OPTIMASI *QUERY* PADA BASIS DATA TERDISTRIBUSI MENGGUNAKAN *MATERIALIZED VIEW*

SKRIPSI

LABORATORIUM REKAYASA PERANGKAT LUNAK

Diajukan untuk Memenuhi Persyaratan Memperoleh Gelar Sarjana Komputer

Disusun Oleh :

DENI YULIUS PRAWIRA WIJAYA

NIM. 115060800111066

KEMENTERIAN RISET TEKNOLOGI DAN PENDIDIKAN TINGGI

UNIVERSITAS BRAWIJAYA

PROGRAM TEKNOLOGI INFORMASI DAN ILMU KOMPUTER

PROGRAM STUDI INFORMATIKA / ILMU KOMPUTER

MALANG

2015

PERNYATAAN

ORISINALITAS SKRIPSI

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang sepengetahuan 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 dari 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, 17 Juni 2015

Mahasiswa,

Deni Yulius Prawira Wijaya

OPTIMASI *QUERY* PADA BASIS DATA TERDISTRIBUSI MENGGUNAKAN *MATERIALIZED VIEW*

SKRIPSI

LABORATORIUM REKAYASA PERANGKAT LUNAK

Diajukan untuk Memenuhi Persyaratan Memperoleh Gelar Sarjana Komputer

Disusun Oleh :

DENI YULIUS PRAWIRA WIJAYA

NIM. 115060800111066

KEMENTERIAN RISET TEKNOLOGI DAN PENDIDIKAN TINGGI

UNIVERSITAS BRAWIJAYA

PROGRAM TEKNOLOGI INFORMASI DAN ILMU KOMPUTER

PROGRAM STUDI INFORMATIKA / ILMU KOMPUTER

MALANG

2015

LEMBAR PERSETUJUAN

OPTIMASI *QUERY* PADA BASIS DATA TERDISTRIBUSI MENGGUNAKAN *MATERIALIZED VIEW*

SKRIPSI

LABORATORIUM REKAYASA PERANGKAT LUNAK

Disusun Oleh :

DENI YULIUS PRAWIRA WIJAYA

NIM. 115060800111066

Skripsi ini telah disetujui oleh dosen pembimbing pada tanggal 1 Juni 2015 :

Dosen Pembimbing I,	**Dosen Pembimbing II,**
Satrio Agung Wicaksono, S.Kom,.M.Kom	**Issa Arwani, S.Kom,.M.Sc**
NIP. 19860521 2012121001	**NIP. 198309222012121003**

LEMBAR PENGESAHAN

OPTIMASI *QUERY* PADA BASIS DATA TERDISTRIBUSI MENGGUNAKAN *MATERIALIZED VIEW*

SKRIPSI

Laboratorium Rekayasa Perangkat Lunak

Untuk memenuhi sebagai persyaratan mencapai gelar Sarjana Komputer

Disusun oleh :

Deni Yulius Prawira Wijaya

NIM. 115060800111066

Skripsi ini diuji dan dinyatakan lulus pada

Tanggal 17 juni 2015

Penguji I	Penguji II
Fajar Pradana, S.ST, M.Eng	**Denny Sagita R.,S.Kom, M.Kom**
NIK. 871121 16110371	**NIK. 85112406110250**

Penguji III

Diah Priharsari, S.T, M.T

Mengetahui,

Ketua Program Studi Informatika/Ilmu Komputer

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

KATA PENGANTAR

Puji dan syukur di panjatkan kehadirat Allah SWT yang telah melimpahkan berkat, kemudahan dan karunia-Nya serta kelancaran sehingga penulis dapat menyelesaikan skripsi dengan judul "Optimasi *Query* Pada Basis Data Terdistribusi Menggunakan *Materialized View* ".Melalui pengantar ini penulis mengucapkan banyak terima kasih karena dalam penyusunan skripsi ini penulis telah mendapat bantuan dan dorongan baik moril maupun materil dari berbagai pihak. Untuk itu pada kesempatan ini penulis ingin mengucapkan terima kasih kepada :

1. Bapak Satrio Agung Wicaksono, S.Kom,.M.Kom selaku dosen pembimbing I dan bapak Issa Arwani, S.Kom,.M.Sc selaku dosen Pembimbing II.
2. Orang tua Ibu Sri Aningsih beserta kakak Andik Wijaya SH dan Indra Prasetya SE, yang telah memberikan kasih sayang dan motivasi serta sarana dan prasaran kepada penulis.
3. Ibu Dinartuti Hartinah yang telah memberikan motivasi serta sarana dan prasaran kepada penulis.
4. Saudari Safira Felicia yang telah memberikan dorongan semangat, motifasi serta kesabarannya.
5. Seluruh pengurus PPTI dan PIDK UB yang telah membantu proses pengumpulan data skripsi.
6. Segenap bapak dan ibu Dosen beserta karyawan Program Teknologi Informasi dan Ilmu Komputer Universitas Brawijaya.
7. Semua teman – teman Kelas TIF –B dan PTIIK, khususnya Informatika / Ilmu Komputer angkatan 2011 terima kasih atas dukungannya selama ini.

Serta semua pihak yang tidak bisa di sebutkan yang telah membantu penulis dalam penyelesaian laporan hasil diskusi ini. Penulis sangat menyadari bahwa laporan hasil diskusi ini masih banyak kekurangan, oleh karena itu penulis mengharapkan masukan dari semua pihak.

Malang, 1 Juni 2015

Penulis

Deni Yulius Prawira Wijaya

ABSTRAK

Deni Yulius Prawira Wijaya. 2015. Optimasi *Query* Pada Basis Data Terdistribusi Menggunakan *Materialized View*. Program Teknologi Informasi dan Ilmu Komputer, Universitas Brawijaya. Malang. Dosen Pembimbing : Satrio Agung Wicaksono, S.Kom,.M.Kom., dan Issa Arwani, S.Kom,.M.Sc.

Pada saat ini di sektor basis data tengah terjadi tren teknologi dengan desain basis data terdistribusi. Tren tersebut terjadi sebagai bentuk perubahan dalam desain basis data dan kelemahan yang ada dalam memberikan layanan informasi. Kelemahan dari sebuah desain basis data yang perlu dipertimbangkan antara lain kegagalan sistem seperti terjadinya *single failure system*, waktu eksekusi yang dibutuhkan sebuah *query*, biaya eksekusi yang dibutuhkan dan pada basis data terdistribusi terdapat pula satu parameter yang perlu dipertimbangkan seperti biaya transaksi antar *server*. Sehingga dengan permasalahan tersebut dibutuhkan sebuah desain basis data dan optimasi *query* untuk menyelesaikan permasalahan-permasalahan tersebut. *Materialized view* merupakan metode optimasi dengan mewujudkan sebuah *view* dalam sebuah tabel sehingga untuk menamplkan data cukup dilakukan dengan mengakses tabel *materialized view* yang telah dibentuk tanpa perlu melakukan akses pada tabel yang sebenarnya. Dengan memanfaatkan metode ini, dalam pengujian yang dilakukan didapatkan hasil waktu eksekusi 103.7142857 kali lebih cepat dari waktu rata-rata eksekusi *query* konvensional terhadap data dengan jumlah 54546 *record* data atau hanya membutuhkan 20.964187328% waktu dari waktu rata-rata *query* konvensional dan waktu eksekusi berjalan 2963.813 kali lebih cepat atau 0.03374% dari waktu rata eksekusi *query* konvensional yang memproses data dengan jumlah 1793104 *record*. Perbandingan hasil tersebut juga berlaku terhadap biaya eksekusi *query*, I/O dan CPU.

Kata kunci : basis data terdistribusi, waktu eksekusi, biaya eksekusi, *materialized view*

ABSTRACT

Deni Yulius Prawira Wijaya. 2015. ***Optimization Query in Disributed Database Using Materialized View.*** *Program of Information Technology and Computer Science, University of Brawijaya, Malang. Advisor*: Satrio Agung Wicaksono, S.Kom,.M.Kom., dan Issa Arwani, S.Kom,.M.Sc.

At this time sector of the database occurs technological trends with design of distributed database. The trend occurs as a form of change in the design of the database and the weaknesses that exist in providing information services. The weaknesses of a design database to be considered among the failure of the system as a single failure system, the time it takes a query execution, costs required and the distributed database there is also a parameter to be considered as transaction costs between servers. So with these problems required a database design and query optimization to solve the problems. Methods materialized view is an optimization method to realize a view in a tabel *so as to display the data just by accessing the* tabel *materialized view that has been set up without the need to access the actual* tabel. *By utilizing this method, in tests performed showed the execution time 103.7142857 times faster than the average execution time of a conventional query with 54.546 data records or just in need of 20.964187328% of the time the average time query execution time running conventional and 2963,813 times faster or 0.03374% of the average execution time of a query coneventional that process data records with the number 1793104. Comparison of these results also apply to the cost of query execution, I / O and CPU.*

Keywords: *distributed database, time of execution, execution costs, materialized views*

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I

PENDAHULUAN

1.1 Latar Belakang

Pada saat ini teknologi telah banyak berkembang pesat hampir diseluruh aspek baik mulai dari segi *software* maupun *hardware*. Perkembangan tersebut juga terbentuk karena adanya kebutuhan dari tuntutan jaman itu sendiri. Basis data saat ini juga telah banyak mengalami perkembangan. Contohnya seperti sebuah instansi seperti universitas pastilah membutuhkan laporan yang berkaitan dengan data mahasiswanya. Pada umumnya penerapan penyimpanan data yang digunakan diterapkan dengan menggunakan basis data terpusat, sehingga pada pengolahannya pengguna dapat melakukan *query* dengan biaya yang lebih kecil jika dibandingkan dengan menggunakan basis data terdistribusi, tetapi pada penggunaan basis data terpusat terdapat beberapa kelemahan yang dapat berakibat fatal pada proses layanan yang diberikan. Sehingga dengan permasalahan ini membuat implementasi dari basis data terdistribusi menjadi banyak di gemari oleh para pendesain basis data.

Kelemahan fatal pada basis data terpusat sebagai contoh jika satu *site* mengalami kegagalan (*down*) dapat menyebabkan semua pengguna yang akan mengakses *site* tersebut tidak dapat mengakses data hingga *site* tersebut berjalan kembali seperti semula atau kejadian ini dapat disebut sebagai *single point failure system*[ARB2-319]. Hal tersebut menyebabkan pelayanan terganggu karena satu *server* yang digunakan tidak dapat memberikan layanan. Dengan permasalahan tersebut maka membuat penerapan basis data terdistribusi menjadi tren teknologi saat ini.

Dalam penerapan basis data terdistribusi, penggunaan eksekusi perintah menggunakan *distributed query*. *Distributed query* dilakukan dengan memanfaatkan katalog pada basis data di *site* utama terhadap tabel data yang terdapat pada *site* lokal. Setelah itu basis data pada *site* pusat akan mengakses katalog tabel yang tersimpan dan mengakses tabel tersebut pada *site* lokal tempat tabel berada untuk memperoleh data yang dibutuhkan. Proses ini dapat terjadi

dengan memanfaatkan sebuah jaringan untuk dimungkinkannya *site* pusat dapat berkomunikasi dengan *site* lokal. Sehingga dengan proses komunikasi antara *site* utama dengan *site* lokal maka biaya komunikasi juga harus diperhitungkan. Biaya komunikasi yang dimaksud dalam proses akses *site* utama terhadap *site* lokal adalah biaya transfer data baik itu dari pusat ke lokal dan juga biaya transfer data dari lokal ke pusat. Dalam proses *distributed query processing* jelas dibutuhkan biaya transfer data yang besar jika data yang dikirimkan sangat besar. Nilai dari proses biaya transfer data yang dibutuhkan pada basis data terdistribusi juga ditentukan dari jumlah *server* yang digunakan. Biaya tersebut muncul karena dibutuhkan biaya ekstra untuk berkomunikasi antar *server* yang digunakan sebagai media penyimpanan, berbeda dengan prinsip basis data terpusat yang hanya memanfaatkan satu *server* saja. Dengan adanya permasalahan biaya yang dibutuhkan juga dapat mempengaruhi kinerja dan lama proses *query* untuk memberikan hasil. Permasalahan tentang kinerja layanan juga akan terasa semakin menurun jika proses *query* dilakukan terhadap data dengan jumlah data yang banyak, dan tabel yang di akses memiliki banyak relasi antara satu tabel dengan tabel yang lainnya.

Dalam hal pemberian layanan, kecepatan proses data merupakan hal yang penting dalam pengolahan basis data. Performa yang bagus merupakan sebuah bentuk layanan yang harus diutamakan demi lancarnya proses transaksi data dan di sisi lain dengan melakukan optimasi *query* selain dapat menghemat waktu eksekusi juga dapat menghemat sumber daya yang digunakan untuk sebuah proses transaksi data seperti menghemat pengguaan disk I/O, RAM dan CPU [OZSU-215].

Sehingga dari kasus-kasus yang ada, penulis mencoba menerapkan basis data terdistribusi dengan memanfaatkan beberapa media virtual sebagai *server* yang saling terhubung dengan satu *server* sebagai pusat. Kemudian dalam prosesnya penulis melakukan *distributed query processing* dengan memanfaatkan sebuah fitur yaitu *materialized view* untuk meminimalkan proses biaya eksekusi *query* seperti I/O, CPU dan Jaringan sebagai langkah optimasi pada basis data.

Pada penyimpanan data khususnya seperti pada sebuah *data warehouse*, dalam lingkungan tersebut dibutuhkan proses penggalian informasi yang cepat

dan tepat sehingga dibutuhkan *query* dengan teknik tertentu yang mampu untuk menampilkan informasi dengan skala besar dan dapat memberikan respon yang cepat contohnya seperti menggunakanan *materialized view* [KARDE-116]. *Materialized view* merupakan sebuah fitur yang dapat digunakan sebagai kumpulan data yang dapat di akses oleh sebuah aplikasi tanpa langsung melakukan akses pada tabel yang tersimpan dalam sebuah basis data. Dengan penggunaan m*aterialized view* dapat memberikan sebuah perbaikan dalam hal waktu proses *query* terutama pada *agregation query* dengan data dan tabel yang besar. *Materialized view* dapat dibentuk berdasarkan dari informasi yang ingin didapatkan dari beberapa tabel sehingga dalam penggunaannya dapat menghemat biaya *query prosesing*. *Materialized view* direfrensikan dari kumpulan tabel yang terdapat dalam basis data yang terefrensikan dari atribut tabel yang akan digunakan atau juga dapat dibentuk dari sebuah pemrosesan data dari satu atau lebih tabel. Pembentukan *materialized view* dapat dilakukan dengan proses yang terdiri dari *selection, join* dan pengelompokan (*group by)* [JON-1]. Dengan adanya *materialized view* aplikasi yang terhubung cukup melakukan sebuah query pemanggilan data tanpa perlu adanya proses *computing* ulang.

Penulis membahas mengenai **"Optimasi *Query* Pada Basis Data *Terdistribusi* Menggunakan *Materialized View*"** dengan tujuan untuk memberikan sebuah cara untuk meminimalkan biaya pemrosesan data pada *distributed query processing* dengan memanfaatkan *materialized view*. Sehingga dengan metode ini diharapkan dapat meningkatkan layanan transaksi data dengan memberikan efisiensi waktu dan proses pada sebuah sistem basis data. Selain itu dengan menggunakan basis data terdistribusi juga dapat menghindarkan terjadinya *single point failure system* ketika diperlukan transaksi data. Dengan cara ini diharapkan dapat memaksimalkan layanan informasi yang dibutuhkan oleh pengguna.

1.2 Rumusan Masalah

Masalah yang dapat dirumuskan sesuai dengan tema yang diambil adalah :

1) Bagaimana mengimplementasikan *materialized view* pada *distributed database*?

2) Bagaimana perbandingan biaya yang dibutuhkan oleh DBMS dengan menggunakan *query* konvensional dan *materialized view*?

1.3 Tujuan

Adapun tujuan dari penelitian ini adalah :

1) Dapat mengimplementasikan *materialized view* pada *distributed query database.*
2) Dapat menguji penggunaan *query* konvensional dan *materialized view* pada basis data terdistribusi.

1.4 Manfaat

Diharapkan untuk dapat mengoptimalkan pemanggilan data pada sebuah basis data terdistribusi sehingga dalam proses pemanggilan data yang dilakukan dapat meminimalkan biaya yang dibutuhkan dengan tujuan menghemat *resource* dan juga waktu yang dibutuhkan serta meminimalkan kesalahan seperti terjadinya *single failure system.*

1.5 Ruang Lingkup Permasalahan

Sehubungan dengan masalah yang dihadapi, maka ruang lingkup dalam penulisan ini adalah pembuatan *materialized view* yang akan di implementasikan pada basis data terdistribusi. Sedangkan ruang lingkup untuk sistemnya adalah sebagai berikut :

1) Melakukan perbandingan biaya yang dibutuhkan DBMS antara menggunakan *query* konvensional dengan *materialized view* pada prosses pemanggilan data.
2) Pengolahan data dan proses *query* dilakukan pada basis data terdistribusi dengan fragmentasi data secara horizontal.
3) Keluaran yang dihasilkan adalah hasil proses berupa biaya yang dibutuhkan DBMS untuk mengeksekusi perintah yang digunakan.
4) Data yang dibutuhkan merupakan data kuantitatif sehingga penulis menggunakan data *dummy* dengan jumlah yang sama dengan data yang sebenarnya.

5) Rancangan basis data yang digunakan merupakan rancangan yang didesain oleh penulis sendiri yang dibuat semirip mungkin dengan rancangan basis data yang digunakan pada instansi terkait.

1.6 Metodologi

Metodologi pada penelitian ini memiliki dua bagian yaitu :

1) Metodologi pengumpulan data
   a. Studi pustaka
   b. Penggalian basis data
   c. Studi literatur sejenis
   d. Wawancara dengan pihak terkait

2) Metodologi pembangunan sistem

   Pengimplementasian perangkat lunak ini memanfaatkan 2 *platform* OS(Microsoft windows 7 dan linux centos 6.5), *vmware workstation* dan DBMS dengan *platform* IBM DB2. Pembangunan perintah atau *query* yang dibentuk menggunakan *query* konvensional dan menggunakan *materialized view*.

1.7 Sistematika Penulisan

Sistematika penulisan ini dimaksud untuk memberikan gambaran mengenai bab-bab yang akan penulis susun dalam laporan skripsi, yaitui :

BAB 1 PENDAHULUAN

Bab ini merupakan pendahuluan yang mencakup uraian tentang latar belakang, permasalahan, ruang lingkup, tujuan, manfaat, metodologi dan sistematika penulisan.

BAB 2 LANDASAN TEORI

Bab ini berisi teori-teori pendukung mengenai basis data, *database management system*, metode optimasi, dan *materialized view*.

BAB 3 METODOLOGI

Bab ini menguraikan secara rinci metode pengumpulan data serta metode pembangunan sistem yang digunakan dalam menganalisis, merancang dan mengimplementasikan sistem.

BAB 4 PERANCANGAN

Bab ini menjelaskan mengenai desain atau perancangan dari bentuk implementasi penelitia. Bentuk desain atau perancangan yang dibentuk meliputi perancangan basis data terdistribusi dan juga mengenai metode *query* yang akan di implementasikan. Dalam perancangan metode *query* yang akan digunakan diharapkan mampu untuk mendapatkan hasil analisis yang dibutuhkan dalam membandingkan penggunaan *qiery* konvensional dengan penggunaan *query* optimasi memanfaatkan *materialized view.*

BAB 5 IMPLEMENTASI

Bab ini menjelaskan tentang bentuk implementasi dari rancangan yang telah dibangun pada bab 4. Dalam bab ini menjelaskan mengenai implementasi media virtual *server*, basis data terdistribusi hingga model-model *query* untuk saling dibandingkan.

BAB 6 PENGUJIAN DAN ANALISA

Bab ini menjelaskan mengenai pengujian dan analisa hasil dari eksekusi query. Proses analisa yang dilakukan adalah dengan melakukan perbandingan biaya yang dibutuhkan oleh sebuah *query* untuk menghasilkan data yang diinginkan pada basis data terdistribusi yang telah terbentuk.

BAB 7 PENUTUP

Bab ini menjelaskan secara singkat tentang kesimpulan dan saran yang dapat digunakan dalam mendesain pembagian data dan mengoptimasi pemanggilan data pada basis data terdistribusi.

BAB II

LANDASAN TEORI

2.1. Kajian Pustaka

Materialized view merupakan sebuah metode optimasi yang dapat digunakan pada sebuah basis data untuk memberikan efek akselerasi dalam menampilkan data secara efisien. Dalam pembentukan *materialized view* dapat di susun melalui hasil *query* dari banyak tabel dan memiliki fungsi *selection* dan *group by* [SSV-1]. Dalam pemanfaatan *materialized view* terdapat tiga tahap, yaitu pembuatan *materialized view (MV),* penggunaan *materialized view* dan perawatannya. Pada penerapan penggunaan *materialized view* dapat dilakukan dengan beberapa pendekatan seperti yang ditulis dalam penelitian Chanowich (2001), yaitu [TRI-66]:

a. Heuristik atau *rule-based*

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

1) Melakukan operasi *selection* di awal untuk mereduksi jumlah baris.
2) Melakukan operasi *projection* di awal untuk mereduksi jumlah atribut.
3) Mengkonversikan query dengan banyak *join* menjadi query dengan banyak *subquery*.
4) Melakukan operasi *selection* dan *join* yang paling kecil keluarannya sebelum operasi lain.

b. Cost-based

Teknik ini mengoptimasikan cost yang dipergunakan dari beberapa alternatif untuk kemudian dipilih salah satu yang menjadi cost

terendah. Teknik ini mengoptimalkan urutan *join* terbalik yang dimungkinkan pada relasi-relasi r1 → r2 →… rn. Teknik ini dipergunakan untuk mendapatkan pohon *left-deep join* yang akan menghasilkan sebuah relasi sebenarnya pada node sebelah kanan yang bukan hasil dari sebuah *intermediate join*.

2.2 Basis data

Basis data merupakan sekumpulan data yang di susun secara sistematis dan terstruktur dalam sebuah media (pc atau *server*) yang dapat di manipulasi dengan tujuan tertentu sesuai dengan kebutuhan pengguna. Basis data terdiri dari sekumpulan data yang perlu didefinisikan berdasar tipe data, struktur dan juga batasannya. Basis data dapat dijadikan sebuah media penyimpanan yang memiliki banyak kelebihan diantaranya sebagai penentu masukan data untuk menhindari adanya duplikasi data.

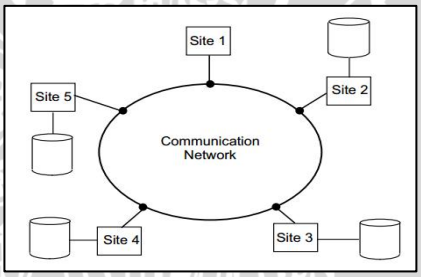
2.2.1 Jenis basis data

Pada penerapannya, basis data dapat di desain menjadi beberapa jenis basis data. Sebuah basis data di bangun berdasarkan kebutuhan dan biaya yang ada karena pembangunan sebuah basis data juga membutuhkan biaya yang variatif. Di sisi lain, jenis basis data yang digunakan juga dapat mempengaruhi performa dan biaya proses yang dibutuhkan sehingga dalam perancangannya juga harus memperhitungkan kelebihan dan kekurangan masing-masing desain berdasar dari kebutuhan yang diinginkan.

2.2.1.1 Basis Data Terdistribusi

Basis data terdistribusi merupakan basis data yang di bangun dan di desain membentuk beberapa *site*. Setiap *site* basis data dapat di bangun pada mesin yang berbeda dan dapat diletakkan di tempat yang berbeda pula. Pada pembangunan sistem basis data ini, setiap *site* pada mesin yang berbeda dapat dihubungkan dengan memanfaatkan sebuah jaringan komunikasi.

Jaringan komunikasi dapat dibentuk dengan menggunakan beberapa jenis topologi berdasar dari fungsi basis data. Penggunaan topologi *mesh* dapat diimplementasikan ketika beberapa *server* sebagai *site* akan dihubungkan dengan 1 buah *server* atau *site* lainnya dengan mekanisme federasi basis data. Penggunaan topologi *ring* dapat digunakan pada basis data terdistribusi dengan maksud terdapat beberapa *server* atau *site* yang akan digunakan pada proses replikasi. Kelebihan dalam penggunaan basis data terdistribusi ini adalah memungkinkan pengguna di 1 *site* dapat mengakses data di*site* yang lain, *reliability* dan *availability*. Dengan kelebihan ini, ketika *site* utama mengakses data dari *site* lain dan terjadi *error* maka sistem dapat mengoperasikan salinan data di situs yang lainnya [LIN-606]. Keungguan inilah yang membuat sistem tidak akan lumpuh secara total karena terjadi kesalahan pada 1 *site* saja. Skema akses basis data dapat ditunjukkan seperti pada gambar 2.1.



Gambar 2.1 Jaringan Basis Data Terdistribusi
(Sumber: OZSU-5)

Terdapat beberapa perbandingan antara penggunaan basis data terpusat dengan basis data terdistribusi. Berikut ini merupakan karakteristik dari basis data terdistribusi [LIN-606]:

1. *Physical distributed*

   Data tidak disimpan pada 1 mesin saja, melainkan data yang digunakan disebar dan disimpan pada mesin yang tersebar pada *site* yang saling terhubung dengan sebuah jaringan dalam basis data terdistribusi.

2. *Logical integrity*

   Dalam penyimpanan data, meskipun data-data disimpan pada mesin dan *site* berbeda yang tersebar dalam basis data terdistribusi tetapi basis data yang terbentuk tetap berada pada satu *logical* yang utuh. Dengan adanya integritas ini memungkinkan pengguna untuk melakukan manajemen basis data dengan menggunakan sistem manajemen basis data terdistribusi (DDBMS).

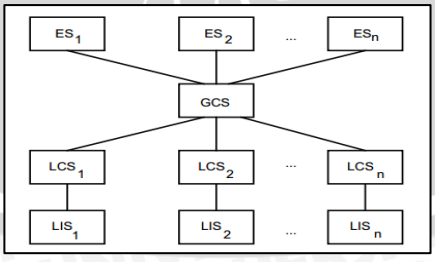
3. *Site autonomy*

   Dalam basis data terdistribusi, data dari setiap *site* dikelola sendiri oleh basis data lokal. Pada setiap *site* memiliki hak otonomi sendiri dalam memberikan hak akses terhadap aplikasi lokal yang terhubung.

Dalam melakukan manipulasi datanya, pengguna membutuhkan perangkat lunak pendukung atau yang bias disebut dengan *database management system* (DBMS). Melalui DBMS, pengguna dapat melakukan pengolahan menggunakan perintah berupa *query*.

2.2.2 Arsitektur Basis Data Terdistribusi

Pada basis data terdistriusi terdapat beberapa macam arsitektur yang dapat digunakan, diantaranya sebagai berikut:



Gambar 2.2 Referensi Arsitektur Basis Data Terdistribusi (Sumber: OZSU-33)

Desain arsitektur pada gambar 2.2 merupakan desain basis data terdistribusi yang di susun secara P2P(peer-to-peer). Pada arsitektur ini

terbagi menjadi bagian yaitu LIS, LCS, GCS dan ES. Berikut ini adalah penjelasan mengenai setiap bagian gambar 2.2:

1) LIS (*Local Internal Schema)* merupakan bagian yang mendefinisikan internal *schema* pada setiap *site* yang terhubung.
2) LCS (*Local Conceptual Schema*) merupakan bagian yang menangani fragmentasi dan replikasi.
3) GCS (*Global Conceptual Schema*) merupakan bagian yang mendeskripsikan struktur data *logic* secara global dari semua *site* yang ada.
4) ES (*External Schema*) merupakan bagian yang mendukung aplikasi dan akses pengguna ke basis data.

2.2.3 Fragmentasi

Fragmentasi merupakan sebuah strategi dan algoritma pemecahan tabel yang dapat dilakukan pada basis data terdistribusi. Terdapat beberapa jenis fragmentasi yang dapat digunakan untuk memecah tabel sesuai dengan kebutuhan dari desain yang digunakan. Strategi yang dapat digunakan diantaranya adalah fragmentasi horizontal, fragmentasi vertikal dan gabungan dari fragmentasi horizontal dan vertikal (*hybrid*).

2.2.3.1 Fragmentasi Horisontal

Fragmentasi horizontal merupakan pemecahan sebuah tabel yang dipecah berdasarkan dari tupel. Dengan strategi ini, sebuah tabel dapat dibagi menjadi beberapa bagian tetapi masih memiliki atribut yang sama. Pada pemecahan datanya dapat dilakukan dengan 2 cara yaitu secara *primary horizontal fragmentation* dan *derived horizontal fragmentation.*

2.2.3.1.1 Primary Horizontal Fragmentation

Pada fragmentasi horisontal jenis ini, *record* data pada tabel utama akan dipecah secara langsung menjadi beberapa bagian sesuai dengan kebutuhan. Pemecahan data jenis ini, pada setiap tabel fragmentasi akan memiliki atribut yang sama dan terdapat

satu atau lebih atribut yang dijadikan sebagai *primary key*. Contoh dari fragmentasi data dengan cara *primary horizontal fragmentation* ditunjukkan pada gambar 2.3.

Tabel 2.1 Tabel PROJ

PNO	PNAME	BUDGET	LOC
P1	Instrumentation	150000	Montreal
P2	Database Develop.	135000	New York
P3	CAD/CAM	250000	New York
P4	Maintenance	310000	Paris

Jika tabel 2.1 dilakukan proses fragmentasi horizontal maka hasilnya dapat ditunjukkan seperti pada gambar 2.3 [OZSU-86]:

$$PROJ_1 = \sigma_{LOC=\text{“Montreal”}}(PROJ)$$

$$PROJ_2 = \sigma_{LOC=\text{“New York”}}(PROJ)$$

$$PROJ_3 = \sigma_{LOC=\text{“Paris”}}(PROJ)$$

$PROJ_1$

PNO	PNAME	BUDGET	LOC
P1	Instrumentation	150000	Montreal

$PROJ_2$

PNO	PNAME	BUDGET	LOC
P2	Database Develop.	135000	New York
P3	CAD/CAM	250000	New York

$PROJ_3$

PNO	PNAME	BUDGET	LOC
P4	Maintenance	310000	Paris

Gambar 2.3 *Primary Horizontal Fragmentation*

(Sumber: OZSU-86)

2.2.3.1.2 Derived Horizontal Fragmentation

Pada fragmentasi horisontal jenis ini, *record* data pada tabel utama akan dipecah menjadi beberapa bagian sesuai dari proses *selction* yang memiliki relasi antara beberapa tabel yang dijadikan sebagai acuan untuk melakukan pemecahan data. Pemecahan data jenis ini, pada setiap tabel fragmentasi akan memiliki atribut yang sama dan terdapat satu atau lebih atribut yang dijadikan sebagai

primary key. Contoh dari fragmentasi data dengan cara *primary horizontal fragmentation* ditunjukkan pada gambar 2.3.

PROJ1: σLOC="New York"∧BUDGET≤200000 (PROJ)
PROJ3: σLOC="New York"∧BUDGET≤200000 (PROJ)
PROJ4: σLOC="New York"∧BUDGET>200000 (PROJ)
PROJ6: σLOC="Paris"∧BUDGET>200000 (PROJ)

Dengan pembagian data pada tabel proj, hasil pemecahan tersebut akan dimanfaatkan sebagai proses pemecahan data pada tabel yang lain secara horisontal. Pemecahan tabel proj yang terbentuk merupakan tabel yang dijadikan acuan pada pembagian data pada tabel yang lain. Contoh dari *derived horizontal fragmentation* ditunjukkan pada Gambar 2.4

$$ASG1 = ASG \ltimes PROJ1$$
$$ASG2 = ASG \ltimes PROJ3$$
$$ASG3 = ASG \ltimes PROJ4$$
$$ASG4 = ASG \ltimes PROJ6$$

ASG_1

ENO	PNO	RESP	DUR
E1	P1	Manager	12
E2	P1	Analyst	24

ASG_2

ENO	PNO	RESP	DUR
E2	P2	Analyst	6
E4	P2	Programmer	18
E5	P2	Manager	24

ASG_3

ENO	PNO	RESP	DUR
E3	P3	Consultant	10
E7	P3	Engineer	36
E8	P3	Manager	40

ASG_4

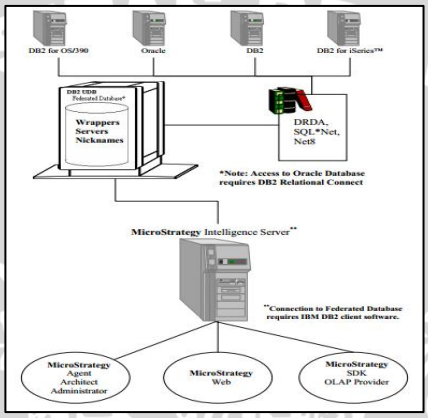
ENO	PNO	RESP	DUR
E3	P4	Engineer	48
E6	P4	Manager	48

Gambar 2.4 *Derived Horizontal Fragmentation*

(Sumber: OZSU-96)

2.2.4 Federasi Basis Data

Pada basis data terdistribusi, desain pada jenis ini dapat dimanfaatkan menjadi federasi basis data. Pada federasi basis data ini, beberapa *site* dapat diletakkan pada media fisik yang berbeda atau dilakukan secara virtual. Pada federasi basis data didesain dengan 1 *site* utama dan beberapa *site* lainnya sebagai tempat data yang telah difragmen. Desain arsitektur dari federasi basis data ditunjukkan seperti pada gambar 2.5.



Gambar 2.5 *Federeated Database Architecture*
(Sumber: MICRO-6)

Berdasar pada gambar 2.5, federasi basis data terdiri dari beberapa *site* yang dapat dibentuk dengan DBMS (*database management system*) yang seragam (homogen) atau dengan DBMS yang berbeda-beda (heterogen). Setelah *site-site* telah terbentuk, *site* utama dihubungkan dengan *site-site* pendukung melalui pembuatan *wrapper, server,* dan *nick name.* Setelah federasi telah terbentuk, *site* utama dapat digunakan sebagai fasilitas pendukung seperti *architect administrator, web* dan juga *OLAP provider.*

2.2.5 Basis Data Relasional

Basis data ini memiliki struktur yang logis terkait cara penyimpanannya. Pada penyimpanan datanya, tabel yang terbentuk dalam basis data jenis ini saling memiliki relasi antar tabelnya. Basis data relasional menggunakan sejumlah tabel 2 dimensi yang setiap tabelnya tersusun dari baris (tupel) dan kolom (atribut). Pada tabel yang akan saling

di relasikan membutuhkan atribut yang harus ditetapkan sebagai kunci tabel [SIL-143]. Kunci tabel dapat dibedakan menjadi 2, yaitu *primary key* dan *foreign key*.

Pada tabel yang memiliki relasi salah satu tabelnya memiliki satu atau lebih atribut yang dijadikan *primary key* sebagai penunjuk dan tabel lainnya memiliki *foreign key* sebagai sasaran pada salah satu atributnya. Sehingga dengan adanya *primary* dan *foreign key* ini dua atau lebih tabel dapat dihubungkan untuk mendapatkan informasi yang terdapat dalam masing-masing tabel. Dalam pemrosesan data, basis data ini juga membutuhkan proses mencari yang lebih lama. Hal ini terjadi karena dalam penggalian informasinya perlu dilakukan penggalian data pada beberapa tabel yang saling memiliki relasi antar satu tabel dengan tabel lainnya.

