IMPLEMENTASI MATERIALIZED QUERY TABLE UNTUK PEMBANGUNAN DATA MART (STUDI KASUS: UNIVERSITAS BRAWIJAYA BAGIAN AKADEMIK)

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Nurcahyo Pujo Nugroho
NIM: 115061000111015

PROGRAM STUDI SISTEM INFORMASI
PROGRAM TEKNOLOGI INFORMASI DAN ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2015

PENGESAHAN

IMPLEMENTASI MATERIALIZED QUERY TABLE UNTUK PEMBANGUNAN DATA MART (STUDI KASUS: UNIVERSITAS BRAWIJAYA BAGIAN AKADEMIK)

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Nurcahyo Pujo Nugroho
NIM: 115061000111015

Skripsi ini telah diuji dan dinyatakan lulus pada
12 November 2015
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	*Dosen Pembimbing II*
Satrio Agung W, S.Kom., M.Kom	Budi Darma S, S.Kom., M.Cs
NIP. 19860521 2012121001	NIP. 198410152014041002

Mengetahui
Ketua Program Studi Sistem Informasi

Suprapto, S.T., M.T
NIP. 19710727 199603 1 001

PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam 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, 12 November 2015

Nurcahyo Pujo Nugroho
NIM: 115061000111015

KATA PENGANTAR

Puji dan syukur dipanjatkan kehadirat Allah SWT yang telah melimpahkan berkat, kemudahan dan karunia-Nya serta kelancaran sehingga penulis dapat menyelesaikan skripsi dengan judul "Implementasi Materialized Query Table Untuk Pembangunan Data Mart (Studi Kasus: Universitas Brawijaya Bagian Akademik)". Melalui pengantar ini penulis mengucapkan banyak terima kasih karena dalam penyusunan skripsi ini penulis telah mendapat banyak bantuan dan dorongan baik moril maupun materil dari berbagai pihak. Untuk itu pada kesempatan ini penulis ingin mengucapkan banyak terimakasih kepada:

1. Bapak Satrio Agung W, S.Kom., M.Kom. selaku dosen pembimbing I dan Bapak Budi Darma S, S.Kom., M.Cs. selaku dosen pembimbing II.
2. Orang tua Bapak Arie Sudiyono dan Ibu Nur Khoiriyah serta adik Nurfitra yang telah memberi kasih sayang, semangat dan dukungan bagi penulis.
3. Saudari Rizki Anggarsasi yang siap sedia memberi dorongan semangat, bantuan dan motivasi serta kesabaran.
4. Saudari Nadia Soraya, Saudara Fuad Haif R dan Alfiyan Arief sebagai teman diskusi dan seperjuangan selama proses bimbingan dan penulisan.
5. Seluruh pengurus PPTI dan PIDK Universitas Brawijaya yang telah membantu proses pengumpulan data skripsi.
6. Segenap bapak/ ibu dosen dan karyawan Program Teknologi Informasi dan Ilmu Komputer Universitas Brawijaya.
7. Semua teman-teman Sistem Informasi angkatan 2011 dan segenap pengurus Eksekutif Mahasiswa Sistem Informasi yang telah memberikan dukungannya.

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

Malang, 12 November 2015

Nurcahyo Pujo Nugroho
115061000111015@ub.ac.id

ABSTRAK

Nurcahyo Pujo Nugroho. 2015. Implementasi Materialized Query Table Untuk Pembangunan Data Mart (Studi Kasus: Universitas Brawijaya Bagian Akademik). Program Teknologi Informasi dan Ilmu Komputer, Universitas Brawijaya. Malang. Dosen Pembimbing: Satrio Agung W, S.Kom., M.Kom. dan Budi Darma S, S.Kom., M.Cs.

Data mart merupakan sistem penyimpanan data historis terintegrasi yang dibangun untuk subjek tertentu. Dengan ruang lingkup lebih kecil dibandingkan dengan data *warehouse* membuat *data mart* lebih mudah dibangun, dikelola dan dirasakan manfaatnya. Peran utama *data mart* adalah sebagai penyedia data bahan analisis untuk mendukung proses pengambilan keputusan. Yang pada pelaksanaannya akan melibatkan serangkaian *query-query* kompleks yang dilakukan berulang-ulang dan diakses oleh banyak pengguna. Di sinilah *materialized query table* akan berperan untuk membantu melakukan efisiensi dalam proses pengambilan dan pengoperasian data.

Pada penelitian ini akan dilakukan pembandingan performa dan ketahanan *query* yang melalui implementasi *materialized query table* dan *query* yang langsung mengakses ke *data mart*. Penelitian akan dilakukan di Universitas Brawijaya yang memiliki mahasiswa dengan jumlah masif dan difokuskan pada bagian akademik. Hasil pengujian menunjukkan dengan mengimplementasikan *materialized query table* dapat meningkatkan performa hingga 53 kali lipat, meningkatkan efisiensi sumber daya hingga 490 kali lipat dan meningkatkan ketahanan hingga 50 kali lipat.

Kata Kunci: *Data Mart, Materialized Query Table, Materialized View*

ABSTRACT

Nurcahyo Pujo Nugroho. 2015. Implementasi *Materialized Query Table* Untuk Pembangunan *Data Mart* (Studi Kasus: Universitas Brawijaya Bagian Akademik). *Program Information Technology and Computer Science, Brawijaya University. Malang. Advisor: Satrio Agung W, S.Kom., M.Kom. and Budi Darma S, S.Kom., M.Cs.*

Data mart is system to store integrated historical data that developed for specific subject. Compared to data warehouse, data mart has smaller scope. This trait makes data mart easier to develop, easier to maintain and produce faster result. Data mart main role is to provide data source for analysis process in support decision making process. In practice data mart will be utilize a series of complex query that will be executed repeatedly with many user at once. To optimize this practice researcher will use materialized query table.

This research will compare performance and durability conventional data mart and which implemented materialized query table. Research will conduct in Brawijaya University that has massive student and focused on academic sector. Test results show with implementation of materialized query table could increase performance up to 53 times, increase resource efficiency up to 490 times and increase durability up to 50 times.

Keywords: *Data Mart, Materialized Query Table, Materialized View*

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB 1 PENDAHULUAN

1.1 Latar belakang

Data mart adalah penyimpanan data historis terintegrasi yang dibangun demi subjek atau tujuan yang spesifik. Data historis yang tersimpan di *data mart* berguna sebagai bahan analisis perkembangan performa perusahaan dari waktu ke waktu atau bahkan sebagai dasar prakiraan keadaan perusahaan di masa mendatang. Untuk itu diperlukan dukungan dari berbagai sumber data internal maupun eksternal yang akan diintegrasikan pada *data mart*. Berbeda dengan *data warehouse* yang mengakomodasi seluruh kebutuhan dari perusahaan, *data mart* dibangun untuk memenuhi kebutuhan departemen tertentu saja seperti pemasaran, keuangan dan sebagainya (Gallo & Perilli, 2010).

Dikarenakan dibangun khusus untuk departemen tertentu maka pengelolaan data dapat dilakukan per departemen tanpa mengganggu kinerja departemen yang lain. Sehingga sistem dapat lebih fokus mengelola data yang dimiliki departemen saja dan respons *query* akan menjadi lebih cepat. Dengan ruang lingkup yang lebih kecil, tujuan yang spesifik serta tak diperlukannya koordinasi antar departemen membuat pembangunan *data mart* dapat dilakukan lebih cepat dan murah sehingga manfaatnya dapat segera dirasakan (Ballard, et al., 2005).

Selain dapat dibangun secara independen berdasarkan sistem lokal di departemen, *data mart* dapat juga dibangun dari irisan *data warehouse* sesuai kebutuhan departemen yang membutuhkan. Sama seperti *data warehouse,* data yang tersimpan di *data mart* berasal dari beragam sumber data yang melalui serangkaian proses ETL (*extract, transform* dan *load*) sehingga data memenuhi format yang dibutuhkan. Proses ETL inilah yang akan menentukan kualitas data yang ada pada *data mart* dan *data warehouse*. *Data mart* menyimpan data menggunakan model multidimensional (*data warehouse schema*) yang berfokus pada pengukuran, matriks atau fakta yang ada pada proses bisnis. Komponen dasar dari model multidimensional adalah tabel dimensi dan tabel fakta (Chhabra & Pahwa, 2014). Tabel Fakta berisi pengukuran numerik seperti kuantitas, jumlah dan sebagainya yang digunakan untuk analisa hubungan antar dimensi. Sementara tabel dimensi berisi dimensi-dimensi entitas perusahaan yang saling terkait melalui tabel fakta. Sebagai contoh jika sebuah universitas akan memiliki tabel fakta mengenai indeks prestasi mahasiswa maka dapat memiliki tabel dimensi fakultas, waktu, program studi dan lain sebagainya.

Model multidimensional yang digunakan *data mart* ini dapat dikatakan sangat efektif untuk penyimpanan data hasil agregasi. Hal itu dikarenakan model ini dapat meminimalkan redundansi data dan meningkatkan performa penyimpanan. Namun meskipun begitu, model ini memiliki kelemahan pada proses pengambilan dan pengoperasian data. Karena seperti yang kita ketahui bersama saat melakukan analisis data, maka akan membutuhkan data yang sangat banyak dan beragam. Sehingga dengan model multidimensional ini *query-query* kompleks

yang melibatkan banyak *join* dan operasi agregasi pada beberapa tabel sekaligus akan sangat sering digunakan. Hal ini tentu saja dapat sangat membebani performa sistem secara keseluruhan terutama jika diakses oleh banyak pengguna dan dalam jumlah besar.

Materialized Query Table atau yang juga disebut *materialized view* adalah salah satu entitas yang digunakan sistem basis data untuk menyimpan hasil *query* agar dapat digunakan berulang kali. Keuntungan menggunakan *materialized query table* adalah untuk mengaksesnya hanya perlu membaca blok lokasi dimana *materialized query table* disimpan tanpa perlu melakukan komputasi ulang pada tiap pengaksesan (Paraboschi, et al., 2003). Sehingga dapat mengurangi biaya *query* dan meningkatkan kecepatan pengaksesan data.

Universitas Brawijaya adalah salah satu universitas besar di Indonesia. Seperti instansi pendidikan pada umumnya, performa akademik mahasiswa merupakan salah satu pilar penting dalam proses bisnis yang ada di Universitas Brawijaya. Sehingga analisa data akademik mahasiswa dengan efektif dan efisien merupakan kebutuhan yang tak terelakkan bagi Universitas Brawijaya. Sementara itu menurut *website* resmi Universitas Brawijaya hingga tahun 2015, Universitas Brawijaya memiliki sejumlah 59.469 mahasiswa aktif dari berbagai strata yang tersebar di 15 fakultas (Universitas Brawijaya, 2015). Dari fakta tersebut dapat dibayangkan betapa banyaknya data akademik yang dihasilkan Universitas Brawijaya pada setiap proses bisnisnya. Ditambah dengan beragamnya sistem penyimpanan data yang digunakan maka jika ingin melakukan analisa secara rutin dan mendalam pada data akademik Universitas Brawijaya secara manual, maka sumber daya yang dibutuhkan akan sangat besar.

Dengan dasar gambaran permasalahan di atas maka dapat dikatakan bahwa *data mart* mampu membantu permasalahan mengenai banyak dan beragamnya data sumber yang dimiliki Universitas Brawijaya. *Data mart* dapat menyederhanakan dan mengagregasi data-data tersebut sehingga menjadi bentuk yang lebih sederhana dan siap digunakan sebagai bahan analisis. Sementara *materialized query table* dapat membantu efisiensi akses data sehingga dengan sumber daya yang lebih sedikit dapat meingkatkan kecepatan akses.

Karena pertimbangan di atas penulis menawarkan penelitian untuk mengimplementasikan *materialized query table* pada pembangunan *data mart.* Diharapkan dengan kombinasi keduanya mampu meningkatkan efektivitas dan efisiensi penyimpanan data sebagai sistem pendukung proses analisis data.

Dikarenakan adanya kerja sama antara Universitas Brawijaya dan IBM tentang kemudahan penggunaan teknologi IBM di lingkungan Universitas Brawijaya, maka penulis berniat untuk memanfaatkan teknologi-teknologi IBM untuk mendukung pelaksanaan penelitian ini. Salah satunya menggunakan sistem manajemen basis data IBM yaitu IBM DB2.

Berdasarkan latar belakang yang telah dikemukakan maka penulis mengusulkan judul penelitian pada skripsi ini yaitu "Implementasi *Materialized Query Table* Untuk Pembangunan *Data Mart* (Studi Kasus: Universitas Brawijaya Bagian Akademik)".

1.2 Rumusan masalah

Berdasarkan latar belakang yang dikemukakan di atas maka rumusan permasalahan pada penelitian ini antara lain:

1. Menggali dan menganalisa kebutuhan pengguna *data mart* di Universitas Brawijaya Bagian Akademik.
2. Merancanga dan membangun *data mart* dengan mengimplementasikan *materialized query table* berdasarkan hasil penggalian dan analisa kebutuhan.
3. Menguji dampak implementasi *materialized query table* terhadap performa dan ketahanan *data mart* yang telah dibangun.

1.3 Tujuan

Adapun tujuan dari penelitian ini adalah membangun *data mart* untuk Universitas Brawijaya Bagian Akademik yang mengimplementasikan *materialized query table* dan menguji dampak implementasi *materialized query table* terhadap *data mart*.

1.4 Manfaat

Manfaat penelitian yang diharapkan dicapai dalam penelitian ini adalah sebagai berikut:

1. Bagi penulis, meningkatkan wawasan mengenai perancangan dan pembangunan *data mart* serta pengimplementasian *materialized query table*.
2. Bagi Universitas Brawijaya, dapat meningkatkan efektifitas dan efisiensi analisis data akademik dan mendukung proses pengambilan keputusan di level manajerial.
3. Bagi Masyarakat Umum, penelitian ini diharapkan dapat menjadi sumber wawasan mengenai pentingnya peran *data mart* dan *materialized query table* pada sebuah proses analisis.
4. Bagi Ilmu Pengetahuan, penelitian ini diharapkan dapat menjadi referensi bagi pengembangan ilmu terkait.

1.5 Batasan masalah

Berdasarkan latar belakang dan rumusan masalah yang telah diuraikan di atas, terdapat beberapa batasan masalah pada penelitian ini diantaranya:

1. Proses penggalian kebutuhan dilakukan di Universitas Brawijaya Bagian Akademik sebagai studi kasus.

2. Penelitian tidak membahas mengenai mekanisme *refresh* data yang dibutuhkan oleh *materialized query table.*
3. Penelitian dilakukan menggunakan sistem manajemen basis data IBM yaitu IBM DB2.

1.6 Sistematika pembahasan

Sistematika penulisan skripsi ini adalah antara lain sebagai berikut:

BAB I: Pendahuluan

Bab ini berisi tentang latar belakang masalah, rumusan masalah dan pokok-pokok bahasan, batasan masalah, tujuan dan manfaat dari penelitian serta sistematika penulisan skripsi.

BAB II : Kajian Pustaka dan Dasar Teori

Bab ini berisi tinjauan pustaka dari beberapa penelitian berkaitan sebelumnya serta landasan teori dilakukannya penelitian mengenai *data warehouse*, *data mart*, *materialized query table* dan lainnya.

BAB III : Metodologi Penelitian

Bab ini berisi penjelasan dan metodologi yang akan digunakan pada penelitian ini meliputi: studi literatur, penggunaan dasar teori, analisis perancangan sistem, *tools* yang digunakan dalam penelitian serta metode pengujian yang digunakan pada penelitian ini.

BAB IV : Analisis dan Perancangan Sistem

Bab ini berisi penjelasan analisis yang telah dilakukan dan melakukan perancangan sistem.

BAB V : Implementasi

Bab ini berisi implementasi perancangan sistem yang telah dilakukan sesuai metodologi yang ditetapkan.

BAB VI : Pengujian

Bab ini berisi hasil pengujian efisiensi dan efektivitas implementasi *materialized query table* untuk pembangunan *data mart*.

BAB VII : Penutup

Dalam bab ini berisi tentang kesimpulan dan saran dari sistem yang berhubungan dengan permasalahan yang telah dibahas serta tindakan yang harus di ambil atas hasil penelitian.

BAB 2 LANDASAN KEPUSTAKAAN

2.1 Kajian Pustaka

Kajian pustaka pada penelitian ini membahas beberapa penelitian terdahulu terkait dengan *data mart* dan implementasi *materialized query table* (*materialized view*) untuk optimasi performa basis data.

Pandya dan Shah (2014) pada penelitiannya yang berjudul "*Proposed Local Data Mart Approach For Data Warehouse Architecture*" mencoba mengusulkan arsitektur *data warehouse* menggunakan pendekatan pengembangan *data mart* lokal. Metode penelitiannya adalah dengan membandingkan metode pembangunan *data warehouse* dengan pendekatan *top bottom* dan *bottom up* sehingga mengusulkan pendekatan lain yaitu pendekatan *data mart* lokal. Hasil penelitiannya menyimpulkan bahwa dengan membangun *data mart* lokal terlebih dahulu sebelum membangun *data warehouse* perusahaan terbukti memiliki beberapa keunggulan yaitu: data mudah dikelola dan dimengerti pada tingkat lokal, mudah menyelesaikan permasalahan kualitas data dikarenakan data telah mengalami segementasi, mudah dilakukan pelacakan sumber data dan memberikan fleksibilitas dalam menghadapi perubahan kebutuhan.

Chhabra dan Pahwa (2014) pada penelitiannya yang berjudul "*Data Mart Designing and Integration Approach*" mencoba melakukan integrasi *data mart* baik yang ada di satu organisasi maupun yang berada di organisasi berbeda. Penelitiannya mencoba mengintegrasi *data mart* melalui tiga cara yaitu: integrasi melalui saling berbagi dimensi, integrasi melalui dimensi yang sesuai dan integrasi melalui generalisasi dimensi. Kesimpulan dari penelitiannya adalah bahwa *data mart* sangat lah mungkin untuk di integrasi satu sama lain, bahkan di lain kesempatan sangat dibutuhkan integrasi untuk memenuhi kompleksitas proses bisnis suatu organisasi yang seringkali berkaitan dengan proses bisnis organisasi lain.

Lawrence dan Rau-Chaplin (2008) pada penelitiannya yang berjudul "*Dynamic View Selection fo OLAP*" yang membahas mengenai pemilihan *view* mana saja yang akan di materialisasi untuk OLAP. Penelitiannya mengatakan bahwa penggunaan *materialized view* atau *materialized query table* akan sangat meningkatkan performa akses data untuk analisis. Namun tantangan besarnya adalah menentukan *view-view* mana saja yang sebaiknya di materialisasi. Sehingga penelitiannya fokus pada pengembangan algoritma untuk melakukan pemilihan *view* secara otomatis.

Berdasarkan penelitian-penelitian yang dijabarkan di atas maka penulis dapat menyimpulkan bahwa dengan membangun *data mart* maka proses analisis akan menjadi semakin mudah. Selain itu *data mart* dapat juga menjadi fondasi untuk dibangunnya sebuah *data warehouse* perusahaan. Jika dikombinasikan dengan *materialized query table* maka proses analisis akan dapat dilakukan dengan lebih cepat dan efisien.

2.2 Data *Warehouse*

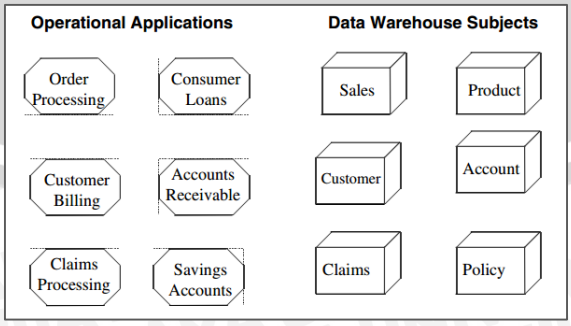
Data Warehouse (DW) pada dasarnya adalah penyimpanan data historis terintegrasi dari beberapa sumber data yang dipersiapkan untuk kepentingan analisis sehingga mampu menjadi komponen pendukung pengambilan keputusan (Turban, et al., 2010). *Data Warehouse* ada untuk menjawab pertanyaan pelaku bisnis akan bisnisnya meliputi performa bisnis dari berbagai operasi yang dilakukan, tren bisnis yang dialami perusahaan dan apa yang dapat dilakukan untuk meningkatkan keberhasilan bisnis. *Data warehouse* memungkinkan para pelaku bisnis untuk mengakses data secara langsung, menyediakan satu versi yang utuh mengenai indikator performa bisnis dan juga mendukung pengkajian data dari berbagai sudut pandang. Proses *data warehouse* terbilang sangat sederhana: Ambil semua data yang sudah dimiliki perusahaan, lakukan proses pembersihan dan transformasi sesuai kebutuhan dan setelahnya disimpan untuk menyediakan informasi strategis yang dibutuhkan (Ponniah, 2001).

Data warehouse bukanlah sekedar sebuah produk *hardware* ataupun *software*, melainkan sebuah lingkungan komputasi dimana pengguna dapat menemukan informasi-informasi strategis dan lingkungan dimana pengguna bersentuhan secara langsung dengan data yang mereka butuhkan untuk membuat keputusan yang lebih baik (Ponniah, 2001). Sehingga dapat dikatakan bahwa *data warehouse* merupakan sistem yang berpusat pada kebutuhan pengguna (*User Centric System*).

2.2.1 Karakteristik *Data Warehouse*

Cara umum untuk mengenali *data warehouse* dapat dilihat dari karakteristik dasarnya yaitu (Turban, et al., 2010):

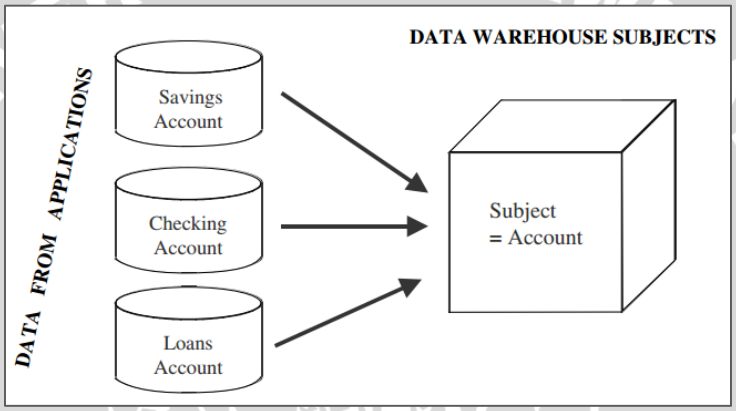
1. Berorientasi Subjek
   Data diorganisir berdasarkan subjek yang mendetail seperti penjualan, produk, konsumen dan hanya berisi mengenai informasi-informasi yang relevan untuk mendukung pengambilan keputusan. Hal ini tentu sangat berbeda dengan basis data operasional yang cenderung berorientasi kepada produk. Seperti yang diilustrasikan Gambar 2.1 berikut:



Gambar 2.1 Perbedaan Orientasi Basis Data Operasional dan Data Warehouse (Ponniah, 2001)

2. Terintegrasi

Untuk melakukan pengambilan keputusan maka diperlukan kajian pada banyak data yang biasanya berasal dari berbagai macam sumber. Sumber data ini tidak hanya berasal dari basis data yang berbeda bahkan juga mencakup data-data mentah dan segmen-segmen data yang ada di *flat file*. Selain itu sumber-sumber data tersebut acap kali memiliki format pendataan sesuai kebutuhan masing-masing sehingga menimbulkan banyak sekali disparitas format dan penamaan pada sumber data. Sehingga sebelum disimpan pada *data warehouse* data-data dari sumber yang beragam itu harus melalui serangkaian proses transformasi, konsolidasi dan integrasi untuk satu-kesatuan informasi strategis. Berikut adalah Gambar 2.2 yang mengilustrasikan proses integrasi pada subjek *data warehouse* (Ponniah, 2001).



Gambar 2.2 Proses Integrasi Data Warehouse (Ponniah, 2001)

3. Bervariansi Waktu

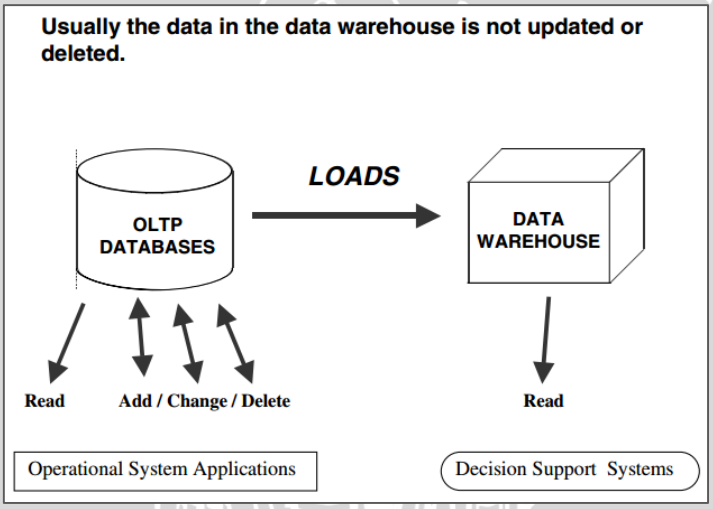
Dikarenakan data yang ada pada *data warehouse* dimaksudkan sebagai bahan analisis dan pengambilan keputusan, maka diperlukan serangkaian data dari masa lalu sampai yang terbaru (data historis). Sebagai contoh jika ingin menganalisa mengenai pola-pola pembelian maka data yang dibutuhkan tidak hanya data pembelian yang baru saja dilakukan namun juga data riwayat pembelian dari beberapa tahun sebelumnya. Sehingga setiap struktur data yang ada pada data warehouse pasti mengandung elemen waktu (Ponniah, 2001). Sebagai contoh jika pada *data warehouse* terdapat unit penjualan maka setiap baris data mengenai penjualan pasti berelasi pada waktu tertentu. Bergantung pada seberapa detail yang dibutuhkan setiap baris data dapat direlasikan sesuai tanggal, minggu, bulan bahkan tahun yang spesifik.

Dengan karakteristik ini maka *data warehouse* memiliki kemampuan untuk:

- Memungkinkan analisa masa lalu
- Merelasikan Informasi yang ada pada saat ini
- Membuat prakiraan untuk kepentingan masa depan

4. *Nonvolatile*

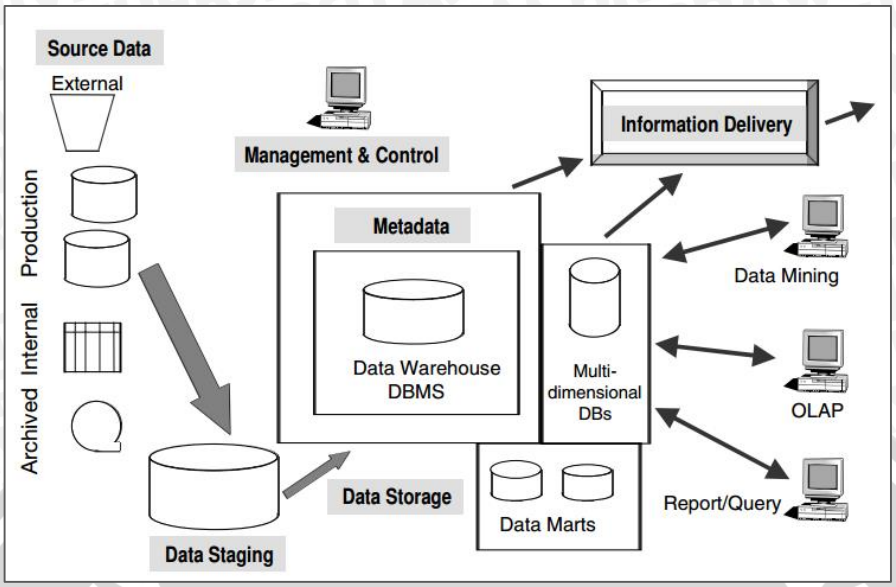
Data dari sistem operasional dipindahkan ke dalam *data warehouse* pada interval-interval terntentu sesuai kebutuhan bisnis. Bahkan perpindahan data pada setiap unit dapat berbeda menyesuaikan proses bisnis yang berjalan. Misalnya untuk memperbaharui data pada unit penjualan dilakukan 2 kali sehari sementara pada unit produksi dilakukan 1 minggu sekali dan lain sebagainya. Karena itu jika proses bisnis melakukan transaksi maka data tidak akan langsung disimpan di *data warehouse* melainkan disimpan dulu di sistem OLTP (*Online Transaction Proscessing*) setelah itu pada interval tertentu data akan dipindahkan melalui serangkaian proses ke *data warehouse*. Proses pemindahan data dari sistem operasional ini tidak akan mengubah data yang ada di *data warehouse* melainkan hanya akan menambahkannya saja sehingga data yang ada di *data warehouse* tidak akan pernah mengalami perubahan (*Nonvolatile*) berbeda dengan data yang ada pada sistem operasional yang akan selalu berubah-ubah untuk mendukung berjalannya proses bisnis (Ponniah, 2001). Seperti diilustrasikan pada Gambar 2.3 berikut:



Gambar 2.3 Gambaran Sifat Nonvolatile Data Warehouse (Ponniah, 2001)

2.2.2 Komponen Utama *Data Warehouse*

Beragam dan bervariasinya model bisnis perusahaan-perusahaan yang mengimplementasikan *data warehouse*, berdampak pada beragamnya arsitektur *data warehouse* yang dibangun menyesuaikan dengan kebutuhan proses bisnis perusahaan tersebut. Namun betapapun *data warehouse* dibangun untuk perusahaan yang besar maupun kecil komponen dasar penyusun *data warehouse* tetaplah sama. Masing-masing perusahaan dapat memperkuat peran dari salah satu atau lebih komponen menyesuaikan kebutuhan. Berikut adalah Gambar 2.4 yang mengilustrasikan interaksi komponen-komponen tersebut :



Gambar 2.4 Interaksi Komponen Utama Data Warehouse (Ponniah, 2001)

Dengan melihat lebih dalam tiap-tiap komponen arsitektur *data warehouse* maka akan memperjelas peranan dan fungsi tiap-tiap komponen agar dapat memaksimalkan dan mengefisiensikan proses *data warehouse* menyesuaikan kebutuhan tiap-tiap perusahaan. Berikut komponen-komponen arsitektur data warehouse:

1. Sumber Data

   Sumber data untuk data warehouse digolongkan menjadi beberapa kategori:

   a. Data Produksi
      Data kategori ini berasal dari beragam aktivitas operasional perusahaan biasanya dari sistem OLTP. Tantangan terbesar untuk data produksi dikarenakan oleh disparitas data yang mungkin didapatkan dari berbagai sistem operasional yang berbeda dalam perusahaan. Sehingga diperlukan standarisasi, transformasi dan integrasi data-data tersebut menjadi data yang lebih berkualitas untuk disimpan dalam *data warehouse*.

   b. Data Internal
      Adalah data-data yang terdokumentasi secara internal oleh masing-masing departemen atau bagian dalam perusahaan yang tidak terhubung secara langsung dengan sistem perusahaan. Bisa berupa *spreadsheet* internal, laporan, profil pelanggan bahkan sampai basis data internal departemen. Data internal akan menambah kompleksitas pada proses transformasi dan integrasi dikarenakan beragamnya format data dan media penyimpanan.

   c. Data Arsip
      Adalah data historis yang secara rutin disimpan dan diarsip di tempat berbeda. Seperti yang diketahui bahwa data warehouse menyimpan data historis sehingga diperlukan proses untuk konversi dan pengambilan data-data arsip perusahaan secara berkala.

d. Data Eksternal

Seperti namanya data eksternal adalah data yang berasal dari luar perusahaan. Data-data ini diperlukan untuk menutupi kekurangan data yang dimiliki internal perusahaan. Sebagai contoh data pergerakan saham dari bursa efek bagi perusahaan investasi. Data kategori ini tentu saja tidak memiliki format sesuai dengan perusahaan sehingga diperlukan transformasi dan integrasi lebih lanjut.

2. Pemroses Data

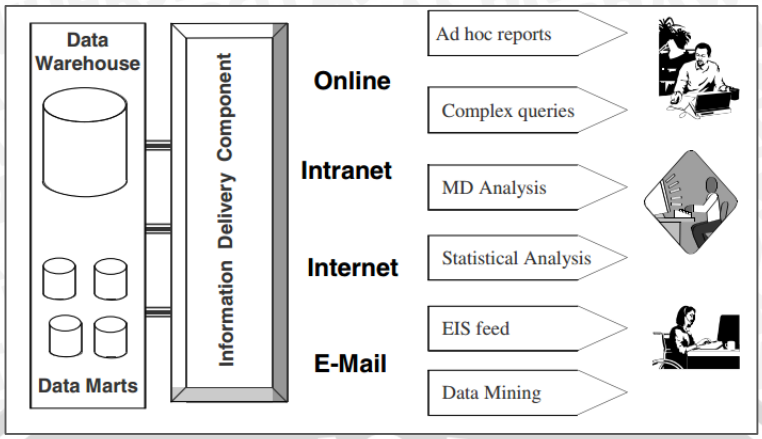
Pemroses data adalah komponen yang berperan untuk menyiapkan, mentransformasi dan mengintegrasikan data dari berbagai sumber yang berbeda untuk siap dalam format yang sama untuk disimpan di *data warehouse*. Keberadaan komponen akan sangat menentukan kualitas data yang akan disimpan dalam *data warehouse* sehingga diperlukan komponen terpisah agar proses dapat terenkapsulasi tanpa mengganggu performa keseluruhan sistem.

3. Penyimpanan Data

Penyimpanan data yang digunakan oleh *data warehouse* adalah penyimpanan yang benar-benar berbeda dengan penyimpanan yang digunakan oleh sistem operasional perusahaan. Penyimpanan data sistem operasional dioptimalkan untuk menyimpan data-data transaksional yang ternormalisasi untuk pemrosesan data yang cepat dan efisien. Data akan sangat cepat mengalami perubahan dalam hitungan menit bahkan detik sehingga kecepatan penulisan data akan sangat vital. Sementara *data warehouse* membutuhkan penyimpanan data historis dengan volume besar untuk kepentingan analisis. Data pada *data warehouse* tidak akan terlalu sering berubah namun dengan interval tertentu bertambah dengan volume besar begitu juga dengan pengambilan datanya. Sehingga diperlukan penyimpanan yang terpisah agar kedua sistem tidak saling mempengaruhi masing-masing performa

4. Penyampaian Informasi

Dengan beragamnya pengguna dan sistem yang membutuhkan data dari *data warehouse* maka diperlukan beragam metode untuk menyediakan dan menyampaikan informasi dari *data warehouse* (Ponniah, 2001). Ilustrasi penyampaian informasi oleh *data warehouse* dijelaskan pada Gambar 2.5 berikut:



Gambar 2.5 Penyampaian Informasi Data Warehouse

Berikut penjelasannya:

a. Laporan *da-hoc* adalah laporan yang telah didefinisikan sebelumnya dalam sistem dan ditujukan oleh pengguna-pengguna praktis yang membutuhkan informasi secara umum.

b. Kompleks *query*, analisis multidimensional dan analisis statistika adalah metode untuk mengakses *data warehouse* menggunakan *query-query* yang lebih kompleks demi kepentingan analisis untuk memenuhi kebutuhan analisis bisnis.

c. *EIS(Executive Information Systems)* adalah informasi yang dikhususkan untuk top eksekutif perusahaan untuk mendukung proses pengambilan keputusan.

d. *Data mining* adalah metodologi untuk membantu menemukan pola dan tren dari penggunaan data yang dimiliki sehingga dapat membantu proses pengambilan keputusan.

e. Tidak hanya jenis aksesnya yang berbeda bahkan media pengaksesnya pun beragam mulai dari *online*, internet, intranet dan email

5. *Metadata*

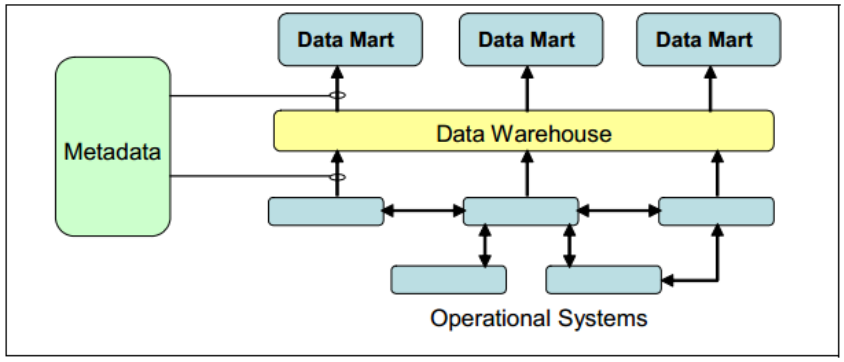
   Adalah komponen yang berperan melakukan proses katalog struktur data yang ada pada *data warehouse*. Dapat juga disebut kamus data yang menyimpan struktur logis dari data yang ada di *data warehouse*.

6. Kontrol dan Manajemen

   Komponen ini dapat dikatakan komponen yang amat penting dibandingkan komponen yang lain dikarenakan komponen inilah yang memfasilitasi komponen-komponen untuk saling berinteraksi satu sama lain. Komponen ini bekerja bersama *metadata* untuk melakukan pemantauan terhadap komponen lain berdasarkan informasi yang disediakan oleh *metadata.*

2.2.3 Arsitektur Data *Warehouse*

Sebagai suatu sistem yang dikembangkan untuk mendukung adanya analisis secara mendalam dan intens, maka dibutuhkan suatu arsitektur yang mampu memenuhi dan mendukung kebutuhan tersebut. Secara umum arsitektur *data warehouse* terdiri dari 3 lapisan yaitu *data mart*, *data warehouse* dan *metadata*. Adapun ilustrasinya digambarkan pada Gambar 2.6 berikut:



Gambar 2.6 Arsitektur Data Warehouse (Ballard, et al., 2005)

Tiga lapisan ini dapat di kembangkan menjadi beberapa lapisan sesuai kebutuhan dari bisnis masing-masing. Ada beberapa alternatif arsitektur yang dapat implementasikan diantaranya (Turban, et al., 2010) :

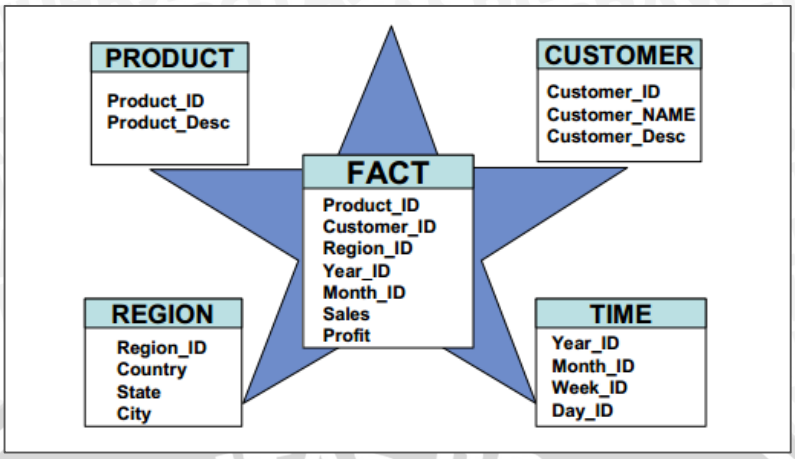
1. *Data Mart* Independen
2. *Data Mart* Saling Terkoneksi
3. *Hub* dan *Spoke*
4. Tersentralisasi
5. *Federated*

2.2.4 Pemodelan Data Multidimensional

Pemodelan data multidimensional merupakan teknik perancangan logis untuk mengorganisir dimensi-dimensi bisnis dan matriks yang dianalisis pada dimensi-dimensi tersebut (Ponniah, 2001). Secara sederhana model data multidimensional akan terdiri dari struktur data yang dibutuhkan untuk merepresentasikan dimensi bisnis yang ada. Model multidimensional yang umum digunakan untuk *data warehouse* adalah (Ballard, et al., 2005):

1. *Star Schema*

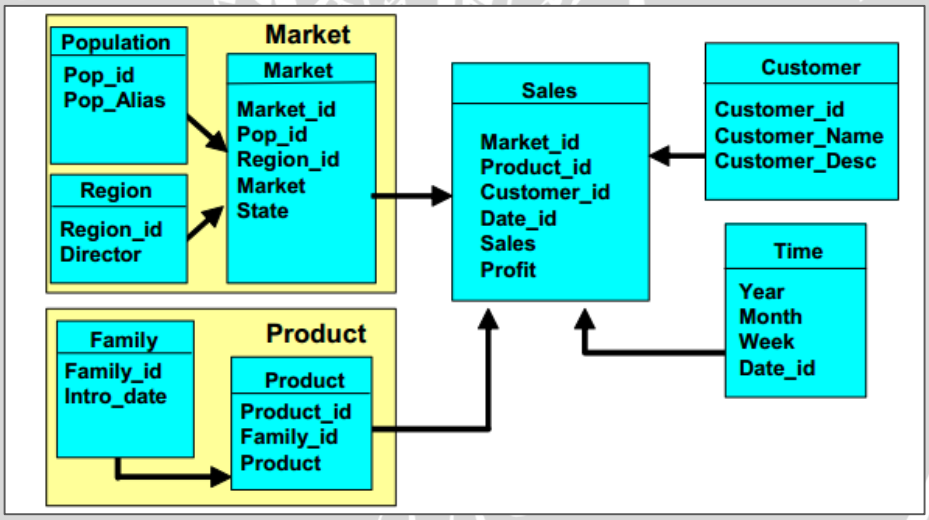
   Merupakan model data yang memiliki pendekatan berbeda dibandingkan model data relasional. Terdiri dari sebuah tabel besar yang terdiri dari kumpulan fakta (*fact table*) dan dikelilingi oleh beberapa tabel yang memuat data-data deskriptif dari fakta tersebut yang sering disebut tabel dimensi. Ketika digambarkan maka akan menyerupai bentuk dari bintang seperti Gambar 2.7 di bawah ini :



Gambar 2.7 Star Schema (Ballard, et al., 2005)

2. *Snowflake Schema*

Merupakan pengembangan dari *star schema* yang mengalami proses normalisasi lebih. Hal ini dilakukan untuk mencapai efisiensi data sehingga meminimalkan redundansi. Walaupun terkadang diperlukan biaya *join*. Adapun ilustrasi skemanya digambarkan pada Gambar 2.8 berikut ini:



Gambar 2.8 Snowflake Schema (Ballard, et al., 2005)

3. *Galaxy Schema*

Merupakan model data yang berasal dari berbagai kombinasi *star schema* dan *snowflake schema.* Sehingga memungkinkan dilakukan *dimension sharing* atau saling berbagi penggunaan dimensi antar *fact table* dan meningkatkan konsistensi dan efisiensi data.

2.2.5 Proses ETL (Extract, Transform dan Load)

Proses ETL merupakan proses utama *data warehouse* atau *data mart*. Proses inilah yang menentukan kualitas data yang nantinya disimpan di *data warehouse* atau *data mart*. Seringkali disebut sebagai *warehousing process*. Fungsi dari proses

ETL adalah mendapatkan data-data relevan yang kemudian di transformasi untuk disimpan di *data warehouse* sebagai informasi yang berguna (Ponniah, 2001).

1. Ekstraksi Data (*Extract*)
   Berfungsi untuk mengumpulkan dan mendapatkan data dari beragam sumber data. Dikarenakan banyak dan beragamnya sumber data maka diperlukan pendekatan berbeda untuk tiap sumber data. Sumber data mungkin sebuah basis data relasional, *data mart*, *data warehouse* lain atau bahkan sekumpulan *flat file*.

2. Transformasi Data (*Transform*)
   Pada setiap implementasi sistem pasti memiliki format data standar masing-masing. Sehingga beragamnya sumber data berarti beragamnya standar yang digunakan. Untuk dapat disimpan di *data warehouse* maka data dari berbagai sumber tersebut haruslah melalui serangkaian proses transformasi agar sesuai dengan format data yang diterapkan oleh *data warehouse*.

3. Penyimpanan Data (*Load*)
   Penyimpanan data ke *data warehouse* tidaklah sesederhana seperti penyimpanan data sistem OLTP. Pada proses penyimpanan data pertama kali akan ada data dengan volume yang sangat besar dalam sekali waktu disimpan. Sementara pada iterasi berikutnya secara data dalam skala yang cukup besar secara berkala akan ditambahkan. Untuk dapat melakukan proses tersebut secara berulang-ulang tentu saja dibutuhkan konfigurasi dan penanganan yang seksama agar proses berjalan efektif dan efisien.

2.3 Data Mart

Saat *data warehouse* mengintegrasi dan mengombinasikan data dari seluruh bagian perusahaan, *data mart* merupakan sebuah irisan dari *data warehouse* yang lebih berfokus pada satu subjek saja atau bagian yang ada di suatu perusahaan seperti marketing dan operasional (Turban, et al., 2010). Dengan sifatnya yang fokus pada suatu subjek data mart tidak diharuskan untuk dapat digunakan oleh subjek yang lainnya hal ini menjadikan *data mart* ter enkapsulasi dari lingkungan perusahaan. Sehingga aktifitas-aktifitas yang dilakukan di suatu *data mart* tidak akan mempengaruhi *data mart* yang lain. Hal inilah yang membuat *data mart* menjadi tempat bagi sebagian besar proses analisis untuk mendukung pengambilan keputusan.

