2.10)$$

Dimana T_0 adalah waktu mulai eksekusi, kita inisialisasikan dengan 0. Sedangkan d adalah ukuran data dalam *byte* yang akan ditransmisikan. Dan S adalah kecepatan jaringan antar situs yang akan melakukan transmisi.

Sedangkan TDP atau *time data process* adalah waktu yang dihabiskan dalam situs itu sendiri ketika melakukan proses *query*. Rumus TDP dapat dilihat pada 2.11.

$$TDP= P_0 + d/ P \quad (2.11)$$

Dimana P_0 adalah waktu mulai eksekusi, kita inisialisasikan dengan 0. Sedangkan d adalah ukuran data dalam *byte* yang akan ditransmisikan. Dan P adalah kecepatan prosesor situs yang akan melakukan transmisi. Ukuran *byte* data yang ditransmisikan, didapatkan dari perhitungan *cost statistics database* [JIA-11].

Dalam DBMS, d adalah jumlah data yang diproses, merupakan hasil dari row size dikalikan jumlah data yang dieksekusi. Untuk *row size* tergantung dari berapa lebar *field* yang kita deklarasikan. Dalam DBMS sudah ada mekanisme untuk menghitung *row size.*

Dalam bukunya, Valduriez tidak menggunakan waktu sebagai standart melainkan biaya. Dimana menginisialisasikan *row size* dengan 1 *byte*. Sehingga dalam hal ini besar data yang ditransfer tergantung dari banyak data yang diproses. Untuk melakukan transfer biaya diinisialisasikan 10 unit. Sedangkan untuk memproses data dalam situs maka biaya diinisialisasikan 1 unit [VAL-11].

2.4.2. *Cost Statistics Database*

Statistics database sangat berguna untuk mengevaluasi kardinalitas dari hasil *query*. Dua asumsi yang harus dipegang adalah, nilai dari atribut harus seragam dan atribut tersebut harus berdiri sendiri, maksudnya adalah nilai dari atribut bukan dari atribut lain. Operan dari relasi dilambangkan dengan R dan S [VAL-11]. Berikut adalah kardinalitas operasi aljabar relasional

Selection

$$card(\sigma_F(R)) = SF_S(F) * card(R) \quad (2.12)$$

Dimana SF_S tergantung formula *selection* yang digunakan. Kondisi SF_S yang mungkin adalah

$$SF_S(A = value) = \frac{1}{card(\pi_A(R))} \quad (2.13)$$

$$SF_S(A > value) = \frac{max(A) - value}{max(A) - min(A)} \quad (2.14)$$

$$SF_s(A < value) = \frac{value - min(A)}{max(A) - min(A)} \quad (2.15)$$

$$SF_s(p(A_i)) \wedge p(A_j) = SF_s(p(A_i)) * SF_s(p(A_j)) \quad (2.16)$$

$$SF_s(p(A_i)) \vee p(A_j) = SF_s(p(A_i)) + SF_s(A_j) - SF_s(p(A_i)) * SF_s(p(A_j)) \quad (2.17)$$

$$SF_s(A \in \{values\}) = SF_s(A = value) * card(\{values\}) \quad (2.18)$$

Projection

$$card(\pi_A(R)) = card(R) \quad (2.19)$$

Cartesian product

$$card(R \times S) = card(R) * card(S) \quad (2.20)$$

Join

Dengan asumsi jika relasi R adalah equi*join* relasi S dengan R.A dan S.B dimana A adalah *primary key* dan B adalah *foreign key* maka

$$card(R \bowtie_{A=B} S) = card(S) \quad (2.21)$$

$$card(R \bowtie S) = SF_j * card(R) * card(S) \quad (2.22)$$

Semijoin

Selectivity factor dari *semijoin* R dengan S. Pada hal ini hanya bergantung pada atribut A dan S.

$$SF_{SJ}(R \ltimes_A S) = \frac{card(\pi_A(S))}{card(dom[A])} \quad (2.23)$$

Fungsi biaya statistik untuk *semijoin* adalah seperti rumus dibawah dimana R.A menjadi *execution plan* dari S dimana S.A adalah *primary key*.

$$card(R \ltimes_A S) = SF_{SJ}(S.A) * card(R) \quad (2.24)$$

Union

Khusus *union*, untuk menghitung berdasarkan statistic database adalah cukup sulit. Hal ini dikarenakan duplikasi data antara R dan S sudah dihapus oleh operasi *union*. Untuk perhitungannya, diasumsikan bahwa tidak ada duplikasi *tuple* didalamnya.

$$card(R) + card(S) \quad (2.25)$$

$$max\{card(R), card(S)\} \quad (2.26)$$

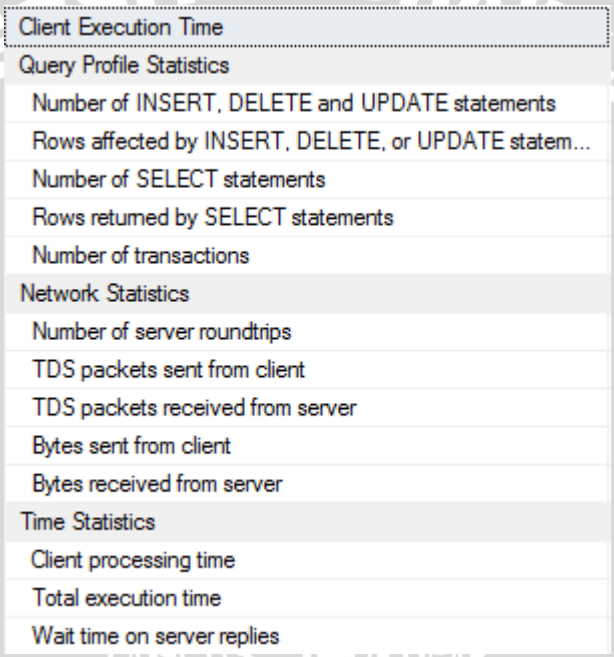
2.5. *Inner Join, Left Join, Right Join*

Inner join, left join, dan *right join* adalah alternatif *join* yang bisa digunakan dalam DBMS. *Inner join* sama dengan *join*, ini berarti hanya menampilkan data yang ada pada 2 tabel. Maksudnya adalah hanya menampilkan data yang sama-sama ada dalam 2 tabel. Sedangkan *left join*, ini berarti hanya menampilkan data dari tabel kiri kemudian dicari yang sesuai pada tabel kanan.

Dan pada *right join*, ini berarti menampilkan data dari tabel kanan kemudian dicari yang sesuai pada tabel kiri. Untuk data yang kecil, *left join* dan *right join* cukup cepat dibandingkan menggunakan *inner join*, sedangkan untuk data yang besar *inner join* lebih cepat dibandingkan keduanya [JIA-11].

2.6. *MsSQL Server Client Statistics*

Client Statistics adalah tool dari *SQL Server* yang menghasilkan informasi tentang setiap eksekusi *query*. Lebih tepatnya untuk menghitung *cost statistics database* dari *query*.



Gambar 2. 4 *MsSQL Server 2008 Client Statistics*

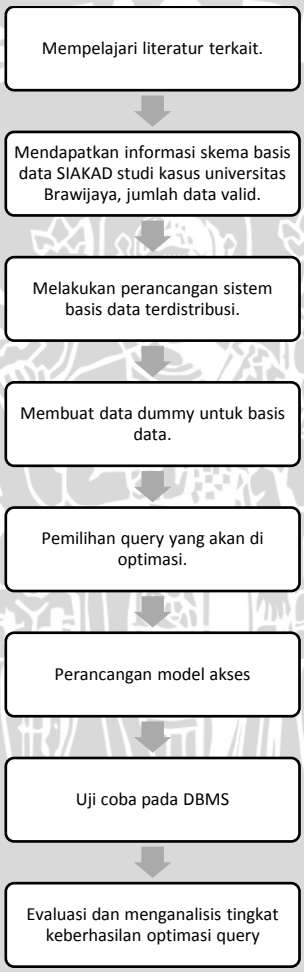
Fitur utama dari *client* statistics ini adalah *query* profile statistics, network statistics, dan time statistics. *Query profile statistics* adalah jumlah dari operasi yang dilakukan baik berupa *select, insert, update,* dan *delete* yang terjadi saat melakukan *query*. Juga jumlah *data* yang dihasilkan dari suatu eksekusi *query*. Sedangkan *network statistics* lebih kepada jumlah data yang ditransfer oleh *client* dan diterima oleh server tempat melakukan *query*. Dan untuk *time statistics* adalah total waktu dalam milisecond terhadap *query* yang dieksekusi yang ditunjukkan dalam baris *total execution time*. Dari sini patokan *query* tersebut optimal selain hasilnya benar adalah *total execution time*nya relatif kecil.

2.7. Metode Perbandingan

Metode perbandingan adalah metode untuk membandingkan dua atau lebih parameter untuk diambil hasil yang terbaik. Dalam penelitian ini, metode perbandingan digunakan sebagai metode evaluasi dan analisa. Metode ini nantinya dalam penelitian ini, akan membandingkan *total execution time* dari *query* dan diambil yang paling minimum. Metode ini dilakukan secara manual dengan membandingkan hasil perhitungan secara mesin dengan DBMS dengan tool *Client Statistics* dari *MsSQL Server*.

BAB III
METODOLOGI DAN PERANCANGAN

Pada bab metodologi dan perancangan ini akan diberikan penjelasan mengenai metode dan langkah-langkah perancangan yang dilakukan untuk dapat melakukan optimasi *query join* pada sistem basis data terdistribusi. Alur penelitian ditunjukkan pada gambar 3.1.



Gambar 3. 1 Alur Penelitian

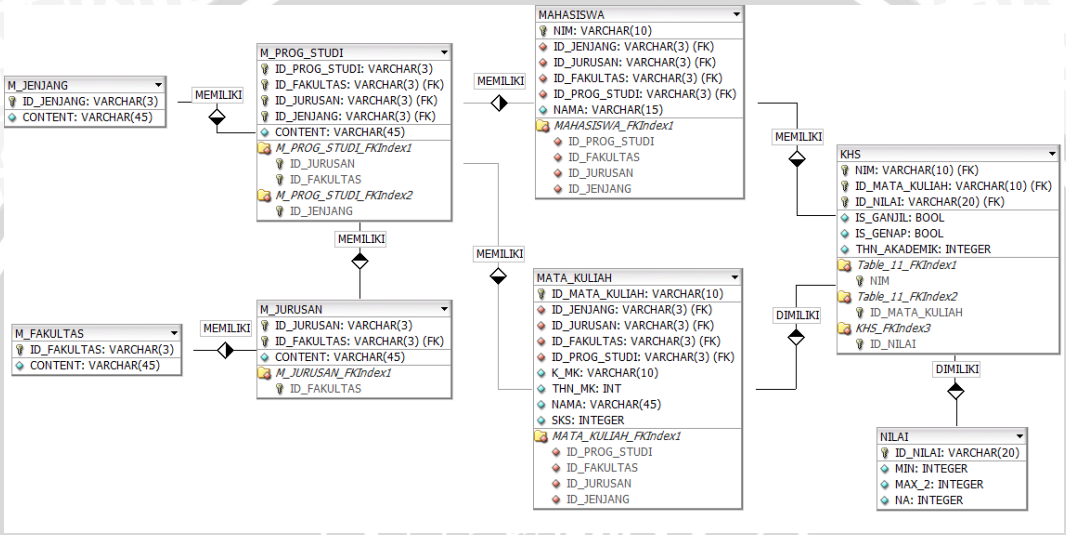
3.1. Studi Literatur

Pada penelitian ini dibutuhkan studi literatur untuk merealisasikan tujuan dan penyelesaian masalah. Teori-teori mengenai definisi aljabar relasional, basis

data terdistribusi dan metode optimasi *query* yang digunakan sebagai dasar penelitian diperoleh dari sumber-sumber atau buku-buku referensi yang berkaitan dengan skripsi, jurnal ataupun *browsing* dari *internet*.

3.2. Studi Kasus

Pada penelitian ini, penulis menggunakan basis data akademik (SIAKAD) Universitas Brawijaya sebagai studi kasus penelitian ini.



Gambar 3. 2 ERD SIAKAD Universitas Brawijaya

3.3. Perancangan Basis Data Terdistribusi

Pada penelitian ini, basis data SIAKAD akan dibagi menjadi 5 fragmen dan diletakkan pada 5 total execution time. Pembagian fragmen ini menggunakan model fragmentasi horizontal sesuai fakultas. Universitas Brawijaya sendiri memiliki 13 fakultas, berikut data mahasiswa S1 aktif per semester ganjil 2012/2013.

Tabel 3. 1 data mahasiswa S1 aktif per semester ganjil 2012/ 2013

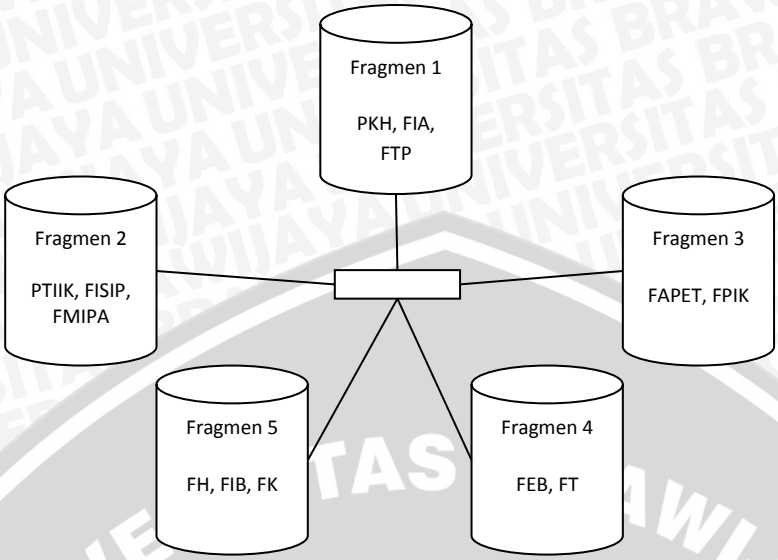
Fakultas	Jumlah Mahasiswa
Ekonomi dan Bisnis	4098
Hukum	2362
Ilmu Administrasi	5824
Ilmu Budaya	3142
Ilmu Sosial dan Ilmu Politik	4635
Kedokteran	3787
Matematika & IPA	2228
Perikanan dan Ilmu Kelautan	3728
Peternakan	5776
Prog.T.Informasi & Ilmu Komputer	2554
Program Kedokteran Hewan	671
Teknik	5146
Teknologi Pertanian	3006

Dari data diatas akan dilakukan *pre-processing* secara manual untuk membagi seluruh data menjadi 5 fragmen. Pembagian data dilakukan dengan fragmentasi horizontal menurut fakultas. Anggota masing-masing fragmen didasarkan pada jumlah mahasiswa yang ada di tiap fakultasnya. Jumlah data tiap fragmen akan dibagi dengan hasil mendekati rata. Maka dari *pre-processing* dihasilkan fragmen sebagai berikut:

Tabel 3. 2 Fragmen yang dihasilkan

Fragmen	Fakultas	Jumlah Mahasiswa	Total Mahasiswa
Fragmen 1	Program Kedokteran Hewan	671	9501
	Ilmu Administrasi	5824	
	Teknologi Pertanian	3006	
Fragmen 2	Prog.T.Informasi & Ilmu Komputer	2554	9417
	Ilmu Sosial dan Ilmu Politik	4635	
	Matematika & IPA	2228	
Fragmen 3	Peternakan	5776	9504
	Perikanan dan Ilmu Kelautan	3728	
Fragmen 4	Ekonomi dan Bisnis	4098	9244
	Teknik	5146	
Fragmen 5	Hukum	2362	9291
	Ilmu Budaya	3142	
	Kedokteran	3787	

Fragmen tersebut masing-masing akan diletakkan kedalam 5 *total execution time* berbeda. Tiap *total execution time* terkoneksi dengan jaringan komputer, jaringan komputer yang diimplementasikan adalah topologi star. Skema jaringannya dalah sebagai berikut



Gambar 3. 3 Rancangan jaringan komputer yang digunakan, topologi star

3.4. Perancangan Data Penelitian

Data yang digunakan dalam penelitian ini adalah data *dummy*, hal ini dikarenakan konsentrasi penelitian ini bukanlah pada validitas data melainkan optimasi *query* yang akan dilakukan. Sehingga tidak diperlukan data asli.