2.3 Aljabar Relasional

Aljabar relasional merupakan sebuah bahasa yang memberikan perintah terdiri dari sekumpulan operasi untuk dieksekusi dengan masukan satu beberapa relasi dari tabel untuk menghasilkan relasi atau data yang baru sesuai dengan perintah operasi yang digunakan. Terdapat beberapa operasi aljabar relasional yang diantaranya operasi *select* yang dilambangkan dengan sigma (σ), operasi *project* yang dilambangkan dengan pi (π), operasi *union* yang dilambangkan dengan "∪", operasi *set difference* yag dilambangkan dengan "-", operasi *Cartesian product* yang dilambangkan dengan "x", operasi *set intersection* yang dilambangkan dengan "∩", operasi *natural join* yang dilambangkan dengan "⋈", operasi *division* yang dilambangkan dengan tanda bagi (÷), operasi *theta join* yang dilambangkan dengan (≤, <, =, >, ≥).

2.3.1 *Select*

Operasi ini digunakan untuk melakukan seleksi tuple yang memenuhi nilai dari data yang ditentukan pada atribut yang ditunjuk dalam suatu tabel. Penggunaan operasi ini dalam aljabar relasional

ditunjukkan dengan simbol sigma (σ). Contoh aljabar relasional dari operasi *select* seperti pada gambar 2.6 [SIL-89].

$$\sigma_{branch\text{-}name = \text{“Perryridge”}}\ (loan)$$

Gambar 2.6 Operasi *Select*
(Sumber: SIL-89)

Dalam perintah aljabar relasional *select*, perintah tersebut dapat diterjemahkan dilakukan seleksi tuple melalui atribut *branch-name* yang memiliki nilai “Perryridge” pada tabel *loan*, sehingga tupel yang ditampilkan hanya tupel yang memiliki nilai “Perryridge” pada atribut *brach-name*.

2.3.2 *Project*

Operasi ini digunakan untuk menampilkan atribut yang ingin ditampilkan dari eksekusi perintah yang ditujukan pada suatu tabel. Penggunaan operasi ini dalam aljabar relasional ditunjukkan dengan simbol pi (π). Contoh aljabar relasional dari operasi *project* seperti pada gambar 2.7 [SIL-90].

$$\Pi_{loan\text{-}number,\ amount}(loan)$$

Gambar 2.7 Operasi *Project*
(Sumber: SIL-90)

Dalam perintah aljabar relasional *project*, perintah tersebut dapat diterjemahkan dengan menampilkan seluruh data *record* pada tabel *loan* tetapi data yang ditampilkan hanya data yang terdapat pada atribut *loan-number* dan amount saja.

2.3.3 *Union*

Operasi ini digunakan untuk menampilkan data dari beberapa tabel yang tersimpan dalam suatu basis data yang digabungkan menjadi satu. Penggunaan operasi ini dalam aljabar relasional ditunjukkan dengan simbol *union* (∪). Contoh aljabar relasional dari operasi *union* seperti pada gambar 2.8 [SIL-91].

$$\Pi_{customer\text{-}name}\,(borrower) \cup \Pi_{customer\text{-}name}\,(depositor)$$

Gambar 2.8 Operasi *Union*
(Sumber: SIL-91)

Dalam perintah aljabar relasional *union*, perintah tersebut digunakan untuk menampilkan data berupa *customer-name* dengan menggabungkan dua data tabel yaitu tabel *borrower* dan tabel *depositor*.

2.3.4 *Natural Join*

Operasi ini berfungsi untuk menggabungkan operasi *select* dan *Cartesian product* menjadi 1 operasi. Penggunaan operasi ini dalam aljabar relasional ditunjukkan dengan simbol "⋈" [SIL-99]. Contoh dari operasi ini seperti pada gambar 2.9.

$$\Pi_{branch\text{-}name}(\sigma_{customer\text{-}city = \text{“Harrison”}}\,(customer \bowtie account \bowtie depositor))$$

Gambar 2.9 Operasi *Natural Join*
(Sumber: SIL-99)

Dari perintah aljabar relasional *natural join*, perintah tersebut digunakan untuk menampilkan data *branch-name* yang memiliki relasi pada tabel *customer*, *account* dan *depositor* pada atribut *customer-city* dengan nilai "Harrison".

2.4 Operator Aljabar Relasional

Operasi aljabar relasional dikembangkan lebih lanjut dari kumpulan dasar aljabar relasional. Pengembangan pada operasi ini memungkinkan dilakukannya proses aritmatika sebagai bagian dari *projection.* Pada operasi ini memungkinkan juga bagi fungsi agregasi seperti fungsi *SUM* dan *AVERAGE*. Disisi lain juga dapat menggunakan *outer-join operation* yang memungkinkan adanya nilai *null.*

2.4.1 *Generalized Projection*

Pada operator ini memungkinkan melakukan operasi *projection (π).* Operator ini memiliki fungsi yang sama persis seperti *project*

operation. Kesamaan tersebut terletak seperti fungsi *projection* pada sub bab 2.3.2.

2.4.2 Fungsi agregasi

Pada fungsi ini memungkinkan dilakukannya fungsi proses perhitungan untuk memperoleh hasil keluaran seperti yang diinginkan. Fungsi agregasi dalam operasi aljabar relasional di lambangkan dengan menggunakan simbol ℑ. Fungsi agregasi ini meliputi *SUM, AVERAGE, COUNT, MIN, MAX,* dan *DISTINCT* [SIL-105].

Gambar 2.10 Aljabar Fungsi Agregasi
(Sumber: ELMASRI-167)

Pada gambar 2.10 menunjukkan sebuah aljabar untuk fungsi agregasi *count* dan *average*. Penggunaan fungsi agregasi pada aljabar relasional ditunjukkan dengan simbol ℑ. Pada gambar 2.10 ditunjukkan proses *count* terhadap atribut Ssn dan proses *average* pada atribut *Salary* pada tabel *EMPLOYEE*.

2.5 Kalkulus Relasional

Relasional kalkulus merupakan sebuah bahasa spesifik dalam baisis data yang menitik beratkan pada bagaimana untuk memperoleh informasi. Kalkulus relasional dibagi menjadi dua jenis yaitu kalkulus relasional tuple dan kalkulus relasional domain. Dua jenis kalkulus relasional tersebut memiliki kesamaan pada tujuan penggunaannya. Dari kedua jenis kalkulus relasional tersebut memiliki perbedaan pada variabel primitive yang digunakan untuk menspesifikasikan sebuah *query*[OZSU-56].

2.5.1 Kalkulus relasional tupel

Dalam menyusun sebuah *query,* perlu disediakan alur prosedur dalam membentuk hasil yang dicari. Kalkulus relasional tupel marupakan bahasa *query nonprocedural*. Kalkulus relasional ini mendeskripsikan

informasi tanpa memberikan sebuah prosedur yang spesifik untuk mendapatkan sebuah informasi. Sebuah *query* dapat di ekspresikan menjadi kalkulus relasional seperti pada gambra 2.19.

Query 1. List the name and address of all employees who work for the 'Research' department.

Q1: {*t*.Fname, *t*.Lname, *t*.Address | EMPLOYEE(*t*) **AND** (∃*d*)(DEPARTMENT(*d*) **AND** *d*.Dname='Research' **AND** *d*.Dnumber=*t*.Dno)}

Gambar 2.11 Kalkulus Relasional Tupel
(Sumber: SIL-118)

Pada persamaan kalkulus relasional, t merupakan semua tupel yang terdapat pada tabel *EMPLOYEE*. sehingga atribut yang yang akan ditampilkan memiliki kesamaan seperti atribut pada tabel *EMPLOYEE*. Pada contoh gambar 2.11 menunjukkan relasional kalkulus pada sebuah *query* yang dilakukan seleksi tupel tabel *DEPARTMENT (d)* pada atribut Dname dengan nilai '*Research*' dan Dnumber memiliki relasi dengan tabel *EMPLOYEE (t)* pada atribut Dno.

2.5.2 Kalkulus Relasional Domain

Bentuk lain dari kalkulus relasional lainnya disebut dengan kalkulus relasional domain. Kalkulus relasional domain menggunakan variabel domain pada nilai dari beberapa atribut domain, bukan nilai untuk sebuah tupel. Kalkulus relasional domain di tujukan sebagai basis secara teoritis secara luas pada penggunaan bahasa QBE. Bentuk expresi kalkulus relasional domain seperti yang ditunjukkan pada gambar 2.12.

Query 1. Retrieve the name and address of all employees who work for the 'Research' department.

Q1: {*q*, *s*, *v* | (∃*z*) (∃*l*) (∃*m*) (EMPLOYEE(*qrstuvwxyz*) **AND** DEPARTMENT(*lmno*) **AND** *l*='Research' **AND** *m*=*z*)}

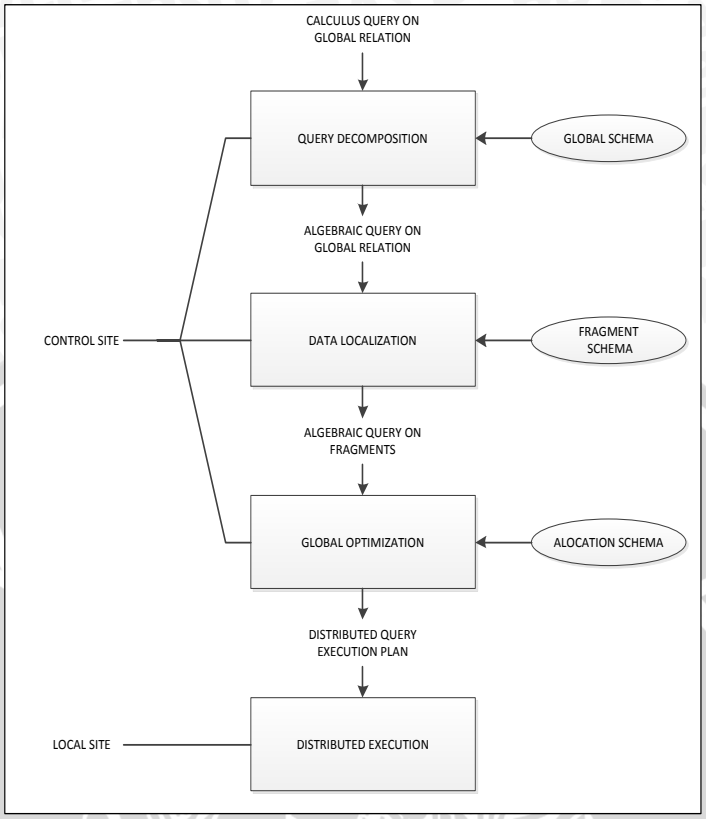
Gambar 2.12 Kalkulus Relasional Domain
(Sumber : SIL-123)

Persamaan kalkulus relasional gambar 2.12 *q, s, v* merupakan variabel domain pada predikat atau atribut yang terdapat pada tabel *EMPLOYEE* dan variabel tersebut merupakan hasil yang dari *query*.

Sedangkan *(∃z)(∃l)(∃m)* merupakan variabel yang menjadi proses *selection* dan relasi antar tabel. Kemudian (*qrstuvwxyz*) merupakan seluruh atribut yang terdapat pada tabel *EMPLOYEE* sebagaimana juga (*lmno*) yang terdapat pada tabel *DEPARTMENT* dan terjadi proses *join* antar kedua tabel dengan relasi yang terdapat pada atribut m pada tabel *DEPARTMENT* dengan z pada tabel *EMPLOYEE*. Sehingga pada gambar 2.12 menunjukkan kalkulus relasional domain dari sebuah *query* yang menampilkan data dari atribut *q, s, v* dengan seleksi pada atribut *l* dengan nilai '*Research*'.

2.6 *Distributed Query Processing*

Pada basis data terdistribusi, terdapat tambahan desain pada sisi *hardware* dan juga *software* pada pengoperasiannya sehingga dibutuhkan biaya tambahan seperti biaya transfer data antar *site* melalui jaringan. Data hasil dari proses *query processing* dapat dihasilkan melalui *site* akhir yang kemudian dikirim ke *site* utama yang mengirim *query* [ARB1-1135]. Proses query yang digunakan merupakan *distributed query processing.* Pada proses tersebut terdapat empat *layer* utama dari *query processing* yang tiga *layer* diantaranya dapat di desain untuk proses optimasi yaitu *query decomposition, data localization,* dan *global query optimization*. Alur layer yang terjadi dapat ditunjukkan seperti pada gambar 2.13.



Gambar 2.13 Skema *Layer Distributed Query Processing* (Sumber: OZSU-216)

Pada lapisan *Query decomposition* dan *data localization* berperan sebagai lapisan yang menyusun ulang *query*. Tiga lapisan yang dapat digunakan dalam proses optimasi terdapat pada proses *query* di *site* pusat yang dijadikan sebagai kendali (*control site*) sedangkan lapian ke empat yaitu *distribution execution* terdapat pada *local site. Distribution execution* berperan menjalankan atau mengeksekusi *query* yang masuk dari lapisan *global optimization* dan memberikan hasil proses dari *query* yang masuk. [OZSU-215]

2.6.1 *Query Decomposition*

Pada lapisan pertama ini, *query* yang berbentuk *calculus* pada *global schema* akan di ubah menjadi *algebraic query*. Proses transformasi dari *calculus query* menjadi *algebraic query* memanfaatkan data relasi yang tersimpan pada *global conceptual schema*. Pada *layer* ini juga

dilakukan proses normalisasi data untuk menghilangkan adanya redudansi data. Proses yang terjadi pada *layer* ini, *calculus query* ditulis ulang dalam bentuk normalisasi kemudian akan dibentuk *normalized query* untuk menganalisis terhadap *query*. Setelah terbentuk *query* yang benar, *query* tersebut selanjutnya ditransformasikan menjadi *algebraic query*.

2.6.2 *Data Localization*

Pada *layer* ini, *query* keluaran dari *layer query decomposition* berbentuk *algebraic query* dilokalisasikan menggunakan data *distribution information* yang tersimpan pada *fragment schema*. Fungsi dari *layer* ini adalah menentukan fragment mana yang digunakan pada sebuah *query* dan mentransformasikan *distributed query* menjadi sebuah *query* pada *fragment*. Proses akhir atau keluaran dari *layer* ini adalah *query algebraic* yang terbentuk pada fragment.

2.6.3 *Global Query Optimization*

Pada *layer* ini, masukan berupa *algebra query* yang terdapat pada skema fragment yang berasal dari proses pada *layer* dua (*data localization*). Tujuan dari *layer* ini adalah mencari strategi untuk memproses sebuah *query* dengan maksud mencari nilai biaya eksekusi paling kecil untuk mengoptimalkan *query processing*. Sehingga pada kasus basis data terdistribusi, strategi yang di pilih mempertimbangkan beberapa aspek seperti biaya proses CPU, ruang disk, I/O disk dan juga biaya transfer data antar *site*.

2.6.4 *Distributed Query Execution*

Layer ini merupaka *layer* terakhir pada *query processing*. Pada layer ini, *query* yang terbentuk diarahkan pada setiap *site* sesuai dengan fragment yang di bentuk dan di proses secara lokal pada setiap *site*. *Query* yang di proses ini dinamakan dengan *local query*.

2.7 *Subset Query*

Subset query merupakan sebuah teknik penyusunan *query* dengan menyisipkan hasil *select* sebagai kondisi syarat untuk pemilihan data. Dengan taknik ini, query yang ditampilkan kurang lebih sama dengan hasil dari *query cross product* yang melibatkan dua tabel. Pada penggunaannya, *subset query* akan memiliki performa yang lebih bagus jika dimanfaatkan untuk memproses data dengan jumlah besar jika dibandingkan dengan pemrosesan data dengan jumlah kecil.

```
SELECT NIM, Nama, Alamat
FROM Mahasiswa
WHERE NIM in ( SELECT NIM FROM Kuliah)
```

Gambar 2.14 *Subset Query*
(Sumber: TRI-67)

Pada gambar 2.14 ditunjukkan sebuah contoh dari *subset query*. *Query* tersebut digunakan untuk menghasilkan sebuah *view* dengan atribut nim nama dan alamat dari tabel mahasiswa. Penggunaan *subset query* diletakkan pada *selection* yang menampilkan seluruh NIM dari tabel kuliah. Pada *query* ini, nim pada *query* utama di lakukan proses *selection* dengan nilai yang muncul dari *subset query*.

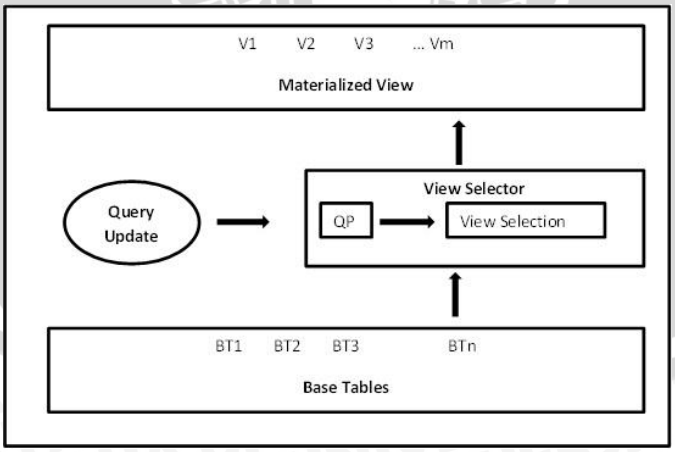
2.8 *Materialized view*

Sebuah hasil *query processing* dari satu atau lebih tabel merupakan sebuah *view*. *View* dapat dijadikan sebagai tabel baru dengan dilakukan pemberian nama atas *view* yang dibentuk [SIL-114]. Pembentukan *materialized view* dalam basis data merupakan sebuah langkah optimasi yang dapat meminimalisir nilai atau biaya transaksi dan proses. Hal tersebut terjadi karena hasil dari sebuah *query processing* diwujudkan menjadi data baru pada sebuah tabel sehingga dalam pemanggilan hasil data tidak dibutuhkan kembali dilakukan *query processing*, sehingga cukup dilakukan pemanggilan data pada *materialized view* yang telah ada. *Materialized view* dibentuk dari *query processing* terhadap beberapa tabel yang memiliki relasi baik itu pada basis data terpusat ataupun basis data terdistribusi sehingga ketika *materialized view*

terbentuk hasil *query* dan relasi antar tabel dapat tetap tersimpan. Dalam pemberdayaan *materialized view* agar dapat digunakan sebagai optimasi, maka terdapat tiga solusi untuk menjaga efisiensi penggunaan basis data [JON-1].

1) ***View design:*** melakukan pembuatan *view* dengan menentukan desain data dari penggabungan tabel.
2) ***View maintenance:*** melakukan perawatan terhadap *view* agar data yang tersimpan didalamnya tetap ter*update* ketika tabel dasar diperbarui.
3) ***View exploitation:*** mendayagunakan *view* yang terbentuk untuk membantu mempercepat *query processing.*

Dalam pembentukan sebuah *materialized view* harus menentukan komposisi atau set dari *view* seperti apa yang akan diwujudkan dalam sebuah *materialized view* untuk mendapatkan performa terbaik dari sebuah *query processing.* Pembentukan *materialized view* sebagai bentuk optimasi pada basis data terdistribusi dilakukan menggunakan *distributed query processing* dengan melihat dari segi tahap pembentukan *view, query complexity, database size, query performance* dan *update performance* seperti yang ditunjukkan pada gambar 2.15 [KARDE-16].



Gambar 2.15 Arsitektur *Materialized View*
(Sumber: KARDE-17)

Pada gambar 2.15 menunjukkan proses dari arsitektur *materialized view.* Proses dibentuk melalui proses membentuk *view selector.* Melalui *view*

selector ini dapat dilakukan proses pembaruan atau *refresh table* untuk membentuk tampilan atau *view* seperti hasil *view* dari penggunaan *query* penyusun.

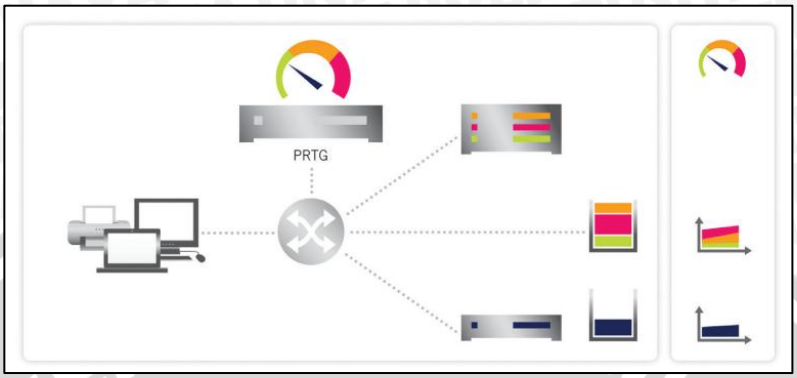
2.8.1 *Materialized Query Table*(MQT)

Materialized query table merupakan sebuah tabel yang didefinisikan sebagai tabel yang terbentuk dari hasil sebuah *query* [IBM]. Prinsip kerja dari *materialized query table* sendiri memiliki fungsi yang sama dengan *materialized view*. Yang membedakan diantara keduanya hanyalah sebutan pada *database management system* yang digunakan. Pada implementasi penulisan ini, penulis memanfaatkan DBMS IBM DB2 sebagai sarana pembentukan basis data. Dalam DBMS tersebut penggunaan *materialized view* disebut sebagai *materialized query table (MQT).* Pembentukan *materialized query table* ini dapat disusun dari hasil proses satu atau lebih tabel yang menghasilkan satu hasil proses. Dalam penggunaannya, *materialized query table* dapat dilakukan perawatan data. Perawatan tersebut dapat dilakukan oleh pengguna secara manual dan juga dapat dilakukan perawatan oleh sistem.

2.9 *Server*

Server merupakan sebuah media fisik yang fungsinya hampir sama seperti *personal computer* (PC) tetapi memiliki keunggulan lebih dari segi performa dan kemampuan dalam memproses data skala besar. Banyak keuntungan lain dalam penggunaan media seperti *server*. *Server* mampu untuk di desain dalam sebuah jaringan sebagai penyedia layanan seperti penyimpanan (*storage*), *web service*, *mail service* dan lain-lain. Pada saat ini terdapat sebuah teknologi yang menjadi tren besar pada segi infrastruktur IT yang telah banyak diterapkan pada sebuah *data center* yaitu virtualisasi *server*. Virtualisasi *server* memungkinkan satu *server* fisik untuk menjalankan beberapa *server* dengan *single platform* maupun *multi platform* dan mampu untuk menangani beban proses komputasi yang besar [PAESLLER-3]. Dengan memanfaatkan virtual *server*, memungkinkan untuk dilakukannya

penggunaan berbagi *resource* pada satu fisik *server* sehingga dalam hal ini dapat dilakukan manajemen *resource* pada setiap virtual *server* yang ada.



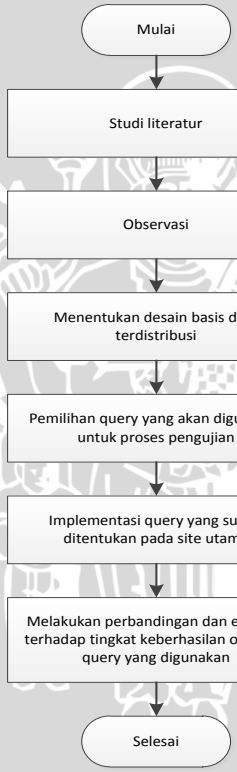
Gambar 2.16 *Server Storage*
(Sumber: PAESSLER-6)

Pada gambar 2.16 menunjukkan terdapat empat aplikasi yang berjalan dan tiga diantaranya ditempatkan pada sebuah virtual *server* sedangkan satu aplikasi lainnya berjalan pada 1 lingkungan *server*. Ketiga aplikasi tersebut dapat berjalan secara bersamaan dengan saling berbagi *resource* yang ada pada satu lingkungan virtual sedangkan satu aplikasi berjalan menggunakan seluruh *resource* secara penuh. *Resource* yang ada pada virtual dapat dibagi termasuk juga media penyimpanan (*disk*) [PAESSLER-6]

BAB III

METODOLOGI

Pada bab metodologi dan perancangan ini akan di berikan penjelasan mengenai metode dan langkah-langkah perancangan implementasi dari pembuatan basis data hingga *query processing* sesuai pada judul karya tulis ini yaitu optimasi query pada basis data terdistribusi menggunakan materialized *view*. Alur penelitian yang dilakukan dan ditunjukkan pada gambar 3.1.



Gambar 3.1 Alur Penelitian

3.1 Studi Literatur

Pada tahap ini, peneliti melakukan studi literatur terhadap aspek-aspek yang akan digunakan sebagai bahan realisasi dari tujuan penulisan. Aspek-aspek yang di pelajari meliputi teori tentang aljabar relasional,

basis data terdistribusi, fragmentasi data dan metode yang digunakan sebagai optimasi *query* yaitu *materialized view* sebagai acuan utama untuk memecahkan masalah yang di angkat sebagai topik masalah penulisan. Seluruh bahan sebagai acuan teori dan implementasi penulis kumpulkan melalui sumber seperti jurnal, penelitian terdahulu, buku-buku dan berselancar di dunia maya (*internet*).

3.2 Observasi

Pada fase ini, peneliti melakukan observasi data dengan cara berdiskusi bersama dengan dosen pembimbing skripsi tentang data yang akan dijadikan sebagai objek penelitian. Peneliti meminta pendapat, saran dan masukan tentang objek yang akan digunakan. Objek penelitian penulis arahkan pada data *dummy* yang memiliki *record data* yang sangat besar, data yang penulis usulkan merupakan data yang tidak absah karena dalam penelitian ini peneliti hanya akan menguji tentang seberapa efisiennya penggunaan *query* optimasi dengan *query* konvesional terhadap pengolahan data yang sangat besar.

Penulis memilih data dengan *record data* yang sangat besar sebagai objek untuk dimanfaatkan dalam proses uji *query* pada basis data terdistribusi. Sehingga proses optimasi *query* hanya ditujukan kepada bagaimana mengolah data secara efisien terhadap sebuah kumpulan data yang memiliki jumlah *record* data sangat besar. Pada perancangan data penelitian yang digunakan dilakukan fragmentasi secara horisontal sehingga data pada tabel utama dibagi menjadi delapan bagian sesuai seperti jumlah *site* yang di bentuk.

3.3 Desain Basis Data Terdistribusi

Pada tahap ini, penulis mendesain sebuah bentuk basis data terdistribusi dengan menggunakan delapan virtual *server*. Desain jaringan basis data terdistribusi di dapat melalui proses diskusi bersama dosen. Virtual *server* yang dibentuk memiliki peranan masing-masing seperti satu virtual *server* yang di desain sebagai *site* utama dan tujuh virtual *server* lainnya bertindak sebagai *site* lokal dimana data sebagai objek penelitian akan ditempatkan. Dari data yang akan digunakan sebagai objek penelitian, data-data yang ada akan dilakukan proses fragmentasi secara horizontal menjadi 8 bagian berdasarkan dari fakultas sehingga

pada akhir pembagian akan terbentuk 8 fragmen data yang akan di simpan pada delapan *site* atau *server*.

3.4 Pemilihan *Query*

Pada penulisan ini, peniliti melakukan penelitian terhadap *reporting query*. *Reporting query* yang dimaksud merupakan sebuah *query* yang digunakan untuk menampilkan sekumpulan data yang dapat dibentukkan kedalam sebuah laporan. Contoh dari penggunaan *query* ini seperti menampilkan data keuangan pembayaran mahasiswa yang dikelompokkan berdasarkan pada setiap fakultas. Pengoptimalan pada *query* yang digunakan untuk menampilkan data keuangan setiap fakultas dimaksudkan agar eksekusi *query* dapat di proses secara cepat dan efisien untuk menghemat waktu dan sumber daya yang ada. Tujuan pengoptimalan *query* ini dikarenakan data yang di proses sangat besar dan juga diperlukan proses *join* dengan tabel yang lain sehingga jika di proses dengan menggunakan *query* konvensional akan membutuhkan waktu yang lama dan juga transfer data yang sangat besar. Pada penulisan ini, peneliti mengangkat dua hasil data yang akan di eksekusi sebagai berikut:

1. Jumlah mahasiswa yang digolongkan berdasarkan angkatan, fakultas, program studi, jenjang dan seleksi.
2. Total pembayaran yang digolongkan berdasarkan fakultas, program studi, jenjang, seleksi, transaksi dan item.

Dari dua hasil data di atas akan di eksekusi dengan beberapa macam *query* untuk melihat perbandingan waktu proses data sebagai langkah optimasi. Pemilihan strategi *query* akan dilakukan dengan menggunakan *query* konvensional yang pada pengolahan datanya akan dilakukan langsung secara terpusat pada *site* utama, data akan di olah secara lokal terlebih dahulu di *site* lokal kemudian hasil *query* dikirimkan ke *site* pusat untuk di olah kembali dan pada strategi terakhir hasil akan dibentukkan pada sebuah *materialized view*. Pencarian metode optimasi paling maksimal dilakukan dengan pendekatan *cost-*

based sehingga dilakukan beberapa percobaan hingga ditemukan nilai proses paling minimal.

3.5 Implementasi Dan Uji Coba Basis Data

Dalam fase ini penulis memanfaatkan *database management system* (DBMS) IBM DB2 dalam merancang basis data terdistribusi. Data yang digunakan merupakan data *dummy* dengan jumlah *record* yang sangat besar. Data tersebut di pecah atau di fragmen menjadi delapan bagian dan disimpan pada setiap *site* lokal yang dihubungkan pada satu *site* utama. Kemudian dilakukan proses katalog *node* dan juga katalog basis data yang digunakan sehingga satu *site* dapat mengakses basis data lain yang terdapat pada *site* yang lainnya. Setelah proses katalog selesai dilanjutkan dengan konfigurasi basis data pada *site* pusat agar dapat melakukan federasi basis data.

3.6 Analisa dan Evaluasi

Pada tahap ini penulis melakukan evaluasi terhadap penelitian yang dilakukan. Proses analisis memanfaatkan beberapa media penunjang seperti data studio IBM DB2 untuk mengetahui waktu eksekusi, *wireshark* untuk mengetahui adanya proses paket data yang dikirimkan. Sehingga dalam penelitian ini akan dilakukan perbandingan pada waktu eksekusi *query*, biaya eksekusi dan paket data yang dikirim. Pembandingan yang dilakukan adalah pada penggunaan *query* konvensional dengan pendekatan beberapa metode dan juga penggunaan *materialized view* (MV) sebagai bentuk proses optimasi *query*. Hasil evaluasi akan diambil sebagai bentuk keberhasilan proses optimasi pada hasil analisa yang memiliki waktu proses paling singkat, dan minimnya biaya eksekusi *query* beserta transfer data yang dilihat dari segi *server*.

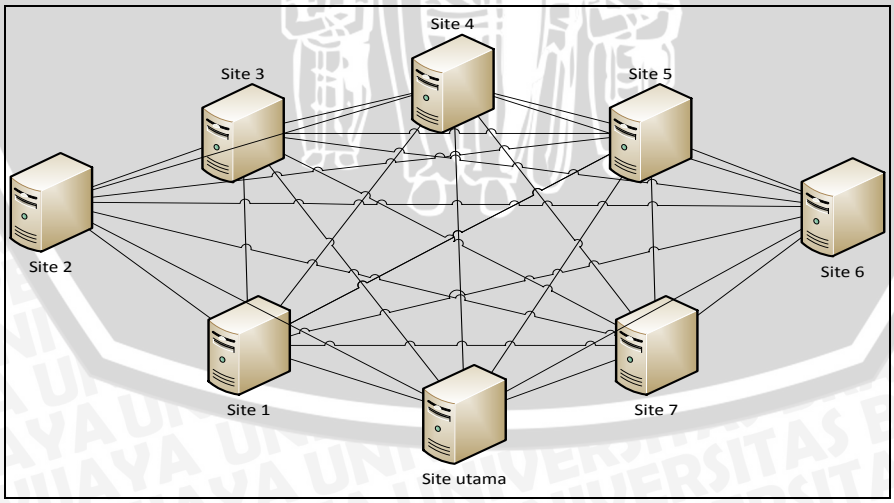
BAB IV

PERANCANGAN

Pada bab ini penulis akan menjelaskan mengenai sketsa atau rancangan dari proses implementasi penelitian. Rancangan ini digunakan sebagai kerangka bentuk proses penyusunan basis data terdistribusi, fragmentasi data dan juga metode *query*. Ketiga rancangan tersebut nantinya akan dibentuk menjadi sebuah implementasi sehingga dapat dilakukan uji coba dan evaluasi terhadap hasil yang terbentuk untuk ditarik menjadi sebuah kesimpulan.

4.1 Jaringan Virtual

Dalam penelitian ini, penulis membentuk virtual dengan menggunakan aplikasi *vmware* sehingga topologi jaringan yang digunakan merupakan topoligi *mesh.* Desain topologi seperti ditunjukkan pada gambar 4.1. Topologi ini memungkinkan setiap virtual yang terbentuk dapat saling berkomunikasi dengan virtual lainnya. Sehingga dalam pembentukan basis data terdistribusi, penulis membentuk delapan virtual yang dapat saling berkomunikasi dalam jaringan yang dibentuk oleh media aplikasi virtualisasi.



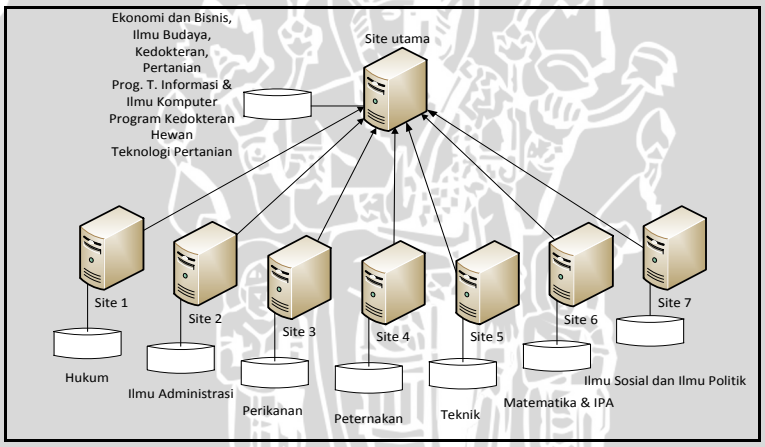
Gambar 4.1 Topologi *Mesh*

Pada gambar 4.1 menunjukan terdapat delapan virtual *server*. Kedelapan virtual *server* tersebut saling dihubungkan sehingga membentuk sebuah jaringan

dengan bentuk topologi *mesh.* Dengan menggunakan topologi ini, memungkinkan kedelapan virtual yang digunakan sebagai *server* basis data dapat saling berkomunikasi antar *site.*

4.2 Basis Data Terdistribusi

Dalam penelitian ini akan dibentuk delapan virtual *server*. Tujuh diantara *server* tersebut akan bertindak sebagai *site* lokal dan satu virtual *server* sebagai *site* utama. Pada langkah pertama dilakukan konfigurasi federasi basis data pada setiap *site* kemudian pertama dilakukan proses katalog *node* dan basis data terhadap *site* lokal hingga proses pembuatan *nick name* untuk setiap tabel yang terdapat pada federasi basis data. Kerangka federasi basis data dari pembentukan delapan virtual ditunjukkan pada gambar 4.2.