2.3.1 Tipe *Data Mart*

Pada dasarnya *data mart* dapat dikelompokkan menjadi dua macam (Ballard, et al., 2005):

1. Dependen
   Adalah *data mart* yang merupakan irisan dari *data warehouse* perusahaan yang mana data-datanya didapatkan dari ekstraksi *data warehouse* perusahaan tersebut. Sehingga terjamin integritas dan konsistensi datanya di

seluruh *data warehouse* perusahaan. Namun untuk mengembangkan *data mart* ini dibutuhkan pengembangan *data warehouse* perusahaan terlebih dahulu sehingga membutuhkan lebih banyak waktu dan biaya.

2. Independen
   Adalah *data mart* yang berdiri sendiri di luar dari *data warehouse* perusahaan dan data-datanya berasal dari luar *data warehouse* perusahaan. Biasanya data berasal dari sistem operasional atau OLTP lokal sehingga integritas dan konsistensi data di seluruh perusahaan patut dipertanyakan. Namun dengan ruang lingkup yang lebih kecil dan tanpa membutuhkan dukungan dari *data warehouse*, *data mart* tipe ini lebih mudah dan murah untuk dikembangkan.

2.3.2 Pembangunan *Data Mart*

Berdasarkan penggalian dan analisis kebutuhan yang telah dilakukan maka akan dilakukan perencanaan sistem *data* mart. Adapun langkah-langkah perencanaan sistem akan dilakukan berdasarkan sembilan langkah yang diajukan oleh *Ralph Kimball* (Prabhu, 2008):

1. Pemilihan Proses Bisnis
   Melakukan pemilihan proses bisnis yang dibutuhkan dan mengacu pada subjek *data mart*. Pada langkah ini akan menentukan proses bisnis apa saja yang akan digunakan oleh *data mart* beserta apa pengukuran keberhasilannya.

2. Menentukan *Grain*
   Merupakan langkah untuk menentukan seberapa detail yang direpresentasikan oleh sebuah baris pada tabel fakta. Hanya dengan menentukan detail ini maka akan didapatkan dimensi-dimensi yang mendukung sebuah tabel fakta.

3. Mengidentifikasi Dimensi Bisnis
   Secara sederhana dimensi bisnis merupakan ukuran-ukuran yang diterapkan pada suatu bisnis dalam menjalankan aktifitasnya. Dimensi dapat juga dikatakan merupakan perbendaharaan kepentingan perusahaan. Dengan identifikasi dimensi yang baik akan membuat *data mart* mudah dimengerti dan digunakan.

4. Pemilihan Fakta
   Grain yang ditentukan pada langkah sebelumnya merupakan patokan dalam menentukan apa saja dan seberapa detail fakta yang dapat digunakan dalam *data mart*. Semua fakta yang ada di *data mart* harus selaras dengan *grain* yang mendefinisikannya. Tidak diperkenankan ada *grain* yang berbeda pada satu fakta. Tabel fakta nantinya akan menyimpan pengukuran-pengukuran numerik sesuai dengan *grain* yang mendefinisikannya.

5. Penyimpanan Hasil Kalkulasi pada Tabel Fakta

Setelah tabel memilih tabel fakta yang akan digunakan maka tabel fakta haruslah diperiksa ulang untuk mengidentifikasi adakah perlunya penyimpanan hasil kalkulasi yang berkaitan pada tabel fakta.

6. Mendefinisikan Tabel-tabel Dimensi
   Setelah tabel fakta didefinisikan maka pada tahap ini dilakukan pengecekan ulang pada tabel-tabel dimensi. Hal ini dilakukan untuk memastikan hierarki dan detail deskripsi dari dimensi sesuai *grain* tabel fakta yang berhubungan.

7. Menentukan Durasi Basis Data dan Rutinitas Pembaharuan Data
   Durasi basis data menentukan seberapa jauh ke belakang data yang ada di basis data disimpan. Sebagai contoh sebuah basis data toko *retail* dapat menyimpan data operasional sampai 5 tahun ke belakang. Durasi basis data ini memungkinkan dilakukannya analisis historis pada data. Selain itu diperlukan pula penentuan seberapa sering pembaharuan data pada tiap tabel dilakukan.

8. Mengidentifikasi Dimensi-dimensi yang Memiliki Perubahan Lamban
   Pada umumnya dimensi pada *data mart* jarang sekali mengalami perubahan. Hal ini dapat dicontohkan pada dimensi fakultas yang mana nama fakultas akan jarang sekali atau hampir tidak pernah mengalami perubahan. Namun, hal ini tidak menutup kemungkinan terjadinya perubahan di masa mendatang. Untuk itu diperlukannya adanya identifikasi dimensi-dimensi yang memiliki perubahan lambat. Dan ditentukan pula bagaimana cara mengatasi perubahan tersebut. Menurut kimball ada 3 cara untuk melakukannya yaitu: menimpa data (*overwrite*), menambahkan data dengan *key* yang sama dan menambahkan kolom untuk melacak perubahan.

9. Menentukan Prioritas dan Mode *Query*
   Setelah menyelesaikan langkah 1 sampai 8 maka rancangan logis *data mart* dapat dikatakan telah selesai. Pada tahap ini yang dilakukan adalah menerjemahkan rancangan logis tersebut menjadi rancangan fisik. Rancangan ini utamanya fokus pada urutan fisik data disimpan pada *hardisk*. Meskipun terlihat sepele namun urutan data ini memegang peranan penting pada performa *data mart*. Dengan penempatan urutan yang tepat maka dapat dihasilkan *query* yang lebih cepat dan efisien.

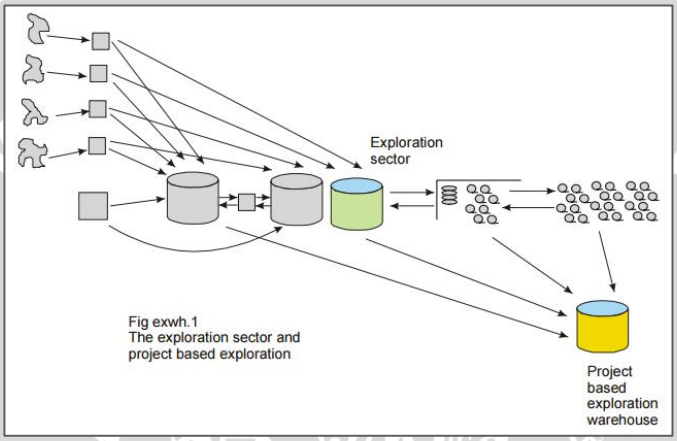
2.3.3 Pengaplikasian *Data Mart*

Dengan semakin beragamnya proses analis yang dibutuhkan oleh sebuah bisnis maka penerapan dan pengaplikasian *data mart* juga menjadi semakin luas, berikut diantaranya (Imhoff, et al., 2003) :

1. OLAP Data Mart
   Data mart didukung dengan sistem manajemen basis data multidimensional atau yang sering disebut *star schema* sangatlah mendukung pengoperasian aktivitas OLAP. Dengan ruang lingkup yang lebih sempit, kebutuhan yang tidak berubah-ubah, *query* yang cenderung mudah diprediksi, waktu respons yang

cepat serta kemudahan untuk menghasilkan laporan membuat data mart menjadi primadona dalam melakukan OLAP.

2. Eksplorasi Data Warehouse
   Selain digunakan sebagi pendukung analisis pada suatu subjek/unit/bagian tertentu dikarenakan mudahnya untuk dibangun, *data mart* juga dapat dijadikan alat untuk melakukan analisis eksplorasi data dalam menemukan hubungan antar dua data yang terpisah. Setelah analisis menghasilkan sesuatu yang berguna maka hasil analisis tersebut akan diformalisasikan melalui pengembangan *data mart* yang lain. Berikut Gambar 2.9 mengilustrasikan proses eksplorasi *data warehouse* berlangsung :



Gambar 2.9 Proses Ekplorasi Data Warehouse (Imhoff, et al., 2003)

3. Data Mining
   Karena sifatnya yang terenkapsulasi dari lingkungan luar *data warehouse* maka pengguna dapat memanfaatkan *data mart* sepenuhnya untuk menganalisa dan menggali hubungan antar data lebih mendalam tanpa harus takut aktifitasnya akan mempengaruhi performa sistem *data warehouse* secara keseluruhan.

4. Pengembangan Aplikasi Khusus
   Dengan bisnis yang berjalan semakin cepat dan berubah-ubah maka tentu kebutuhan akan aplikasi untuk membantu berjalannya proses bisnis atau menganalisa proses binsis yang sudah berjalan akan semakin beragam. Di sinilah *data mart* berperan besar dengan sifatnya yang mudah dibuat dan diubah sesuai kebutuhan maka pengembang akan semakin mudah untuk menghadirkan atau menambahkan fungsionalitas baru tanpa mempengaruhi kinerja sistem secara keseluruhan.

2.4 *Materialized Query Table*

View pada basis data merupakan sebuah objek virtual. Maksudnya *view* hanyalah sebuah serangkaian *query* yang disimpan dan setiap digunakan maka akan melakukan komputasi ulang pada basis data. Sementara *materialized view* atau *materialized query table* adalah objek basis data yang tidak hanya

menyimpan serangkaian perintah *query* namun berikut hasil *query*-nya juga. Sehingga keuntungan utama dari *materialized query table* adalah kemampuan untuk memanfaatkan hasil *query* yang telah disimpan tanpa perlu melakukan komputasi ulang pada *query* di basis data (Paraboschi, et al., 2003). Istilah *materialized query table* dan *materialized view* sebenarnya adalah dua istilah yang berbeda namun mengacu pada objek yang sama. Istilah *materialiazed view* merupakan istilah yang digunakan oleh produk produk buatan ORACLE sementara istilah *materialized query table* merupakan istilah yang digunakan oleh produk IBM. Dikarenakan penelitian ini menggunakan teknologi IBM maka selanjutnya istilah *materialized query table* lah yang akan digunakan.

2.4.1 Meminimalkan Biaya

Dalam merancang desain *materialized query table* ada beberapa batasan yang diberikan oleh sistem. Dimana batasan ini tidak akan terlihat oleh pengguna berikut diantaranya (Theodoratos, et al., 2009):

1. Biaya Evaluasi *Query*
   Adalah total biaya yang dibutuhkan *basis data* untuk mengevaluasi seluruh *query* yang digunakan secara menyeluruh maupun parsial pada *materialized query table*. Jumlah total ini juga memiliki skala yang mengindikasikan frekuensi atau prioritas dari sebuah *query*.

2. Biaya Pemeliharaan *View*
   Adalah jumlah total biaya yang digunakan untuk membuat perubahan dan pemeliharaan pada *materialized query table*. Tiap indikator mengindikasikan seberapa sering frekuensi perubahan dan perubahan sumber data yang berkaitan.

3. Biaya Operasional
   Menimalkan biaya operasional sebuah *materialized query table* merupakan sebuah dilema. Biaya operasional merupakan kombinasi antara biaya evaluasi *query* dan biaya pemeliharan *view*. Untuk meminimalkan biaya pemeliharaan *view* dapat dicapai dengan melakukan replikasi relasi sumber data yang ada di *data warehouse*. Namun hal itu akan meningkatkan biaya evaluasi *query*, karena *query* harus dievaluasi di tiap sumber data. Untuk meminimalkan biaya evaluasi *query*, dapat dicapai dengan mematerialisasi seluruh *view* yang ada sehingga semua *query* yang masuk langsung mengakses pada *view*. Namun dengan banyaknya *view* secara otomatis biaya pemeliharaan akan meningkat drastis. Untuk itu untuk mencapai biaya operasional yang optimal diperlukan adanya keseimbangan antara biaya pemeliharaan *view* dan biaya evaluasi *query*.

4. Total Ukuran dari *View*
   Pada kasus *data warehouse* terdistribusi yang sering menjadi permasalahan adalah data transmisi per waktu yang melalui jaringan. Sehingga pada kasus ini perancang akan lebih tertarik untuk meminimalkan ukuran *materialized query table* untuk menjawab *query* yang masuk.

2.4.2 Aljabar Relasional

Untuk mempermudah proses perancangan *materialized query table* maka akan digunakan bantuan aljabar relasional. Aljabar relasional merupakan bahasa *query* prosedural yang terdiri dari serangkaian operasi dengan masukan satu atau beberapa relasi dan menghasilkan relasi lain sebagai keluaran (Silberschatz, et al., 2009).

ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

Gambar 2.10 Contoh Relasi Instructor

Gambar 2.10 merupakan contoh bentuk relasi atau pada basis data dapat disebut tabel. Sebuah relasi terdiri dari kolom dan *tule* (baris). Aljabar relasional memiliki operasi dasar yaitu *select*, *project*, *union*, *set difference*, *cartessian product* dan *rename*. Serta operasi tambahan yaitu *set intersection*, *natural join* dan *assignment*. Berikut penjelasan masing-masing operasi:

1. Operasi *Select*
   Operasi ini akan melakukan seleksi data per baris dengan kondisi yang telah ditentukan sebelumnya. Operasi ini dilambangkan dengan karakter latin sigma (σ). Jadi jika dari relasi *instructor* pada Gambar 2.10 ingin dipilih instruktur yang memiliki departemen "*History*" maka operasinya dapat dituliskan:

   $$\sigma_{dept\_name="History"}(instructor)$$

   Secara umum operasi perbandingan yang dapat digunakan adalah '=', '<', '<=', '>', '>=' dan '!='. Sementara untuk menggabungkan 2 atau lebih operasi dapat menggunakan penghubung *and* (^), *or* (ᵛ) dan *not* (¬).

2. Operasi *Project*
   Adalah operasi untuk melakukan seleksi kolom tertentu dari sebuah relasi. Operasi ini dilambangkan dengan karakter latin *Pi* (∏). Sebagai contoh jika ingin memilih kolom *ID* dan *name* dari relasi *instructor* maka operasi dapat dituliskan:

   $$\Pi_{ID,\ name}(instructor)$$

3. Operasi *Union*

   Adalah operasi yang menggabungkan dua relasi atau *result set* yang memiliki kolom yang sama. Operasi dilambangkan dengan karakter (U). Sebagai contoh jika ingin memilih baris yang memiliki departemen *History* maka dapat dituliskan:

   $$\sigma_{dept\_name="History"}(instructor)$$

   Jika ingin memilih baris yang memiliki departemen *Finance* maka dapat dituliskan:

   $$\sigma_{dept\_name="Finance"}(instructor)$$

   Dan jika ingin menggabungkan keduanya maka dapat dituliskan dengan:

   $$\sigma_{dept\_name="History"}(instructor) \cup \sigma_{dept\_name="Finance"}(instructor)$$

4. Operasi Set Difference

   Adalah operasi untuk melakukan pencarian tupel yang hanya berada pada satu relasi. Operasi dilambangkan dengan karakter minus (-). Sebagai contoh jika ingin mencari instruktur yang memiliki departemen "*History*" dengan gaji 62000 dapat dituliskan:

   $$\Pi_{id}(\sigma_{dept\_name="History"\wedge salary=62000}(instructor)) -$$
   $$\Pi_{id}(\sigma_{dept\_name="History"\wedge salary>60000}(instructor))$$

5. Operasi Cartessian Product

   Adalah operasi yang mengombinasikan informasi dari dua relasi. Operasi dilambangkan dengan karakter cross (x). Sebagai contoh jika ingin mengetahui instruktur yang memiliki departemen "Physics" menempa mata kuliah apa dapat dituliskan:

   $$\Pi_{name,\ course\_id}(\sigma_{instructor.ID=teaches.ID}((\sigma_{dept\_name="Physics"}(instructor)) \times teaches))$$

6. Operasi *Rename*

   Adalah operasi untuk memberi nama pada bagian relasi. Operasi dilambangkan dengan Greek letter rho kecil (ρ). Sebagai contoh jika ingin memberi nama instruktur pada relasi instructor dapat dituliskan:

   $$\rho_{NamaInstruktur/name}(instructor)$$

7. Operasi Set Intersection

   Adalah operasi untuk memilih data yang menjadi bagian dari dua relasi atau lebih. Operasi dilambangkan dengan karakter (∩). Sebagai contoh jika ingin mencari instruktur yang mengajar mata kuliah "Music" dan "Design" dapat dituliskan:

   $$\Pi_{id}(\sigma_{course\_name="Music"}(teaches)) \cap \Pi_{id}(\sigma_{course\_name="Design"}(teaches))$$

8. Operasi Natural Join

   Adalah operasi *binary* untuk mengombinasikan pilihan tertentu dengan Cartessian Product ke dalam satu operasi. Operasi dilambangkan dengan simbol join (⋈). Sebagai contoh jika ingin mencari nama semua instruktur beserta nama id kursus yang mereka tempa dapat ditulis:

$$\Pi_{\text{id, course\_id}}(\text{instructor} \bowtie \text{teaches})$$

9. Operasi Assignment

   Adalah operasi untuk menampung hasil operasi pada variabel sementara. Operasi dilambangkan dengan simbol (←). Sebagai contoh penulisan:

$$\text{temp1} \leftarrow \text{instructor x teaches}$$

$$\text{temp2} \leftarrow \sigma\ (\textit{temp}1)$$

$$\text{Result} = \Pi_{R\,U\,S}(\text{temp2})$$

2.4.3 Batasan Berorientasi Sistem

Dalam merancang desain *materialized query table* ada beberapa batasan yang diberikan oleh sistem. Dimana batasan ini tidak akan terlihat oleh pengguna berikut diantaranya (Theodoratos, et al., 2009):

1. Batasan Ruang Penyimpanan
   Walaupun kian hari biaya perangkat penyimpanan kian murah. Namun kita tidak bisa mengasumsikan bahwa ruang penyimpanan akan menjadi tak terbatas. Sehingga diperlukan pertimbangan untuk mengelola ruang penyimpanan yang dimiliki *server*.

2. Batasan Pemeliharaan *View*
   Seringkali yang menjadi permasalahan bukan merupakan ruang penyimpanan namun biaya pemeliharaan. *Data Warehouse* acapkali melakukan perubahan pada periode tertentu dengan volume yang sangat besar. Sehingga jika ditambah biaya untuk mengelola *view* yang tinggi maka beban *server* dan jaringan akan sangat berat. Untuk itu diperlukannya pertimbangan untuk mengelola biaya pemeliharaan *view*.

3. Kemampuan Self Maintanability
   Adalah kemampuan *materialized query table* untuk mengelola dirinya sendiri seperti kapan harus melakukan perubahan dan sebagainya. Kemampuan bertujuan untuk meringankan beban pengoperasian oleh pengguna dan dalam beberapa kasus dapat mengoptimalkan pengoperasian *materialized query table*. Namun apabila tidak dirancang dengan baik maka hanya akan membebani kinerja sistem yang berjalan.

4. Menjawab Masukan *Query* Secara Eksklusif Menggunakan *MQT*
   Pendekatan ini berarti mengubah seluruh masukan *query* untuk secara langsung mengakses pada *materialized query table*. Hal ini tentu akan sangat

menguntungkan disisi sumber data yang tidak akan berinteraksi secara langsung. Namun biaya dari segi akses data juga tidak ringan.

2.4.4 Batasan Berorientasi Pengguna

Dalam desainnya, *materialized query table* juga tak lupa harus selalu mempertimbangkan kebutuhan pengguna yang memiliki beberapa batasan diantaranya (Theodoratos, et al., 2009) :

1. Batasan Biaya Hasil Data
   Merupakan hubungan antara waktu yang dibutuhkan antara waktu untuk menjawab *query* pengguna dan waktu perubahan dilakukan.
2. Batasan Waktu Respons *Query*
   Adalah batasan dimana waktu respons *query* tidak boleh melebihi sebuah ukuran waktu. Dikarenakan jika terlalu lama maka pengguna akan kecewa dengan performa sistem.

2.5 Pengujian

Pengujian merupakan serangkaian proses untuk memastikan pembangunan *data warehouse* telah memenuhi kebutuhan yang direncanakan. Dapat dikatakan proses pengujian ini merupakan proses dasar yang menentukan keberhasilan pembangunan *data warehouse*. Berbeda dengan pengujian perangkat lunak yang berfokus pada kode program, pengujian *data warehouse* lebih fokus pada data dan informasi. Dapat dikatakan pengujian *data warehouse* bertujuan untuk memastikan kebenaran informasi yang disampaikan ke pengguna dengan melibatkan data dengan volume yang besar (Golfarelli & Rizzi, 2011).

2.5.1 Pengujian Performa (*Performance Testing*)

Merupakan pengujian yang dilakukan untuk memastikan performa sistem memenuhi kriteria pada kondisi normal. Pada *data warehouse* pengujian performa akan mengirimkan serangkaian grup *query* ke *frontend* sistem secara bersamaan untuk mengetahui berapa lama waktu yang dibutuhkan untuk memprosesnya. Kondisi normal pada pengujian performa meliputi dari lingkungan sistem, besar data sampai jumlah pengguna yang mengakses sistem pada waktu yang sama.

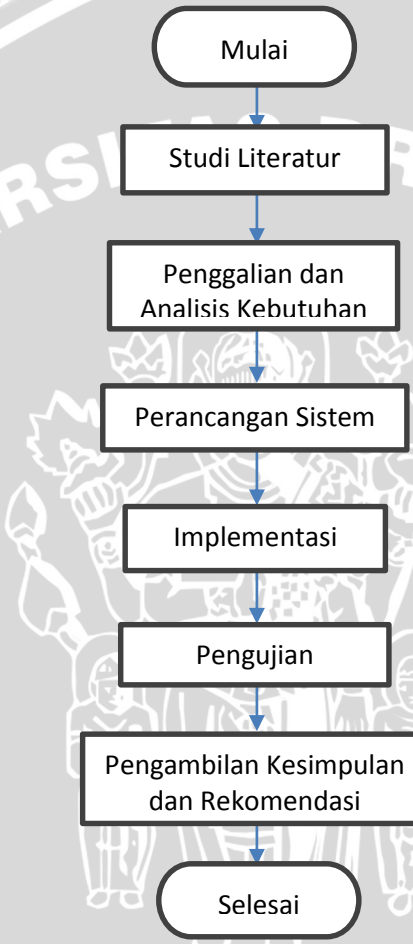
2.5.2 Pengujian Daya Tahan (*Stress Testing*)

Merupakan pengujian untuk mengetahui seberapa baik performa sistem dalam kondisi puncak beban akses dan data. Pengujian ini akan secara bertahap meningkatkan jumlah *query* dan pengguna yang mengakses sistem. Pada setiap peningkatan akan dicatat berapa lama waktu sistem memproses dan peningkatan akan dilakukan terus sampai sistem benar-benar tidak mampu menangani permintaan lagi.

BAB 3 METODOLOGI PENELITIAN DAN PERANCANGAN

3.1 Metode Penelitian

Berisikan tentang metode penelitian yang dilakukan selama penelitian skripsi. Alur metode penelitian ini dapat dijelaskan pada Gambar 3.1



Gambar 3.1 Alur Metodologi Penelitian

3.1.1 Studi Literatur

Tahap studi literatur merupakan tahap pengumpulan referensi dari buku, *e-book*, atau jurnal untuk memperoleh penjelasan tentang teori yang mendukung penelitian. Dari hasil studi literatur yang dilakukan, terdapat beberapa teori yang mendukung penelitian, antara lain:

1. Penelitian yang berkaitan sebelumnya
2. *Data Warehouse*
3. *Data Mart*
4. *Materialized Query Table* atau *Materialized View*

3.1.2 Penggalian dan Analisis Kebutuhan

Untuk mengetahui seperti apa sistem yang dibutuhkan oleh pengguna maka akan dilakukan proses penggalian kebutuhan dan analisis kebutuhan yang lebih detailnya sebagai berikut:

1. Penggalian Kebutuhan
   Tahap penggalian kebutuhan merupakan tahap dimana penulis menggali kebutuhan dari sistem. Adapun tahap penggalian kebutuhan akan menggunakan dua cara yaitu wawancara dan observasi. Wawancara yang dilakukan menggunakan teknik wawancara semi terstruktur, dimana pertanyaan dimulai dari pertanyaan khusus yang telah disiapkan dan dilanjutkan dengan penggalian lebih lanjut oleh pewawancara. Pada tahap ini penulis melakukan analisis kebutuhan dengan mewawancarai pengguna sistem. Sementara observasi yang dilakukan merupakan observasi tidak langsung, dimana peneliti tidak terlibat dalam proses bisnis yang sedang berjalan tapi hanya melakukan pengamatan terhadap proses bisnis yang berlangsung pada Universitas Brawijaya Bagian Akademik. Pengamatan ini bertujuan untuk mengumpulkan data terkait kebutuhan pengguna yang belum terpenuhi dari proses bisnis lama yang telah berjalan. Hasil wawancara dan observasi terlampir pada LAMPIRAN A.
2. Analisa Kebutuhan
   Setelah penggalian kebutuhan pengguna, maka akan dilakukan proses analisis untuk merumuskan kebutuhan pengguna akan *data mart* dengan lebih runtut dan terstruktur. Analisis ini diperlukan untuk mempermudah penerjemahan bahasa pengguna menjadi bahasa lebih teknis sehingga memudahkan untuk dilakukan perancangan sistem.

3.1.3 Perancangan Sistem

Berdasarkan penggalian dan analisis kebutuhan yang telah dilakukan maka akan dilakukan proses perancangan sistem *data mart*. Adapun proses ini akan terbagi menjadi beberapa sub proses yaitu perancangan arsitektur, perancangan basis data OLTP, perancangan data penelitian, perancangan *data mart*, perancangan proses ETL dan perancangan MQT.

3.1.3.1 Perancangan Arsitektur

Perancangan arsitektur sistem dilakukan untuk memetakan bagaimana nantinya arsitektur sistem diimplementasikan. Perancangan ini akan berbentuk gambaran bagaimana hubungan jaringan antara *server* OLTP, *server data mart*, *materialized query table* dan *client*.

3.1.3.2 Perancangan Basis Data OLTP

Dikarenakan penulis tidak mempunyai akses pada sistem OLTP yang berjalan saat ini maka diperlukan perancangan basis data OLTP berdasarkan hasil penggalian dan analisis kebutuhan. Perancangan basis data akan dilakukan dengan

membuat rancangan ERD yang merujuk pada penelitian yang dilakukan sebelumnya.

3.1.3.3 Perancangan Data Penelitian

Salah satu komponen utama dari *data mart* adalah data. Untuk menghasilkan hasil analisis yang berkualitas membutuhkan data yang berkualitas pula. Untuk itu pada tahap ini diperlukan adanya perancangan data penelitian berupa data *dummy*. Data *dummy* dipilih dikarenakan pada penelitian ini menitik beratkan pada sisi volume dan frekuensi data bukan pada validitas data. Namun meskipun begitu, demi mensimulasikan kondisi data di lapangan maka dalam penelitian ini terdapat beberapa data yang asli. Data asli seperti data fakultas, jurusan, program studi, jenjang didapat dari Dikti serta data mata kuliah didapat dari Universitas Brawijaya Bagian Pusat Informasi dan Keluhan. Data lainnya berupa data *dummy* di bentuk dengan bantuan program *Microsoft Office Excel* dan bahasa pemrograman *Java*.

3.1.3.4 Perancangan *Data Mart*

Perancangan *data mart* dilakukan dengan metodologi 9 langkah yang diajukan oleh *Kimball*. Adapun langkah langkahnya yaitu

1. Pemilihan Proses Bisnis
   Melakukan pemilihan proses bisnis yang dibutuhkan oleh Universitas Brawijaya Bagian Akademik dalam *data mart*.
2. Menentukan *Grain*
   Menentukan seberapa detail proses bisnis yang telah dipilih dari Bagian Akademik akan disimpan pada tabel fakta.
3. Menentukan Dimensi Bisnis
   Melakukan identifikasi dimensi-dimensi apa saja yang digunakan Universitas Brawijaya Bagian Akademik untuk mendukung *grain* yang telah ditentukan pada langkah sebelumnya.
4. Pemilihan Fakta
   Menentukan fakta-fakta yang dibutuhkan Universitas Brawijaya Bagian Akademik yang berpatokan pada *grain* yang telah ditentukan.
5. Penyimpanan Hasil Kalkulasi pada Tabel Fakta
   Melakukan analisis terhadap tabel fakta yang telah didefinisikan sebelumnya adakah hasil kalkulasi lain yang terkait pada tabel fakta tersebut.
6. Mendefinisikan Tabel-tabel Dimensi
   Setelah tabel fakta selesai didefinisikan maka dilakukan proses pendefinisian tabel-tabel dimensi yang berkaitan dengan tabel fakta serta hierarki, deskripsi dan detail yang menyertainya.
7. Menentukan Durasi Basis Data dan Rutinitas Pembaharuan Data

Memutuskan durasi dari data yang akan disimpan pada tabel. Yang mana durasi tiap tabel dapat berbeda dari satu sama lain.

8. Mengidentifikasi Dimensi-dimensi yang Memiliki Perubahan Lamban
   Pada umumnya dimensi pada *data mart* sering kali mengalami perubahan. Hal ini dapat dicontohkan pada dimensi fakultas yang mana nama fakultas akan jarang sekali atau hampir tidak pernah mengalami perubahan. Namun, hal ini tidak menutup kemungkinan terjadinya perubahan di masa mendatang. Untuk itu diperlukannya adanya identifikasi dimensi-dimensi yang memiliki perubahan lambat. Dan ditentukan pula bagaimana cara mengatasi perubahan tersebut. Menurut *kimball* ada 3 cara untuk melakukannya yaitu: menimpa data (*overwrite*), menambahkan data dengan *key* yang sama dan menambahkan kolom untuk melacak perubahan.
9. Menentukan Prioritas dan Mode *Query*
   Melakukan proses transformasi rancangan logis menjadi rancangan fisik. Untuk melakukan optimasi pada *query* yang nantinya akan digunakan

3.1.3.5 Perancangan Proses ETL

Proses perancangan ETL dilakukan dengan memetakan sumber dan proses transformasi data sebelum data disimpan pada *data mart*. Untuk membantu mempermudah proses pemetaan tersebut penulis menggunakan program *IBM Design Studio*.

3.1.3.6 Perancangan Materialized Query Table (MQT)

Perancangan MQT dilakukan untuk mempermudah dalam pemilihan *query* untuk diimplementasikan sebagai MQT. Perancangan dilakukan dengan bantuan aljabar relasional.

3.1.4 Implementasi Sistem

Implementasi sistem akan didasarkan pada hasil rancangan yang telah dibuat pada proses perancangan sistem. Sehingga tahapan-tahapan implementasi dapat dikatakan hampir sama dengan tahapan-tahapan di perancangan yaitu :

1. Implementasi Arsitektur
   Arsitektur sistem akan diimplementasikan menggunakan teknologi virtualisasi. Dengan menggunakan perangkat lunak *VMWare*.
2. Implementasi Basis Data OLTP
   Implementasi dilakukan dengan mengeksekusi DDL (*Data Definition Languae*) dari skema basis data ke *server* basis data OLTP. Pada implementasi ini penulis menggunakan sistem basis data IBM DB2
3. Pembuatan Data Penelitian
   Merupakan proses pembuatan data serta penyimpanan data ke basis data OLTP. Proses pembuatan menggunakan bantuan program *Ms Office Excel,*

bahasa pemrograman Java dan fasilitas *import* yang dimiliki sistem basis data IBM DB2.

4. Implementasi *Data Mart*
   Data Mart diimplementasikan dengan cara mengeksekusi DDL (*Data Definition Language*) di *server data mart*.

5. Proses ETL
   Implementasi Proses ETL dilakukan dengan men-*deploy* rancangan proses ETL yang dibuat di *IBM Data Designer* ke *IBM Info Sphere*. Setelah itu dilakukan konfigurasi penjadwalan pada proses ETL.

6. Implementasi MQT
   Proses implementasi MQT dilakukan dengan menerjemahkan aljabar relasional yang telah dibuat menjadi SQL (*Structured Query Language*). Setelah itu dilakukan pendeployan ke *server data mart*.

3.1.5 Pengujian

Pengujian sistem dilakukan untuk menguji dampak penggunaan *materialized query table* pada pembangunan *data mart* terhadap performa *query* data dan kehandalannya (*relibialiaty*). dan performa sistem yang telah diimplementasikan pada tahap sebelumnya. Pengujian dilakukan dengan dua cara yaitu pengujian performa (*performance test*) dan pengujian daya tahan (*stress test*).

3.1.5.1 Pengujian Performa (Performance *Test*)

Pengujian ini memiliki dua skenario yaitu melakukan eksekusi *query* untuk mendapatkan lama waktu eksekusi dan menganalisis *query access plan* untuk mendapatkan berapa banyak sumber daya yang digunakan. Sumber daya yang digunakan akan diukur dalam satuan *timeron* yang merupakan satuan pengukuran yang digunakan oleh *IBM DB2* untuk menghitung biaya *CPU* dan IO *query*. Untuk eksekusi *query* pengujian ini menggunakan bantuan perangkat lunak *apache-jmeter* sementara menghitung *timeron* akan dibantu *tools query visual explain* yang dimiliki *IBM Design Studio*. Pengujian ini akan membandingkan antara *query* konvensional dengan MQT.

3.1.5.2 Pengujian Daya Tahan (*Stress Test*)

Pengujian ini memiliki skenario untuk menjalankan *query* secara simultan oleh beberapa pengguna. Untuk menguji batas daya tahan *query* jumlah pengguna akan ditingkatkan secara bertahap sampai sistem tidak mampu menangani permintaan lagi. Pengujian ini akan dibantu oleh perangkat lunak *apache-jmeter*. Sama seperti pengujian performa pada pengujian ini akan membandingkan antara *query* konvensional dan MQT.

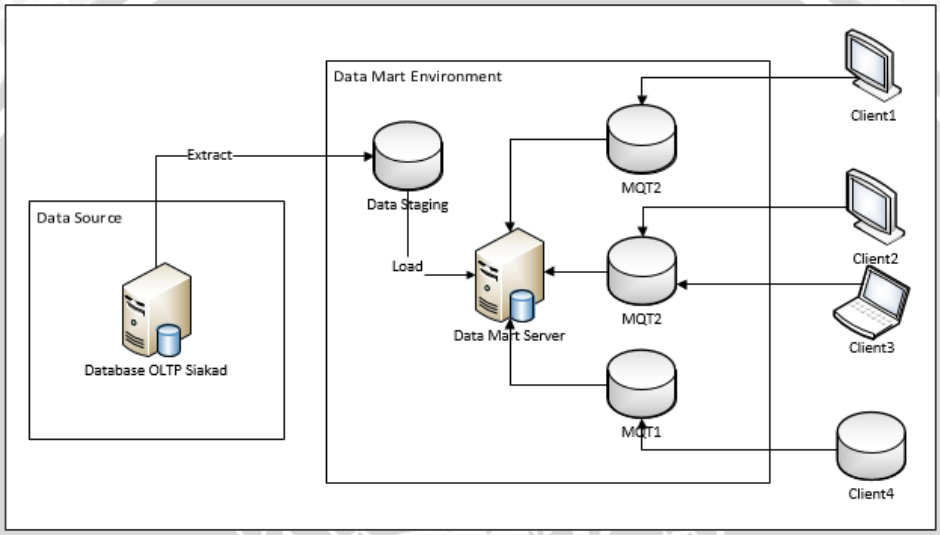
3.1.6 Pengambilan Kesimpulan dan Rekomendasi

Setelah tahap pengujian dilakukan maka akan disusun sebuah grafik yang merepresentasikan hasil pengujian dari kedua sistem sehingga akan dapat ditarik

suatu kesimpulan akan pengaruh penerapan *materialized query table* pada pembangunan *data mart*. Setelah itu akan disusun pula rekomendasi untuk pengembangan pada penelitian lebih lanjut.

3.2 Perancangan Arsitektur

Pada penelitian ini perancangan arsitektur *data mart* didasarkan pada arsitektur *data warehouse*. Sehingga terdapat beberapa komponen utama yaitu sumber data, pemroses data (*data staging*), penyimpanan data dan penyampaian informasi. Sementara untuk *meta data* dan manajemen kontrol akan ditangani oleh sistem basis data yang digunakan.



Gambar 3.2 Rancangan Arsitektur Data Mart

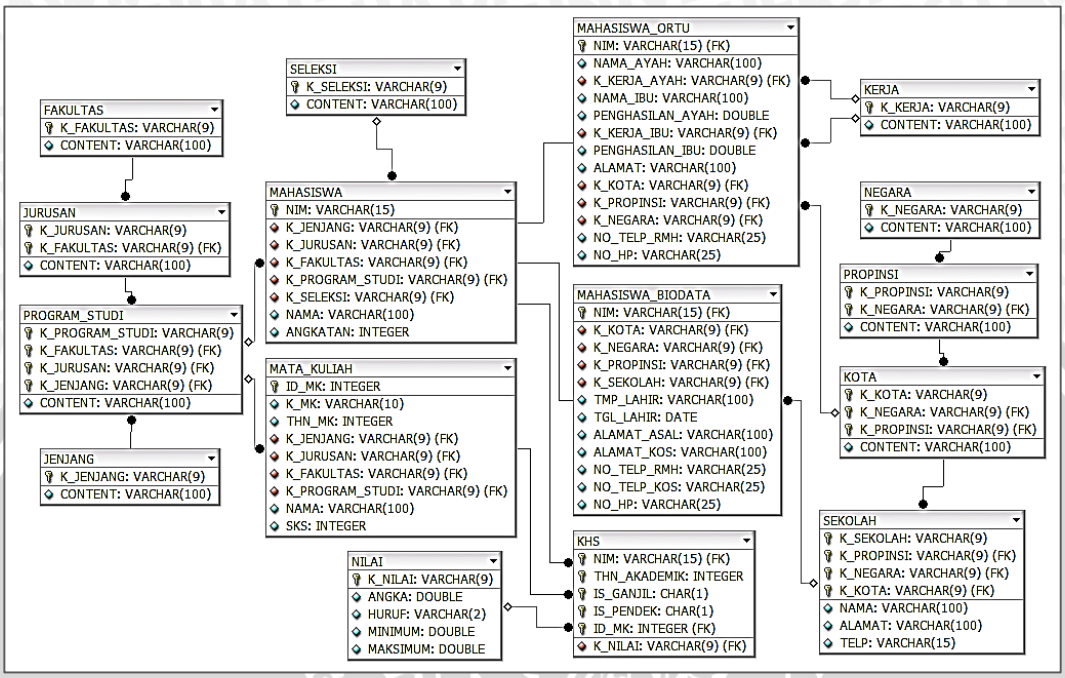
Gambar 3.2 menunjukkan rancangan arsitektur sistem *data mart* yang nantinya akan diimplementasikan. Sesuai dengan gambar rancangan arsitektur tersebut pertama-tama akan dibuat virtualisasi sumber data dari sistem OLTP Siakad. Kemudian data dari sistem OLTP Siakad diekstraksi ke dalam *data staging* untuk dilakukan transformasi. Setelah data telah memenuhi kriteria *data mart* maka data akan disimpan pada *server data mart*. Untuk memanfaatkan data maka diperlukan jembatan *MQT* untuk melakukan agregasi dan perhitungan pada *data mart*.

Secara singkat proses perancangan arsitektur dapat dijalankan seperti:

1. Ekstraksi data dari sumber data yang pada penelitian ini menggunakan sistem OLTP Siakad.
2. Lakukan transformasi data pada data yang telah di ekstrak pada sistem *data staging* yang nantinya akan menggunakan *IBM Websphere Application Server*.
3. Simpan data yang telah ditransformasi pada *data mart*.
4. Buat *Materialized Query Table* (MQT) sebagai jembatan agregasi dan penghitungan data.
5. Hasil data dapat langsung diolah sebagai bahan analisis maupun sebagai data sumber bagi *data warehouse* perusahaan.

3.3 Perancangan Basis Data OLTP

Rancangan sistem basis data OLTP yang digunakan dalam penelitian ini merupakan hasil adaptasi dari penelitian sebelumnya yaitu basis data SIAKAD Universitas Brawijaya (Chandramitasari, 2014).



Gambar 3.3 Rancangan ERD Basis Data Siakad (Chandramitasari, 2014)

Gambar 3.3 menunjukkan rancangan ERD yang dibuat untuk basis data SIAKAD. Pada ERD tersebut terdapat beberapa tabel yaitu: Tabel Fakultas, Tabel Jurusan, Tabel Program Studi, Tabel Jenjang, Tabel Seleksi, Tabel Mahasiswa, Tabel Mata kuliah, Tabel Orang Tua Mahasiswa, Tabel Biodata Mahasiswa, Tabel Kerja, Tabel Negara, Tabel Propinsi, Tabel Kota, Tabel Asal Sekolah, Tabel Nilai, dan Tabel KHS.

3.4 Perancangan Data Penelitian

Salah satu komponen utama dari *data mart* adalah data. Untuk menghasilkan hasil analisis yang berkualitas membutuhkan data yang berkualitas pula. Untuk itu pada tahap ini diperlukan adanya perancangan data penelitian berupa data *dummy*. Data *dummy* dipilih dikarenakan pada penelitian ini menitik beratkan pada sisi volume dan frekuensi data bukan pada validitas data. Namun meskipun begitu, demi menyerupai kondisi data di lapangan maka dalam penelitian ini terdapat beberapa data yang asli. Data-data tersebut diantaranya adalah data fakultas, jurusan, jenjang, program studi dan mata kuliah. Data mata kuliah didapat dari Bagian Pusat Informasi Dokumentasi dan Keluhan Universitas Brawijaya. Sementara yang lain berdasarkan data yang terdaftar di Dikti (Direktorat Jendral Pendidikan Tinggi Republik Indonesia, 2015). Adapun detail data fakultas, program studi dan jumlah mahasiswa dapat dilihat pada LAMPIRAN C.

3.5 Perancangan *Data Mart*

Pada tahap ini akan dilakukan proses perancangan model multidimensional untuk diterapkan pada *data mart*. Adapun analisis kebutuhan dan perancangan sistem *data mart* pada penelitian ini didasarkan pada metode sembilan langkah yang diusulkan oleh *Kimball*.

3.5.1 Pemilihan Proses Bisnis

Pada suatu instansi pendidikan, didalamnya tentu saja banyak terdapat berbagai proses bisnis yang menopang keberlangsungan instansi tersebut. Proses-proses bisnis tersebut tentu memiliki tingkat prioritas dan skala yang berbeda. Untuk itu dalam langkah ini dilakukan pemilihan proses bisnis yang benar-benar membutuhkan dukungan *data mart*. Sehingga *data mart* dapat dibangun dengan efektif dan efisien.

Tabel 3.1 Proses Bisnis yang Dipilih

Proses Bisnis	Deskripsi
Penerimaan Mahasiswa Baru	Merupakan proses yang berjalan tiap tahun sekali pada setiap perguruan tinggi. Proses ini bertujuan untuk merekam dan memantau perkembangan penerimaan mahasiswa baru berdasarkan sekolah dan daerah asal.
Pemantauan Status Mahasiswa	Merupakan proses pemantauan bagaimana mahasiswa aktif tersebar di seluruh fakultas dan jurusan. Proses ini bertujuan untuk mengkaji perkembangan fakultas dan jurusan berdasarkan mahasiswa aktif dan tidak didalamnya.
Pemantauan Perkembangan Indeks Prestasi Mahasiswa	Merupakan proses pemantauan perekembangan proses akademik mahasiswa berdasarkan indeks prestasi di tiap-tiap fakultas dan jurusan. Sehingga dapat digunakan sebagai acuan perkembangan akademik mahasiswa.

Tabel 3.1 diatas menunjukkan proses bisnis dari Universitas Brawijaya Bagian Akademik yang akan di akomodasi oleh *data mart*. Pemilihan ini berdasarkan hasil wawancara dengan pemangku kepentingan di Universitas Brawijaya Bagian Akademik serta diskusi dengan dosen pembimbing.

3.5.2 Penentuan *Grain*

Setelah proses bisnis ditentukan maka langkah berikutnya adalah menentukan seberapa detail level atomik data yang nantinya akan direpresentasikan di tabel fakta.

Tabel 3.2 Grain Fakta

Grain	Deskripsi	Proses Bisnis Terkait
Rekap mahasiswa aktif per semester sesuai sebarannya	Merepresentasikan jumlah mahasiswa yang berstatus aktif yang ada di fakultas dan jurusan pada tiap semester berdasarkan sebaran sekolah dan daerah asalnya	Pemantauan Status Mahasiswa
Entri kartu hasil studi mahasiswa	Merupakan nilai indeks prestasi mahasiswa per semester di fakultas dan jurusan masing-masing berdasarkan sebaran sekolah dan daerah asalnya	Pemantauan Perkembangan Indeks Prestasi Mahasiswa

Grain pada Tabel 3.2 diatas merupakan satuan terkecil baris yang akan disimpan di tabel fakta *data mart*. Sehingga hal ini memungkinkan dilakukannya pemrosesan data yang lebih beragam.

3.5.3 Menentukan Dimensi Bisnis

Langkah selanjutnya adalah menentukan dimensi-dimensi bisnis dari *grains* yang telah ditentukan sebelumnya.