3.5. Pemilihan *Query*

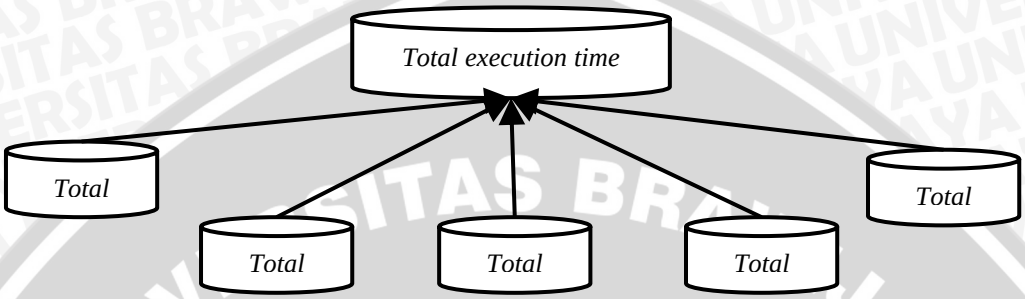
Pada penelitian ini, *query* yang akan dioptimasi adalah *reporting query*. *Reporting Query* adalah *query* yang menghasilkan data berbentuk laporan. Sebagai contoh adalah *query* untuk melihat mahasiswa dengan IPK *cumlaude*. *Query* akan menghasilkan data mahasiswa dengan IPK sesuai order yang diberikan. *Query* yang nantinya akan dioptimasi dipertimbangkan dengan *database administrator* Universitas Brawijaya untuk *reporting query* yang sering digunakan.Daftar *reportingquery* yang akan dioptimasi adalah sebagai berikut:

- Mahasiswa dengan IPK *Cumlaud*.
- Mahasiswa yang pernah mendapatkan peringatan tahun pertama.
- Seluruh mata kuliah yang ditawarkan di universitas Brawijaya.

Tiga *query* diatas dalam prosesnya akan melibatkan seluruh *total execution time* yang ada sehingga sudah memenuhi untuk digunakan dalam penelitian ini.

3.6. Perancangan Model Akses

Pada sub bab 3.5 terdapat 3 *query* yang akan dioptimasi. Secara konvensional selama ini, pemilihan tidak begitu mementingkan *join* order padahal *join* order sangatlah penting dalam optimasi *query* pada sistem basis data terdistribusi.

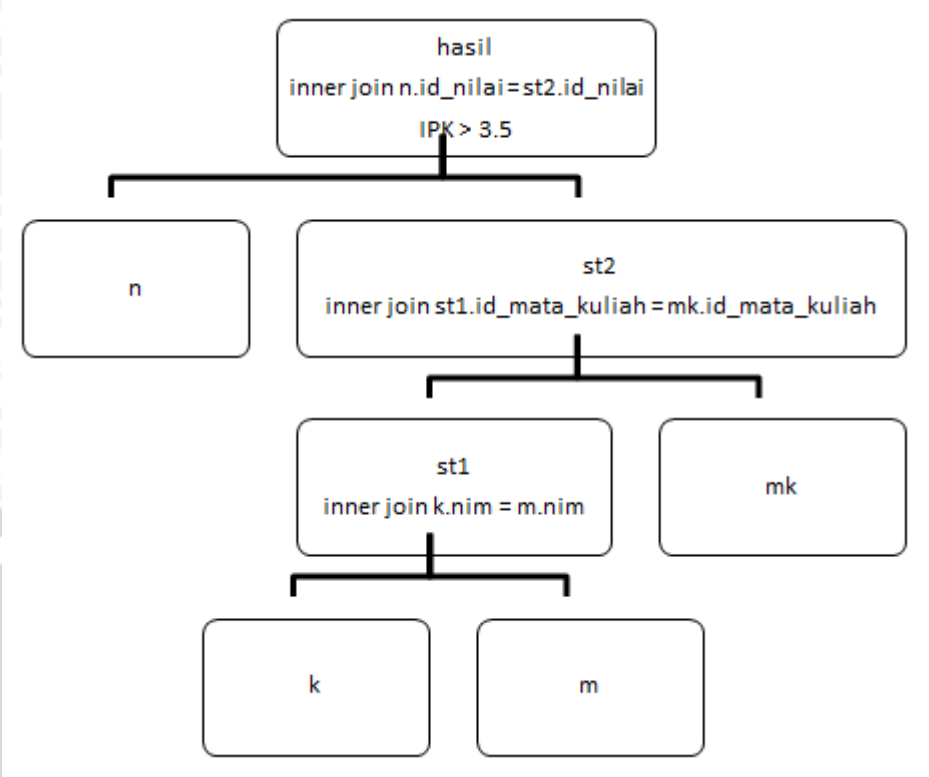


Gambar 3. 4 model akses saat *query* dieksekusi. Semua data dikirim ke total execution time pembuatan pelaporan

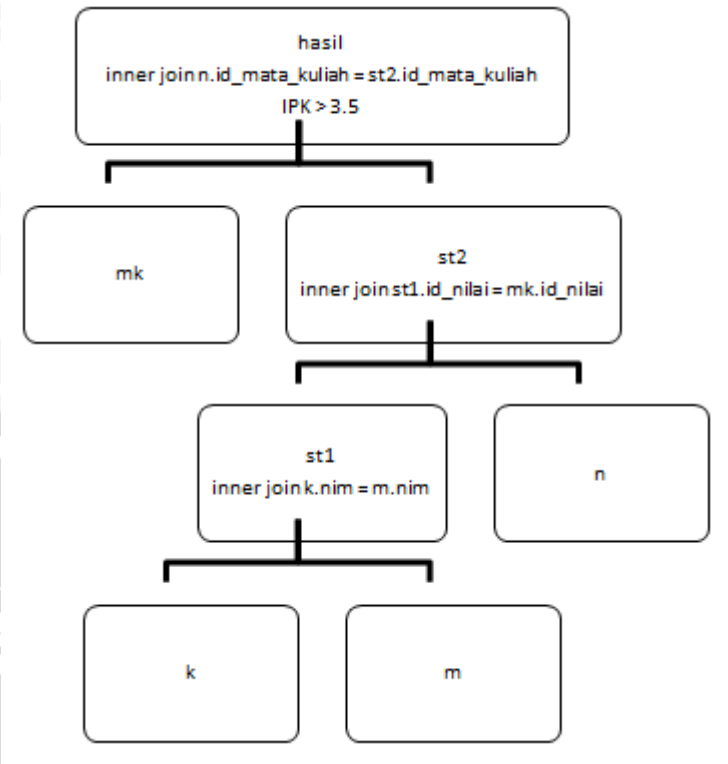
Model akses adalah skema hubungan secara logis antar *total execution time* dalam basis data terdistribusi sehingga menjadi satu kesatuan basis data. Dalam basis data terdistribusi terdapat 2 komponen yaitu *Local Conceptual Schema* (LCS) dan *Global Conceptual Schema* (GCS). Lcs adalah model akses yang dilakukan dalam *total execution time* secara lokal, sedangkan gcs adalah kumpulan dari lcs dalam bentuk global. Sesuai dengan metode SBA bahwa model akses yang mungkin akan dipetakan sesuai relasinya. Kemudian diambil nilai yang terendah.

3.6.1. Model akses *query* 1

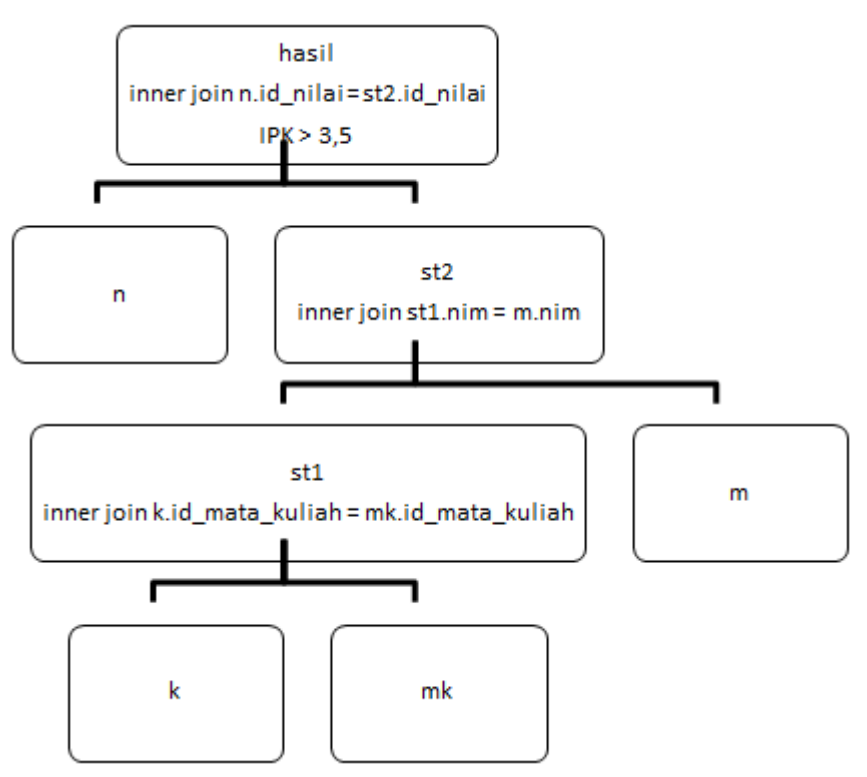
Untuk mendapatkan hasil sesuai permintaan dari *query* 1, dapatkan mahasiswa dengan IPK *cumlaude*, maka dibutuhkan beberapa tabel yaitu *khs*, *mata_kuliah*, *m_nilai*, dan *mahasiswa*. Dalam hal ini tabel *khs* memiliki relasi dengan 3 tabel lainnya dimana 3 tabel tersebut hanya terelasi pada tabel *khs*. Sehingga lcs yang mungkin dari *query* 1 adalah:



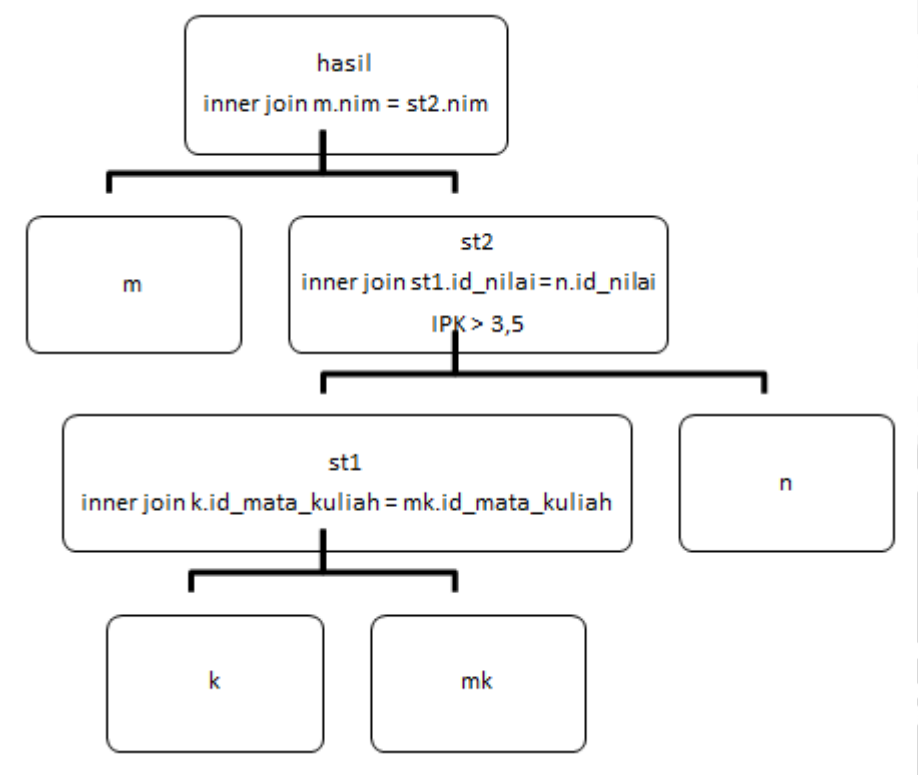
Gambar 3. 5 Rancangan A *query* 1



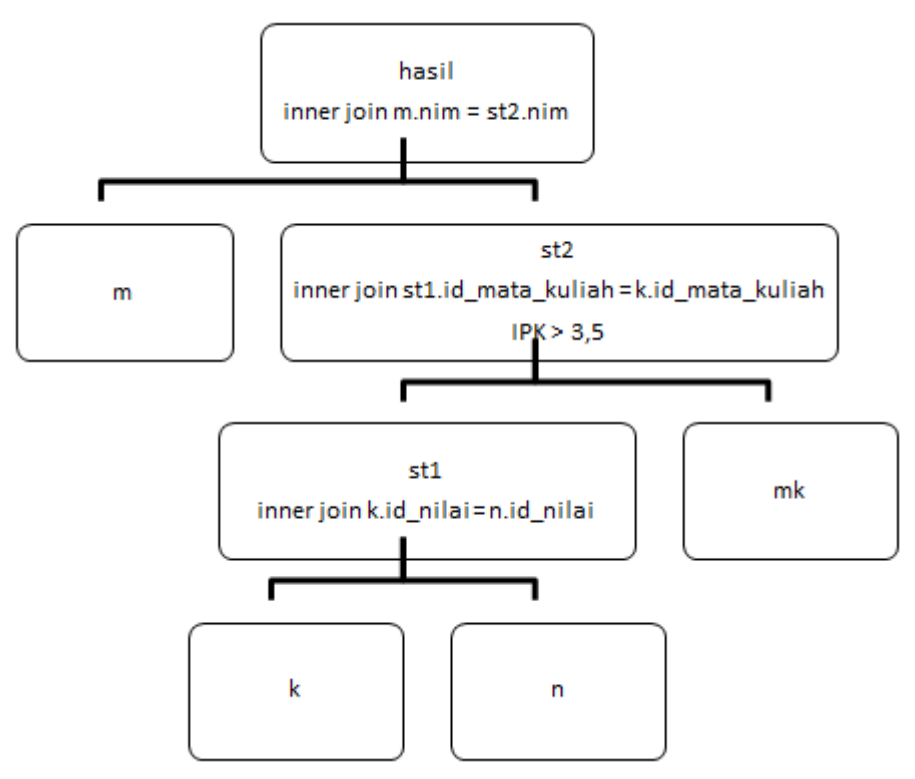
Gambar 3. 6 Rancangan B *query* 1



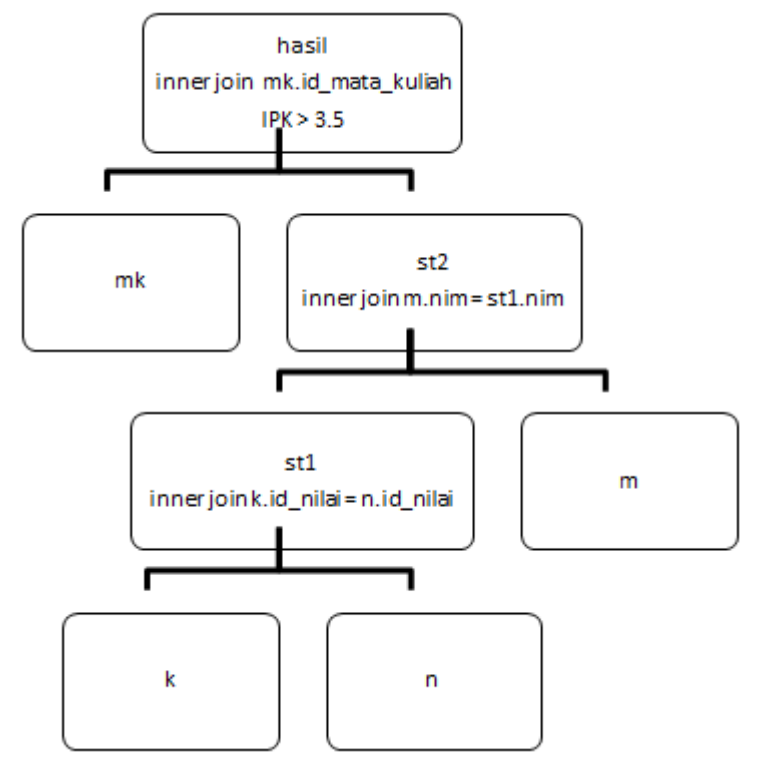
Gambar 3. 7 Rancangan C *query* 1



Gambar 3. 8 Rancangan D *query* 1



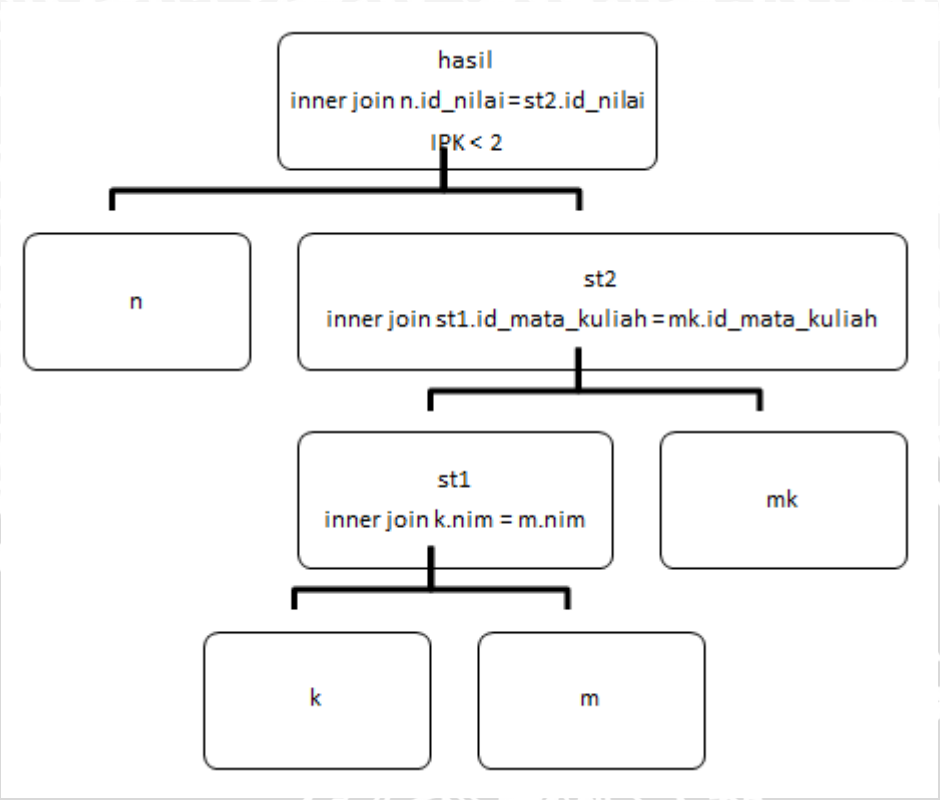
Gambar 3. 9 Rancangan E *query* 1



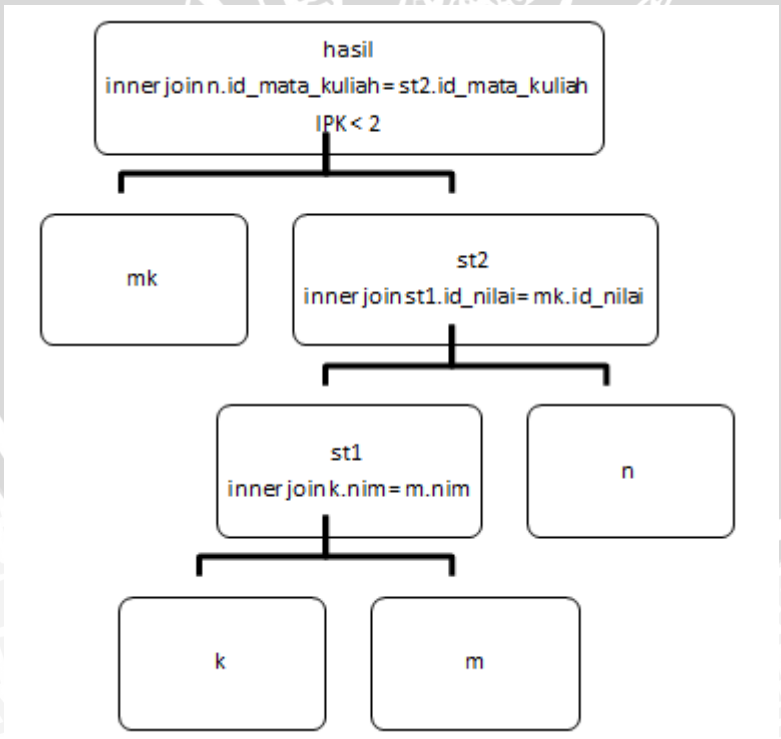
Gambar 3. 10 Rancangan F *query* 1

3.6.2. Model akses *query* 2

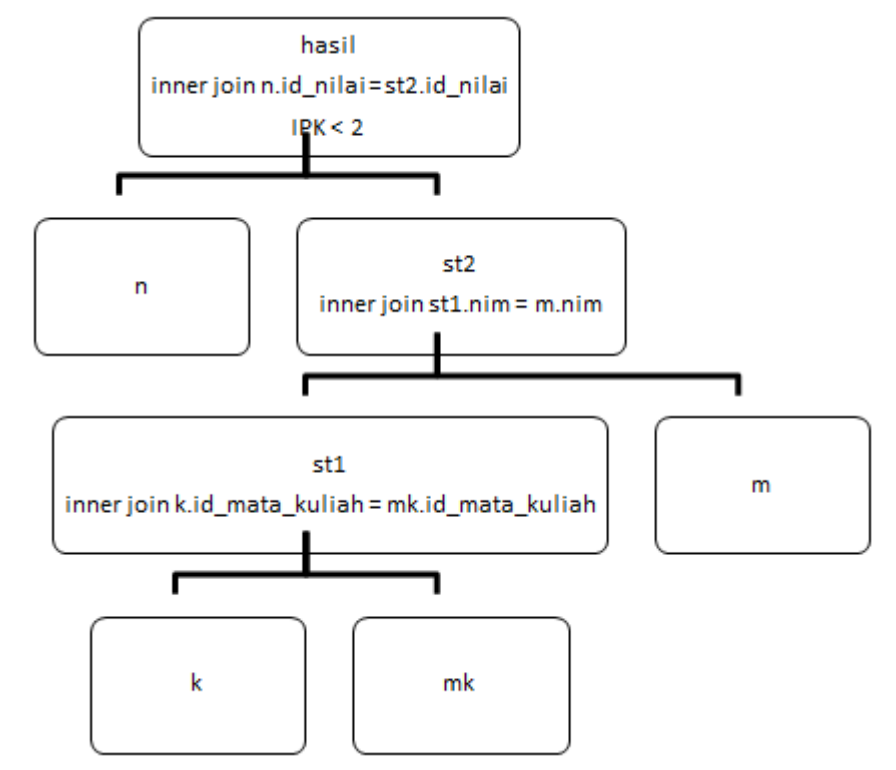
Untuk mendapatkan hasil sesuai permintaan dari *query* 2, dapatkan mahasiswa dengan yang pernah mendapatkan peringatan akademik pada tahun pertama. Maka dibutuhkan beberapa tabel yaitu *khs, mata_kuliah, m_nilai,* dan *mahasiswa*. Dalam hal ini tabel *khs* memiliki relasi dengan 3 tabel lainnya dimana 3 tabel tersebut hanya terelasi pada tabel *khs*. Sehingga lcs yang mungkin dari *query* 2 adalah:



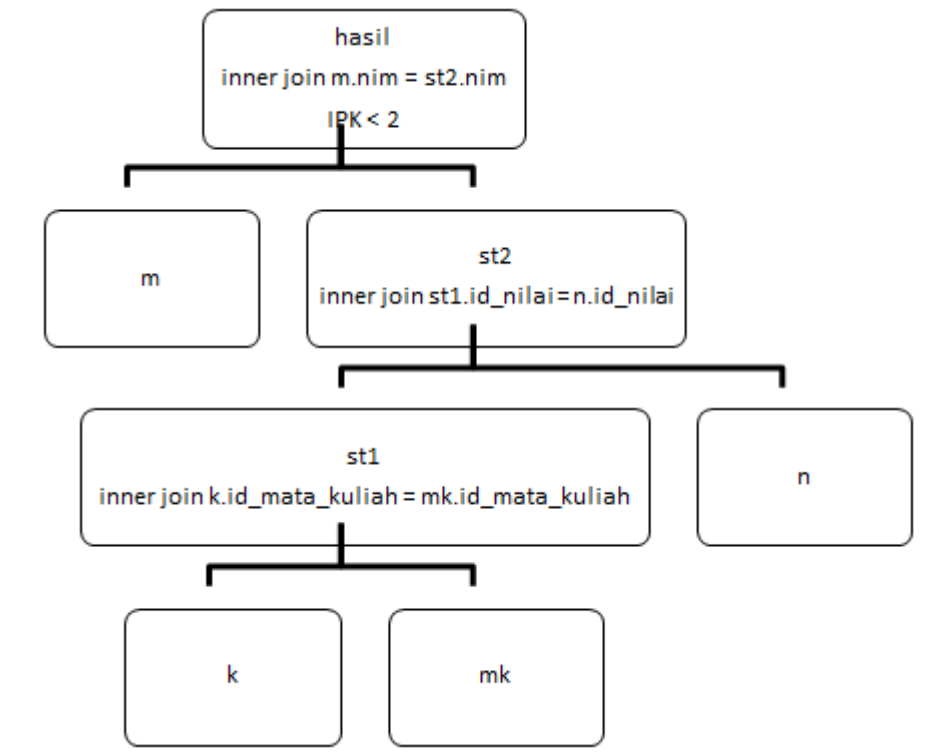
Gambar 3. 11 Rancangan A *query* 2



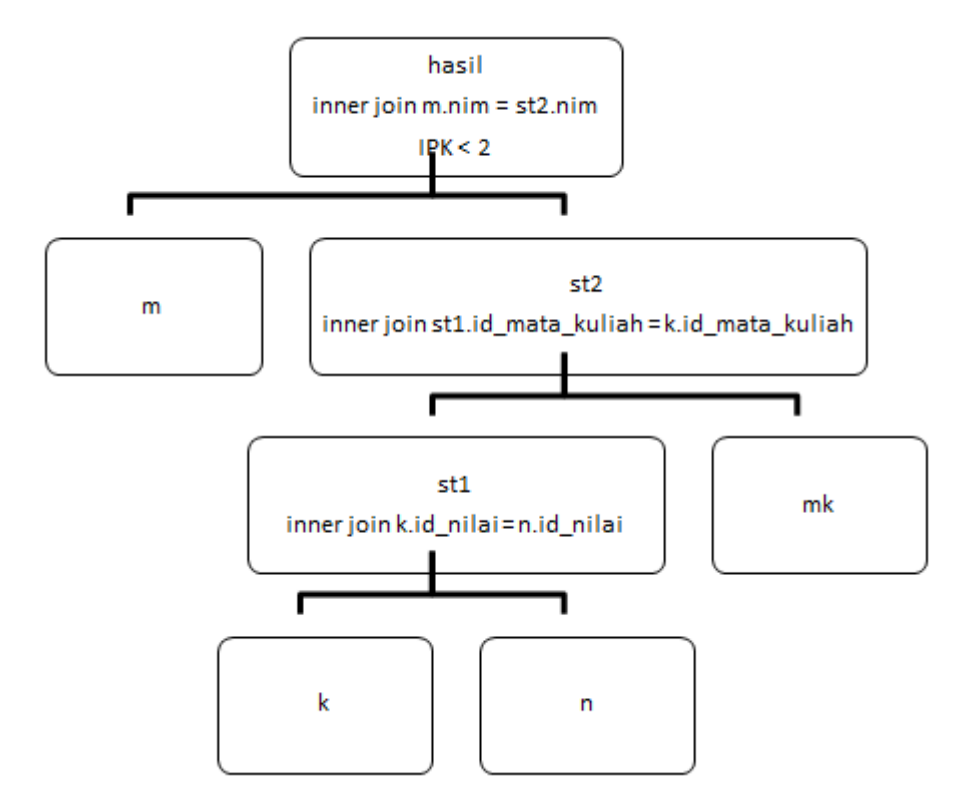
Gambar 3. 12 Rancangan B *query* 2



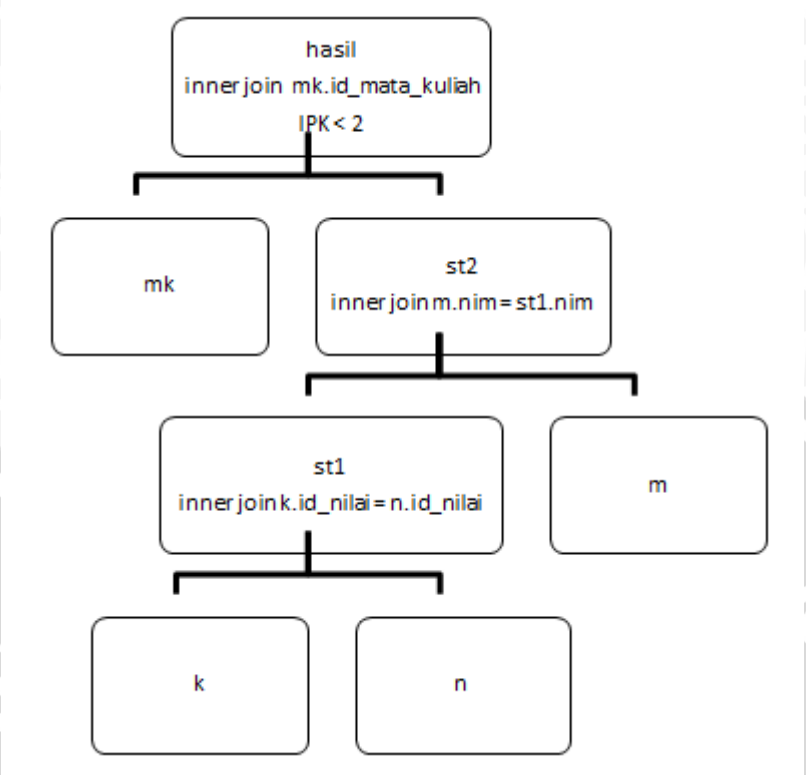
Gambar 3. 13 Rancangan C *query* 2



Gambar 3. 14 Rancangan D *query* 2



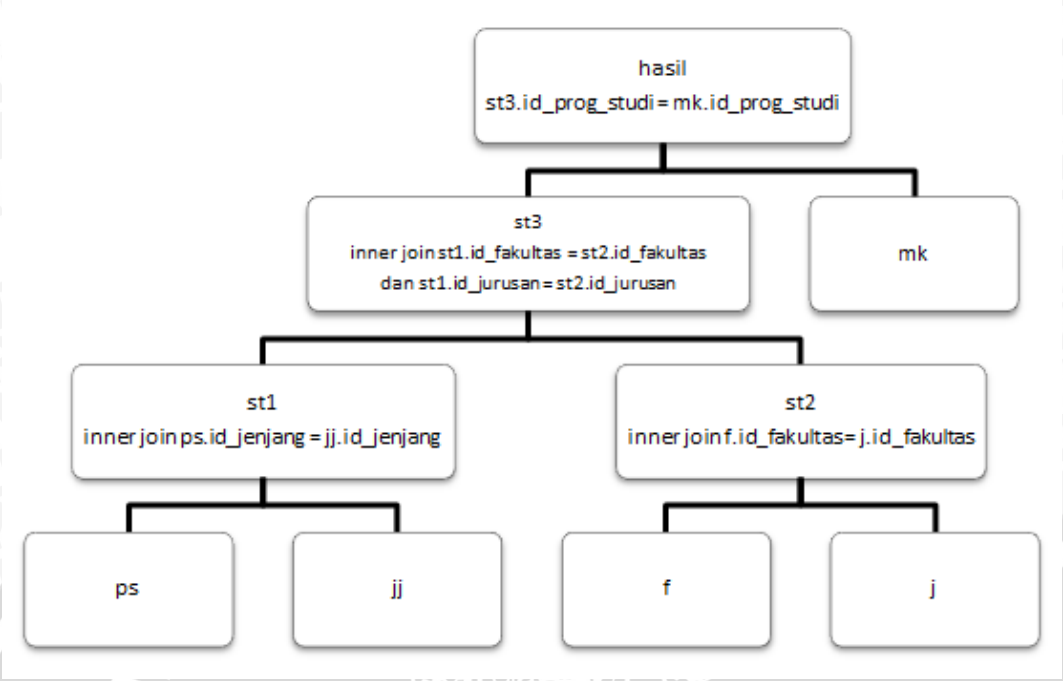
Gambar 3. 15 Rancangan E *query* 2



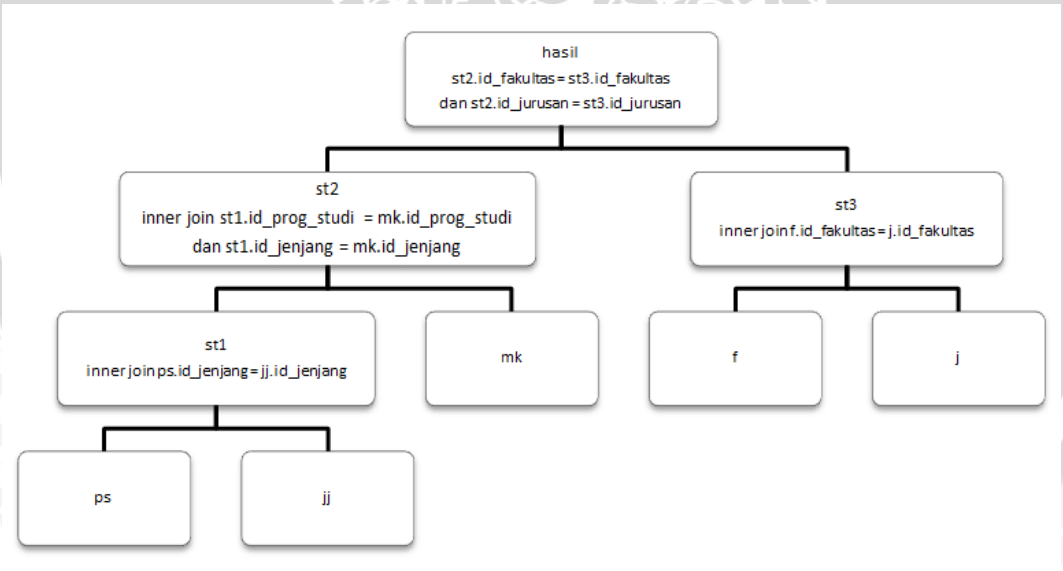
Gambar 3. 16 Rancangan F *query* 2

3.6.3. Model akses *query* 3

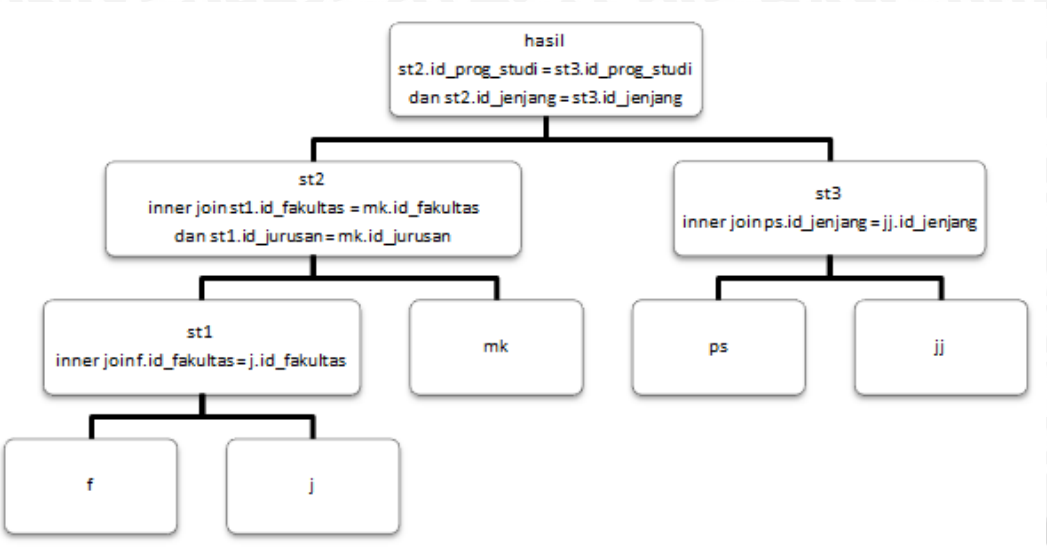
Untuk mendapatkan hasil sesuai permintaan dari *query* 3, dapatkan mata kuliah yang ditawarkan diseluruh universitas Brawijaya. Maka dibutuhkan beberapa tabel yaitu *mata_kuliah, m_fakultas, m_jurusan, m_jenjang* dan *m_prog_studi*. Dalam hal ini tabel *m_jenjang* berelasi dengan *m_prog_studi* begitu juga tabel *m_fakultas* berelasi dengan *m_jurusan*. Sehingga lcs yang mungkin dari *query* 3 adalah



Gambar 3. 17 Rancangan A *query* 3



Gambar 3. 18 Rancangan B *query* 3



Gambar 3. 19 Rancangan C *query* 3

3.7. Uji Coba

Dalam penelitian ini rancangan LCS yang ada akan diimplementasikan dalam *query* dengan DBMS MsSQL Server 2008. Data yang digunakan adalah data *dummy* yang telah dirancang sebelumnya sebanyak total 1393907 data dengan rincian sebagai berikut:

Tabel 3. 3 data penelitian

Total execution time	Tabel	Jumlah data	
1	m_fakultas	14	307122
	m_jurusan	6	
	m_prog_studi	15	
	mata_kuliah	789	
	mahasiswa	9501	
	khs	296789	
	m_nilai	8	
2	m_fakultas	14	260616

	m_jurusan	9	
	m_prog_studi	14	
	mata_kuliah	703	
	mahasiswa	9417	
	khs	250451	
	m_nilai	8	
3	m_fakultas	14	277076
	m_jurusan	5	
	m_prog_studi	12	
	mata_kuliah	460	
	mahasiswa	9504	
	khs	267073	
	m_nilai	8	
4	m_fakultas	14	269042
	m_jurusan	9	
	m_prog_studi	11	
	mata_kuliah	571	
	mahasiswa	9244	
	khs	259185	
	m_nilai	8	
5	m_fakultas	14	280051
	m_jurusan	5	
	m_prog_studi	16	
	mata_kuliah	701	
	mahasiswa	9291	
	khs	270016	
	m_nilai	8	
Total			1393907

3.8. Evaluasi dan Analisa

Untuk evaluasi, penulis akan menggunakan salah satu fitur *MsSQL Server* yaitu *statistics database*. Rancangan LCS yang mungkin, yang dihasilkan melalui pendekatan *semijoin*, akan dibandingkan masing-masing dan dibandingkan dengan *query* konvensional yang ada sebelumnya. Aspek yang dibandingkan adalah total time masing-masing rancangan *query*.

BAB IV
IMPLEMENTASI

4.1 Lingkungan Implementasi

Lingkungan implementasi yang akan dijelaskan pada bab ini adalah lingkungan implementasi perangkat keras dan lunak.

4.1.1 Lingkungan Perangkat Keras

Perangkat keras yang digunakan dalam penelitian ini adalah:

1. Komputer *Dekstop Core* i3 3,2GHz
2. Memori DDR3 4GB
3. Hardisk Eksternal 1 TB
4. Monitor 14”

4.1.2 Lingkungan Perangkat Lunak

Perangkat lunak yang digunakan dalam penelitian ini adalah:

1. Sistem Operasi *Windows 7 Ultimate* 32 *bit*
2. *Microsoft SQL Server* 2008
3. *Microsoft Excel* 2010
4. *VMWare* 9

4.2 Implementasi Virtual Komputer (Situs)

Dalam penelitian ini, situs berupa virtual komputer menggunakan VMWare 9 dengan spesifikasi:

1. Sistem Operasi *Windows 7 Ultimate* 32 bit
2. *Single Core*
3. Memori 512 MB
4. *Storage* 60 GB
5. *Network Adapter Host Only* 1 MBps

Virtual Machine ini dibuat sebanyak 5 buah dan akan dijalankan menggunakan *network adapter host only* sehingga berada pada *subnet* yang sama sehingga semua site bisa terhubung dalam satu jaringan komputer *virtual*.

Dalam penelitian ini, IP yang digunakan oleh situs-situs tersebut adalah sebagai berikut:

Tabel 4. 1 Daftar IP seluruh situs

Situs 1	192.168.216.11
Situs 2	192.168.216.12
Situs 3	192.168.216.13
Situs 4	192.168.216.14
Situs 5	192.168.216.15
Situs untuk pembuatan laporan	192.168.216.16

4.3 Implementasi Basis Data

Dalam penelitian ini basis data yang digunakan adalah basis data SIAKAD studi kasus Universitas Brawijaya. ERD yang ada pada bab III akan diimplementasikan pada DBMS MSSQL Server 2008. *Source code Data definition languange* (DDL) sebagaimana terlampir pada *source code* 4.1.

4.4 Implementasi Pembuatan Data *Dummy*

Dalam penelitian ini, data *dummy* dibuat menggunakan *Microsoft Excel* 2010 dengan formula yang ada dalamnya. Beberapa fungsi yang digunakan terdapat pada tabel 4.2.

Tabel 4. 2 Formula yang digunakan dalam pembuatan data dummy menggunakan Microsoft Excel 2010

1	if(logical_test;[value_if_true];[value_if_false])	Formula untuk mengembalikan nilai benar atau salah
2	mod(number;divisor)	Formula modulus untuk

		menentukan nama mahasiswa atau mahasiswi berdasar NIM
3	right(text;[num_chars])	Formula untuk mengambil substring dimulai dari kanan.
4	randbetween(bottom;top)	Formula untuk mengembalikan bilangan random dengan batas bawah dan batas atas
5	&	Untuk menggabungkan string dalam cell

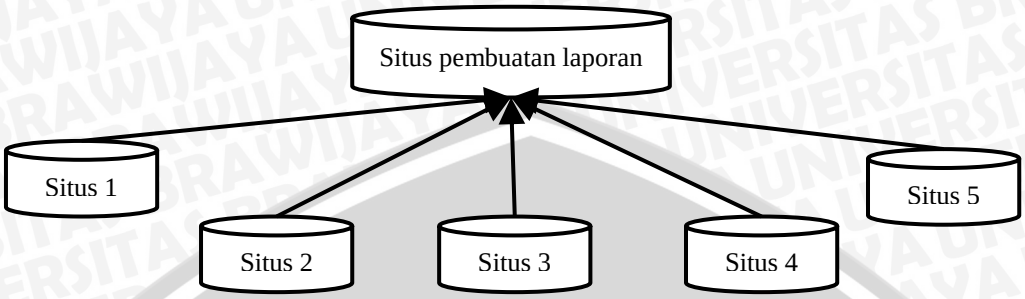
4.5 Implementasi Model Akses

Seperti yang telah dijelaskan pada bab 3 sub bab 3.6, bahwa terdapat beberapa rancangan LCS untuk masing-masing *query* 1, 2, dan 3. Dalam implementasinya dengan SQL, penulis menggunakan *view*. *View* dengan pertimbangan cukup ringkas untuk mengemas *query* yang panjang. Untuk implementasi SQLnya pada *query* 1 sebagaimana terlampir pada *source code* 2-7. Untuk *query* 2 sebagaimana terlampir pada *source code* 8-13. Dan untuk *query* 3, sebagaimana terlampir pada *source code* 14-16.

4.5.1.1 Implementasi *Linked server*

Untuk membuat setiap situs terkoneksi diperlukan suatu penghubung. Dalam MSSQL Server, hal ini bisa dilakukan dengan membuat *linked server*. *Linked server* adalah metode untuk *remote* suatu basis data dalam mesin lain dimana *privilege* nya sesuai dengan *user* yang diberikan. Jadi dalam implementasi *linked server* perlu *user* pada basis data yang di*remote* untuk *login*.

Sesuai dengan model global pada bab 3 gambar 3.4



Gambar 3.4 model akses saat *query* dieksekusi. Semua data dikirim ke situs pembuatan pelaporan

Maka dibutuhkan 5 *linked server* dari masing-masing situs ke situs pembuatan pelaporan. *Source code* implementasi *linked server* terlampir pada *source code* 17.

Sebelum membuat *linked server* terlebih dahulu harus memiliki *user* yang akan di*remote*. *Source code* untuk membuat *user* yang nantinya diremote terlampir pada *source code* 18.