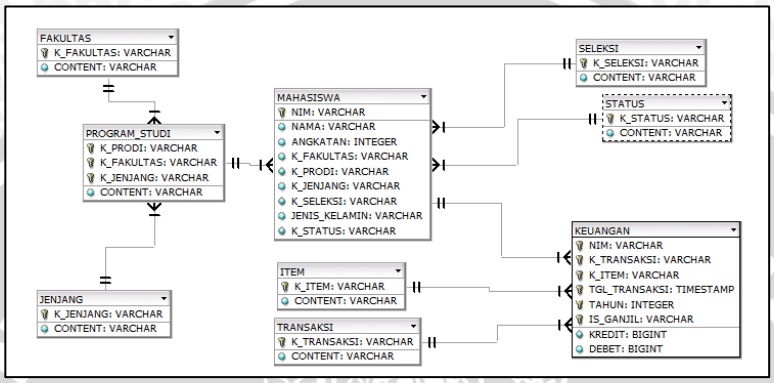
Gambar 4.2 Basis Data Terdistribusi
(Sumber: PPTI-UB)

Pada setiap basis data yang terdapat di masing-masing *site* lokal akan disimpan data yang telah di fragmen menjadi delapan bagian secara horisontal. Fragmentasi dilakukan berdasarkan dari tabel data tentang fakultas, sehingga untuk data lain akan mengikuti pemecahannya sesuai dengan fragmnetasi yang dilakukan terhadap tabel fakultas.

4.3 *Relational Database Management System*

Dalam penelitian ini, penulis merancang basis data terdistribusi dengan skema relasi antar tabel yang sama pada setiap *site* yang digunakan. Dalam

perancangannya terdapat sembilan tabel yang dibentuk dan masing-masing tabel tersebut memiliki relasi antar satu dengan yang lainnya. Perancangan ini bertujuan pula untuk membentuk *primary key* dan *foreign key* yang saling menghubungkan antar tabel. Rancangan *relational database* yang digunakan seperti yang ditunjukkan pada gambar 4.3.



Gambar 4.3 RDBMS
(Sumber: PPTI-UB)

Pada RDBMS gambar 4.3 terdapat beberapa relasi yang saling menghubungkan antara satu tabel dengan tabel yang lainnya. Pada RDBMS tersebut terdapat sembilan tabel dan memiliki delapan relasi. Relasi yang terbentuk antar tabel yang digunakan terdapat satu jenis relasi yaitu *one to many*. Pada relasi *one to many* menandakan bahwa satu *record* data pada tabel tertentu digunakan pada beberapa data di tabel yang saling berelasi.

4.4 Model Akses *Query*

Dalam penelitian ini akan dilakukan uji coba terhadap dua *query* dengan tujuan perolehan akhir yang berbeda-beda. Kedua *query* tersebut akan di bentuk secara konvensional tetapi dengan metode pengolahan data yang berbeda. Uji coba akan dilakukan dengan tiga cara, yaitu :

1. Dilakukan secara lokal

   Dengan metode ini proses dilakukan dengan cara mengolah data terlebih dahulu di *site* lokal kemudian data yang telah di olah sesuai permintaan dikirim menuju *site* pusat. Sehingga pada metode ini data di olah di *local conceptual schema* (LCS) terlebih dahulu kemudian hasil

olah dikirim menuju *site* pusat menjadi *global conceptual schema* (GCS) untuk di olah pada *site* pusat.

2. Memanfaatkan *materialized view*

   Dengan metode ini proses dilakukan dengan cara seperti pada proses 2, yaitu data akan di olah secara lokal terlebih dahulu kemudian data akan dikirim menuju *site* pusat untuk digabungkan. Setelah data yang diinginkan terbentuk, data yang berupa *view* akan diwujudkan menjadi *materialized view* sehingga data akan tetap tersimpan. Dengan tersimpannya data *materialized view* ini jika dilakukan pemanggilan data, *site* utama cukup hanya dengan memanggil *materialized view* yang telah dibentuk.

Kedua cara *query processing* diterapkan untuk mencari hasil paling minimal dalam eksekusi setiap perintahnya. Pembandingan optimasi ini dilakukan dengan menggunakan pendekatan secara *cost-based* sehingga yang menjadi acuan adalah waktu paling minimal yang dibutuhkan untuk eksekusi *query.*

4.4.1 Model Akses *Query 1*

Pada perancangan model akses *query* 1, *query* ini digunakan untuk menampilkan hasil berupa total seluruh mahasiswa yang digolongkan sesuai dengan angkatan, fakultas, program studi, seleksi dan jenjang. Sehingga pada proses ini memerlukan data yang berada pada tabel MAHASISWA, FAKULTAS, PROGRAM_STUDI, JENJANG dan SELEKSI. Tabel MAHASISWA memiliki relasi dengan tabel FAKULTAS yang terletak pada atribut kode fakultas, tabel PROGRAM_STUDI yang memiliki relasi pada atribut kode prodi, kode fakultasqiery, kode jenjang, dan tabel SELEKSI yang memiliki relasi pada atribut kode seleksi.

4.4.1.1 Model Akses *Query* 1 Model 1

Pada model 1, proses *selection* dari tabel MAHASISWA (MHS) ,FAKULTAS (FAK), PROGRAM_STUDI (PRODI), JENJANG dan SELEKSI dilakukan di *site* lokal sebagai LCS. Kemudian setelah data terbentuk dilakukan proses *union* sebagai GCS pada *site* utama untuk membentuk data secara utuh dari gabungan data yang dihasilkan dari LCS masing-masing *site*. Proses untuk model ini seperti ditunjukkan oleh *aljabar relational dan relational calculus* pada gambar 4.4.

a. Relational calculus tuple

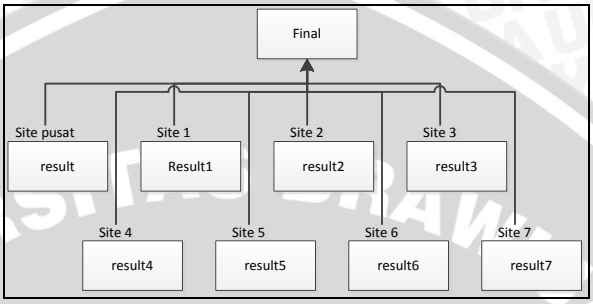
{x1.angkatan, x1.fakultas, x1.program_studi, x1.jenjang, x1.seleksi x1.,jumlah)| (∀m)(MAHASISWA(m)(x1.angkatan. = m.angkatan ∧ ((∀p)(PROGRAM_STUDI(p)(m.k_fakultas, m.k_prodi, m.k_jenjang] = p.k_fakultas, p.k_prodi, p.k_jenjang ∧ p.content. = x1.program_studi ∧ ((∀f)(FAKULTAS(f)(p.k_fakultas = f.k_fakultas ∧ f.content = x1.fakultas)) ∧ (∀j)(jenjang(j)(p.k_jenjang = j.k_jenjang ∧ j.content = x1.jenjang))))) ∧ (∀s)(seleksi(s)(m.k_seleksi = s.k_seleksi ∧ s.content = x1.seleksi ∧ count(m.nim) = x1.jumlah))))}

b. Relational calculus domain

{angkatan : x1 , content: x2 , content : x3 , content : x4 , content : x5 , content: x21count(nim : x6) as jumlah | (∀x8)(∀x9)(∀x10)(∀x11)(∀x13) (∀x19) (∀x20) (MAHASISWA(nim : x6, nama : x7, angkatan : x1 , k_fakultas : x8, k_prodi : x9, k_jenjang : x10, k_seleksi : x11, jenis_kelamin : x12, k_status : x13) ∧ PROGRAM_STUDI(k_prodi : x16 , k_fakultas : x17, k_jenjang : x18, content : x3 ∧ (FAKULTAS(k_fakultas : x14, content : x2) ∧ x14=x16) ∧ (JENJANG(k_jenjang :

x15, content : x4) ∧ x15=x18)) ∧ x9=x16 ∧ x8=x17 ∧x10=x18 ∧ (SELEKSI(k_seleksi : x19, content : x5) ∧ x11=x19) ∧ (STATUS(k_status: x20, content:x21) ∧ x13=x20))}

c. Aljabar relasional jumlah mahasiswa



Gambar 4.4 Aljabar Relasional Model Akses *Query* 1 Model 1

report $\leftarrow \rho_{R(angkatan,\ k\_fakultas,\ k\_\ program\_studi,\ k\_jenjang,\ k\_seleksi,\ k\_status,\ jumlah)}$(angkatan, k_fakultas, k_program_studi, k_jenjang, k_seleksi, k_status $\Im$ count $nim^{(mahasiswa)}$)

result $\leftarrow \pi_{(angkatan,\ fakultas,\ program\_studi,\ jenjang,\ seleksi,\ status,\ jumlah)}$(report $\bowtie_{K\_PRODI,K\_FAKULTAS,K\_JENJANG}$ PROGRAM_STUDI ∧ PROGRAM_STUDI $\bowtie_{K\_JENJANG}$ JENJANG ∧ PROGRAM_STUDI $\bowtie_{K\_FAKULTAS}$ FAKULTAS ∧ report$\bowtie_{K_SELEKSI}$ SELEKSI ∧ report$\bowtie_{K_STATUS}$ STATUS)

report1 $\leftarrow \rho_{R1(angkatan,\ k\_fakultas,\ k\_\ program\_studi,\ k\_jenjang,\ k\_seleksi,\ k\_status,\ jumlah)}$(angkatan, k_fakultas, k_program_studi, k_jenjang, k_seleksi, k_status $\Im$ count $nim^{(mahasiswa1)}$)

result1 $\leftarrow \pi_{(angkatan,\ fakultas,\ program\_studi,\ jenjang,\ seleksi,\ status,\ jumlah)}$(report1 $\bowtie_{K\_PRODI,K\_FAKULTAS,K\_JENJANG}$ PROGRAM_STUDI1 ∧ PROGRAM_STUDI1 $\bowtie_{K\_JENJANG}$ JENJANG1 ∧ PROGRAM_STUDI1 $\bowtie_{K\_FAKULTAS}$ FAKULTAS1 ∧ report1$\bowtie_{K_SELEKSI}$ SELEKSI1 ∧ report1$\bowtie_{K_STATUS}$ STATUS1)

report2 $\leftarrow \rho_{R2(angkatan,\ k\_fakultas,\ k\_\ program\_studi,\ k\_jenjang,\ k\_seleksi,\ k\_status,\ jumlah)}$(angkatan, k_fakultas, k_program_studi, k_jenjang, k_seleksi, k_status $\Im$ count $nim^{(mahasiswa2)}$)

result2 $\leftarrow \pi_{(angkatan,\ fakultas,\ program\_studi,\ jenjang,\ seleksi,\ status,\ jumlah)}$(report2 $\bowtie_{K\_PRODI,K\_FAKULTAS,K\_JENJANG}$ PROGRAM_STUDI2 $\wedge$ PROGRAM_STUDI2 $\bowtie_{K\_JENJANG}$ JENJANG2 $\wedge$ PROGRAM_STUDI2 $\bowtie_{K\_FAKULTAS}$ FAKULTAS2 $\wedge$ report2$\bowtie_{K_SELEKSI}$ SELEKSI2 $\wedge$ report2$\bowtie_{K_STATUS}$ STATUS2)

report3 $\leftarrow \rho_{R3(angkatan,\ k\_fakultas,\ k\_\ program\_studi,\ k\_jenjang,\ k\_seleksi,\ k\_status,\ jumlah)}$(angkatan, k_fakultas, k_program_studi, k_jenjang, k_seleksi, k_status $\Im$ count $nim^{(mahasiswa3)}$)

result3 $\leftarrow \pi_{(angkatan,\ fakultas,\ program\_studi,\ jenjang,\ seleksi,\ status,\ jumlah)}$(report3 $\bowtie_{K\_PRODI,K\_FAKULTAS,K\_JENJANG}$ PROGRAM_STUDI3 $\wedge$ PROGRAM_STUDI3 $\bowtie_{K\_JENJANG}$ JENJANG3 $\wedge$ PROGRAM_STUDI3 $\bowtie_{K\_FAKULTAS}$ FAKULTAS3 $\wedge$ report3$\bowtie_{K_SELEKSI}$ SELEKSI3 $\wedge$ report3$\bowtie_{K_STATUS}$ STATUS3)

report4 $\leftarrow \rho_{R4(angkatan,\ k\_fakultas,\ k\_\ program\_studi,\ k\_jenjang,\ k\_seleksi,\ k\_status,\ jumlah)}$(angkatan, k_fakultas, k_program_studi, k_jenjang, k_seleksi, k_status $\Im$ count $nim^{(mahasiswa4)}$)

result4 $\leftarrow \pi_{(angkatan,\ fakultas,\ program\_studi,\ jenjang,\ seleksi,\ status,\ jumlah)}$(report4 $\bowtie_{K\_PRODI,K\_FAKULTAS,K\_JENJANG}$ PROGRAM_STUDI4 $\wedge$ PROGRAM_STUDI4 $\bowtie_{K\_JENJANG}$ JENJANG4 $\wedge$ PROGRAM_STUDI4 $\bowtie_{K\_FAKULTAS}$ FAKULTAS4 $\wedge$ report4$\bowtie_{K_SELEKSI}$ SELEKSI4 $\wedge$ report4$\bowtie_{K_STATUS}$ STATUS4)

report5 $\leftarrow \rho_{R5(angkatan,\ k\_fakultas,\ k\_\ program\_studi,\ k\_jenjang,\ k\_seleksi,\ k\_status,\ jumlah)}$(angkatan, k_fakultas, k_program_studi, k_jenjang, k_seleksi, k_status $\Im$ count $nim^{(mahasiswa5)}$)

result5 $\leftarrow \pi_{(angkatan,\ fakultas,\ program\_studi,\ jenjang,\ seleksi,\ status,\ jumlah)}$(report5 $\bowtie_{K\_PRODI,K\_FAKULTAS,K\_JENJANG}$ PROGRAM_STUDI5 $\wedge$ PROGRAM_STUDI5 $\bowtie_{K\_JENJANG}$ JENJANG5 $\wedge$ PROGRAM_STUDI5 $\bowtie_{K\_FAKULTAS}$ FAKULTAS5 $\wedge$ report5$\bowtie_{K_SELEKSI}$ SELEKSI5 $\wedge$ report5$\bowtie_{K_STATUS}$ STATUS5)

report6 $\leftarrow \rho_{R6(angkatan,\ k\_fakultas,\ k\_\ program\_studi,\ k\_jenjang,\ k\_seleksi,\ k\_status,\ jumlah)}$(angkatan, k_fakultas, k_program_studi, k_jenjang, k_seleksi, k_status $\Im$ count $nim^{(mahasiswa6)}$)

result6 $\leftarrow \pi_{(angkatan,\ fakultas,\ program\_studi,\ jenjang,\ seleksi,\ status,\ jumlah)}$(report6 $\bowtie_{K\_PRODI,K\_FAKULTAS,K\_JENJANG}$ PROGRAM_STUDI6 $\wedge$ PROGRAM_STUDI6 $\bowtie_{K\_JENJANG}$ JENJANG6 $\wedge$ PROGRAM_STUDI6 $\bowtie_{K\_FAKULTAS}$ FAKULTAS6 $\wedge$ report6$\bowtie_{K_SELEKSI}$ SELEKSI6 $\wedge$ report6$\bowtie_{K_STATUS}$ STATUS6)

report7 $\leftarrow \rho_{R7(angkatan,\ k\_fakultas,\ k\_\ program\_studi,\ k\_jenjang,\ k\_seleksi,\ k\_status,\ jumlah)}$(angkatan, k_fakultas, k_program_studi, k_jenjang, k_seleksi, k_status $\Im$ count $nim^{(mahasiswa7)}$)

result7 $\leftarrow \pi_{(angkatan,\ fakultas,\ program\_studi,\ jenjang,\ seleksi,\ status,\ jumlah)}$(report7 $\bowtie_{K\_PRODI,K\_FAKULTAS,K\_JENJANG}$ PROGRAM_STUDI7 $\wedge$ PROGRAM_STUDI7 $\bowtie_{K\_JENJANG}$ JENJANG7 $\wedge$ PROGRAM_STUDI7 $\bowtie_{K\_FAKULTAS}$ FAKULTAS7 $\wedge$ report7$\bowtie_{K_SELEKSI}$ SELEKSI7 $\wedge$ report7$\bowtie_{K_STATUS}$ STATUS7)

final ← result ∪ result1 ∪ result2 ∪ result3 ∪ result4 ∪ result5 ∪ result6 ∪ result7

Dari desain *aljabar relational* pada gambar 4.4 dan *relational calculus* menunjukkan bahwa *site* pusat meminta data dari masing-masing *site* sesuai permintaan yang diberikan. Data yang dibentuk sebagai LCS merupakan data yang telah di olah, Setelah LCS yang terbentuk terdiri dari proses *selection* yang saling menghubungkan data antar tabel berdasar relasi yang terbentuk. Data yang saling dihubungkan untuk menjadi relasi antar tabel antara lain k_prodi dan k_fakultas pada tabel PROGRAM_STUDI dengan tabel MAHASISWA, k_jenjang pada tabel PROGRAM_STUDI dengan tabel JENJANG, k_fakultas pada tabel PROGRAM_STUDI dengan tabel FAKULTAS dan k_seleksi pada tabel MAHASISWA dengan tabel SELEKSI yang terjadi pada masing-masing *site* seperti yang ditunjukkan pada aljabar relasional *query* 1. Setelah LCS terbentuk kemudian data tersebut digabungkan di *site* utama. Sehingga hasil *union* akan terbentuk pada GCS terdiri dari data yang telah di olah berasal dari *site* lokal. Hasil yang keluaran dari *query* model 1 ini dapat dilihat seperti pada tabel 4.1.

Tabel 4.1 Model *View Query* 1

ANGKATAN	FAKULTAS	PROGRAM _STUDI	JENJANG	SELEKSI	JML
2006	FEB	Ekonomi Pembangunan	S1	SPMK	1
2006	FH	Ilmu Hukum	S1	REG	1
2006	FIA	Ilmu administrasi bisnis	S1	SPMK	1

4.4.1.2 Model akses *query* 1 model 2

Pada *query* 1 model 2 ini, proses penyusunan *materialized view* yang dilakukan sama seperti pada *query* 1 model 1. Proses *selection* dari tabel MAHASISWA (MHS), FAKULTAS (FAK),

PROGRAM_STUDI (PRODI), JENJANG dan SELEKSI dilakukan secara lokal dan data dibentuk pada LCS yang kemudian data di proses menjadi GCS di *site* utama untuk digabungkan bersama data dari setiap *site* lokal. Hasil dari *union* ini kemudian dibentuk dalam sebuah *materialized view*. Proses untuk model ini seperti ditunjukkan oleh *relational calculus* dan aljabar relasional pada gambar 4.5.

a. *Relational calculus tuple*

{x1.angkatan, x1.fakultas, x1.program_studi, x1.jenjang, x1.seleksi, jumlah)| (∀m)(MQT_JUMLAH MAHASISWA(m.angkatan = x1.angkatan ∧ m.fakultas. = x1.fakultas ∧ m.program_studi = x1.program_studi ∧ m.jenjang = x1.jenjang ∧ m.seleks = x1.seleksi ∧ m.jumlah = x1.jumlah))}

b. *Relational calculus domain*

{ <angkatan, fakultas, program_studi, jenjang, seleksi, jumlah> | MQT_JUMLAH_MAHASISWA(angkatan, fakultas, program_studi, jenjang, seleksi, jumlah)}

c. *Aljabar relational*

SITE PUSAT

$\pi_{angkatan,fakultas,program\_studi,jenjang,seleksi,jumlah}(MQT\_JUMLAH\_MAHASISWA)$

Gambar 4.5 Aljabar Relasional Model Akses *Query* 1 Model 2

Dari desain *aljabar relational* pada gambar 4.5 dan *relational calculus* menunjukkan bahwa *site* pusat hanya melakukan pemanggilan data pada sebuah *materialized view*. *Materialized view* di bentuk dengan susunan *query* yang sama seperti *aljabar relational* pada model 1. Pemanggilan tersebut hanya dilakukan pada *site* pusat tanpa melibatkan *site* yang lain. Desain keluaran dari *query* 1 model 2 sama seperti pada *query* 1

model 1, sehingga format tabel dapat dilihat juga seperti pada tabel 4.1.

4.4.2 Model akses *query* 2

Pada perancangan model akses *query* 2, *query* ini digunakan untuk menampilkan hasil berupa total jumlah keuangan mahasiswa di setiap fakultas yang ada. Sehingga pada proses ini memerlukan data yang berada pada tabel MAHASISWA, KEUANGAN ,FAKULTAS, PROGRAM_STUDI dan JENJANG, SELEKSI, ITEM, dan TRANSAKSI. Tabel MAHASISWA memiliki relasi dengan tabel KEUANGAN pada atribut NIM. Pada tabel PROGRAM_STUDI memiliki relasi pada atribut kode fakultas dan kode prodi, dan kode jenjang dengan tabel MAHASISWA. pada tabel JENJANG memiliki relasi dengan tabel PROGRAM_STUDI pada atribut kode jenjang. Tabel FAKULTAS memiliki relasi dengan tabel PROGRAM _STUDI pada atribut kode fakultas. Tabel item memiliki relasi pada atribut kode ITEM dengan tabel KEUANGAN dan tabel TRANSAKSI yang memiliki relasi pada atribut kode transaksi dengan tabel KEUANGAN

4.4.2.1 Model Akses *Query* 2 model 1

Pada *query* 2 model 1, proses *selection* dari tabel MAHASISWA (MHS), KEUANGAN (KEU) dan FAKULTAS (FAK) dilakukan di *site* lokal sebagai LCS. Kemudian setelah data terbentuk data dikirim ke *site* pusat untuk dilakukan proses *union* sebagai GCS pada *site* utama untuk membentuk data secara utuh dari gabungan data yang dihasilkan dari LCS masing-masing *site*. Proses untuk model ini seperti ditunjukkan oleh *aljabar relational,relational calculus,* dan *relational algebra* pada gambar 4.6.

a. *Relational calculus tuple*

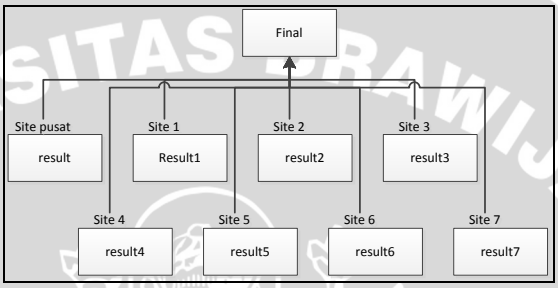
{x1(fakultas, program_studi, jenjang, seleksi, item, jumlah_debet, jumlah_kredit) | (∀k)(KEUANGAN(sum(k.debet)] = x1.jumlah_debet ∧ sum(k.kredit) = x1.jumlah_kredit ∧ (∀m)(MAHASISWA(m)(x1.angkatan. = m.angkatan ∧ ((∀p)(PROGRAM_STUDI(p)(m.k_fakultas, m.k_prodi, m.k_jenjang] = p.k_fakultas, p.k_prodi, p.k_jenjang ∧ p.content. = x1.program_studi ∧ ((∀f)(FAKULTAS(f)(p.k_fakultas = f.k_fakultas ∧ f.content = x1.fakultas)) ∧ (∀j)(jenjang(j)(p.k_jenjang = j.k_jenjang ∧ j.content = x1.jenjang)))) ∧ (∀s)(seleksi(s)(m.k_seleksi = s.k_seleksi ∧ s.content = x1.seleksi))))) ∧ (∀i)(item(i.k_item = k.k_item ∧ i.content=x1.item.)))))}

b. *Relational calculus domain*

{ fakultas : x, program_stuidi : x1, jenjang : x2, seleksi : x3 , transaksi : x4,item : x5, tgl_transaksi : x6 jumlah_debet : x7, jumla_kredit : x8 | (∀x9,x12,x13)(KEUANGAN(nim : x9, tahun : x10, is_ganjil : x11, k_transaksi : x12, k_item : x13, tgl_transaksi : x14, debet : x15, kredit : x16) ∧ (∀x19,x20,x21,x22)(MAHASISWA(nim : x17, nama : x18, angkatan : x19, k_fakultas : x19, k_prodi : x20, k_jenjang : x21, k_seleksi : x22, jenis_kelamin : x23, k_status : x24) ∧ x9=x17 ∧ (∀x25,x26,x27)(PROGRAM_STUDI(k_prodi : x25, k_fakultas : x26, k_jenjang : x27, content : x28) ∧ x19=x26 ∧ x20=x25 ∧ x21=x27 ∧ x28=x1 ∧ (∀x29)(FAKULTAS(k_fakultas : x29, content : x30) ∧ x26=x29 ∧ x30 = x) ∧ (∀x31)(jenjang(k_jenjang : x31, content : x32) ∧ x27=x31 ∧ 32 = x2)) ∧

(∀x33)(SELEKSI(k_seleksi : x33, content : x34) ∧ x22=x33 ∧ x34 = x3)) ∧ (∀x35)(ITEM(k_item : x35, content : x36) ∧ x35=x13 ∧ x36 = x5) ∧ (∀x37)(TRANSAKSI(k_transaksi : x38, content : x39) ∧ x38=x14 ∧ x38 = x4) ∧ sum(x15) = x7 ∧ sum(x16) = x8))}

c. Aljabar relasional keuangan



Gambar 4.6 Aljabar Relasional Model Akses *Query* 2 Model 1

report $\leftarrow \rho_{R(k\_fakultas,\ k\_\ program\_studi,\ k\_jenjang,\ k\_seleksi,\ k\_item,total\_\ debet,\ total\_kredit)}$(k_fakultas, k_program_studi, k_jenjang, k_seleksi, k_item $\Im$ sum debet, sum kredit$^{(KEUANGAN \bowtie_{nim} MAHASISWA)}$)

result $\leftarrow \pi_{(fakultas,\ program\_studi,\ jenjang,\ seleksi,\ item,\ debet,\ kredit)}$(report $\bowtie_{K\_PRODI,K\_FAKULTAS,K\_JENJANG}$ PROGRAM_STUDI ∧ PROGRAM_STUDI $\bowtie_{K\_JENJANG}$ JENJANG ∧ PROGRAM_STUDI $\bowtie_{K\_FAKULTAS}$ FAKULTAS ∧ report$\bowtie_{K_SELEKSI}$ SELEKSI ∧ report$\bowtie_{K_ITEM}$ ITEM)

report1 $\leftarrow \rho_{R1(k\_fakultas,\ k\_\ program\_studi,\ k\_jenjang,\ k\_seleksi,\ k\_item,total\_\ debet,\ total\_kredit)}$(k_fakultas, k_program_studi, k_jenjang, k_seleksi, k_item $\Im$ sum debet, sum kredit$^{(KEUANGAN1 \bowtie_{nim} MAHASISWA1)}$)

result1 $\leftarrow \pi_{(fakultas,\ program\_studi,\ jenjang,\ seleksi,\ item,\ debet,\ kredit)}$(report1 $\bowtie_{K\_PRODI,K\_FAKULTAS,K\_JENJANG}$ PROGRAM_STUDI1 ∧ PROGRAM_STUDI1 $\bowtie_{K\_JENJANG}$ JENJANG1 ∧

PROGRAM_STUDI1 $\bowtie_{K\_FAKULTAS}$ FAKULTAS1 $\wedge$ report1$\bowtie_{K_SELEKSI}$ SELEKSI1 $\wedge$ report1$\bowtie_{K_ITEM}$ ITEM1)

report2 $\leftarrow \rho_{R2(k\_fakultas,\ k\_\ program\_studi,\ k\_jenjang,\ k\_seleksi,\ k\_item,total\_\ debet,\ total\_kredit)}$(k_fakultas, k_program_studi, k_jenjang, k_seleksi, k_item $\Im$ sum debet, sum kredit$^{(KEUANGAN2\ \bowtie_{nim}\ MAHASISWA2)}$)

result2 $\leftarrow \pi_{(fakultas,\ program\_studi,\ jenjang,\ seleksi,\ item,\ debet,\ kredit)}$(report2 $\bowtie_{K\_PRODI,K\_FAKULTAS,K\_JENJANG}$ PROGRAM_STUDI2 $\wedge$ PROGRAM_STUDI2 $\bowtie_{K\_JENJANG}$ JENJANG2 $\wedge$ PROGRAM_STUDI2 $\bowtie_{K\_FAKULTAS}$ FAKULTAS2 $\wedge$ report2$\bowtie_{K_SELEKSI}$ SELEKSI2 $\wedge$ report2$\bowtie_{K_ITEM}$ ITEM2)

report3 $\leftarrow \rho_{R3(k\_fakultas,\ k\_\ program\_studi,\ k\_jenjang,\ k\_seleksi,\ k\_item,total\_\ debet,\ total\_kredit)}$(k_fakultas, k_program_studi, k_jenjang, k_seleksi, k_item $\Im$ sum debet, sum kredit$^{(KEUANGAN3\ \bowtie_{nim}\ MAHASISWA3)}$)

result3 $\leftarrow \pi_{(fakultas,\ program\_studi,\ jenjang,\ seleksi,\ item,\ debet,\ kredit)}$(report3 $\bowtie_{K\_PRODI,K\_FAKULTAS,K\_JENJANG}$ PROGRAM_STUDI3 $\wedge$ PROGRAM_STUDI3 $\bowtie_{K\_JENJANG}$ JENJANG3 $\wedge$ PROGRAM_STUDI3 $\bowtie_{K\_FAKULTAS}$ FAKULTAS3 $\wedge$ report3$\bowtie_{K_SELEKSI}$ SELEKSI3 $\wedge$ report3$\bowtie_{K_ITEM}$ ITEM3)

report4 $\leftarrow \rho_{R4(k\_fakultas,\ k\_\ program\_studi,\ k\_jenjang,\ k\_seleksi,\ k\_item,total\_\ debet,\ total\_kredit)}$(k_fakultas, k_program_studi, k_jenjang, k_seleksi, k_item $\Im$ sum debet, sum kredit$^{(KEUANGAN4\ \bowtie_{nim}\ MAHASISWA4)}$)

result4 $\leftarrow \pi_{(fakultas,\ program\_studi,\ jenjang,\ seleksi,\ item,\ debet,\ kredit)}$(report4 $\bowtie_{K\_PRODI,K\_FAKULTAS,K\_JENJANG}$ PROGRAM_STUDI4 $\wedge$ PROGRAM_STUDI4 $\bowtie_{K\_JENJANG}$ JENJANG4 $\wedge$

PROGRAM_STUDI4 $\bowtie_{K\_FAKULTAS}$ FAKULTAS4 $\wedge$ report4$\bowtie_{K_SELEKSI}$ SELEKSI4 $\wedge$ report4$\bowtie_{K_ITEM}$ ITEM4)

report5 $\leftarrow \rho_{R5(k\_fakultas,\ k\_\ program\_studi,\ k\_jenjang,\ k\_seleksi,\ k\_item,total\_\ debet,\ total\_kredit)}$(k_fakultas, k_program_studi, k_jenjang, k_seleksi, k_item $\Im$ sum debet, sum kredit$^{(KEUANGAN5\ \bowtie_{nim}\ MAHASISWA5)}$)

result5 $\leftarrow \pi_{(fakultas,\ program\_studi,\ jenjang,\ seleksi,\ item,\ debet,\ kredit)}$(report5 $\bowtie_{K\_PRODI,K\_FAKULTAS,K\_JENJANG}$ PROGRAM_STUDI5 $\wedge$ PROGRAM_STUDI5 $\bowtie_{K\_JENJANG}$ JENJANG5 $\wedge$ PROGRAM_STUDI5 $\bowtie_{K\_FAKULTAS}$ FAKULTAS5 $\wedge$ report5$\bowtie_{K_SELEKSI}$ SELEKSI5 $\wedge$ report5$\bowtie_{K_ITEM}$ ITEM5)

report6 $\leftarrow \rho_{R6(k\_fakultas,\ k\_\ program\_studi,\ k\_jenjang,\ k\_seleksi,\ k\_item,total\_\ debet,\ total\_kredit)}$(k_fakultas, k_program_studi, k_jenjang, k_seleksi, k_item $\Im$ sum debet, sum kredit$^{(KEUANGAN6\ \bowtie_{nim}\ MAHASISWA6)}$)

result6 $\leftarrow \pi_{(fakultas,\ program\_studi,\ jenjang,\ seleksi,\ item,\ debet,\ kredit)}$(report6 $\bowtie_{K\_PRODI,K\_FAKULTAS,K\_JENJANG}$ PROGRAM_STUDI6 $\wedge$ PROGRAM_STUDI6 $\bowtie_{K\_JENJANG}$ JENJANG6 $\wedge$ PROGRAM_STUDI6 $\bowtie_{K\_FAKULTAS}$ FAKULTAS6 $\wedge$ report6$\bowtie_{K_SELEKSI}$ SELEKSI6 $\wedge$ report6$\bowtie_{K_ITEM}$ ITEM6)

report7 $\leftarrow \rho_{R7(k\_fakultas,\ k\_\ program\_studi,\ k\_jenjang,\ k\_seleksi,\ k\_item,total\_\ debet,\ total\_kredit)}$(k_fakultas, k_program_studi, k_jenjang, k_seleksi, k_item $\Im$ sum debet, sum kredit$^{(KEUANGAN7\ \bowtie_{nim}\ MAHASISWA7)}$)

result7 $\leftarrow \pi_{(fakultas,\ program\_studi,\ jenjang,\ seleksi,\ item,\ debet,\ kredit)}$(report7 $\bowtie_{K\_PRODI,K\_FAKULTAS,K\_JENJANG}$ PROGRAM_STUDI7 $\wedge$ PROGRAM_STUDI7 $\bowtie_{K\_JENJANG}$ JENJANG7 $\wedge$ PROGRAM_STUDI7 $\bowtie_{K\_FAKULTAS}$ FAKULTAS7 $\wedge$ report7$\bowtie_{K_SELEKSI}$ SELEKSI7 $\wedge$ report7$\bowtie_{K_ITEM}$ ITEM7)

final ← result ∪ result1 ∪ result2 ∪ result3 ∪ result4 ∪ result5 ∪ result6 ∪ result7

Dari desain *relational calculus, dan aljabar relational* pada gambar 4.6 menunjukkan bahwa *site* pusat meminta data dari masing-masing *site* sesuai permintaan yang diberikan. Data yang dibentuk sebagai LCS merupakan data yang telah di olah, Setelah LCS yang terbentuk terdiri dari proses *selection* yang saling menghubungkan data antar tabel berdasar relasi yang terbentuk. Pada model *query* ini, penulis memanfaatkan *sub query*. Data yang saling dihubungkan untuk menjadi relasi antar tabel pada *sub query* antara lain nim pada tabel MAHASISWA dengan tabel KEUANGAN. Sedangkan data yang saling dihubungkan untuk menjadi relasi antar tabel pada *query* utama antara lain k_prodi dan k_fakultas pada tabel PROGRAM_STUDI dengan tabel MAHASISWA, k_jenjang pada tabel PROGRAM_STUDI dengan tabel JENJANG, k_fakultas pada tabel PROGRAM_STUDI dengan tabel FAKULTAS, k_seleksi pada tabel SELEKSI dengan tabel MAHASISWA, k_item pada tabel ITEM dengan tabel KEUANGAN dan k_transaksi pada tabel TRANSAKSI dengan tabel KEUANGAN. Proses *join* yang terjadi dilakukan pada setiap *site*. Setelah LCS terbentuk kemudian data tersebut digabungkan di *site* utama. Sehingga hasil *union* akan terbentuk pada GCS terdiri dari data yang telah di olah berasal dari *site* lokal. Hasil yang keluaran dari *query* model 1 ini dapat dilihat seperti pada tabel 4.2.