Tabel 3.3 Dimensi-dimensi Bisnis

Dimensi	Deskripsi
Negara	Merupakan data negara asal mahasiswa atau calon mahasiswa
Provinsi	Merupakan data provinsi asal mahasiswa atau calon mahasiswa
Kota	Merupakan data kota asal mahasiswa atau calon mahasiswa
Asal Sekolah	Merupakan data sekolah asal mahasiswa atau calon mahasiswa
Seleksi	Merupakan metode seleksi pendaftaran calon mahasiswa di Universitas Brawijaya
Jenjang	Merupakan jenjang pendidikan yang ditempuh mahasiswa seperti diploma dan strata.
Mahasiswa	Data detail mahasiswa yang terdaftar di Universitas Brawijaya
Program Studi	Program studi yang terdaftar di Universitas Brawijaya
Jurusan	Jurusan yang terdaftar di Universitas Brawijaya
Fakultas	Fakultas yang terdaftar di Universitas Brawijaya
Periode Akademik	Merupakan periode akademik yang dilalui oleh mahasiswa tiap tahun dan semester yang bertipe genap, ganjil atau pendek.
Angkatan	Tahun angkatan pendaftaran mahasiswa baru
Mata Kuliah	Adalah detail mata kuliah yang terdaftar di universitas brawijaya

Dimensi-dimensi yang ada di Tabel 3.3 didapatkan dari menganalisa kebutuhan dimensi *grain* yang telah ditentukan sebelumnya.

3.5.4 Memilih Fakta

Langkah selanjutnya adalah melakukan pemilihan fakta sesuai *grain* yang telah didefinisikan sebelumnya. Pada langkah ini fakta juga akan dihubungkan dengan dimensi-dimensi terkait.

Tabel 3.4 Fakta yang Dipilih

Fakta	Deskripsi	Dimensi Terkait
Jumlah mahasiswa aktif per semester sesuai sebarannya	Merupakan fakta yang berisikan tentang seberapa banyak mahasiswa yang memiliki status aktif berdasarkan sebaran akademiknya, sekolah asal dan daerah asalnya.	Negara, Provinsi, Kota, Asal Sekolah, Jenjang, Seleksi, Mahasiswa, Program Studi, Jurusan, Fakultas Angkatan, Status Mahasiswa, Periode Akademik
Nilai indeks prestasi mahasiswa sesuai sebarannya	Merupakan fakta yang berisikan rekapan nilai indeks prestasi mahasiswa sesuai sebaran akademik, sekolah asal dan daerah asal	Negara, Provinsi, Kota, Asal Sekolah, Jenjang, Seleksi, Mahasiswa, Program Studi, Jurusan, Fakultas, Angkatan, Periode Akademik, Dosen, Mata Kuliah

Fakta-fakta yang pada Tabel 3.4 di atas merupakan pengukuran-pengukuran yang nantinya akan disimpan pada tabel fakta.

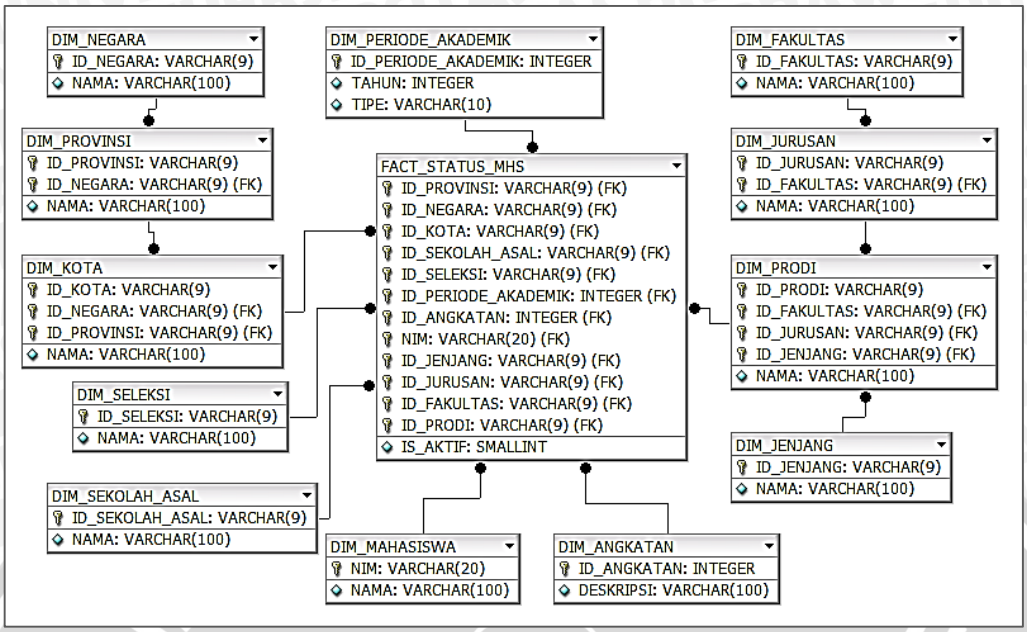
3.5.5 Penyimpanan Hasil Kalkulasi Pada Tabel Fakta

Setelah fakta diidentifikasi dan dipilih. Maka langkah selanjutnya adalah mengidentifikasi adanya hasil kalkulasi atau agregasi yang perlu disimpan pada tabel fakta. Namun pada studi kasus ini dikarenakan fakta menggunakan *grain* yang paling atomik sehingga tidak memungkinkan adanya hasil kalkulasi dan agregasi pada tabel fakta.

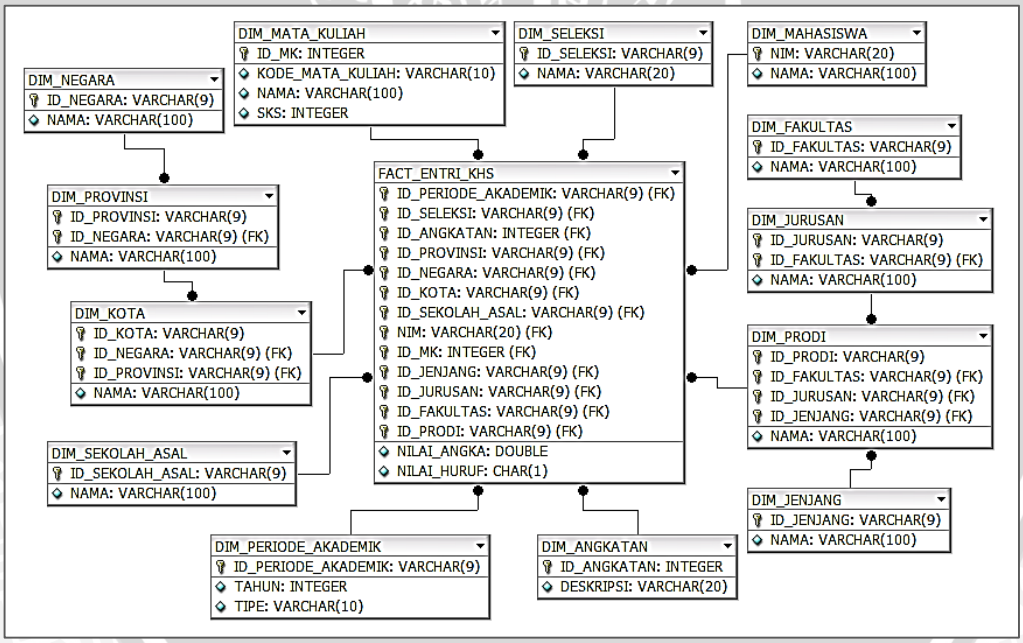
3.5.6 Mendefinisikan Tabel-tabel Dimensi

Langkah berikutnya adalah mendefinisikan isi dari dimensi-dimensi dan fakta di atas menjadi sebuah tabel. Untuk mempermudah pendefinisian tabel maka akan disusun ERD dengan menghubungkan antara tabel fakta dan tabel dimensi. ERD akan disusun menggunakan skema *snowflake* dengan dasar *grain* dan fakta yang dipilih sebelumnya.

Masing-masing *grain* akan direpresentasikan oleh satu gambar ERD. *Grain* "Rekap Mahasiswa Aktif" direpresentasikan oleh Gambar 3.4 dan *grain* "Entri Kartu Hasil Studi Mahasiswa" direpresentasikan oleh Gambar 3.5.



Gambar 3.4 Skema Snowflake yang Merepresentasikan Grain "Rekap Mahasiswa Aktif Sesuai Sebarannya"



Gambar 3.5 Skema Snowflake yang Merepresentasikan Grain "Entry Kartu Hasil Studi Mahasiswa"

Meskipun kedua *grain* digambarkan pada gambar yang berbeda namun seperti yang terlihat di Gambar 3.4 dan Gambar 3.5 masing-masing *grain* tersebut saling berbagi penggunaan dimensi. Hal ini akan membantu untuk menjaga integritas data. Adapun detail tabel-tabel diatas adalah sebagai berikut:

1. Tabel Dimensi Negara (DIM_NEGARA)

Tabel 3.5 Dimensi Negara

Atribut	Tipe Data	Panjang
ID_NEGARA	VARCHAR	9
NAMA	VARCHAR	100

Tabel 3.5 berisi detail atribut dari tabel dimensi negara. Sesuai namanya dimensi negara berfungsi untuk menyimpan detail informasi dari negara asal mahasiswa. Sebagai kunci identitas terdapat atribut ID_NEGARA dengan tipe data VARCHAR.

2. Tabel Dimensi Propinsi (DIM_PROPINSI)

Tabel 3.6 Dimensi Propinsi

Atribut	Tipe Data	Panjang
ID_PRONVINSI	VARCHAR	9
ID_NEGARA	VARCHAR	9
NAMA	VARCHAR	100

Tabel 3.6 di atas berisi detail atribut dari dimensi propinsi. Sesuai namanya dimensi ini berfungsi untuk menyimpan detail informasi propinsi asal mahasiswa. Dimensi ini memiliki relasi dengan dimensi negara dengan kunci penghubung adalah atribut ID_NEGARA.

3. Tabel Dimensi Kota (DIM_KOTA)

Tabel 3.7 Dimensi Kota

Atribut	Tipe Data	Panjang
ID_KOTA	VARCHAR	9
ID_PRONVINSI	VARCHAR	9
ID_NEGARA	VARCHAR	9
NAMA	VARCHAR	100

Tabel 3.7 berisi detail atribut dari dimensi kota. Dimensi ini ini memiliki relasi dengan dimensi propinsi dan dimensi negara dengan kunci penghubung adalah atribut ID_PROPINSI dan ID_NEGARA.

4. Tabel Dimensi Asal Sekolah (DIM_SEKOLAH_ASAL)

Tabel 3.8 Dimensi Asal Sekolah

Atribut	Tipe Data	Panjang
ID_SEKOLAH	VARCHAR	9

NAMA	VARCHAR	100

Tabel 3.8 berisi detail atribut dari dimensi asal sekolah. Sesuai namanya dimensi asal sekolah berfungsi untuk menyimpan detail informasi dari sekolah asal mahasiswa. Sebagai kunci identitas terdapat atribut ID_SEKOLAH dengan tipe data VARCHAR.

5. Tabel Dimensi Seleksi (DIM_SELEKSI)

Tabel 3.9 Dimensi Seleksi

Atribut	Tipe Data	Panjang
ID_SELEKSI	VARCHAR	9
NAMA	VARCHAR	100

Tabel 3.9 berisi detail atribut dari dimensi seleksi. Sesuai namanya dimensi seleksi berfungsi untuk menyimpan detail informasi dari jenis seleksi yang diambil mahasiswa. Sebagai kunci identitas terdapat atribut ID_SELEKSI dengan tipe data VARCHAR.

6. Tabel Dimensi Jenjang (DIM_JENJANG)

Tabel 3.10 Dimensi Jenjang

Atribut	Tipe Data	Panjang
ID_JENJANG	VARCHAR	9
NAMA	VARCHAR	100

Tabel 3.10 berisi detail atribut dari dimensi jenjang. Sesuai namanya dimensi jenjang berfungsi untuk menyimpan detail informasi dari jenjang yang diambil mahasiswa. Sebagai kunci identitas terdapat atribut ID_JENJANG dengan tipe data VARCHAR.

7. Tabel Dimensi Mahasiswa (DIM_MAHASISWA)

Tabel 3.11 Dimensi Mahasiswa

Atribut	Tipe Data	Panjang
NIM	VARCHAR	10
NAMA	VARCHAR	100

Tabel 3.11 berisi detail atribut dari dimensi mahasiswa. Sesuai namanya dimensi mahasiswa berfungsi untuk menyimpan detail informasi mengenai mahasiswa. Sebagai kunci identitas terdapat atribut NIM dengan tipe data VARCHAR.

8. Tabel Dimensi Program Studi (DIM_PRODI)

Tabel 3.12 Dimensi Program Studi

Atribut	Tipe Data	Panjang
ID_PRODI	VARCHAR	9
ID_JURUSAN	VARCHAR	9
ID_FAKULTAS	VARCHAR	9
NAMA	VARCHAR	100

Tabel 3.12 berisi detail atribut dari dimensi program studi. Dimensi ini memiliki relasi dengan dimensi jurusan dan dimensi fakultas dengan kunci penghubung adalah atribut ID_JURUSAN dan ID_FAKULTAS. Sebagai kunci identitas terdapat atribut ID_PRODI dengan tipe data VARCHAR.

9. Tabel Dimensi Jurusan (DIM_JURUSAN)

Tabel 3.13 Dimensi Jurusan

Atribut	Tipe Data	Panjang
ID_JURUSAN	VARCHAR	9
ID_FAKULTAS	VARCHAR	9
NAMA	VARCHAR	100

Tabel 3.13 berisi detail atribut dari dimensi jurusan. Dimensi ini memiliki relasi dengan dimensi fakultas dengan kunci penghubung adalah atribut ID_FAKULTAS. Sebagai kunci identitas terdapat atribut ID_JURUSAN dengan tipe data VARCHAR.

10. Tabel Dimensi Fakultas (DIM_FAKULTAS)

Tabel 3.14 Dimensi Fakultas

Atribut	Tipe Data	Panjang
ID_FAKULTAS	VARCHAR	9
NAMA	VARCHAR	100

Tabel 3.14 berisi detail atribut dari dimensi fakultas. Sesuai namanya dimensi fakultas berfungsi untuk menyimpan detail informasi mengenai fakultas. Sebagai kunci identitas terdapat atribut ID_FAKULTAS dengan tipe data VARCHAR.

11. Tabel Dimensi Periode Akademik (DIM_PERIODE_AKADEMIK)

Tabel 3.15 Dimensi Periode Akademik

Atribut	Tipe Data	Panjang
ID_PERIODE_AKADEMIK	INTEGER	11
TAHUN	INTEGER	11
TIPE	VARCHAR	10

Tabel 3.15 berisi detail atribut dari dimensi periode akademik. Sesuai namanya dimensi periode akademik berfungsi untuk menyimpan detail informasi mengenai periode akademik. Sebagai kunci identitas terdapat atribut ID_PERIODE_AKADEMIK dengan tipe data INTEGER.

12. Tabel Dimensi Angkatan (DIM_ANGKATAN)

Tabel 3.16 Dimensi Angkatan

Atribut	Tipe Data	Panjang
ID_ANGKATAN	INTEGER	11
DESKRIPSI	VARCHAR	20

Tabel 3.16 berisi detail atribut dari dimensi angkatan. Sesuai namanya dimensi angkatan berfungsi untuk menyimpan detail informasi mengenai angkatan. Sebagai kunci identitas terdapat atribut ID_ANGKATAN dengan tipe data INTEGER.

13. Tabel Dimensi Mata Kuliah (DIM_MATKUL)

Tabel 3.17 Dimensi Mata Kuliah

Atribut	Tipe Data	Panjang
ID_MK	INTEGER	11
K_MK	VARCHAR	10
NAMA	VARCHAR	100
SKS	INTEGER	11

Tabel 3.17 berisi detail atribut dari dimensi mata kuliah. Sesuai namanya dimensi mata kuliah berfungsi untuk menyimpan detail informasi mengenai mata kuliah. Sebagai kunci identitas terdapat atribut ID_MK dengan tipe data INTEGER.

3.5.7 Menentukan Durasi Basis Data

Diasumsikan pembangunan *data mart* untuk Universitas Brawijaya Bagian Pendidikan memiliki durasi basis data 5 tahun. Dimulai dari tahun 2010 sampai tahun 2015. Data yang digunakan merupakan data *dummy* untuk keperluan

pengujian yang di-*generate* berdasarkan data statistik dari data pangkalan data pendidikan tinggi.

3.5.8 Mengidentifikasi Dimensi yang Memiliki Perubahan Lamban

Dengan asumsi tidak diperlukan adanya pelacakan perubahan data pada tabel dimensi maka dapat dikatakan semua dimensi dikategorikan *slow changing dimension* tipe 1 (SCD1). Hal ini berarti jika ada perubahan data maka data lama yang ada di tabel dimensi akan langsung di timpa (*overwrite*).

3.5.9 Menentukan Prioritas dan Mode Akses *Query*

Menentukan prioritas dan mode akses *query* maksudnya adalah menentukan bagaimana data yang ada di *data warehouse* atau *data mart* diakses dan digunakan oleh pengguna maupun sistem yang lain. Pada penelitian ini penulis menfokuskan akses data dengan menggunakan *materialized query table* (MQT) untuk meningkatkan performa *data mart*. Adapun detail bagaimana nantinya MQT akan digunakan akan dibahas pada sub bab selanjutnya yaitu "4.6 Perancangan *Materialized Query Tabel* (MQT)".

3.6 Perancangan Proses ETL

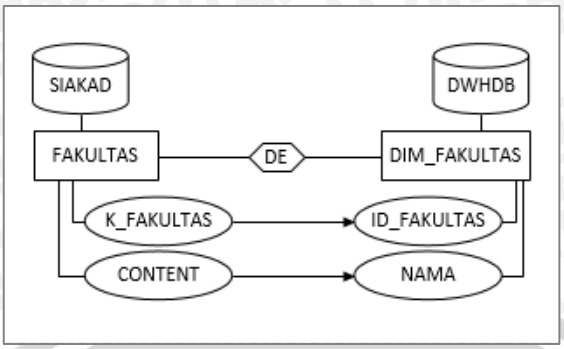
Setelah model multidimensional *data mart* selesai dirancang maka pada tahap ini akan dilakukan perancangan bagaimana proses perpindahan data dari sistem OLTP ke *data mart* atau yang sering disebut proses ETL. Secara mendasar proses ETL terdiri dari 3 tahap yaitu:

1. Sistem akan memilih dan mengekstraksi data dari basis data OLTP yang dalam kasus ini adalah basis data SIAKAD Akademik Universitas Brawijaya.
2. Data basil ekstraksi akan ditampung sementara di komponen *data staging* untuk dilakukan transformasi atau penyesuaian data dengan format yang dibutuhkan *data mart*.
3. Setelah format data telah sesuai maka data siap disimpan pada *data mart*.

Untuk mempermudah proses perancangan proses ETL seperti yang dimaksud diatas, penulis akan membuat diagram pemetaan dari sumber data, *data staging* dan penyimpanan data.

3.6.1 Dimensi Fakultas

Sumber data dimensi fakultas berasal dari tabel FAKULTAS di basis data SIAKAD.

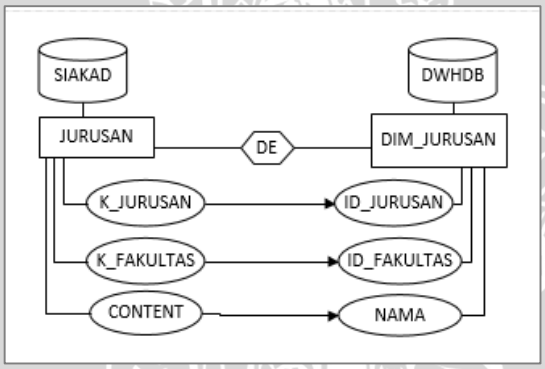


Gambar 3.6 Pemetaan Data Dimensi Fakultas

Seperti yang ditunjukkan Gambar 3.6 untuk melakukan proses ETL data dapat di petakan secara langsung dari tabel FAKULTAS ke tabel DIM_FAKULTAS. Hal ini dikarenakan kedua tabel memiliki atribut dan format yang sama.

3.6.2 Dimensi Jurusan

Sumber data dimensi fakultas berasal dari tabel JURUSAN di basis data SIAKAD.

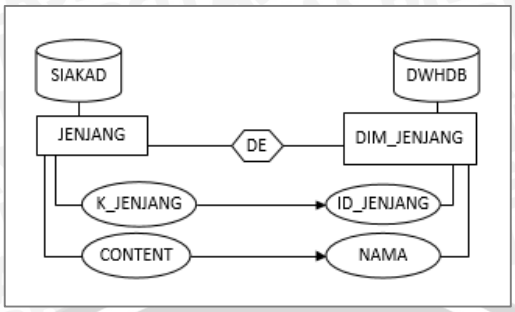


Gambar 3.7 Pemetaan Data Dimensi Jurusan

Seperti yang ditunjukkan Gambar 3.7 untuk melakukan proses ETL data dapat di petakan secara langsung dari tabel JURUSAN ke tabel DIM_JURUSAN. Hal ini dikarenakan kedua tabel memiliki atribut dan format yang sama.

3.6.3 Dimensi Jenjang

Sumber data dimensi fakultas berasal dari tabel JENJANG di basis data SIAKAD.

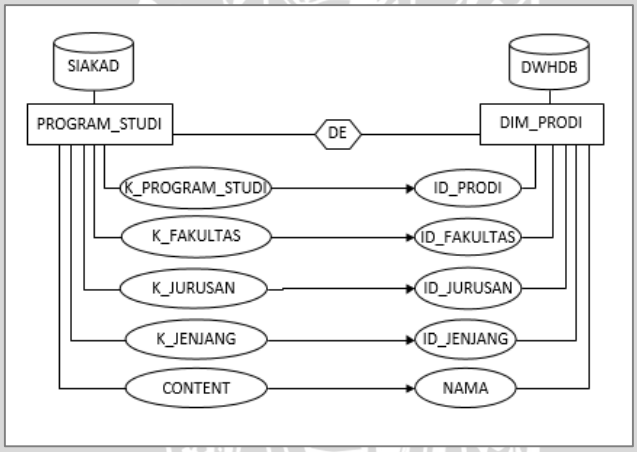


Gambar 3.8 Pemetaan Data Dimensi Jenjang

Gambar 3.8 menunjukkan proses ETL yang digunakan dimensi fakultas yang menggunakan sumber data dari basis data SIAKAD tabel JENJANG. Karena kolom dan format sudah sesuai maka tidak ada transformasi data yang dilakukan sehingga hanya dilakukan pemetaan kolom.

3.6.4 Dimensi Program Studi

Sumber data dimensi fakultas berasal dari tabel PROGRAM_STUDI di basis data SIAKAD.

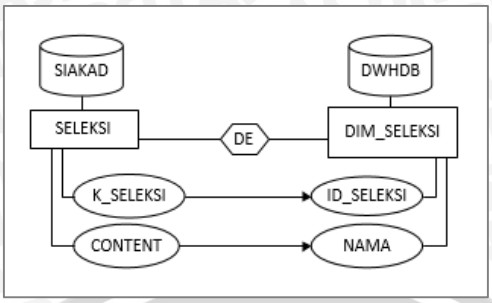


Gambar 3.9 Pemetaan Data Dimensi Program Studi

Gambar 3.9 menunjukkan proses ETL dari sumber data tabel PROGRAM_ STUDI di basis data SIAKAD menuju dimensi program studi. Karena kedua tabel memiliki kolom dan format yang sesuai maka pemetaan data dapat dilakukan secara langsung.

3.6.5 Dimensi Seleksi

Sumber data dimensi fakultas berasal dari tabel SELEKSI di basis data SIAKAD.

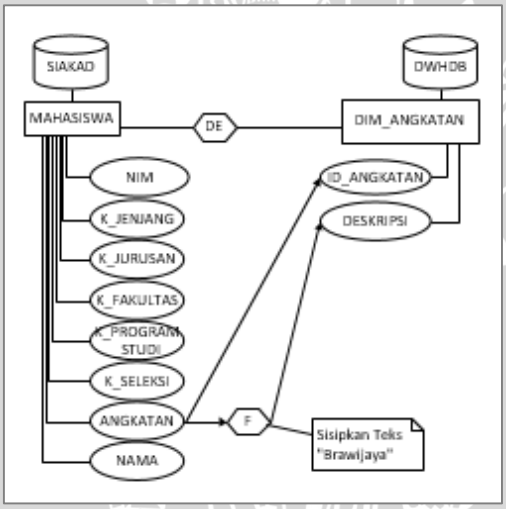


Gambar 3.10 Pemetaan Data Dimensi Seleksi

Gambar 3.10, data pada dimensi seleksi berasal dari tabel seleksi di basis data SIAKAD. Dikarenakan format tabel seleksi dan dimensi seleksi sudah sesuai maka proses transformasi data tidak diperlukan. Sehingga setelah data diekstraksi dari tabel seleksi dapat langsung disimpan di dimensi fakultas.

3.6.6 Dimensi Angkatan

Sumber data dimensi fakultas berasal dari tabel ANGKATAN di basis data SIAKAD.



Gambar 3.11 Pemetaan Data Dimensi Angkatan

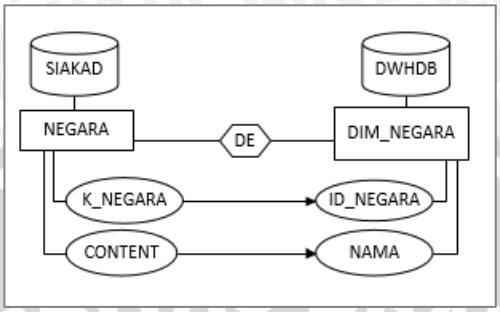
Gambar 3.11, data pada dimensi angkatan berasal dari tabel mahasiswa di basis data SIAKAD. Dikarenakan format tabel mahasiswa dan dimensi angkatan masih belum sesuai maka diperlukan proses transformasi untuk menyesuaikannya. Adapun proses transformasi yang dilakukan adalah:

1. Data dari tabel mahasiswa dipilih 2 kolom yaitu kolom angkatan dan kolom deskripsi. Kolom deskripsi adalah kolom yang dibuat untuk memenuhi kebutuhan dimensi angkatan dengan cara menggabungkan kata ‘Brawijaya’ dengan kolom angkatan.
2. Setelah data didapatkan dilakukanlah proses *distinct* untuk menghilangkan data-data kembar.

3. Setelah itu data siap untuk disimpan di dimensi angkatan.

3.6.7 Dimensi Negara

Sumber data dimensi fakultas berasal dari tabel NEGARA di basis data SIAKAD.

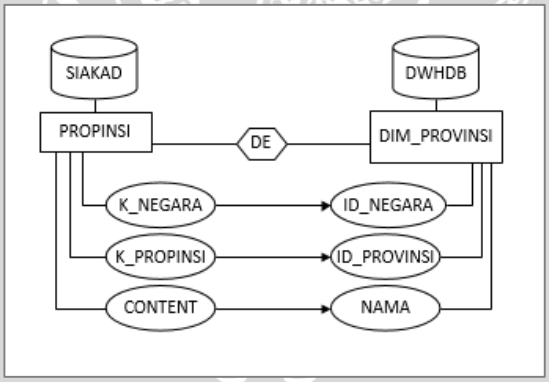


Gambar 3.12 Pemetaan Data Dimensi Negara

Seperti yang terlihat pada Gambar 3.12, data pada dimensi negara berasal dari tabel negara di basis data SIAKAD. Dikarenakan format tabel negara dan dimensi negara sudah sesuai maka proses transformasi data tidak diperlukan. Sehingga setelah data diekstraksi dari tabel negara dapat langsung disimpan di dimensi negara.

3.6.8 Dimensi Propinsi

Sumber data dimensi fakultas berasal dari tabel PROPINSI di basis data SIAKAD.

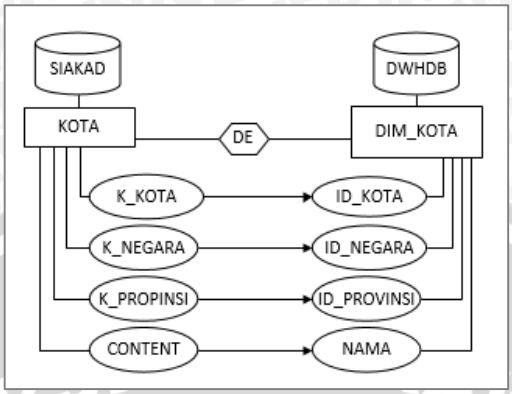


Gambar 3.13 Pemetaan Data Dimensi Propinsi

Seperti yang terlihat pada Gambar 3.13, data pada dimensi propinsi berasal dari tabel jenjang di basis data SIAKAD. Dikarenakan format tabel propinsi dan dimensi propinsi sudah sesuai maka proses transformasi data tidak diperlukan. Sehingga setelah data diekstraksi dari tabel propinsi dapat langsung disimpan di dimensi propinsi. Yang perlu diperhatikan adalah karena dimensi propinsi bergantung pada dimensi negara maka sebelum melakukan proses ETL pada dimensi propinsi diharuskan melakukan proses ETL pada dimensi negara terlebih dahulu untuk menjaga integritas data.

3.6.9 Dimensi Kota

Sumber data dimensi fakultas berasal dari tabel KOTA di basis data SIAKAD.

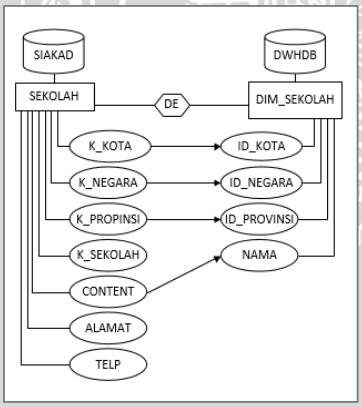


Gambar 3.14 Pemetaan Data Dimensi Kota

Seperti yang terlihat pada Gambar 3.14, data pada dimensi kota berasal dari tabel kota di basis data SIAKAD. Dikarenakan format tabel kota dan dimensi kota sudah sesuai maka proses transformasi data tidak diperlukan. Namun yang perlu diperhatikan adalah karena dimensi kota tergantung pada dimensi negara dan propinsi maka sebelum melakukan proses ETL untuk dimensi kota maka diharuskan melakukan proses ETL di dua dimensi tersebut terlebih dahulu untuk menjaga integritas data.

3.6.10 Dimensi Asal Sekolah

Sumber data dimensi fakultas berasal dari tabel SEKOLAH di basis data SIAKAD.

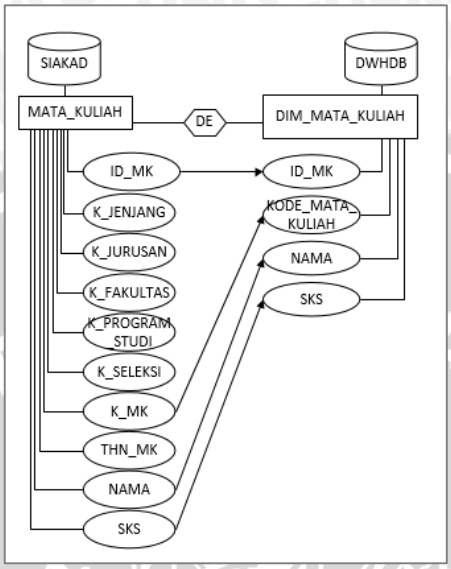


Gambar 3.15 Pemetaan Data Dimensi Sekolah Asal

Seperti yang terlihat pada Gambar 3.15, data pada dimensi asal sekolah berasal dari tabel sekolah di basis data SIAKAD. Dikarenakan ada perbedaan antara kebutuhan dimensi asal sekolah dan kondisi tabel sekolah, maka diperlukan proses seleksi kolom terlebih dahulu pada tabel sekolah. Adapun kolom yang diseleksi adalah kolom K_SEKOLAH dan NAMA. Setelah itu data dapat disimpan pada dimensi sekolah asal.

3.6.11 Dimensi Mata Kuliah

Sumber data dimensi fakultas berasal dari tabel MATA KULIAH di basis data SIAKAD.

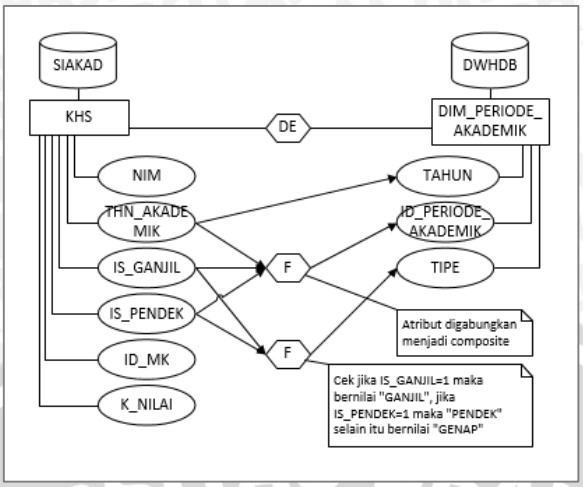


Gambar 3.16 Pemetaan Data Dimensi Mata Kuliah

Seperti yang terlihat pada Gambar 3.16, data pada dimensi mata kuliah berasal dari tabel mata kuliah di basis data SIAKAD. Dikarenakan ada perbedaan antara kebutuhan dimensi mata kuliah dan kondisi tabel mata kuliah, maka diperlukan proses seleksi kolom terlebih dahulu pada tabel mata kuliah. Adapun kolom yang diseleksi adalah kolom ID_MK, K_MK, SKS dan NAMA. Setelah itu data dapat disimpan pada dimensi mata kuliah.

3.6.12 Dimensi Periode Akademik

Sumber data dimensi fakultas berasal dari tabel KHS di basis data SIAKAD.



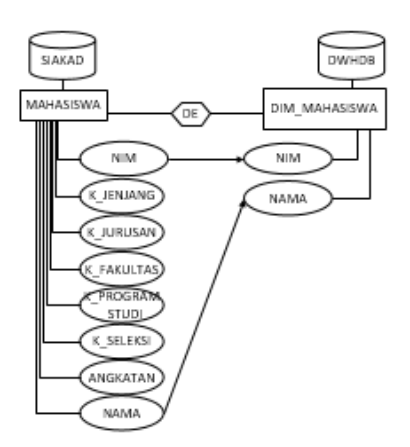
Gambar 3.17 Pemetaan Data Dimensi Periode Akademik

Seperti yang terlihat pada Gambar 3.17, data pada dimensi periode akademik berasal dari tabel KHS di basis data SIAKAD. Dikarenakan format tabel KHS dan dimensi periode akademik masih belum sesuai maka diperlukan proses transformasi untuk menyesuaikannya. Adapun proses transformasi yang dilakukan adalah:

1. Data dari tabel KHS dipilih 3 kolom yaitu kolom ID, THN_AKADEMIK dan TIPE. Kolom ID adalah kolom yang didapat dengan menggabungkan kolom THN_AKADEMIK, IS_GANJIL dan IS_PENDEK. Sementara kolom TIPE berasal dari pengecekan jika KOLOM IS_PENDEK bernilai '1' maka berisi 'PENDEK' jika KOLOM IS_GANJIL benilai '1' maka akan berisi 'GANJIL' selain itu Kolom tipe akan berisi 'GENAP'
2. Setelah data didapatkan dilakukanlah proses *distinct* untuk menghilangkan data-data kembar.
3. Setelah itu data siap untuk disimpan di dimensi periode akademik.

3.6.13 Dimensi Mahasiswa

Sumber data dimensi fakultas berasal dari tabel MAHASISWA di basis data SIAKAD.

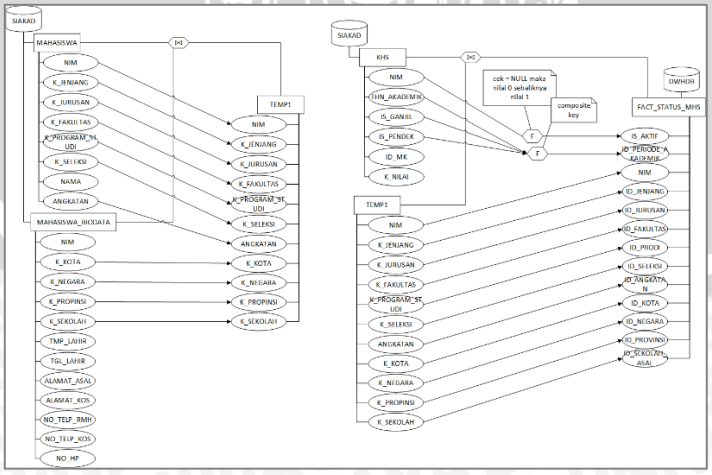


Gambar 3.18 Pemetaan Data Dimensi Mahasiswa

Seperti yang terlihat pada Gambar 3.18, data pada dimensi mahasiswa berasal dari tabel mahasiswa di basis data SIAKAD. Dikarenakan ada perbedaan antara kebutuhan dimensi mahasiswa dan kondisi tabel mahasiswa, maka diperlukan proses seleksi kolom terlebih dahulu pada tabel mahasiswa. Adapun kolom yang diseleksi adalah kolom NIM dan NAMA. Setelah itu data dapat disimpan pada dimensi mata kuliah.

3.6.14 Fakta Status Mahasiswa

Sesuai pembahasan bab 3.5.4 mengenai pemilihan fakta. Fakta status mahasiswa akan merekam persebaran mahasiswa berdasarkan statusnya di Universitas Brawijaya. Tabel fakta ini membutuhkan beberapa dimensi untuk mendukung fungsinya. Dimensi-dimensi tersebut antara lain dimensi fakultas, dimensi jurusan, dimensi jenjang, dimensi program studi, dimensi seleksi, dimensi negara, dimensi propinsi, dimensi kota, dimensi asal sekolah, dimensi angkatan, dimensi periode akademik dan dimensi mahasiswa. Adapun pemetaan datanya adalah sebagai berikut:



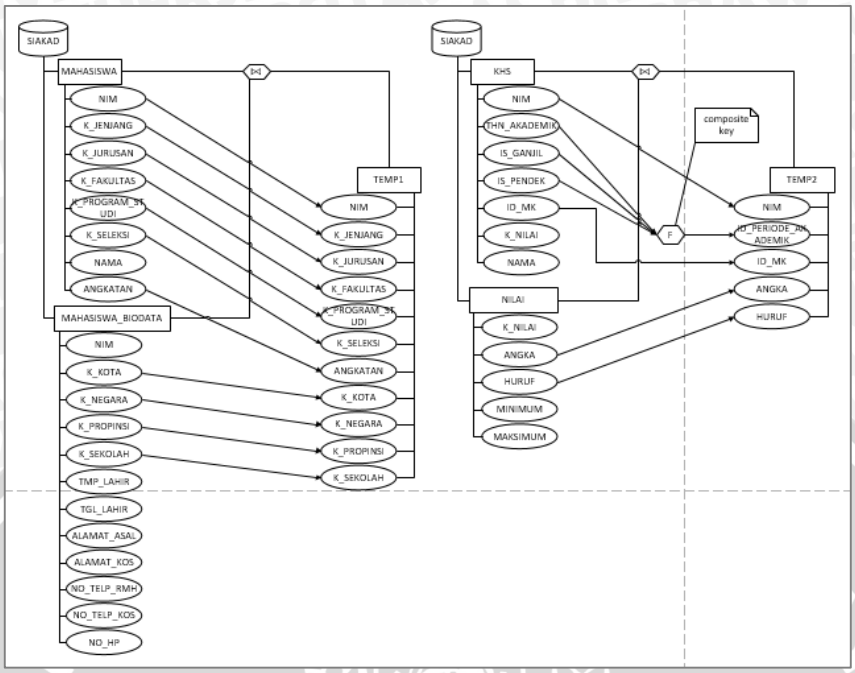
Gambar 3.19 Pemetaan Data Fakta Status Mahasiswa

Seperti yang terlihat pada Gambar 3.19 bahwa sumber data fakta status mahasiswa adalah tabel mahasiswa, tabel mahasiswa biodata dan tabel KHS di basis data SIAKAD. Untuk dapat disimpan di fakta status mahasiswa maka data-data tersebut harus melalui serangkaian proses transformasi terlebih dahulu adapun proses transformasinya adalah:

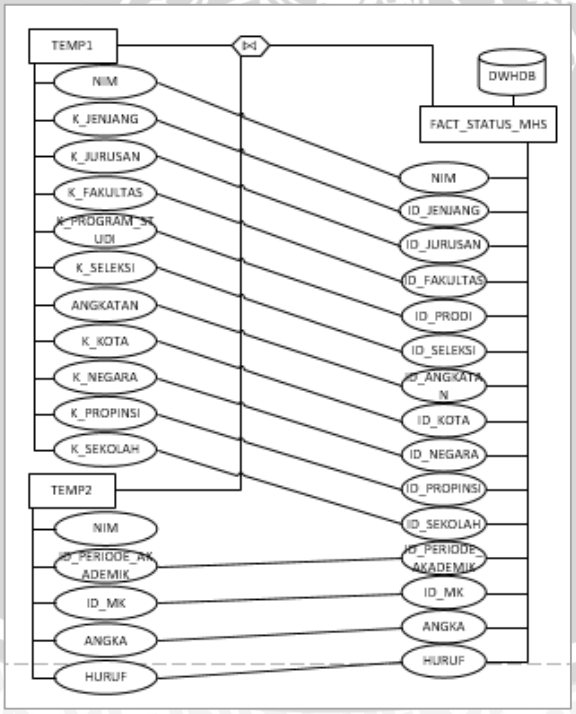
1. Dilakukannya proses *inner join* pada tabel MAHASISWA dan tabel MAHASISWA BIODATA kemudian dilakukan seleksi kolom yaitu kolom NIM, K_JENJANG, K_JURUSAN, K_FAKUTAS, K_PROGRAM_ STUDI, K_SELEKSI, ANGKATAN, K_KOTA, K_NEGARA, K_PROPINSI dan K_SEKOLAH. Setelah itu hasil *join* dan seleksi disimpan pada *result set 1*.
2. Kemudian *Result Set 1* di *left join*-kan dengan tabel KHS kemudian dilakukan proses seleksi kolom yaitu kolom NIM, K_JENJANG, K_JURUSAN, K_FAKUTAS, K_PROGRAM_STUDDI, K_SELEKSI, ANGKATAN, K_KOTA, K_NEGARA, K_PROPINSI, K_SEKOLAH, ID_PERIODE_AKADEMIK dan IS_AKTIF. Kolom ID_PERIODE_ AKADEMIK didapat dari penggabungan kolom THN_AKADEMIK, IS_PENDEK dan IS_GANJIL. Sementara kolom IS_AKTIF didapat dari pengecekan adakah *record* KHS mahasiswa yang bersangkutan pada periode akademik tersebut jika ada maka kolom IS_AKTIF akan bernilai 1 dan sebaliknya akan 0.
3. Setelah itu data siap untuk disimpan di fakta.

3.6.15 Fakta Entri Kartu Hasil Studi

Sesuai pembahasan bab 3.5.4 mengenai pemilihan fakta. Fakta entri kartu hasil studi akan merekam persebaran prestasi akademik mahasiswa di Universitas Brawijaya. Tabel fakta ini membutuhkan beberapa dimensi untuk mendukung fungsinya. Dimensi-dimensi tersebut antara lain dimensi fakultas, dimensi jurusan, dimensi jenjang, dimensi program studi, dimensi seleksi, dimensi negara, dimensi propinsi, dimensi kota, dimensi asal sekolah, dimensi angkatan, dimensi periode akademik, dimensi mata kuliah dan dimensi mahasiswa. Adapun pemetaan datanya adalah sebagai berikut:



Gambar 3.20 Pemetaan Data Fact Entri KHS 1



Gambar 3.21Pemetaan Data Fact Entri KHS 2

Seperti yang terlihat pada Gambar 3.20 dan Gambar 3.21 bahwa sumber data fakta entri KHS adalah tabel mahasiswa, tabel mahasiswa biodata, tabel nilai dan tabel KHS di basis data SIAKAD. Untuk dapat disimpan di fakta entri KHS maka

data-data tersebut harus melalui serangkaian proses transformasi terlebih dahulu adapun proses transformasinya adalah:

1. Dilakukannya proses *inner join* pada tabel MAHASISWA dan tabel MAHASISWA BIODATA kemudian dilakukan seleksi kolom yaitu kolom NIM, K_JENJANG, K_JURUSAN, K_FAKUTAS, K_PROGRAM_ STUDI, K_SELEKSI, ANGKATAN, K_KOTA, K_NEGARA, K_PROPINSI dan K_SEKOLAH. Setelah itu hasil *join* dan seleksi disimpan pada *result set 1*.
2. Dilakukannya proses *inner join* pada tabel KHS dan tabel NILAI kemudian dilakukan seleksi kolom yaitu kolom NIM, ID_MK, ANGKA, HURUF dan ID_PERIODE AKADEMIK. Kolom ID_PERIODE_ AKADEMIK didapat dari penggabungan kolom THN_AKADEMIK, IS_PENDEK dan IS_GANJIL. Setelah itu hasil *join* dan seleksi disimpan pada *result set 2*.
3. Kemudian dilakukan proses join antara *result set 1* dan *result set 2*.
4. Setelah itu data siap untuk disimpan di fakta.

3.7 Perancangan Materialized Query Table

Perancangan *materialized query table* pada dasarnya adalah menentukan dan merancang *query* apa saja yang akan digunakan. *Query* yang dibahas di sini adalah *reporting query* yang biasanya digunakan untuk pelaporan. Berdasarkan hasil wawancara di Universitas Brawijaya Bagian Akademik dan Konsultasi dengan dosen pembimbing yang ada di **Error! Reference source not found.** maka *query eporting* yang akan digunakan pada penelitian ini adalah:

1. Mendapatkan jumlah mahasiswa berstatus aktif di masing-masing fakultas, jenjang, jurusan, program studi, seleksi per periode akademik.
2. Mendapatkan rata-rata IPK mahasiswa setiap periode akademik di masing-masing fakultas, jenjang, program studi dan seleksi.
3. Mendapatkan jumlah mahasiswa berdasar asal sekolah di masing-masing fakultas, jenjang, jurusan, program studi, seleksi dan periode akademik.
4. Mendapatkan jumlah mahasiswa berdasar asal daerah di masing-masing fakultas, jenjang, jurusan, program studi, seleksi dan periode akademik.