4.5 Implementasi Pengujian

Pada pengujian, data yang digunakan adalah data *dummy* sudah dibuat seperti dijelaskan pada sub bab 3.8. dalam pengujiannya dilakukan sebanyak 10 kali masing-masing rancangan dan diambil nilai rata-ratanya. Setelah didapatkan rata-rata akan dibandingkan *total time* ketika melakukan eksekusi *query* dan diambil yang paling rendah.

BAB V
HASIL DAN ANALISA

5.1 Hasil Pengujian

Pada bab 4 telah dibahas tentang implementasi model akses, yaitu rancangan lcs dari pendekatan berbasis semi*join*. Terdapat beberapa rancangan lcs untuk masing-masing *query*. Masing-masing rancangan dan rancangan konvensional akan dibandingkan *total time* nya sesuai dengan sistematika yang telah dipaparkan sebelumnya dimana *total time* terendah adalah model akses paling optimal.

5.1.1. Hasil Pengujian *Query* 1

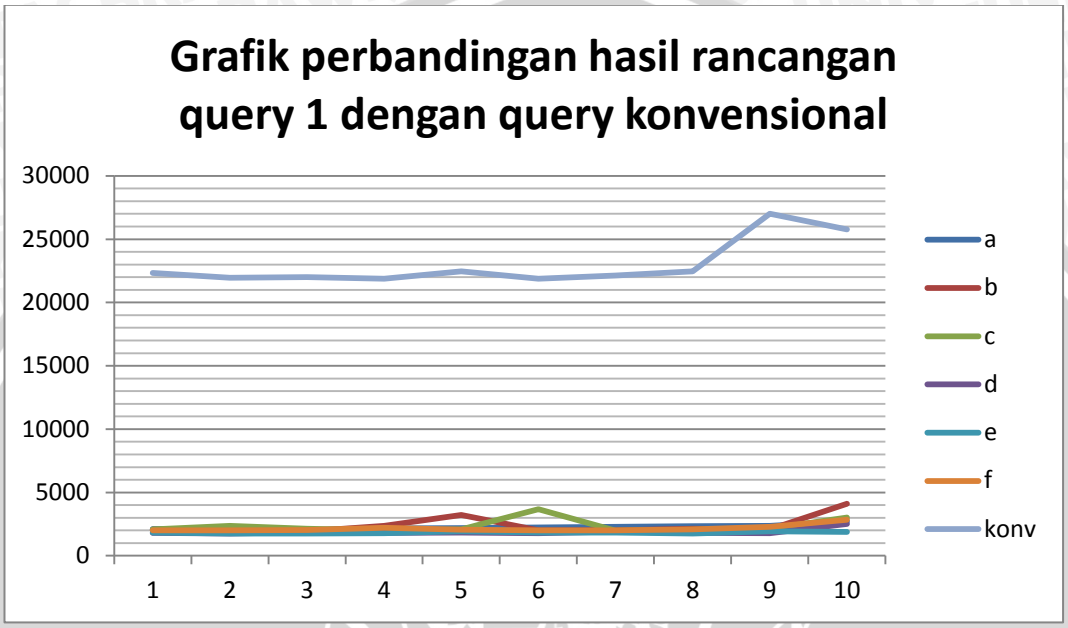
Query 1 adalah terkait menampilkan data mahasiswa dengan IPK *cumlaude* yang ada diseluruh Universitas Brawijaya. Sesuai dengan perancangan sebelumnya bahwa terdapat 6 rancangan baru yang akan dibandingkan. Untuk pengujian *query* dilakukan 10 kali. Nilai yang diambil adalah nilai rata-rata dari 10 kali proses *query* yang dilakukan. Hasil uji coba dapat dilihat pada tabel 5.1

Tabel 5. 1 Hasil *total time* eksekusi rancangan *query* 1

Query	Percobaan (milisecond)										
	1	2	3	4	5	6	7	8	9	10	Rata[2]
Konv	22338	21963	22004	21885	22467	21884	22119	22467	27001	25772	22990.0
Plan A	2103	2088	2038	2171	2192	2239	2293	2328	2359	2719	2253.0
Plan B	2053	1995	1930	2354	3209	1999	2009	2124	2071	4118	2386.2
Plan C	2079	2360	2138	1996	2093	3672	1976	2048	2133	3031	2352.6
Plan D	1798	1757	1851	1807	1834	1778	1906	1808	1781	2513	1883.3
Plan E	1822	1764	1746	1786	1924	1839	1820	1766	1928	1869	1826.4
Plan F	2012	2008	2040	2207	2062	2008	1995	2091	2287	2832	2154.2

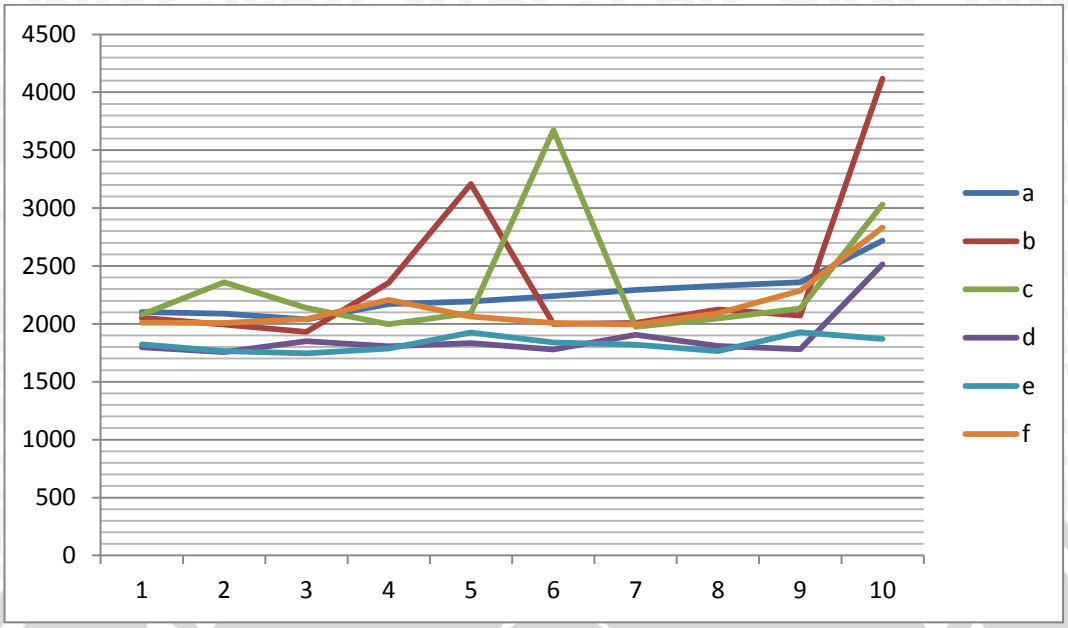
Data yang digunakan adalah data tabel khs, m_nilai, mata_kuliah, dan mahasiswa dengan total data sebanyak 1393735 data. Baris dengan blok warna kuning merupakan rancangan paling optimal.

Perbandingan *total time* masing-masing rancangan dapat dilihat pada grafik 5.1.



Gambar 5. 1 Perbandingan hasil eksekusi rancangan *query* 1 dengan *query* konvensional

Dibandingkan dengan *query* konvensional, maka rancangan yang dieksekusi hasilnya sangat jauh berbeda. Sedangkan perbandingan eksekusi antar rancangan sendiri dapat dilihat pada grafik 5.2.



Gambar 5. 2 Grafik Perbandingan hasil eksekusi rancangan *query* 1

Dari grafik diatas dapat dilihat bahwa rancangan E adalah yang paling konstan. Sehingga ketika diambil rata-rata dari 10 kali percobaan *query*, didapatkan rancangan E dengan hasil paling rendah. Dari sini diambil disimpulkan untuk rancangan E adalah rancangan paling optimal.

5.1.2. Hasil Pengujian *Query* 2

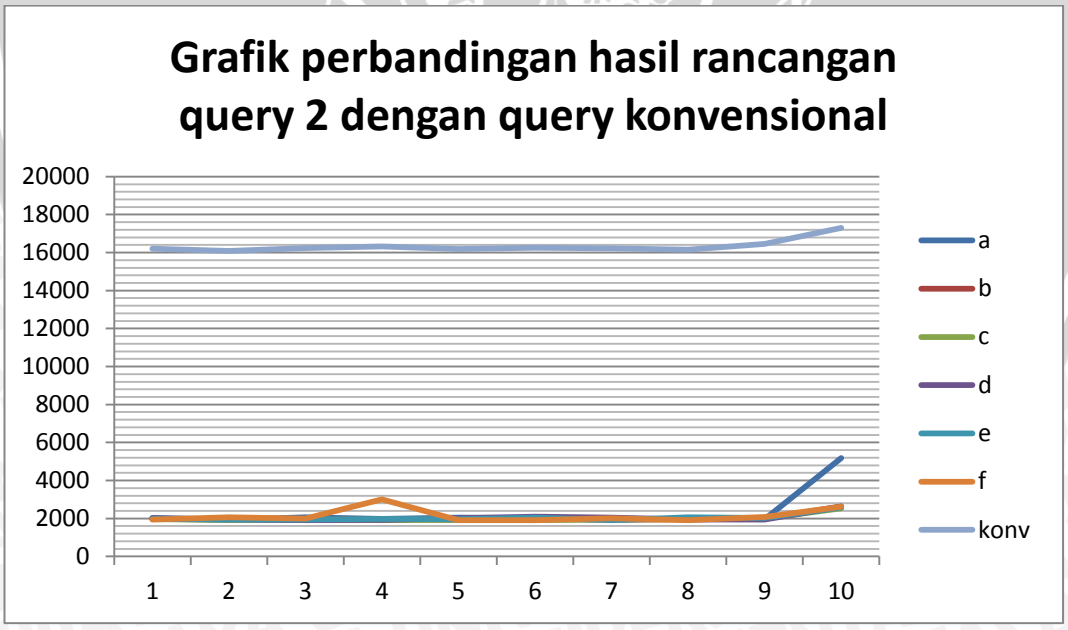
Query 2 adalah terkait menampilkan data mahasiswa yang pernah mendapatkan peringatan akademik pada tahun pertama. Secara tidak langsung IPK selama semester 1 dan 2 tidak lebih dari 2,00. Untuk pengujian *query* dilakukan 10 kali. Nilai yang diambil adalah nilai rata-rata dari 10 kali proses *query* yang dilakukan. Dari 6 model akses untuk *query* 2 didapatkan hasil:

Tabel 5. 2 Hasil *total time* eksekusi rancangan *query* 2

Query	Percobaan (milisecond)										
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**Rata[2]**
Konv	16211	16079	16239	16319	16182	16255	16220	16152	16453	17296	16340.6
Plan A	2021	1935	2065	1973	2038	2021	1961	1979	1966	5185	2314.4
Plan B	1983	1946	1922	1977	1923	2001	1935	1968	2002	2630	2028.7
Plan C	1956	1929	1953	1917	1927	1909	1948	1964	1999	2533	2003.5
Plan D	2008	1970	1915	1936	2010	2093	2042	1948	1947	2616	2048.5
Plan E	1977	1948	1952	1980	1963	2050	1913	2068	2022	2579	2045.2
Plan F	1942	2062	1997	3014	1908	1910	2014	1905	2073	2620	2144.5

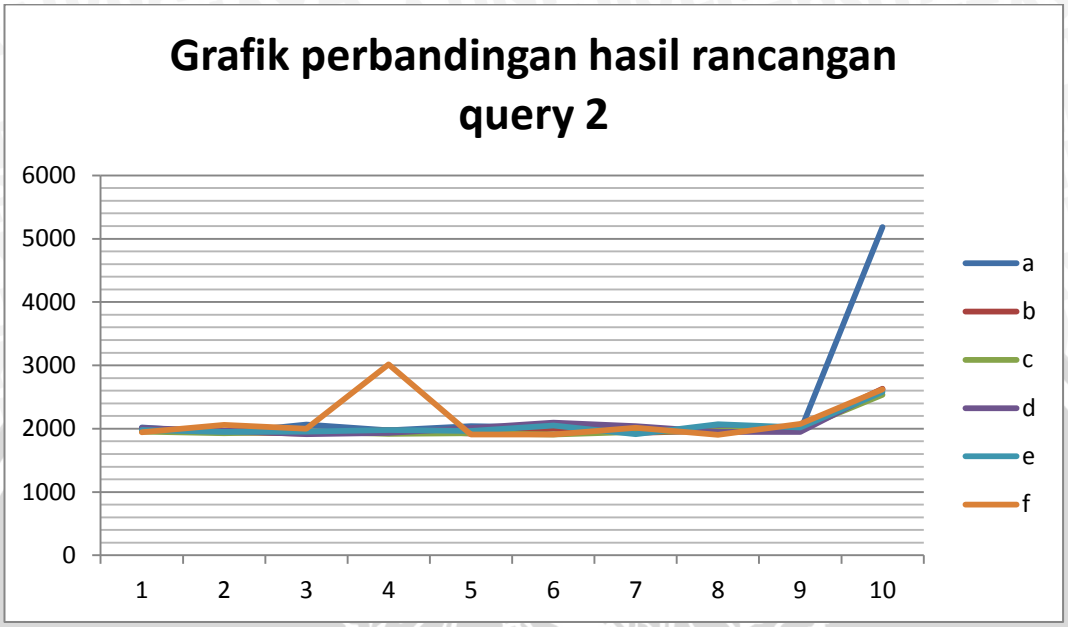
Data yang digunakan adalah data tabel khs, m_nilai, mata_kuliah, dan mahasiswa dengan total data sebanyak 1393735 data. Baris dengan blok warna kuning merupakan rancangan paling optimal.

Perbandingan *total time* masing-masing rancangan dapat dilihat pada grafik 5.3.



Gambar 5. 3 Grafik Perbandingan hasil eksekusi rancangan *query* 2 dengan *query* konvensional

Seperti hal nya pada *query* 1, bahwa *query* konvensional sangat jauh berbeda kecepatan eksekusinya dibandingkan rancangan yang ada. Sedangkan perbandingan eksekusi antar rancangan sendiri dapat dilihat pada grafik 5.4.



Gambar 5. 4 Grafik Perbandingan hasil eksekusi rancangan *query* 2

Dari grafik diatas dapat dilihat bahwa rancangan C adalah yang paling konstan. Sehingga ketika diambil rata-rata dari 10 kali percobaan *query*, didapatkan rancangan C dengan hasil paling rendah. Dari sini diambil disimpulkan untuk rancangan C adalah rancangan paling optimal.