Tabel 4.2 Model *View Query* 2

FAKULTAS	ROGRAM_STUDI	JENJANG	SELEKSI	ITEM	TOTAL DEBET	TOTAL KREDIT
FEB	Ekonomi Pembangunan	S1	SPMK	DBP	78000000	78000000
FEB	Ekonomi Pembangunan	S1	SPMK	SPP	200000000	200000000
FEB	Ekonomi Pembangunan	S1	SPMK	JAKET	60000000	60000000

4.4.2.2 Model Akses *Query* 2 model 2

Pada *query* 2 model 2 ini, proses yang dilakukan sama seperti pada *query* 2 model 2. Proses *selection* dari tabel MAHASISWA (MHS), KEUANGAN (KEU), FAKULTAS (FAK), PROGRAM_STUDI (PRODI), ITEM, dan TRANSAKSI dilakukan secara lokal dan data dibentuk pada LCS yang kemudian data di proses menjadi GCS di *site* utama untuk digabungkan bersama data dari setiap *site* lokal. Hasil dari *union* berbentuk GCS ini kemudian dibentuk dalam sebuah *materialized view.* Setelah *materialized view* terbentuk maka pemanggilan data cukup dilakukan pada *site* utama. Proses pemanggilan data untuk model ini seperti ditunjukkan oleh *relational calculus,* dan aljabar relasional pada gambar 4.7.

a. Relational calculus tuple

{x1.fakultas, x1.program_studi, x1.jenjang, x1.seleksi, x1.item, x1.jumlah_debet, x1.jumlah_kredit)|
(∀m)(MQT_JUMLAH_KEUANGAN(m.fakultas = x1.fakultas ∧ m.program_studi = x1.program_studi ∧ m.jenjang = x1.jenjang ∧ m.seleksi = x1.seleksi ∧ m.jumlah = x1.jumlah ∧ m.item = x1.item ∧ m.jumlah_denet = x1.jumlah_debet ∧ m.jumlah_kredit = x1.jumlah_kredit))}

b. Relational calculus domain

{ <fakultas, program_studi, jenjang, seleksi, item, jumlah_debet, jumlah_kredit> | MQT_JUMLAH_KEUANGAN(fakultas, program_studi, jenjang, seleksi, item, jumlah_debet, jumlah_kredit)}

c. Aljabar relational

SITE PUSAT

$\pi_{\text{fakultas,program\_studi,jenjang,seleksi,item,total\_debet,total\_kredit}}(\text{MQT\_JUMLAH\_KEUANGAN})$

Gambar 4.7 Aljabar Relasional Model Akses Query 2 Model 2

Dari desain *relational calculus* dan *aljabar relational* pada gambar 4.7 menunjukkan bahwa *site* pusat hanya melakukan pemanggilan data pada sebuah *materialized view*. *Materialized view* di bentuk dengan susunan *query* yang sama seperti *relational calculus* pada model 1. Pemanggilan tersebut hanya dilakukan pada *site* pusat tanpa melibatkan *site* yang lain. Desain keluaran dari *query* 2 model 2 sama seperti pada *query* 2 model 1, sehingga format tabel dapat dilihat juga seperti pada tabel 4.2.

BAB V

IMPLEMENTASI

Implementasi dari penelitian yang penulis lakukan akan dijelaskan pada bab ini. Proses implementasi ini dibutuhkan media penunjang seperti perangkat keras, perangkat lunak hingga implementasi *query* agar dapat dilakukan pengujian terhadap objek penelitian.

5.1 Lingkungan Perangkat Keras

Perangkat keras yang digunakan dalam penelitian ini adalah :

1. Komputer *Desktop core* i3 3,2 GHz.
2. Memori DDR3 16GB.
3. *HDD* eksternal 1 TB.

5.2 Lingkungan Perangkat Lunak

Perangkat lunak yang diguakan untuk menunjang penelitian ini adalah :

1. Sistem operasi *windows 7 ultimate* 64 bit.
2. Sistem operasi *centos 6.5* 64 bit.
3. *VMware Workstation* versi 10.
4. *Wireshark 1.12.4* 64 bit.
5. IBM DB2 *expc* 64 bit.
6. IBM Data Studio.
7. *Microsoft excel* 2010.
8. PuTTY.
9. WinSCP.

5.3 Implementasi virtual *server (site)*

Dalam penelitian ini, penulis membentuk virtual *server* dengan memanfaatkan aplikasi VMware 10. Berikut ini adalah spesifikasi yang ditanamkan pada setiap virtual *server*:

1. Sistem operasi centos 64 bit.
2. *Single core.*
3. Memori (RAM) 2GB.
4. *Disk* 30 GB.
5. *Network connection bridge.*

Virtual *server* dibentuk sebanyak delapan virtual dan dijalankan menggunakan arsitektur jaringan *network address translation* dengan tujuan agar setiap virtual juga dapat terkoneksi ke jaringan internet untuk mengunduh paket tambahan installasi IBM DB2 *expc.* Setelah seluruh virtual terbentuk kemudian di konfigurasi lebih lanjut agar dapat saling berkomunikasi antar virtual. Dalam proses untuk saling menghubungkan akan terbentuk *IP address* yang digunakan oleh masing-masing virtual sebagai *site* yang ditunjukkan pada tabel 5.1.

Tabel 5.1 Daftar *IP Address* Virtual *Server*

Virtual *server*	**IP Address**
Site utama	172.21.4.224
Site 1	172.21.4.232
Site 2	172.21.4.233
Site 3	172.21.4.237
Site 4	172.21.4.248
Site 5	172.21.4.210
Site 6	172.21.4.209
Site 7	172.21.4.250

5.4 Implementasi Basis Data

Dalam penelitian ini, penulis melakukan implementasi basis data dengan menggunakan *database management system* (DBMS) IBM DB2 *expc* 64 bit. Pada setiap *site* juga memerlukan beberapa paket tambahan untuk dapat meng*install* DBMS tersebut diantaranya paket *install* untuk *library* c++. Setelah DBMS ter*install* pada masing-masing *site*, selanjutnya dilakukan pebangunan basis data. dalam pembangunannya dibentuk beberapa tabel sebagai pengelompokan data dan tabel yang dibentuk antara lain tabel fakultas, tabel jenjang, tabel program studi, tabel status, tabel seleksi, tabel mahasiswa, tabel item, tabel transaksi dan tabel keuangan.

5.4.1 Tabel Fakultas

Dalam basis data yang digunakan dibentuk tabel fakultas. Tabel ini memiliki dua atribut yaitu k_fakultas yang menyimpan kode fakultas dan *conten* yang berisi data nama fakultas. Pembentukan tabel ditunjukkan seperti pada gambar 5.1.

```
1  CREATE TABLE FAKULTAS (
2    K_FAKULTAS VARCHAR(2) NOT NULL,
3    CONTENT VARCHAR(15) NOT NULL,
4    PRIMARY KEY(K_FAKULTAS)
5  );
```

Gambar 5.1 *Source Create* Tabel FAKULTAS

Pada gambar 5.1 ditunjukkan sebuah *query* pembuatan tabel fakultas. Pada tabel fakultas terdapat 2 atribut. Berikut ini adalah penjelasan dari gambar 5.1:

Baris 1 : merupakan perintah membuat tabel dengan nama FAKULTAS.

Baris 2 : nama atribut yang akan dibentuk pada tabel dengan nama K_FAKULTAS, tipe data *varchar* (2) dan dengan kondisi tidak boleh kosong (*not null*).

Baris 3 : nama atribut yang akan dibentuk pada tabel dengan nama *CONTENT*, tipe data *varchar* (15) dan dengan kondisi tidak boleh kosong (*not null*).

Baris 4 : merupakan perintah yang menandakan atribut K_FAKULTAS sebagai *primary key*

Baris 5 : merupakan penutup dari eksekusi *query*.

5.4.2 Tabel Jenjang

Dalam basis data yang digunakan dibentuk tabel jenjang. Tabel ini memiliki dua atribut yaitu k_jenjang yang menyimpan kode jenjang dan -*conten* yang berisi data nama jenjang. Pembentukan tabel ditunjukkan seperti pada gambar 5.2.

```
1  CREATE TABLE JENJANG (
2    K_JENJANG VARCHAR(2) NOT NULL,
3    CONTENT VARCHAR(15) NOT NULL,
4    PRIMARY KEY(K_JENJANG)
5  );
```

Gambar 5.2 *Source Create* Tabel JENJANG

Pada gambar 5.2 ditunjukkan sebuah *query* pembuatan tabel jenjang. Pada tabel jenjang terdapat dua atribut. Berikut ini adalah penjelasan dari gambar 5.2:

Baris 1 : merupakan perintah membuat tabel dengan nama JENJANG.

Baris 2 : nama atribut yang akan dibentuk pada tabel dengan nama K_JENJANG, tipe data *varchar* (2) dan dengan kondisi tidak boleh kosong (*not null*).

Baris 3 : nama atribut yang akan dibentuk pada tabel dengan nama *CONTENT*, tipe data *varchar* (15) dan dengan kondisi tidak boleh kosong (*not null*).

Baris 4 : merupakan perintah yang menandakan atribut K_JENJANG sebagai *primary key*

Baris 5 : merupakan penutup dari eksekusi *query*.

5.4.3 Tabel Program Studi

Dalam basis data yang digunakan dibentuk tabel program_studi. Tabel ini memiliki empat atribut yaitu k_prodi, k_fakultas, k_jenjang dan *content.* atribut k_prodi yang menyimpan kode program studi, atribut k_fakultas yang menyimpan kode fakultas, atribut k_jenjang yang menyimpan kode jenjang dan *conten* yang berisi data nama program studi. Pembentukan tabel ditunjukkan seperti pada gambar 5.3.

```
1  CREATE TABLE PROGRAM_STUDI (
2    K_PRODI VARCHAR(2) NOT NULL,
3    K_FAKULTAS VARCHAR(2) NOT NULL,
4    K_JENJANG VARCHAR(2) NOT NULL,
5    CONTENT VARCHAR(50) NOT NULL,
6    PRIMARY KEY(K_PRODI, K_FAKULTAS, K_JENJANG),
7    FOREIGN KEY(K_FAKULTAS)   REFERENCES FAKULTAS(K_FAKULTAS),
8    FOREIGN KEY(K_JENJANG) REFERENCES JENJANG(K_JENJANG)
9  );
```

Gambar 5.3 *Source Create* Tabel PROGRAM_STUDI

Pada gambar 5.3 ditunjukkan sebuah *query* pembuatan tabel program_studi. Pada tabel program_studi terdapat empat atribut. Berikut ini adalah penjelasan dari gambar 5.3:

Baris 1 : merupakan perintah membuat tabel dengan nama PROGRAM_STUDI.

Baris 2 : nama atribut yang akan dibentuk pada tabel dengan nama K_PRODI, tipe data *varchar* (2) dan dengan kondisi tidak boleh kosong (*not null*).

Baris 3 : nama atribut yang akan dibentuk pada tabel dengan nama K_PRODI, tipe data *varchar* (2) dan dengan kondisi tidak boleh kosong (*not null*).

Baris 4 : nama atribut yang akan dibentuk pada tabel dengan nama K_FAKULTAS, tipe data *varchar* (2) dan dengan kondisi tidak boleh kosong (*not null*).

Baris 5 : nama atribut yang akan dibentuk pada tabel dengan nama *CONTENT*, tipe data *varchar* (50) dan dengan kondisi tidak boleh kosong (*not null*).

Baris 6 : merupakan perintah yang menandakan atribut K_PRODI, K_FAKULTAS dan K_JENJANG sebagai *primary key*.

Baris 7 : merupakan perintah referensi data pada atribut K_FAKULTAS sebagai *foreign key* yang mengarah pada tabel FAKULTAS.

Baris 8 : merupakan perintah referensi data pada atribut K_JENJANG sebagai *foreign key* yang mengarah pada tabel JENJANG.

Baris 9 : merupakan penutup dari eksekusi *query*.

5.4.4 Tabel Seleksi

Dalam basis data yang digunakan dibentuk tabel seleksi. Tabel ini memiliki dua atribut yaitu k_seleksi yang menyimpan kode seleksi dan

conten yang berisi data nama seleksi. Pembentukan tabel ditunjukkan seperti pada gambar 5.4.

```
1  CREATE TABLE SELEKSI (
2    K_SELEKSI VARCHAR(2) NOT NULL,
3    CONTENT VARCHAR(20) NOT NULL,
4    PRIMARY KEY(K_SELEKSI)
5  );
```

Gambar 5.4 *Source Create* Tabel SELEKSI

Pada gambar 5.4 ditunjukkan sebuah *query* pembuatan tabel seleksi. Pada tabel seleksi terdapat dua atribut. Berikut ini adalah penjelasan dari gambar 5.4:

Baris 1 : merupakan perintah membuat tabel dengan nama SELEKSI.

Baris 2 : nama atribut yang akan dibentuk pada tabel dengan nama K_SELEKSI, tipe data *varchar* (2) dan dengan kondisi tidak boleh kosong (*not null*).

Baris 3 : nama atribut yang akan dibentuk pada tabel dengan nama *CONTENT*, tipe data *varchar* (20) dan dengan kondisi tidak boleh kosong (*not null*).

Baris 4 : merupakan perintah yang menandakan atribut K_SELEKSI sebagai *primary key*.

Baris 5 : merupakan penutup dari eksekusi *query*.

5.4.5 Tabel Status

Dalam basis data yang digunakan dibentuk tabel status. Tabel ini memiliki dua atribut yaitu k_status yang menyimpan kode status dan -*conten* yang berisi data nama status. Pembentukan tabel ditunjukkan seperti pada gambar 5.5.

```
1  CREATE TABLE STATUS (
2    K_STATUS VARCHAR(2) NOT NULL,
3    CONTENT VARCHAR(15) NOT NULL,
4    PRIMARY KEY(K_STATUS)
5  );
```

Gambar 5.5 *Source Create* Tabel STATUS

Pada gambar 5.5 ditunjukkan sebuah *query* pembuatan tabel status. Pada tabel status terdapat dua atribut. Berikut ini adalah penjelasan dari gambar 5.5:

Baris 1 : merupakan perintah membuat tabel dengan nama STATUS.

Baris 2 : nama atribut yang akan dibentuk pada tabel dengan nama K_STATUS, tipe data *varchar* (2) dan dengan kondisi tidak boleh kosong (*not null*).

Baris 3 : nama atribut yang akan dibentuk pada tabel dengan nama *CONTENT*, tipe data *varchar* (20) dan dengan kondisi tidak boleh kosong (*not null*).

Baris 4 : merupakan perintah yang menandakan atribut K_STATUS sebagai *primary key.*

Baris 5 : merupakan penutup dari eksekusi *query.*

5.4.6 Tabel Mahasiswa

Dalam basis data yang digunakan dibentuk tabel mahasiswa. Tabel ini memiliki sembilan atribut yaitu nim yang menyimpan nim mahasiswa, nama yang berisi data nama mahasiswa, angkatan yang berisi data angkatan, k_fakultas yang berisi data kode fakultas, k_prodi yang berisi data kode program studi, k_jenjang yang berisi data kode jenjang, k_seleksi yang berisi data kode seleksi, jenis_kelamin yang berisi data jenis kelamin dan k_status yang berisi kode status dari mahasiswa. Pembentukan tabel ditunjukkan seperti pada gambar 5.6.

```
1  CREATE TABLE MAHASISWA (
2    NIM VARCHAR(15) NOT NULL,
3    NAMA VARCHAR(30) NOT NULL,
4    ANGKATAN INTEGER NOT NULL,
5    K_FAKULTAS VARCHAR(2) NOT NULL,
6    K_PRODI VARCHAR(2) NOT NULL,
7    K_JENJANG VARCHAR(2) NOT NULL,
8    K_SELEKSI VARCHAR(2) NOT NULL,
9    JENIS_KELAMIN VARCHAR(1) NOT NULL,
10   K_STATUS VARCHAR(2) NOT NULL,
11   PRIMARY KEY(NIM),
12   FOREIGN KEY(K_PRODI,K_FAKULTAS,K_JENJANG) REFERENCES  PROGRAM_STUDI
13   (K_PRODI,K_FAKULTAS,K_JENJANG),
14   FOREIGN KEY(K_STATUS) REFERENCES STATUS(K_STATUS),
15   FOREIGN KEY(K_SELEKSI) REFERENCES SELEKSI(K_SELEKSI)
16 );
```

Gambar 5.6 *Source Create* Tabel MAHASISWA

Pada gambar 5.6 ditunjukkan sebuah *query* pembuatan tabel mahsiswa. Pada tabel mahasiswa terdapat sembilan atribut. Berikut ini adalah penjelasan dari gambar 5.6:

Baris 1 : merupakan perintah membuat tabel dengan nama MAHASISWA.

Baris 2 : nama atribut yang akan dibentuk pada tabel dengan nama nim, tipe data *varchar* (15) dan dengan kondisi tidak boleh kosong (*not null*).

Baris 3 : nama atribut yang akan dibentuk pada tabel adalah nama, tipe data *varchar* (30) dan dengan kondisi tidak boleh kosong (*not null*).

Baris 4 : nama atribut yang akan dibentuk pada tabel dengan nama angkatan, tipe data *integer* dan dengan kondisi tidak boleh kosong (*not null*).

Baris 5 : nama atribut yang akan dibentuk pada tabel dengan nama k_fakultas, tipe data *varchar* (2) dan dengan kondisi tidak boleh kosong (*not null*).

Baris 6 : nama atribut yang akan dibentuk pada tabel dengan nama k_prodi, tipe data *varchar* (2) dan dengan kondisi tidak boleh kosong (*not null*).

Baris 7 : nama atribut yang akan dibentuk pada tabel dengan nama jenjang, tipe data *varchar* (2) dan dengan kondisi tidak boleh kosong (*not null*).

Baris 8 : nama atribut yang akan dibentuk pada tabel dengan nama k_seleksi, tipe data *varchar* (2) dan dengan kondisi tidak boleh kosong (*not null*).

Baris 9 : nama atribut yang akan dibentuk pada tabel dengan nama jenis_kelamin, tipe data *varchar* (1) dan dengan kondisi tidak boleh kosong (*not null*).

Baris 10 : nama atribut yang akan dibentuk pada tabel dengan nama k_status, tipe data *varchar* (2) dan dengan kondisi tidak boleh kosong (*not null*).

Baris 11 : merupakan perintah yang menandakan atribut nim sebagai *primary key*.

Baris 12-13 : merupakan perintah yang menandakan k_prodi, k_fakultas, k_jenjang sebagai *foreign key* dan mereferensikan pada atribut yang sama dalam tabel PROGRAM_STUDI.

Baris 14 : merupakan perintah yang menandakan k_status sebagai *foreign key* dan mereferensikan pada atribut yang sama dalam tabel STATUS.

Baris 15 : merupakan perintah yang menandakan k_seleksi sebagai *foreign key* dan mereferensikan pada atribut yang sama dalam tabel SELEKSI.

Baris 16 : merupakan penutup dari eksekusi *query*.

5.4.7 Tabel Item

Dalam basis data yang digunakan dibentuk tabel item. Tabel ini memiliki dua atribut yaitu k_item yang menyimpan kode item dan *conten* yang berisi data nama item. Pembentukan tabel ditunjukkan seperti pada gambar 5.7.

```
1  CREATE TABLE ITEM (
2    K_ITEM VARCHAR(2) NOT NULL,
3    CONTENT VARCHAR(45) NOT NULL,
4    PRIMARY KEY(K_ITEM)
5  );
```

Gambar 5.7 *Source Create* Tabel ITEM

Pada gambar 5.7 ditunjukkan sebuah *query* pembuatan tabel item. Pada tabel item terdapat dua atribut. Berikut ini adalah penjelasan dari gambar 5.7:

Baris 1 : merupakan perintah membuat tabel dengan nama ITEM.

Baris 2 : nama atribut yang akan dibentuk pada tabel dengan nama K_ITEM, tipe data *varchar* (2) dan dengan kondisi tidak boleh kosong (*not null*).

Baris 3 : nama atribut yang akan dibentuk pada tabel dengan nama *CONTENT*, tipe data *varchar* (45) dan dengan kondisi tidak boleh kosong (*not null*).

Baris 4 : merupakan perintah yang menandakan atribut K_ITEM sebagai *primary key*.

Baris 5 : merupakan penutup dari eksekusi *query*.

5.4.8 Tabel Transaksi

Dalam basis data yang digunakan dibentuk tabel transaksi. Tabel ini memiliki dua atribut yaitu k_transaksi yang menyimpan kode transaksi dan *conten* yang berisi data nama transaksi. Pembentukan tabel ditunjukkan seperti pada gambar 5.8.

```
1  CREATE TABLE TRANSAKSI (
2    K_TRANSAKSI VARCHAR(2) NOT NULL,
3    CONTENT VARCHAR(20) NOT NULL,
4    PRIMARY KEY(K_TRANSAKSI)
5  );
```

Gambar 5.8 *Source Create* Tabel TRANSAKSI

Pada gambar 5.8 ditunjukkan sebuah *query* pembuatan tabel transaksi. Pada tabel transaksi terdapat dua atribut. Berikut ini adalah penjelasan dari gambar 5.8:

Baris 1 : merupakan perintah membuat tabel dengan nama TRANSAKSI.

Baris 2 : nama atribut yang akan dibentuk pada tabel dengan nama K_TRANSAKSI, tipe data *varchar* (2) dan dengan kondisi tidak boleh kosong (*not null*).

Baris 3 : nama atribut yang akan dibentuk pada tabel dengan nama *CONTENT*, tipe data *varchar* (20) dan dengan kondisi tidak boleh kosong (*not null*).

Baris 4 : merupakan perintah yang menandakan atribut K_TRANSAKSI sebagai *primary key*.

Baris 5 : merupakan penutup dari eksekusi *query*.

5.4.9 Tabel Keuangan

Dalam basis data yang digunakan dibentuk tabel keuangan. Tabel ini memiliki delapan atribut yaitu nim yang menyimpan nomor nim mahasiswa, tahun yang berisi data tahun pembayaran, is_ganjil yang menandakan semester genap dan ganjil, k_transaksi yang berisi data kode transaksi, k_item yang berisi data kode item, debet yang berisi data besar tagihan debet dan kredit yang berisi data besar tagihan kredit. Pembentukan tabel ditunjukkan seperti pada gambar 5.9.

```
1  CREATE TABLE KEUANGAN (
2    NIM VARCHAR(15) NOT NULL,
3    TAHUN INTEGER NOT NULL,
4    IS_GANJIL VARCHAR(1) NOT NULL,
5    K_TRANSAKSI VARCHAR(2) NOT NULL,
6    K_ITEM VARCHAR(2) NOT NULL,
7    TGL_TRANSAKSI TIMESTAMP NOTNULL,
8    DEBET BIGINT NOT NULL,
9    KREDIT BIGINT NOT NULL,
10   PRIMARY KEY(NIM, TAHUN,IS_GANJIL,K_TRANSAKSI, K_ITEM, TGL_TRANSAKSI),
11   FOREIGN KEY(NIM) REFERENCES MAHASISWA(NIM),
12   FOREIGN KEY(K_TRANSAKSI) REFERENCES TRANSAKSI(K_TRANSAKSI),
13   FOREIGN KEY(K_ITEM) REFERENCES ITEM(K_ITEM)
14 );
```

Gambar 5.9 *Source Create* Tabel KEUANGAN

Pada gambar 5.9 ditunjukkan sebuah *query* pembuatan tabel keuangan. Pada tabel keuangan terdapat delapan atribut. Berikut ini adalah penjelasan dari gambar 5.9:

Baris 1 : merupakan perintah membuat tabel dengan nama KEUANGAN.

Baris 2 : nama atribut yang akan dibentuk pada tabel dengan nama NIM, tipe data *varchar* (15) dan dengan kondisi tidak boleh kosong (*not null*).

Baris 3 : nama atribut yang akan dibentuk pada tabel dengan nama TAHUN, tipe data *integer* dan dengan kondisi tidak boleh kosong (*not null*).

Baris 4 : nama atribut yang akan dibentuk pada tabel dengan nama IS_GANJIL, tipe data *varchar* (1) dan dengan kondisi tidak boleh kosong (*not null*).

Baris 5 : nama atribut yang akan dibentuk pada tabel dengan nama K_TRANSAKSI, tipe data *varchar* (2) dan dengan kondisi tidak boleh kosong (*not null*).

Baris 6 : nama atribut yang akan dibentuk pada tabel dengan nama K_ITEM, tipe data *varchar* (2) dan dengan kondisi tidak boleh kosong (*not null*).

Baris 7 : nama atribut yang akan dibentuk pada tabel dengan nama TGL_TRANSAKSI, tipe data *timestamp* dan dengan kondisi tidak boleh kosong (*not null).*

Baris 8 : nama atribut yang akan dibentuk pada tabel dengan nama DEBET, tipe data *bigint* dan dengan kondisi tidak boleh kosong (*not null*).

Baris 9 : nama atribut yang akan dibentuk pada tabel dengan nama KREDIT, tipe data *bigint* dan dengan kondisi tidak boleh kosong (*not null).*

Baris 10 : merupakan perintah yang menandakan atribut NIM, TAHUN, IS_GANJIL, K_TRANSAKSI, K_ITEM dan TGL_TRANSAKSI sebagai *primary key.*

Baris 11 : merupakan perintah yang menandakan atribut nim sebagai *foreign key* dan mereferensikan pada atribut yang sama dalam tabel MAHASISWA.

Baris 12 : merupakan perintah yang menandakan atribut K_TRANSAKSI sebagai *foreign key* dan mereferensikan pada atribut yang sama dalam tabel transaksi.

Baris 11 : merupakan perintah yang menandakan atribut K_ITEM sebagai *foreign key* dan mereferensikan pada atribut yang sama dalam tabel ITEM.

Baris 12 : merupakan penutup dari eksekusi *query*.

5.5 Implementasi data

Pada penelitian ini, penulis menggunakan data *dummy* yang dibuat dengan mengacu pada jumlah mahasiswa aktif di universitas brawijaya. Dalam jumlah mahasiswa aktif tersebut digolongkan berdasarkan angkatan, fakultas, program studi dan jenjang. Dalam data yang digunakan penulis menggunakan data dengan jumlah mahasiswa jenjang S1. Pembuatan data dalam penelitian ini memanfaatkan *Microsoft excel* 2010 sebagai media untuk membuat data pendukung. Beberapa fungsi yang dimanfaatkan pada *Microsoft excel* penulis cantumkan pada tabel 5.2.

Tabel 5.2 Formula Excel

No	Formula	Penjelasan
1	=randbetween(bottom,up)	Berfungsi untuk menghasilkan nilai acak mulai dari nilai bottom hingga nilai up.
2	&	Berfungsi untuk menggabungkan data antar sel.
3	=vlookup(lookup_value,tabel_array,col_index_num,range_lookup)	Berfungsi untuk mencari data dan memberikan nilai berdasarkan parameter yang ada pada satu deret kolom dari beberapa deret kolom yang dijadikan sebagai referensi.

4	=right(text,[num_char])	Berfungsi untuk mengambil karakter pada suatu sel.
5	=[sel]+1	Berfungsi untuk membuat nomor secara *increment.*

Fungsi – fungsi pada tabel 5.2 digunakan untuk membuat data *dummy* nim mahasiswa, angkatan, kode fakultas, kode program studi, kode seleksi dan kode status pada tabel mahasiswa. Fungsi yang tertera pada tabel 5.2 juga digunakan untuk memberikan kode tambahan pada tabel K_TEMP yang merupakan tabel tambahan yang difungsikan sebagai tabel untuk membentuk data pada tabel keuangan.

Pada pembuatan data pada tabel keuangan, penulis membentuk 2 tabel tambahan terlepas dari tabel yang tertera pada desain RDBMS yaitu tabel K_TEMP dan tabel MASUK. Penyusunan kedua tabel yang ditunjukkan pada gambar 5.10 dan 5.11

5.5.1 Tabel Masuk

Tabel MASUK merupakan tabel yang berfungsi untuk membentuk data *dummy* pada tabel KEUANGAN. Tabel ini digunakan untuk membentuk data tanggal dan waktu transaksi keuangan pada tabel keuangan. Sedangkan pembentukan data dilakukan dengan memanfaatkan fitur *storage procedure* yang terdapat pada DB2. Pembentukan tabel MASUK yang ditunjukkan pada gambar 5.10.

```
1  CREATE TABLE MASUK(
2    KE INT NOT NULL GENERATED ALWAYS AS IDENTITY (START WITH 1 INCREMENT BY 1),
3   WAKTU TIMESTAMP,
4    PRIMARY KEY(KE)
5  );
```

Gambar 5.10 *Create* Tabel MASUK

Pada gambar 5.10 merupakan sebuah *query* yang digunakan untuk membentu tabel MASUK. Pada tabel tersebut memiliki dua atribut dengan satu atribut yang bersifat *auto increment.* Berikut ini adalah penjelasan *query* dari gambar 5.10:

Baris 1 : merupakan perintah untuk membentuk tabel dengan nama MASUK.

Baris 2 : merupakan pembentukan atribut dengan nama atribut KE, bertipe data integer dan dengan kondisi tidak bias kosong (*not null*) serta *auto increment.*

Baris 3 : merupakan pembentukan atribut dengan nama waktu yang bertipe data *timestamp.*

Baris 4 : merupaka perintah yang menyatakan atribu KE sebagai *primary key* pada tabel MASUK.

Baris 5 : merupakan akhir atau penutup dari *query.*

5.5.2 Tabel K_TEMP

Tabel K_TEMP merupakan tabel yang berfungsi untuk membentuk data *dummy* pada tabel KEUANGAN. Tabel ini digunakan sebagai tabel yang di gabungkan dengan tabel MASUK untuk membentuk data pada tabel KEUANGAN. Pembentukan tabel K_TEMP yang ditunjukkan pada gambar 5.11.

```
1  CREATE TABLE K_TEMP (
2    NIM VARCHAR(15) NOT NULL,
3    TAHUN INTEGER NOT NULL,
4    IS_GANJIL VARCHAR(1) NOT NULL,
5    K_TRANSAKSI VARCHAR(2) NOT NULL,
6    K_ITEM VARCHAR(2) NOT NULL,
7    DEBET BIGINT NOT NULL,
8    KREDIT BIGINT NOT NULL,
9    KE INTEGER NOT NULL,
10   PRIMARY KEY(NIM, TAHUN,IS_GANJIL,K_TRANSAKSI, K_ITEM, KE),
11   FOREIGN KEY(NIM) REFERENCES MAHASISWA(NIM),
12   FOREIGN KEY(K_TRANSAKSI) REFERENCES TRANSAKSI(K_TRANSAKSI),
13   FOREIGN KEY(K_ITEM) REFERENCES ITEM(K_ITEM),
14   FOREIGN KEY(KE) REFERENCES MASUK(KE)
15 );
```

Gambar 5.11 *Create* Tabel K_TEMP

Pada gambar 5.11 merupakan sebuah *query* yang digunakan untuk membentu tabel K_TEMP. Pada tabel tersebut memiliki delapan atribut. Berikut ini adalah penjelasan *query* dari gambar 5.11:

Baris 1 : merupakan perintah untuk membentuk tabel dengan nama K_TEMP.

Baris 2 : merupakan atribut yang dibentuk pada tabel dengan nama NIM, memiliki tipe data *varchar* (15) dan dengan kondisi tidak boleh kosong (*not null*).

Baris 3 : merupakan atribut yang dibentuk pada tabel dengan nama TAHUN, memiliki tipe data *integer* dan dengan kondisi tidak boleh kosong (*not null*).

Baris 4 : merupakan atribut yang dibentuk pada tabel dengan nama IS_GANJIL, memiliki tipe data *varchar* (1) dan dengan kondisi tidak boleh kosong (*not null*).

Baris 5 : merupakan atribut yang dibentuk pada tabel dengan nama K_TRANSAKSI, memiliki tipe data *varchar (2)* dan dengan kondisi tidak boleh kosong (*not null*).

Baris 6 : merupakan atribut yang dibentuk pada tabel dengan nama K_ITEM, memiliki tipe data *varchar (2)* dan dengan kondisi tidak boleh kosong (*not null*).

Baris 7 : merupakan atribut yang dibentuk pada tabel dengan nama DEBET, memiliki tipe data *bigint* dan dengan kondisi tidak boleh kosong (*not null*).

Baris 8 : merupakan atribut yang dibentuk pada tabel dengan nama KREDIT, memiliki tipe data *bigint* dan dengan kondisi tidak boleh kosong (*not null*).

Baris 9 : merupakan atribut yang dibentuk pada tabel dengan nama KE, memiliki tipe data *integer* dan dengan kondisi tidak boleh kosong (*not null*).

Baris 10 : merupakan perintah yang menyatakan atribut NIM, TAHUN, IS_GANJIL, K_TRANSAKSI, K_ITEM dan KE merupakan *primary key.*

Baris 11 : merupakan perintah yang menyatakan atribut NIM merupakan sebagai *foreign key* yang mereferensi pada atribut yang sama pada tabel MAHASISWA.

Baris 12 : merupakan perintah yang menyatakan atribut K_TRANSAKSI merupakan sebagai *foreign key* yang mereferensi pada atribut yang sama pada tabel TRANSAKSI.

Baris 13 : merupakan perintah yang menyatakan atribut K_ITEM merupakan sebagai *foreign key* yang mereferensi pada atribut yang sama pada tabel ITEM.

Baris 14 : merupakan perintah yang menyatakan atribut KE merupakan sebagai *foreign key* yang mereferensi pada atribut yang sama pada tabel MASUK.

Baris 15 : merupakan pembentukan atribut dengan nama waktu yang bertipe data *timestamp.*

5.5.3 *Generate Timestamp*

Setelah tabel terbentuk kemudian dilakukan pembuatan data pada tabel masuk. Pembuatan data pada tabel masuk dilakukan secara otomatis oleh sistem. Pada tabel masuk, data dari atribut ke diisi secara *auto increment* saat terdapat data masuk dan pada atribut waktu di isi dengan waktu sistem (*current timestamp*). Proses pembuatan pada tabel masuk dilakukan dengan memanfaatkan *storage procedure.* Pembuatan data pada tabel masuk ditunjukkan seperti pada gambar 5.12.

```
1  CREATE OR REPLACE PROCEDURE PROC_TS (IN p_start INT , IN p_end INT )
2  SPECIFIC PROC_TS
3  LANGUAGE SQL
4  TS: BEGIN
5    DECLARE v_current INTEGER;
6    SET v_current = p_start;
7    WHILE (v_current <= p_end) DO
8        insert into masuk(waktu) values(CURRENT TIMESTAMP);
9      SET v_current = v_current + 1;
10   END WHILE;
11 END TS
```

Gambar 5.12 *Source Storage Procedur Insert Data Into* MASUK

Pada gambar 5.12 merupakan *query storage procedure* yang berisi perulangan. Pada *query* tersebut dilakukan proses *insert* berulang yang dapat ditentukan berapa kali dilakukan proses perulangan. Berikut ini adalah penjelasan *query* dari gambar 5.12:

Baris 1 : merupakan perintah membentuk atau mengganti prosedur jika telah ada prosedur dengan nama yang sama dan pada prosedur yang dibentuk terdapat dua parameter masukan dengan tipe data *integer.*

Baris 2-3 : merupakan spesifikasi bahasa yang digunakan.

Baris 4 : penanda proses *statement query* dimulai.

Baris 5 : deklarasi variabel dengan nama variabel *v_current* yang memiliki tipe data integer.

Baris 6 : merupakan perintah menyamakan variabel *v_current* yang telah dideklarasikan dengan parameter masukan *p_start*.

Baris 7 : merupakan proses perulangan yang memanfaatkan jenis perulangan *do-while* dengan batasan nilai *v_current* sama dengan *p_end* untuk menghentikan proses perulangan.

Baris 8 : merupakan perintah *insert* yang ditujukan ke tabel masuk pada atribut waktu dengan nilai data berupa *current timestamp* yaitu waktu sistem saat itu.

Baris 9 : merupakan *statement* penambahan nilai pada variabel *v_current* yang digunakan sebagai pembatas perulangan.