Setelah *query reporting* ditentukan maka selanjutnya adalah merancang algoritma bagaimana *query* tersebut nantinya akan dijalankan. Perancangan *query* akan dilakukan dengan menggunakan bantuan aljabar relasional.

3.7.1 *Query* Jumlah Mahasiswa Aktif

Query ini digunakan untuk menampilkan hasil berupa sebaran jumlah mahasiswa aktif per fakultas, jurusan, jenjang, program studi, seleksi, angkatan dan periode akademik. Sehingga untuk melakukannya diperlukan tabel DIM_FAKULTAS, DIM_JURUSAN, DIM_JENJANG, DIM_PROGRAM_ STUDI, DIM_SELEKSI, DIM_ANGKATAN, DIM_PERIODE_AKADEMIK dan

FACT_STATUS_MHS. Ke-8 tabel tersebut akan mengalami proses *join* kemudian dihitung mana mahasiswa yang memiliki status aktif. Sehingga dalam aljabar relasional bentuknya adalah sebagai berikut:

```
1.  RESULTSET1←( ID_FAKULTAS, ID_JURUSAN, ID_JENJANG, ID_PRODI, ID_SELEKSI,
2.  ID_ANGKATAN, ID_PERIODE_AKADEMIK 𝒢 COUNT(IS_AKTIF) →JUMLAH
3.    (π ID_FAKULTAS, ID_JURUSAN, ID_JENJANG, ID_PRODI, ID_SELEKSI,
4.        ID_ANGKATAN, ID_PERIODE_AKADEMIK
5.         (σ F.IS_AKTIF = 1 (FACT_STATUS_MHS))
6.    )
7.  )
8.
9.  ρ PROGRAM_STUDI←DP.NAMA, JURUSAN←DJ.NAMA, FAKULTAS←DF.NAMA,
10.   JENJANG←DJE.NAMA, SELEKSI←DS.NAMA, ANGKATAN←F.ID_ANGKATAN,
11.   TAHUN_AKADEMIK←DPA.TAHUN, TIPE_AKADEMIK←DPA.TIPE
12. ( π DP.NAMA, DJ.NAMA, DF.NAMA, DJE.NAMA, DS.NAMA, F.ID_ANGKATAN, DPA.TAHUN, DPA.TIPE
13.   (ρ F (RESULTSET1) ⋈ (ρ DP (DIM_PRODI))
14.     ⋈ F.ID_JURUSAN = DJ.ID_JURUSAN AND F.ID_FAKULTAS = DJ.ID_FAKULTAS (ρ DJ (DIM_JURUSAN))
15.       ⋈ F.ID_FAKULTAS = DF.ID_FAKULTAS (ρ DF (DIM_FAKULTAS))
16.       ⋈ F.ID_JENJANG = DJE.ID_JENJANG (ρ DJE (DIM_JENJANG))
17.       ⋈ F.ID_SELEKSI = DS.ID_SELEKSI  (ρ DS (DIM_SELEKSI))
18.       ⋈ F.ID_ANGKATAN = DA.ID_ANGKATAN (ρ DA (DIM_ANGKATAN))
19.       ⋈ F.ID_PERIODE_AKADEMIK = DPA.ID_PERIODE_AKADEMIK
20.          (ρ DPA (DIM_PERIODE_AKADEMIK))
21.   )
22. )
```

Adapun penjelasan aljabar di atas adalah sebagai berikut:

1. Baris 1-2 : adalah proses *grouping* kolom-kolom yang mendukung fungsi agregasi *count* untuk menghitung jumlah mahasiswa. Untuk memudahkan pembacaan maka hasil join akan disimpan pada *RESULTSET1* untuk dipakai di baris-baris berikutnya.
2. Baris 3-4 : adalah proses *projection* kolom-kolom yang akan digunakan pada dari tabel FACT_STATUS_MHS.
3. Baris 5 : merupakan proses *selection* untuk memilih baris yang kolom IS_AKTF-nya bernilai 1 saja yang akan ditampilkan.
4. Baris 9-11 : adalah proses *renaming* dari kolom-kolom yang telah diseleksi untuk memudahkan penggunaan kolom-kolom tersebut.

5. Baris 12 : adalah proses *projection* kolom-kolom yang digunakan dalam *query reporting* ini.
6. Baris 13-20 : merupakan proses *join* dari 7 tabel dimensi dan tabel fakta status mahasiswa.

3.7.2 *Query* Rata-rata IPK Mahasiswa

Query ini digunakan untuk menampilkan hasil berupa sebaran rata-rata IPK mahasiswa per fakultas, jurusan, jenjang, program studi, seleksi, angkatan dan periode akademik. Sehingga untuk mendapatknya memerlukan tabel DIM_FAKULTAS, DIM_JURUSAN, DIM_JENJANG, DIM_PROGRAM_ STUDI, DIM_SELEKSI, DIM_ANGKATAN, DIM_PERIODE_AKADEMIK, DIM_MATA_KULIAH dan FACT_PERKEMBANGAN_AKADEMIK. Ke-9 tabel tersebut akan mengalami proses *join* kemudian dihitung rata-rata IPK-nya. Maka dalam aljabar relasional bentuknya adalah sebagai berikut:

1. **RESULTSET1**←(F.ID_FAKULTAS, F.ID_JURUSAN, F.ID_JENJANG, F.ID_PRODI, F.ID_SELEKSI,
2. F.ID_ANGKATAN, F.ID_PERIODE_AKADEMIK $\mathcal{G}$ (SUM(F.NILAI_ANGKA x DM.SKS)/ SUM(DMK.SKS))→RATA_IPK
3. (π F.ID_FAKULTAS, F.ID_JURUSAN, F.ID_JENJANG, F.ID_PRODI,
4. F.ID_SELEKSI, F.ID_ANGKATAN, F.ID_PERIODE_AKADEMIK
5. (ρ F (FACT_ENTRI_KHS) ⋈ F.ID_MK = DMA.ID_MK (ρ DMK (DIM_MATA_KULIAH)))
6.)
7.)
8.
9. ρ PROGRAM_STUDI←DP.NAMA, JURUSAN←DJ.NAMA,
10. FAKULTAS←DF.NAMA, JENJANG←DJE.NAMA, SELEKSI←DS.NAMA,
11. ANGKATAN←F.ID_ANGKATAN, TAHUN_AKADEMIK←DPA.TAHUN, TIPE_AKADEMIK←DPA.TIPE
12. (π DP.NAMA, DJ.NAMA, DF.NAMA, DJE.NAMA, DS.NAMA, F.ID_ANGKATAN, DPA.TAHUN, DPA.TIPE
13. (ρ F (**RESULTSET1**) ⋈ (ρ DP (DIM_PRODI))
14. ⋈ F.ID_JURUSAN = DJ.ID_JURUSAN AND F.ID_FAKULTAS = DJ.ID_FAKULTAS (ρ DJ (DIM_JURUSAN))
15. ⋈ F.ID_FAKULTAS = DF.ID_FAKULTAS (ρ DF (DIM_FAKULTAS))
16. ⋈ F.ID_JENJANG = DJE.ID_JENJANG (ρ DJE (DIM_JENJANG))
17. ⋈ F.ID_ANGKATAN = DA.ID_ANGKATAN (ρ DA (DIM_ANGKATAN))
18. ⋈ F.ID_PERIODE_AKADEMIK = DPA.ID_PERIODE_AKADEMIK
19. (ρ DPA (DIM_PERIODE_AKADEMIK))
20. ⋈ F.ID_MK = DMA.ID_MK (ρ DMK (DIM_MATA_KULIAH))
21.)
22.)

Adapun penjelasan aljabar di atas adalah sebagai berikut:

1. Baris 1-2 : adalah proses *grouping* kolom-kolom yang mendukung fungsi agregasi *sum* untuk menghitung IPK mahasiswa. Untuk memudahkan pembacaan maka hasil join akan disimpan pada *RESULTSET1* untuk dipakai di baris-baris berikutnya.
2. Baris 3-4 : adalah proses *projection* kolom-kolom yang akan digunakan pada dari tabel FACT_ENTRI_KHS.
3. Baris 5 : merupakan proses *join* dari tabel FACT_ENTRI_KHS dan DIM_MATA KULIAH.
4. Baris 9-11 : adalah proses *renaming* dari kolom-kolom yang telah diseleksi untuk memudahkan penggunaan kolom-kolom tersebut.
5. Baris 12 : adalah proses *projection* kolom-kolom yang digunakan dalam *query reporting* ini.
6. Baris 13-20 : merupakan proses *join* dari 7 tabel dimensi dan tabel fakta status mahasiswa.

3.7.3 *Query* Jumlah Mahasiswa Berdasar Asal Sekolah

Query ini digunakan untuk menampilkan hasil berupa sebaran jumlah mahasiswa berdasarkan daerah asal fakultas, jurusan, jenjang, program studi, seleksi, angkatan, asal sekolah sekolah dan periode akademik. Sehingga untuk mendapatknya memerlukan tabel DIM_FAKULTAS, DIM_JURUSAN, DIM_JENJANG, DIM_PROGRAM_STUDI, DIM_SELEKSI, DIM_ ANGKATAN, DIM_PERIODE_AKADEMIK, DIM_SEKOLAH_ASAL dan FACT_STATUS_MHS. Ke-9 tabel tersebut akan mengalami proses *join* kemudian dihitung jumlah mahasiswanya. Maka dalam aljabar relasional bentuknya adalah sebagai berikut:

```
1.  RESULTSET1←( ID_FAKULTAS, ID_JURUSAN, ID_JENJANG, ID_PRODI, ID_SELEKSI,
2.  ID_ANGKATAN, ID_PERIODE_AKADEMIK, ID_SEKOLAH_ASAL 𝒢 COUNT(NIM) →JUMLAH
3.    (π ID_FAKULTAS, ID_JURUSAN, ID_JENJANG, ID_PRODI, ID_SELEKSI,
4.        ID_ANGKATAN, ID_PERIODE_AKADEMIK, ID_SEKOLAH_ASAL
5.         (FACT_STATUS_MHS)
6.    )
7.  )
8.
9.  ρ PROGRAM_STUDI←DP.NAMA, JURUSAN←DJ.NAMA, FAKULTAS←DF.NAMA, JENJANG←DJE.NAMA,
10.   SELEKSI←DS.NAMA, ANGKATAN←F.ID_ANGKATAN, TAHUN_AKADEMIK←DPA.TAHUN,
11.   TIPE_AKADEMIK←DPA.TIPE, ASAL_SEKOLAH←DSA.NAMA,
12. ( π DP.NAMA, DJ.NAMA, DF.NAMA, DJE.NAMA, DS.NAMA,
13.       F.ID_ANGKATAN,  DPA.TAHUN, DPA.TIPE, DSA.NAMA
14.   (ρ F (RESULTSET1) ⋈ (ρ DP (DIM_PRODI))
15.     ⋈ F.ID_JURUSAN = DJ.ID_JURUSAN AND F.ID_FAKULTAS = DJ.ID_FAKULTAS (ρ DJ (DIM_JURUSAN))
16.       ⋈ F.ID_FAKULTAS = DF.ID_FAKULTAS (ρ DF (DIM_FAKULTAS))
```

17. ⋈ F.ID_JENJANG = DJE.ID_JENJANG (ρ DJE (DIM_JENJANG))
18. ⋈ F.ID_SELEKSI = DS.ID_SELEKSI (ρ DS (DIM_SELEKSI))
19. ⋈ F.ID_ANGKATAN = DA.ID_ANGKATAN (ρ DA (DIM_ANGKATAN))
20. ⋈ F.ID_PERIODE_AKADEMIK = DPA.ID_PERIODE_AKADEMIK
21. (ρ DPA (DIM_PERIODE_AKADEMIK))
22. ⋈ F.ID_SEKOLAH_ASAL = DSA.ID_SEKOLAH_ASAL (ρ DSA (DIM_SEKOLAH_ASAL))
23.)
24.)

Adapun penjelasan aljabar di atas adalah sebagai berikut:

1. Baris 1-2 : adalah proses *grouping* kolom-kolom yang mendukung fungsi agregasi *count* untuk menghitung jumlah mahasiswa. Untuk memudahkan pembacaan maka hasil join akan disimpan pada *RESULTSET1* untuk dipakai di baris-baris berikutnya.
2. Baris 3-4 : adalah proses *projection* kolom-kolom yang akan digunakan pada dari tabel FACT_STATUS_MHS.
3. Baris 5 : merupakan proses pemilihan tabel FACT_STATUS_MHS.
4. Baris 9-11 : adalah proses *renaming* dari kolom-kolom yang telah diseleksi untuk memudahkan penggunaan kolom-kolom tersebut.
5. Baris 12-13 : adalah proses *projection* kolom-kolom yang digunakan dalam *query reporting* ini.
6. Baris 14-22 : merupakan proses *join* dari 8 tabel dimensi dan tabel fakta status mahasiswa.

3.7.4 *Query* Jumlah Mahasiswa Berdasar Asal Daerah

Query ini digunakan untuk menampilkan hasil berupah sebaran jumlah mahasiswa berdasarkan daerah asal per fakultas, jurusan, jenjang, program studi, seleksi, angkatan dan periode akademik. Sehingga untuk mendapatknya memerlukan tabel DIM_FAKULTAS, DIM_JURUSAN, DIM_JENJANG, DIM_PROGRAM_STUDI, DIM_SELEKSI, DIM_ANGKATAN, DIM_ PERIODE_AKADEMIK, DIM_NEGARA, DIM_PROVINSI, DIM_KOTA dan FACT_STATUS_MHS. Ke-9 tabel tersebut akan mengalami proses *join* kemudian dihitung jumlah mahasiswanya. Maka dalam aljabar relasional bentuknya adalah sebagai berikut:

```
1.  RESULTSET1←( ID_FAKULTAS, ID_JURUSAN, ID_JENJANG, ID_PRODI, ID_SELEKSI,
2.  ID_ANGKATAN, ID_PERIODE_AKADEMIK, ID_KOTA, ID_PROVINSI, ID_NEGARA 𝒢 COUNT(NIM) →JUMLAH
3.    (π ID_FAKULTAS, ID_JURUSAN, ID_JENJANG, ID_PRODI, ID_SELEKSI,
4.        ID_ANGKATAN, ID_PERIODE_AKADEMIK, ID_KOTA, ID_PROVINSI, ID_NEGARA
5.         (FACT_STATUS_MHS)
6.    )
7.  )
8.
9.  ρ PROGRAM_STUDI←DP.NAMA, JURUSAN←DJ.NAMA, FAKULTAS←DF.NAMA, JENJANG←DJE.NAMA,
10.   SELEKSI←DS.NAMA, ANGKATAN←F.ID_ANGKATAN, TAHUN_AKADEMIK←DPA.TAHUN,
11.   TIPE_AKADEMIK←DPA.TIPE, NEGARA←DN.NAMA, PROPINSI←DPR.NAMA, KOTA←DK.NAMA
12. ( π DP.NAMA, DJ.NAMA, DF.NAMA, DJE.NAMA, DS.NAMA, DN.NAMA, DPR.NAMA, DK.NAMA,
13.       F.ID_ANGKATAN,  DPA.TAHUN, DPA.TIPE, DSA.NAMA
14.    (ρ F (RESULTSET1) ⋈ (ρ DP (DIM_PRODI))
15.      ⋈ F.ID_JURUSAN = DJ.ID_JURUSAN AND F.ID_FAKULTAS = DJ.ID_FAKULTAS (ρ DJ (DIM_JURUSAN))
16.        ⋈ F.ID_FAKULTAS = DF.ID_FAKULTAS (ρ DF (DIM_FAKULTAS))
17.        ⋈ F.ID_JENJANG = DJE.ID_JENJANG (ρ DJE (DIM_JENJANG))
18.        ⋈ F.ID_SELEKSI = DS.ID_SELEKSI  (ρ DS (DIM_SELEKSI))
19.        ⋈ F.ID_ANGKATAN = DA.ID_ANGKATAN (ρ DA (DIM_ANGKATAN))
20.        ⋈ F.ID_PERIODE_AKADEMIK = DPA.ID_PERIODE_AKADEMIK
21.       (ρ DPA (DIM_PERIODE_AKADEMIK))
22.        ⋈ F.ID_NEGARA = DN.ID_NEGARA (ρ DN (DIM_NEGARA))
23.        ⋈ F.ID_ID_PROPINSI = DPR.ID_PROPINSI (ρ DN (DIM_PROPINSI))
24.        ⋈ F.ID_KOTA = DK.ID_KOTA (ρ DN (DIM_KOTA))
25.    )
26. )
```

Adapun penjelasan aljabar di atas adalah sebagai berikut:

1. Baris 1-2 : adalah proses *grouping* kolom-kolom yang mendukung fungsi agregasi *count* untuk menghitung jumlah mahasiswa. Untuk memudahkan pembacaan maka hasil join akan disimpan pada *RESULTSET1* untuk dipakai di baris-baris berikutnya.
2. Baris 3-4 : adalah proses *projection* kolom-kolom yang akan digunakan pada dari tabel FACT_STATUS_MHS.
3. Baris 5 : merupakan proses pemilihan tabel FACT_STATUS_MHS.
4. Baris 9-11 : adalah proses *renaming* dari kolom-kolom yang telah diseleksi untuk memudahkan penggunaan kolom-kolom tersebut.
5. Baris 12-13 : adalah proses *projection* kolom-kolom yang digunakan dalam *query reporting* ini.
6. Baris 14-24 : merupakan proses *join* dari 10 tabel dimensi dan tabel fakta status mahasiswa.

BAB 4 IMPLEMENTASI

4.1 Implementasi Arsitektur

Lingkungan implementasi yang akan dijelaskan pada bab ini adalah lingkungan implementasi perangkat keras dan lunak di sisi klien dan *server*.

4.1.1 Lingkungan *Server*

Pada penelitian ini server akan disimulasikan oleh virtual komputer dengan menggunakan perangkat lunak *VMWare Workstation 10.0* dengan spesifikasi visualisasi:

Perangkat	**Spesifikasi**
Sistem Operasi	CentOS 6.6 versi 64 bit
Prosessor	*Dual Core* (Dua Prosessor)
RAM	2048 MB
Hardisk	50 GB 100 GB 100 GB 200 GB
Konfigurasi Alamat IP	192.168.17.128
Database Management System	IBM DB2
Data Warehouse Server	*Web Sphere Application Server* *SQL Warehouse Administration* *Cubing Services Administration* *Intellegent Mining Administration*

4.1.2 Lingkungan *Client*

Untuk lingkungan klien perangkat lunak dan perangkat keras yang digunakan antara lain:

Perangkat	**Spesifikasi**
Prosessor	*Intel Core i5-4200M CPU @ 2.5 GHz 2.5 GHz*
RAM	4 GB
Hardisk Internal	1 TB
Klien Basis Data	*IBM Data & Design Studio 4.1.0.0 Client*

4.2 Basis Data OLTP

Dalam penelitian ini penulis mengimplementasikan basis data dengan menggunakan sistem manajemen basis data IBM DB2. Proses implementasi dilakukan dengan membuat perintah DDL (*Data Definition Language*) dari skema yang telah dirancang pada bab 3.1.3.2. Setelah itu perintah DDL yang dibuat di *deploy* ke *server* basis data SIAKAD.

Adapun secara rinci perintah DDL yang telah dibuat dapat dilihat pada LAMPIRAN B.

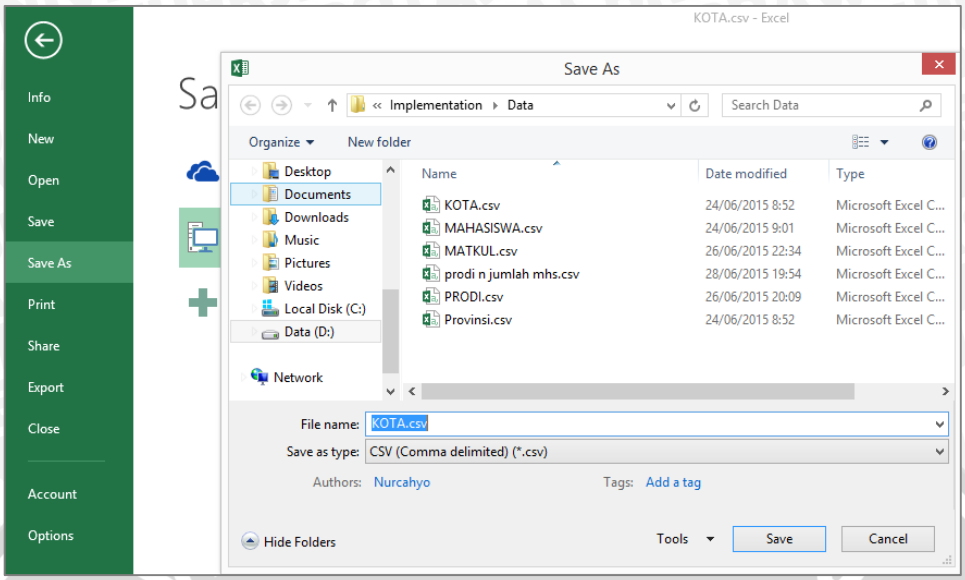
4.3 Pembuatan Data Penelitian

Sesuai dengan yang dibahas pada bab 4 mengenai perancangan data penelitian, telah disebutkan bahwa dalam penelitian ini akan menggunakan 2 macam data. Data yang pertama adalah data sebenarnya yang berasal Dirjen Pendidikan Tinggi dan Universitas Brawijaya. Sedangkan data ke dua adalah data *dummy* yang dibuat untuk memenuhi kebutuhan simulasi berjalannya basis data OLTP Siakad di kehidupan nyata. Untuk dapat berjalan di basis data OLTP Siakad tentu keduanya membutuhkan perlakuan yang berbeda. Berikut penjelasan mendetailnya.

4.3.1 Data Asli

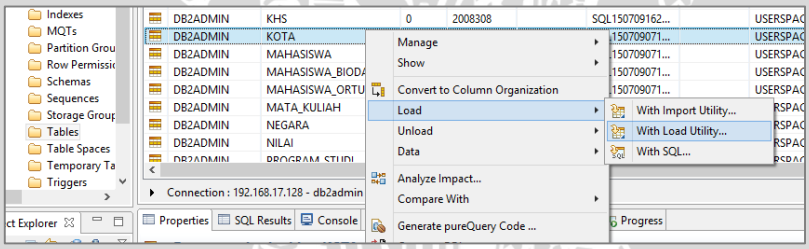
Data ini disebut data asli dikarenakan penulis memperoleh data asli dari sumber yang dapat dipertanggung jawabkan. Adapun data-data tersebut adalah: Data Fakultas, Data Jurusan, Data Jenjang, Data Program Studi, Data Seleksi, Data Mata Kuliah, Data Nilai, Data Negara, Data Propinsi, Data Kota, Data Sekolah dan Data Daftar Pekerjaan. Data-data tersebut didapatkan dalam format dokumen *Microsoft Office Excel*. Sehingga untuk digunakan dalam basis data DB2 penulis menggunakan utilitas *import data*. Adapun berikut adalah langkah-langkah *import data* yang penulis lakukan:

1. Langkah pertama adalah simpan ulang file *excel* ke dalam format CSV (*Comma Separated Value*) dengan Cara pada program *Microsoft Office Excel* klik *File>Save As* dan beri nama file baru setelah itu ubah *file type* menjadi ".csv". Lebih jelasnya dapat dilihat pada Gambar 4.1. Yang perlu diperhatikan adalah sebelum melakukan penyimpanan ulang sebaiknya dilakukan penyesuaian kolom yang ada pada file *excel* dengan yang ada di basis data baik dari segi jumlah kolom dan tipe datanya.



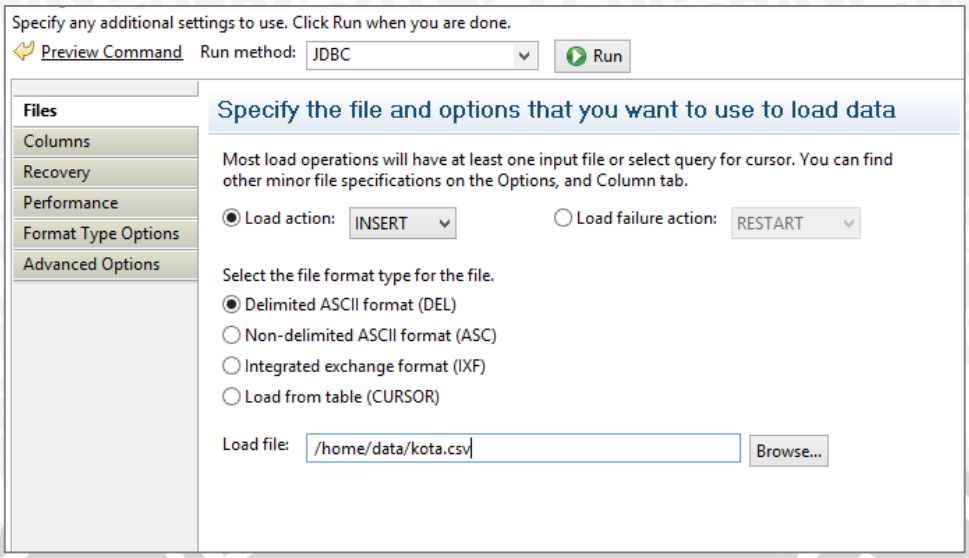
Gambar 4.1 Penyimpanan Kemabali dengan Format .CSV

2. Setelah dokumen tersimpan dengan format .csv maka selanjutnya adalah membuka program *Data Studio* dan melakukan koneksi dengan basis data SIAKAD. Setelah itu pilih *Tables* dan klik kanan pada tabel yang ingin melakukan impor data. Setelah itu pilih menu *Load>With Load Utility*. Lebih jelasnya dapat dilihat pada Gambar 4.2.



Gambar 4.2 Membuka Utilitas Impor Data

3. Setelah muncul *wizard* untuk melakukan impor data maka pada masukan *"Load File"* isikan alamat dokumen .csv yang ada di *server*. Setelah itu jalankan perintah impor dengan mengklik tombol *Run* di atas. Selengkapnya dapat dilihat pada Gambar 4.3.



Gambar 4.3 Load Wizard

4. Setelah proses selesai maka data dapat langsung digunakan.

4.3.2 Data *Dummy*

Selain data asli yang didapatkan penulis di atas, masih terdapat beberapa tabel yang belum memiliki data sehingga diperlukan pembuatan data *dummy* untuk mengisinya. Adapun tabel-tabel yang akan berisi data *dummy* antara lain adalah tabel mahasiswa, tabel mahasiswa biodata, tabel mahasiswa orang tua dan tabel KHS. Data *dummy* dapat dipilih karena pada penelitian ini penulis menitik beratkan pada performa *data mart* bukan pada kebenaran informasi. Untuk membuat data *dummy* maka penulis menggunakan bantuan bahasa pemrograman *java* dan objek basis data *stored procedures*. Yang perlu diperhatikan adalah data *dummy* dapat dibuat hanya jika data-data asli di atas telah selesai di impor dan siap digunakan di sistem basis data SIAKAD. Adapun berikut cara pembuatan data *dummy* yang penulis gunakan.

1. Data Mahasiswa, Mahasiswa Biodata dan Orang Tua Mahasiswa
   Data Mahasiswa, Mahasiswa Biodata dan Orang Tua Mahasiswa dibuat secara bersamaan dan disimpan di tabel sementara yang memiliki semua kolom dari 3 tabel tersebut. Setelah data selesai di buat di tabel sementara maka dengan bantuan *SQL* data mulai dipisah-pisahkan sesuai tabel tujuannya. Untuk membuat data mahasiswa penulis menggunakan bantuan bahasa pemrograman *java*. Berikut adalah kode sumbernya:

```
1  private static void generateStageMhs(Connection conn, String fakultas
   , String jurusan, String jenjang, String prodi, int jumlah) {
2    try {
3      String sql = "SELECT K_KOTA, K_PROPINSI, K_NEGARA, K_SEKOLAH
       FROM SIAKAD.SEKOLAH";
4      Statement sta = conn.createStatement(ResultSet.
       TYPE_SCROLL_INSENSITIVE, ResultSet.CONCUR_READ_ONLY);
5      ResultSet dataSekolah = sta.executeQuery(sql);
6      dataSekolah.last();
7      int jmlSekolah = dataSekolah.getRow();
8      String[] seleksi = new String[] {"01", "02", "03", "04", "05"};
9      String[] kerja = new String[] {"01", "02", "03", "04", "05", "06"
       , "07", "08", "09", "10", "11", "12", "13", "14", "15"};
10     int[] angkatan = new int[] {2014, 2013, 2012, 2011, 2010};
11     Random rand = new Random();
12     for (int i = 0; i < jumlah; i++) {
13       String strId = String.format("%05d", i);
14       int rowIdSklh = rand.nextInt(jmlSekolah) + 1;
15       int rowIdSeleksi = rand.nextInt(seleksi.length);
16       int rowIdKerja = rand.nextInt(kerja.length);
17       dataSekolah.absolute(rowIdSklh);
18       sql = "INSERT INTO DATA_STAGING.DETAIL_MAHASISWA"
19         + "(NIM, K_JENJANG, K_JURUSAN, K_FAKULTAS, K_PROGRAM_STUDI, "
20         + "K_SELEKSI, NAMA, ANGKATAN, K_KOTA, K_NEGARA, K_PROPINSI, "
21         + "K_SEKOLAH, TMP_LAHIR, TGL_LAHIR, ALAMAT_ASAL, ALAMAT_KOS, "
22         + "NO_TELP_RMH, NO_TELP_KOS, NO_HP, NAMA_AYAH, K_KERJA_AYAH, "
23         + "PENGHASILAN_AYAH, NAMA_IBU, K_KERJA_IBU, PENGHASILAN_IBU, "
24         + "ALAMAT_ORTU, K_KOTA_ORTU, K_PROPINSI_ORTU, K_NEGARA_ORTU,
           NO_HP_ORTU) VALUES("
25           + "'UB-"+ fakultas + prodi + strId +"', "
26           + "'"+ jenjang +"', "
27           + "'"+ jurusan +"', "
```

Gambar 4.4 Mahasiswa Generator-1

```
28           + "'"+ fakultas +"', "
29           + "'"+ prodi +"', "
30           + "'"+ seleksi[rowIdSeleksi] +"', "
31           + "'Mahasiswa-"+ fakultas + prodi + strId +"', "
32           + "'"+ angkatan[rowIdSeleksi] +"', "
33           + "'"+ dataSekolah.getString("K_KOTA") +"', "
34           + "'"+ dataSekolah.getString("K_NEGARA") +"', "
35           + "'"+ dataSekolah.getString("K_PROPINSI") +"', "
36           + "'"+ dataSekolah.getString("K_SEKOLAH") +"', "
37           + "'"+ dataSekolah.getString("K_KOTA") +"', "
38           + "'"+ "1991-01-01" +"', "
39           + "'Alamat-"+ fakultas + prodi + strId +"', "
40           + "'Alamat Kos - "+ fakultas + prodi + strId +"', "
41           + "'Telp-"+ fakultas + prodi + strId +"', "
42           + "'Telp Kos-"+ fakultas + prodi + strId +"', "
43           + "'HP-"+ fakultas + prodi + strId +"', "
44           + "'Ayah-"+ fakultas + prodi + strId +"', "
45           + "'"+ kerja[rowIdKerja] +"', "
46           + "'"+ Integer.toString(2000000 + i) +"', "
47           + "'Ibu-"+ fakultas + prodi + strId +"', "
48           + "'"+ kerja[rowIdKerja] +"', "
49           + "'"+ Integer.toString(1500000 + i) +"', "
50           + "'Alamat-"+ fakultas + prodi + strId +"', "
51           + "'"+ dataSekolah.getString("K_KOTA") +"', "
52           + "'"+ dataSekolah.getString("K_PROPINSI") +"', "
53           + "'"+ dataSekolah.getString("K_NEGARA") +"', "
54           + "'HP ORTU-"+ fakultas + prodi + strId +"' "
55         + ")";
56       sta = conn.createStatement();
57       System.out.println(sql);
58       sta.execute(sql);
59     }
```

Gambar 4.5 Mahasiswa Generator-2

Dari Gambar 4.4 dan Gambar 4.5 yang merupakan bahasa pemrograman *java* berikut adalah penjelasannya:

Baris 1	:	Merupakan deklarasi fungsi untuk membuat data mahasiswa
Baris 3-7	:	Mendapatkan data sekolah dan daerah asal sebagai bahan data acak yang nantinya akan dipasangkan dengan data mahasiswa
Baris 8-10	:	Mendeklarasikan data kode seleksi, kode kerja dan angkatan sebagai bahan dasar data acak
Baris 13-17	:	Mengacak data untuk dipasangkan dengan data mahasiswa
Baris 18-58	:	Merupakan proses menyimpan data mahasiswa ke tabel sementara. Nama mahasiswa akan dibuat dengan kombinasi teks "Mahasiswa-" + kode fakultas + kode jurusan + kode prodi. Begitu juga dengan data alamat, telepon, nama ayah, nama ibu. Sementara data penghasilan ayah dan ibu akan diisi oleh angka acak

2. Data KHS

Data KHS hanya dapat dibuat hanya jika data mahasiswa telah selesai dibuat. Diasumsikan bahwa setiap mahasiswa mengambil 48 mata kuliah di program studinya. Data KHS dibuat dengan bantuan *stored prosedure* DB2.

```
1  CREATE PROCEDURE GENERATE_KHS(PRODI VARCHAR(9))
2  BEGIN
3    DECLARE NILAI INTEGER;
4    DECLARE KPRODI VARCHAR(9);
5    FOR V AS CUR1 CURSOR FOR
6      SELECT NIM FROM DB2ADMIN.MAHASISWA WHERE K_PROGRAM_STUDI = PRODI
7    DO
8      FOR X AS CUR2 CURSOR FOR
9      SELECT ID_MK FROM DB2ADMIN.MATA_KULIAH WHERE K_PROGRAM_STUDI =
       PRODI ORDER BY RAND() FETCH FIRST 48 ROWS ONLY
10     DO
11     SET NILAI = CAST((FLOOR(RAND()*10)) AS INTEGER);
12     INSERT INTO DB2ADMIN.KHS(NIM, THN_AKADEMIK, IS_GANJIL, IS_PENDEK
       , ID_MK, K_NILAI)
13       VALUES(V.NIM, FLOOR(RAND()*(2015-2010)+2010), CAST((FLOOR(RAND
         ()*2)) AS CHAR(1)), CAST((FLOOR(RAND()*2)) AS CHAR(1)), X.
         ID_MK,
14         CASE  NILAI
15           WHEN 1 THEN '001'
16           WHEN 2 THEN '002'
17           WHEN 3 THEN '003'
18           WHEN 4 THEN '004'
19           WHEN 5 THEN '005'
20           WHEN 6 THEN '006'
21           WHEN 7 THEN '007'
22           WHEN 8 THEN '008'
23           ELSE '009'
24         END
25       );
26     END FOR;
27   END FOR;
28 END@
```

Gambar 4.6 Kode Stored Procedur KHS Generator

Penjelasan kode sumber pada Gambar 4.6 di atas adalah:

Baris 1	:	Merupakan deklarasi *stored procedure*.
Baris 3-4	:	Deklarasi variabel NILAI untuk menampung nilai acak, dan variabel KPRODI untuk menampung kode program studi yang didapatkan.
Baris 5-7	:	Mendapatkan data mahasiswa per program studi agar lebih ringan.
Baris 8-10	:	Mendapatkan data 48 mata kuliah acak yang ada di program studi bersangkutan
Baris 11	:	Membuat nilai acak
Baris 12-24	:	Menyimpan data ke tabel KHS. Dengan ketentuan kolom angkatan akan diisi acak dengan rentang 2010-2015. Kolom IS_PENDEK dan IS_GANJIL akan diisi acak antara 0 dan 1.

4.4 Implementasi *Data Mart*

Implementasi *data mart* dibagi menjadi dua tahapan yaitu tahapan persiapan dan eksekusi. Dalam tahapan persiapan dilakukan konfigurasi basis data untuk siap menjalankan tahapan eksekusi. Sementara dalam tahapan eksekusi akan melakukan implementasi skema *data mart* (model multidimensional) dengan cara mengubah rancangan skema yang ada di bab 3 menjadi DDL (*Data Definition Language*). Setelah itu DDL di *deploy* di *server data mart*. Adapun sistem basis data yang digunakan untuk implementasi adalah IBM DB2.

4.4.1 Persiapan Basis Data

Tak dapat dipungkiri bahwa kebutuhan *data mart* akan data berbeda jauh dengan basis data transaksional seperti OLTP. Penyimpanan yang besar dan kecepatan baca yang tinggi merupakan kebutuhan dasar dari *data mart*. Sehingga konfigurasi pada basis data mutlak diperlukan.

Pada penelitian ini penulis menggunakan skenario partisi basis data. Basis data akan disimpan pada tiga *hardisk* yang berbeda. Dua *hardisk* untuk menangani data dan satu lainnya digunakan untuk menyimpan dokumen *log*. Hal ini dilakukan untuk mencapai kestabilan pembacaan dan penyimpanan data. Beban kerja akan dibagi ke tiga *hardisk*. Sehingga performa *data mart* akan semakin meningkat dan stabil. Konfigurasi ini akan dilakukan dengan bantuan *command line*.

```
[nurcahyo@192 ~]$ su
Password:
[root@192 nurcahyo]# mkdir /data1/dwhdb
[root@192 nurcahyo]# mkdir /data2/dwhdb
[root@192 nurcahyo]# mkdir /logarchmeth1/dwhdb
[root@192 nurcahyo]# groups db2admin
db2admin : db2admin db2as
[root@192 nurcahyo]# chown -R db2admin.db2admin /data1/dwhdb
[root@192 nurcahyo]# chown -R db2admin.db2admin /data2/dwhdb
[root@192 nurcahyo]# chown -R db2admin.db2admin /logarchmeth1/dwhdb
[root@192 nurcahyo]# su db2admin
[db2admin@192 nurcahyo]$ cd /home/db2admin
[db2admin@192 ~]$ mkdir dwhdb
[db2admin@192 ~]$ db2 create database DWHDB on /data1/dwhdb/, /data2/dwhdb/ dbpa
th on /home/db2admin/dwhdb/
DB20000I  The CREATE DATABASE command completed successfully.
```

Gambar 4.7 Kode Konfigurasi Server

Gambar 4.7 menunjukkan proses konfigurasi basis data pada *server* dengan menggunakan *command line*. Berikut penjelasannya:

Baris 1 : Adalah perintah untuk berganti pengguna menjadi pengguna *root*.

Baris 3-5 : Perintah membuat folder "dwhdb" di setiap *hardisk* yang nantinya akan menjadi tempat penyimpanan data.

Baris 6-10 : Perintah mengganti kepemilikan folder "dwhdb" di setiap hardisk menjadi milik pengguna "db2admin". Sehingga akan dapat digunakan oleh basis data.

Baris 11-13 : Perintah membuat folder "dwhdb" di *home folder* pengguna "db2admin". Yang nantinya akan menjadi tempat penyimpanan *metadata* basis data.

Baris 14-16 : Perintah membuat basis data dengan nama "DWHDB" dengan ketentuan:

- disimpan di hardisk "/data1" dan "/data2"
- dokumen lognya disimpan di "/logarchmeth1"
- metadatanya disimpan di "/home/db2admin/dwhdb"

Setelah selesai melakukan konfigurasi melalui *command line* tahap selanjutnya adalah konfigurasi menggunakan perintah SQL (*Structured Query Language*) dengan perintah :

```
1. CREATE BUFFERPOOL DWHBP IMMEDIATE ALL DBPARTITIONNUMS SIZE
   1000 AUTOMATIC PAGESIZE 32768;
2. CREATE REGULAR TABLESPACE USER_SPACE1 IN DATABASE PARTITION
   GROUP IBMDEFAULTGROUP PAGESIZE 32768 MANAGED BY AUTOMATIC
   STORAGE USING STOGROUP IBMSTOGROUP AUTORESIZE YES BUFFERPOOL
   DWHBP OVERHEAD INHERIT TRANSFERRATE INHERIT DROPPED TABLE
   RECOVERY ON DATA TAG NONE;
3. CREATE REGULAR TABLESPACE INDEX_SPACE1 IN DATABASE PARTITION
   GROUP IBMDEFAULTGROUP PAGESIZE 32768 MANAGED BY AUTOMATIC
   STORAGE USING STOGROUP IBMSTOGROUP AUTORESIZE YES BUFFERPOOL
   DWHBP OVERHEAD INHERIT TRANSFERRATE INHERIT DROPPED TABLE
   RECOVERY ON DATA TAG NONE;
4. CREATE LARGE TABLESPACE LARGE_SPACE1 IN DATABASE PARTITION
   GROUP IBMDEFAULTGROUP PAGESIZE 32768 MANAGED BY AUTOMATIC
   STORAGE USING STOGROUP IBMSTOGROUP AUTORESIZE YES BUFFERPOOL
   DWHBP OVERHEAD INHERIT TRANSFERRATE INHERIT DROPPED TABLE
   RECOVERY ON DATA TAG NONE;
```

Berikut penjelasan kode SQL di atas:

Baris 1 : Adalah perintah untuk membuat *buffer pool* baru dengan ketentuan memiliki *page size* 32MB.

Baris 2 : Adalah perintah untuk membuat *tabel space* reguler dengan nama "USER_SPACE1". *Tabel space* ini akan digunakan sebagai tempat penyimpan data reguler.

Baris 3 : Adalah perintah untuk membuat *tabel space* reguler dengan nama "INDEX_SPACE1". *Tabel space* ini akan digunakan sebagai tempat penyimpan data index.

Baris 4 : Adalah perintah untuk membuat *long tabel space* dengan nama "LARGE_SPACE1". *Tabel space* ini berguna sebagai tempat penyimpan *large object* seperti gambar dan teks panjang.

Alasan pembuatan *buffer pool* adalah karena *buffer pool* berguna sebagai penyimpanan sementara pada basis data. Semakin ukurannya maka akan semakin tinggi kecamatan tulisnya dan lambat bacanya dan sebaliknya. Karena *data mart* membutuhkan kecepatan baca yang tinggi maka akan menggunakan *buffer pool* dengan ukuran besar. Sementara pembuatan tiga *table space* digunakan sebagai pemisahan penyimpanan antara data reguler, data *index* dan data berukuran besar. Hal ini dilakukan untuk memastikan kestabilan dan kecepatan *data mart*.

4.4.2 Tabel Dimensi Fakultas

Dimensi fakultas adalah dimensi yang berperan untuk menyimpan detail informasi fakultas di Universitas Brawijaya. Berikut adalah DDL untuk membuat Tabel dimensi fakultas di *data mart server*.

```
1. CREATE TABLE MART.DIM_FAKULTAS (
2.   ID_FAKULTAS VARCHAR(9) NOT NULL,
3.   NAMA VARCHAR(100) NULL,
4.   PRIMARY KEY(ID_FAKULTAS)
5. ) IN USER SPACE1 INDEX IN INDEX SPACE1 LONG IN LARGE SPACE1;
```

Penjelasan kode DDL di atas adalah sebagai berikut:

Baris 1 : Adalah perintah untuk membuat tabel dengan nama DIM_FAKULTAS. Dengan ketentuan tabel tersebut akan berada di *schema* MART.

Baris 2-3 Adalah pendeklarasian atribut yang akan dimiliki oleh DIM_FAKULTAS. Adapun atributnya adalah ID_FAKULTAS dengan tipe data VARCHAR memiliki panjang karakter 9 dan NAMA dengan tipe data VARCHAR memiliki panjang karakter 100.

Baris 4 Adalah pendeklarasian bahwa atribut ID_FAKULTAS akan menjadi *primary Key* dari tabel DIM_FAKULTAS.

Baris 5 Pendeklarasian bahwa tabel DIM_FAKULTAS akan menggunakan *table space* "USER_SPACE1" sebagai penyimpanan data reguler, *table space* "INDEX_SPACE1" sebagai penyimpanan data *index* dan *table space* "LARGE_SPACE1" sebagai penyimpanan data besar.

4.4.3 Tabel Dimensi Jurusan

Dimensi jurusan adalah dimensi yang berperan untuk menyimpan detail informasi jurusan di Universitas Brawijaya. Dikarenakan dimensi jurusan berada di bawah dimensi fakultas maka untuk mengeksekusi perintah DDL harus dilakukan setelah dimensi fakultas selesai di *deploy* ke server. Berikut adalah DDL untuk membuat Tabel dimensi jurusan di *data mart server*.

```
1. CREATE TABLE MART.DIM_JURUSAN (
2.   ID_JURUSAN VARCHAR(9) NOT NULL,
3.   ID_FAKULTAS VARCHAR(9) NOT NULL,
4.   NAMA VARCHAR(100) NULL,
5.   PRIMARY KEY(ID_JURUSAN, ID_FAKULTAS),
6.   FOREIGN KEY(ID_FAKULTAS)
7.      REFERENCES MART.DIM_FAKULTAS(ID_FAKULTAS)
8. ) IN USER_SPACE1 INDEX IN INDEX_SPACE1 LONG IN LARGE_SPACE1;
```

Berikut adalah penjelasan dari kode DDL diatas:

Baris 1 : Adalah perintah untuk membuat tabel dengan nama DIM_JURUSAN. Dengan ketentuan tabel tersebut akan berada di *schema* MART.