5.1.3. Hasil Pengujian *Query* 3

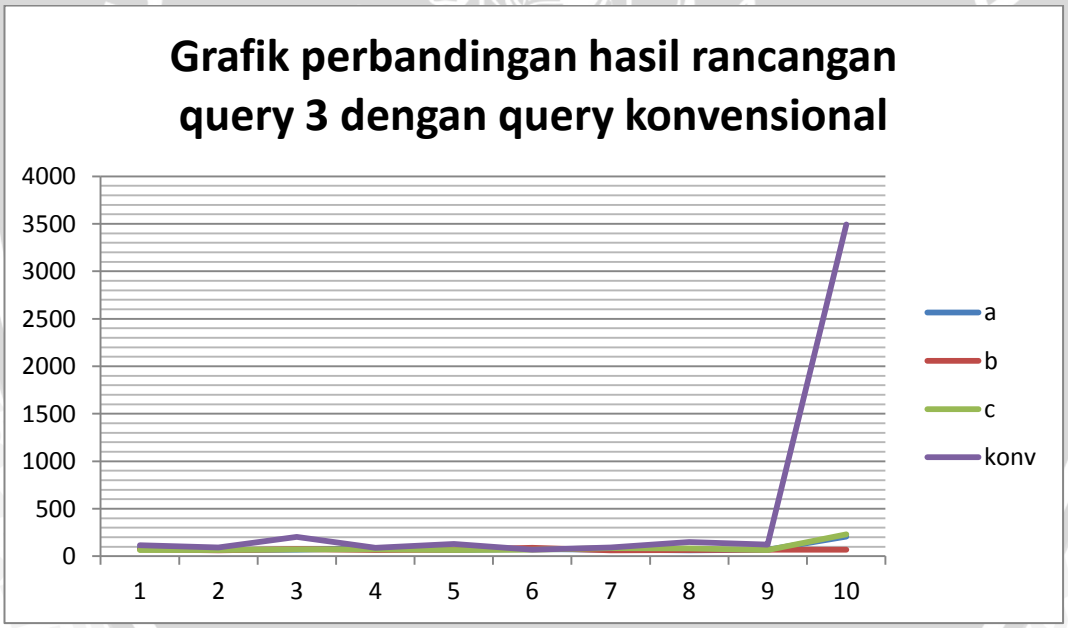
Query 3 adalah terkait menampilkan data seluruh mata kuliah yang ada di universitas Bwrawijaya. Untuk pengujian *query* dilakukan 10 kali. Nilai yang diambil adalah nilai rata-rata dari 10 kali proses *query* yang dilakukan. Dari 4 model akses untuk *query* 2 didapatkan hasil:

Tabel 5. 3 Hasil *total time* eksekusi rancangan *query* 2

Query	Percobaan (milisecond)										
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**Rata^2**
Konv	115	94	202	89	128	68	91	148	122	3495	455.2
Plan A	88	64	70	84	65	77	69	62	69	207	85.5
Plan B	72	63	74	67	69	90	63	64	70	70	70.2
Plan C	66	67	73	71	65	70	86	82	67	232	87.9

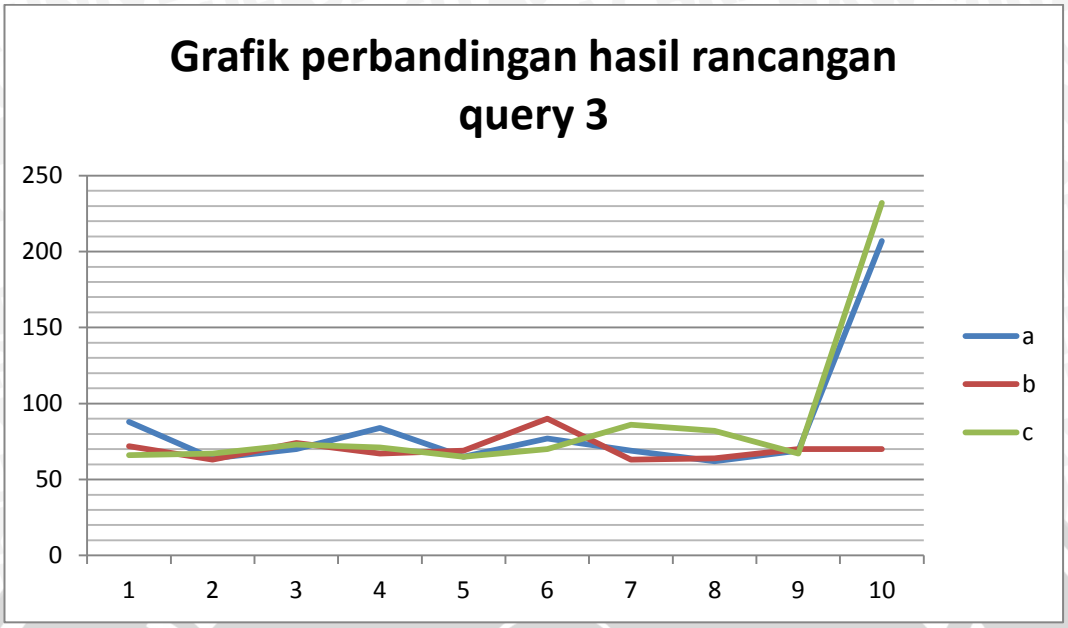
Data yang digunakan adalah data tabel khs, m_nilai, mata_kuliah, dan mahasiswa dengan total data sebanyak 1393735 data. Baris dengan blok warna kuning merupakan rancangan paling optimal.

Perbandingan *total time* masing-masing rancangan dapat dilihat pada grafik 5.5.



Gambar 5. 5 Grafik perbandingan hasil eksekusi rancangan *query* 3

Perbandingan *total time* masing-masing rancangan dapat dilihat pada grafik 5.6.

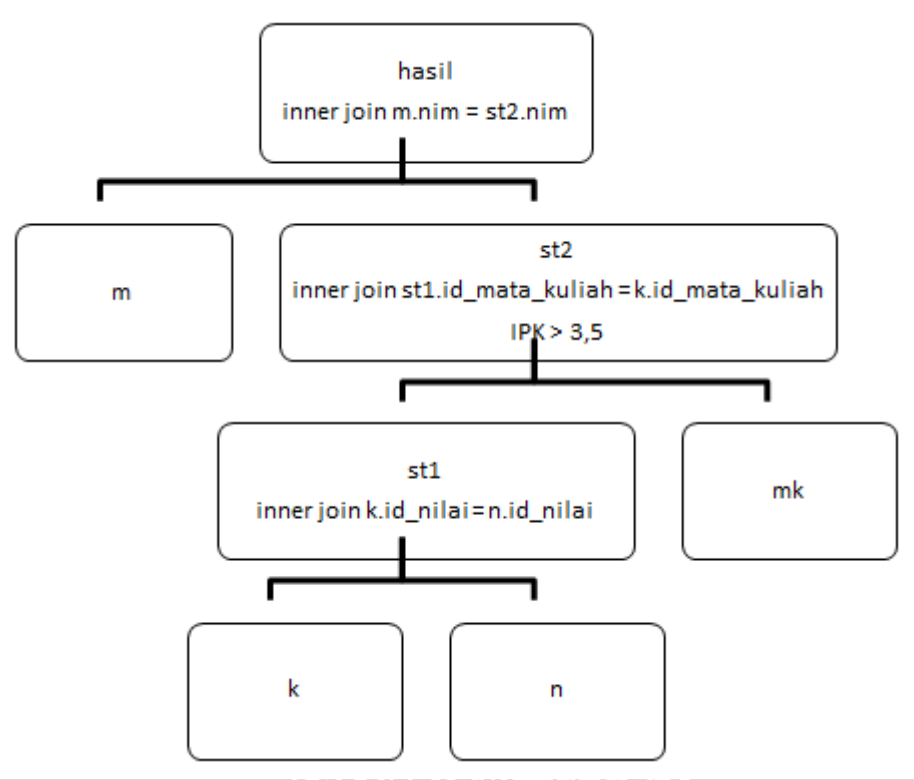


Gambar 5. 6 Grafik perbandingan hasil eksekusi rancangan *query* 3

Dari grafik diatas dapat dilihat bahwa rancangan B adalah yang paling konstan. Sehingga ketika diambil rata-rata dari 10 kali percobaan *query*, didapatkan rancangan B dengan hasil paling rendah. Dari sini diambil disimpulkan untuk rancangan B adalah rancangan paling optimal.

5.2 Analisa Hasil

Pada *query* 1, dengan *query* menampilkan data mahasiswa dengan IPK *cumlaud* model yang paling optimal adalah rancangan E, bila dibandingkan dengan model rancangan yang lain seperti terlihat pada gambar 5.7

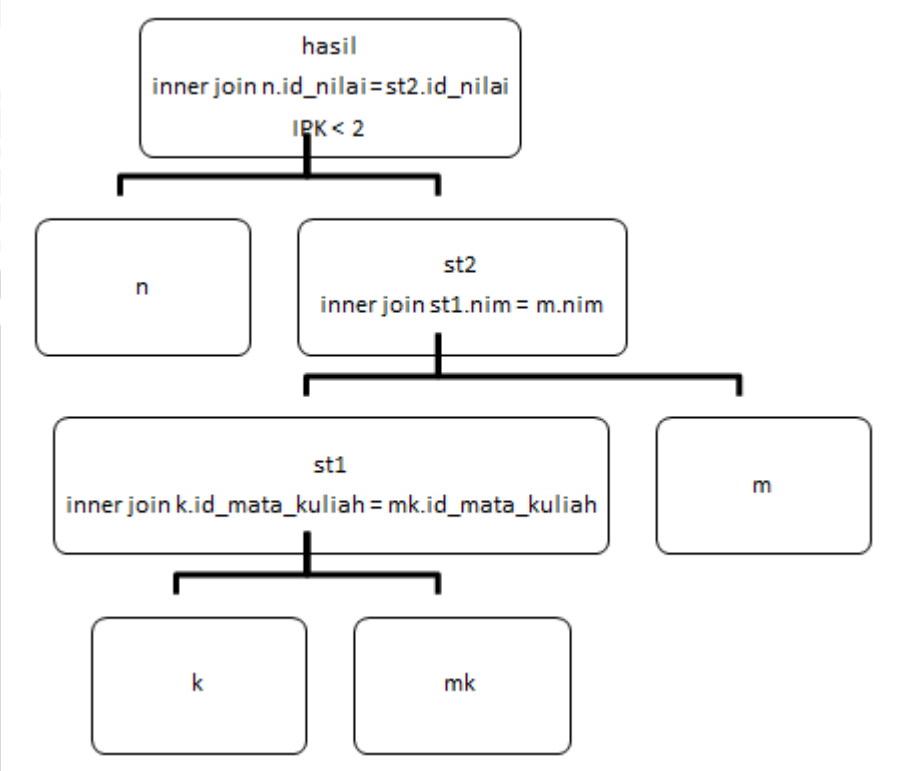


Gambar 5. 7 Rancangan E *query* 1

Pada rancangan ini, didapatkan hasil yang cepat dikarenakan order *join* yang tepat dan mendahulukan proses seleksi. Pada rancangan ini, tabel khs (k) dengan jumlah data sebanyak 1310148 di*join*kan dengan tabel nilai (n) dengan jumlah data sebanyak 8 data. Dari proses ini dihasilkan data sebanyak 1310148 data. Hasil ini ditampung dalam *view sub tree* 1 (st) kemudian di*join*kan kembali dengan tabel mata kuliah (mk) yang memiliki data sejumlah 3224 data. Dari *join* ini dihasilkan 1310148 data, namun disini terdapat proses seleksi yaitu pemilihan mahasiswa dengan IPK *cumlaude* sehingga dari 1310148 data, yang memenuhi syarat seleksi adalah 62 data yang disimpan pad *sub tree* 2 (st2). Data pada st2 di*join*kan kembali dengan tabel mahasiswa (m) dengan jumlah data 46957 data

untuk mendapatkan nama mahasiswa yang memenuhi IPK *cumlaude*. Dari *join* tersebut didapatkan hasil 62 data yang memenuhi. Pada rancangan ini seleksi dilakukan pada nilai IPK dimana nilai tersebut didapatkan dari hasil *join* antara tabel khs, mk, dan n. Penerapan seleksi mempengaruhi jumlah data yang diproses nantinya, tentunya proses ini juga berpengaruh pada kecepatan proses data selanjutnya sehingga eksekusi *query* bisa menjadi lebih cepat. Pada *query* konvensional untuk mengambil data yang sama dibutuhkan waktu sekitar 22990 *miliseconds*, sedangkan dengan rancangan E ini hanya dibutuhkan waktu sekitar 1826,4 *miliseconds.* Dari waktu eksekusi yang dibutuhkan bisa dilihat bahwa waktu eksekusi rancangan *query* E 12,59 kali lebih cepat dari pada *query* konvensional, sehingga dari sini bisa disimpulkan bahwa dengan metode ini kecepatan naik sekitar 1259% didapatkan dari hasil perhitungan waktu eksekusi konvensional dibagi waktu eksekusi rancangan E dikalikan 100%.

Pada *query* 2, dengan *query* menampilkan data mahasiswa yang pernah mendapatkan peringatan tahun pertama model yang paling optimal adalah rancangan C

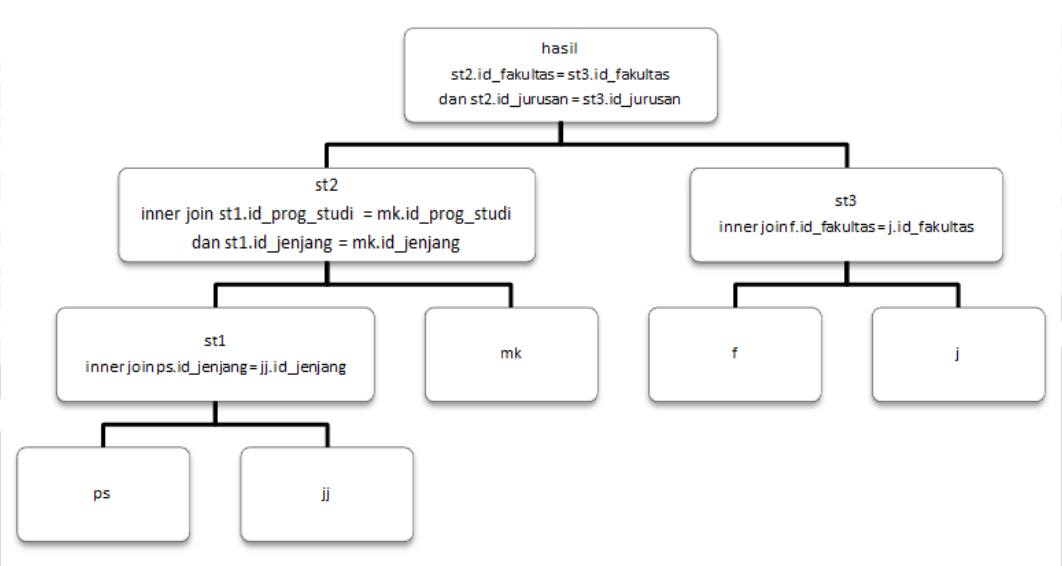


Gambar 5. 8 Rancangan C *query* 2

Pada *query* 2 tujuannya mendapatkan mahasiswa yang pernah mendapatkan peringatan akademik tahun pe rtama. Pada rancangan ini, tabel khs (k) dengan jumlah data sebanyak 1310148 di*join*kan dengan tabel mata kuliah (mk) yang memiliki data sejumlah 3224 data. Dari proses ini dihasilkan data sebanyak 1310148 data. Hasil ini ditampung dalam *view sub tree* 1 (st) kemudian di*join*kan kembali dengan tabel mahasiswa (m) dengan jumlah data 46957 data. Disini terdapat proses seleksi yaitu mendapatkan nilai untuk tahun pertama saja, yaitu semester 1 dan semester 2. Hal ini dikarenakan dalam *query* ini akan didaptkan mahasiswa yang pernah mendapatkan peringatan akademik pada tahun pertama, sehingga dari proses seleksi didapatkan hasil sebanyak 337358 data. Hasil ini ditampung sepenuhnya pada *sub tree* 2 (st2), kemudian baru di*join*

dengan tabel nilai (n) yang memiliki data sejulah 8 data untuk dihitung IPK mahasiswa yang tidak memenuhi syarat tersebut. Dari *join* tersebut didapatkan hasil 8065 data yang memenuhi. Pada rancangan ini terdapat 2 kali proses seleksi, yaitu seleksi semester yang akan diproses, kemudian seleksi pada IPK yang memenuhi syarat seleksi sebagai IPK yang memenuhi syarat pelanggaran akademik. Telah dijelaskan sebelumnya bahwa penerapan seleksi mempengaruhi jumlah data yang diproses nantinya, tentunya proses ini juga berpengaruh pada kecepatan proses data selanjutnya sehingga eksekusi *query* bisa menjadi lebih cepat. Pada *query* konvensional untuk mengambil data yang sama dibutuhkan waktu sekitar 16340 *miliseconds*, sedangkan dengan rancangan E ini hanya dibutuhkan waktu sekitar 2003,5 *miliseconds.* Dari waktu eksekusi yang dibutuhkan bisa dilihat bahwa waktu eksekusi rancangan *query* C 8,16 kali lebih cepat dari pada *query* konvensional, sehingga dari sini bisa disimpulkan bahwa dengan metode ini kecepatan naik sekitar 816% didapatkan dari hasil perhitungan waktu eksekusi konvensional dibagi waktu eksekusi rancangan C dikalikan 100%.