Baris 10 : merupakan *statement* yang menandakan akhir dari *statement* perulangan.

Baris 11 : merupakan *statement* diakhirinya atau ditutupnya perintah pada *storage procedure*.

5.5.4 *Generate* Data Keuangan

Pada tahap ini, setelah data terbentuk pada tabel MASUK dan K_TEMP tahap selanjutnya adalah membentuk data pada tabelkeuangan dengan menggabungkan data pada tabel MASUK dan K_TEMP. Penggabungan kedua tabel menggunakan proses *inner join* pada atribut KE di kedua tabel. Proses menggabungkan dan memasukkan data kedalam tabel keuangan dapat dilihat pada gambar 5.13.

```
1  INSERT INTO KEUANGAN (
2    SELECT K.NIM, K.TAHUN, K.IS_GANJIL, K.K_TRANSAKSI, K.K_ITEM, W.WAKTU, K.DEBET,  K.KREDIT
3    FROM K_TEMP K, MASUK W
4    WHERE K.KE=W.MASUK
5  );
```

Gambar 5.13 *Source Insert Into* KEUANGAN

Pada gambar 5.13 ditunjukkan sebuah *query insert* data yang dilakukan terhadap tabel keuangan. Dalam *query* tersebut melibatkan dua

tabel yaitu tabel MASUK dan K_TEMP yang memiliki relasi pada sebuah atribut. Berikut ini adalah penjelasan dari *query* pada gambar 5.13:

Baris 1 : merupakan perintah untuk melakukan proses *insert* ke dalam tabel KEUANGAN.

Baris 2-4 : merupakan *statement query* untuk menampilkan atribut apa saja yang hendak dikeluarkan dari tabel K_TEMP dan MASUK yang saling memiliki relasi pada atribut ke sehingga memungkinkan untuk dilakukan prose *join*.

Baris 5 : merupakan baris penutup dari *query*.

5.5.5 *Drop* Tabel K_TEMP dan Tabel MASUK

Setelah data pada tabel KEUANGAN telah terbentuk, maka langkah selanjutnya adalah menghapus tabel K_TEMP dan tabel MASUK. Penghapusan kedua tabel tersebut karena kedua tabel hanya sebagai sarana untuk mendapatkan data. penghapusan tabel K_TEMP dan tabel MASUK dapat dilihat pada gambar 5.14.

```
1  DROP TABLE K_TEMP;
2  DROP TABLE MASUK;
```

Gambar 5.14 *Source Drop* Tabel K_TEMP dan MASUK

Pada gambar 5.14 ditunjukkan *query* yang digunakan untuk menghapus tabel. Pada permasalahan ini tabel yang dihapus merupakan tabel K_TEMP dan MASUK. Proses penghapusan dilakukan terhadap tabel K_TEMP terlebih dahulu karena pada tabel ini me*reference* terhadap tabel yang lain seperti tabel MASUK dan MAHASISWA. Setelah tabel K_TEMP terhapus dengan menggunakan *query* seperti pada gambar 5.14 pada baris 1, dapat dilanjutkan dengan menghapus tabel masuk dengan menggunakan perintas seperti pada gambar 5.14 baris 2.

5.6 Implementasi Federasi Basis Data

Pada penelitian ini, penulis membangun sebuah basis data terdistribusi untuk melakukan penelitian. Basis data yang dibangun terdiri dari delapan *server*

yang dibentuk melalui virtualisasi menjadi *site.* Dari delapan *site*, terdapat satu *site* yang berperan sebagai *site* utama sebagai basis data pusat. Dalam pembangunannya, perlu di konfigurasi pada setiap *site* agar federasi basis data dapat dilakukan. Pada pembangungan federasi basisdata perlu dilakukan konfigurasi basis data, *catalog* basis data hingga pembentukan jaringan beserta penamaan inisialisasi tabel yang akan *remote* pada setiap *site*nya.

5.6.1 Konfigurasi Basis Data

Pada tahap awal basis data terbentuk, servis federasi basis data masih belum di fungsikan. Sehingga perlu aktifkan terlebih dahulu fungsi pada DBMS di masing-masing *site.* Perintah aktifasi fungsi federasi basis data seperti yang ditunjukkan pada gambar 5.15.

```
1  db2 UPDATE DBM CFG USING FEDERATED YES
```

Gambar 5.15 *Source Update dbm cfg*

Pada gambar 5.15 ditunjukkan sebuah perintah yang berjalan di db2 dengan tujuan untuk menyalakan fungsi federasi basis data. perintah pada gambar 5.15 diimplementasikan pada semua basis data yang dibentuk di kedelapan *site.* Dengan mengaktifkan fungsi federasi pada tahap awal ini memungkinkan setiap *site* dapat berkomunikasi sebagai federasi basis data pada basis data yang terbentuk dan di katalogkan.

5.6.7 *Catalog Node*

Setelah dilakukan aktifasi fungsi feerasi basis data pada setiap *site*, langkah selanjutnya adalah dilakukan proses *catalog node*. Fungsi dari *catalog node* adalah melakukan referensi *node* atau jaringan agar DBMS dapat terhubung pada basis data lain yang tergabung pada sebuah jaringan di *site* yang berbeda. Perintah *catalog node* seperti yang ditunjukkan pada gambart 5.16.

```
1  db2 CATALOG TCPIP NODE SERVER1 REMOTE 172.21.4.232  SERVER 50000
2  db2 CATALOG TCPIP NODE SERVER1 REMOTE 172.21.4.233  SERVER 50000
3  db2 CATALOG TCPIP NODE SERVER1 REMOTE 172.21.4.237  SERVER 50000
4  db2 CATALOG TCPIP NODE SERVER1 REMOTE 172.21.4.248  SERVER 50000
5  db2 CATALOG TCPIP NODE SERVER1 REMOTE 172.21.4.210  SERVER 50000
6  db2 CATALOG TCPIP NODE SERVER1 REMOTE 172.21.4.209  SERVER 50000
7  db2 CATALOG TCPIP NODE SERVER1 REMOTE 172.21.4.250  SERVER 50000
```

Gambar 5.16 *Source Catalog Node*

Pada gambar 5.16 ditunjukkan perintah *catalog node* yang berjalan di terminal linux. Perintah tersebut merupakan perintah *catalog* jaringan yang dilakukan di *site* pusat. Perintah tersebut melakukan penandaan pada jaringan TCP/IP dengan ip tujuan(*site*) yang akan difederasikan dengan membentuk sebagai *node* dan mengakses basis data melalui port yang digunakan pada *site* untuk komunikasi basis data pada port 50000. Berikut ini adalah penjelasan dari perintah pada gambar 5.16:

Baris 1 : pembuatan *catalog node* atau jaringan melalui terminal linux dengan nama *node SERVER1* yang melakukan *remote* pada ip 172.21.4.232 melalui port 50000.

Baris 2 : pembuatan *catalog node* atau jaringan melalui terminal linux dengan nama *node SERVER2* yang melakukan *remote* pada ip 172.21.4.233 melalui port 50000.

Baris 3 : pembuatan *catalog node* atau jaringan melalui terminal linux dengan nama *node SERVER3* yang melakukan *remote* pada ip 172.21.4.237 melalui port 50000.

Baris 4 : pembuatan *catalog node* atau jaringan melalui terminal linux dengan nama *node SERVER4* yang melakukan *remote* pada ip 172.21.4.248 melalui port 50000.

Baris 5 : pembuatan *catalog node* atau jaringan melalui terminal linux dengan nama *node SERVER4* yang melakukan *remote* pada ip 172.21.4.210 melalui port 50000.

Baris 6 : pembuatan *catalog node* atau jaringan melalui terminal linux dengan nama *node SERVER6* yang melakukan *remote* pada ip 172.21.4.209 melalui port 50000.

Baris 7 : pembuatan *catalog node* atau jaringan melalui terminal linux dengan nama *node SERVER7* yang melakukan *remote* pada ip 172.21.4.250 melalui port 50000.

5.6.8 *Catalog* Basis data

Catalog basis data merupakan tahap pemberian tanda terhadap basis data pada *site* lain yang tergabung pada federasi basis data. proses pendaftaran nama basis data dihubungkan dengan katalog *node* yang telah dibuat sebelumnya. Dengan proses katalog basis data ini, *site* utama dapat terhubung dengan basis data lain yang terdapat pada *site* lain yang tergabung dalam federasi basis data. Bentuk perintah yang digunakan berjalan pada terminal db2 di linux dapat dilihat seperti pada gambar 5.17.

```
1 db2 CATALOG DB HUKUM AT NODE SERVER1
2 db2 CATALOG DB FIA AT NODE SERVER2
3 db2 CATALOG DB PRKANAN AT NODE SERVER3
4 db2 CATALOG DB PTRNAKAN AT NODE SERVER4
5 db2 CATALOG DB TEKNIK AT NODE SERVER5
6 db2 CATALOG DB MIPA AT NODE SERVER6
7 db2 CATALOG DB FISIP AT NODE SERVER7
```

Gambar 5.17 *Source Catalog Basis Data*

Pada gambar 5.17 ditunjukkan perintah membuat katalog basis data yang berjalan pada *site* pusat dengan memanfaatkan terminal linux. Pada perintah tersebut dibentuk katalog basis data pada *site* pusat dengan nama masing-masing basis data yang akan dihubungkan dan didaftarkan sesuai dengan *node server* yang ada. Berikut ini adalah penjelasan dari perintah yang digunakan pada gambar 5.17:

Baris 1 : pembuatan *catalog* bais data melalui terminal linux dengan nama *db* HUKUM yang yang dapat diakses melalui *node SERVER1*.

Baris 2 : pembuatan *catalog* bais data melalui terminal linux dengan nama *db* FIA yang yang dapat diakses melalui *node SERVER2*.

Baris 3 : pembuatan *catalog* bais data melalui terminal linux dengan nama *db* PRKANAN yang yang dapat diakses melalui *node SERVER3*.

Baris 4 : pembuatan *catalog* bais data melalui terminal linux dengan nama *db* PTRNAKAN yang yang dapat diakses melalui *node SERVER4*.

Baris 5 : pembuatan *catalog* bais data melalui terminal linux dengan nama *db* TEKNIK yang yang dapat diakses melalui *node SERVER5*.

Baris 6 : pembuatan *catalog* bais data melalui terminal linux dengan nama *db* MIPA yang yang dapat diakses melalui *node SERVER6*.

Baris 7 : pembuatan *catalog* bais data melalui terminal linux dengan nama *db* FISIP yang yang dapat diakses melalui *node SERVER7*.

Saat perintah pada gambar 5.17 dieksekusi, maka DBMS pada *site* pusat dapat terhubung dengan basis data yang terdapat pada *site* lain. Penghubungan basis data dilakukan sesuai dengan basis data yang akan dihubungkan melalui *node server* yang telah dibuat. Pada proses penghubungannya, DBMS pusat tinggal meminta menghubungkan terhadap basis data pada *site* lain dengan mencantumkan *user* dan *password* basis data.

5.6.9 Membuat *wrapper*

Wrapper merupakan sebuah jembatan komunikasi yang digunakan. Fungsi dari *wrapper* sendiri merupakan sebagai media protokol yang menghubungkan *site* utama dengan *site* lain yang tergabung pada federasi basis data. Pada penelitian ini penulis membuat *wrapper* dengan nama DRDA sehingga jika dipantau dari segi komunikasi jaringan protokol yang digunakan adalah DRDA. Proses pembuatan *wrapper* dapat dilakukan dengan dua cara, yaitu dapat melakukan eksekusi perintah langsung melalui terminal linux atau juga memanfaatkan aplikasi pendukung seperti data studio. Perintah yang penulis gunakan untuk membuat *wrapper* seperti pada gambar 5.18.

```
1  CREATE WRAPPER DRDA LIBRARY 'libdb2drda.so' OPTIONS ( ADD DB2_FENCED 'N' );
```

Gambar 5.18 *Source Create Wrapper* DRDA

Pada gambar 5.18 ditunjukkan sebuah perintah untuk membuat sebuah *wrapper*. pada perintah tersebut dibuat sebuah *wrapper* dengan nama DRDA yang menggunakan *library* 'libdb2drda.so' dengan penambahan opsi DB2_FENCED 'N'. pada perintah tersebut harus disesuaikan pula *library* yang digunakan. Dalam penelitian ini, penulis menjalankan DBMS DB2 pada *platform* linux sehingga *library* yang digunakan adalah 'libdb2drda.so'.

5.6.10 Membuat *Server* Federasi Basis Data

Pada tahap ini, penulis membuat *server* federasi basis data dengan bantuan data studio. Pembuatan *server* basis data berfungsi untuk memberikan tanda pada basis DBMS *site* pusat agar dapat terhubung ke *server* pada *site* lain melalui *wrapper* yang telah dibuat dengan autentikasi nama pengguna dan sandi basis data yang dituju. Perintah dari pembuatan *server* federasi basis data yang ditunukkan seperti pada gambar 5.19.

```
1  CREATE SERVER SERVER1 TYPE DB2/UDB VERSION '10.5' WRAPPER DRDA AUTHORIZATION
2  "db2inst1" PASSWORD "skripsi" OPTIONS ( NODE 'SERVER1', ADD DBNAME 'HUKUM' );
3  CREATE SERVER SERVER2 TYPE DB2/UDB VERSION '10.5' WRAPPER DRDA AUTHORIZATION
4  "db2inst2" PASSWORD "skripsi" OPTIONS ( NODE 'SERVER2', ADD DBNAME 'FIA' );
5  CREATE SERVER SERVER3 TYPE DB2/UDB VERSION '10.5' WRAPPER DRDA AUTHORIZATION
6  "db2inst3" PASSWORD "skripsi" OPTIONS ( NODE 'SERVER3', ADD DBNAME 'PRKANAN' );
7  CREATE SERVER SERVER4 TYPE DB2/UDB VERSION '10.5' WRAPPER DRDA AUTHORIZATION
8  "db2inst4" PASSWORD "skripsi" OPTIONS ( NODE 'SERVER4', ADD DBNAME 'PTRNAKAN' );
9  CREATE SERVER SERVER5 TYPE DB2/UDB VERSION '10.5' WRAPPER DRDA AUTHORIZATION
10 "db2inst5" PASSWORD "skripsi" OPTIONS ( NODE 'SERVER5', ADD DBNAME 'TEKNIK' );
11 CREATE SERVER SERVER6 TYPE DB2/UDB VERSION '10.5' WRAPPER DRDA AUTHORIZATION
12 "db2inst6" PASSWORD "skripsi" OPTIONS ( NODE 'SERVER6', ADD DBNAME 'MIPA' );
13 CREATE SERVER SERVER7 TYPE DB2/UDB VERSION '10.5' WRAPPER DRDA AUTHORIZATION
14 "db2inst7" PASSWORD "skripsi" OPTIONS ( NODE 'SERVER7', ADD DBNAME 'FISIP' );
```

Gambar 5.19 *Source Create Server*

Pada gambar 5.19 ditunjukkan perintah-perintah yang digunakan untuk membuat *server* federasi basis data. dalam eksekusi perintahnya, penulis memanfaatkan bantuan data studio sebagai media eksekusi perintah. Berikut ini adalah penjelasan dari perintah-perintah pada gambar 5.19:

Baris 1-2 : pembuatan *server* federasi bais data melalui data studio dengan nama sesuai dengan katalog *SERVER1* tipe DBMS DB2/UDB versi 10.5, menggunakan

wrapper DRDA dengan otoritas pengguna db2inst1 dan password skripsi pada *node SERVER1 dan* nama basis data yang dituju adalah HUKUM.

Baris 3-4 : pembuatan *server* federasi bais data melalui data studio dengan nama sesuai dengan katalog *SERVER2* tipe DBMS DB2/UDB versi 10.5, menggunakan *wrapper* DRDA dengan otoritas pengguna db2inst2 dan password skripsi pada *node SERVER2 dan* nama basis data yang dituju adalah FIA.

Baris 5-6 : pembuatan *server* federasi bais data melalui data studio dengan nama sesuai dengan katalog *SERVER3* tipe DBMS DB2/UDB versi 10.5, menggunakan *wrapper* DRDA dengan otoritas pengguna db2inst3 dan password skripsi pada *node SERVER3 dan* nama basis data yang dituju adalah PRKANAN.

Baris 7-8 : pembuatan *server* federasi bais data melalui data studio dengan nama sesuai dengan katalog *SERVE4* tipe DBMS DB2/UDB versi 10.5, menggunakan *wrapper* DRDA dengan otoritas pengguna db2inst4 dan password skripsi pada *node SERVER4 dan* nama basis data yang dituju adalah PTRNAKAN.

Baris 9-10 : pembuatan *server* federasi bais data melalui data studio dengan nama sesuai dengan katalog *SERVE5* tipe DBMS DB2/UDB versi 10.5, menggunakan *wrapper* DRDA dengan otoritas pengguna db2inst5 dan password skripsi pada *node SERVER5 dan* nama basis data yang dituju adalah TEKNIK.

Baris 11-12 : pembuatan *server* federasi bais data melalui data studio dengan nama sesuai dengan katalog *SERVE6* tipe DBMS DB2/UDB versi 10.5, menggunakan *wrapper* DRDA dengan otoritas pengguna db2inst6 dan

password skripsi pada *node SERVER6 dan* nama basis data yang dituju adalah MIPA.

Baris 13-14 : pembuatan *server* federasi bais data melalui data studio dengan nama sesuai dengan katalog *SERVE7* tipe DBMS DB2/UDB versi 10.5, menggunakan *wrapper* DRDA dengan otoritas pengguna db2inst7 dan password skripsi pada *node SERVER7 dan* nama basis data yang dituju adalah FISIP.

5.6.11 Pembuatan *User Mapping*

Pada tahap ini dilakukan pembuatan *user mapping* pada *site* pusat. Pembuatan *user mapping* berguna untuk memetakan *user* pada *site* federasi. Dalam pemetaannya dilakukan untuk *user* pada *site* pusat. Perintah pemetaan *user* seperti yang ditunjukkan pada gambar 5.20.

```
1  CREATE USER MAPPING FOR "SKRIPSI" SERVER SERVER1 OPTIONS ( ADD REMOTE_AUTHID
2  'db2inst1', ADD REMOTE_PASSWORD 'skripsi' );
3  CREATE USER MAPPING FOR "SKRIPSI" SERVER SERVER2 OPTIONS ( ADD REMOTE_AUTHID
4  'db2inst2', ADD REMOTE_PASSWORD 'skripsi' );
5  CREATE USER MAPPING FOR "SKRIPSI" SERVER SERVER3 OPTIONS ( ADD REMOTE_AUTHID
6  'db2inst3', ADD REMOTE_PASSWORD 'skripsi' );
7  CREATE USER MAPPING FOR "SKRIPSI" SERVER SERVER4 OPTIONS ( ADD REMOTE_AUTHID
8  'db2inst4', ADD REMOTE_PASSWORD 'skripsi' );
9  CREATE USER MAPPING FOR "SKRIPSI" SERVER SERVER5 OPTIONS ( ADD REMOTE_AUTHID
10 'db2inst5', ADD REMOTE_PASSWORD 'skripsi' );
11 CREATE USER MAPPING FOR "SKRIPSI" SERVER SERVER6 OPTIONS ( ADD REMOTE_AUTHID
12 'db2inst6', ADD REMOTE_PASSWORD 'skripsi' );
13 CREATE USER MAPPING FOR "SKRIPSI" SERVER SERVER7 OPTIONS ( ADD REMOTE_AUTHID
14 'db2inst7', ADD REMOTE_PASSWORD 'skripsi' );
```

Gambar 5.20 *Source Create User Mapping*

Pada gambar 5.20 ditunjukkan perintah-perintah yang digunakan untuk membentu pemataan *user site* utama. pada perintah tersebut dibuat untuk memetakan pengguna pada *server* tujuan dengan *user* lokal beserta *password*nya. Berikut ini adalah penjelasan dari perintah-perintah pada gambar 5.20:

Baris 1-2 : pembuatan *user mapping* untuk *user* yang digunakan pada *site* utama yaitu SKRIPSI pada *SERVER1* dengan tambahan *remote authentication id* merupakan *user* lokal yang digunakan yaitu db2inst1 dan *remote password* adalah ‘skripsi’.

Baris 3-4 : pembuatan *user mapping* untuk *user* yang digunakan pada *site* utama yaitu SKRIPSI pada *SERVER2* dengan tambahan *remote authentication id* merupakan *user* lokal yang digunakan yaitu db2inst2 dan *remote password* adalah ‘skripsi’.

Baris 5-6 : pembuatan *user mapping* untuk *user* yang digunakan pada *site* utama yaitu SKRIPSI pada *SERVER3* dengan tambahan *remote authentication id* merupakan *user* lokal yang digunakan yaitu db2inst3 dan *remote password* adalah ‘skripsi’.

Baris 7-8 : pembuatan *user mapping* untuk *user* yang digunakan pada *site* utama yaitu SKRIPSI pada *SERVER4* dengan tambahan *remote authentication id* merupakan *user* lokal yang digunakan yaitu db2inst4 dan *remote password* adalah ‘skripsi’.

Baris 9-10 : pembuatan *user mapping* untuk *user* yang digunakan pada *site* utama yaitu SKRIPSI pada *SERVER5* dengan tambahan *remote authentication id* merupakan *user* lokal yang digunakan yaitu db2inst5 dan *remote password* adalah ‘skripsi’.

Baris 11-12 : pembuatan *user mapping* untuk *user* yang digunakan pada *site* utama yaitu SKRIPSI pada *SERVER6* dengan tambahan *remote authentication id* merupakan *user* lokal yang digunakan yaitu db2inst6 dan *remote password* adalah ‘skripsi’.

Baris 13-14 : pembuatan *user mapping* untuk *user* yang digunakan pada *site* utama yaitu SKRIPSI pada *SERVER7* dengan tambahan *remote authentication id* merupakan *user* lokal yang digunakan yaitu db2inst7 dan *remote password* adalah ‘skripsi’.

5.6.12 Pembuatan *Nick Name* Tabel

Pada tahap ini dilakukan pembuatan *nick name* tabel yang merupakan penginisialisasi tabel pada *site* lokal. Penggunaak *nick name* ini berfungsi pada pemanggilan data dari tabel-tabel yang tersimpan pada basis data di*site* lokal. Berikut ini adalah pembuatan *nick name* yang ditunjukkan seperti pada gambar 5.21:

```
1  CREATE NICKNAME SKRIPSI.FAK1 FOR SERVER1.DB2INST1.FAKULTAS;
2  CREATE NICKNAME SKRIPSI.PROD1 FOR SERVER1.DB2INST1.PROGRAM_STUDI;
3  CREATE NICKNAME SKRIPSI.MHS1 FOR SERVER1.DB2INST1.MAHASISWA;
4  CREATE NICKNAME SKRIPSI.KEU1 FOR SERVER1.DB2INST1.KEUANGAN;
5
6  CREATE NICKNAME SKRIPSI.FAK2 FOR SERVER2.DB2INST2.FAKULTAS;
7  CREATE NICKNAME SKRIPSI.PROD2 FOR SERVER2.DB2INST2.PROGRAM_STUDI;
8  CREATE NICKNAME SKRIPSI.MHS2 FOR SERVER2.DB2INST2.MAHASISWA;
9  CREATE NICKNAME SKRIPSI.KEU2 FOR SERVER2.DB2INST2.KEUANGAN;
10
11 CREATE NICKNAME SKRIPSI.FAK3 FOR SERVER3.DB2INST3.FAKULTAS;
12 CREATE NICKNAME SKRIPSI.PROD3 FOR SERVER3.DB2INST3.PROGRAM_STUDI;
13 CREATE NICKNAME SKRIPSI.MHS3 FOR SERVER3.DB2INST3.MAHASISWA;
14 CREATE NICKNAME SKRIPSI.KEU3 FOR SERVER3.DB2INST3.KEUANGAN;
15
16 CREATE NICKNAME SKRIPSI.FAK4 FOR SERVER4.DB2INST4.FAKULTAS;
17 CREATE NICKNAME SKRIPSI.PROD4 FOR SERVER4.DB2INST4.PROGRAM_STUDI;
18 CREATE NICKNAME SKRIPSI.MHS4 FOR SERVER4.DB2INST4.MAHASISWA;
19 CREATE NICKNAME SKRIPSI.KEU4 FOR SERVER4.DB2INST4.KEUANGAN;
20
21 CREATE NICKNAME SKRIPSI.FAK5 FOR SERVER5.DB2INST5.FAKULTAS;
22 CREATE NICKNAME SKRIPSI.PROD5 FOR SERVER5.DB2INST5.PROGRAM_STUDI;
23 CREATE NICKNAME SKRIPSI.MHS5 FOR SERVER5.DB2INST5.MAHASISWA;
24 CREATE NICKNAME SKRIPSI.KEU5 FOR SERVER5.DB2INST5.KEUANGAN;
25
26 CREATE NICKNAME SKRIPSI.FAK6 FOR SERVER6.DB2INST6.FAKULTAS;
27 CREATE NICKNAME SKRIPSI.PROD6 FOR SERVER6.DB2INST6.PROGRAM_STUDI;
28 CREATE NICKNAME SKRIPSI.MHS6 FOR SERVER6.DB2INST6.MAHASISWA;
29 CREATE NICKNAME SKRIPSI.KEU6 FOR SERVER6.DB2INST6.KEUANGAN;
30
31 CREATE NICKNAME SKRIPSI.FAK7 FOR SERVER7.DB2INST7.FAKULTAS;
32 CREATE NICKNAME SKRIPSI.PROD7 FOR SERVER7.DB2INST7.PROGRAM_STUDI;
33 CREATE NICKNAME SKRIPSI.MHS7 FOR SERVER7.DB2INST7.MAHASISWA;
34 CREATE NICKNAME SKRIPSI.KEU7 FOR SERVER7.DB2INST7.KEUANGAN;
```

Gambar 5.21 *Source Create Nick Name*

Pada gambar 5.21 ditunjukkan perintah-perintah pembuatan *nick name* tabel pada *site* utama yang tersimpan pada masing-masing *site* lokal. Pada pembuatannya dibuat nama inisialisasi dengan skema seperti yang digunakan pada *site* utama yang ditujukan untuk tabel yang tersimpan pada *site* lokal dengan *server* dan skema yang digunakan pada *site* lokal. Berikut ini adalah penjelasan perintah-perintah pada gambar 5.21:

Baris 1 : pembuatan *nick name* dengan skema skripsi dan nama inisial FAK1 untuk tabel FAKULTAS yang terdapat pada *site* lokal dengan skema DB2INST1 pada *SERVER*1.

Baris 2 : pembuatan *nick name* dengan skema skripsi dan nama inisial PROD1 untuk tabel PROGRAM_STUDI yang terdapat pada *site* lokal dengan skema DB2INST1 pada *SERVER*1.

Baris 3 : pembuatan *nick name* dengan skema skripsi dan nama inisial MHS1 untuk tabel MAHASISWA yang terdapat pada *site* lokal dengan skema DB2INST1 pada *SERVER*1.

Baris 4 : pembuatan *nick name* dengan skema skripsi dan nama inisial KEU1 untuk tabel KEUANGAN yang terdapat pada *site* lokal dengan skema DB2INST1 pada *SERVER*1.

Baris 6 : pembuatan *nick name* dengan skema skripsi dan nama inisial FAK2 untuk tabel FAKULTAS yang terdapat pada *site* lokal dengan skema DB2INST2 pada *SERVER*2.

Baris 7 : pembuatan *nick name* dengan skema skripsi dan nama inisial PROD2 untuk tabel PROGRAM_STUDI yang terdapat pada *site* lokal dengan skema DB2INST2 pada *SERVER*2.

Baris 8 : pembuatan *nick name* dengan skema skripsi dan nama inisial MHS2 untuk tabel MAHASISWA yang terdapat pada *site* lokal dengan skema DB2INST2 pada *SERVER*2.

Baris 9 : pembuatan *nick name* dengan skema skripsi dan nama inisial KEU2 untuk tabel KEUANGAN yang terdapat pada *site* lokal dengan skema DB2INST2 pada *SERVER*2.

Baris 11 : pembuatan *nick name* dengan skema skripsi dan nama inisial FAK3 untuk tabel FAKULTAS yang terdapat pada *site* lokal dengan skema DB2INST3 pada *SERVER*3.

Baris 12 : pembuatan *nick name* dengan skema skripsi dan nama inisial PROD3 untuk tabel PROGRAM_STUDI yang terdapat pada *site* lokal dengan skema DB2INST3 pada *SERVER*3.

Baris 13 : pembuatan *nick name* dengan skema skripsi dan nama inisial MHS3 untuk tabel MAHASISWA yang terdapat pada *site* lokal dengan skema DB2INST3 pada *SERVER*3.

Baris 14 : pembuatan *nick name* dengan skema skripsi dan nama inisial KEU3 untuk tabel KEUANGAN yang terdapat pada *site* lokal dengan skema DB2INST3 pada *SERVER*3.

Baris 16 : pembuatan *nick name* dengan skema skripsi dan nama inisial FAK4 untuk tabel FAKULTAS yang terdapat pada *site* lokal dengan skema DB2INST4 pada *SERVER*4.

Baris 17 : pembuatan *nick name* dengan skema skripsi dan nama inisial PROD4 untuk tabel PROGRAM_STUDI yang terdapat pada *site* lokal dengan skema DB2INST4 pada *SERVER*4.

Baris 18 : pembuatan *nick name* dengan skema skripsi dan nama inisial MHS4 untuk tabel MAHASISWA yang terdapat pada *site* lokal dengan skema DB2INST4 pada *SERVER*4.

Baris 19 : pembuatan *nick name* dengan skema skripsi dan nama inisial KEU4 untuk tabel KEUANGAN yang terdapat pada *site* lokal dengan skema DB2INST4 pada *SERVER*4.

Baris 21 : pembuatan *nick name* dengan skema skripsi dan nama inisial FAK5 untuk tabel FAKULTAS yang terdapat pada *site* lokal dengan skema DB2INST5 pada *SERVER*5.

Baris 22 : pembuatan *nick name* dengan skema skripsi dan nama inisial PROD5 untuk tabel PROGRAM_STUDI yang terdapat pada *site* lokal dengan skema DB2INST5 pada *SERVER*5.

Baris 23 : pembuatan *nick name* dengan skema skripsi dan nama inisial MHS5 untuk tabel MAHASISWA yang terdapat pada *site* lokal dengan skema DB2INST5 pada *SERVER*5.

Baris 24 : pembuatan *nick name* dengan skema skripsi dan nama inisial KEU5 untuk tabel KEUANGAN yang terdapat pada *site* lokal dengan skema DB2INST5 pada *SERVER*5.

Baris 26 : pembuatan *nick name* dengan skema skripsi dan nama inisial FAK6 untuk tabel FAKULTAS yang terdapat pada *site* lokal dengan skema DB2INST6 pada *SERVER*6.

Baris 27 : pembuatan *nick name* dengan skema skripsi dan nama inisial PROD6 untuk tabel PROGRAM_STUDI yang terdapat pada *site* lokal dengan skema DB2INST6 pada *SERVER*6.

Baris 28 : pembuatan *nick name* dengan skema skripsi dan nama inisial MHS6 untuk tabel MAHASISWA yang terdapat pada *site* lokal dengan skema DB2INST6 pada *SERVER*6.

Baris 29 : pembuatan *nick name* dengan skema skripsi dan nama inisial KEU6 untuk tabel KEUANGAN yang terdapat pada *site* lokal dengan skema DB2INST6 pada *SERVER*6.

Baris 31 : pembuatan *nick name* dengan skema skripsi dan nama inisial FAK7 untuk tabel FAKULTAS yang terdapat pada *site* lokal dengan skema DB2INST7 pada *SERVER*7.

Baris 32 : pembuatan *nick name* dengan skema skripsi dan nama inisial PROD7 untuk tabel PROGRAM_STUDI yang terdapat pada *site* lokal dengan skema DB2INST7 pada *SERVER*7.

Baris 33 : pembuatan *nick name* dengan skema skripsi dan nama inisial MHS7 untuk tabel MAHASISWA yang terdapat pada *site* lokal dengan skema DB2INST7 pada *SERVER*7.

Baris 34 : pembuatan *nick name* dengan skema skripsi dan nama inisial KEU7 untuk tabel KEUANGAN yang terdapat pada *site* lokal dengan skema DB2INST7 pada *SERVER*7.

5.7 Implementasi fragmentasi data

Dalam penelitian ini, penulis melakukan fragmentasi data yang dilakukan secara horisontal. Seluruh data yang telah dibuat di*import* ke dalam basis data yang terletak pada *site* utama. proses fragmentasi dilakukan secara horisontal berdasarkan basis data yang digunakan dalam universitas brawijaya. Pada

implementasi fragmentasi data, data mahasiswa secara keseluruhan di fragmen berdasar fakultas sehingga fragmentasi data pada beberapa tabel lain mengacu juga berdasar fakultas kecuali untuk fragmentasi keuangan mengikuti nim mahasiswa yang tersimpan pada tabel mahasiswa setiap *site.*

5.7.1 Fragmentasi *Site*1

Fragmentasi data dilakukan secara horizontal. Pada fragmentasi data yang akan diletakkan pada *site*1 mengacu pada data fakultas, program studi dan mahasiswa yang terdapat pada fakultas hukum. Dalam proses memasukkan data pada *site*1 menggunakan pemecahan data dengan data fakultas hukum yang memiliki kode fakultas '02'. Proses fragmentasi data yang dilakukan pada *site* utama seperti yang ditunjukkan oleh gambar 5.22.

```
1  INSERT INTO FAK1(SELECT K_FAKULTAS, CONTENT FROM FAKULTAS WHERE K_FAKULTAS='02');
2
3  INSERT INTO PROD1(
4  SELECT P.K_PRODI, P.K_FAKULTAS, P.K_JENJANG, P.CONTENT
5  FROM PROGRAM_STUDI P, FAKULTAS F, JENJANG J
6  WHERE P.K_FAKULTAS=F.K_FAKULTAS
7  AND P.K_JENJANG=J.K_JENJANG
8  AND P.K_FAKULTAS='02'
9  );
10
11 INSERT INTO MHS1 (
12 SELECT
13   MAHASISWA.NIM,
14   MAHASISWA.NAMA,
15   MAHASISWA.ANGKATAN,
16   MAHASISWA.K_FAKULTAS,
17   MAHASISWA.K_PRODI,
18   MAHASISWA.K_JENJANG,
19   MAHASISWA.K_SELEKSI,
20   MAHASISWA.JENIS_KELAMIN,
21   MAHASISWA.K_STATUS
22   FROM MAHASISWA, SELEKSI, STATUS, PROGRAM_STUDI
23   WHERE MAHASISWA.K_SELEKSI=SELEKSI.K_SELEKSI
24   AND MAHASISWA.K_STATUS=STATUS.K_STATUS
25   AND MAHASISWA.K_PRODI=PROGRAM_STUDI.K_PRODI
26   AND MAHASISWA.K_FAKULTAS=PROGRAM_STUDI.K_FAKULTAS
27   AND MAHASISWA.K_JENJANG=PROGRAM_STUDI.K_JENJANG
28   AND PROGRAM_STUDI.K_FAKULTAS ='02'
29   );
30 INSERT INTO KEU1(
31 SELECT NIM, TAHUN, IS_GANJIL, K_TRANSAKSI, K_ITEM, TGL_TRANSAKSI, DEBET, KREDIT
32 FROM KEUANGAN
33 WHERE NIM IN (SELECT NIM FROM MHS1));
```

Gambar 5.22 *Source* Fragmentasi Data *Site 1*

Berikut ini adalah penjelasan perintah-perintah pada gambar 5.22:

Baris 1 : merupakan perintah memasukkan data pada tabel *nick name* FAK1 melalui proses *projection* dan *selection* pada tabel FAKULTAS yang pada atribut k_fakultas memiliki nilai '02'

Baris 3-9 : merupakan perintah memasukkan data pada tabel *nick name* PROD1 melalui proses *projection* pada tabel PROGRAM_STUDI, FAKULTAS dan JENJANG yang saling berelasi selanjutnya *selection* pada atribut k_fakultas dan k_jenjang dimana atribut k_fakultas pada tabel PROGRAM_STUDI memiliki nilai '02'.