Baris 2-4 : Perintah untuk mendeklarasikan atribut tabel DIM_JURUSAN. Adapun atribut-atribut tersebut adalah ID_JURUSAN dengan tipe data VARCHAR dan panjang karakter 9, ID_FAKULTAS dengan tipe data VARCHAR dan panjang karakter 9 dan NAMA dengan tipe data VARCHAR dan panjang karakter 100.

Baris 5 : Adalah pendeklarasian atribut ID_JURUSAN dan ID_FAKULTAS sebagai *primary key*

Baris 6-7 : Adalah pendeklarasian *foreign key* sebagai penyambung antara dimensi jurusan dan dimensi fakultas. Adapun atribut yang berperan sebagai *key* adalah ID_FAKULTAS.

Baris 8 : Pendeklarasian bahwa tabel DIM_JURUSAN akan menggunakan *table space* "USER_SPACE1" sebagai penyimpanan data reguler, *table space* "INDEX_SPACE1" sebagai penyimpanan data *index* dan *table space* "LARGE_SPACE1" sebagai penyimpanan data besar.

4.4.4 Tabel Dimensi Jenjang

Dimensi jenjang adalah dimensi yang berperan untuk menyimpan detail informasi tingkat jenjang di Universitas Brawijaya. Berikut adalah DDL untuk membuat Tabel dimensi jenjang di *data mart server*.

```
1. CREATE TABLE MART.DIM_JENJANG (
2.   ID_JENJANG VARCHAR(9) NOT NULL,
3.   NAMA VARCHAR(100) NULL,
4.   PRIMARY KEY(ID_JENJANG)
5. ) IN USER_SPACE1 INDEX IN INDEX_SPACE1 LONG IN LARGE_SPACE1;
```

Berikut adalah penjelasan kode DDL diatas:

Baris 1 : Adalah perintah untuk membuat tabel dengan nama DIM_JENJANG. Dengan ketentuan tabel tersebut akan berada di *schema* MART.

Baris 2-3 : Adalah pendeklarasian atribut yang akan dimiliki oleh DIM_JENJANG. Adapun atributnya adalah ID_JENJANG dengan tipe data VARCHAR memiliki panjang karakter 9 dan NAMA dengan tipe data VARCHAR memiliki panjang karakter 100.

Baris 4 : Adalah pendeklarasian bahwa atribut ID_JENJANG akan menjadi *primary Key* dari tabel DIM_JENJANG.

Baris 5 : Pendeklarasian bahwa tabel DIM_JENJANG akan menggunakan *table space* "USER_SPACE1" sebagai penyimpanan data reguler, *table space* "INDEX_SPACE1" sebagai penyimpanan data *index* dan *table space* "LARGE_SPACE1" sebagai penyimpanan data besar.

4.4.5 Tabel Dimensi Program Studi

Dimensi program studi adalah dimensi yang berperan untuk menyimpan detail informasi program studi di Universitas Brawijaya. Dikarenakan dimensi program studi berada di bawah dimensi fakultas, dimensi jenjang dan dimensi jurusan maka untuk mengeksekusi perintah DDL harus dilakukan setelah dimensi fakultas, dimensi jenjang dan dimensi jurusan selesai di *deploy* ke server. Berikut adalah DDL untuk membuat Tabel dimensi jurusan di *data mart server*.

```
1. CREATE TABLE MART.DIM_PRODI (
2.   ID_PRODI VARCHAR(9) NOT NULL,
3.   ID_FAKULTAS VARCHAR(9) NOT NULL,
4.   ID_JURUSAN VARCHAR(9) NOT NULL,
5.   ID_JENJANG VARCHAR(9) NOT NULL,
6.   NAMA VARCHAR(100) NULL,
7.   PRIMARY KEY(ID_PRODI, ID_FAKULTAS, ID_JURUSAN,
   ID_JENJANG),
8.   FOREIGN KEY(ID_JURUSAN, ID_FAKULTAS)
9.     REFERENCES MART.DIM_JURUSAN(ID_JURUSAN, ID_FAKULTAS),
10.  FOREIGN KEY(ID_JENJANG)
11.      REFERENCES MART.DIM_JENJANG(ID_JENJANG)
12.  ) IN USER_SPACE1 INDEX IN INDEX_SPACE1 LONG IN
   LARGE_SPACE1;
```

Berikut adalah penjelasan kode DDL diatas:

Baris 1 : Adalah perintah untuk membuat tabel dengan nama DIM_PRODI. Dengan ketentuan tabel tersebut akan berada di *schema* MART.

Baris 2-6 : Perintah untuk mendeklarasikan atribut tabel DIM_PRODI. Adapun atribut-atribut tersebut adalah ID_PRODI dengan tipe data VARCHAR dan panjang karakter 9, ID_JENJANG dengan tipe data VARCHAR dan panjang karakter 9, ID_JURUSAN dengan tipe data VARCHAR dan panjang karakter 9, ID_FAKULTAS dengan tipe data VARCHAR dan panjang karakter 9 dan NAMA dengan tipe data VARCHAR dan panjang karakter 100.

Baris 7 : Adalah pendeklarasian atribut ID_PRODI, ID_ JURUSAN, ID_JENJANG dan ID_FAKULTAS sebagai *primary key*

Baris 8-9 : Adalah pendeklarasian *foreign key* sebagai penyambung antara dimensi prodi dan dimensi jurusan. Adapun atribut yang berperan sebagai *key* adalah ID_FAKULTAS dan ID_JURUSAN.

Baris 10-11 : Adalah pendeklarasian *foreign key* sebagai penyambung antara dimensi prodi dan dimensi jenjang. Adapun atribut yang berperan sebagai *key* adalah ID_JENJANG.

Baris 12 : Pendeklarasian bahwa tabel DIM_PRODI akan menggunakan *table space* "USER_SPACE1" sebagai penyimpanan data

reguler, *table space* "INDEX_SPACE1" sebagai penyimpanan data *index* dan *table space* "LARGE_SPACE1" sebagai penyimpanan data besar.

4.4.6 Tabel Dimensi Seleksi

Dimensi seleksi adalah dimensi yang berperan untuk menyimpan detail informasi tipe seleksi masuk mahasiswa. Berikut adalah DDL untuk membuat Tabel dimensi seleksi di *data mart server*.

```
1. CREATE TABLE MART.DIM_SELEKSI (
2.   ID_SELEKSI VARCHAR(9) NOT NULL,
3.   NAMA VARCHAR(100) NULL,
4.   PRIMARY KEY(ID_SELEKSI)
5. ) IN USER_SPACE1 INDEX IN INDEX_SPACE1 LONG IN LARGE_SPACE1;
```

Berikut adalah penjelasan dari kode DDL diatas:

Baris 1 : Adalah perintah untuk membuat tabel dengan nama DIM_SELEKSI. Dengan ketentuan tabel tersebut akan berada di *schema* MART.

Baris 2-3 : Adalah pendeklarasian atribut yang akan dimiliki oleh DIM_SELEKSI. Adapun atributnya adalah ID_SELEKSI dengan tipe data VARCHAR memiliki panjang karakter 9 dan NAMA dengan tipe data VARCHAR memiliki panjang karakter 100.

Baris 4 : Adalah pendeklarasian bahwa atribut ID_SELEKSI akan menjadi *primary Key* dari tabel DIM_SELEKSI.

Baris 5 : Pendeklarasian bahwa tabel DIM_SELEKSI akan menggunakan *table space* "USER_SPACE1" sebagai penyimpanan data reguler, *table space* "INDEX_SPACE1" sebagai penyimpanan data *index* dan *table space* "LARGE_SPACE1" sebagai penyimpanan data besar.

4.4.7 Tabel Dimensi Angkatan

Dimensi angkatan adalah dimensi yang berperan untuk menyimpan detail informasi tingkat angkatan mahasiswa. Berikut adalah DDL untuk membuat Tabel dimensi angkatan di *data mart server*.

```
1. CREATE TABLE MART.DIM_ANGKATAN (
2.   ID_ANGKATAN INTEGER NOT NULL,
3.   DESKRIPSI VARCHAR(20) NULL,
4.   PRIMARY KEY(ID_ANGKATAN)
5. ) IN USER_SPACE1 INDEX IN INDEX_SPACE1 LONG IN LARGE_SPACE1;
```

Berikut adalah penjelasan dari kode DDL di atas:

Baris 1 : Adalah perintah untuk membuat tabel dengan nama DIM_ANGKATAN. Dengan ketentuan tabel tersebut akan berada di *schema* MART.

Baris 2-3 : Adalah pendeklarasian atribut yang akan dimiliki oleh DIM_ANGKATAN. Adapun atributnya adalah ID_ANGKATAN dengan tipe data VARCHAR memiliki panjang karakter 9 dan NAMA dengan tipe data VARCHAR memiliki panjang karakter 100.

Baris 4 : Adalah pendeklarasian bahwa atribut ID_ANGKATAN akan menjadi *primary Key* dari tabel DIM_ANGKATAN.

Baris 5 : Pendeklarasian bahwa tabel DIM_ANGKATAN akan menggunakan *table space* "USER_SPACE1" sebagai penyimpanan data reguler, *table space* "INDEX_SPACE1" sebagai penyimpanan data *index* dan *table space* "LARGE_SPACE1" sebagai penyimpanan data besar.

4.4.8 Tabel Dimensi Periode Akademik

Dimensi periode akademik adalah dimensi yang berperan untuk menyimpan detail informasi periode akademik Universitas Brawijaya. Berikut adalah DDL untuk membuat Tabel dimensi periode akademik di *data mart server*.

```
1. CREATE TABLE MART.DIM_PERIODE_AKADEMIK (
2.   ID_PERIODE_AKADEMIK VARCHAR(9) NOT NULL,
3.   TAHUN INTEGER NULL,
4.   TIPE VARCHAR(10) NULL,
5.   PRIMARY KEY(ID_PERIODE_AKADEMIK)
6. ) IN USER_SPACE1 INDEX IN INDEX_SPACE1 LONG IN LARGE_SPACE1;
```

Berikut adalah penjelasan kode DDL diatas:

Baris 1 : Adalah perintah untuk membuat tabel dengan nama DIM_PERIODE_AKADEMIK. Dengan ketentuan tabel tersebut akan berada di *schema* MART.

Baris 2-4 : Adalah pendeklarasian atribut yang akan dimiliki oleh DIM_PERIODE_AKADEMIK. Adapun atributnya adalah ID_PERIODE_AKADEMIK dengan tipe data VARCHAR memiliki panjang karakter 9, TAHUN dengan tipe data INTEGER dan NAMA dengan tipe data VARCHAR memiliki panjang karakter 100.

Baris 5 : Adalah pendeklarasian bahwa atribut ID_PERIODE_AKADEMIK akan menjadi *primary Key* dari tabel DIM_PERIODE AKADEMIK.

Baris 6 : Pendeklarasian bahwa tabel DIM_PERIODE_ AKADEMIK akan menggunakan *table space* "USER_SPACE1" sebagai penyimpanan data reguler, *table space* "INDEX_SPACE1" sebagai penyimpanan data *index* dan *table space* "LARGE_SPACE1" sebagai penyimpanan data besar.

4.4.9 Tabel Dimensi Mata Kuliah

Dimensi mata kuliah adalah dimensi yang berperan untuk menyimpan detail informasi mata kuliah yang dimiliki Universitas Brawijaya. Berikut adalah DDL untuk membuat Tabel dimensi mata kuliah di *data mart server*.

```
1. CREATE TABLE MART.DIM_MATA_KULIAH (
2.   ID_MK INTEGER NOT NULL,
3.   KODE_MATA_KULIAH VARCHAR(10) NOT NULL,
4.   NAMA VARCHAR(100) NULL,
5.   SKS INTEGER NULL,
6.   PRIMARY KEY(ID_MK)
7. ) IN USER_SPACE1 INDEX IN INDEX_SPACE1 LONG IN LARGE_SPACE1;
```

Berikut adalah penjelasan dari kode DDL diatas:

Baris 1 : Adalah pe1rintah untuk membuat tabel dengan nama DIM_MATA_KULIAH. Dengan ketentuan tabel tersebut akan berada di *schema* MART.

Baris 2-5 : Adalah pendeklarasian atribut yang akan dimiliki oleh DIM_MATA_KULIAH. Adapun atributnya adalah ID_MK dengan tipe data INTEGER, KODE_MATA_ KULAH dengan tipe data VARCHAR memiliki panjang karakter 10, NAMA dengan tipe data VARCHAR memiliki panjang karakter 100 dan SKS dengan tipe data INTEGER.

Baris 6 : Adalah pendeklarasian bahwa atribut ID_MK akan menjadi *primary key* dari tabel DIM_MATA_KULIAH.

Baris 7 : Pendeklarasian bahwa tabel DIM_MATA_KULIAH akan menggunakan *table space* "USER_SPACE1" sebagai penyimpanan data reguler, *table space* "INDEX_SPACE1" sebagai penyimpanan data *index* dan *table space* "LARGE_SPACE1" sebagai penyimpanan data besar.

4.4.10 Tabel Dimensi Negara

Dimensi fakultas adalah dimensi yang berperan untuk menyimpan detail informasi negara asal mahasiswa. Berikut adalah DDL untuk membuat Tabel dimensi negara di *data mart server*.

```
1. CREATE TABLE MART.DIM_NEGARA (
2.   ID_NEGARA VARCHAR(9) NOT NULL,
3.   NAMA VARCHAR(100) NULL,
4.   PRIMARY KEY(ID_NEGARA)
5. ) IN USER_SPACE1 INDEX IN INDEX_SPACE1 LONG IN LARGE_SPACE1;
```

Berikut adalah penjelasan dari kode DDL di atas:

Baris 1 : Adalah perintah untuk membuat tabel dengan nama DIM_NEGARA. Dengan ketentuan tabel tersebut akan berada di *schema* MART.

Baris 2-3 : Adalah pendeklarasian atribut yang akan dimiliki oleh DIM_NEGARA. Adapun atributnya adalah ID_NEGARA dengan tipe data VARCHAR memiliki panjang karakter 9 dan NAMA dengan tipe data VARCHAR memiliki panjang karakter 100.

Baris 4 : Adalah pendeklarasian bahwa atribut ID_NEGARA akan menjadi *primary key* dari tabel DIM_NEGARA.

Baris 5 : Pendeklarasian bahwa tabel DIM_NEGARA akan menggunakan *table space* "USER_SPACE1" sebagai penyimpanan data reguler, *table space* "INDEX_SPACE1" sebagai penyimpanan data *index* dan *table space* "LARGE_SPACE1" sebagai penyimpanan data besar.

4.4.11 Tabel Dimensi Propinsi

Dimensi jurusan adalah dimensi yang berperan untuk menyimpan detail informasi propinsi asal mahasiswa. Dikarenakan dimensi jurusan berada di bawah dimensi negara maka untuk mengeksekusi perintah DDL harus dilakukan setelah dimensi negara selesai di *deploy* ke server. Berikut adalah DDL untuk membuat Tabel dimensi propinsi di *data mart server*.

```
1. CREATE TABLE MART.DIM_PROVINSI (
2.   ID_PROVINSI VARCHAR(9) NOT NULL,
3.   ID_NEGARA VARCHAR(9) NOT NULL,
4.   NAMA VARCHAR(100) NULL,
5.   PRIMARY KEY(ID_PROVINSI, ID_NEGARA),
6.   FOREIGN KEY(ID_NEGARA)
7.     REFERENCES MART.DIM_NEGARA(ID_NEGARA)
8. ) IN USER_SPACE1 INDEX IN INDEX_SPACE1 LONG IN LARGE_SPACE1;
```

Berikut adalah penjelasan dari kode DDL di atas:

Baris 1 : Adalah perintah untuk membuat tabel dengan nama DIM_PROVINSI. Dengan ketentuan tabel tersebut akan berada di *schema* MART.

Baris 2-4 : Perintah untuk mendeklarasikan atribut tabel DIM_PROVINSI. Adapun atribut-atribut tersebut adalah ID_PROVINSI dengan tipe data VARCHAR dan panjang karakter 9, ID_NEGARA dengan tipe data VARCHAR dan panjang karakter 9 dan NAMA dengan tipe data VARCHAR dan panjang karakter 100.

Baris 5 : Adalah pendeklarasian atribut ID_PROVINSI dan ID_NEGARA sebagai *primary key*

Baris 6-7 : Adalah pendeklarasian *foreign key* sebagai penyambung antara dimensi propinsi dan dimensi negara. Adapun atribut yang berperan sebagai *key* adalah ID_NEGARA.

Baris 8 : Pendeklarasian bahwa tabel DIM_PROVINSI akan menggunakan *table space* "USER_SPACE1" sebagai penyimpanan data reguler, *table space* "INDEX_SPACE1" sebagai penyimpanan data *index* dan *table space* "LARGE_SPACE1" sebagai penyimpanan data besar.

4.4.12 Tabel Dimensi Kota

Dimensi kota adalah dimensi yang berperan untuk menyimpan detail informasi kota asal mahasiswa. Dikarenakan dimensi kota berada di bawah dimensi negara dan dimensi propinsi maka untuk mengeksekusi perintah DDL harus dilakukan setelah dimensi negara dan dimensi propinsi selesai di *deploy* ke server. Berikut adalah DDL untuk membuat Tabel dimensi kota di *data mart server*.

```
1. CREATE TABLE MART.DIM_KOTA (
2.   ID_KOTA VARCHAR(9) NOT NULL,
3.   ID_NEGARA VARCHAR(9) NOT NULL,
4.   ID_PROVINSI VARCHAR(9) NOT NULL,
5.   NAMA VARCHAR(100) NULL,
6.   PRIMARY KEY(ID_KOTA, ID_NEGARA, ID_PROVINSI),
7.   FOREIGN KEY(ID_PROVINSI, ID_NEGARA)
8.     REFERENCES MART.DIM_PROVINSI(ID_PROVINSI, ID_NEGARA)
9. ) IN USER_SPACE1 INDEX IN INDEX_SPACE1 LONG IN LARGE_SPACE1;
```

Berikut adalah penjelasan dari kode DDL di atas:

Baris 1 : Adalah perintah untuk membuat tabel dengan nama DIM_KOTA. Dengan ketentuan tabel tersebut akan berada di *schema* MART.

Baris 2-5 : Perintah untuk mendeklarasikan atribut tabel DIM_KOTA. Adapun atribut-atribut tersebut adalah ID_KOTA dengan tipe data VARCHAR dan panjang karakter 9, ID_PROVINSI dengan tipe data VARCHAR dan panjang karakter 9, ID_NEGARA dengan tipe data VARCHAR dan panjang karakter 9 dan NAMA dengan tipe data VARCHAR dan panjang karakter 100.

Baris 6 : Adalah pendeklarasian atribut ID_PRODI, ID_ PROVINSI, dan ID_NEGARA sebagai *primary key*

Baris 7-8 : Adalah pendeklarasian *foreign key* sebagai penyambung antara dimensi kota dan dimensi propinsi. Adapun atribut yang berperan sebagai *key* adalah ID_PROVINSI dan ID_NEGARA.

Baris 9 : Pendeklarasian bahwa tabel DIM_KOTA akan menggunakan *table space* "USER_SPACE1" sebagai penyimpanan data reguler, *table space* "INDEX_SPACE1" sebagai penyimpanan data *index* dan *table space* "LARGE_SPACE1" sebagai penyimpanan data besar.

4.4.13 Tabel Dimensi Sekolah Asal

Dimensi sekolah asal adalah dimensi yang berperan untuk menyimpan detail informasi sekolah asal mahasiswa. Berikut adalah DDL untuk membuat Tabel dimensi sekolah asal di *data mart server*.

```
1. CREATE TABLE MART.DIM_SEKOLAH_ASAL (
2.   ID_SEKOLAH_ASAL VARCHAR(9) NOT NULL,
3.   NAMA VARCHAR(100) NULL,
4.   PRIMARY KEY(ID_SEKOLAH_ASAL)
5. ) IN USER_SPACE1 INDEX IN INDEX_SPACE1 LONG IN LARGE_SPACE1;
```

Berikut adalah penjelasan dari kode DDL di atas:

Baris 1 : Adalah perintah untuk membuat tabel dengan nama DIM_SEKOLAH_ASAL. Dengan ketentuan tabel tersebut akan berada di *schema* MART.

Baris 2-3 : Adalah pendeklarasian atribut yang akan dimiliki oleh DIM_SEKOLAH_ASAL. Adapun atributnya adalah ID_SEKOLAH ASAL dengan tipe data VARCHAR memiliki panjang karakter 9 dan NAMA dengan tipe data VARCHAR memiliki panjang karakter 100.

Baris 4 : Adalah pendeklarasian bahwa atribut ID_SEKOLAH_ ASAL akan menjadi *primary key* dari tabel DIM_SEKOLAH_ASAL.

Baris 5 : Pendeklarasian bahwa tabel DIM_SEKOLAH_ASAL akan menggunakan *table space* "USER_SPACE1" sebagai penyimpanan data reguler, *table space* "INDEX_SPACE1" sebagai penyimpanan data *index* dan *table space* "LARGE_SPACE1" sebagai penyimpanan data besar.

4.4.14 Tabel Dimensi Mahasiswa

Dimensi fakultas adalah dimensi yang berperan untuk menyimpan detail informasi mahasiswa. Berikut adalah DDL untuk membuat Tabel dimensi mahasiswa di *data mart server*.

```
1. CREATE TABLE MART.DIM_MAHASISWA (
2.   NIM VARCHAR(20) NOT NULL,
3.   NAMA VARCHAR(100) NULL,
4.   PRIMARY KEY(NIM)
5. ) IN USER_SPACE1 INDEX IN INDEX_SPACE1 LONG IN LARGE_SPACE1;
```

Berikut adalah penjelasan dari kode DDL diatas:

Baris 1 : Adalah perintah untuk membuat tabel dengan nama DIM_MAHASISWA. Dengan ketentuan tabel tersebut akan berada di *schema* MART.

Baris 2-3 : Adalah pendeklarasian atribut yang akan dimiliki oleh DIM_MAHASISWA. Adapun atributnya adalah NIM dengan tipe data VARCHAR memiliki panjang karakter 20 dan NAMA dengan tipe data VARCHAR memiliki panjang karakter 100.

Baris 4 : Adalah pendeklarasian bahwa atribut NIM akan menjadi *primary key* dari tabel DIM_MAHASISWA.

Baris 5 : Pendeklarasian bahwa tabel DIM_MAHASISWA akan menggunakan *table space* "USER_SPACE1" sebagai penyimpanan data reguler, *table space* "INDEX_SPACE1" sebagai penyimpanan data *index* dan *table space* "LARGE_SPACE1" sebagai penyimpanan data besar.

4.4.15 Tabel Fakta Status Mahasiswa

Tabel fakta status mahasiswa adalah tabel fakta untuk merekam *record* status mahasiswa per periode akademiknya. Untuk mendukung eksistensinya tabel fakta ini terhubung dengan beberapa tabel dimensi yaitu: dimensi fakultas, dimensi jurusan, dimensi jenjang, dimensi program studi, dimensi seleksi, dimensi angkatan, dimensi negara, dimensi provinsi, dimensi kota, dimensi sekolah asal, dimensi periode akademik dan dimensi mahasiswa. Sehingga sebelum men-*deploy* kode DDL tabel fakta status mahasiswa harus telah selesai men-*mendeploy* kode DDL dari dimensi-dimensi tersebut. Berikut adalah kode DDL tabel fakta status mahasiswa.

```
1. CREATE TABLE MART.FACT_STATUS_MHS (
2.   ID_PROVINSI VARCHAR(9) NOT NULL,
3.   ID_NEGARA VARCHAR(9) NOT NULL,
4.   ID_KOTA VARCHAR(9) NOT NULL,
5.   ID_SEKOLAH_ASAL VARCHAR(9) NOT NULL,
6.   ID_SELEKSI VARCHAR(9) NOT NULL,
7.   ID_PERIODE_AKADEMIK VARCHAR(9) NOT NULL,
8.   ID_ANGKATAN INTEGER NOT NULL,
9.   NIM VARCHAR(20) NOT NULL,
10.  ID_JENJANG VARCHAR(9) NOT NULL,
11.  ID_JURUSAN VARCHAR(9) NOT NULL,
12.  ID_FAKULTAS VARCHAR(9) NOT NULL,
13.  ID_PRODI VARCHAR(9) NOT NULL,
14.  IS_AKTIF SMALLINT NOT NULL,
15.  PRIMARY KEY(ID_PROVINSI, ID_NEGARA, ID_KOTA,
   ID_SEKOLAH_ASAL, ID_SELEKSI, ID_PERIODE_AKADEMIK,
   ID_ANGKATAN, NIM, ID_JENJANG, ID_JURUSAN, ID_FAKULTAS,
   ID_PRODI),
16.  FOREIGN KEY(ID_KOTA, ID_NEGARA, ID_PROVINSI)
17.    REFERENCES MART.DIM_KOTA(ID_KOTA, ID_NEGARA,
   ID_PROVINSI),
18.  FOREIGN KEY(ID_SEKOLAH_ASAL)
19.    REFERENCES MART.DIM_SEKOLAH_ASAL(ID_SEKOLAH_ASAL),
20.  FOREIGN KEY(ID_SELEKSI)
21.    REFERENCES MART.DIM_SELEKSI(ID_SELEKSI),
22.  FOREIGN KEY(ID_PERIODE_AKADEMIK)
23.    REFERENCES
   MART.DIM_PERIODE_AKADEMIK(ID_PERIODE_AKADEMIK),
24.  FOREIGN KEY(ID_ANGKATAN)
25.    REFERENCES MART.DIM_ANGKATAN(ID_ANGKATAN),
26.  FOREIGN KEY(NIM)
27.    REFERENCES MART.DIM_MAHASISWA(NIM),
28.  FOREIGN KEY(ID_PRODI, ID_FAKULTAS, ID_JURUSAN,
   ID_JENJANG)
29.    REFERENCES MART.DIM_PRODI(ID_PRODI, ID_FAKULTAS,
   ID_JURUSAN, ID_JENJANG)
30.  ) IN USER_SPACE1 INDEX IN INDEX_SPACE1 LONG IN
```

Berikut adalah penjelasan dari kode DDL di atas:

Baris 1 : Adalah perintah untuk membuat tabel dengan nama FACT_STATUS_MHS. Dengan ketentuan tabel tersebut akan berada di *schema* MART.

Baris 2-13 : Perintah untuk mendeklarasikan atribut tabel FACT_STATUS_MHS yang nantinya berperan sebagai penghubung antara tabel fakta dan dimensi-dimensi terkait. Atribut-atribut ini sama persis dengan atribut yang menjadi *primary key* di dimensi dimensi bersangkutan.

Baris 14 : Adalah pendeklarasian atribut IS_AKTIF yang memiliki tipe data SMALLINT. Atribut inilah yang nantinya berperan sebagai penyimpan fakta.

Baris 15 : Adalah pendeklarasian atribut ID_FAKULTAS, ID_ JURUSAN, ID_JENJANG, ID_PRODI, ID_SELEKSI, ID_ANGKATAN, ID_NEGARA ID_PROVINSI, ID_KOTA, ID_SEKOLAH_ASAL, ID_PERIODE_ AKADEMIK dan NIM sebagai *primary key*

Baris 16-17 : Adalah pendeklarasian *foreign key* sebagai penyambung antara fakta status mahasiswa dan dimensi kota. Adapun atribut yang berperan sebagai *key* adalah ID_KOTA, ID_PROVINSI dan ID_NEGARA.

Baris 18-19 Adalah pendeklarasian *foreign key* sebagai penyambung antara fakta status mahasiswa dan dimensi sekolah asal. Adapun atribut yang berperan sebagai *key* adalah ID_SEKOLAH_ASAL

Baris 20-21 Adalah pendeklarasian *foreign key* sebagai penyambung antara fakta status mahasiswa dan dimensi seleksi. Adapun atribut yang berperan sebagai *key* adalah ID_SELEKSI

Baris 22-23 Adalah pendeklarasian *foreign key* sebagai penyambung antara fakta status mahasiswa dan dimensi periode akademik. Adapun atribut yang berperan sebagai *key* adalah ID_PERIODE_AKADEMIK

Baris 24-25 Adalah pendeklarasian *foreign key* sebagai penyambung antara fakta status mahasiswa dan dimensi angkatan. Adapun atribut yang berperan sebagai *key* adalah ID_ANGKATAN

Baris 26-27 Adalah pendeklarasian *foreign key* sebagai penyambung antara fakta status mahasiswa dan dimensi mahasiswa. Adapun atribut yang berperan sebagai *key* adalah NIM

Baris 28-29 : Adalah pendeklarasian *foreign key* sebagai penyambung antara fakta status mahasiswa dan dimensi program studi. Adapun atribut yang berperan sebagai *key* adalah ID_PRODI, ID_JURUSAN, ID_JENJANG dan ID_FAKULTAS.

Baris 30 : Pendeklarasian bahwa tabel FACT_STATUS_MHS akan menggunakan *table space* "USER_SPACE1" sebagai penyimpanan data reguler, *table space* "INDEX_SPACE1" sebagai penyimpanan data *index* dan *table space* "LARGE_SPACE1" sebagai penyimpanan data besar.

4.4.16 Tabel Fakta Entri KHS

Tabel fakta entri KHS adalah tabel fakta untuk merekam *record* hasil studi mahasiswa per periode akademiknya. Untuk mendukung eksistensinya tabel fakta ini terhubung dengan beberapa tabel dimensi yaitu: dimensi fakultas, dimensi jurusan, dimensi jenjang, dimensi program studi, dimensi seleksi, dimensi angkatan, dimensi negara, dimensi provinsi, dimensi kota, dimensi sekolah asal,

dimensi periode akademik, dimensi mata kuliah dan dimensi mahasiswa. Sehingga sebelum men-*deploy* kode DDL tabel fakta status mahasiswa harus telah selesai men-*mendeploy* kode DDL dari dimensi-dimensi tersebut. Berikut adalah kode DDL tabel fakta status mahasiswa.

```
1. CREATE TABLE MART.FACT_ENTRI_KHS (
2.   ID_PERIODE_AKADEMIK VARCHAR(9) NOT NULL,
3.   ID_SELEKSI VARCHAR(9) NOT NULL,
4.   ID_ANGKATAN INTEGER NOT NULL,
5.   ID_PROVINSI VARCHAR(9) NOT NULL,
6.   ID_NEGARA VARCHAR(9) NOT NULL,
7.   ID_KOTA VARCHAR(9) NOT NULL,
8.   ID_SEKOLAH_ASAL VARCHAR(9) NOT NULL,
9.   NIM VARCHAR(20) NOT NULL,
10.  ID_MK INTEGER NOT NULL,
11.  ID_JENJANG VARCHAR(9) NOT NULL,
12.  ID_JURUSAN VARCHAR(9) NOT NULL,
13.  ID_FAKULTAS VARCHAR(9) NOT NULL,
14.  ID_PRODI VARCHAR(9) NOT NULL,
15.  NILAI_ANGKA DOUBLE NULL,
16.  NILAI_HURUF VARCHAR(2) NULL,
17.  PRIMARY KEY(ID_PERIODE_AKADEMIK, ID_SELEKSI, ID_ANGKATAN,
   ID_PROVINSI, ID_NEGARA, ID_KOTA, ID_SEKOLAH_ASAL, NIM,
   ID_MK, ID_JENJANG, ID_JURUSAN, ID_FAKULTAS, ID_PRODI),
18.  FOREIGN KEY(ID_PERIODE_AKADEMIK)
19.    REFERENCES
   MART.DIM_PERIODE_AKADEMIK(ID_PERIODE_AKADEMIK),
20.  FOREIGN KEY(ID_SELEKSI)
21.    REFERENCES MART.DIM_SELEKSI(ID_SELEKSI),
22.  FOREIGN KEY(ID_ANGKATAN)
23.    REFERENCES MART.DIM_ANGKATAN(ID_ANGKATAN),
24.  FOREIGN KEY(ID_KOTA, ID_NEGARA, ID_PROVINSI)
25.    REFERENCES MART.DIM_KOTA(ID_KOTA, ID_NEGARA,
   ID_PROVINSI),
26.  FOREIGN KEY(ID_SEKOLAH_ASAL)
27.    REFERENCES MART.DIM_SEKOLAH_ASAL(ID_SEKOLAH_ASAL),
28.  FOREIGN KEY(NIM)
29.    REFERENCES MART.DIM_MAHASISWA(NIM),
30.  FOREIGN KEY(ID_MK)
31.    REFERENCES MART.DIM_MATA_KULIAH(ID_MK),
32.  FOREIGN KEY(ID_PRODI, ID_FAKULTAS, ID_JURUSAN,
   ID_JENJANG)
33.    REFERENCES MART.DIM_PRODI(ID_PRODI, ID_FAKULTAS,
   ID_JURUSAN, ID_JENJANG)
34.  ) IN USER_SPACE1 INDEX IN INDEX_SPACE1 LONG IN
   LARGE_SPACE1;
```

Adapun penjelasan dari kode DDL di atas adalah:

Baris 1 : Adalah perintah untuk membuat tabel dengan nama FACT_ENTRI_KHS. Dengan ketentuan tabel tersebut akan berada di *schema* MART.

Baris 2-14 : Perintah untuk mendeklarasikan atribut tabel FACT_ ENTRI_KHS yang nantinya berperan sebagai penghubung

antara tabel fakta dan dimensi-dimensi terkait. Atribut-atribut ini sama persis dengan atribut yang menjadi *primary key* di dimensi dimensi bersangkutan.

Baris 15-16 : Adalah pendeklarasian atribut NILAI_ANGKA yang memiliki tipe data DOUBLE dan atribut NILAI_HURUF yang memiliki tipe data VARCHAR dengan panjang 2 karakter. Atribut inilah yang nantinya berperan sebagai penyimpan fakta.

Baris 17 : Adalah pendeklarasian atribut ID_FAKULTAS, ID_ JURUSAN, ID_JENJANG, ID_PRODI, ID_SELEKSI, ID_ANGKATAN, ID_NEGARA ID_PROVINSI, ID_KOTA, ID_SEKOLAH_ASAL, ID_PERIODE_ AKADEMIK, ID_MK dan NIM sebagai *primary key*

Baris 18-19 Adalah pendeklarasian *foreign key* sebagai penyambung antara fakta entri KHS dan dimensi periode akademik. Adapun atribut yang berperan sebagai *key* adalah ID_PERIODE_AKADEMIK

Baris 20-21 Adalah pendeklarasian *foreign key* sebagai penyambung antara fakta entri KHS dan dimensi seleksi. Adapun atribut yang berperan sebagai *key* adalah ID_SELEKSI

Baris 22-23 Adalah pendeklarasian *foreign key* sebagai penyambung antara fakta entri KHS dan dimensi angkatan. Adapun atribut yang berperan sebagai *key* adalah ID_ANGKATAN

Baris 24-25 : Adalah pendeklarasian *foreign key* sebagai penyambung antara fakta entri KHS dan dimensi kota. Adapun atribut yang berperan sebagai *key* adalah ID_KOTA, ID_PROVINSI dan ID_NEGARA.

Baris 26-27 Adalah pendeklarasian *foreign key* sebagai penyambung antara fakta entri KHS dan dimensi sekolah asal. Adapun atribut yang berperan sebagai *key* adalah ID_SEKOLAH_ASAL

Baris 28-29 Adalah pendeklarasian *foreign key* sebagai penyambung antara fakta entri KHS dan dimensi mahasiswa. Adapun atribut yang berperan sebagai *key* adalah NIM

Baris 30-31 Adalah pendeklarasian *foreign key* sebagai penyambung antara fakta entri KHS dan dimensi mata kuliah. Adapun atribut yang berperan sebagai *key* adalah ID_MK

Baris 32-33 : Adalah pendeklarasian *foreign key* sebagai penyambung antara fakta entri KHS dan dimensi program studi. Adapun atribut yang berperan sebagai *key* adalah ID_PRODI, ID_JURUSAN, ID_JENJANG dan ID_FAKULTAS.

Baris 34 : Pendeklarasian bahwa tabel FACT_ENTRI_KHS akan menggunakan *table space* “USER_SPACE1” sebagai

penyimpanan data reguler, *table space* "INDEX_SPACE1" sebagai penyimpanan data *index* dan *table space* "LARGE_SPACE1" sebagai penyimpanan data besar.

4.5 Proses ETL

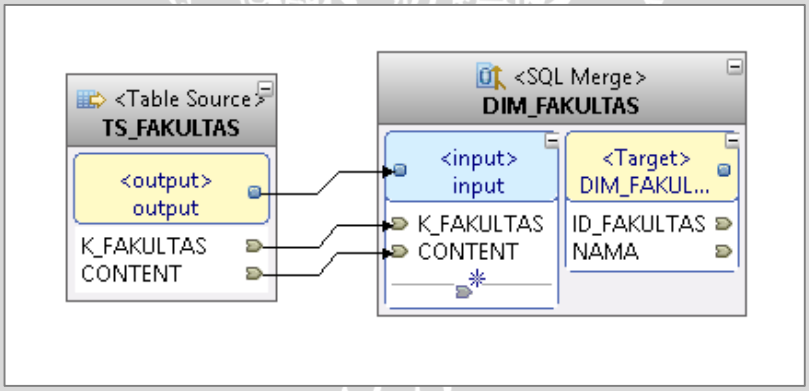
Implementasi proses ETL dilakukan dengan bantuan perangkat lunak *IBM Design Studio* dan *IBM Web Sphere Application Server*. Implementasi akan melalui beberapa tahap yaitu: tahap pembuatan alur data, tahap pembuatan alur kontrol, tahap pemasangan dan penjadwalan.

4.5.1 Pembuatan Alur Data

Pembuatan alur data didasarkan pada perancangan ETL pada bab 3. Tujuannya untuk mengontrol proses bergeraknya data mulai dari proses pengambilan data dari sumber data kemudian transformasi data di *data staging* dan akhirnya penyimpanan data di *data mart*. Sehingga tiap-tiap tabel dimensi dan fakta yang ada di *data mart* akan memiliki satu alur data.

4.5.1.1 Dimensi Fakultas

Sesuai namanya dimensi fakultas adalah dimensi untuk menyimpan detail fakultas asal mahasiswa. Dimensi ini dapat dikatakan berdiri sendiri tanpa bergantung adanya dimensi lain.



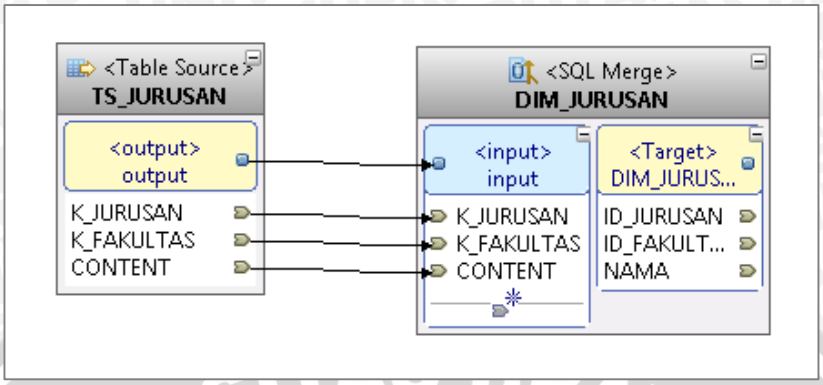
Gambar 4.8 Pemetaan Data Dimensi Fakultas

Seperti yang terlihat pada Gambar 4.8, data pada dimensi fakultas berasal dari tabel fakultas di basis data SIAKAD. Dikarenakan format tabel fakultas dan dimensi fakultas sudah sesuai maka proses transformasi data tidak diperlukan. Sehingga setelah data diekstraksi dari tabel fakultas dapat langsung disimpan di dimensi fakultas

4.5.1.2 Dimensi Jurusan

Sesuai namanya dimensi jurusan merupakan dimensi yang berisi detail informasi jurusan asal mahasiswa. Dimensi ini merupakan dimensi di bawah

dimensi fakultas. Sehingga keberadaannya terikat erat dengan dimensi fakultas. Berikut adalah pemetaan dimensi jurusan:

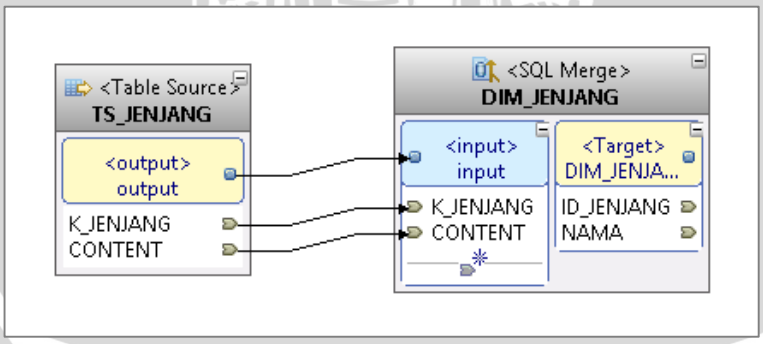


Gambar 4.9 Pemetaan Data Dimensi Jurusan

Seperti yang terlihat pada Gambar 4.9, data pada dimensi jurusan berasal dari tabel jurusan di basis data SIAKAD. Dikarenakan format tabel jurusan dan dimensi jurusan sudah sesuai maka proses transformasi data tidak diperlukan. Sehingga setelah data diekstraksi dari tabel jurusan dapat langsung disimpan di dimensi jurusan. Yang perlu diperhatikan adalah karena dimensi jurusan bergantung pada dimensi fakultas maka sebelum melakukan proses ETL pada dimensi jurusan diharuskan melakukan proses ETL pada dimensi fakultas terlebih dahulu untuk menjaga integritas data.

4.5.1.3 Dimensi Jenjang

Sesuai namanya dimensi jenjang merupakan dimensi yang berisi detail informasi tingkat jenjang studi mahasiswa. Dimensi ini dapat dikatakan berdiri sendiri tanpa bergantung adanya dimensi lain:

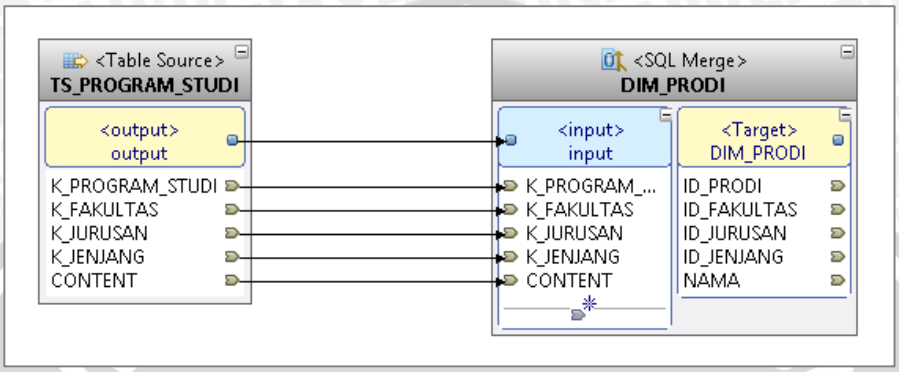


Gambar 4.10 Pemetaan Data Dimensi Jenjang

Seperti yang terlihat pada Gambar 4.10, data pada dimensi jenjang berasal dari tabel jenjang di basis data SIAKAD. Dikarenakan format tabel jenjang dan dimensi jenjang sudah sesuai maka proses transformasi data tidak diperlukan. Sehingga setelah data diekstraksi dari tabel jenjang dapat langsung disimpan di dimensi jenjang.

4.5.1.4 Dimensi Program Studi

Sesuai namanya dimensi program studi merupakan dimensi yang berisi detail informasi program studi mahasiswa. Dimensi ini merupakan dimensi di bawah dimensi fakultas, jurusan dan jenjang. Sehingga keberadaannya terikat erat dengan ketiga dimensi tersebut. Berikut adalah pemetaan dimensi program studi:

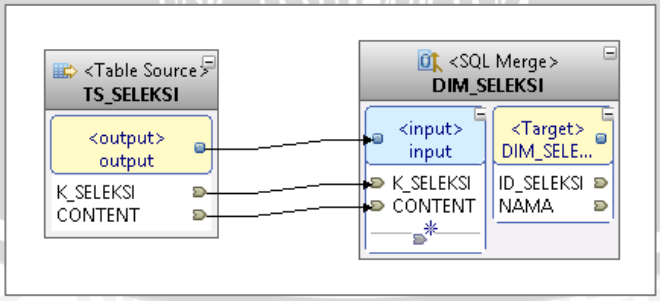


Gambar 4.11 Pemetaan Data Dimensi Program Studi

Seperti yang terlihat pada Gambar 4.11, data pada dimensi program studi berasal dari tabel program studi di basis data SIAKAD. Dikarenakan format tabel program studi dan dimensi program studi sudah sesuai maka proses transformasi data tidak diperlukan. Namun yang perlu diperhatikan adalah karena dimensi program studi tergantung pada dimensi jenjang, jurusan dan fakultas maka sebelum melakukan proses ETL untuk dimensi jenjang maka diharuskan melakukan proses ETL di tiga dimensi tersebut terlebih dahulu untuk menjaga integritas data.