Pada *query* 3, dengan *query* menampilkan data mata kuliah yang ditawarkan di universitas Brawijaya model yang paling optimal adalah rancangan B



Gambar 5. 9 rancangan B *query* 3

Berbeda dengan rancangan *query* sebelumnya, pada rancangan ini tidak ada proses seleksi seperti halnya ketika menentukan IPK seperti *query* sebelumnya. Pada rancangan ini hanya menganalisa pengaruh *join* order. Dalam prosesnya, sesuai relasi yang dimiliki maka tabel m_prog_studi (ps) dengan jumlah data sebanyak 68 data dan tabel m_jenjang (jj) dengan jumlah data sebanyak 1 di*join* terlebih dahulu baru di*join* kembali dengan tabel mata_kuliah (mk) dengan jumlah data sebanyak 3224 data. Hasil dari *join* ditampung pada *sub tree* 2 (st2) dengan data sebanyak 3224 data. Disisi lain tabel m_fakultas (m) dengan jumlah data sebanyak 13 data di*join* dengan tabel m_jurusan (j) dengan jumlah data sebanyak 35 data. Hasil dari *join* ditampung pada *sub tree* 3 (st3) dengan data sebanyak 35 data. Data pada st2 kemudian di*join* dengan st3 untuk mendapatkan hasil yang di*order*. Dari *join* tersebut didapatkan hasil 8065 data yang memenuhi. Pada *query* konvensional untuk mengambil data yang sama dibutuhkan waktu sekitar 455,2 *miliseconds*, sedangkan dengan rancangan E ini hanya dibutuhkan waktu sekitar 70,2 *miliseconds*. Dari waktu eksekusi yang

dibutuhkan bisa dilihat bahwa waktu eksekusi rancangan *query* B 6,48 kali lebih cepat dari pada *query* konvensional, sehingga dari sini bisa disimpulkan bahwa dengan metode ini kecepatan naik sekitar 648% didapatkan dari hasil perhitungan waktu eksekusi konvensional dibagi waktu eksekusi rancangan B dikalikan 100%.

Dari sini hasil yang didapatkan, bahwa metode SBA berhasil optimal untuk mengoptimasi *query* pada basis data terdistribusi.

BAB VI
KESIMPULAN DAN SARAN

6.1. Kesimpulan

Kesimpulan yang dapat diberikan dari penelitian ini adalah

1. Biaya komunikasi dalam basis data terdistribusi bisa diminimalkan, salah satu caranya adalah melakukan pendekatan berbasis *semijoin* dengan metode *Semijoin Based Approach* (SBA).
2. *Join order* bisa dioptimalkan dengan metode SBA, yaitu dengan memetakan semua rancangan yang mungkin. Rancangan tersebut dieksekusi dan diambil yang waktunya terendah. Dengan waktu yang paling rendah maka dikatakan *join order*nya sudah optimal dan efektif.
3. Percobaan pada data yang besar cukup optimal. Untuk *query* 1, yaitu mendapatkan data mahasiswa dengan IPK *cumlaude* dengan jumlah data sebesar 1393735 data, optimasi bisa mencapai 1259%. Untuk *query* 2, yaitu mendapatkan data mahasiswa yang pernah mendapatkan peringatan akademik pada tahun pertama, dengan jumlah data yang sama, optimasi hanya mencapai 816%. Sedangkan pada *query* 3, yaitu mendapatkan data mata kuliah yang ditawarkan di Universitas Brawijaya, dengan jumlah data 3393 data, optimasi mencapai 648%. Masing-masing *query* memiliki *order* yang berbeda, yang terpenting dalam metode SBA ini adalah uturan *order join*, dimana pertimbangannya adalah jumlah *data* pada tabel, dan juga mendahulukan proses *selection* dan *projection*. Dan metode ini tetap menghasilkan hasil yang optimal ketika diterapkan pada data besar maupun kecil.

6.2. Saran

Saran yang dapat diberikan untuk pengembangan penelitian ini adalah

1. Metode *Semijoin Based Approach* adalah salah satu metode optimasi *query* terdistribusi. Tetapi tidak menutup kemungkinan jika ada metode lain yang lebih optimal untuk diimplementasikan.

2. *Query* yang dilakukan berupa *single thread,* tidak menutup kemungkinan jika dilakukan secara *multi thread*.
3. Pembagian fragmen kedepannya dilakukan secara otomatis menggunakan *software* supaya prosesnya bisa lebih cepat dan efisien.

DAFTAR PUSTAKA

[VAL-11] Valduriez, P., & Ozsu, T. (2011). Principle of Distributed Database Systems. 3rd edition. Pearson Education, Inc. New York.

[AL0-09] Alom,B.M.M., Henskens, F., & Hannaford, H. (2009). Query Processing and Optimization in Distributed Database Systems. IJCSNS. VOL.9 No.9. halaman 143-152. 2009.

[SIL-04] Silberschatz, Korth, & Sudarshan (2004). Database System Concepts. 4th edition. The McGraw-Hill Companies. Murray Hill.

[SUG-10] Sugeng, Winarno. (2010). Jaringan Komputer dengan TCP/IP. Modula. Bandung.

[ELM-10] Elmasri, Ramez & Navathe, Shamkat (2010). Fundamentals od Database Systems. 6th edition. Addison Wesley.

[JIA-11] Jiang, Shun. (2011). Optimizing Join Query in Distributer Database. A Capstone Project. University of North Carolina Wilmington.

[BEL-92] Bell, D., Grimson, J. (1992). Distributed Database System. Addison-Wesley Publishing Company, Inc. Great Britain.

[CHI-84] Chin, W. C. 1984. An Optimization of Queries in Distributed Database Systems. JOURNAL OF PARALLEL AND DISTRIBUTED COMPUTING 3. halaman 137-157. 1986.

[WAN-09] Wang, Di. (2009). Efficient Query Optimization for Distributed Join in Database Federation. Faculty of the Worcester Polytechnic Institute.

[TAY-10] Taylor, R. (2010). Query Optimization for Distributed Database Systems. Computing Laboratory Project. University of Oxford.

[ABD-03] Abdullah, D. (2003). Query Optimization in Distributed Databases.Middle East Technical University.

Lampiran 1. *Source code* DDL SIAKAD Universitas

Source code 1 DDL SIAKAD Universitas

```
1   create table m_fakultas
2   (
3   id_fakultas varchar(3),
4   content varchar(45) not null,
5   constraint pk_m_fakultas primary key(id_fakultas)
6   )
7
8   create table m_jenjang
9   (
10  id_jenjang varchar(3),
11  content varchar(45) not null,
12  constraint pk_m_jenjang primary key (id_jenjang)
13  )
14
15  create table m_jurusan
16  (
17  id_jurusan varchar(3),
18  id_fakultas varchar(3),
19  id_jenjang varchar(3),
20  content varchar(45) not null,
21  constraint fk_m_fakultas foreign key (id_fakultas) references m_fakultas(id_fakultas),
22  constraint fk_m_jenjang  foreign key (id_jenjang) references m_jenjang(id_jenjang),
23  constraint pk_m_jurusan primary key (id_jurusan, id_fakultas, id_jenjang)
24  )
25
26  create table m_prog_studi
27  (
28  id_prog_studi varchar(3),
29  id_jurusan varchar(3),
30  id_fakultas varchar(3),
31  id_jenjang varchar(3),
32  content varchar(45) not null,
33  constraint fk_m_jurusan foreign key (id_jurusan, id_fakultas, id_jenjang) references m_jurusan(id_jurusan,
34  id_fakultas, id_jenjang),
35  constraint pk_m_prog_studi primary key (id_prog_studi, id_jurusan, id_fakultas, id_jenjang)
36  )
37
38  create table mahasiswa
39  (
40  nim varchar(15),
41  id_prog_studi varchar(3),
42  id_jurusan varchar(3),
43  id_fakultas varchar(3),
44  id_jenjang varchar(3),
45  nama varchar(45) not null,
46  angkatan int not null,
47  constraint fk_m_prog_studi foreign key (id_prog_studi, id_jurusan, id_fakultas, id_jenjang) references
48  m_prog_studi(id_prog_studi, id_jurusan, id_fakultas, id_jenjang),
49  constraint pk_mahasiswa primary key (nim)
50  )
51
52  create table mata_kuliah
53  (
54  id_mata_kuliah varchar(10),
55  id_fakultas varchar(3),
56  id_jenjang varchar(3),
57  id_jurusan varchar(3),
58  id_prog_studi varchar(3),
59  k_mk varchar(10),
60  thn_mk int,
61  nama varchar(45) not null,
62  sks int not null,
63  constraint uk_m_prog_studi2 unique (id_prog_studi, id_jurusan, id_fakultas, id_jenjang, k_mk, thn_mk),
```

```
64  constraint pk_mata_kuliah primary key (id_mata_kuliah),
65  constraint fk_m_prog_studi2 foreign key (id_prog_studi, id_jurusan, id_fakultas, id_jenjang) references
66  m_prog_studi(id_prog_studi, id_jurusan, id_fakultas, id_jenjang)
67  )
68
69  create table m_nilai
70  (
71  id_nilai varchar(2),
72  min_ int not null,
73  max_ int not null,
74  na int not null,
75  constraint pk_nilai primary key(id_nilai)
76  )
77
78  create table khs
79  (
80  nim varchar(15),
81  id_mata_kuliah varchar(10),
82  is_ganjil bit,
83  is_genap bit,
84  thn_akademik int not null,
85  id_nilai varchar(2),
86  constraint fk_m_nilai foreign key (id_nilai) references m_nilai(id_nilai),
87  constraint fk_m_mata_kuliah foreign key (id_mata_kuliah) references mata_kuliah(id_mata_kuliah),
88  constraint fk_mahasiswa foreign key (nim) references mahasiswa(nim),
89  )
```

Lampiran 2. *Source code* Implementasi SQL Model *Query* 1

Source code 2 Rancangan A

```
1   create view q1a1
2   as
3   select      m.nim, m.nama,
4                       k.id_mata_kuliah, k.id_nilai
5   from        khs k inner join mahasiswa m
6                       on k.nim = m.nim
7
8   create view q1a2
9   as
10  select      a1.nim, a1.nama, a1.id_nilai,
11                      mk.sks
12  from        q1A1 a1 inner join mata_kuliah mk
13                      on a1.id_mata_kuliah = mk.id_mata_kuliah
14
15  create view q1a3
16  as
17  select      a2.nim, a2.nama,
18                      sum(n.na * a2.sks)/ sum(a2.sks) as ipk
19  from        q1a2 a2 inner join m_nilai n
20                      on a2.id_nilai = n.id_nilai
21  group by a2.nim, a2.nama
22  having      sum(n.na * a2.sks)/ sum(a2.sks) > 3.5
```

Source code 3 Rancangan B

```
1   create view q1b1
2   as
3   select      m.nim, m.nama,
4                       k.id_mata_kuliah, k.id_nilai
5   from        khs k inner join mahasiswa m
6                       on k.nim = m.nim
7
8   create view q1b2
9   as
10  select      b1.nim, b1.nama, b1.id_mata_kuliah,
11                      n.na
12  from        q1b1 b1 inner join m_nilai n
13                      on b1.id_nilai = n.id_nilai
14
15  create view q1b3
16  as
17  select      b2.nim, b2.nama,
18                      sum(b2.na * mk.sks)/ sum(mk.sks) as ipk
19  from        q1b2 b2 inner join mata_kuliah mk
20                      on b2.id_mata_kuliah = mk.id_mata_kuliah
21  group by b2.nim, b2.nama
22  having      sum(b2.na * mk.sks)/ sum(mk.sks) > 3.5
```

Source code 4 Rancangan C

```
1   create view q1c1
2   as
3   select      k.nim, k.id_nilai, mk.sks
4   from        khs k inner join mata_kuliah mk
5                       on k.id_mata_kuliah = mk.id_mata_kuliah
6
7   create view q1c2
8   as
9   select      c1.id_nilai, c1.sks, c1.nim, m.nama
```

```
10  from    q1c1 c1 inner join mahasiswa m
11                  on c1.nim = m.nim
12
13  create view q1c3
14  as
15  select  c2.nim, c2.nama,
16                  sum(n.na * c2.sks)/ sum(c2.sks) as ipk
17  from    q1c2 c2 inner join m_nilai n
18                  on c2.id_nilai = n.id_nilai
19  group by c2.nim, c2.nama
20  having  sum(n.na * c2.sks)/ sum(c2.sks) > 3.5
```

Source code 5 Rancangan D

```
1   create view q1d1
2   as
3   select  k.nim, k.id_nilai, mk.sks
4   from    khs k inner join mata_kuliah mk
5                   on k.id_mata_kuliah = mk.id_mata_kuliah
6
7   create view q1d2
8   as
9   select  d1.nim,
10                  sum(n.na * d1.sks)/ sum(d1.sks) as ipk
11  from    q1d1 d1 inner join m_nilai n
12                  on d1.id_nilai = n.id_nilai
13  group by d1.nim
14  having  sum(n.na * d1.sks)/ sum(d1.sks) > 3.5
15
16  create view q1d3
17  as
18  select  d2.nim, m.nama, d2.ipk
19  from    q1d2 d2 inner join mahasiswa m
20                  on d2.nim = m.nim
```

Source code 6 Rancangan E

```
1   create view q1e1
2   as
3   select  k.nim, k.id_mata_kuliah, n.na
4   from    khs k inner join m_nilai n
5                   on k.id_nilai = n.id_nilai
6
7   create view q1e2
8   as
9   select  e1.nim,
10                  sum(e1.na * mk.sks)/ sum(mk.sks) as ipk
11  from    q1e1 e1 inner join mata_kuliah mk
12                  on e1.id_mata_kuliah = mk.id_mata_kuliah
13  group by e1.nim
14  having  sum(e1.na * mk.sks)/ sum(mk.sks) > 3.5
15
16  create view q1e3
17  as
18  select  m.nim, m.nama, e2.ipk
19  from    q1e2 e2 inner join mahasiswa m
20                  on e2.nim = m.nim
```