Baris 11-29 : merupakan perintah memasukkan data pada tabel *nick name* MHS1 melalui proses *projection* pada tabel MAHASISWA, SELEKSI, STATUS, PROGRAM_STUDI, FAKULTAS dan JENJANG yang saling berelasi selanjutnya *selection* pada atribut k_fakultas, k_prodi, k_jenjang, k_status dan k_seleksi dimana atribut k_fakultas pada tabel PROGRAM_STUDI memiliki nilai '02'.

Baris 30-33 : merupakan perintah memasukkan data pada tabel *nick name* KEU1 melalui proses *projection* pada tabel MAHASISWA dan MHS1. Kemudian proses *selection* dilakukan pada atribut nim pada tabel MAHASISWA dengan nilai dari *projection sub query* yang menampilkan nim dari tabel MHS1.

5.7.2 Fragmentasi *Site*2

Fragmentasi data dilakukan secara horizontal. Pada fragmentasi data yang akan diletakkan pada *site*2 mengacu pada data fakultas, program studi dan mahasiswa yang terdapat pada fakultas ilmu administrasi. Dalam proses memasukkan data pada *site*2 menggunakan pemecahan data dengan data fakultas ilmu administrasi yang memiliki kode fakultas '03'. Proses fragmentasi data yang dilakukan pada *site* utama ditunjukkan oleh gambar 5.23.

```
1  INSERT INTO FAK2(SELECT K_FAKULTAS, CONTENT FROM FAKULTAS WHERE K_FAKULTAS='03');
2
3  INSERT INTO PROD2(
4  SELECT P.K_PRODI, P.K_FAKULTAS, P.K_JENJANG, P.CONTENT
5  FROM PROGRAM_STUDI P, FAKULTAS F, JENJANG J
6  WHERE P.K_FAKULTAS=F.K_FAKULTAS
7  AND P.K_JENJANG=J.K_JENJANG
8  AND P.K_FAKULTAS='03'
9  );
10
11 INSERT INTO MHS2 (
12 SELECT
13  MAHASISWA.NIM,
14  MAHASISWA.NAMA,
15  MAHASISWA.ANGKATAN,
16  MAHASISWA.K_FAKULTAS,
17  MAHASISWA.K_PRODI,
18  MAHASISWA.K_JENJANG ,
19  MAHASISWA.K_SELEKSI ,
20  MAHASISWA.JENIS_KELAMIN ,
21  MAHASISWA.K_STATUS
22  FROM MAHASISWA, SELEKSI, STATUS, PROGRAM_STUDI
23  WHERE MAHASISWA.K_SELEKSI=SELEKSI.K_SELEKSI
24  AND MAHASISWA.K_STATUS=STATUS.K_STATUS
25  AND MAHASISWA.K_PRODI=PROGRAM_STUDI.K_PRODI
26  AND MAHASISWA.K_FAKULTAS=PROGRAM_STUDI.K_FAKULTAS
27  AND MAHASISWA.K_JENJANG=PROGRAM_STUDI.K_JENJANG
28  AND PROGRAM_STUDI.K_FAKULTAS ='03'
29  );
30 NSERT INTO KEU2(
31 SELECT NIM, TAHUN, IS_GANJIL, K_TRANSAKSI, K_ITEM, TGL_TRANSAKSI, DEBET, KREDIT
32 FROM KEUANGAN
33 WHERE NIM IN (SELECT NIM FROM MHS2));
```

Gambar 5.23 *Source* Fragmentasi Data *Site 2*

Berikut ini adalah penjelasan perintah-perintah pada gambar 5.24:

Baris 1 : merupakan perintah memasukkan data pada tabel *nick name* FAK2 melalui proses *projection* dan *selection* pada tabel FAKULTAS yang pada atribut k_fakultas memiliki nilai ‘02’

Baris 3-9 : merupakan perintah memasukkan data pada tabel *nick name* PROD2 melalui proses *projection* pada tabel PROGRAM_STUDI, FAKULTAS dan JENJANG yang saling berelasi selanjutnya *selection* pada atribut k_fakultas dan k_jenjang dimana atribut k_fakultas pada tabel PROGRAM_STUDI memiliki nilai ‘03’.

Baris 11-29 : merupakan perintah memasukkan data pada tabel *nick name* MHS2 melalui proses *projection* pada tabel MAHASISWA, SELEKSI, STATUS, PROGRAM_STUDI, FAKULTAS dan JENJANG yang saling berelasi selanjutnya *selection* pada atribut k_fakultas, k_prodi, k_jenjang, k_status dan k_seleksi

dimana atribut k_fakultas pada tabel PROGRAM_STUDI memiliki nilai ‘03’.

Baris 30-33 : merupakan perintah memasukkan data pada tabel *nick name* KEU2 melalui proses *projection* pada tabel MAHASISWA dan MHS2. Kemudian proses *selection* dilakukan pada atribut nim pada tabel MAHASISWA dengan nilai dari *projection sub query* yang menampilkan nim dari tabel MHS2.

5.7.3 Fragmentasi *Site*3

Fragmentasi data dilakukan secara horizontal. Pada fragmentasi data yang akan diletakkan pada *site*3 mengacu pada data fakultas, program studi dan mahasiswa yang terdapat pada fakultas perikakan. Dalam proses memasukkan data pada *site*3 menggunakan pemecahan data dengan data fakultas perikanan yang memiliki kode fakultas ‘09’. Proses fragmentasi data yang dilakukan pada *site* utama ditunjukkan oleh gambar 5.24.

```
1  INSERT INTO FAK3(SELECT K_FAKULTAS, CONTENT FROM FAKULTAS WHERE K_FAKULTAS='09');
2
3  INSERT INTO PROD3(
4  SELECT P.K_PRODI, P.K_FAKULTAS, P.K_JENJANG, P.CONTENT
5  FROM PROGRAM_STUDI P, FAKULTAS F, JENJANG J
6  WHERE P.K_FAKULTAS=F.K_FAKULTAS
7  AND P.K_JENJANG=J.K_JENJANG
8  AND P.K_FAKULTAS='09'
9  );
10
11 INSERT INTO MHS3(
12 SELECT
13   MAHASISWA.NIM,
14   MAHASISWA.NAMA,
15   MAHASISWA.ANGKATAN,
16   MAHASISWA.K_FAKULTAS,
17   MAHASISWA.K_PRODI,
18   MAHASISWA.K_JENJANG,
19   MAHASISWA.K_SELEKSI,
20   MAHASISWA.JENIS_KELAMIN,
21   MAHASISWA.K_STATUS
22   FROM MAHASISWA, SELEKSI, STATUS, PROGRAM_STUDI
23   WHERE MAHASISWA.K_SELEKSI=SELEKSI.K_SELEKSI
24   AND MAHASISWA.K_STATUS=STATUS.K_STATUS
25   AND MAHASISWA.K_PRODI=PROGRAM_STUDI.K_PRODI
26   AND MAHASISWA.K_FAKULTAS=PROGRAM_STUDI.K_FAKULTAS
27   AND MAHASISWA.K_JENJANG=PROGRAM_STUDI.K_JENJANG
28   AND PROGRAM_STUDI.K_FAKULTAS ='09'
29   );
30 INSERT INTO KEU3(
31 SELECT NIM, TAHUN, IS_GANJIL, K_TRANSAKSI, K_ITEM, TGL_TRANSAKSI, DEBET, KREDIT
32 FROM KEUANGAN
33 WHERE NIM IN (SELECT NIM FROM MHS3));
```

Gambar 5.24 *Source* Fragmentasi Data *Site 3*

Berikut ini adalah penjelasan perintah-perintah pada gambar 5.24:

Baris 1 : merupakan perintah memasukkan data pada tabel *nick name* FAK3 melalui proses *projection* dan *selection*

pada tabel FAKULTAS yang pada atribut k_fakultas memiliki nilai ‘09’

Baris 3-9 : merupakan perintah memasukkan data pada tabel *nick name* PROD3 melalui proses *projection* pada tabel PROGRAM_STUDI, FAKULTAS dan JENJANG yang saling berelasi selanjutnya *selection* pada atribut k_fakultas dan k_jenjang dimana atribut k_fakultas pada tabel PROGRAM_STUDI memiliki nilai ‘09’.

Baris 11-29 : merupakan perintah memasukkan data pada tabel *nick name* MHS3 melalui proses *projection* pada tabel MAHASISWA, SELEKSI, STATUS, PROGRAM_STUDI, FAKULTAS dan JENJANG yang saling berelasi selanjutnya *selection* pada atribut k_fakultas, k_prodi, k_jenjang, k_status dan k_seleksi dimana atribut k_fakultas pada tabel PROGRAM_STUDI memiliki nilai ‘09’.

Baris 30-33 : merupakan perintah memasukkan data pada tabel *nick name* KEU3 melalui proses *projection* pada tabel MAHASISWA dan MHS3. Kemudian proses *selection* dilakukan pada atribut nim pada tabel MAHASISWA dengan nilai dari *projection sub query* yang menampilkan nim dari tabel MHS3.

5.7.4 Fragmentasi *Site*4

Fragmentasi data dilakukan secara horizontal. Pada fragmentasi data yang akan diletakkan pada *site*4 mengacu pada data fakultas, program studi dan mahasiswa yang terdapat pada fakultas peternakan. Dalam proses memasukkan data pada *site*4 menggunakan pemecahan data dengan data fakultas peternakan yang memiliki kode fakultas ‘10’. Proses fragmentasi data yang dilakukan pada *site* utama ditunjukkan oleh gambar 5.25.

```
1  INSERT INTO FAK4(SELECT K_FAKULTAS, CONTENT FROM FAKULTAS WHERE K_FAKULTAS='10');
2
3  INSERT INTO PROD4(
4  SELECT P.K_PRODI, P.K_FAKULTAS, P.K_JENJANG, P.CONTENT
5  FROM PROGRAM_STUDI P, FAKULTAS F, JENJANG J
6  WHERE P.K_FAKULTAS=F.K_FAKULTAS
7  AND P.K_JENJANG=J.K_JENJANG
8  AND P.K_FAKULTAS='10'
9  );
10
11 INSERT INTO MHS4 (
12 SELECT
13   MAHASISWA.NIM,
14   MAHASISWA.NAMA,
15   MAHASISWA.ANGKATAN,
16   MAHASISWA.K_FAKULTAS,
17   MAHASISWA.K_PRODI,
18   MAHASISWA.K_JENJANG,
19   MAHASISWA.K_SELEKSI,
20   MAHASISWA.JENIS_KELAMIN,
21   MAHASISWA.K_STATUS
22   FROM MAHASISWA, SELEKSI, STATUS, PROGRAM_STUDI
23   WHERE MAHASISWA.K_SELEKSI=SELEKSI.K_SELEKSI
24   AND MAHASISWA.K_STATUS=STATUS.K_STATUS
25   AND MAHASISWA.K_PRODI=PROGRAM_STUDI.K_PRODI
26   AND MAHASISWA.K_FAKULTAS=PROGRAM_STUDI.K_FAKULTAS
27   AND MAHASISWA.K_JENJANG=PROGRAM_STUDI.K_JENJANG
28   AND PROGRAM_STUDI.K_FAKULTAS ='10'
29   );
30 INSERT INTO KEU4(
31 SELECT NIM, TAHUN, IS_GANJIL, K_TRANSAKSI, K_ITEM, TGL_TRANSAKSI, DEBET, KREDIT
32 FROM KEUANGAN
33 WHERE NIM IN (SELECT NIM FROM MHS4));
```

Gambar 5.25 *Source* Fragmentasi Data *Site 4*

Berikut ini adalah penjelasan perintah-perintah pada gambar 5.25:

Baris 1 : merupakan perintah memasukkan data pada tabel *nick name* FAK4 melalui proses *projection* dan *selection* pada tabel FAKULTAS yang pada atribut k_fakultas memiliki nilai ‘10’

Baris 3-9 : merupakan perintah memasukkan data pada tabel *nick name* PROD4 melalui proses *projection* pada tabel PROGRAM_STUDI, FAKULTAS dan JENJANG yang saling berelasi selanjutnya *selection* pada atribut k_fakultas dan k_jenjang dimana atribut k_fakultas pada tabel PROGRAM_STUDI memiliki nilai ‘10’.

Baris 11-29 : merupakan perintah memasukkan data pada tabel *nick name* MHS4 melalui proses *projection* pada tabel MAHASISWA, SELEKSI, STATUS, PROGRAM_STUDI, FAKULTAS dan JENJANG yang saling berelasi selanjutnya *selection* pada atribut k_fakultas, k_prodi, k_jenjang, k_status dan k_seleksi

dimana atribut k_fakultas pada tabel PROGRAM_STUDI memiliki nilai '10'.

Baris 30-34 : merupakan perintah memasukkan data pada tabel *nick name* KEU4 melalui proses *projection* pada tabel MAHASISWA dan MHS4. Kemudian proses *selection* dilakukan pada atribut nim pada tabel MAHASISWA dengan nilai dari *projection sub query* yang menampilkan nim dari tabel MHS4.

5.7.5 Fragmentasi *Site*5

Fragmentasi data dilakukan secara horizontal. Pada fragmentasi data yang akan diletakkan pada *site*5 mengacu pada data fakultas, program studi dan mahasiswa yang terdapat pada fakultas teknik. Dalam proses memasukkan data pada *site*5 menggunakan pemecahan data dengan data fakultas teknik yang memiliki kode fakultas '11'. Proses fragmentasi data yang dilakukan pada *site* utama ditunjukkan oleh gambar 5.26.

```
1  INSERT INTO FAK5(SELECT K_FAKULTAS, CONTENT FROM FAKULTAS WHERE K_FAKULTAS='11');
2
3  INSERT INTO PROD5(
4  SELECT P.K_PRODI , P.K_FAKULTAS , P.K_JENJANG, P.CONTENT
5  FROM PROGRAM_STUDI P, FAKULTAS F, JENJANG J
6  WHERE P.K_FAKULTAS=F.K_FAKULTAS
7  AND P.K_JENJANG=J.K_JENJANG
8  AND P.K_FAKULTAS='11'
9  );
10
11 INSERT INTO MHS5 (
12 SELECT
13  MAHASISWA.NIM,
14  MAHASISWA.NAMA,
15  MAHASISWA.ANGKATAN,
16  MAHASISWA.K_FAKULTAS ,
17  MAHASISWA.K_PRODI ,
18  MAHASISWA.K_JENJANG ,
19  MAHASISWA.K_SELEKSI ,
20  MAHASISWA.JENIS_KELAMIN ,
21  MAHASISWA.K_STATUS
22  FROM MAHASISWA, SELEKSI, STATUS, PROGRAM_STUDI
23  WHERE MAHASISWA.K_SELEKSI=SELEKSI.K_SELEKSI
24  AND MAHASISWA.K_STATUS=STATUS.K_STATUS
25  AND MAHASISWA.K_PRODI=PROGRAM_STUDI.K_PRODI
26  AND MAHASISWA.K_FAKULTAS=PROGRAM_STUDI.K_FAKULTAS
27  AND MAHASISWA.K_JENJANG=PROGRAM_STUDI.K_JENJANG
28  AND PROGRAM_STUDI.K_FAKULTAS ='11'
29  );
30 INSERT INTO KEU5(
31 SELECT NIM, TAHUN, IS_GANJIL, K_TRANSAKSI, K_ITEM, TGL_TRANSAKSI, DEBET, KREDIT
32 FROM KEUANGAN
33 WHERE NIM IN (SELECT NIM FROM MHS5));
```

Gambar 5.26 *Source* Fragmentasi Data *Site 5*

Berikut ini adalah penjelasan perintah-perintah pada gambar 5.26:

Baris 1 : merupakan perintah memasukkan data pada tabel *nick name* FAK5 melalui proses *projection* dan *selection*

pada tabel FAKULTAS yang pada atribut k_fakultas memiliki nilai ‘11’

Baris 3-9 : merupakan perintah memasukkan data pada tabel *nick name* PROD5 melalui proses *projection* pada tabel PROGRAM_STUDI, FAKULTAS dan JENJANG yang saling berelasi selanjutnya *selection* pada atribut k_fakultas dan k_jenjang dimana atribut k_fakultas pada tabel PROGRAM_STUDI memiliki nilai ‘11’.

Baris 11-29 : merupakan perintah memasukkan data pada tabel *nick name* MHS5 melalui proses *projection* pada tabel MAHASISWA, SELEKSI, STATUS, PROGRAM_STUDI, FAKULTAS dan JENJANG yang saling berelasi selanjutnya *selection* pada atribut k_fakultas, k_prodi, k_jenjang, k_status dan k_seleksi dimana atribut k_fakultas pada tabel PROGRAM_STUDI memiliki nilai ‘11’.

Baris 30-34 : merupakan perintah memasukkan data pada tabel *nick name* KEU5 melalui proses *projection* pada tabel MAHASISWA dan MHS5. Kemudian proses *selection* dilakukan pada atribut nim pada tabel MAHASISWA dengan nilai dari *projection sub query* yang menampilkan nim dari tabel MHS5.

5.7.6 Fragmentasi *Site*6

Fragmentasi data dilakukan secara horizontal. Pada fragmentasi data yang akan diletakkan pada *site*6 mengacu pada data fakultas, program studi dan mahasiswa yang terdapat pada fakultas mipa. Dalam proses memasukkan data pada *site*6 menggunakan pemecahan data dengan data fakultas mipa yang memiliki kode fakultas ‘07’. Proses fragmentasi data yang dilakukan pada *site* utama ditunjukkan oleh gambar 5.27.

```
1  INSERT INTO FAK6(SELECT K_FAKULTAS, CONTENT FROM FAKULTAS WHERE K_FAKULTAS='07');
2
3  INSERT INTO PROD6(
4  SELECT P.K_PRODI, P.K_FAKULTAS, P.K_JENJANG, P.CONTENT
5  FROM PROGRAM_STUDI P, FAKULTAS F, JENJANG J
6  WHERE P.K_FAKULTAS=F.K_FAKULTAS
7  AND P.K_JENJANG=J.K_JENJANG
8  AND P.K_FAKULTAS='07'
9  );
10
11 INSERT INTO MHS6 (
12 SELECT
13  MAHASISWA.NIM,
14  MAHASISWA.NAMA,
15  MAHASISWA.ANGKATAN,
16  MAHASISWA.K_FAKULTAS,
17  MAHASISWA.K_PRODI,
18  MAHASISWA.K_JENJANG ,
19  MAHASISWA.K_SELEKSI ,
20  MAHASISWA.JENIS_KELAMIN ,
21  MAHASISWA.K_STATUS
22  FROM MAHASISWA, SELEKSI, STATUS, PROGRAM_STUDI
23  WHERE MAHASISWA.K_SELEKSI=SELEKSI.K_SELEKSI
24  AND MAHASISWA.K_STATUS=STATUS.K_STATUS
25  AND MAHASISWA.K_PRODI=PROGRAM_STUDI.K_PRODI
26  AND MAHASISWA.K_FAKULTAS=PROGRAM_STUDI.K_FAKULTAS
27  AND MAHASISWA.K_JENJANG=PROGRAM_STUDI.K_JENJANG
28  AND PROGRAM_STUDI.K_FAKULTAS ='07'
29  );
30 INSERT INTO KEU6(
31 SELECT NIM, TAHUN, IS_GANJIL, K_TRANSAKSI, K_ITEM, TGL_TRANSAKSI, DEBET, KREDIT
32 FROM KEUANGAN
33 WHERE NIM IN (SELECT NIM FROM MHS6));
```

Gambar 5.27 *Source* Fragmentasi Data *Site 6*

Berikut ini adalah penjelasan perintah-perintah pada gambar 5.27:

Baris 1 : merupakan perintah memasukkan data pada tabel *nick name* FAK6 melalui proses *projection* dan *selection* pada tabel FAKULTAS yang pada atribut k_fakultas memiliki nilai ‘07’

Baris 3-9 : merupakan perintah memasukkan data pada tabel *nick name* PROD6 melalui proses *projection* pada tabel PROGRAM_STUDI, FAKULTAS dan JENJANG yang saling berelasi selanjutnya *selection* pada atribut k_fakultas dan k_jenjang dimana atribut k_fakultas pada tabel PROGRAM_STUDI memiliki nilai ‘07’.

Baris 11-29 : merupakan perintah memasukkan data pada tabel *nick name* MHS6 melalui proses *projection* pada tabel MAHASISWA, SELEKSI, STATUS, PROGRAM_STUDI, FAKULTAS dan JENJANG yang saling berelasi selanjutnya *selection* pada atribut k_fakultas, k_prodi, k_jenjang, k_status dan k_seleksi

dimana atribut k_fakultas pada tabel PROGRAM_STUDI memiliki nilai ‘07’.

Baris 30-34 : merupakan perintah memasukkan data pada tabel *nick name* KEU6 melalui proses *projection* pada tabel MAHASISWA dan MHS6. Kemudian proses *selection* dilakukan pada atribut nim pada tabel MAHASISWA dengan nilai dari *projection sub query* yang menampilkan nim dari tabel MHS6.

5.7.7 Fragmentasi *Site*7

Fragmentasi data dilakukan secara horizontal. Pada fragmentasi data yang akan diletakkan pada *site*7 mengacu pada data fakultas, program studi dan mahasiswa yang terdapat pada fakultas ilmu sosial dan politik. Dalam proses memasukkan data pada *site*7 menggunakan pemecahan data dengan data fakultas ilmu sosial dan politik yang memiliki kode fakultas ‘05’. Proses fragmentasi data yang dilakukan pada *site* utama ditunjukkan oleh gambar 5.28.

```
1  INSERT INTO FAK7(SELECT K_FAKULTAS, CONTENT FROM FAKULTAS WHERE K_FAKULTAS='05');
2
3  INSERT INTO PROD7(
4  SELECT P.K_PRODI, P.K_FAKULTAS, P.K_JENJANG, P.CONTENT
5  FROM PROGRAM_STUDI P, FAKULTAS F, JENJANG J
6  WHERE P.K_FAKULTAS=F.K_FAKULTAS
7  AND P.K_JENJANG=J.K_JENJANG
8  AND P.K_FAKULTAS='05'
9  );
10
11 INSERT INTO MHS7 (
12 SELECT
13   MAHASISWA.NIM,
14   MAHASISWA.NAMA,
15   MAHASISWA.ANGKATAN,
16   MAHASISWA.K_FAKULTAS,
17   MAHASISWA.K_PRODI,
18   MAHASISWA.K_JENJANG,
19   MAHASISWA.K_SELEKSI,
20   MAHASISWA.JENIS_KELAMIN,
21   MAHASISWA.K_STATUS
22   FROM MAHASISWA, SELEKSI, STATUS, PROGRAM_STUDI
23   WHERE MAHASISWA.K_SELEKSI=SELEKSI.K_SELEKSI
24   AND MAHASISWA.K_STATUS=STATUS.K_STATUS
25   AND MAHASISWA.K_PRODI=PROGRAM_STUDI.K_PRODI
26   AND MAHASISWA.K_FAKULTAS=PROGRAM_STUDI.K_FAKULTAS
27   AND MAHASISWA.K_JENJANG=PROGRAM_STUDI.K_JENJANG
28   AND PROGRAM_STUDI.K_FAKULTAS ='05'
29   );
30 INSERT INTO KEU7(
31 SELECT NIM, TAHUN, IS_GANJIL, K_TRANSAKSI, K_ITEM, TGL_TRANSAKSI, DEBET, KREDIT
32 FROM KEUANGAN
33 WHERE NIM IN (SELECT NIM FROM MHS7));
```

Gambar 5.28 *Source* Fragmentasi Data *Site 7*

Berikut ini adalah penjelasan perintah-perintah pada gambar 5.28:

Baris 1 : merupakan perintah memasukkan data pada tabel *nick name* FAK7 melalui proses *projection* dan *selection* pada tabel FAKULTAS yang pada atribut k_fakultas memiliki nilai '05'

Baris 3-9 : merupakan perintah memasukkan data pada tabel *nick name* PROD7 melalui proses *projection* pada tabel PROGRAM_STUDI, FAKULTAS dan JENJANG yang saling berelasi selanjutnya *selection* pada atribut k_fakultas dan k_jenjang dimana atribut k_fakultas pada tabel PROGRAM_STUDI memiliki nilai '05'.

Baris 11-29 : merupakan perintah memasukkan data pada tabel *nick name* MHS7 melalui proses *projection* pada tabel MAHASISWA, SELEKSI, STATUS, PROGRAM_STUDI, FAKULTAS dan JENJANG yang saling berelasi selanjutnya *selection* pada atribut k_fakultas, k_prodi, k_jenjang, k_status dan k_seleksi dimana atribut k_fakultas pada tabel PROGRAM_STUDI memiliki nilai '05'.

Baris 30-34 : merupakan perintah memasukkan data pada tabel *nick name* KEU7 melalui proses *projection* pada tabel MAHASISWA dan MHS7. Kemudian proses *selection* dilakukan pada atribut nim pada tabel MAHASISWA dengan nilai dari *projection sub query* yang menampilkan nim dari tabel MHS7.

5.7.8 Data *Site* Utama

Data pada *site* utama merupakan data dari FAKULTAS, PROGRAM_STUDI, MAHASISWA dan KEUANGAN yang merupakan data selain yang terdapat pada *nick name* yang terbentuk. Setelah dilakukan fragmentasi data, maka selanjutnya dilakukan penghapusan data agar tidak terjadi penumpukan data. pada data utama dilakukan

penghapusan *record* data yang sama pada setiap *nick name* yang terbentuk. Proses fragmentasi atau penghapusan data yang dilakukan pada *site* utama ditunjukkan oleh gambar 5.29.

```
1   DELETE FROM KEUANGAN
2   WHERE NIM IN (
3     SELECT NIM FROM MHS1
4     UNION
5     SELECT NIM FROM MHS2
6     UNION
7     SELECT NIM FROM MHS3
8     UNION
9     SELECT NIM FROM MHS4
10    UNION
11    SELECT NIM FROM MHS5
12    UNION
13    SELECT NIM FROM MHS6
14    UNION
15    SELECT NIM FROM MHS7
16  )
17
18  DELETE FROM MAHASISWA
19  WHERE NIM IN (
20    SELECT NIM FROM MHS1
21    UNION
22    SELECT NIM FROM MHS2
23    UNION
24    SELECT NIM FROM MHS3
25    UNION
26    SELECT NIM FROM MHS4
27    UNION
28    SELECT NIM FROM MHS5
29    UNION
30    SELECT NIM FROM MHS6
31    UNION
32    SELECT NIM FROM MHS7
33  )
34
35  DELETE FROM PROGRAM_STUDI
36  WHERE K_PRODI IN (
37    SELECT K_PRODI FROM PROD1
38    UNION
39    SELECT K_PRODI FROM PROD2
40    UNION
41    SELECT K_PRODI FROM PROD3
42    UNION
43    SELECT K_PRODI FROM PROD4
44    UNION
45    SELECT K_PRODI FROM PROD5
46    UNION
47    SELECT K_PRODI FROM PROD6
48
49    UNION
50    SELECT K_PRODI FROM PROD7
51  )
52
53  DELETE FROM FAKULTAS
54  WHERE K_FAKULTAS IN(
55    SELECT K_FAKULTAS FROM FAK1
56    UNION
57    SELECT K_FAKULTAS FROM FAK2
58    UNION
59    SELECT K_FAKULTAS FROM FAK3
60    UNION
61    SELECT K_FAKULTAS FROM FAK4
62    UNION
63    SELECT K_FAKULTAS FROM FAK5
64    UNION
65    SELECT K_FAKULTAS FROM FAK6
66    UNION
67    SELECT K_FAKULTAS FROM FAK7
68  )
```

Gambar 5.29 Hapus Data Utama

Berikut ini adalah penjelasan perintah-perintah pada gambar 5.29:

Baris 1-16 : *query* pada baris ini menuunjukkan perintah menghapus *record* data pada tabel KEUANGAN dengan kondisi NIM yang terdapat pada hasil *view* dari MHS1, MHS2, MHS3, MHS4, MHS5, MHS6 dan MHS7.

Baris 18-33 : *query* pada baris ini menuunjukkan perintah menghapus *record* data pada tabel MAHASISWA dengan kondisi NIM yang terdapat pada hasil *view* dari MHS1, MHS2, MHS3, MHS4, MHS5, MHS6 dan MHS7.

Baris 35-51 : *query* pada baris ini menuunjukkan perintah menghapus *record* data pada tabel PROGRAM_STUDI dengan kondisi K_PRODI yang terdapat pada hasil

view dari PROD1, PROD2, PROD3, PROD4, PROD5, PROD6, dan PROD7.

Baris 53-68 : *query* pada baris ini menuunjukkan perintah menghapus *record* data pada tabel FAKULTAS dengan kondisi K_FAKULTAS yang terdapat pada hasil *view* dari FAK1, FAK2, FAK3, FAK4, FAK5, FAK6 dan FAK7.

5.8 Implementasi Model Akses

Pada implementasi model akses, penulis mendesain dua bentuk model akses pada dua kasus yang penulis angkat. Model akses pertama merupakan model akses yang menggunakan *query* konvensional sedangkan untuk model akses kedua memanfaatkan *materialized query table* yang disediakan oleh DBMS. Sehingga pada penelitian ini, penulis membentuk empat *view* dengan dua model *view* dari pengujian yang dilakukan.

5.8.1 *Query* 1 model 1

Pada *query* 1 model 1, model *view* yang terbentuk terdiri dari data jumlah mahasiswa yang digolongkan berdasar angkatan, fakultas, program studi, jenjang, dan seleksi. Pada pembentukan *view*, terdapat beberapa tabel yang perlu di akses oleh penggunaak *query* model ini. Tabel yang perlu di akses antara lain tabel fakultas, jenjang, program studi, mahasiswa, seleksi dan status. Bentuk *query* model 1 ditunjukkan seperti pada gambar 5.30 dan 5.31.

```
1  SELECT M.ANGKATAN,F.CONTENT FAKULTAS,P.CONTENT AS PRODI,J.CONTENT AS
2  JENJANG,S.CONTENT AS SELEKSI, COUNT(M.NIM) AS JUMLAH
3  FROM FAKULTAS F, MAHASISWA M, PROGRAM_STUDI P, JENJANG J, SELEKSI S
4  WHERE P.K_PRODI=M.K_PRODI
5  AND F.K_FAKULTAS=P.K_FAKULTAS
6  AND J.K_JENJANG=P.K_JENJANG
7  AND S.K_SELEKSI=M.K_SELEKSI
8  GROUP BY M.ANGKATAN,F.CONTENT , P.CONTENT, J.CONTENT, S.CONTENT
9
10 UNION
11 SELECT M.ANGKATAN,F.CONTENT FAKULTAS,P.CONTENT AS PRODI,J.CONTENT AS
12 JENJANG,S.CONTENT AS SELEKSI, COUNT(M.NIM) AS JUMLAH
13 FROM FAK1 F, MHS1 M, PROD1 P, JENJANG J, SELEKSI S
14 WHERE P.K_PRODI=M.K_PRODI
15 AND F.K_FAKULTAS=P.K_FAKULTAS
16 AND J.K_JENJANG=P.K_JENJANG
17 AND S.K_SELEKSI=M.K_SELEKSI
18 GROUP BY M.ANGKATAN,F.CONTENT , P.CONTENT, J.CONTENT, S.CONTENT
19
20 UNION
21 SELECT M.ANGKATAN,F.CONTENT FAKULTAS,P.CONTENT AS PRODI,J.CONTENT AS
22 JENJANG,S.CONTENT AS SELEKSI, COUNT(M.NIM) AS JUMLAH
23 FROM FAK2 F, MHS2 M, PROD2 P, JENJANG J, SELEKSI S
24 WHERE P.K_PRODI=M.K_PRODI
25 AND F.K_FAKULTAS=P.K_FAKULTAS
26 AND J.K_JENJANG=P.K_JENJANG
27 AND S.K_SELEKSI=M.K_SELEKSI
28 GROUP BY M.ANGKATAN,F.CONTENT , P.CONTENT, J.CONTENT, S.CONTENT
29
30 UNION
31 SELECT M.ANGKATAN,F.CONTENT FAKULTAS,P.CONTENT AS PRODI,J.CONTENT AS
32 JENJANG,S.CONTENT AS SELEKSI, COUNT(M.NIM) AS JUMLAH
33 FROM FAK3 F, MHS3 M, PROD3 P, JENJANG J, SELEKSI S
34 WHERE P.K_PRODI=M.K_PRODI
35 AND F.K_FAKULTAS=P.K_FAKULTAS
36 AND J.K_JENJANG=P.K_JENJANG
37 AND S.K_SELEKSI=M.K_SELEKSI
38 GROUP BY M.ANGKATAN,F.CONTENT , P.CONTENT, J.CONTENT, S.CONTENT
39
40 UNION
41 SELECT M.ANGKATAN,F.CONTENT FAKULTAS,P.CONTENT AS PRODI,J.CONTENT AS
42 JENJANG,S.CONTENT AS SELEKSI, COUNT(M.NIM) AS JUMLAH
43 FROM FAK4 F, MHS4 M, PROD4 P, JENJANG J, SELEKSI S
44 WHERE P.K_PRODI=M.K_PRODI
45 AND F.K_FAKULTAS=P.K_FAKULTAS
46 AND J.K_JENJANG=P.K_JENJANG
47 AND S.K_SELEKSI=M.K_SELEKSI
48 GROUP BY M.ANGKATAN,F.CONTENT , P.CONTENT, J.CONTENT, S.CONTENT
```

Gambar 5.30 *Query* 1 Model 1 Bagian 1

```
49 UNION
50 SELECT M.ANGKATAN,F.CONTENT FAKULTAS,P.CONTENT AS PRODI,J.CONTENT AS
51 JENJANG,S.CONTENT AS SELEKSI, COUNT(M.NIM) AS JUMLAH
52 FROM FAK5 F, MHS5 M, PROD5 P, JENJANG J, SELEKSI S
53 WHERE P.K_PRODI=M.K_PRODI
54 AND F.K_FAKULTAS=P.K_FAKULTAS
55 AND J.K_JENJANG=P.K_JENJANG
56 AND S.K_SELEKSI=M.K_SELEKSI
57 GROUP BY M.ANGKATAN,F.CONTENT , P.CONTENT, J.CONTENT, S.CONTENT
58
59 UNION
60 SELECT M.ANGKATAN,F.CONTENT FAKULTAS,P.CONTENT AS PRODI,J.CONTENT AS
61 JENJANG,S.CONTENT AS SELEKSI, COUNT(M.NIM) AS JUMLAH
62 FROM FAK6 F, MHS6 M, PROD6 P, JENJANG J, SELEKSI S
63 WHERE P.K_PRODI=M.K_PRODI
64 AND F.K_FAKULTAS=P.K_FAKULTAS
65 AND J.K_JENJANG=P.K_JENJANG
66 AND S.K_SELEKSI=M.K_SELEKSI
67 GROUP BY M.ANGKATAN,F.CONTENT , P.CONTENT, J.CONTENT, S.CONTENT
68
69 UNION
70 SELECT M.ANGKATAN,F.CONTENT FAKULTAS,P.CONTENT AS PRODI,J.CONTENT AS
71 JENJANG,S.CONTENT AS SELEKSI, COUNT(M.NIM) AS JUMLAH
72 FROM FAK7 F, MHS7 M, PROD7 P, JENJANG J, SELEKSI S
73 WHERE P.K_PRODI=M.K_PRODI
74 AND F.K_FAKULTAS=P.K_FAKULTAS
75 AND J.K_JENJANG=P.K_JENJANG
76 AND S.K_SELEKSI=M.K_SELEKSI
77 GROUP BY M.ANGKATAN,F.CONTENT , P.CONTENT, J.CONTENT, S.CONTENT
```

Gambar 5.31 *Query* 1 Model 1 Bagian 2

Berikut ini adalah penjelasan dari gambar 5.30 dan gambar 5.31:

Baris 1-8 : merupakan baris *query* untuk membuat LCS dengan isi *view* yang terdiri dari angkatan, fakultas, prodi, jenjang, seleksi dan jumlah mahasiswa yang mengakses tabel FAKULTAS, MAHASISWA, PROGRAM_STUDI, JENJANG dan SELEKSI dengan proses *join* dari relasi yang ada antar tabel. Dalam *query* tersebut juga dilakukan proses pengelompokan dan LCS terbentuk pada *site* pusat.