4.5.1.5 Dimensi Seleksi

Sesuai namanya dimensi seleksi adalah dimensi untuk menyimpan detail proses seleksi masuk mahasiswa. Dimensi ini dapat dikatakan berdiri sendiri tanpa bergantung adanya dimensi lain. Berikut adalah pemetaan data dimensi seleksi:

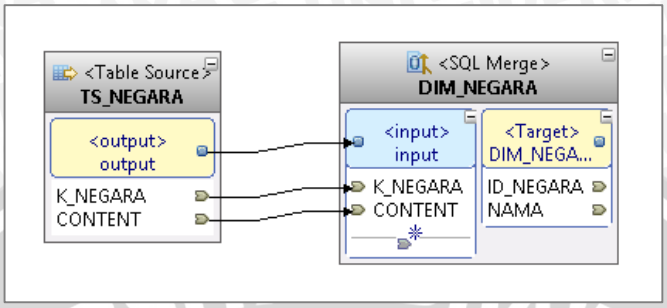


Gambar 4.12 Pemetaan Data Dimensi Seleksi

Seperti yang terlihat pada Gambar 4.12, data pada dimensi seleksi berasal dari tabel seleksi di basis data SIAKAD. Dikarenakan format tabel seleksi dan dimensi seleksi sudah sesuai maka proses transformasi data tidak diperlukan. Sehingga setelah data diambil dari tabel seleksi dapat langsung disimpan di dimensi fakultas.

4.5.1.6 Dimensi Negara

Sesuai namanya dimensi negara adalah dimensi untuk menyimpan detail negara asal mahasiswa. Dimensi ini dapat dikatakan berdiri sendiri tanpa bergantung adanya dimensi lain. Berikut adalah pemetaan data dimensi negara:

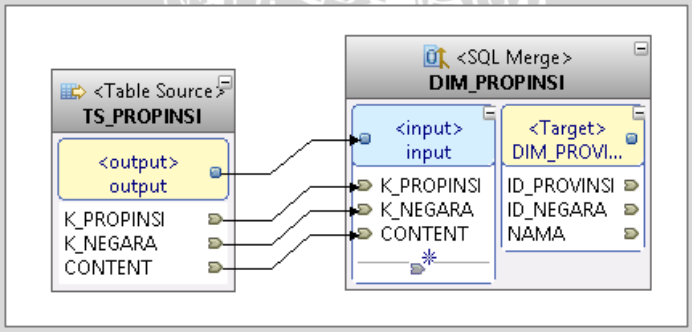


Gambar 4.13 Pemetaan Data Dimensi Negara

Seperti yang terlihat pada Gambar 4.13, data pada dimensi negara berasal dari tabel negara di basis data SIAKAD. Dikarenakan format tabel negara dan dimensi negara sudah sesuai maka proses transformasi data tidak diperlukan. Sehingga setelah data diekstraksi dari tabel negara dapat langsung disimpan di dimensi negara.

4.5.1.7 Dimensi Propinsi

Sesuai namanya dimensi propinsi merupakan dimensi yang berisi detail propinsi asal mahasiswa. Dimensi ini merupakan dimensi di bawah dimensi negara. Sehingga keberadaannya terikat erat dengan dimensi negara. Berikut adalah pemetaan dimensi propinsi

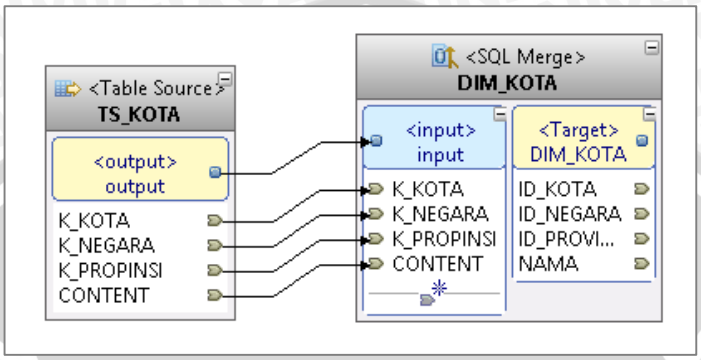


Gambar 4.14 Pemetaan Data Dimensi Propinsi

Seperti yang terlihat pada Gambar 4.14, data pada dimensi propinsi berasal dari tabel jenjang di basis data SIAKAD. Dikarenakan format tabel propinsi dan dimensi propinsi sudah sesuai maka proses transformasi data tidak diperlukan. Sehingga setelah data diekstraksi dari tabel propinsi dapat langsung disimpan di dimensi propinsi. Yang perlu diperhatikan adalah karena dimensi propinsi bergantung pada dimensi negara maka sebelum melakukan proses ETL pada dimensi propinsi diharuskan melakukan proses ETL pada dimensi negara terlebih dahulu untuk menjaga integritas data.

4.5.1.8 Dimensi Kota

Sesuai namanya dimensi kota merupakan dimensi yang berisi detail informasi kota asal mahasiswa. Dimensi ini merupakan dimensi di bawah dimensi negara, dan propinsi. Sehingga keberadaannya terikat erat dengan kedua dimensi tersebut. Berikut adalah pemetaan dimensi kota:

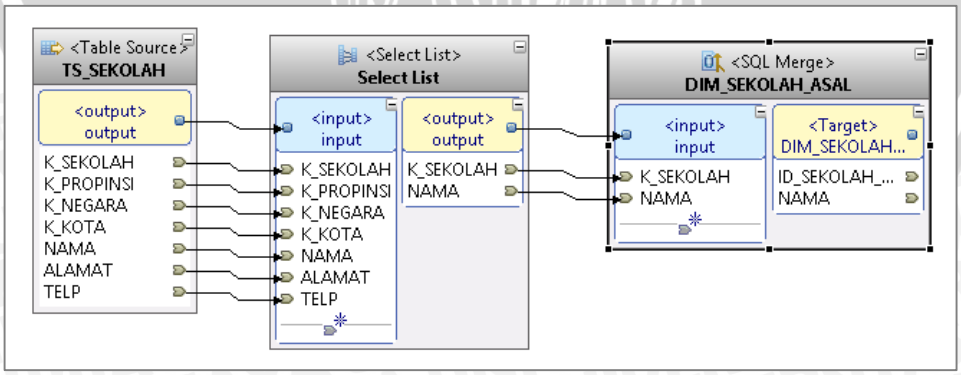


Gambar 4.15 Pemetaan Data Dimensi Kota

Seperti yang terlihat pada Gambar 4.15, data pada dimensi kota berasal dari tabel kota di basis data SIAKAD. Dikarenakan format tabel kota dan dimensi kota sudah sesuai maka proses transformasi data tidak diperlukan. Namun yang perlu diperhatikan adalah karena dimensi kota tergantung pada dimensi negara dan propinsi maka sebelum melakukan proses ETL untuk dimensi kota maka diharuskan melakukan proses ETL di dua dimensi tersebut terlebih dahulu untuk menjaga integritas data.

4.5.1.9 Dimensi Sekolah Asal

Sesuai namanya dimensi asal sekolah adalah dimensi untuk menyimpan detail sekolah menengah atas asal mahasiswa. Dimensi ini dapat dikatakan berdiri sendiri tanpa bergantung adanya dimensi lain. Berikut adalah pemetaan data dimensi asal sekolah:



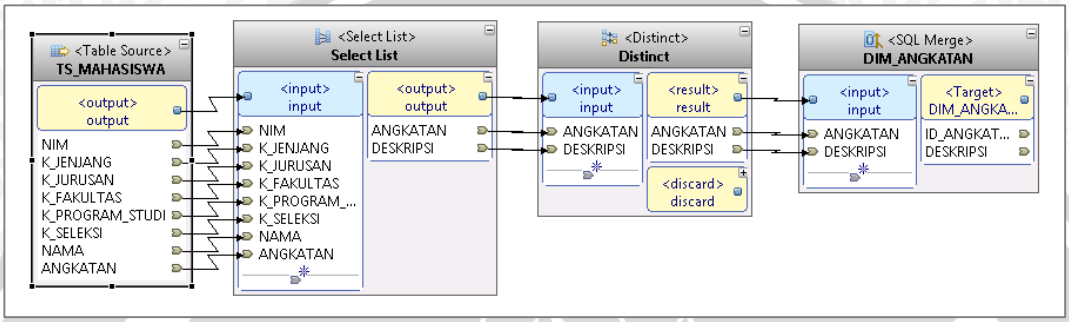
Gambar 4.16 Pemetaan Data Dimensi Sekolah Asal

Seperti yang terlihat pada Gambar 4.16, data pada dimensi asal sekolah berasal dari tabel sekolah di basis data SIAKAD. Dikarenakan ada perbedaan antara kebutuhan dimensi asal sekolah dan kondisi tabel sekolah, maka diperlukan proses

seleksi kolom terlebih dahulu pada tabel sekolah. Adapun kolom yang diseleksi adalah kolom K_SEKOLAH dan NAMA. Setelah itu data dapat disimpan pada dimensi sekolah asal.

4.5.1.10 Dimensi Angkatan

Sesuai namanya dimensi angkatan adalah dimensi untuk menyimpan detail tingkat angkatan mahasiswa. Dimensi ini dapat dikatakan berdiri sendiri tanpa bergantung adanya dimensi lain. Berikut adalah pemetaan data dimensi angkatan:



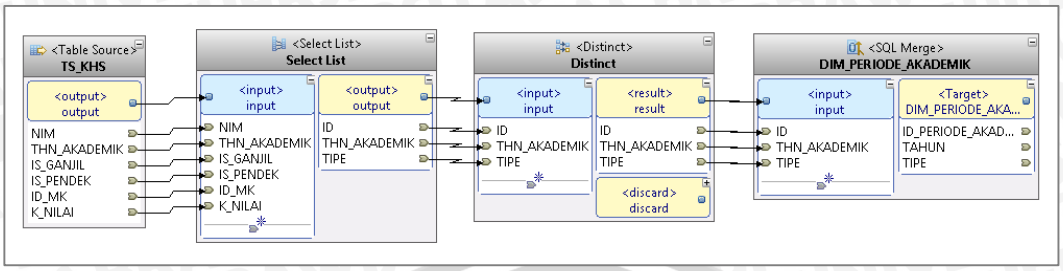
Gambar 4.17 Pemetaan Data Dimensi Angkatan

Seperti yang terlihat pada Gambar 4.17, data pada dimensi angkatan berasal dari tabel mahasiswa di basis data SIAKAD. Dikarenakan format tabel mahasiswa dan dimensi angkatan masih belum sesuai maka diperlukan proses transformasi untuk menyesuaikannya. Adapun proses transformasi yang dilakukan adalah:

1. Data dari tabel mahasiswa dipilih 2 kolom yaitu kolom angkatan dan kolom deskripsi. Kolom deskripsi adalah kolom yang dibuat untuk memenuhi kebutuhan dimensi angkatan dengan cara menggabungkan kata 'Brawijaya' dengan kolom angkatan.
2. Setelah data didapatkan dilakukanlah proses *distinct* untuk menghilangkan data-data kembar.
3. Setelah itu data siap untuk disimpan di dimensi angkatan.

4.5.1.11 Dimensi Periode Akademik

Sesuai namanya dimensi periode akademik adalah dimensi untuk menyimpan detail informasi periode akademik yang digunakan Universitas Brawijaya. Dimensi ini dapat dikatakan berdiri sendiri tanpa bergantung adanya dimensi lain. Berikut adalah pemetaan data dimensi periode akademik:



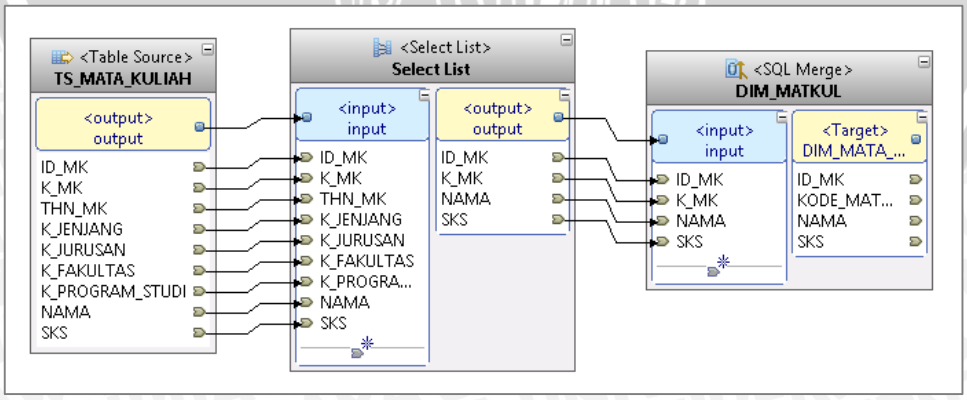
Gambar 4.18 Pemetaan Data Dimensi Periode Akademik

Seperti yang terlihat pada Gambar 4.18, data pada dimensi periode akademik berasal dari tabel KHS di basis data SIAKAD. Dikarenakan format tabel KHS dan dimensi periode akademik masih belum sesuai maka diperlukan proses transformasi untuk menyesuaikannya. Adapun proses transformasi yang dilakukan adalah:

1. Data dari tabel KHS dipilih 3 kolom yaitu kolom ID, THN_AKADEMIK dan TIPE. Kolom ID adalah kolom yang didapat dengan menggabungkan kolom THN_AKADEMIK, IS_GANJIL dan IS_PENDEK. Sementara kolom TIPE berasal dari pengecekan jika KOLOM IS_PENDEK bernilai '1' maka berisi 'PENDEK' jika KOLOM IS_GANJIL benilai '1' maka akan berisi 'GANJIL' selain itu Kolom tipe akan berisi 'GENAP'
2. Setelah data didapatkan dilakukanlah proses *distinct* untuk menghilangkan data-data kembar.
3. Setelah itu data siap untuk disimpan di dimensi periode akademik.

4.5.1.12 Dimensi Mata Kuliah

Sesuai namanya dimensi mata kuliah adalah dimensi untuk menyimpan detail informasi mata kuliah yang dimiliki Universitas Brawijaya. Dimensi ini dapat dikatakan berdiri sendiri tanpa bergantung adanya dimensi lain. Berikut adalah pemetaan data dimensi mata kuliah:



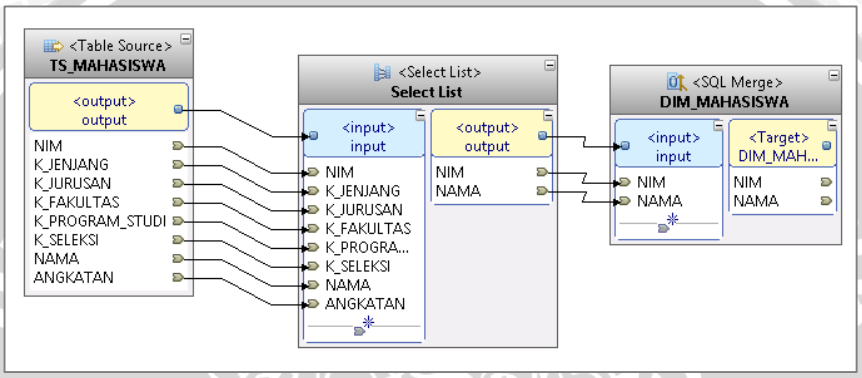
Gambar 4.19 Pemetaan Data Dimensi Mata Kuliah

Seperti yang terlihat pada Gambar 4.19, data pada dimensi mata kuliah berasal dari tabel mata kuliah di basis data SIAKAD. Dikarenakan ada perbedaan antara

kebutuhan dimensi mata kuliah dan kondisi tabel mata kuliah, maka diperlukan proses seleksi kolom terlebih dahulu pada tabel mata kuliah. Adapun kolom yang diseleksi adalah kolom ID_MK, K_MK, SKS dan NAMA. Setelah itu data dapat disimpan pada dimensi mata kuliah.

4.5.1.13 Dimensi Mahasiswa

Sesuai namanya dimensi mahasiswa adalah dimensi untuk menyimpan detail informasi mahasiswa Universitas Brawijaya. Dimensi ini dapat dikatakan berdiri sendiri tanpa bergantung adanya dimensi lain. Berikut adalah pemetaan data dimensi mahasiswa:

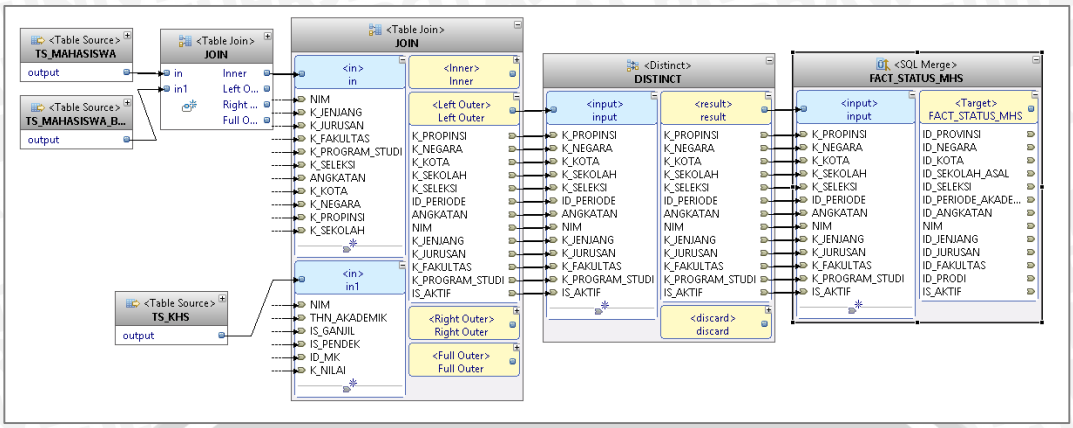


Gambar 4.20 Pemetaan Data Dimensi Mahasiswa

Seperti yang terlihat pada Gambar 4.20, data pada dimensi mahasiswa berasal dari tabel mahasiswa di basis data SIAKAD. Dikarenakan ada perbedaan antara kebutuhan dimensi mahasiswa dan kondisi tabel mahasiswa, maka diperlukan proses seleksi kolom terlebih dahulu pada tabel mahasiswa. Adapun kolom yang diseleksi adalah kolom NIM dan NAMA. Setelah itu data dapat disimpan pada dimensi mata kuliah.

4.5.1.14 Fakta Entri KHS

Sesuai pembahasan bab 3.5.4 mengenai pemilihan fakta. Fakta status mahasiswa akan merekam persebaran mahasiswa berdasarkan statusnya di Universitas Brawijaya. Tabel fakta ini membutuhkan beberapa dimensi untuk mendukung fungsinya. Dimensi-dimensi tersebut antara lain dimensi fakultas, dimensi jurusan, dimensi jenjang, dimensi program studi, dimensi seleksi, dimensi negara, dimensi propinsi, dimensi kota, dimensi asal sekolah, dimensi angkatan, dimensi periode akademik dan dimensi mahasiswa. Adapun pemetaan datanya adalah sebagai berikut:



Gambar 4.21 Pemetaan Data Fakta Status Mahasiswa

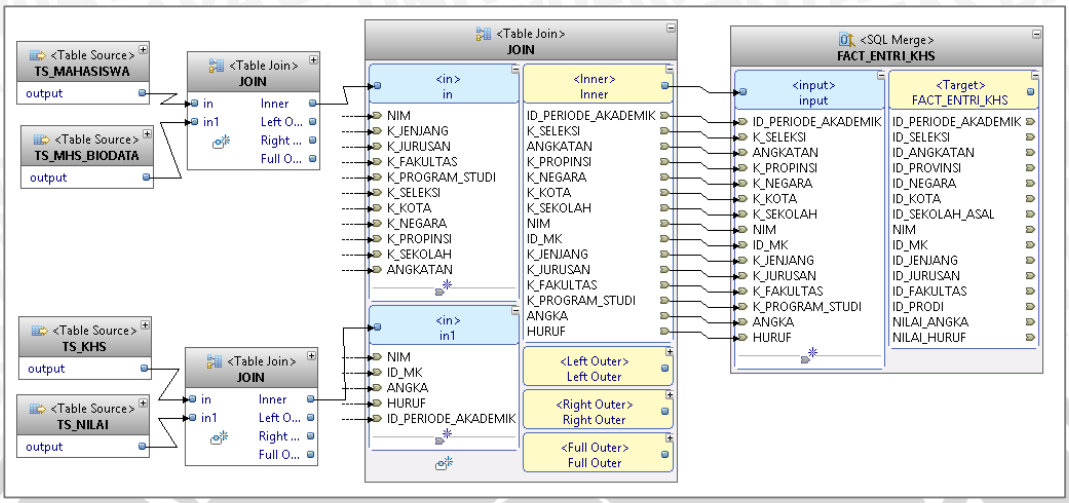
Seperti yang terlihat pada Gambar 4.21 bahwa sumber data fakta status mahasiswa adalah tabel mahasiswa, tabel mahasiswa biodata dan tabel KHS di basis data SIAKAD. Untuk dapat disimpan di fakta status mahasiswa maka data-data tersebut harus melalui serangkaian proses transformasi terlebih dahulu adapun proses transformasinya adalah:

1. Dilakukannya proses *inner join* pada tabel MAHASISWA dan tabel MAHASISWA BIODATA kemudian dilakukan seleksi kolom yaitu kolom NIM, K_JENJANG, K_JURUSAN, K_FAKUTAS, K_PROGRAM_ STUDI, K_SELEKSI, ANGKATAN, K_KOTA, K_NEGARA, K_PROPINSI dan K_SEKOLAH. Setelah itu hasil *join* dan seleksi disimpan pada *result set 1*.
2. Kemudian *Result Set 1* di *left join*-kan dengan tabel KHS kemudian dilakukan proses seleksi kolom yaitu kolom NIM, K_JENJANG, K_JURUSAN, K_FAKUTAS, K_PROGRAM_STUDDI, K_SELEKSI, ANGKATAN, K_KOTA, K_NEGARA, K_PROPINSI, K_SEKOLAH, ID_PERIODE_AKADEMIK dan IS_AKTIF. Kolom ID_PERIODE_ AKADEMIK didapat dari penggabungan kolom THN_AKADEMIK, IS_PENDEK dan IS_GANJIL. Sementara kolom IS_AKTIF didapat dari pengecekan adakah *record* KHS mahasiswa yang bersangkutan pada periode akademik tersebut jika ada maka kolom IS_AKTIF akan bernilai 1 dan sebaliknya akan 0.
3. Setelah itu data siap untuk disimpan di fakta.

4.5.1.15 Fakta Status Mahasiswa

Sesuai pembahasan bab 3.5.4 mengenai pemilihan fakta. Fakta entri kartu hasil studi akan merekam persebaran prestasi akademik mahasiswa di Universitas Brawijaya. Tabel fakta ini membutuhkan beberapa dimensi untuk mendukung fungsinya. Dimensi-dimensi tersebut antara lain dimensi fakultas, dimensi jurusan, dimensi jenjang, dimensi program studi, dimensi seleksi, dimensi negara, dimensi propinsi, dimensi kota, dimensi asal sekolah, dimensi angkatan, dimensi periode

akademik, dimensi mata kuliah dan dimensi mahasiswa. Adapun pemetaan datanya adalah sebagai berikut:



Gambar 4.22 Pemetaan Data Fact Entri KHS

Seperti yang terlihat pada Gambar 4.22 bahwa sumber data fakta entri KHS adalah tabel mahasiswa, tabel mahasiswa biodata, tabel nilai dan tabel KHS di basis data SIAKAD. Untuk dapat disimpan di fakta entri KHS maka data-data tersebut harus melalui serangkaian proses transformasi terlebih dahulu adapun proses transformasinya adalah:

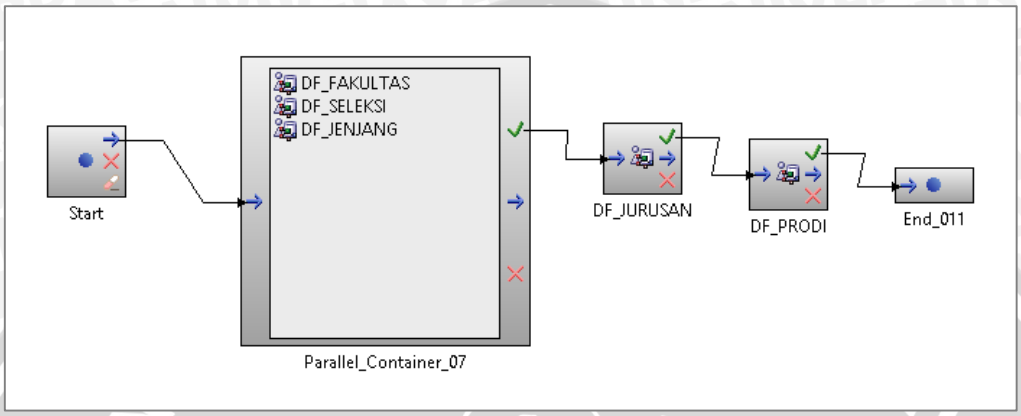
1. Dilakukannya proses *inner join* pada tabel MAHASISWA dan tabel MAHASISWA BIODATA kemudian dilakukan seleksi kolom yaitu kolom NIM, K_JENJANG, K_JURUSAN, K_FAKUTAS, K_PROGRAM_ STUDI, K_SELEKSI, ANGKATAN, K_KOTA, K_NEGARA, K_PROPINSI dan K_SEKOLAH. Setelah itu hasil *join* dan seleksi disimpan pada *result set 1*.
2. Dilakukannya proses *inner join* pada tabel KHS dan tabel NILAI kemudian dilakukan seleksi kolom yaitu kolom NIM, ID_MK, ANGKA, HURUF dan ID_PERIODE AKADEMIK. Kolom ID_PERIODE_ AKADEMIK didapat dari penggabungan kolom THN_AKADEMIK, IS_PENDEK dan IS_GANJIL. Setelah itu hasil *join* dan seleksi disimpan pada *result set 2*.
3. Kemudian dilakukan proses join antara *result set 1* dan *result set 2*.
4. Setelah itu data siap untuk disimpan di fakta.

4.5.2 Pembuatan Alur Kontrol

Alur kontrol dibuat untuk melakukan kontrol terhadap alur data yang telah dibuat. Mulai dari menentukan urutan eksekusi, waktu eksekusi sampai penanganan jika eksekusi gagal atau berhasil dilakukan. Pada penelitian ini ada 4 alur kontrol yang dibuat. Yaitu alur kontrol universitas, alur kontrol daerah, alur kontrol mahasiswa dan alur kontrol fakta.

4.5.2.1 Alur Kontrol Universitas

Alur kontrol Universitas adalah alur kontrol yang mengatur tentang alur data yang berhubungan dengan universitas dan divisinya. Alur kontrol ini mengatur alur data dimensi fakultas, dimensi jenjang, dimensi jurusan, dimensi program studi dan dimensi seleksi.

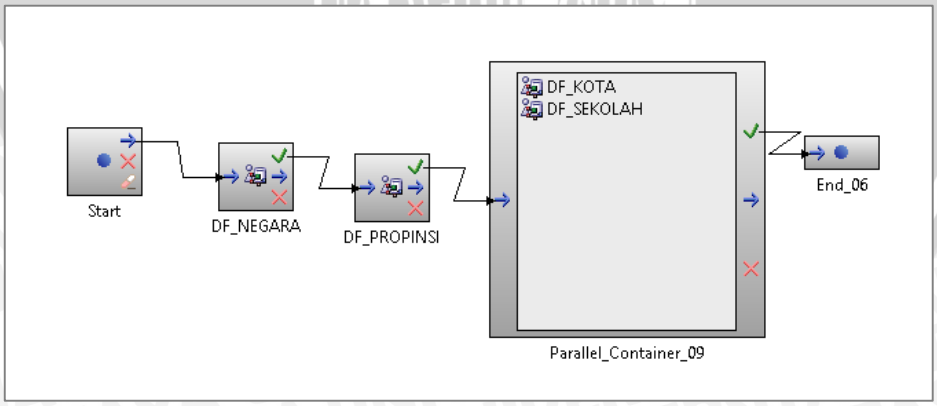


Gambar 4.23 Alur Kontrol Universitas

Seperti yang ditunjukkan Gambar 4.23, alur kontrol ini dimulai dengan mengeksekusi alur data fakultas, alur data seleksi dan alur data jenjang. Hal ini dapat dilakukan karena ke-tiga alur data tersebut merupakan alur data yang mandiri tidak memerlukan alur data yang lain. Setelah itu baru mengeksekusi alur data jurusan dan akhirnya mengeksekusi alur data program studi.

4.5.2.2 Alur Kontrol Daerah

Alur kontrol daerah mengatur alur data yang berhubungan dengan daerah asal mahasiswa seperti alur data negara, alur data propinsi, alur data kota dan alur data sekolah asal.

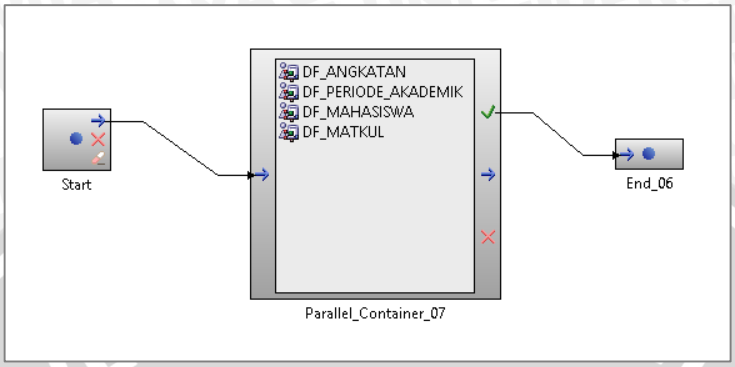


Gambar 4.24 Alur Kontrol Daerah

Seperti yang ditunjukkan Gambar 4.24, alur kontrol ini dimulai dengan mengeksekusi alur data negara dilanjutkan dengan alur data propinsi dan terakhir alur data kota dan alur data sekolah secara bersama-sama.

4.5.2.3 Alur Kontrol Mahasiswa

Alur kontrol mahasiswa mengatur alur data yang berhubungan dengan dimensi mahasiswa yaitu alur data mahasiswa, alur data angkatan, alur data periode akademik dan alur data mata kuliah.

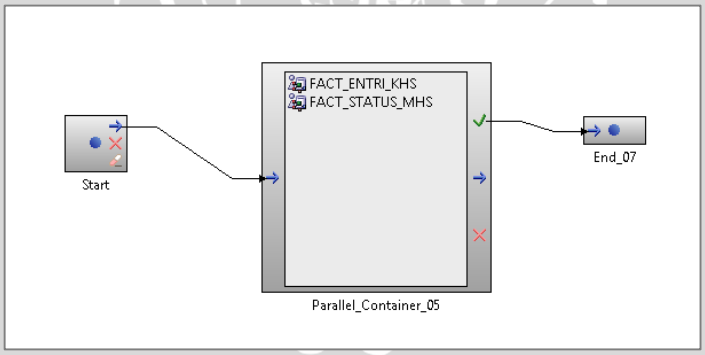


Gambar 4.25 Alur Kontrol Mahasiswa

Seperti yang ditunjukkan oleh Gambar 4.25, alur kontrol ini langsung mengeksekusi ke-4 alur data secara bersamaan. Hal ini dilakukan karena tidak adanya ketergantungan alur data satu dengan yang lainnya.

4.5.2.4 Alur Kontrol Fakta

Alur kontrol fakta mengatur alur data fakta yang ada pada penelitian ini yaitu alur data fakta entri khas dan alur data fakta status mahasiswa.



Gambar 4.26 Alur Kontrol Fakta

Seperti yang ditunjukkan oleh Gambar 4.26, alur kontrol ini langsung mengeksekusi ke-2 alur data secara bersamaan. Hal ini dilakukan karena tidak adanya ketergantungan alur data satu dengan yang lainnya.

4.5.3 Pemasangan dan Penjadwalan

Pemasangan merupakan proses penerapan alur data dan alur kontrol yang telah dibuat ke *server*. Sementara penjadwalan adalah proses pengaturan kapan waktu alur kontrol dijalankan. Adapun langkah-langkah pemasangan sampai penjadwalan adalah sebagai berikut:

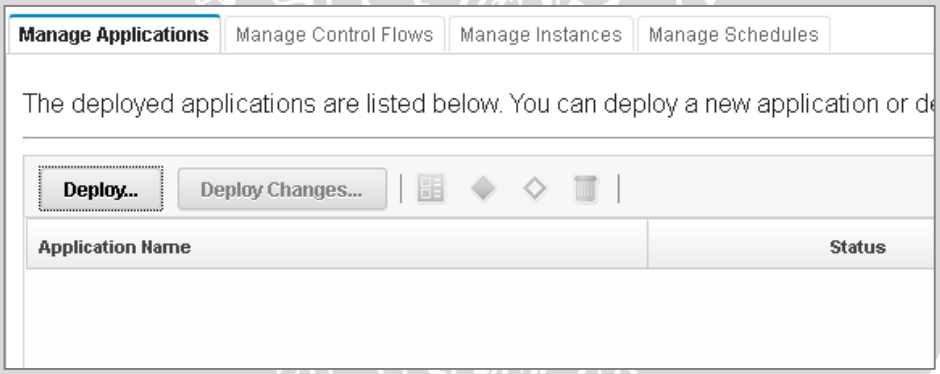
1. Masuk ke halaman *IBM Websphere*

Gambar 4.27 Halaman Login IBM Websphere

Sesuai Gambar 4.27 yang pertama harus dilakukan adalah masuk ke halaman *IBM Websphere* dengan memasukkan *username* dan *password*.

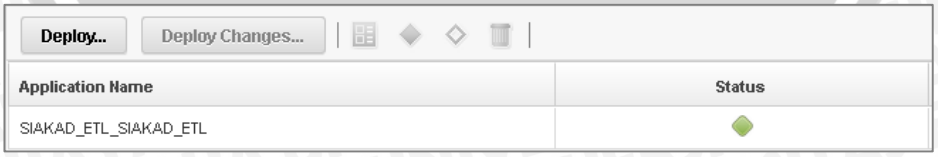
2. *Upload* Alur Data dan Alur Kontrol

Setelah masuk tekan tombol *deploy* seperti yang ditunjukkan Gambar 4.28



Gambar 4.28 Proses Upload

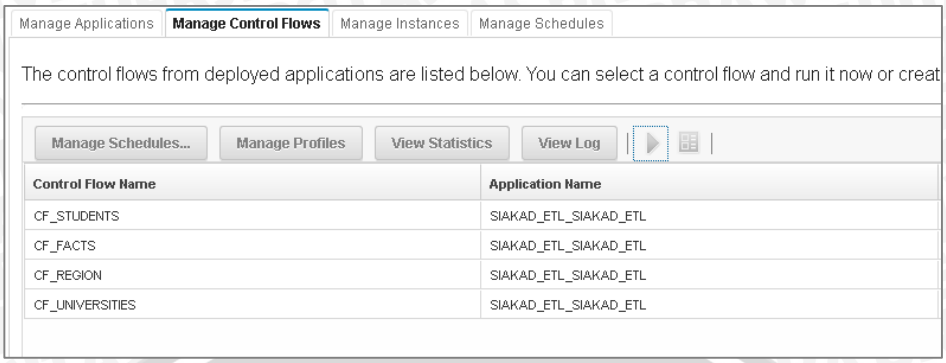
Setelah itu ikut langkah-langkahnya hingga hasil akhirnya aplikasi ETL siap dijalankan seperti yang ditunjukkan Gambar 4.29



Gambar 4.29 Aplikasi Siap

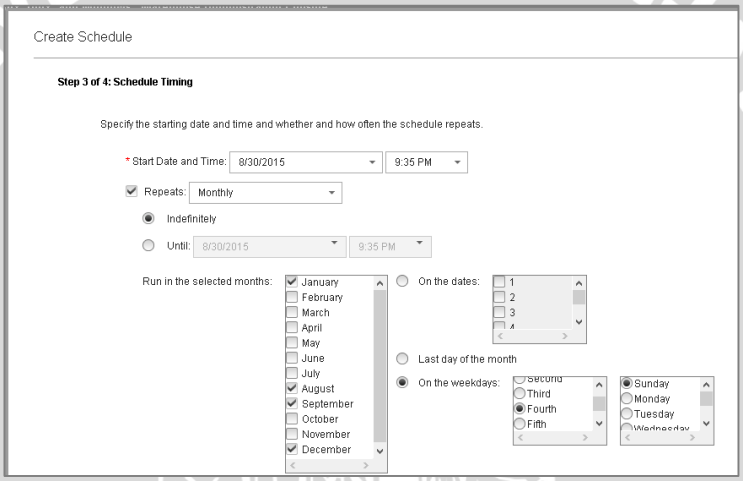
3. Konfigurasi Penjadwalan

Konfigurasi penjadwalan dilakukan dengan mengklik tombol *manage schedule* pada menu *manage control flow*. Seperti yang ditunjukkan Gambar 4.30



Gambar 4.30 Kelola Penjadwalan

Setelah itu pilih jadwal yang diinginkan. Pada penelitian ini jadwal proses ETL yang dipilih adalah setiap hari Minggu bulan Desember, Januari, Juli dan Agustus seperti yang ditunjukkan Gambar 4.31.



Gambar 4.31 Pemilihan Jadwal

Hari Minggu dipilih karena merupakan hari ketika kuliah libur dan aktivitas pengubahan data sedang minim sehingga kinerja sistem tidak akan terganggu. Sementara bulan Desember, Juli, Agustus dan Desember dipilih karena perubahan data hanya mungkin dilakukan pada bulan-bulan tersebut.

4.6 Materialized Query Table

Pada dasarnya *materialized query table* adalah tabel yang menyimpan hasil *query* tertentu yang dalam kasus penelitian ini adalah *reporting query*. Implementasi *materialized query table* dilakukan dengan menerjemahkan rancangan aljabar relasional yang telah dibuat pada bab 4.6 menjadi DDL (*Data Definition Language*) dan di *deploy* di *data mart server*.

4.6.1 MQT Jumlah Mahasiswa Aktif

Materialized Query Table ini akan menyimpan hasil *query* yang menghitung sebaran jumlah mahasiswa aktif per fakultas, jurusan, jenjang, program studi,

seleksi, angkatan dan periode akademik. Adapun tabel yang digunakan dalam MQT ini adalah DIM_FAKULTAS, DIM_JURUSAN, DIM_JENJANG, DIM_ PROGRAM_STUDI, DIM_SELEKSI, DIM_ANGKATAN, DIM_PERIODE_ AKADEMIK dan FACT_STATUS_MHS. Ke-8 tabel tersebut akan mengalami proses *join* kemudian dihitung mana mahasiswa yang memiliki status aktif. Sehingga DDL MQT Jumlah Mahasiswa Aktif adalah sebagai berikut:

```
1.  CREATE TABLE MART.SEBARAN_MHS_AKTIF AS (
2.    SELECT
3.      DF.NAMA AS FAKULTAS,
4.      DJ.NAMA AS JURUSAN,
5.      DP.NAMA AS PRODI,
6.      DJE.NAMA AS JENJANG,
7.      DS.NAMA AS SELEKSI,
8.      DA.ID_ANGKATAN AS ANGKATAN,
9.      DPA.TAHUN || ' ' || DPA.TIPE PERIODE_AKADEMIK,
10.     R1.JUMLAH AS JUMLAH
11.   FROM (SELECT
12.           F.ID_FAKULTAS, F.ID_JURUSAN, F.ID_JENJANG,
13.           F.ID_PRODI, F.ID_SELEKSI, F.ID_ANGKATAN,
14.           F.ID_PERIODE_AKADEMIK,
15.           COUNT(F.IS_AKTIF) AS JUMLAH
16.         FROM MART.FACT_STATUS_MHS F
17.         WHERE F.IS_AKTIF = 1
18.         GROUP BY
19.           F.ID_FAKULTAS, F.ID_JURUSAN, F.ID_JENJANG,
20.           F.ID_PRODI, F.ID_SELEKSI, F.ID_ANGKATAN,
21.           F.ID_PERIODE_AKADEMIK) R1
22.   INNER JOIN MART.DIM_PRODI DP ON
23.     R1.ID_PRODI = DP.ID_PRODI AND R1.ID_JURUSAN =
DP.ID_JURUSAN AND R1.ID_FAKULTAS = DP.ID_FAKULTAS
24.   INNER JOIN MART.DIM_JURUSAN DJ ON
25.     R1.ID_JURUSAN = DJ.ID_JURUSAN AND R1.ID_FAKULTAS =
DJ.ID_FAKULTAS
26.   INNER JOIN MART.DIM_FAKULTAS DF ON
27.     R1.ID_FAKULTAS = DF.ID_FAKULTAS
28.   INNER JOIN MART.DIM_JENJANG DJE ON
29.     R1.ID_JENJANG = DJE.ID_JENJANG
30.   INNER JOIN MART.DIM_SELEKSI DS ON
31.     R1.ID_SELEKSI = DS.ID_SELEKSI
32.   INNER JOIN MART.DIM_ANGKATAN DA ON
33.     R1.ID_ANGKATAN = DA.ID_ANGKATAN
34.   INNER JOIN MART.DIM_PERIODE_AKADEMIK DPA ON
35.     R1.ID_PERIODE_AKADEMIK = DPA.ID_PERIODE_AKADEMIK
)DATA INITIALLY DEFERRED REFRESH DEFERRED;
36.
37.
38.
```

Berikut adalah penjelasan dari kode DDL diatas :

Baris 1 : Adalah perintah untuk membuat tabel dengan nama SEBARAN_MHS_AKTIF

Baris 2-10 : Perintah untuk memilih kolom-kolom yang nantinya akan menjadi atribut tabel SEBARAN_MHS_AKTIF.

Baris 11-21 : Perintah untuk melakukan *subquery* yang akan melakukan penghitungan agregasi jumlah mahasiswa aktif dan *grouping* pada kolom ID_FAKULTAS, ID_JURUSAN, ID_JENJANG, ID_PRODI, ID_SELEKSI, ID_ANGKATAN dan ID_ PERIODE_AKADEMIK pada tabel FACT_STATUS_MHS. Setelah itu melakukan proses *renaming result set* menjadi R1

Baris 22-24 : Adalah perintah untuk melakukan proses *join* antara tabel FACT_STATUS_MHS dan tabel DIM_PRODI dengan syarat kolom ID_PRODI, ID_JURUSAN dan ID_FAKULTAS memiliki nilai yang sama.

Baris 25-27 : Adalah perintah untuk melakukan proses *join* antara tabel FACT_STATUS_MHS dan tabel DIM_JURUSAN dengan syarat kolom ID_JURUSAN dan ID_FAKULTAS memiliki nilai yang sama.

Baris 28-29 : Adalah perintah untuk melakukan proses *join* antara tabel FACT_STATUS_MHS dan tabel DIM_FAKULTAS dengan syarat kolom ID_FAKULTAS memiliki nilai yang sama.

Baris 30-31 : Adalah perintah untuk melakukan proses *join* antara tabel FACT_STATUS_MHS dan tabel DIM_JENJANG dengan syarat kolom ID_JENJANG memiliki nilai yang sama.

Baris 32-33 : Adalah perintah untuk melakukan proses *join* antara tabel FACT_STATUS_MHS dan tabel DIM_SELEKSI dengan syarat kolom ID_SELEKSI memiliki nilai yang sama.

Baris 34-35 : Adalah perintah untuk melakukan proses *join* antara tabel FACT_STATUS_MHS dan tabel DIM_ANGKATAN dengan syarat kolom ANGKATAN memiliki nilai yang sama.

Baris 36-37 : Adalah perintah untuk melakukan proses *join* antara tabel FACT_STATUS_MHS dan tabel DIM_PERIODE_ AKADEMIK dengan syarat kolom ID_PERIODE_ AKADEMIK memiliki nilai yang sama.

Baris 38 : Adalah penutup deklarasi *materialized query table* dengan opsi pengisian data ditangguhkan dan langsung dilakukan penyegaran data.

4.6.2 MQT Rata-rata IPK Mahasiswa

Materialized Query Table ini akan menyimpan hasil *query* yang menghitung sebaran rata-rata IPK mahasiswa per fakultas, jurusan, jenjang, program studi, seleksi, angkatan dan periode akademik. Adapun tabel yang digunakan dalam MQT ini adalah DIM_FAKULTAS, DIM_JURUSAN, DIM_JENJANG, DIM_ PROGRAM_STUDI, DIM_SELEKSI, DIM_ANGKATAN, DIM_PERIODE_ AKADEMIK, DIM_MATA_KULIAH dan FACT_PERKEMBANGAN_ AKADEMIK. Ke-9 tabel tersebut

akan mengalami proses *join* kemudian dihitung rata-rata IPK-nya. Sehingga DDL MQT Rata-rata IPK Mahasiswa adalah sebagai berikut:

```
1.  CREATE TABLE MART.RATA_IPK_MHS AS (
2.    SELECT
3.      DF.NAMA FAKULTAS,
4.      DJ.NAMA JURUSAN,
5.      DP.NAMA PRODI,
6.      DJEN.NAMA JENJANG,
7.      DS.NAMA SELEKSI,
8.      DA.ID_ANGKATAN ANGKATAN,
9.      DPA.TAHUN || ' ' || DPA.TIPE PERIODE_AKADEMIK,
10.     R1.RATA_IPK
11.   FROM (SELECT
12.           F.ID_FAKULTAS, F.ID_JURUSAN, F.ID_PRODI,
13.           F.ID_JENJANG, F.ID_SELEKSI, F.ID_ANGKATAN,
14.           F.ID_PERIODE_AKADEMIK,
15.           SUM(F.NILAI_ANGKA*DMK.SKS) / SUM(DMK.SKS) RATA_IPK
16.         FROM MART.FACT_ENTRI_KHS F
17.         INNER JOIN MART.DIM_MATA_KULIAH DMK
18.           ON F.ID_MK = DMK.ID_MK
19.         GROUP BY
20.           F.ID_FAKULTAS, F.ID_JURUSAN, F.ID_PRODI,
21.           F.ID_JENJANG, F.ID_SELEKSI, F.ID_ANGKATAN,
22.           F.ID_PERIODE_AKADEMIK
23.         HAVING SUM(DMK.SKS) > 0) R1
24.   INNER JOIN MART.DIM_PRODI DP ON
25.     R1.ID_PRODI    =   DP.ID_PRODI    AND    R1.ID_JURUSAN    =
26. DP.ID_JURUSAN AND R1.ID_FAKULTAS = DP.ID_FAKULTAS
27.   INNER JOIN MART.DIM_JURUSAN DJ ON
28.     R1.ID_JURUSAN   =   DJ.ID_JURUSAN   AND   R1.ID_FAKULTAS   =
29. DJ.ID_FAKULTAS
30.   INNER JOIN MART.DIM_FAKULTAS DF ON
31.     R1.ID_FAKULTAS = DF.ID_FAKULTAS
32.   INNER JOIN MART.DIM_JENJANG DJEN ON
33.     R1.ID_JENJANG = DJEN.ID_JENJANG
34.   INNER JOIN MART.DIM_SELEKSI DS ON
35.     R1.ID_SELEKSI = DS.ID_SELEKSI
36.   INNER JOIN MART.DIM_ANGKATAN DA ON
37.     R1.ID_ANGKATAN = DA.ID_ANGKATAN
38.   INNER JOIN MART.DIM_PERIODE_AKADEMIK DPA ON
39.     R1.ID_PERIODE_AKADEMIK = DPA.ID_PERIODE_AKADEMIK
40. )DATA INITIALLY DEFERRED REFRESH DEFERRED;
```

Berikut adalah penjelasan dari kode DDL di atas:

Baris 1 : Adalah perintah untuk membuat tabel dengan nama RATA_IPK_MHS.