Source code 7 Rancangan F

```
1   create view q1f1
2   as
3   select  k.nim, k.id_mata_kuliah, n.na
4   from    khs k inner join m_nilai n
```

```
5                       on k.id_nilai = n.id_nilai
6
7   create view q1f2
8   as
9   select      f1.nim, m.nama, f1.na, f1.id_mata_kuliah
10  from        q1f1 f1 inner join mahasiswa m
11                      on f1.nim = m.nim
12
13  create view q1f3
14  as
15  select      f2.nim, f2.nama,
16                      sum(f2.na * mk.sks)/ sum(mk.sks)as ipk
17  from        q1f2 f2 inner join mata_kuliah mk
18                      on f2.id_mata_kuliah = mk.id_mata_kuliah
19  group by f2.nim, f2.nama
20  having      sum(f2.na * mk.sks)/ sum(mk.sks)>3.5
```

Lampiran 3. *Source code* Implementasi SQL Model *Query* 2

Source code 8 Rancangan A

```
1   create view q2a1
2   as
3   select      m.nim, m.nama,
4                       k.id_mata_kuliah, k.id_nilai
5   from        khs k inner join mahasiswa m
6                       on k.nim = m.nim
7   where       k.thn_akademik < (m.angkatan+1)
8
9   create view q2a2
10  as
11  select      a1.nim, a1.nama, a1.id_nilai,
12                      mk.sks
13  from        q2A1 a1 inner join mata_kuliah mk
14                      on a1.id_mata_kuliah = mk.id_mata_kuliah
15
16  create view q2a3
17  as
18  select      a2.nim, a2.nama,
19                      sum(n.na * a2.sks)/ sum(a2.sks) as ipk
20  from        q2a2 a2 inner join m_nilai n
21                      on a2.id_nilai = n.id_nilai
22  group by a2.nim, a2.nama
23  having      sum(n.na * a2.sks)/ sum(a2.sks) <2
```

Source code 9 Rancangan B

```
1   create view q2b1
2   as
3   select      m.nim, m.nama,
4                       k.id_mata_kuliah, k.id_nilai
5   from        khs k inner join mahasiswa m
6                       on k.nim = m.nim
7   where       k.thn_akademik < (m.angkatan+1)
8
9   create view q2b2
10  as
11  select      b1.nim, b1.nama, b1.id_mata_kuliah,
12                      n.na
13  from        q2b1 b1 inner join m_nilai n
14                      on b1.id_nilai = n.id_nilai
15
16  create view q2b3
17  as
18  select      b2.nim, b2.nama,
19                      sum(b2.na * mk.sks)/ sum(mk.sks) as ipk
20  from        q2b2 b2 inner join mata_kuliah mk
21                      on b2.id_mata_kuliah = mk.id_mata_kuliah
22  group by b2.nim, b2.nama
23  having      sum(b2.na * mk.sks)/ sum(mk.sks) <2
24
25  select * from q2b3
```

Source code 10 Rancangan C

```
1   create view q2c1
2   as
3   select      k.nim, k.thn_akademik, k.id_nilai, mk.sks
4   from        khs k inner join mata_kuliah mk
5                       on k.id_mata_kuliah = mk.id_mata_kuliah
```

```
6
7   create view q2c2
8   as
9   select      c1.id_nilai, c1.sks, c1.nim, m.nama
10  from        q2c1 c1 inner join mahasiswa m
11                      on c1.nim = m.nim
12  where       c1.thn_akademik < (m.angkatan+1)
13
14  create view q2c3
15  as
16  select      c2.nim, c2.nama,
17                      sum(n.na * c2.sks)/ sum(c2.sks) as ipk
18  from        q2c2 c2 inner join m_nilai n
19                      on c2.id_nilai = n.id_nilai
20  group by c2.nim, c2.nama
21  having      sum(n.na * c2.sks)/ sum(c2.sks) < 2
```

Source code 11 Rancangan D

```
1   create view q2d1
2   as
3   select      k.nim, k.thn_akademik, k.id_nilai, mk.sks
4   from        khs k inner join mata_kuliah mk
5                       on k.id_mata_kuliah = mk.id_mata_kuliah
6
7   create view q2d2
8   as
9   select      d1.nim, d1.sks, d1.thn_akademik, n.na
10  from        q2d1 d1 inner join m_nilai n
11                      on d1.id_nilai = n.id_nilai
12
13  create view q2d3
14  as
15  select      d2.nim, m.nama,
16                      sum(d2.na * d2.sks)/ sum(d2.sks) as ipk
17  from        q2d2 d2 inner join mahasiswa m
18                      on d2.nim = m.nim
19  where       d2.thn_akademik < (m.angkatan+1)
20  group by d2.nim, m.nama
21  having      sum(d2.na * d2.sks)/ sum(d2.sks) < 2
```

Source code 12 Rancangan E

```
1   create view q2e1
2   as
3   select      k.nim, k.thn_akademik, k.id_mata_kuliah, n.na
4   from        khs k inner join m_nilai n
5                       on k.id_nilai = n.id_nilai
6
7   create view q2e2
8   as
9   select      e1.nim, e1.na, e1.thn_akademik, mk.sks
10  from        q2e1 e1 inner join mata_kuliah mk
11                      on e1.id_mata_kuliah = mk.id_mata_kuliah
12
13  create view q2e3
14  as
15  select      e2.nim, m.nama, sum(e2.na * e2.sks)/ sum(e2.sks) as ipk
16  from        q2e2 e2 inner join mahasiswa m
17                      on e2.nim = m.nim
18  where       e2.thn_akademik < (m.angkatan+1)
19  group by e2.nim, m.nama
20  having      sum(e2.na * e2.sks)/ sum(e2.sks) < 2
21
22
```

Source code 13 Rancangan F

```
1   create view q2f1
2   as
3   select      k.nim, k.thn_akademik, k.id_mata_kuliah, n.na
4   from        khs k inner join m_nilai n
5                       on k.id_nilai = n.id_nilai
6
7   create view q2f2
8   as
9   select      f1.nim, m.nama, f1.na, f1.id_mata_kuliah
10  from        q2f1 f1 inner join mahasiswa m
11                      on f1.nim = m.nim
12  where       f1.thn_akademik < (m.angkatan+1)
13
14  create view q2f3
15  as
16  select      f2.nim, f2.nama,
17                      sum(f2.na * mk.sks)/ sum(mk.sks)as ipk
18  from        q2f2 f2 inner join mata_kuliah mk
19                      on f2.id_mata_kuliah = mk.id_mata_kuliah
20  group by f2.nim, f2.nama
21  having      sum(f2.na * mk.sks)/ sum(mk.sks)<2
```

Lampiran 4. *Source code* implementasi SQL Model *Query* 3

Source code 14 Rancangan A

```
1   create      view q3a1 as
2   select      f.id_fakultas, f.content as fakultas, j.id_jurusan, j.content as jurusan
3   from        m_fakultas f inner join m_jurusan j on f.id_fakultas=j.id_fakultas
4
5   create view q3a2 as
6   select      qa1.fakultas,
7                       qa1.jurusan,
8                       mk.k_mk,
9                       mk.thn_mk,
10                      mk.nama,
11                      mk.sks,
12                      mk.id_jenjang, mk.id_prog_studi
13  from        q3a1 qa1 inner join mata_kuliah mk
14                      on qa1.id_fakultas = mk.id_fakultas and qa1.id_jurusan = mk.id_jurusan
15
16  create view q3a3 as
17  select      ps.id_prog_studi, ps.content as prodi, jj.id_jenjang, jj.content as jenjang, ps.id_jurusan
18  from        m_prog_studi ps inner join m_jenjang jj on ps.id_jenjang=jj.id_jenjang
19
20  create view q3a4 as
21  select      qa2.fakultas,
22                      qa2.jurusan,
23                      qa3.prodi,
24                      qa3.jenjang,
25                      qa2.k_mk,
26                      qa2.thn_mk,
27                      qa2.nama,
28                      qa2.sks
29  from        q3a2 qa2 inner join q3a3  qa3
30                      on qa2.id_prog_studi = qa3.id_prog_studi
31                      and qa2.id_jenjang = qa3.id_jenjang
```

Source code 15 Rancangan B

```
1   create view q3b1 as
2   select      ps.id_prog_studi, ps.content as prodi, jj.id_jenjang, jj.content as jenjang, ps.id_jurusan
3   from        m_prog_studi ps inner join m_jenjang jj on ps.id_jenjang=jj.id_jenjang
4
5   create view q3b2 as
6   select      qb1.prodi,
7                       qb1.jenjang,
8                       mk.k_mk,
9                       mk.thn_mk,
10                      mk.nama,
11                      mk.sks,
12                      mk.id_fakultas, mk.id_jurusan
13  from        q3b1 qb1 inner join mata_kuliah mk
14                      on qb1.id_jenjang = mk.id_jenjang and qb1.id_prog_studi = mk.id_prog_studi
15
16  create      view q3b3 as
17  select      f.id_fakultas, f.content as fakultas, j.id_jurusan, j.content as jurusan
18  from        m_fakultas f inner join m_jurusan j on f.id_fakultas=j.id_fakultas
19
20  create view q3b4 as
21  select      qb3.fakultas,
22                      qb3.jurusan,
23                      qb2.prodi,
24                      qb2.jenjang,
25                      qb2.k_mk,
26                      qb2.thn_mk,
27                      qb2.nama,
```

```
28                          qb2.sks
29      from        q3b2 qb2 inner join q3b3  qb3
30                          on qb2.id_fakultas = qb3.id_fakultas
31                          and qb2.id_jurusan = qb3.id_jurusan
```

Source code 16 Rancangan C

```
1       create      view q3c1 as
2       select      f.id_fakultas, f.content as fakultas, j.id_jurusan, j.content as jurusan
3       from        m_fakultas f inner join m_jurusan j on f.id_fakultas=j.id_fakultas
4
5       create view q3c2 as
6       select      ps.id_prog_studi, ps.content as prodi, jj.id_jenjang, jj.content as jenjang, ps.id_jurusan
7       from        m_prog_studi ps inner join m_jenjang jj on ps.id_jenjang=jj.id_jenjang
8
9       create view q3c3 as
10      select      qc1.id_fakultas, qc1.fakultas,
11                          qc1.id_jurusan, qc1.jurusan,
12                          qc2.id_prog_studi, qc2.prodi,
13                          qc2.id_jenjang, qc2.jenjang
14      from        q3c1 qc1 inner join q3c2 qc2 on qc1.id_jurusan = qc2.id_jurusan
15
16      create view q3c4 as
17      select      qc3.fakultas,
18                          qc3.jurusan,
19                          qc3.prodi,
20                          qc3.jenjang,
21                          mk.k_mk,
22                          mk.thn_mk,
23                          mk.nama,
24                          mk.sks
25      from        q3c3 qc3 inner join mata_kuliah mk
26                          on qc3.id_prog_studi = mk.id_prog_studi
```

Lampiran 5. *Source code* Implementasi *Linked server*

Source code 17 *Linked server*

```
1  EXEC master.dbo.sp_addlinkedserver @server = N'ip_address', @srvproduct=N'SQL Server'
2
3  EXEC master.dbo.sp_addlinkedsrvlogin
4  @rmtsrvname=N'ip_address',@useself=N'False',@locallogin=NULL,@rmtuser=NULL,@rmtpassword=NULL
5
6  EXEC master.dbo.sp_addlinkedsrvlogin
7  @rmtsrvname=N'ip_address',@useself=N'False',@locallogin=N'report',@rmtuser=N'username',@rmtpassword='p
8  assword'
9
10 EXEC master.dbo.sp_serveroption @server=N'ip_address', @optname=N'collation compatible',
11 @optvalue=N'false'
12
13 EXEC master.dbo.sp_serveroption @server=N'ip_address', @optname=N'data access', @optvalue=N'true'
14
15 EXEC master.dbo.sp_serveroption @server=N'ip_address', @optname=N'dist', @optvalue=N'false'
16
17 EXEC master.dbo.sp_serveroption @server=N'ip_address', @optname=N'pub', @optvalue=N'false'
18
19 EXEC master.dbo.sp_serveroption @server=N'ip_address', @optname=N'rpc', @optvalue=N'true'
20
21 EXEC master.dbo.sp_serveroption @server=N'ip_address', @optname=N'rpc out', @optvalue=N'true'
22
23 EXEC master.dbo.sp_serveroption @server=N'ip_address', @optname=N'sub', @optvalue=N'false'
24
25 EXEC master.dbo.sp_serveroption @server=N'ip_address', @optname=N'connect timeout', @optvalue=N'0'
26
27 EXEC master.dbo.sp_serveroption @server=N'ip_address', @optname=N'collation name', @optvalue=null
28
29 EXEC master.dbo.sp_serveroption @server=N'ip_address', @optname=N'lazy schema validation',
30 @optvalue=N'false'
31
32 EXEC master.dbo.sp_serveroption @server=N'ip_address', @optname=N'query timeout', @optvalue=N'0'
33
34 EXEC master.dbo.sp_serveroption @server=N'ip_address', @optname=N'use remote collation', @optvalue=N'true'
35
36 EXEC master.dbo.sp_serveroption @server=N'ip_address', @optname=N'remote proc transaction promotion',
37 @optvalue=N'true'
```

Lampiran 6. *Source code* Menambah *User*

Source code 18 Menambahkan *user* pada basis data

```
1  CREATE LOGIN [username] WITH PASSWORD=N'"=password"', DEFAULT_DATABASE=[database_name],
2  DEFAULT_LANGUAGE=[default_language], CHECK_EXPIRATION=OFF, CHECK_POLICY=OFF
3
4  EXEC sys.sp_addsrvrolemember @loginame = N'username', @rolename = N'user_role'
5
6  ALTER LOGIN [report] DISABLE
```

BIODATA PENULIS

Nama : Mohamad Syauqi Reza

Tempat Tanggal Lahir: Bandung, 3 April 1991

Agama : Islam

Alamat : Borobamban Rt 4 Rw 6 No 63
Asrikaton Pakis Malang 65154

Hp : 085755355383

Email : syauqi.reza@live.com

FB/ YM : syauqi.reza

Pendidikan Formal

	Tahun	Lembaga
1	1996 – 1997	TK Muslimat 02 Singosari
2	1997 – 2003	SDN Tamanharjo 1
3	2003 – 2006	SMP Negeri 3 Singosari
4	2006 – 2009	SMA Negeri 10 Malang
5	2009 – 2013	Program S1 Jurusan Ilmu Komputer Universitas Brawijaya

Pendidikan Non- Formal

	Tahun	Organisasi/ Kepanitiaan	Keterangan
1	2009 – 2010	Paduan Suara Mahasiswa UB	Anggota
2	2010 – 2012	Komunitas Mahasiswa Ilmu Komputer UB	Divisi Project Development
3	2011	Seminar Peka Teknologi 2011 UB	Ketua Pelaksana

Pengalaman Organisasi dan Kepanitiaan

	Tahun	Organisasi/ Kepanitiaan	Keterangan
1	2009 – 2010	Paduan Suara Mahasiswa UB	Anggota
2	2010 – 2012	Komunitas Mahasiswa Ilmu Komputer UB	Divisi Project Development
3	2011	Seminar Peka Teknologi 2011 UB	Ketua Pelaksana