Baris 10-18 : merupakan baris *query* untuk membuat LCS dengan isi *view* yang terdiri dari angkatan, fakultas, prodi, jenjang, seleksi dan jumlah mahasiswa yang mengakses *nick name* tabel FAK1, MHS1, PRODI1, JENJANG dan SELEKSI dengan proses *join* dari relasi yang ada antar tabel. Dalam *query* tersebut juga dilakukan proses pengelompokan dan LCS terbentuk pada *site* 1.

Baris 20-28 : merupakan baris *query* untuk membuat LCS dengan isi *view* yang terdiri dari angkatan, fakultas, prodi, jenjang, seleksi dan jumlah mahasiswa yang mengakses *nick name* tabel FAK2, MHS2, PRODI2, JENJANG dan SELEKSI dengan proses *join* dari relasi yang ada antar tabel. Dalam *query* tersebut juga dilakukan proses pengelompokan dan LCS terbentuk pada *site* 2.

Baris 30-38 : merupakan baris *query* untuk membuat LCS dengan isi *view* yang terdiri dari angkatan, fakultas, prodi, jenjang, seleksi dan jumlah mahasiswa yang mengakses *nick name* tabel FAK3, MHS3, PRODI3, JENJANG dan SELEKSI dengan proses *join* dari relasi yang ada antar tabel. Dalam *query* tersebut juga dilakukan proses pengelompokan dan LCS terbentuk pada *site* 3.

Baris 40-48 : merupakan baris *query* untuk membuat LCS dengan isi *view* yang terdiri dari angkatan, fakultas, prodi, jenjang, seleksi dan jumlah mahasiswa yang mengakses *nick name* tabel FAK4, MHS4, PRODI4, JENJANG dan SELEKSI dengan proses *join* dari relasi yang ada antar tabel. Dalam *query* tersebut juga dilakukan proses pengelompokan dan LCS terbentuk pada *site* 4.

Baris 49-57 : merupakan baris *query* untuk membuat LCS dengan isi *view* yang terdiri dari angkatan, fakultas, prodi, jenjang, seleksi dan jumlah mahasiswa yang mengakses *nick name* tabel FAK5, MHS5, PRODI5, JENJANG dan SELEKSI dengan proses *join* dari relasi yang ada antar tabel. Dalam *query* tersebut juga dilakukan proses pengelompokan dan LCS terbentuk pada *site* 5.

Baris 59-67 : merupakan baris *query* untuk membuat LCS dengan isi *view* yang terdiri dari angkatan, fakultas, prodi, jenjang, seleksi dan jumlah mahasiswa yang mengakses *nick name* tabel FAK6, MHS6, PRODI6, JENJANG dan SELEKSI dengan proses *join* dari relasi yang ada antar tabel. Dalam *query* tersebut juga dilakukan proses pengelompokan dan LCS terbentuk pada *site* 6.

Baris 69-77 : merupakan baris *query* untuk membuat LCS dengan isi *view* yang terdiri dari angkatan, fakultas, prodi, jenjang, seleksi dan jumlah mahasiswa yang mengakses *nick name* tabel FAK7, MHS7, PRODI7, JENJANG dan SELEKSI dengan proses *join* dari relasi yang ada antar tabel. Dalam *query* tersebut juga dilakukan proses pengelompokan dan LCS terbentuk pada *site* 7.

5.8.2 *Query* 1 Model 2

Pada *query* 1 model 2, model *view* yang terbentuk terdiri dari data jumlah mahasiswa yang digolongkan berdasar angkatan, fakultas, program studi, jenjang, seleksi dan jumlah mahasiswa. Pada pembentukan *view*, *query* hanya melakukan akses terhadap satu tabel saja. Tabel yang perlu di akses antara lain tabel MV_JUMLAH_MAHASISWA. Bentuk *query* 1 model 2 ditunjukkan seperti pada gambar 5.33.

```
1  SELECT ANGKATAN, FAKULTAS, PRODI, JENJANG, SELEKSI, JUMLAH FROM
2  MV_JUMLAH_MAHASISWA;
```

Gambar 5.32. *Query* 1 Model 2

Berikut ini adalah penjelasan dari gambar 5.33:

Baris 1-2 : pada baris *query* ini, merupakan *query* yang digunakan untuk memanggil *materialized view* yang telah terbentuk dari proses eksekusi *query* dengan penyusunan *view* yang sama seperti pada *query* 1 model 2.

5.8.3 *Query* 2 model 1

Pada *query* 2 model 1, model *view* yang terbentuk terdiri dari data jumlah keuangan yang digolongkan berdasarkan fakultas, program studi, seleksi, dan item pembayaran. Pada pembentukan *view*, terdapat beberapa tabel yang perlu di akses oleh penggunaak *query* model ini. Tabel yang perlu di akses antara lain tabel fakultas, jenjang, program studi, mahasiswa, seleksi, item dan keuangan. Bentuk *query* model 1 ditunjukkan seperti pada gambar 5.33, 5.34, dan 5.35.

```
1   SELECT F.CONTENT FAKULTAS,P.CONTENT AS PRODI,J.CONTENT AS JENJANG,S.CONTENT AS
2   SELEKSI, I.CONTENT AS ITEM, KT.JUMLAH_DEBET, KT.JUMLAH_KREDIT
3   FROM FAKULTAS F, PROGRAM_STUDI P, JENJANG J, SELEKSI S, ITEM I,
4   (SELECT M.K_FAKULTAS, M.K_PRODI, M.K_JENJANG, M.K_SELEKSI , K.K_ITEM, SUM(K.DEBET)
5   AS JUMLAH_DEBET, SUM(K.KREDIT) AS JUMLAH_KREDIT
6   FROM KEUANGAN K, MAHASISWA M
7   WHERE K.NIM=M.NIM
8   GROUP BY M.K_FAKULTAS, M.K_PRODI, M.K_JENJANG, M.K_SELEKSI , K.K_ITEM) KT
9   WHERE KT.K_PRODI=P.K_PRODI
10  AND F.K_FAKULTAS=P.K_FAKULTAS
11  AND J.K_JENJANG=KT.K_JENJANG
12  AND S.K_SELEKSI=KT.K_SELEKSI
13
14  UNION
15  SELECT F.CONTENT FAKULTAS,P.CONTENT AS PRODI,J.CONTENT AS JENJANG,S.CONTENT AS
16  SELEKSI, I.CONTENT AS ITEM, KT.JUMLAH_DEBET, KT.JUMLAH_KREDIT
17  FROM FAK1 F, PROD1 P, JENJANG J, SELEKSI S, ITEM I,
18  (SELECT M.K_FAKULTAS, M.K_PRODI, M.K_JENJANG, M.K_SELEKSI , K.K_ITEM, SUM(K.DEBET)
19  AS JUMLAH_DEBET, SUM(K.KREDIT) AS JUMLAH_KREDIT
20  FROM KEU1 K, MHS1 M
21  WHERE K.NIM=M.NIM
22  GROUP BY M.K_FAKULTAS, M.K_PRODI, M.K_JENJANG, M.K_SELEKSI , K.K_ITEM) KT
23  WHERE KT.K_PRODI=P.K_PRODI
24  AND F.K_FAKULTAS=P.K_FAKULTAS
25  AND J.K_JENJANG=KT.K_JENJANG
26  AND S.K_SELEKSI=KT.K_SELEKSI
27
28  UNION
29  SELECT F.CONTENT FAKULTAS,P.CONTENT AS PRODI,J.CONTENT AS JENJANG,S.CONTENT AS
30  SELEKSI, I.CONTENT AS ITEM, KT.JUMLAH_DEBET, KT.JUMLAH_KREDIT
31  FROM FAK2 F, PROD2 P, JENJANG J, SELEKSI S, ITEM I,
32  (SELECT M.K_FAKULTAS, M.K_PRODI, M.K_JENJANG, M.K_SELEKSI , K.K_ITEM, SUM(K.DEBET)
33  AS JUMLAH_DEBET, SUM(K.KREDIT) AS JUMLAH_KREDIT
34  FROM KEU2 K, MHS2 M
35  WHERE K.NIM=M.NIM
36  GROUP BY M.K_FAKULTAS, M.K_PRODI, M.K_JENJANG, M.K_SELEKSI , K.K_ITEM) KT
37  WHERE KT.K_PRODI=P.K_PRODI
38  AND F.K_FAKULTAS=P.K_FAKULTAS
39  AND J.K_JENJANG=KT.K_JENJANG
40  AND S.K_SELEKSI=KT.K_SELEKSI
41
42  UNION
43  SELECT F.CONTENT FAKULTAS,P.CONTENT AS PRODI,J.CONTENT AS JENJANG,S.CONTENT AS
44  SELEKSI, I.CONTENT AS ITEM, KT.JUMLAH_DEBET, KT.JUMLAH_KREDIT
45  FROM FAK3 F, PROD3 P, JENJANG J, SELEKSI S, ITEM I,
46  (SELECT M.K_FAKULTAS, M.K_PRODI, M.K_JENJANG, M.K_SELEKSI , K.K_ITEM, SUM(K.DEBET)
47  AS JUMLAH_DEBET, SUM(K.KREDIT) AS JUMLAH_KREDIT
48  FROM KEU3 K, MHS3 M
```

Gambar 5.33 *query* 2 Model 1 Bagian 1

```
49  WHERE K.NIM=M.NIM
50  GROUP BY  M.K_FAKULTAS, M.K_PRODI, M.K_JENJANG, M.K_SELEKSI , K.K_ITEM) KT
51  WHERE KT.K_PRODI=P.K_PRODI
52  AND F.K_FAKULTAS=P.K_FAKULTAS
53  AND J.K_JENJANG=KT.K_JENJANG
54  AND S.K_SELEKSI=KT.K_SELEKSI
55
56  UNION
57  SELECT F.CONTENT FAKULTAS,P.CONTENT AS PRODI,J.CONTENT AS JENJANG,S.CONTENT AS
58  SELEKSI, I.CONTENT AS ITEM, KT.JUMLAH_DEBET, KT.JUMLAH_KREDIT
59  FROM FAK4 F, PROD4 P, JENJANG J, SELEKSI S,  ITEM I,
60  (SELECT  M.K_FAKULTAS, M.K_PRODI, M.K_JENJANG, M.K_SELEKSI , K.K_ITEM, SUM(K.DEBET)
61  AS JUMLAH_DEBET, SUM(K.KREDIT) AS JUMLAH_KREDIT
62  FROM KEU4 K, MHS4 M
63  WHERE K.NIM=M.NIM
64  GROUP BY  M.K_FAKULTAS, M.K_PRODI, M.K_JENJANG, M.K_SELEKSI , K.K_ITEM) KT
65  WHERE KT.K_PRODI=P.K_PRODI
66  AND F.K_FAKULTAS=P.K_FAKULTAS
67  AND J.K_JENJANG=KT.K_JENJANG
68  AND S.K_SELEKSI=KT.K_SELEKSI
69
70  UNION
71  SELECT F.CONTENT FAKULTAS,P.CONTENT AS PRODI,J.CONTENT AS JENJANG,S.CONTENT AS
72  SELEKSI, I.CONTENT AS ITEM, KT.JUMLAH_DEBET, KT.JUMLAH_KREDIT
73  FROM FAK5 F, PROD5 P, JENJANG J, SELEKSI S,  ITEM I,
74  (SELECT  M.K_FAKULTAS, M.K_PRODI, M.K_JENJANG, M.K_SELEKSI , K.K_ITEM, SUM(K.DEBET)
75  AS JUMLAH_DEBET, SUM(K.KREDIT) AS JUMLAH_KREDIT
76  FROM KEU5 K, MHS5 M
77  WHERE K.NIM=M.NIM
78  GROUP BY  M.K_FAKULTAS, M.K_PRODI, M.K_JENJANG, M.K_SELEKSI , K.K_ITEM) KT
79  WHERE KT.K_PRODI=P.K_PRODI
80  AND F.K_FAKULTAS=P.K_FAKULTAS
81  AND J.K_JENJANG=KT.K_JENJANG
82  AND S.K_SELEKSI=KT.K_SELEKSI
83
84  UNION
85  SELECT F.CONTENT FAKULTAS,P.CONTENT AS PRODI,J.CONTENT AS JENJANG,S.CONTENT AS
86  SELEKSI, I.CONTENT AS ITEM, KT.JUMLAH_DEBET, KT.JUMLAH_KREDIT
87  FROM FAK6 F, PROD6 P, JENJANG J, SELEKSI S,  ITEM I,
88  (SELECT  M.K_FAKULTAS, M.K_PRODI, M.K_JENJANG, M.K_SELEKSI , K.K_ITEM, SUM(K.DEBET)
89  AS JUMLAH_DEBET, SUM(K.KREDIT) AS JUMLAH_KREDIT
90  FROM KEU6 K, MHS6 M
91  WHERE K.NIM=M.NIM
92  GROUP BY  M.K_FAKULTAS, M.K_PRODI, M.K_JENJANG, M.K_SELEKSI , K.K_ITEM) KT
93  WHERE KT.K_PRODI=P.K_PRODI
94  AND F.K_FAKULTAS=P.K_FAKULTAS
95  AND J.K_JENJANG=KT.K_JENJANG
96  AND S.K_SELEKSI=KT.K_SELEKSI
```

Gambar 5.34 *query* 2 Model 1 Bagian 2

```
97  UNION
98  SELECT F.CONTENT FAKULTAS,P.CONTENT AS PRODI,J.CONTENT AS JENJANG,S.CONTENT AS
100 SELEKSI, I.CONTENT AS ITEM, KT.JUMLAH_DEBET, KT.JUMLAH_KREDIT
101 FROM FAK7 F, PROD7 P, JENJANG J, SELEKSI S,  ITEM I,
102 (SELECT  M.K_FAKULTAS, M.K_PRODI, M.K_JENJANG, M.K_SELEKSI , K.K_ITEM, SUM(K.DEBET)
103 AS JUMLAH_DEBET, SUM(K.KREDIT) AS JUMLAH_KREDIT
104 FROM KEU7 K, MHS7 M
105 WHERE K.NIM=M.NIM
106 GROUP BY  M.K_FAKULTAS, M.K_PRODI, M.K_JENJANG, M.K_SELEKSI , K.K_ITEM) KT
107 WHERE KT.K_PRODI=P.K_PRODI
108 AND F.K_FAKULTAS=P.K_FAKULTAS
109 AND J.K_JENJANG=KT.K_JENJANG
110 AND S.K_SELEKSI=KT.K_SELEKSI
```

Gambar 5.35 *query* 2 Model 1 Bagian 3

Berikut ini merupakan penjelasan dari gambar 5.33, 5.34 dan 5.35:

Baris 1-12 : merupakan pembentukan *view* yang menampilkan data atribut fakultas, prodi, jenjang, seleksi, item, jumlah debet dan jumlah kredit dengan mengakses tabel FAKULTAS, JENJANG, PROGRAM_STUDI, MAHASISWA, SELEKSI, KEUANGAN, dan ITEM. Eksekusi *query* ini dilakukan pada *site* utama sebagai LCS.

Baris 14-26 : merupakan pembentukan *view* yang menampilkan data atribut fakultas, prodi, jenjang, seleksi, item, jumlah debet dan jumlah kredit dengan mengakses tabel FAK1, JENJANG, PRODI1, MHS1, SELEKSI, KEU1, dan ITEM. Eksekusi *query* ini dilakukan pada *site*1 sebagai LCS.

Baris 28-40 : merupakan pembentukan *view* yang menampilkan data atribut fakultas, prodi, jenjang, seleksi, item, jumlah debet dan jumlah kredit dengan mengakses tabel FAK2, JENJANG, PRODI2, MHS2, SELEKSI, KEU2, dan ITEM. Eksekusi *query* ini dilakukan pada *site*2 sebagai LCS.

Baris 42-54 : merupakan pembentukan *view* yang menampilkan data atribut fakultas, prodi, jenjang, seleksi, item, jumlah debet dan jumlah kredit dengan mengakses tabel FAK3, JENJANG, PRODI3, MHS3, SELEKSI, KEU3, dan ITEM. Eksekusi *query* ini dilakukan pada *site*3 sebagai LCS.

Baris 56-68 : merupakan pembentukan *view* yang menampilkan data atribut fakultas, prodi, jenjang, seleksi, item, jumlah debet dan jumlah kredit dengan mengakses tabel FAK4, JENJANG, PRODI4, MHS4, SELEKSI, KEU4, dan

ITEM. Eksekusi *query* ini dilakukan pada *site*4 sebagai LCS.

Baris 70-82 : merupakan pembentukan *view* yang menampilkan data atribut fakultas, prodi, jenjang, seleksi, item, jumlah debet dan jumlah kredit dengan mengakses tabel FAK5, JENJANG, PRODI5, MHS5, SELEKSI, KEU5, dan ITEM. Eksekusi *query* ini dilakukan pada *site*5 sebagai LCS.

Baris 84-96 : merupakan pembentukan *view* yang menampilkan data atribut fakultas, prodi, jenjang, seleksi, item, jumlah debet dan jumlah kredit dengan mengakses tabel FAK6, JENJANG, PRODI6, MHS6, SELEKSI, KEU6, dan ITEM. Eksekusi *query* ini dilakukan pada *site*6 sebagai LCS.

Baris 97-110 : merupakan pembentukan *view* yang menampilkan data atribut fakultas, prodi, jenjang, seleksi, item, jumlah debet dan jumlah kredit dengan mengakses tabel FAK7, JENJANG, PRODI7, MHS7, SELEKSI, KEU7, dan ITEM. Eksekusi *query* ini dilakukan pada *site*7 sebagai LCS.

5.8.4 *Query* 2 Model 2

Pada *query* 2 model 2, model *view* yang terbentuk terdiri dari data jumlah mahasiswa yang digolongkan berdasar fakultas, program studi, jenjang, seleksi dan item. Pada pembentukan *view*, *query* hanya melakukan akses terhadap satu tabel saja. Tabel yang perlu di akses antara lain tabel MV_JUMLAH_KEUANGAN. Bentuk *query* 2 model 2 ditunjukkan seperti pada gambar 5.36.

```
1  SELECT FAKULTAS, PRODI, JENJANG,SELEKSI,  ITEM, JUMLAH_DEBET,JUMLAH_KREDIT
2  FROM MV_JUMLAH_KEUANGAN
```

Gambar 5.36 *Query* 2 Model 2

Berikut ini merupakan penjelasan dari gambar 5.36:

Baris 1-2 : merupakan *query* yang digunakan untuk memanggil *materialized view* yang telah dibentuk dengan isi yang sama seperti pada *query* 2 model 1.

5.9 Implementasi *Linked Server*

Pada penelitian ini, penulis mendesain sebuah *linked server* yang memanfaatkan SSH. Tujuan dari implementasi ini adalah untuk mempermudah proses pengoperasian secara manual kepada virtual *server* yang digunakan sebagai *site*. Pada implementasinya, penulis memanfaatkan PuTTY sebagai media aksesnya. Dengan menggunakan PuTTY, pengguna dapat *log in* pada sebuah virtual dengan menggunakan akun *user* yang telah dibuat didalam virtual dan penulis juga dapat mengoperasikan secara penuh sesuai dengan hak akses yang diberikan OS kepada akun *user* yang digunakan.

Pada IBM DB2, *linked server* memiliki peran yang hampir sama seperti pada PuTTY yaitu sebagai *remote server*. Pada IBM DB2, *linked server* memiliki peran yang lebih khusus dalam hal *remote server* yaitu dapat digunakan untuk mengakses suatu basis data dari komputer lain. Pada implementasinya dilakukan proses *catalog* jaringan sebagai node dan juga *catalog* basis data sehingga memungkinkan juga sebagai sarana pembentukan federasi basis data. Dalam penelitian ini penulis membuat delapan virtual yang satu diantaranya digunakan sebagai *site* pusat sehingga diperlukan tujuh *linked server* agar *site* pusat dapat mengakses data pada masing-masing virtual.

5.10 Implementasi Pengujian

Pada implementasi pengujian, penulis menggunakan data *dummy* sebagai objek penelitian. Setelah data tersebut difragmen, dilakukan proses uji coba *query* dengan beberapa model seperti yang dijelaskan pada sub bab 5.8 sebanyak 10 kali eksekusi percobaan yang kemudian diambil nilai rata-ratanya. Setelah didapatkan nilai rata-ratanya selanjutnya nilai tersebut dibandingkan dengan nilai total eksekusi *query* dan diambil nilai yang paling rendah.

BAB VI

PENGUJIAN DAN ANALISA

Pada bab ini akan dijelaskan mengenai hasil dari proses implementasi rancangan basis data terdistribusi seperti yang penulis angkat. Hasil dan analis yang akan diuraikan antara lain waktu eksekusi dan biaya yang dibutuhkan oleh beberapa *query* yang penilis desain. Hasil proses dari eksekusi *query* akan dibandingkan antara satu model *query* dengan model lainnya yang masih dalam lingkup satu permasalahan. Model *query* yang digunakan antara lain *query* konvensional dan *query* yang memanfaatkan *materialized view*. Perbandingan analisa waktu dalam penilaian yang dilakukan untuk menunjukkan hasil efisiensi dari bagaimana cara proses melakukan pemanggilan data sebagai bentuk optimasi *query* pada basis data terdistribusi.

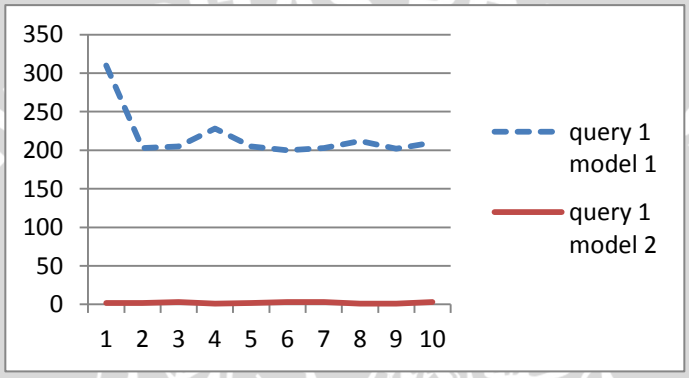
6.1 Pengujian *Query* 1

Query 1 merupakan sebuah *query* yang di desain untuk kasus permasalahan mencari jumlah mahasiswa. Pemodelan *view* pada kasus ini dibentuk dengan mencari jumlah mahasiswa aktif yang digolongkan berdasarkan angkatan, fakultas, program studi, jenjang dan seleksi masuk mahasiswa yang statusnya masih aktif. Pada analisa *query* dengan kasus ini dilakukan uji ciba sebanyak sepuluh kali kemudian ditarik nilai rata-rata dari masing-masing *query*. Selain hasil waktu eksekusi, akan dinilai pula nilai biaya yang terjadi dan ada atau tidaknya paket data berisi informasi data yang diminta *site* pusat kepada *site* yang lain. Hasil waktu eksekusi *query* dapat dilihat pada tabel 6.1.

Tabel 6.1 Hasil Waktu Eksekusi *Query* 1

Desain *query*	Percobaan (*milisecond*)										Rata-rata (*milisecond*)
	1	2	3	4	5	6	7	8	9	10	
Model 1	310	203	205	228	205	200	203	212	202	210	217.8
Model 2	2	2	3	1	2	3	3	1	1	3	2.1

Pada tabel 6.1 ditunjukkan data hasil eksekusi *query* untuk kasus 1. Pada *query* tersebut dilakukan akses kepada tabel fakultas, jenjang, program studi, seleksi dan mahasiswa dengan jumlah total data yang di akses sebesar 54546 *record* data yang terdiri dari 14 re*record* data fakultas, 84 *record* data program studi 54240 *record* data mahasiswa yang masing-masing data tersebut difragmen menjadi delapan bagian dan 208 *record* data seleksi dengan rincian 26 *record* data yang terletak pada masing-masing *site*. Perbandingan eksekusi *query* pada kasus 1 dapat dilihat pada gambar 6.1.



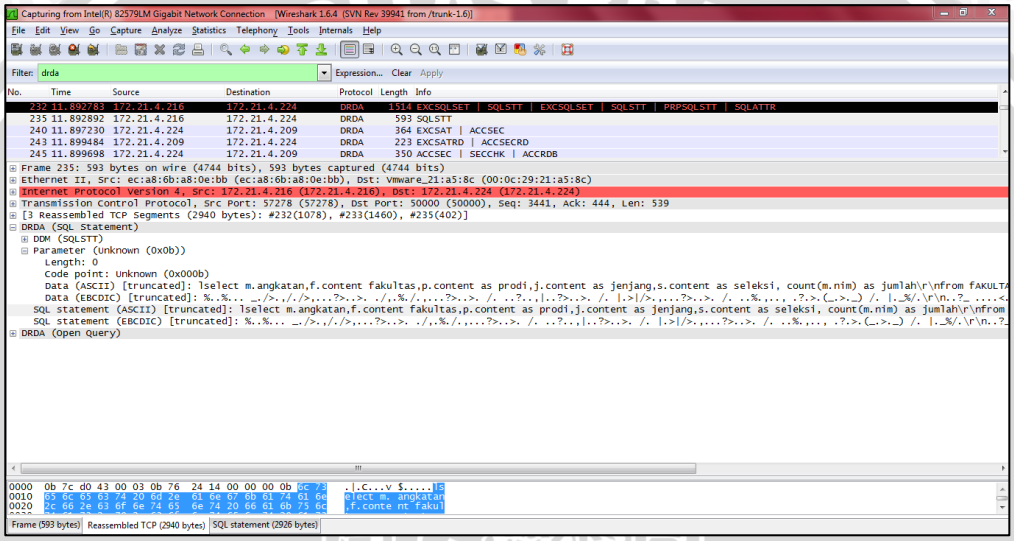
Gambar 6.1 Perbandingan Eksekusi *Query* 1

Pada gambar 6.1 ditunjukkan hasil perbandingan eksekusi *query* konvensional dengan *query* yang memanfaatkan *materialized view*. Perbandingan tersebut didapat dengan melakukan uji coba sebanyak 10 kali terhadap masing-masing *query*. Dari hasil pada gambar 6.1 didapatkan perbandingan yang sangat signifikan terhadap waktu yang dibutuhkan untuk mengeksekusi *query* secara konvensional dengan *query* yang memanfaatkan *materialize view*. Selanjutnya akan diberikan hasil perbandingan yang dibutuhkan untuk mengeksekusi sebuah *query* yang dilakukan dari segi biaya eksekusi, biaya CPU dan biaya I/O yang ditunjukkan pada tabel 6.2.

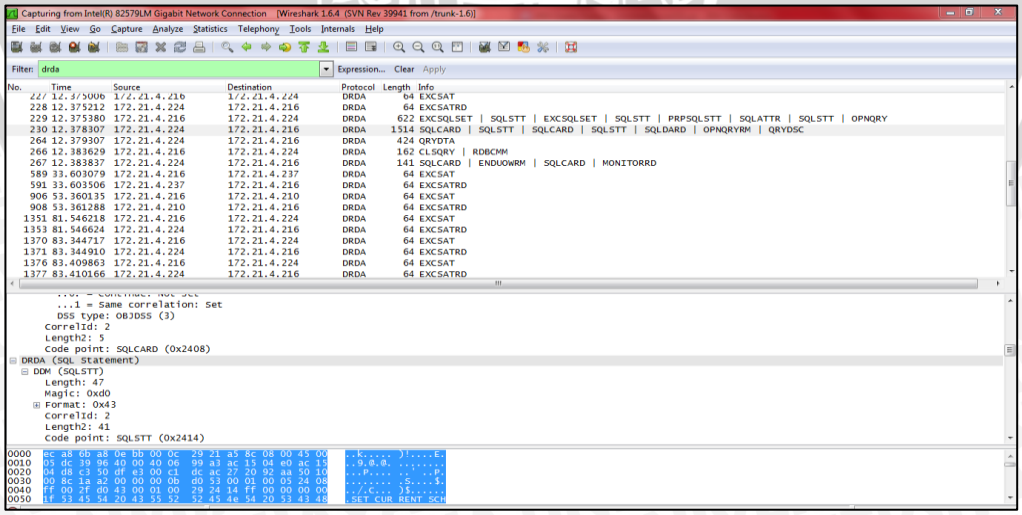
Tabel 6.2 Perbandingan Biaya Eksekusi *Query* 1

Desain *Query*	**Estimasi biaya(*time round*)**		
	Total biaya	**Total CPU**	**Total I/O**
Model 1	3,150.66	911,583,488.00	804
Model 2	212.72	4,012,021.00	31

Dari hasil pada tabel 6.2 ditunjukkan bahwa biaya yang dibutuhkan untuk mengeksekusi *query* model 1 lebih besar jika dibandingkan dengan *query* model 2. Hal tersebut terjadi karena pada *query* model 1 ketika dieksekusi, *query* tersebtu mengakses banyak tabel yang tersimpan pada beberapa *site* federasi, sedangkan pada *query* model 2 hanya dibutuhkan mengakses pada satu tabel saja. Hasil pemantauan jaringan akan ditunjukkan pada gambar 6.2 untuk *query* 1 model 1 dan gambar 6.3 untuk *query* 1 model 2.

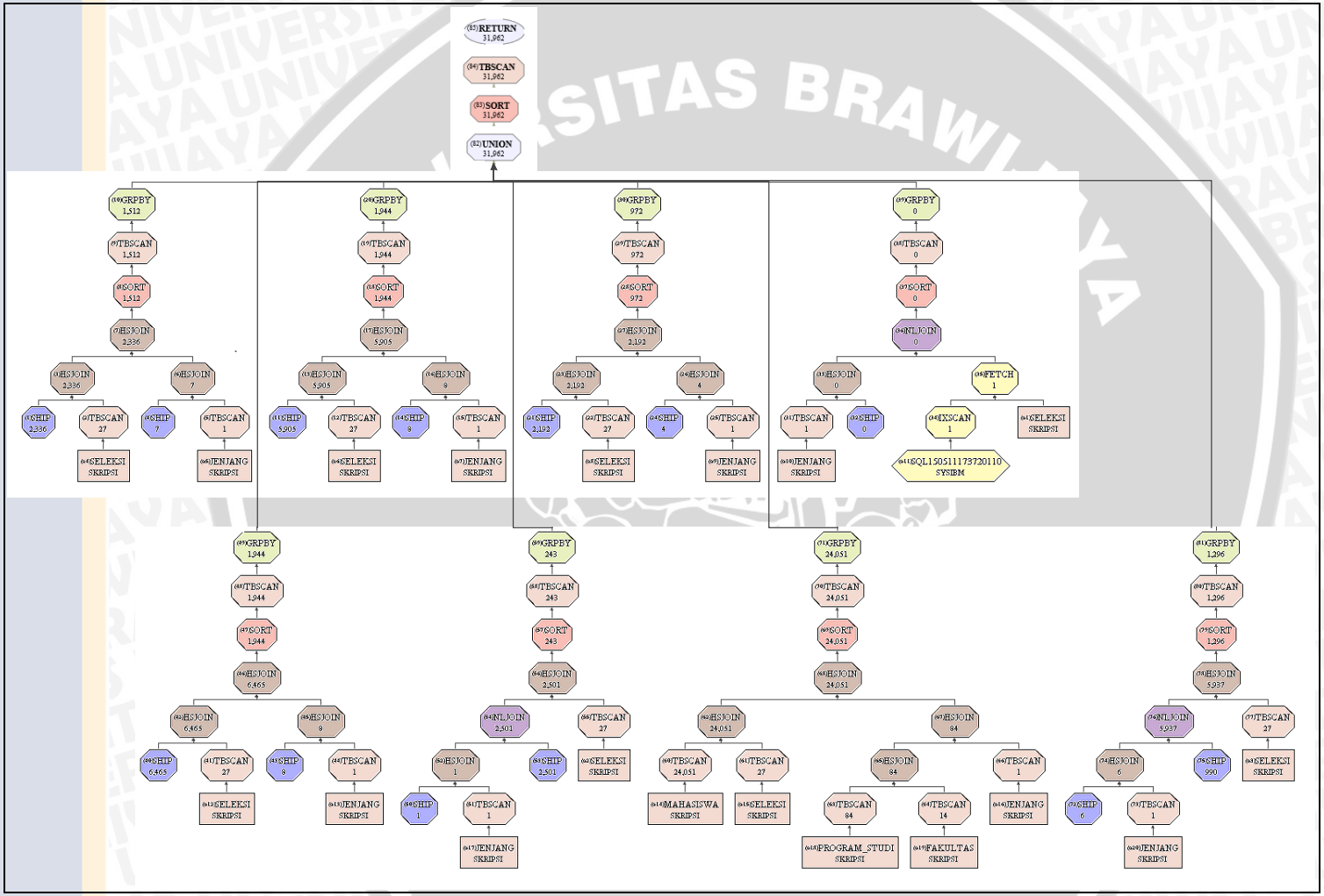


Gambar 6.2 Pemantauan Jaringan *Query* 1 Model 1

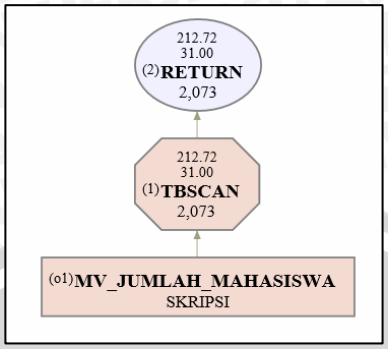


Gambar 6.3 Pemantauan Jaringan *Query* 1 Model 2

Dari hasil yang ditunjukkan pada gambar 6.2 dan 6.3 menunjukkan komunikasi pada jaringan federasi basis data yang menggunakan protokol DRDA. Pada hasil pemantauan tersebut didapatkan ketika mengakses *query* 1 model 1 terjadi komunikasi antara *site* pusat dengan *site* lainnya yang melakukan *request* data seperti yang ditunjukkan seperti pada gambar 6.2. Sedangkan ketika *query* 1 model 2 di eksekusi, tetap terjadi komunikasi antara *site* pusat dengan *site* lainnya tetapi *site* pusat tidak melakukan *request* data seperti yang ditunjukkan pada gabar 6.3. Kemudian dilakukan pemantauan proses akses tabel (*Access plan*) yang memanfaatkan *query tune up* pada data studio seperi yang dapat dilihat gambar 6.4 dan 6.5.



Gambar 6.4 *Access Plan Local Site Query 1*



Gambar 6.5 *Access Plan Materialized View Query 1*

Pada gambar 6.4 yang merupakan *access plan* dari eksekusi *query* konvensional menunjukkan terdapat *SHIP* yang merupkan pengiriman data tabel yang menjadi *nick name* pada *site* utama. Sedangkan pada gambar 6.5 merupakan *access plan* dari eksekusi *query* yang menggunakan *materialized view*. Pada gambar 6.5 menunjukkan tidak adanya *SHIP* seperti pada gambar 6.4 dan pada *access plan* yang menggunakan *materialized view* hanya menunjukkan akses terhadap satu tabel saja yaitu MV_JUMLAH_MAHASISWA.