Baris 2-10 : Perintah untuk memilih kolom-kolom yang nantinya akan menjadi atribut tabel RATA_IPK_MHS.

Baris 11-23 : Perintah untuk melakukan *subquery* yang akan melakukan penghitungan agregasi rata-rata IPK mahasiswa dan *grouping* pada kolom ID_FAKULTAS, ID_JURUSAN, ID_JENJANG, ID_PRODI, ID_SELEKSI, ID_ANGKATAN dan ID_

PERIODE_AKADEMIK pada tabel FACT_STATUS_MHS. Setelah itu melakukan proses *renaming result set* menjadi R1

Baris 24-26 : Adalah perintah untuk melakukan proses *join* antara tabel FACT_ENTRI_KHS dan tabel DIM_PRODI dengan syarat kolom ID_PRODI, ID_JURUSAN dan ID_FAKULTAS memiliki nilai yang sama.

Baris 27-29 : Adalah perintah untuk melakukan proses *join* antara tabel FACT_ENTRI_KHS dan tabel DIM_JURUSAN dengan syarat kolom ID_JURUSAN dan ID_FAKULTAS memiliki nilai yang sama.

Baris 30-31 : Adalah perintah untuk melakukan proses *join* antara tabel FACT_ENTRI_KHS dan tabel DIM_FAKULTAS dengan syarat kolom ID_FAKULTAS memiliki nilai yang sama.

Baris 32-33 : Adalah perintah untuk melakukan proses *join* antara tabel FACT_ENTRI_KHS dan tabel DIM_JENJANG dengan syarat kolom ID_JENJANG memiliki nilai yang sama.

Baris 34-35 : Adalah perintah untuk melakukan proses *join* antara tabel FACT_ENTRI_KHS dan tabel DIM_SELEKSI dengan syarat kolom ID_SELEKSI memiliki nilai yang sama.

Baris 36-37 : Adalah perintah untuk melakukan proses *join* antara tabel FACT_ENTRI_KHS dan tabel DIM_ANGKATAN dengan syarat kolom ANGKATAN memiliki nilai yang sama.

Baris 38-39 : Adalah perintah untuk melakukan proses *join* antara tabel FACT_ENTRI_KHS dan tabel DIM_PERIODE_ AKADEMIK dengan syarat kolom ID_PERIODE_ AKADEMIK memiliki nilai yang sama.

Baris 40 : Adalah penutup deklarasi *materialized query table* dengan opsi pengisian data ditangguhkan dan langsung dilakukan penyegaran data.

4.6.3 MQT Persebaran Mahasiswa Berdasar Asal Sekolah

Materialized Query Table ini akan menyimpan hasil *query* yang menghitung sebaran jumlah mahasiswa aktif per fakultas, jurusan, jenjang, program studi, seleksi, angkatan dan periode akademik. Adapun tabel yang digunakan dalam MQT ini adalah DIM_FAKULTAS, DIM_JURUSAN, DIM_JENJANG, DIM_ PROGRAM_STUDI, DIM_SELEKSI, DIM_ANGKATAN, DIM_PERIODE_ AKADEMIK, DIM_SEKOLAH_ASAL dan FACT_STATUS_MHS. Ke-9 tabel tersebut akan mengalami proses *join* kemudian dihitung jumlah mahasiswanya. Sehingga DDL MQT Jumlah Mahasiswa Aktif adalah sebagai berikut:

```
1.  CREATE TABLE MART.SEBARAN_SEKOLAH_MHS AS (
2.    SELECT
3.      DF.NAMA AS FAKULTAS,
4.      DJ.NAMA AS JURUSAN,
5.      DP.NAMA AS PRODI,
6.      DJE.NAMA AS JENJANG,
7.      DS.NAMA AS SELEKSI,
8.      DA.ID_ANGKATAN AS ANGKATAN,
9.      DSA.NAMA AS SEKOLAH_ASAL,
10.     R1.JUMLAH AS JUMLAH
11.   FROM (SELECT
12.           F.ID_FAKULTAS, F.ID_JURUSAN, F.ID_JENJANG,
13.           F.ID_PRODI, F.ID_SELEKSI, F.ID_ANGKATAN,
14.           F.ID_SEKOLAH_ASAL, COUNT(F.NIM) AS JUMLAH
15.         FROM MART.FACT_STATUS_MHS F
16.         GROUP BY
17.           F.ID_FAKULTAS, F.ID_JURUSAN, F.ID_JENJANG,
18.           F.ID_PRODI, F.ID_SELEKSI, F.ID_ANGKATAN,
19.           F.ID_SEKOLAH_ASAL) R1
20.   INNER JOIN MART.DIM_PRODI DP ON
21.     R1.ID_PRODI   =   DP.ID_PRODI   AND   R1.ID_JURUSAN   =
22. DP.ID_JURUSAN AND R1.ID_FAKULTAS = DP.ID_FAKULTAS
23.   INNER JOIN MART.DIM_JURUSAN DJ ON
24.     R1.ID_JURUSAN   =   DJ.ID_JURUSAN   AND   R1.ID_FAKULTAS   =
25. DJ.ID_FAKULTAS
26.   INNER JOIN MART.DIM_FAKULTAS DF ON
27.     R1.ID_FAKULTAS = DF.ID_FAKULTAS
28.   INNER JOIN MART.DIM_JENJANG DJE ON
29.     R1.ID_JENJANG = DJE.ID_JENJANG
30.   INNER JOIN MART.DIM_SELEKSI DS ON
31.     R1.ID_SELEKSI = DS.ID_SELEKSI
32.   INNER JOIN MART.DIM_ANGKATAN DA ON
33.     R1.ID_ANGKATAN = DA.ID_ANGKATAN
34.   INNER JOIN MART.DIM_SEKOLAH_ASAL DSA ON
35.     R1.ID_SEKOLAH_ASAL = DSA.ID_SEKOLAH_ASAL
36. )DATA INITIALLY DEFERRED REFRESH DEFERRED;
```

Adapun penjelasan dari kode DDL di atas adalah sebagai berikut:

Baris 1 : Adalah perintah untuk membuat tabel dengan nama SEBARAN_SEKOLAH_MHS

Baris 2-10 : Perintah untuk memilih kolom-kolom yang nantinya akan menjadi atribut tabel SEBARAN_SEKOLAH_MHS

Baris 11-19 : Perintah untuk melakukan *subquery* yang akan melakukan penghitungan agregasi jumlah mahasiswa dan *grouping* pada kolom ID_FAKULTAS, ID_JURUSAN, ID_JENJANG, ID_PRODI, ID_SELEKSI, ID_ANGKATAN dan ID_ SEKOLAH_ASAL pada tabel FACT_STATUS_MHS. Setelah itu melakukan proses *renaming result set* menjadi R1

Baris 20-22 : Adalah perintah untuk melakukan proses *join* antara tabel FACT_STATUS_MHS dan tabel DIM_PRODI dengan syarat

kolom ID_PRODI, ID_JURUSAN dan ID_FAKULTAS memiliki nilai yang sama.

Baris 23-25 : Adalah perintah untuk melakukan proses *join* antara tabel FACT_STATUS_MHS dan tabel DIM_JURUSAN dengan syarat kolom ID_JURUSAN dan ID_FAKULTAS memiliki nilai yang sama.

Baris 26-27 : Adalah perintah untuk melakukan proses *join* antara tabel FACT_STATUS_MHS dan tabel DIM_FAKULTAS dengan syarat kolom ID_FAKULTAS memiliki nilai yang sama.

Baris 28-29 : Adalah perintah untuk melakukan proses *join* antara tabel FACT_STATUS_MHS dan tabel DIM_JENJANG dengan syarat kolom ID_JENJANG memiliki nilai yang sama.

Baris 30-31 : Adalah perintah untuk melakukan proses *join* antara tabel FACT_STATUS_MHS dan tabel DIM_SELEKSI dengan syarat kolom ID_SELEKSI memiliki nilai yang sama.

Baris 32-33 : Adalah perintah untuk melakukan proses *join* antara tabel FACT_STATUS_MHS dan tabel DIM_ANGKATAN dengan syarat kolom ANGKATAN memiliki nilai yang sama.

Baris 34-35 : Adalah perintah untuk melakukan proses *join* antara tabel FACT_STATUS_MHS dan tabel DIM_SEKOLAH_ ASAL dengan syarat kolom ID_SEKOLAH_ASAL memiliki nilai yang sama.

Baris 36 : Adalah penutup deklarasi *materialized query table* dengan opsi pengisian data ditangguhkan dan langsung dilakukan penyegaran data.

4.6.4 MQT Persebaran Mahasiswa Berdasar Asal Daerah

Materialized Query Table ini akan menyimpan hasil *query* yang menghitung sebaran jumlah mahasiswa aktif per fakultas, jurusan, jenjang, program studi, seleksi, angkatan dan periode akademik. Adapun tabel yang digunakan dalam MQT ini adalah DIM_FAKULTAS, DIM_JURUSAN, DIM_JENJANG, DIM_ PROGRAM_STUDI, DIM_SELEKSI, DIM_ANGKATAN, DIM_PERIODE_ AKADEMIK, DIM_NEGARA, DIM_PROVINSI, DIM_KOTA dan FACT_ STATUS_MHS. Ke-11 tabel tersebut akan mengalami proses *join* kemudian dihitung jumlah mahasiswanya. Sehingga DDL MQT Jumlah Mahasiswa Aktif adalah sebagai berikut:

```
1.  CREATE TABLE MART.SEBARAN_DAERAH_MHS AS (
2.    SELECT
3.      DF.NAMA FAKULTAS,
4.      DJ.NAMA JURUSAN,
5.      DP.NAMA PRODI,
6.      DJE.NAMA JENJANG,
7.      DS.NAMA SELEKSI,
8.      DA.ID_ANGKATAN ANGKATAN,
9.      DK.NAMA KOTA,
10.     DPR.NAMA PROVINSI,
11.     DN.NAMA NEGARA,
12.     R1.JUMLAH JUMLAH
13.   FROM (SELECT
14.           F.ID_FAKULTAS, F.ID_JURUSAN, F.ID_JENJANG,
15.           F.ID_PRODI, F.ID_SELEKSI, F.ID_ANGKATAN,
16.           F.ID_KOTA, F.ID_PROVINSI, F.ID_NEGARA,
17.           COUNT(F.NIM) JUMLAH
18.         FROM MART.FACT_STATUS_MHS F
19.         GROUP BY
20.           F.ID_FAKULTAS, F.ID_JURUSAN, F.ID_JENJANG,
21.           F.ID_PRODI, F.ID_SELEKSI, F.ID_ANGKATAN,
22.           F.ID_KOTA, F.ID_PROVINSI, F.ID_NEGARA) R1
23.   INNER JOIN MART.DIM_PRODI DP ON
24.     R1.ID_PRODI   =   DP.ID_PRODI   AND   R1.ID_JURUSAN   =
25. DP.ID_JURUSAN AND R1.ID_FAKULTAS = DP.ID_FAKULTAS
26.   INNER JOIN MART.DIM_JURUSAN DJ ON
27.     R1.ID_JURUSAN  =  DJ.ID_JURUSAN  AND  R1.ID_FAKULTAS  =
28. DJ.ID_FAKULTAS
29.   INNER JOIN MART.DIM_FAKULTAS DF ON
30.     R1.ID_FAKULTAS = DF.ID_FAKULTAS
31.   INNER JOIN MART.DIM_JENJANG DJE ON
32.     R1.ID_JENJANG = DJE.ID_JENJANG
33.   INNER JOIN MART.DIM_SELEKSI DS ON
34.     R1.ID_SELEKSI = DS.ID_SELEKSI
35.   INNER JOIN MART.DIM_ANGKATAN DA ON
36.     R1.ID_ANGKATAN = DA.ID_ANGKATAN
37.   INNER JOIN MART.DIM_KOTA DK ON
38.     R1.ID_KOTA   =   DK.ID_KOTA   AND   R1.ID_PROVINSI   =
39. DK.ID_PROVINSI AND R1.ID_NEGARA = DK.ID_NEGARA
40.   INNER JOIN MART.DIM_PROVINSI DPR ON
41.     R1.ID_PROVINSI  =  DPR.ID_PROVINSI  AND  R1.ID_NEGARA  =
42. DPR.ID_NEGARA
43.   INNER JOIN MART.DIM_NEGARA DN ON
44.     R1.ID_NEGARA = DN.ID_NEGARA
45. )DATA INITIALLY DEFERRED REFRESH DEFERRED;
```

Adapun penjelasan dari kode DDL di atas adalah sebagai berikut:

Baris 1 : Adalah perintah untuk membuat tabel dengan nama SEBARAN_DAERAH_MHS

Baris 2-12 : Perintah untuk memilih kolom-kolom yang nantinya akan menjadi atribut tabel SEBARAN_DAERAH_MHS

Baris 13-22 : Perintah untuk melakukan *subquery* yang akan melakukan penghitungan agregasi jumlah mahasiswa aktif dan *grouping*

pada kolom ID_FAKULTAS, ID_JURUSAN, ID_JENJANG, ID_PRODI, ID_SELEKSI, ID_ANGKATAN, ID_NEGARA, ID_PROVINSI dan ID_KOTA pada tabel FACT_STATUS_ MHS. Setelah itu melakukan proses *renaming result set* menjadi R1

Baris 23-25 : Adalah perintah untuk melakukan proses *join* antara tabel FACT_STATUS_MHS dan tabel DIM_PRODI dengan syarat kolom ID_PRODI, ID_JURUSAN dan ID_FAKULTAS memiliki nilai yang sama.

Baris 26-28 : Adalah perintah untuk melakukan proses *join* antara tabel FACT_STATUS_MHS dan tabel DIM_JURUSAN dengan syarat kolom ID_JURUSAN dan ID_FAKULTAS memiliki nilai yang sama.

Baris 29-30 : Adalah perintah untuk melakukan proses *join* antara tabel FACT_STATUS_MHS dan tabel DIM_FAKULTAS dengan syarat kolom ID_FAKULTAS memiliki nilai yang sama.

Baris 31-32 : Adalah perintah untuk melakukan proses *join* antara tabel FACT_STATUS_MHS dan tabel DIM_JENJANG dengan syarat kolom ID_JENJANG memiliki nilai yang sama.

Baris 33-34 : Adalah perintah untuk melakukan proses *join* antara tabel FACT_STATUS_MHS dan tabel DIM_SELEKSI dengan syarat kolom ID_SELEKSI memiliki nilai yang sama.

Baris 35-36 : Adalah perintah untuk melakukan proses *join* antara tabel FACT_STATUS_MHS dan tabel DIM_ANGKATAN dengan syarat kolom ANGKATAN memiliki nilai yang sama.

Baris 37-39 : Adalah perintah untuk melakukan proses *join* antara tabel FACT_STATUS_MHS dan tabel DIM_KOTA dengan syarat kolom ID_KOTA, ID_PROVINSI dan ID_NEGARA memiliki nilai yang sama.

Baris 40-42 : Adalah perintah untuk melakukan proses *join* antara tabel FACT_STATUS_MHS dan tabel DIM_PROVINSI dengan syarat kolom ID_PROVINSI dan ID_NEGARA memiliki nilai yang sama.

Baris 43-44 : Adalah perintah untuk melakukan proses *join* antara tabel FACT_STATUS_MHS dan tabel DIM_NEGARA dengan syarat kolom ID_NEGARA memiliki nilai yang sama.

Baris 45 : Adalah penutup deklarasi *materialized query table* dengan opsi pengisian data ditangguhkan dan langsung dilakukan penyegaran data.

BAB 5 PENGUJIAN

Pada bab ini akan dijelaskan mengenai hasil pengujian dari proses implementasi *materialized query table* pada *data mart* seperti yang penulis angkat. Seperti yang tertulis di bab 3, pengujian yang digunakan terdapat dua macam yaitu pengujian performa (*performance testing*) dan pengujian ketahanan (*stress testing*). Baik pengujian performa dan ketahanan dilakukan dengan membandingkan *query* konvensional dan *materialized query table*.

5.1 Pengujian Performa

Pengujian performa dilakukan dengan bantuan dua indikator yaitu kecepatan eksekusi *query* dan sumber daya yang dibutuhkan *query*. Untuk mengukur kecepatan eksekusi *query* dilakukan dengan cara mengeksekusi tiap *query* sebanyak 10 kali dan membandingkannya. Sementara untuk sumber daya yang digunakan *query* diukur menggunakan bantuan IBM Data Studio *Access Plan Diagram*. Hasil pengukuran akan memiliki satuan *timeron* yang merupakan penghitungan hasil biaya *CPU* dan *IO* yang digunakan oleh DB2.

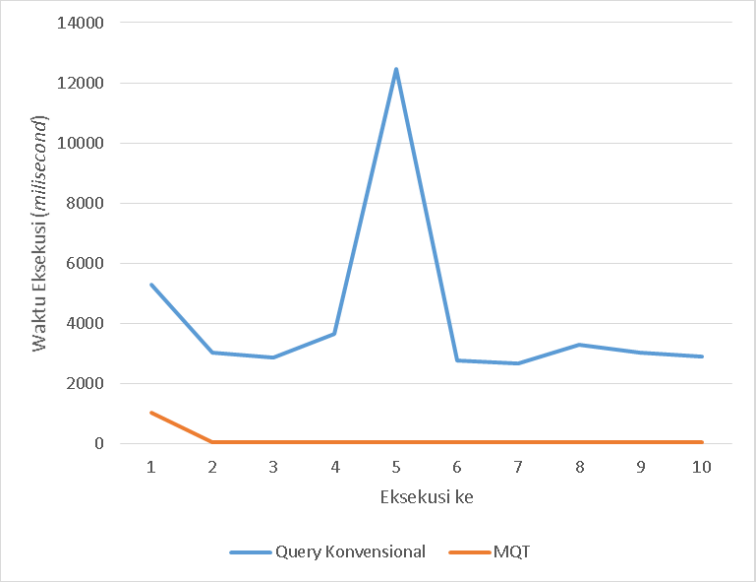
5.1.1 Pengujian *Query* Jumlah Mahasiswa Aktif

Query jumlah mahasiswa aktif merupakan sebuah *query* yang didesain untuk menghitung sebaran jumlah mahasiswa aktif di Universitas Brawijaya. Secara mendasar *query* akan menghitung mahasiswa yang statusnya aktif di setiap fakultas, jurusan, tingkat jenjang, program studi, jalur seleksi, angkatan dan periode akademik. Query ini akan memproses data dari tabel fakta status mahasiswa yang memiliki data sebanyak 769.195 baris. Pengujian *query* dilakukan dengan cara melakukan eksekusi *query* sebanyak 10 kali kemudian di rata-rata. Adapun hasil pengujian yang telah dilakukan dapat dilihat di Tabel 5.1

Tabel 5.1 Hasil Waktu Eksekusi Query Jumlah Mahasiswa Aktif

Jenis *Query*	Percobaa (*milisecond*)										Rata-rata
	1	2	3	4	5	6	7	8	9	10	
Konvensional	5273	3027	2878	3659	12458	2776	2651	3285	3016	2887	4191
MQT	1014	57	44	45	51	47	45	43	45	42	143,3

Pada tabel Tabel 5.1 terlihat jelas hasil waktu eksekusi *query* dengan cara konvensional dan MQT memiliki perbedaan waktu eksekusi yang sangat signifikan. Untuk lebih jelasnya, akan dilakukan perbandingan menggunakan grafik yang ditunjukkan oleh Gambar 5.1.



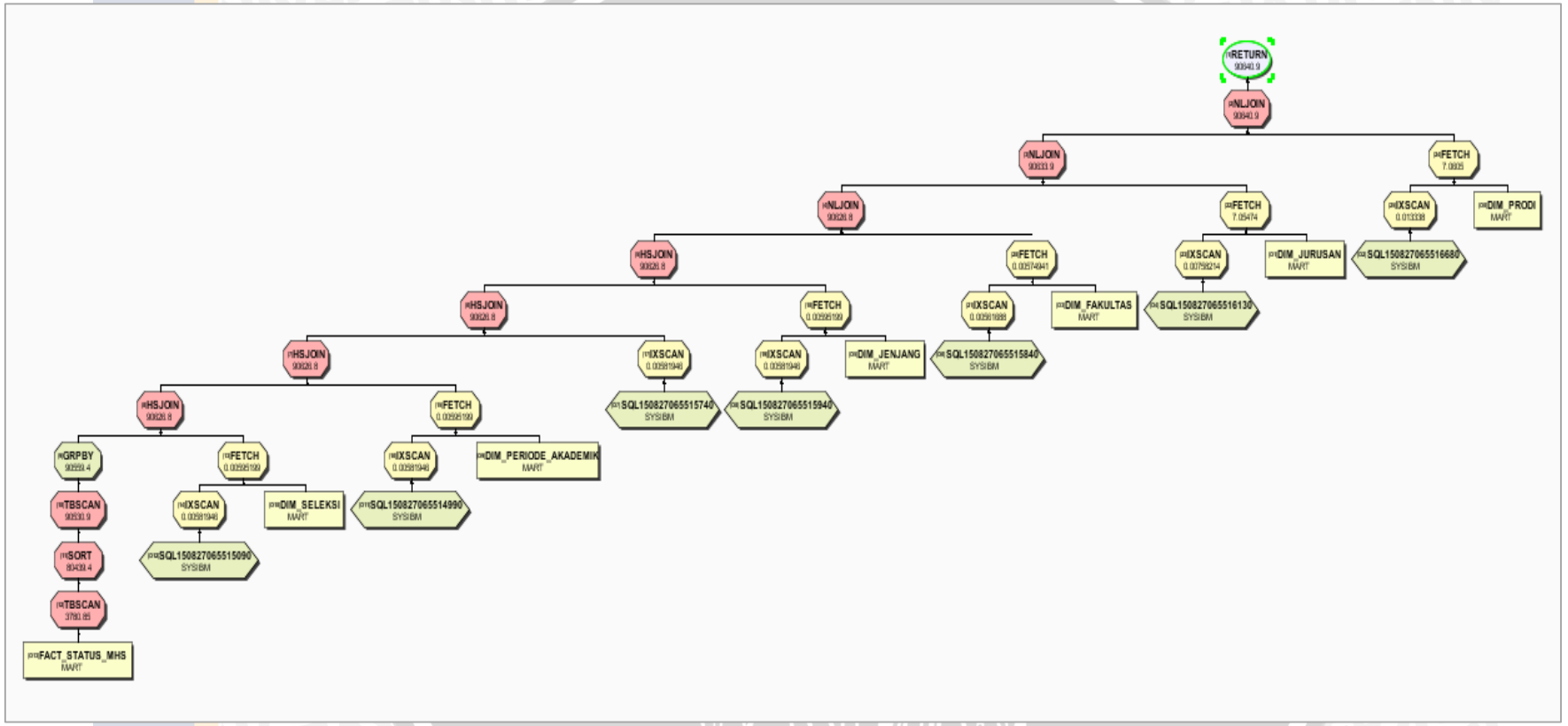
Gambar 5.1 Perbandingan Ekseksusi Query Jumlah MHS Aktif

Seperti yang ditunjukkan Gambar 5.1, maka semakin jelas terlihat perbedaan waktu eksekusi *query* konvensional dan MQT. Setelah menguji waktu eksekusi *query* maka selanjutnya adalah menganalisa sumber daya yang digunakan oleh masing masing *query* yang ditunjukkan Tabel 5.2.

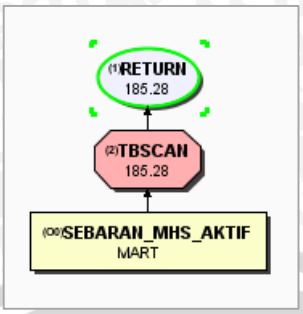
Tabel 5.2 Perbandingan Biaya Sumber Daya Query Jumlah MHS Aktif

Tipe Query	Biaya Sumber Daya (*timerons*)		
	Biaya CPU	Biaya IO	Total Biaya
Konvensional	1.273.480.000.000.000	25339	90.640,9
MQT	1.503.670.000.000	206	185,28

Seperti yang terlihat di Tabel 5.2 bahwa biaya *query* konvensional jauh lebih tinggi dibandingkan biaya menggunakan MQT. Hal ini disebabkan karena *query* konvensional melakukan operasi di banyak tabel sementara MQT hanya melakukan operasi di MQT itu sendiri. Lebih detailnya dapat dianalisa menggunakan model akses diagram yang dapat dilihat pada Gambar 5.2 dan Gambar 5.3.



Gambar 5.2 Diagram Akses Query Jumlah MHS Konvensional



Gambar 5.3 Akses Diagram Query Jumlah MHS MQT

Gambar 5.2 menampilkan akses diagram dari *query* konvensional yang secara jelas menunjukkan banyak sekali tabel digunakan dalam 1 *query* sehingga hal ini tentu akan berdampak besar terhadap biaya sumber daya yang digunakana. Sementara Gambar 5.3 akses diagram dari *materialized query table* yang terlihat sangat sederhana karena hanya menggunakan 1 tabel.

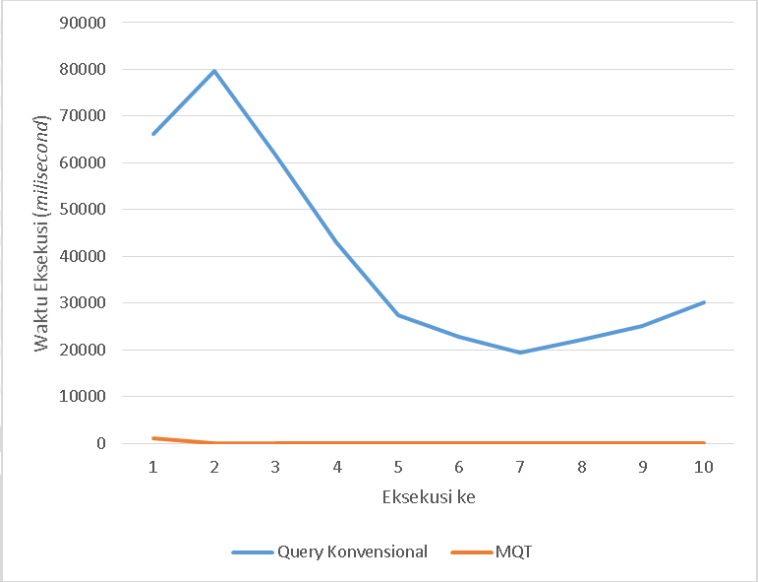
5.1.2 Pengujian *Query* Rata-rata IPK Mahasiswa

Query rata-rata IPK mahasiswa merupakan sebuah *query* yang didesain untuk menghitung rata-rata IPK mahasiswa di Universitas Brawijaya. Secara mendasar *query* akan rata-rata IPK mahasiswa di setiap fakultas, jurusan, tingkat jenjang, program studi, jalur seleksi, angkatan dan periode akademik. Query ini akan memproses data dari tabel fakta entri KHS yang memiliki data sebanyak 2.008.308 baris. Pengujian *query* dilakukan dengan cara melakukan eksekusi *query* sebanyak 10 kali kemudian di rata-rata. Adapun hasil pengujian yang telah dilakukan dapat dilihat di Tabel 5.3

Tabel 5.3 Hasil Waktu Eksekusi Query Rata-rata IPK

Jenis *Query*	Percobaa (*milisecond*)										Rata-rata
	1	2	3	4	5	6	7	8	9	10	
Konvensional	66276	79621	61752	42986	27422	22772	19487	22292	25219	30189	39801,6
MQT	1230	52	52	51	73	62	49	50	51	52	172,2

Pada tabel Tabel 5.3 terlihat jelas hasil waktu eksekusi *query* dengan cara konvensional dan MQT memiliki perbedaan waktu eksekusi yang sangat signifikan. Untuk lebih jelasnya, akan dilakukan perbandingan menggunakan grafik yang ditunjukkan oleh Gambar 5.4.



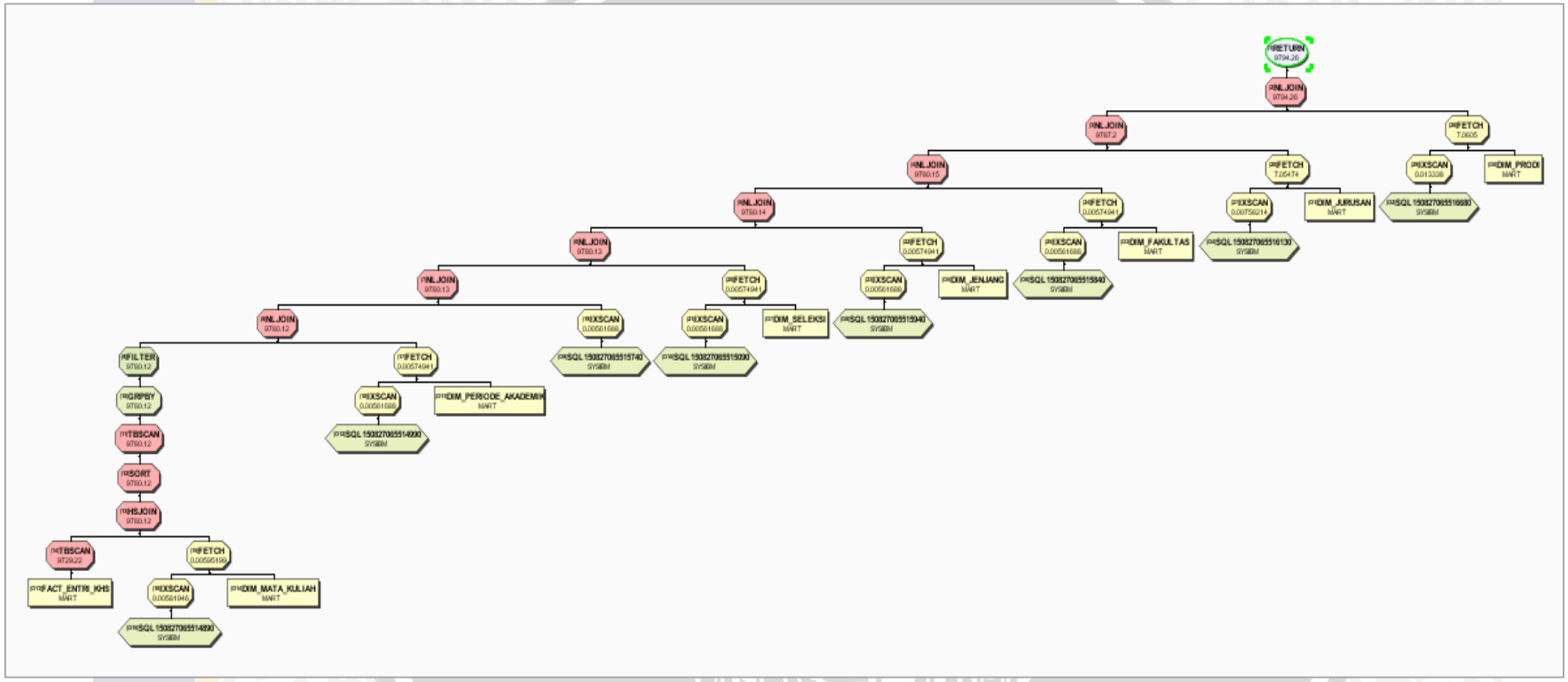
Gambar 5.4 Perbandingan Eksekusi Query Rata-rata IPK MHS

Seperti yang ditunjukkan Gambar 5.4, maka semakin jelas terlihat perbedaan waktu eksekusi *query* konvensional dan MQT. Setelah menguji waktu eksekusi *query* maka selanjutnya adalah menganalisa sumber daya yang digunakan oleh masing masing *query* yang ditunjukkan Tabel 5.4.

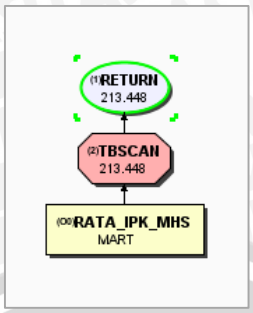
Tabel 5.4 Perbandingan Biaya Sumber Daya Query Rata-rata IPK MHS

Tipe Query	Biaya Sumber Daya (*timerons*)		
	Biaya CPU	Biaya IO	Total Biaya
Konvensional	385.724.000.000.000	7.941	9.794,26
MQT	1.522.740.000.000	237	203,45

Seperti yang terlihat di Tabel 5.4 bahwa biaya *query* konvensional jauh lebih tinggi dibandingkan biaya menggunakan MQT. Hal ini disebabkan karena *query* konvensional melakukan operasi di banyak tabel sementara MQT hanya melakukan operasi di MQT itu sendiri. Lebih detailnya dapat dianalisa menggunakan model akses diagram yang dapat dilihat pada Gambar 5.5 dan Gambar 5.6.



Gambar 5.5 Diagram Akses Query Rata-rata IPK MHS Konvensional



Gambar 5.6 Diagram Akses Query Rata-rata IPK MHS MQT

Gambar 5.5 menampilkan akses diagram dari *query* konvensional yang secara jelas menunjukkan banyak sekali tabel digunakan dalam 1 *query* sehingga hal ini tentu akan berdampak besar terhadap biaya sumber daya yang digunakana. Sementara Gambar 5.6 akses diagram dari *materialized query table* yang terlihat sangat sederhana karena hanya menggunakan 1 tabel.

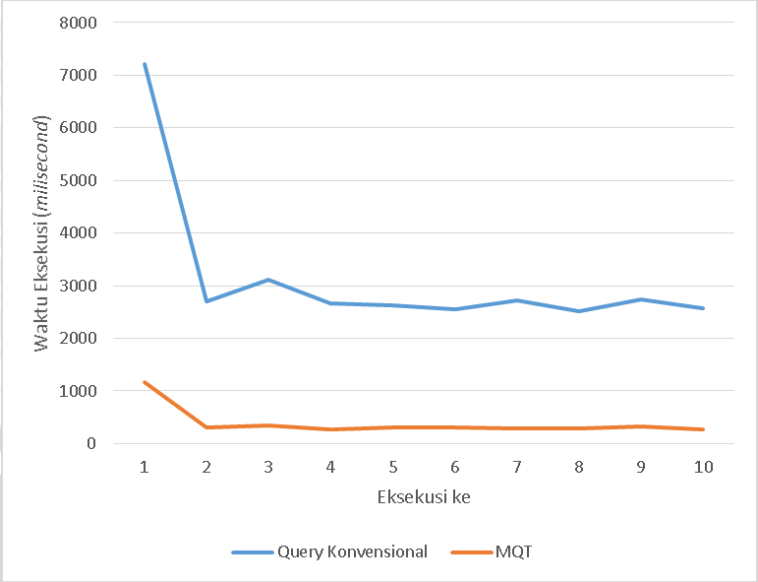
5.1.3 Pengujian *Query* Persebaran Mahasiswa Berdasar Asal Sekolah

Query jumlah mahasiswa aktif merupakan sebuah *query* yang didesain untuk menghitung sebaran asal sekolah mahasiswa yang terdaftar di Universitas Brawijaya. Secara mendasar *query* akan menghitung mahasiswa jumlah mahasiswa dikelompokkan berdasarkan sekolahnya di setiap fakultas, jurusan, tingkat jenjang, program studi, jalur seleksi dan angkatan. Query ini akan memproses data dari tabel fakta status mahasiswa yang memiliki data sebanyak 769.195 baris. Pengujian *query* dilakukan dengan cara melakukan eksekusi *query* sebanyak 10 kali kemudian di rata-rata. Adapun hasil pengujian yang telah dilakukan dapat dilihat di Tabel 5.5

Tabel 5.5 Hasil Waktu Eksekusi Query Asal Sekolah MHS

Jenis *Query*	Percobaa (*milisecond*)										Rata-rata
	1	2	3	4	5	6	7	8	9	10	
Konvensional	7206	2712	3114	2671	2637	2549	2716	2518	2739	2576	3.143,8
MQT	1173	299	347	279	303	311	280	294	318	273	387,7

Pada tabel Tabel 5.5 terlihat jelas hasil waktu eksekusi *query* dengan cara konvensional dan MQT memiliki perbedaan waktu eksekusi yang sangat signifikan. Untuk lebih jelasnya, akan dilakukan perbandingan menggunakan grafik yang ditunjukkan oleh Gambar 5.7.



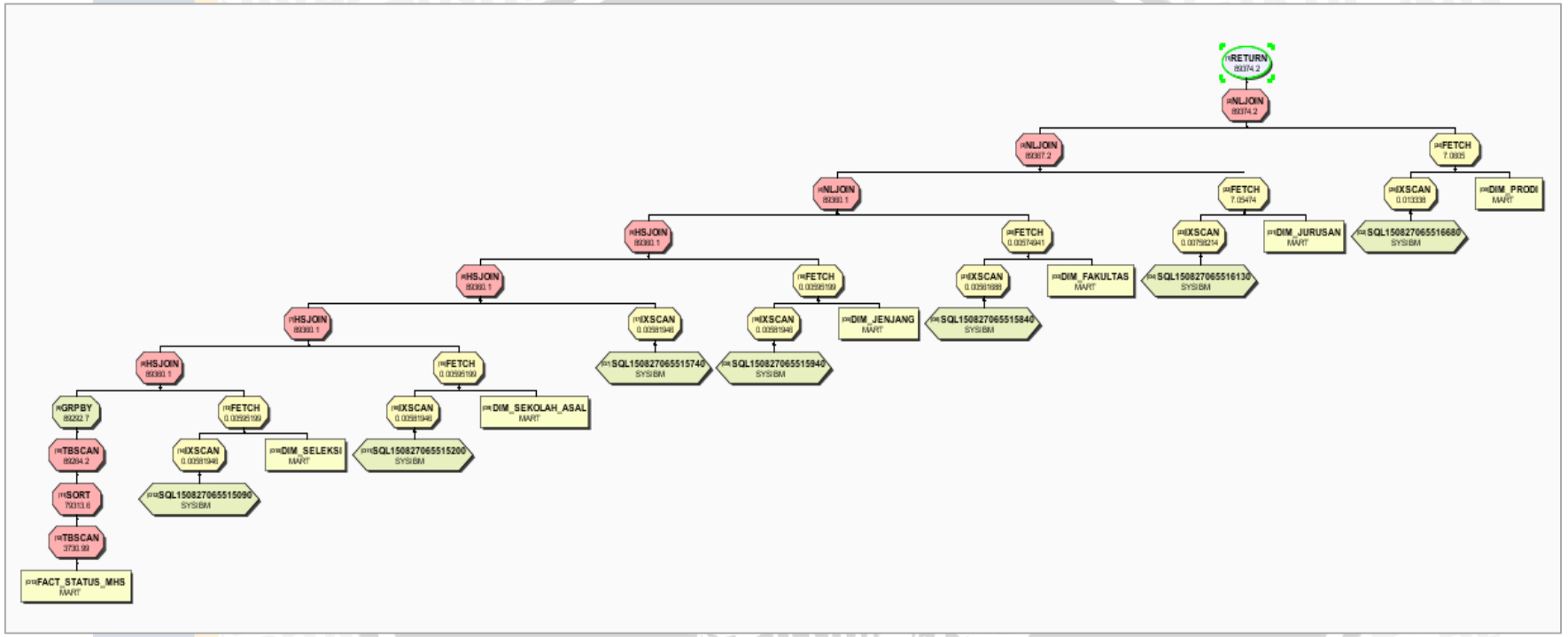
Gambar 5.7 Perbandingan Waktu Eksekusi Query Asal Sekolah MHS

Seperti yang ditunjukkan Gambar 5.7, maka semakin jelas terlihat perbedaan waktu eksekusi *query* konvensional dan MQT. Setelah menguji waktu eksekusi *query* maka selanjutnya adalah menganalisa sumber daya yang digunakan oleh masing masing *query* yang ditunjukkan Tabel 5.6.

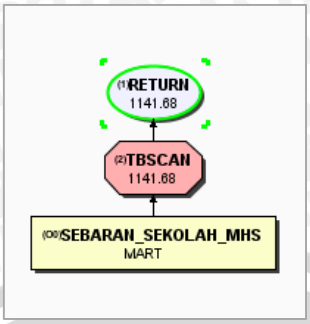
Tabel 5.6 Perbandingan Biaya Sumber Daya Query Asal Sekolah MHS

Tipe Query	Biaya Sumber Daya (*timerons*)		
	Biaya CPU	Biaya IO	Total Biaya
Konvensional	1.239.020.000.000.000	25.021	89.374,2
MQT	24.644.200.000.000	1.249	1.141,7

Seperti yang terlihat di Tabel 5.6 bahwa biaya *query* konvensional jauh lebih tinggi dibandingkan biaya menggunakan MQT. Hal ini disebabkan karena *query* konvensional melakukan operasi di banyak tabel sementara MQT hanya melakukan operasi di MQT itu sendiri. Lebih detailnya dapat dianalisa menggunakan model akses diagram yang dapat dilihat pada Gambar 5.8 dan Gambar 5.9.



Gambar 5.8 Diagram Akses Query Asal Sekolah MHS Konvensional



Gambar 5.9 Diagram Akses Query Asal Sekolah MHS MQT

Gambar 5.8 menampilkan akses diagram dari *query* konvensional yang secara jelas menunjukkan banyak sekali tabel digunakan dalam 1 *query* sehingga hal ini tentu akan berdampak besar terhadap biaya sumber daya yang digunakana. Sementara Gambar 5.9 akses diagram dari *materialized query table* yang terlihat sangat sederhana karena hanya menggunakan 1 tabel.

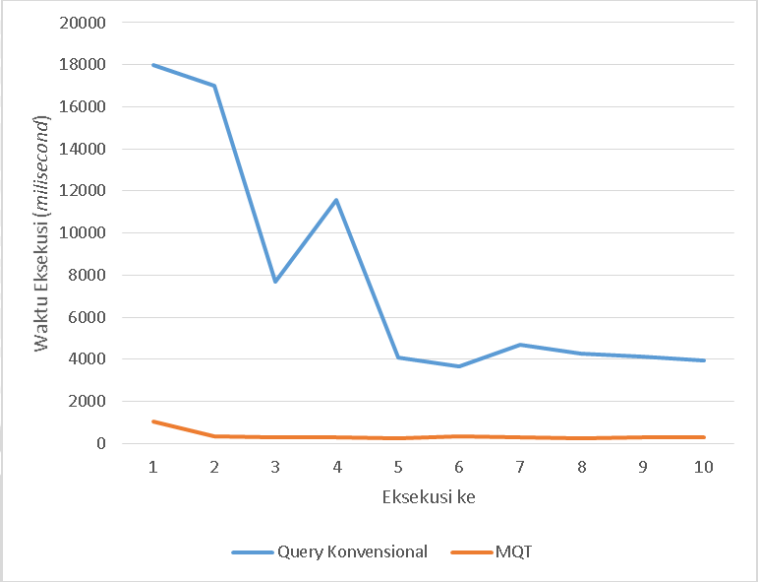
5.1.4 Pengujian *Query* Persebaran Mahasiswa Berdasar Asal Daerah

Query jumlah mahasiswa aktif merupakan sebuah *query* yang didesain untuk menghitung sebaran asal daerah mahasiswa yang terdaftar di Universitas Brawijaya. Secara mendasar *query* akan menghitung mahasiswa jumlah mahasiswa dikelompokkan berdasarkan asal kota, provinsi dan negara di setiap fakultas, jurusan, tingkat jenjang, program studi, jalur seleksi dan angkatan. Query ini akan memproses data dari tabel fakta status mahasiswa yang memiliki data sebanyak 769.195 baris. Pengujian *query* dilakukan dengan cara melakukan eksekusi *query* sebanyak 10 kali kemudian di rata-rata. Adapun hasil pengujian yang telah dilakukan dapat dilihat di Tabel 5.7

Tabel 5.7 Hasil Waktu Eksekusi Query Asal Daerah MHS

Jenis *Query*	Percobaa (*milisecond*)										Rata-rata
	1	2	3	4	5	6	7	8	9	10	
Konvensional	17994	17000	7697	11591	4073	3662	4698	4270	4158	3937	7.908
MQT	1032	331	295	277	276	327	287	272	277	278	365,2

Pada tabel Tabel 5.7 terlihat jelas hasil waktu eksekusi *query* dengan cara konvensional dan MQT memiliki perbedaan waktu eksekusi yang sangat signifikan. Untuk lebih jelasnya, akan dilakukan perbandingan menggunakan grafik yang ditunjukkan oleh Gambar 5.10.