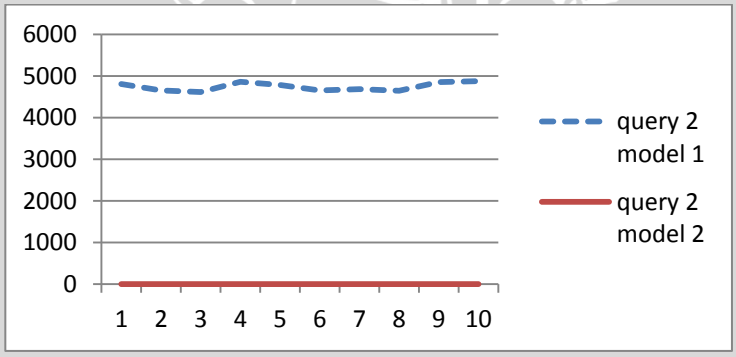
5.2 Pengujian *Query* 2

Query 2 merupakan sebuah *query* yang di desain untuk kasus permasalahan mencari jumlah total nilai transaksi. Pemodelan *view* pada kasus ini dibentuk dengan mencari jumlah total nilai transaksi mahasiswa aktif yang digolongkan berdasarkan fakultas, program studi, jenjang, seleksi dan item dari seluruh mahasiswa yang statusnya masih aktif. Pada analisa *query* dengan kasus ini dilakukan uji ciba sebanyak sepuluh kali kemudian ditarik nilai rata-rata dari masing-masing *query*. Selain hasil waktu eksekusi, akan dinilai pula nilai biaya yang terjadi dan ada atau tidaknya paket data berisi informasi data yang diminta *site* pusat kepada *site* yang lain. Hasil waktu eksekusi *query* dapat dilihat pada tabel 6.3.

Tabel 6.3 Hasil Waktu Eksekusi *Query* 2

Desain *query*	**Percobaan (milisecond)**										**Rata-rata**
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	
Model 1	4806	4649	4612	4861	4782	4651	4682	4645	4855	4878	4742.1
Model 2	3	1	1	2	1	2	2	2	1	1	2.1

Pada tabel 6.3 ditunjukkan data hasil eksekusi *query* untuk kasus 2. Pada *query* tersebut dilakukan akses kepada tabel fakultas, jenjang, program studi, seleksi, mahasiswa, item, transaksi dan keuanagn dengan jumlah total data yang di akses sebesar 1847650 *record* data yang terdiri dari 14 re*record* data fakultas, 84 *record* data program studi, 54240 *record* data mahasiswa dan 1793104 *record* data yang masing-masing data tersebut difragmen menjadi delapan bagian dan 208 *record* data seleksi dengan rincian 26 *record* data yang terletak pada masing-masing *site*. Perbandingan eksekusi *query* pada kasus 2 dapat dilihat pada gambar 6.6.



Gambar 6.6 Perbandingan Eksekusi *Query* 2

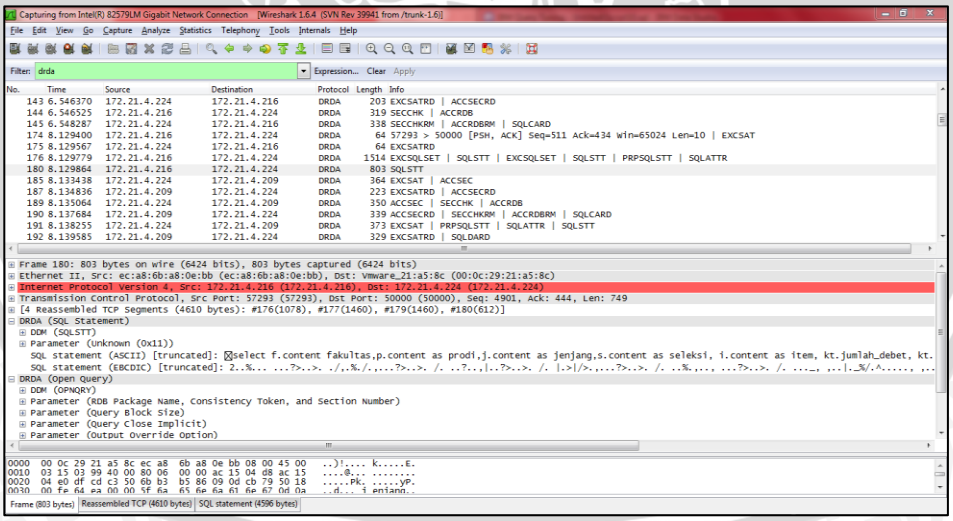
Pada gambar 6.6 ditunjukkan hasil perbandingan eksekusi *query* konvensional dengan *query* yang memanfaatkan *materialized view*. Perbandingan tersebut didapat dengan melakukan uji coba sebanyak 10 kali terhadap masing-masing *query*. Dari hasil pada gambar 6.6 didapatkan perbandingan yang sangat signifikan terhadap waktu yang dibutuhkan untuk mengeksekusi *query* secara konvensional dengan *query* yang memanfaatkan

materialize view. Selanjutnya akan diberikan hasil perbandingan yang dibutuhkan untuk mengeksekusi sebuah *query* yang dilakukan dari segi biaya eksekusi, biaya CPU dan biaya I/O yang ditunjukkan pada tabel 6.4.

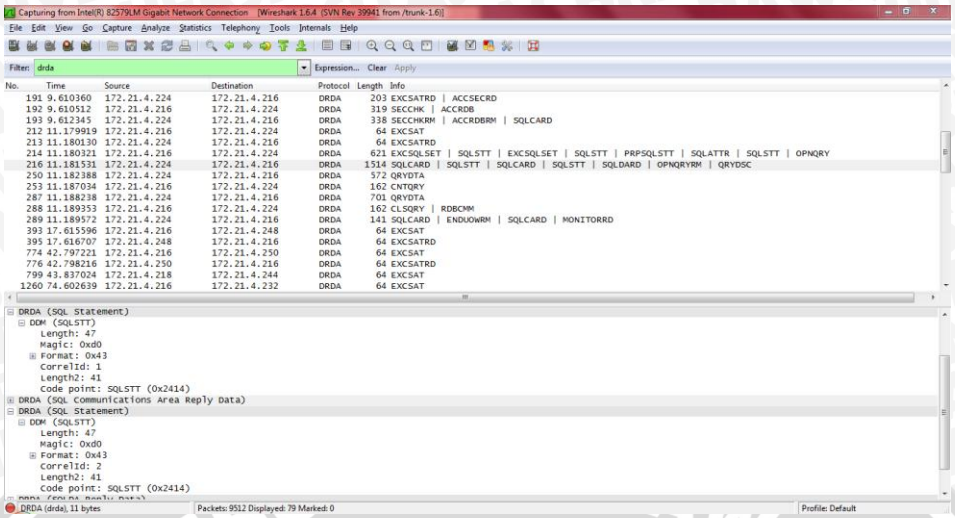
Tabel 6.4 Perbandingan Biaya Eksekusi *Query* 2

Desain *Query*	**Estimasi biaya(*time round*)**		
	Total biaya	**Total CPU**	**Total I/O**
Model 1	168,926.70	32,259,338,240.00	52,896.00
Model 2	826.69	88,715,288.00	850

Dari hasil pada tabel 6.4 ditunjukkan bahwa biaya yang dibutuhkan untuk mengeksekusi *query* model 1 lebih besar jika dibandingkan dengan *query* model 2. Hal tersebut terjadi karena pada *query* model 1 ketika dieksekusi, *query* tersebtu mengakses banyak tabel yang tersimpan pada beberapa *site* federasi, sedangkan pada *query* model 2 hanya dibutuhkan mengakses pada satu tabel saja. Hasil pemantauan jaringan akan ditunjukkan pada gambar 6.7 untuk *query* 2 model 1 dan gambar 6.8 untuk *query* 2 model 2.

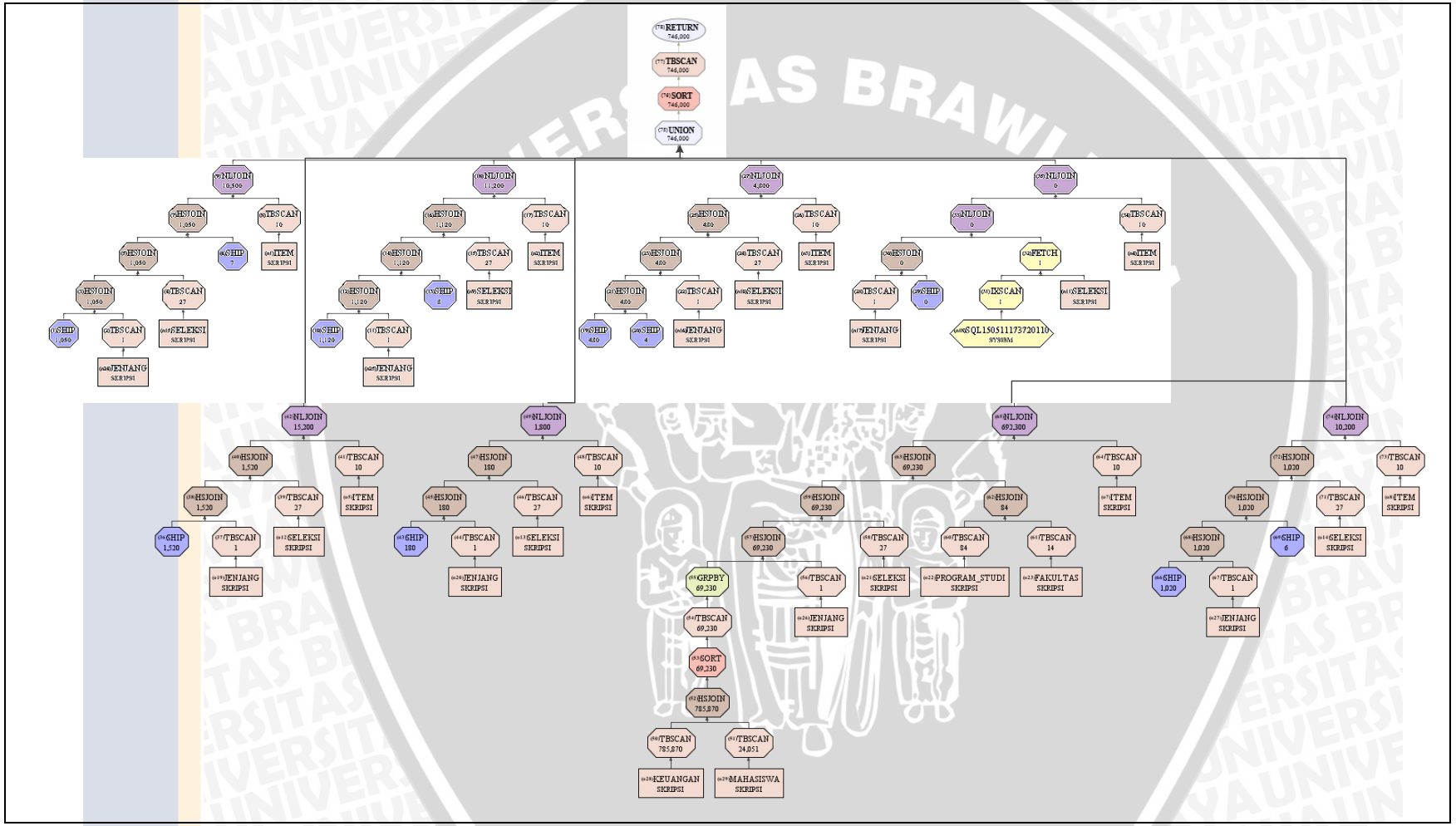


Gambar 6.7 Pemantauan Jaringan *Query* 2 Model 1

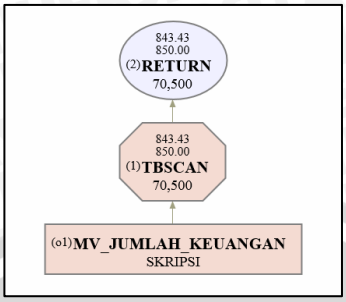


Gambar 6.8 Pemantauan Jaringan *Query* 2 Model 2

Dari hasil yang ditunjukkan pada gambar 6.7 dan 6.8 menunjukkan komunikasi pada jaringan federasi basis data yang menggunakan protokol DRDA. Pada hasil pemantauan tersebut didapatkan ketika mengakses *query* 2 model 1 terjadi komunikasi antara *site* pusat dengan *site* lainnya yang melakukan *request* data seperti yang ditunjukkan seperti pada gambar 6.7. Sedangkan ketika *query* 2 model 2 di eksekusi, tetap terjadi komunikasi antara *site* pusat dengan *site* lainnya tetapi *site* pusat tidak melakukan *request* data seperti yang ditunjukkan pada gambar 6.8. Kemudian dilakukan pemantauan proses akses tabel (*Access plan*) yang memanfaatkan *query tune up* pada data studio seperi yang dapat dilihat gambar 6.9 dan 6.10.



Gambar 6.9 *Access Plan Local Site Query 2*



Gambar 6.10 *Access Plan Materialized View Query 2*

Pada gambar 6.9 yang merupakan *access plan* dari eksekusi *query* konvensional menunjukkan terdapat *SHIP* yang merupkan pengiriman data tabel yang menjadi *nick name* pada *site* utama. Sedangkan pada gambar 6.10 merupakan *access plan* dari eksekusi *query* yang menggunakan *materialized view*. Pada gambar 6.10 menunjukkan tidak adanya *SHIP* seperti pada gambar 6.9 dan pada *access plan* yang menggunakan *materialized view* hanya menunjukkan akses terhadap satu tabel saja yaitu MV_JUMLAH_KEUANGAN.

6.3 Analisa Hasil

Pada penggunaan *query* 1 yang digunakan untuk menampilkan jumlah mahasiswa, penggunaan *query* 1 model 1 pada saat akan di eksekusi akan mengakses banyak tabel. Berbeda dengan penggunaan *query* 1 model 2, ketika di eksekusi *query* ini hanya akan mengakses satu tabel saja. Hasil perbandingan penggunaan dapat dilihat seperti pada tabel 6.1. pada penggunaannya, *query* 1 model 2 memiliki nilai waktu, biaya dan transfer paket data yang lebih minim jika dibandingkan dengan dengan penggunaan *query* 1 model 1. Sehingga dengan hasil tersebut dapat dilihat dengan memanfaatkan *materialized view* seperti yang diterapkan pada *query* 1 model 2 akan memberikan nilai penggunaan yang lebih efisien jika dibandingkan dengan penggunaan *query* konvensional. Pada *query* 1 model 2 menunjukkan peningkatan kinerja yang ditandai dengan semakin kecilnya waktu yang dibutuhkah untuk mengeksekusi sebuah *query* sebesar 103.7142857 kali lebih

cepat atau waktu eksekusi rata-rata *query* 1 model 2 hanya menjadi 0.964187328% dari waktu rata-rata akses *query* 1 model 1. Efisiensi waktu ini didapatkan dari beberapa hal seperti banyaknya mengakses tabel yang dibuktikan dari pemantauan jaringan menggunakan wireshark. Pada pemantauan wireshark untuk *query* 1 model 1 terdapat komunikasi *request* data terhadap beberapa *site* yang tertangkap pada *SQL statement.* Sedangkan pada *query* 1 model 2 saat dieksekusi, wireshark juga memantau dan meng*capture* komunikasi yang terjadi tetapi tidak terdapat *request* data yang dikirimkan kepada *site* lain. Kemudian ketika dilakukan *query tune up* menggunakan data studio, didapatkan hasil terdapat proses *ship* pada eksekusi *query* 1 model 1, sedangkan pada eksekusi *query* 1 model 2 tidak terdapat proses *ship*. Sehingga dengan kejadian ini juga menyebabkan tidak banyak terpakainya *resource* virtual dari *site* lain dan penggunaan *resource* total dapat diminalisir dengan ditunjukkan kecilnya biaya *query*, biaya CPU dan biaya I/O.

Hal serupa juga terjadi terhadap eksekusi dari *query* 2. Pada saat mengeksekusi *query* 2 model 1 dibutuhkan waktu yang lebih lama jika dibandingkan dengan mengakses *query* 2 model 2. Pada saat mengeksekusi *query* 2 model 2 waktu rata-rata yang dibutuhkan adalah 2963.813 kali lebih cepat dibandingkan dengan waktu rata-rata eksekusi *query* 2 model 1 atau dengan kata lain hanya membutuhkan waktu 0.03374% dari waktu rata-rata eksekusi *query* 2 model 1. Hal tersebut terjadi karena pada saat mengeksekusi *query* 2 model 2 hanya butuh mengakses satu tabel saja. Berbeda dengan ketika mengeksekusi *query* 2 model 1 yang perlu untuk mengakses banyak tabel dan tersebar pada beberapa *site* lain yang tergabung dengan federasi basis data. Proses akses tabel pada penggunaan *query* 2 model 1 ditandai dengan adanya *request* data kepada *site* yang tergabung pada federasi basis data seperti yang ditunjukkan pada *capture* data dalam gambar 6.7. Pada gambar 6.7 ditunjukkan pada *SQL statement* terdapat *request* data terhadap *site* lain yang tergabung dalam federasi basis data. Kemudian ketika dilakukan *query tune up* menggunakan data studio, didapatkan hasil terdapat proses *ship*

pada eksekusi *query* 2 model 1, sedangkan pada eksekusi *query* 2 model 2 tidak terdapat proses *ship*. Sehingga ketika *query* 2 model 1 di eksekusi maka akan membutuhkan *resource* tambahan yang berjalan pada *site* yang dituju. Proses tersebut berbeda ketika *query* 2 model 2 di eksekusi karena pada *query* 2 model 2 hanya akan menggunakan *resource* pada satu *site* saja dimana *materialized view* yang telah terbentuk tersimpan.

BAB VII

KESIMPULAN

7.1 Kesimpuan

Sebagaimana tujuan dari penelitian ini dimaksudkan untuk melakukan implementasi *materialized view* pada basis data terdistribusi dan melakukan perbandingan terhadap penggunaan *query* konvensional, maka dari hasil penelitian yang diperoleh, berikut ini merupakan kesimpulan dari penelitian yang telah dilakukan:

1. *Materialized view* diimplementasikan pada *site* utama dalam basis data terdisribusi dengan menggunakan mekanisme federasi basis data. Dengan mekanisme ini memungkinkan bagi *site* pusat dapat mengakses data pada *site* lain. Desain dari *materialized view* dibentuk dari penyusunan *query* konvensional sehingga model *view* yang terbentuk sama seperti model *view* pada *query* konvensional.
2. Penggunaan *materialized view* pada basis data terdistribusi memiliki perbandingan nilai waktu eksekusi yang lebih optimal jika dibandingkan dengan penggunaan *query* konvensional. Hasil diperoleh pada eksekusi *query* 1 model 2 yang menggunakan *materialized view* menunjukkan 103.7142857 kali lebih cepat atau waktu eksekusi rata-rata hanya menjadi 0.964187328% dari waktu rata-rata akses *query* 1 model 1 yang merupakan *query* konvensional. Sedangkan pada *query* 2 model 2 yang menggunakan *materialized view* menunjukkan 2963.813 kali lebih cepat dibandingkan dengan waktu rata-rata eksekusi *query* konvensional pada *query* 2 model 1 atau dengan kata lain hanya membutuhkan waktu 0.03374% dari waktu rata-rata eksekusi *query* konvensional pada *query* 2 model 1.
3. Penggunaan *materialized view* pada basis data terdistribusi dapat meminimalkan biaya penggunaan *resource* seperti CPU dan I/O yang ada jika dibandingkan dengan *query* konvensional. Hal tersebut terjadi

karena ketika memanfaatkan *materialized view*, untuk menampilkan *view* tersebut DBMS hanya membutuhkan *resource* pada *server* yang menyimpan *materialized view* tersebut. Sehingga jika *materialized view* tersimpan pada *site* utama dari federasi basis data, maka *resource* yang dibutuhkan untuk menampilkan model *view* lebih minimal karena tidak membutuhkan proses dan *resource* dari *site* yang lain.

4. Penggunaan *materialized view* pada basis data terdistribusi dapat meminimalkan biaya transfer data yang terjadi antar *site*. Hal tersebut terjadi ketika model *view* dari sebuah *materialized view* dipanggil, *site* yang menyimpan *materialized view* tersebut tidak melakukan *request* data kepada *site* lokal, tetapi ketika dilakukan pembaruan (*refresh table*) dibutuhkan biaya jaringan. Biaya tersebut terjadi karena adanya *request* data dari *site* pusat terhadap *site* lokal.

7.2 Saran

Dengan hasil implementasi dari penelitian ini, terdapat beberapa saran yang dapat digunakan dalam penerapan dan desain basis data antara lain:

1. Penggunaan *materialized view* merupakan sebuah metode optimasi yang sangat efisien jika diterapkan pada basis data terdistribusi.
2. Pada penyusunan *materialized view* di sarankan *query* yang digunakan untuk membentuk *view* merupakan *query* yang optimal, hal tersebut dikarenakan akan berpengaruh terhadap *refresh* tabel atau dalam perawatan *materialized view*.
3. Efisiensi waktu dan biaya dimungkinkan dapat terjadi jika dilakukan penggabungan beberapa metode optimasi *query*.

DAFTAR PUSTAKA

[NICA] Nica A., *Incremental maintenance of materialized views with outerjoins*, Sybase, An SAP Company, Waterloo, Ontario, Canada, 2011

[SIL] Silberschatz, Korth and Sudarshan , *Database System Concepts*, McGraw Hill,Fourth Edition, 2001

[OZSU] Ozsu MT, Valduriez Patrick, *Principles of Distributed Database Systems*, Third Edition, Springer Science+Business Media, LLC 2011

[ELMASRI] Elmasri and Navethe, *Fundamentals of Database Systems*, 6th Edition, Addison-Wesley, 2010

[LIN] Zhou L., Li T., Chen Y. and Yu Y.,*The Semi-join Query Optimization in Distributed Database System*, Atlantis Press, 2012

[JON] Goldstein J. and Larson Per-Åke, Optimizing Queries Using Materialized *views*: A Practical, Scalable Solution, Microsoft Research, One Microsoft Way, Redmond, WA 98052

[ARB1] Raipurkar A. and Bamnote G.R., *Query Processing In Distributed Database Through Data Distribution*, International Journal of Advanced Research in Computer and Communication Engineering Vol. 2, Issue 2, Februari 2013

[ARB2] Raipurkar A. and Bamnote G.R., *Query Optimization In Distributed Database System*, International Journal Of Computer Science And Applications Vol. 6, No.2, April 2013

[TRI] Wahyu T., *PENGOPTIMASIAN PENCARIAN DATA DENGAN ALGORITMA SUBSET QUERY*, Jurnal Artificial , ICT Research Center UNAS Vol.2 No.1 Januari 2008

[MICRO] MicroStrategy Labs and VLDB Technology Group, *Configuring Federated Database Systems in IBM DB2 Universal Database with the microStrategy Platform*, MicroStrategy Confidential, June, 2001

[KARDE] Karde P. P. and Thakare V. M., *SELECTION OF MATERIALIZED VIEW USING QUERY OPTIMIZATION IN DATABASE MANAGEMENT AN EFFICIENT METHODOLOGY*, International Journal of Database Management Systems (IJDMS), Vol.2, No.4, November 2010

[SSV] Agrawal Sanjay, Chaudhuri Surajit, and Narasayy Vivek,*Automated Selection of Materialized Views and Indexes for SQL Databases*, Proceedings of the 26th International Conference on Very Large Databases, Cairo, Egypt, 2000

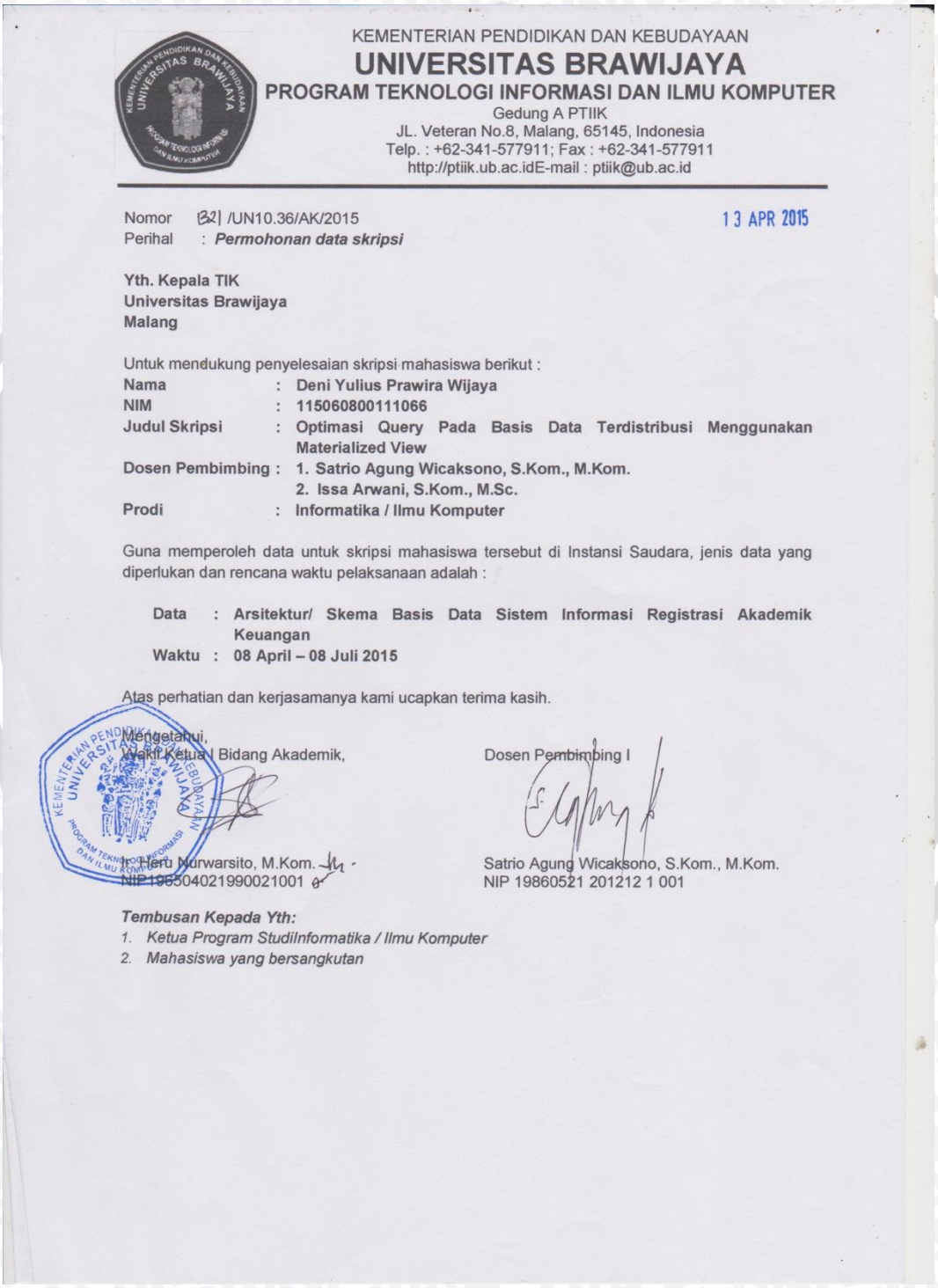
[PAESLLER] Dirk P., *Server Virtualization and Network Management*,PAESLLER, August 2008—Last Update: March 2012

[IBM] IBM, Repository : ww.ibm.com/developerworks/data/library/techarticle/dm-0509melynk, di akses pada tanggal 14 maret 2015

LAMPIRAN

Lampiran 1

Surat Permohonan Data Skripsi Kepada TIK UB



KEMENTERIAN PENDIDIKAN DAN KEBUDAYAAN
UNIVERSITAS BRAWIJAYA
PROGRAM TEKNOLOGI INFORMASI DAN ILMU KOMPUTER
Gedung A PTIIK
JL. Veteran No.8, Malang, 65145, Indonesia
Telp. : +62-341-577911; Fax : +62-341-577911
http://ptiik.ub.ac.idE-mail : ptiik@ub.ac.id

Nomor 82 /UN10.36/AK/2015 — 13 APR 2015
Perihal : ***Permohonan data skripsi***

Yth. Kepala TIK
Universitas Brawijaya
Malang

Untuk mendukung penyelesaian skripsi mahasiswa berikut :

Nama	:	**Deni Yulius Prawira Wijaya**
NIM	:	**115060800111066**
Judul Skripsi	:	**Optimasi Query Pada Basis Data Terdistribusi Menggunakan Materialized View**
Dosen Pembimbing	:	**1. Satrio Agung Wicaksono, S.Kom., M.Kom.** **2. Issa Arwani, S.Kom., M.Sc.**
Prodi	:	**Informatika / Ilmu Komputer**

Guna memperoleh data untuk skripsi mahasiswa tersebut di Instansi Saudara, jenis data yang diperlukan dan rencana waktu pelaksanaan adalah :

Data	:	**Arsitektur/ Skema Basis Data Sistem Informasi Registrasi Akademik Keuangan**
Waktu	:	**08 April – 08 Juli 2015**

Atas perhatian dan kerjasamanya kami ucapkan terima kasih.

Mengetahui,
Wakil Ketua I Bidang Akademik,

Ir. Heru Nurwarsito, M.Kom.
NIP 19650402199002100 1

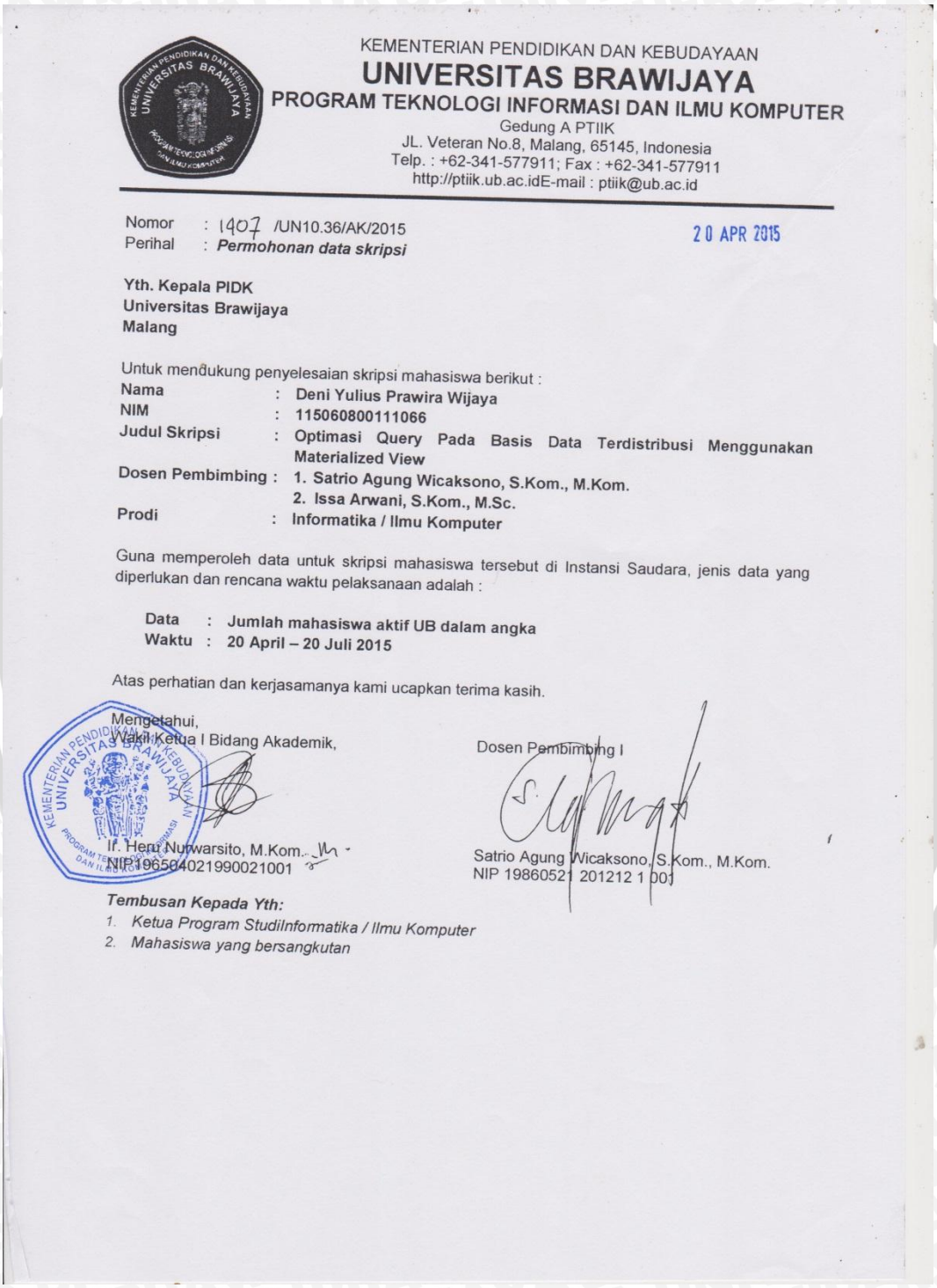
Dosen Pembimbing I

Satrio Agung Wicaksono, S.Kom., M.Kom.
NIP 19860521 201212 1 001

Tembusan Kepada Yth:
1. *Ketua Program StudiInformatika / Ilmu Komputer*
2. *Mahasiswa yang bersangkutan*

Lampiran 2

Surat Permohonan Data Skripsi Kepada PIDK UB



KEMENTERIAN PENDIDIKAN DAN KEBUDAYAAN
UNIVERSITAS BRAWIJAYA
PROGRAM TEKNOLOGI INFORMASI DAN ILMU KOMPUTER
Gedung A PTIIK
JL. Veteran No.8, Malang, 65145, Indonesia
Telp. : +62-341-577911; Fax : +62-341-577911
http://ptiik.ub.ac.idE-mail : ptiik@ub.ac.id

Nomor : 1407 /UN10.36/AK/2015
Perihal : ***Permohonan data skripsi***

20 APR 2015

Yth. Kepala PIDK
Universitas Brawijaya
Malang

Untuk mendukung penyelesaian skripsi mahasiswa berikut :

Nama	:	**Deni Yulius Prawira Wijaya**
NIM	:	**115060800111066**
Judul Skripsi	:	**Optimasi Query Pada Basis Data Terdistribusi Menggunakan Materialized View**
Dosen Pembimbing	:	**1. Satrio Agung Wicaksono, S.Kom., M.Kom.**
		2. Issa Arwani, S.Kom., M.Sc.
Prodi	:	**Informatika / Ilmu Komputer**

Guna memperoleh data untuk skripsi mahasiswa tersebut di Instansi Saudara, jenis data yang diperlukan dan rencana waktu pelaksanaan adalah :

Data : Jumlah mahasiswa aktif UB dalam angka
Waktu : 20 April – 20 Juli 2015

Atas perhatian dan kerjasamanya kami ucapkan terima kasih.

Mengetahui,
Wakil Ketua I Bidang Akademik,

Ir. Heru Nurwarsito, M.Kom.
NIP.196504021990021001

Dosen Pembimbing I

Satrio Agung Wicaksono, S.Kom., M.Kom.
NIP 19860521 201212 1 001

Tembusan Kepada Yth:
1. *Ketua Program StudiInformatika / Ilmu Komputer*
2. *Mahasiswa yang bersangkutan*