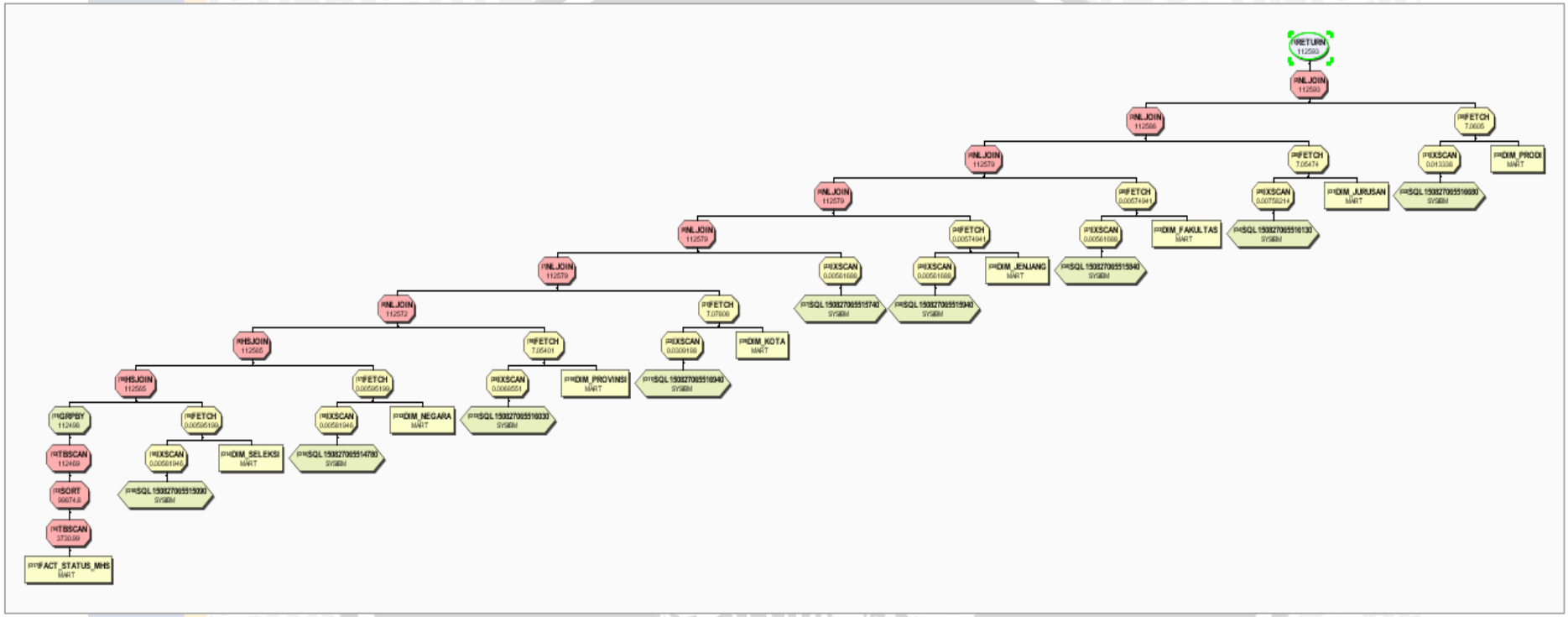
Gambar 5.10 Perbandingan Eksekusi Query Asal Daerah MHS

Seperti yang ditunjukkan Gambar 5.10, maka semakin jelas terlihat perbedaan waktu eksekusi *query* konvensional dan MQT. Setelah menguji waktu eksekusi *query* maka selanjutnya adalah menganalisa sumber daya yang digunakan oleh masing masing *query* yang ditunjukkan Tabel 5.8.

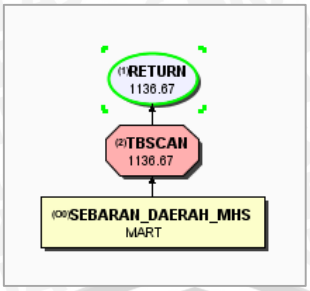
Tabel 5.8 Perbandingan Biaya Sumber Daya Query Asal Daerah MHS

Tipe Query	Biaya Sumber Daya (*timerons*)		
	Biaya CPU	Biaya IO	Total Biaya
Konvensional	1.437.190.000.000.000	31.017	112.593
MQT	21.203.900.000.000	1.249	1.136,7

Seperti yang terlihat di Tabel 5.8 bahwa biaya *query* konvensional jauh lebih tinggi dibandingkan biaya menggunakan MQT. Hal ini disebabkan karena *query* konvensional melakukan operasi di banyak tabel sementara MQT hanya melakukan operasi di MQT itu sendiri. Lebih detailnya dapat dianalisa menggunakan model akses diagram yang dapat dilihat pada Gambar 5.11 dan Gambar 5.12.



Gambar 5.11 Diagram Akses Query Asal Daerah MHS



Gambar 5.12 Diagram Akses Query Asal Daerah MHS MQT

Gambar 5.11 menampilkan akses diagram dari *query* konvensional yang secara jelas menunjukkan banyak sekali tabel digunakan dalam 1 *query* sehingga hal ini tentu akan berdampak besar terhadap biaya sumber daya yang digunakana. Sementara Gambar 5.12 akses diagram dari *materialized query table* yang terlihat sangat sederhana karena hanya menggunakan 1 tabel.

5.1.5 Analisa Pengujian

Berdasarkan pengujian performa yang dilakukan pada bab 5.1.1, bab 5.1.2, bab 5.1.3 dan bab 5.1.4 dapat dilihat bahwa waktu eksekusi *query* konvensional cenderung memiliki volatilitas yang tinggi atau sering berubah-ubah jika dibandingkan dengan MQT. Hal ini dapat dilihat pada Gambar 5.1, Gambar 5.4, Gambar 5.7 dan Gambar 5.10. Waktu eksekusi *query* konvensional yang diwakili garis biru memiliki pola naik turun bahkan setelah eksekusi pertama. Berbeda dengan waktu eksekusi MQT yang diwakili garis kuning memiliki pola turun dan stabil.

Volatilitas tinggi ini dapat dikaitkan dengan kebutuhan sumberdaya yang jauh lebih tinggi untuk memproses *query* konvensional jika dibandingkan dengan MQT. Perbandingan penggunaan sumber daya oleh *query* dapat dilihat dari Tabel 5.2, Tabel 5.4, Tabel 5.6 dan Tabel 5.8. Tingginya sumber daya yang dibutuhkan *query* konvensional membuat sistem kesulitan untuk mengalokasikan dengan efektif dan efisien.

5.2 Pengujian Ketahanan

Pengujian ketahan dilakukan dengan bantuan perangkat lunak *apache-jmeter* yang memiliki skenario melakukan eksekusi *query* dari beberapa pengguna secara bersamaan. Jumlah pengguna akan ditingkatkan secara bertahap sampai dirasa sistem tidak mampu melayani secara maksimal lagi.

Untuk melakukan pengujian ini diperlukan konfigurasi basis data *db2* untuk bisa menerima koneksi dari banyak pengguna secara bersamaan. hal ini dilakukan dengan cara mengeksekusi kode konfigurasi basis data berikut di *server*.

```
update dbm cfg using MAX_COORDAGENTS 10000 AUTOMATIC
update dbm cfg using MAX_CONNECTION 10000 AUTOMATIC
update db cfg using MAX_APPLS 10000 AUTOMATIC
```

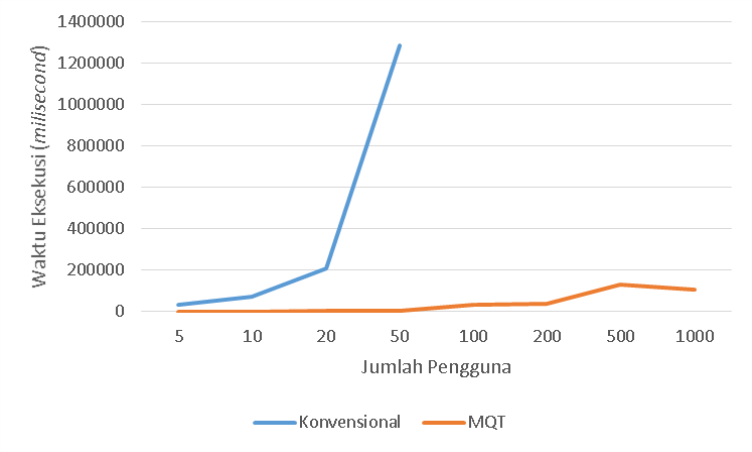
Baris 1 kode diatas merupakan konfigurasi untuk meningkatkan jumlah aplikasi terkoordinasi menggunakan basis data hingga 10.000 aplikasi. Baris 2 merupakan

konfigurasi untuk meningkatkan jumlah koneksi yang bisa dibuat hingga 10.000 koneksi. Dan baris 3 merupakan konfigurasi untuk meningkatkan jumlah aplikasi hingga 10.000 aplikasi di level basis data.

Adapun pengguna yang digunakan dalam pengujian ini dimulai dari 5 pengguna, kemudian meningkat menjadi 10, 20, 50, 100, 200, 500 dan terakhir 1000 pengguna. Hasil pengujian dari tiap-tiap kelompok pengguna akan dirata-rata dan akan dibandingkan antara *query* konvensional dan MQT.

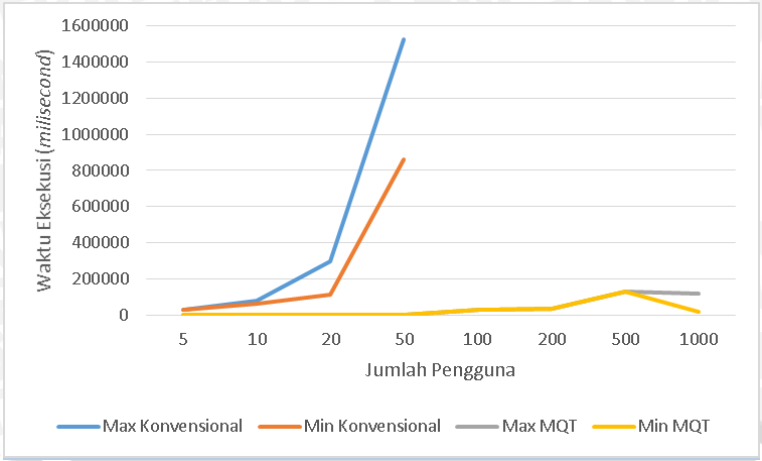
5.2.1 Pengujian Query Jumlah Mahasiswa Aktif

Pengujian ketahanan query jumlah mahasiswa aktif dilakukan melalui serangkaian tahapan. Tiap-tiap tahapan akan melakukan eksekusi query dengan sejumlah pengguna secara bersamaan. Jumlah pengguna di tiap-tiap tahap akan selalu ditingkatkan dimulai dari 5 pengguna meningkat menjadi 10, 20, 50, 100, 200, 500 dan akhirnya 1000 pengguna. Di setiap tahap hasil eksekusi dari pengguna-penguna tersebut akandirata-rata, dicari nilai minumum maksimum dan dicari nilai deviasinya. Hal ini berguna untuk mengetahui ketahanan performa basis data pada tiap-tiap tahapan. Seperti pada pengujian performa hasil pengujian *query* konvensional akan dibandingkan dengan MQT. Adapun perbandingan hasil pengujian dapat dilihat pada Gambar 5.13, Gambar 5.14 dan Gambar 5.15.



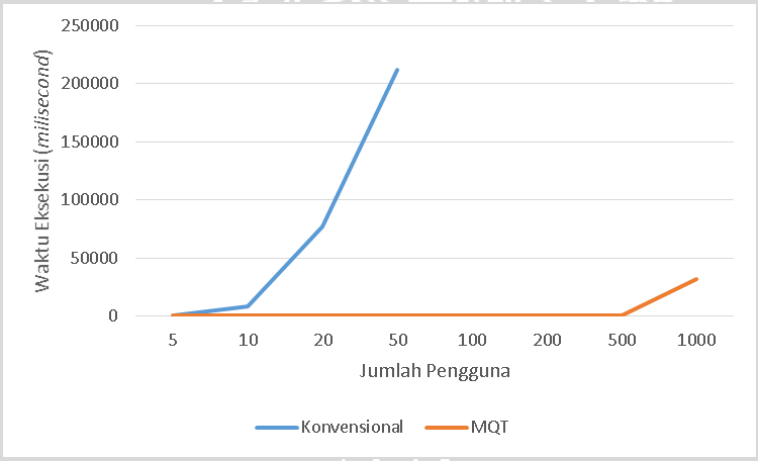
Gambar 5.13 Rata-rata Waktu Eksekusi Query Jumlah MHS Aktif

Gambar 5.13 menunjukkan grafik perbandingan rata-rata waktu eksekusi *query* jumlah mahasiswa aktif. Disana terlihat bahwa rata-rata eksekusi *query* konvensional membutuhkan waktu yang jauh lebih tinggi. Bahkan hanya sampai tahap 50 pengguna sistem sudah tidak mampu menangani *query* konvensional. Hal itu tercermin dari rata-rata waktu eksekusi *query* konvensional yang mencapai 21 menit lebih pada tahap jumlah pengguna 50 orang. Sementara rata-rata eksekusi MQT jauh lebih cepat dan stabil bahkan sampai tahap pengguna mencapai 1000 orang sistem masih mampu melayani *query* MQT dengan baik. Hal ini terlihat dari rata-rata waktu eksekusi yang hanya mencapai 1-2 menit saja pada jumlah pengguna 1000 orang.



Gambar 5.14 Nilai Minimum dan Maksimum Waktu Eksekusi Query Jumlah MHS Aktif

Gambar 5.14 menunjukkan perbandingan nilai minimum dan maksimum *query* konvensional dengan *query* MQT. Hal ini berguna untuk mengetahui perbandingan rentang antara maksimum dan minimum eksekusi. Seperti yang terlihat *query* konvensional memiliki rentang yang cukup jauh antara minimum waktu eksekusi dan maksimum waktu eksekusi. Sehingga hal ini memungkinkan perbedaan jauh antara waktu eksekusi penguna satu dengan yang lain.



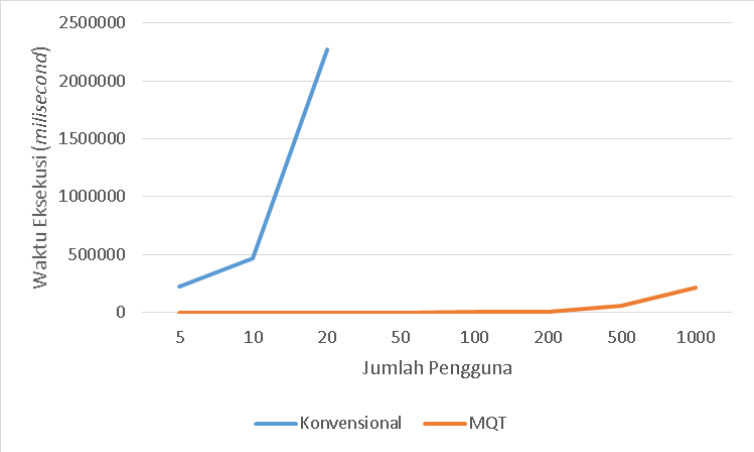
Gambar 5.15 Deviasi Waktu Eksekusi Query Jumlah MHS Aktif

Gambar 5.15 menunjukkan perbandingan nilai deviasi eksekusi antara *query* konvensional dan *query* MQT. Peran deviasi di sini adalah untuk mengetahui kestabilan waktu eksekusi *query*. Semakin tinggi nilai deviasi maka semakin tidak stabil waktu eksekusi *query* tersebut. Seperti yang terlihat di Gambar 5.15 nilai deviasi *query* konvensional sangat tinggi sehingga waktu eksekusi *query* satu dengan yang lain dapat sangat berbeda.

5.2.2 Pengujian Qury Rata-rata IPK Mahasiswa

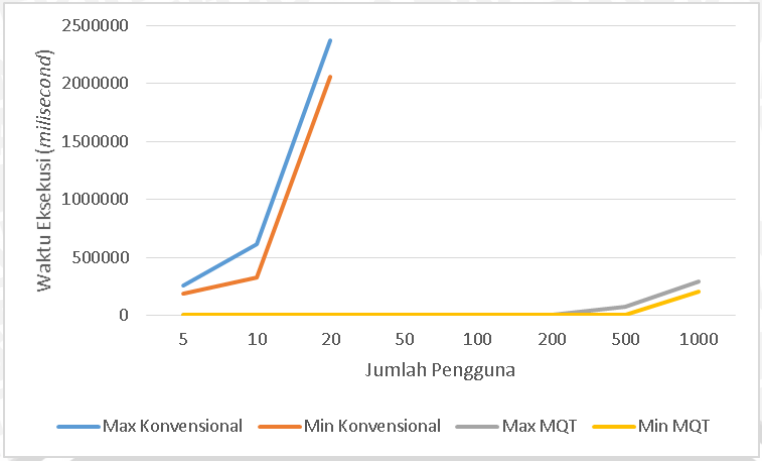
Pengujian ketahanan *query* jumlah mahasiswa aktif dilakukan melalui serangkaian tahapan. Tiap-tiap tahapan akan melakukan eksekusi query dengan

sejumlah pengguna secara bersamaan. Jumlah pengguna di tiap-tiap tahap akan selalu ditingkatkan dimulai dari 5 pengguna meningkat menjadi 10, 20, 50, 100, 200, 500 dan akhirnya 1000 pengguna. Di setiap tahap hasil eksekusi dari pengguna-penguna tersebut akandirata-rata, dicari nilai minumum maksimum dan dicari nilai deviasinya. Hal ini berguna untuk mengetahui ketahanan performa basis data pada tiap-tiap tahapan. Seperti pada pengujian performa hasil pengujian *query* konvensional akan dibandingkan dengan MQT. Adapun perbandingan hasil pengujian dapat dilihat pada Gambar 5.16, Gambar 5.17 dan Gambar 5.18.



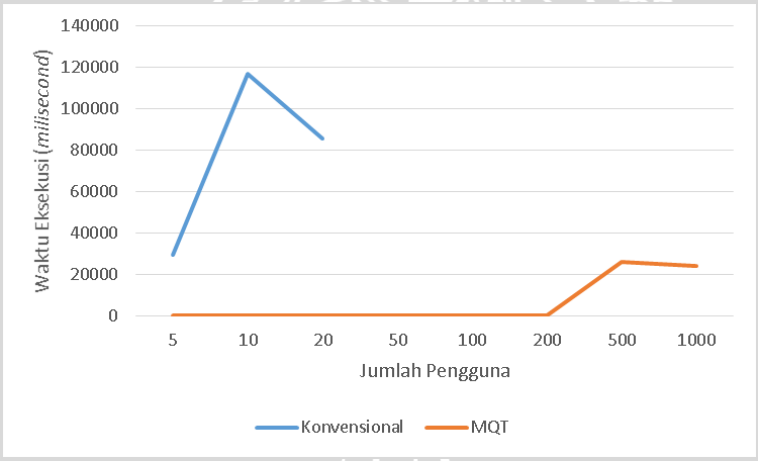
Gambar 5.16 Rata-rata Waktu Eksekusi Query Rata-rata IPK MHS

Gambar 5.16 menunjukkan grafik perbandingan rata-rata waktu eksekusi *query* jumlah mahasiswa aktif. Disana terlihat bahwa rata-rata eksekusi *query* konvensional membutuhkan waktu yang jauh lebih tinggi. Bahkan hanya sampai tahap 20 pengguna sistem sudah tidak mampu menangani *query* konvensional. Hal itu tercermin dari rata-rata waktu eksekusi *query* konvensional yang mencapai 37 menit lebih pada tahap jumlah pengguna 20 orang. Sementara rata-rata eksekusi MQT jauh lebih cepat dan stabil bahkan sampai tahap pengguna mencapai 1000 orang sistem masih mampu melayani *query* MQT dengan baik. Hal ini terlihat dari rata-rata waktu eksekusi yang hanya mencapai 2-3 menit saja pada jumlah pengguna 1000 orang.



Gambar 5.17 Nilai Minimum dan Maksimum Waktu Eksekusi Query Rata-rata IPK MHS

Gambar 5.17 menunjukkan perbandingan nilai minimum dan maksimum *query* konvensional dengan *query* MQT. Hal ini berguna untuk mengetahui perbandingan rentang antara maksimum dan minimum eksekusi. Seperti yang terlihat *query* konvensional memiliki rentang yang cukup jauh antara minimum waktu eksekusi dan maksimum waktu eksekusi. Sehingga hal ini memungkinkan perbedaan jauh antara waktu eksekusi penguna satu dengan yang lain.



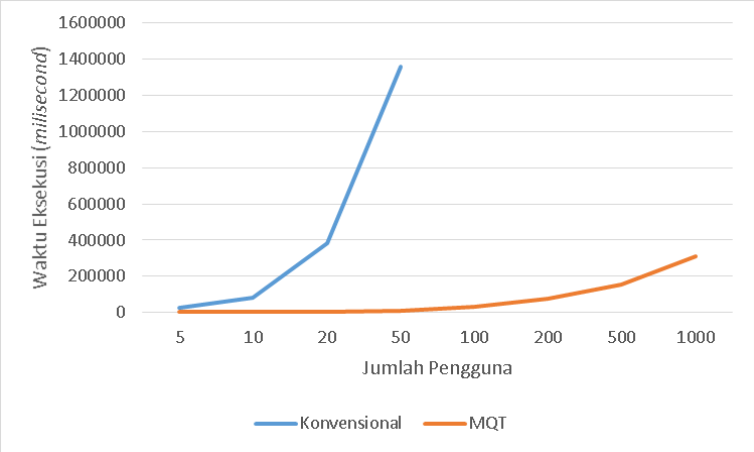
Gambar 5.18 Deviasi Waktu Eksekusi Query Jumlah MHS Aktif

Gambar 5.18 menunjukkan perbandingan nilai deviasi eksekusi antara *query* konvensional dan *query* MQT. Peran deviasi di sini adalah untuk mengetahui kestabilan waktu eksekusi *query*. Semakin tinggi nilai deviasi maka semakin tidak stabil waktu eksekusi *query* tersebut. Seperti yang terlihat di Gambar 5.18 nilai deviasi *query* konvensional sangat tinggi sehingga waktu eksekusi *query* satu dengan yang lain dapat sangat berbeda.

5.2.3 Pengujian Query Persebaran Mahasiswa Berdasar Asal Sekolah

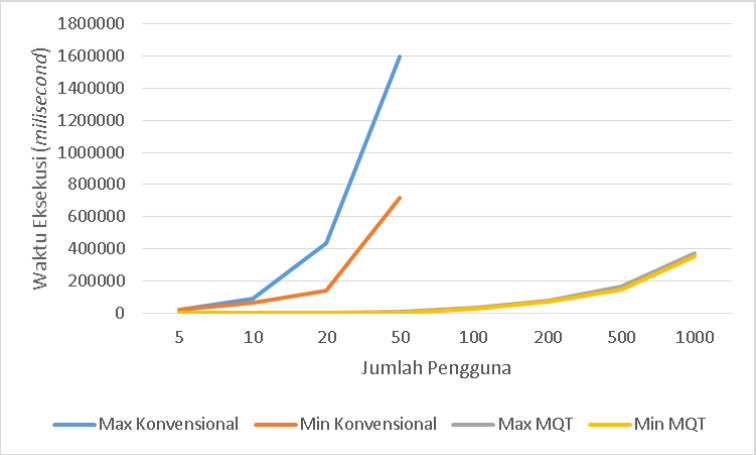
Pengujian ketahanan *query* jumlah mahasiswa aktif dilakukan melalui serangkaian tahapan. Tiap-tiap tahapan akan melakukan eksekusi query dengan

sejumlah pengguna secara bersamaan. Jumlah pengguna di tiap-tiap tahap akan selalu ditingkatkan dimulai dari 5 pengguna meningkat menjadi 10, 20, 50, 100, 200, 500 dan akhirnya 1000 pengguna. Di setiap tahap hasil eksekusi dari pengguna-penguna tersebut akandirata-rata, dicari nilai minumum maksimum dan dicari nilai deviasinya. Hal ini berguna untuk mengetahui ketahanan performa basis data pada tiap-tiap tahapan. Seperti pada pengujian performa hasil pengujian *query* konvensional akan dibandingkan dengan MQT. Adapun perbandingan hasil pengujian dapat dilihat pada Gambar 5.19, Gambar 5.20 dan Gambar 5.21.



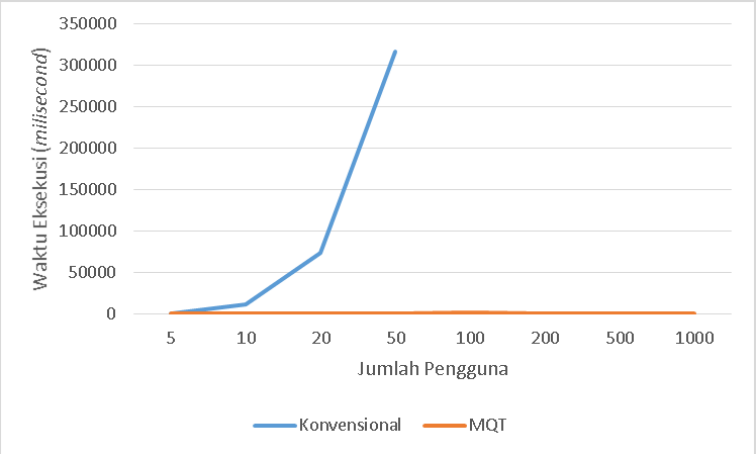
Gambar 5.19 Rata-rata Waktu Eksekusi Query Asal Sekolah MHS

Gambar 5.19 menunjukkan grafik perbandingan rata-rata waktu eksekusi *query* jumlah mahasiswa aktif. Disana terlihat bahwa rata-rata eksekusi *query* konvensional membutuhkan waktu yang jauh lebih tinggi. Bahkan hanya sampai tahap 50 pengguna sistem sudah tidak mampu menangani *query* konvensional. Hal itu tercermin dari rata-rata waktu eksekusi *query* konvensional yang mencapai 22 menit lebih pada tahap jumlah pengguna 50 orang. Sementara rata-rata eksekusi MQT jauh lebih cepat dan stabil bahkan sampai tahap pengguna mencapai 1000 orang sistem masih mampu melayani *query* MQT dengan baik. Hal ini terlihat dari rata-rata waktu eksekusi yang hanya mencapai 5 menit saja pada jumlah pengguna 1000 orang.



Gambar 5.20 Nilai Minimum dan Maksimum Waktu Eksekusi Query Asal Sekolah MHS

Gambar 5.20 menunjukkan perbandingan nilai minimum dan maksimum *query* konvensional dengan *query* MQT. Hal ini berguna untuk mengetahui perbandingan rentang antara maksimum dan minimum eksekusi. Seperti yang terlihat *query* konvensional memiliki rentang yang cukup jauh antara minimum waktu eksekusi dan maksimum waktu eksekusi. Sehingga hal ini memungkinkan perbedaan jauh antara waktu eksekusi penguna satu dengan yang lain.



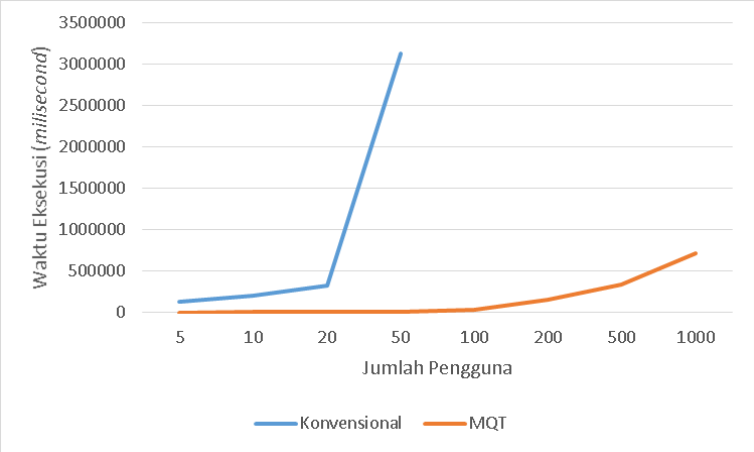
Gambar 5.21 Deviasi Waktu Eksekusi Query Asal Sekolah MHS

Gambar 5.21 menunjukkan perbandingan nilai deviasi eksekusi antara *query* konvensional dan *query* MQT. Peran deviasi di sini adalah untuk mengetahui kestabilan waktu eksekusi *query*. Semakin tinggi nilai deviasi maka semakin tidak stabil waktu eksekusi *query* tersebut. Seperti yang terlihat di Gambar 5.21 nilai deviasi *query* konvensional sangat tinggi sehingga waktu eksekusi *query* satu dengan yang lain dapat sangat berbeda.

5.2.4 Pengujian Query Persebaran Mahasiswa Berdasar Asal Daerah

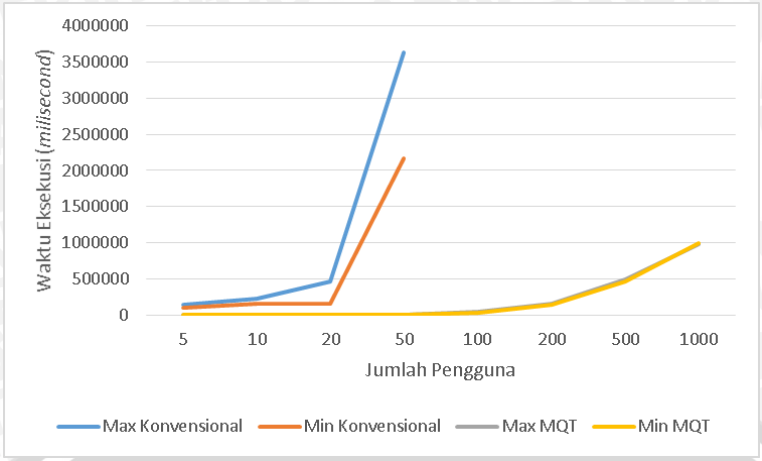
Pengujian ketahanan *query* jumlah mahasiswa aktif dilakukan melalui serangkaian tahapan. Tiap-tiap tahapan akan melakukan eksekusi query dengan

sejumlah pengguna secara bersamaan. Jumlah pengguna di tiap-tiap tahap akan selalu ditingkatkan dimulai dari 5 pengguna meningkat menjadi 10, 20, 50, 100, 200, 500 dan akhirnya 1000 pengguna. Di setiap tahap hasil eksekusi dari pengguna-penguna tersebut akandirata-rata, dicari nilai minumum maksimum dan dicari nilai deviasinya. Hal ini berguna untuk mengetahui ketahanan performa basis data pada tiap-tiap tahapan. Seperti pada pengujian performa hasil pengujian *query* konvensional akan dibandingkan dengan MQT. Adapun perbandingan hasil pengujian dapat dilihat pada Gambar 5.22, Gambar 5.23 dan Gambar 5.24.



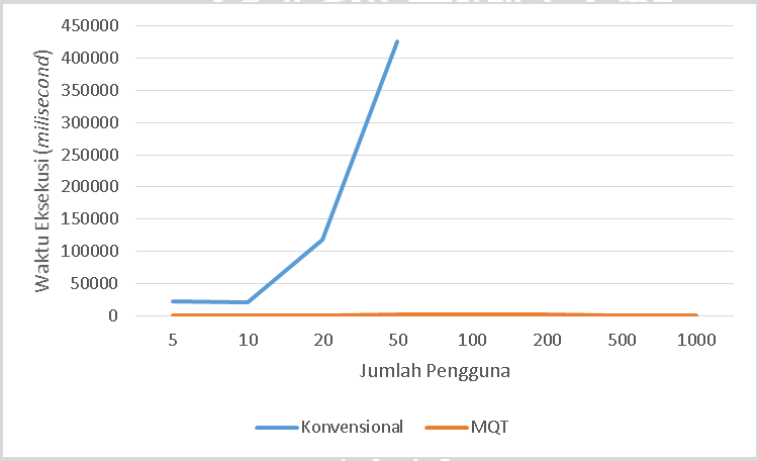
Gambar 5.22 Rata-rata Waktu Eksekusi Query Asal Daerah MHS

Gambar 5.22 menunjukkan grafik perbandingan rata-rata waktu eksekusi *query* jumlah mahasiswa aktif. Disana terlihat bahwa rata-rata eksekusi *query* konvensional membutuhkan waktu yang jauh lebih tinggi. Bahkan hanya sampai tahap 50 pengguna sistem sudah tidak mampu menangani *query* konvensional. Hal itu tercermin dari rata-rata waktu eksekusi *query* konvensional yang mencapai 52 menit lebih pada tahap jumlah pengguna 50 orang. Sementara rata-rata eksekusi MQT jauh lebih cepat dan stabil bahkan sampai tahap pengguna mencapai 1000 orang sistem masih mampu melayani *query* MQT dengan baik. Hal ini terlihat dari rata-rata waktu eksekusi yang hanya mencapai 11 menit saja pada jumlah pengguna 1000 orang.



Gambar 5.23 Nilai Minimum dan Maksimum Waktu Eksekusi Query Jumlah MHS Aktif

Gambar 5.23 menunjukkan perbandingan nilai minimum dan maksimum *query* konvensional dengan *query* MQT. Hal ini berguna untuk mengetahui perbandingan rentang antara maksimum dan minimum eksekusi. Seperti yang terlihat *query* konvensional memiliki rentang yang cukup jauh antara minimum waktu eksekusi dan maksimum waktu eksekusi. Sehingga hal ini memungkinkan perbedaan jauh antara waktu eksekusi penguna satu dengan yang lain.



Gambar 5.24 Deviasi Waktu Eksekusi Query Jumlah MHS Aktif

Gambar 5.24 menunjukkan perbandingan nilai deviasi eksekusi antara *query* konvensional dan *query* MQT. Peran deviasi di sini adalah untuk mengetahui kestabilan waktu eksekusi *query*. Semakin tinggi nilai deviasi maka semakin tidak stabil waktu eksekusi *query* tersebut. Seperti yang terlihat di Gambar 5.24 nilai deviasi *query* konvensional sangat tinggi sehingga waktu eksekusi *query* satu dengan yang lain dapat sangat berbeda.

BAB 6 PENUTUP

6.1 Kesimpulan

Sebagaimana tujuan dari penelitian ini dimaksudkan untuk melakukan implementasi *materialized query table* untuk pembangunan *data mart* dan melakukan perbandingan terhadap penggunaan *query* secara konvensional, maka berikut ini merupakan kesimpulan dari penelitian yang telah dilakukan:

1. Rancangan *data mart* menghasilkan 2 tabel fakta dan 13 tabel dimensi yang membentuk model multidimensional *snowflake*. Sementara dalam rancangan *materialized query table* digunakan sebagai mode akses *query*.
2. Implementasi MQT pada *data mart* terbukti dapat meningkatkan performa *data mart*. Hal ini dapat dilihat dari pengujian performa waktu eksekusi *query* jumlah mahasiswa aktif yang pada eksekusi pertama *query* konvensional membutuhkan waktu 5273 *milisecond* dan *query* MQT membutuhkan waktu 1014 *milisecond*. Sementara pada eksekusi kedua dan selanjutnya *query* konvensional membutuhkan waktu dengan rentang 2651-3285 *milisecond* dan *query* MQT membutuhkan waktu dengan rentang 42-57 *milisecond*.
3. Implementasi MQT pada *data mart* terbukti dapat meningkatkan efisiensi penggunaan sumber daya basis data. Hal ini dapat dilihat pada pengujian performa dimana *query* konvensional membutuhkan sumber daya 90.640,9 *timerons* sementara *query* MQT hanya membutuhkan sumber daya 185,28 *timerons*.
4. Implementasi MQT pada *data mart* dapat meningkatkan ketahanan *data mart*. Hal ini terlihat dari jumlah pengguna yang mampu ditangani sistem menggunakan *query* konvensional jauh lebih sedikit dari pada jumlah pengguna yang mampu ditangani ketika menggunakan *data mart*.

6.2 Saran

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

1. *Materialized query table* merupakan sebuah fitur yang mampu mengoptimalkan kinerja *data mart*.
2. Pada penyusunan *materialized query table* disarankan menggunakan *query* yang optimal dikarenakan akan mempengaruhi proses *refresh* yang diperlukan oleh MQT.
3. Untuk mendapatkan hasil yang lebih optimal pada implementasi MQT maka disarankan untuk lebih memperhatikan dan mengelola metode refresh yang digunakan.

DAFTAR PUSTAKA

Ballard, C. et al., 2005. Data Mart Consolidation: Getting Control of Your Enterprise Information. First penyunt. New York: IBM Corp.

Chandramitasari, W., 2014. Optimasi Online Analytical Processing (OLAP) pada Data Warehouse dengan Pendekatan Materialized Query Table, Malang: Universitas Brawijaya.

Chhabra, R. & Pahwa, P., 2014. Data Mart Designing and Integration Approach. International Journal of Computer Science and Mobile Computing, III(4), pp. 74-79.

Direktorat Jendral Pendidikan Tinggi Republik Indonesia, 2015. Pangkalan Data Pendidikan Tinggi Direktorat Jendral Pendidikan Tinggi. [Online] Available at: http://forlap.dikti.go.id/ [Diakses Juli 2015].

Gallo, C. & Perilli, M., 2010. Data Warehouse Design and Management: Theory and Practice, Foggia: Foggia University.

Golfarelli, M. & Rizzi, S., 2011. Data Warehouse Testing. International Journal of Data Warehousing and Mining, 7(2).

Imhoff, C., Galemmo, N. & Geiger, J., 2003. Mastering Data Warehouse Design. Indiana: Wiley Publishing.

Lawrence, M. & Rau-Chaplin, A., 2008. Dynamic View selection for OLAP. International Journal of Data Warehousing and Mining, IV(1), pp. 47-61.

Pandya, B. & Shah, S., 2014. Proposed Local Data Mart Approach For Data Warehouse Architecture. International Journal of Emerging Technology and Advanced Engineering, IV(2).

Paraboschi, S., Sindoni, G., Baralis, E. & Teniente, E., 2003. Materialized View in Multidimensional Databases. Dalam: M. Rafanelli, penyunt. Multidimensional Databases: Problems and Solutions. Hershey: Idea Group Publishing, pp. 222-250.

Paraboschi, S., Sindoni, G., Baralis, E. & Teniente, E., 2003. Materialized Views in Multidimensional Databases. Dalam: M. Rafanelli, penyunt. Multidimensional Databases: Problems and Solutions. Hershey: Idea Group Publishing, pp. 222-251.

Ponniah, P., 2001. Data Warehousing Fundamentals: A Comprehensive Guide for IT Professional. New York: John Wiley & Sons Inc.

Prabhu, 2008. Data Warehousing: Concepts, Techniques, Products and Applications. 3rd penyunt. s.l.:PHI Learning Pvt. Ltd..

Silberschatz, A., Korth, H. & Sudarshan, 2009. Database System Concepts. 6th penyunt. New York: Mc Graw-Hill.

Theodoratos, D., Xu, W. & Simitsis, A., 2009. Materialized View Selection for Data Warehouse Design. Dalam: Encyclopedia of Data Warehouse and Data Mining Second Edition. Chicago: IGI Global, pp. 1182-1187.

Turban, E., Sharda, R., Delen, D. & King, D., 2010. Business Intelligence A Managerial Approach. 2nd penyunt. New Jersey: Prentice Hall.

Universitas Brawijaya, 2015. Profil Universitas Brawijaya. [Online] Available at: http://www.ub.ac.id [Diakses 5 Mei 2015].

LAMPIRAN A HASIL WAWANCARA

1. Apa saja tugas dan fungsi dari Bagian Akademik Universitas Brawijaya?

Tugas dan fungsi Bagian Akademik Universitas Brawijaya adalah mempersiapkan bahan penyusunan peraturan bidang akademik untuk buku pedoman akademik dan kalender akademik, melakukan proses penerimaan mahasiswa baru, menghimpun dan mengkaji serta menyusun saran pemecahan masalah di bidang registrasi dan statistik, mempersiapkan dan melakukan urusan pelaksanaan registrasi dan statistik mahasiswa, memperisapkan kartu tanda mahasiswa dan pemberian nomor induk, melaksanakan tugas lain yang diberikan atasan langsung.

2. Dalam menjalankan tugas dan fungsi Bagian Akademik Universitas Brawijaya, sistem informasi apakah yang digunakan dan apa peran masing-masing sistem informasi tersebut?

Sistem secara keseluruhan disebut SIAKAD (Sistem Akademik) Universitas Brawijaya didalamnya terdapat subsistem yaitu

SIMPEL = Sistem Pelaporan Online

SIAM = Sistem Informasi Akademik Mahasiswa

SIUDA = Sistem Informasi Wisuda

Siregi = Sistem Informasi Registrasi

3. Seberapa sering proses evaluasi akademik mahasiswa dilakukan Universitas Brawijaya?

Proses evaluasi dilakukan di setiap akhir semester di fakultas masing-masing

4. View data (tampilan data) seperti apa yang dapat digunakan untuk membantu proses evaluasi akademik Universitas Brawijaya?

Untuk membantu pelaporan yang diguankan antara lain:

- Laporan Rata-rata IPK mahasiswa per fakultas, jurusan, jenjang, prodi dan angkatan
- Laporan sebaran jumlah mahasiswa aktif di tiap-tiap fakultas, jurusan, jenjang dan prodi
- Laporan sebaran sekolah asal mahasiswa di tiap-tiap fakultas, jurusan, jenjang dan program studi
- Laporan sebaran daerah asal mahasiswa di tiap-tiap fakultas, jurusan, jenjang dan program studi

LAMPIRAN B IMPLEMENTASI BASIS DATA OLTP SIAKAD

Berikut adalah implementasi *source code data definition language* (DDL) dari skema basis data sistem OLTP SIAKAD

1. Tabel Fakultas

```
CREATE TABLE SIAKAD.FAKULTAS (
  K_FAKULTAS VARCHAR(9) NOT NULL,
  CONTENT VARCHAR(45) NULL,
  PRIMARY KEY(K_FAKULTAS)
);
```

Syntax SQL diatas adalah DDL untuk membuat tabel fakultas dengan kunci utama adalah K_FAKULTAS.

2. Tabel Jurusan

```
CREATE TABLE SIAKAD.JURUSAN (
  K_JURUSAN VARCHAR(9) NOT NULL,
  K_FAKULTAS VARCHAR(9) NOT NULL,
  CONTENT VARCHAR(50) NULL,
  PRIMARY KEY(K_JURUSAN, K_FAKULTAS),
  FOREIGN KEY(K_FAKULTAS)
    REFERENCES SIAKAD.FAKULTAS(K_FAKULTAS)
);
```

Syntax SQL diatas adalah DDL untuk membuat tabel jurusan dengan kunci utama K_JURUSAN, K_FAKULTAS dan memiliki referensi ke tabel fakultas.

3. Tabel Jenjang

```
CREATE TABLE SIAKAD.JENJANG (
  K_JENJANG VARCHAR(9) NOT NULL,
  CONTENT VARCHAR(25) NULL,
  PRIMARY KEY(K_JENJANG)
);
```

Syntax SQL diatas adalah DDL untuk membuat tabel jenjang dengan kunci utama adalah K_JENJANG.

4. Tabel Program Studi

```
CREATE TABLE SIAKAD.PROGRAM_STUDI (
  K_PROGRAM_STUDI VARCHAR(9) NOT NULL,
  K_FAKULTAS VARCHAR(9) NOT NULL,
  K_JURUSAN VARCHAR(9) NOT NULL,
  K_JENJANG VARCHAR(9) NOT NULL,
```

```
  CONTENT VARCHAR(50) NULL,
  PRIMARY KEY(K_PROGRAM_STUDI, K_FAKULTAS, K_JURUSAN,
K_JENJANG),
  FOREIGN KEY(K_JURUSAN, K_FAKULTAS)
    REFERENCES SIAKAD.JURUSAN(K_JURUSAN, K_FAKULTAS),
  FOREIGN KEY(K_JENJANG)
    REFERENCES SIAKAD.JENJANG(K_JENJANG)
);
```

Syntax SQL diatas adalah DDL untuk membuat tabel program studi dengan kunci utama adalah K_PROGRAM_STUDI, K_JENJANG, K_JURUSAN dan K_FAKULTAS serta memiliki referensi ke tabel fakultas, jurusan dan jenjang

5. Tabel Seleksi

```
CREATE TABLE SIAKAD.SELEKSI (
  K_SELEKSI VARCHAR(9) NOT NULL,
  CONTENT VARCHAR(45) NULL,
  PRIMARY KEY(K_SELEKSI)
);
```

Syntax SQL diatas adalah DDL untuk membuat tabel seleksi dengan kunci utama adalah K_SELEKSI.

6. Tabel Negara

```
CREATE TABLE SIAKAD.NEGARA (
  K_NEGARA VARCHAR(9) NOT NULL,
  CONTENT VARCHAR(50) NULL,
  PRIMARY KEY(K_NEGARA)
);
```

Syntax SQL diatas adalah DDL untuk membuat tabel negara dengan kunci utama adalah K_NEGARA.

7. Tabel Propinsi

```
CREATE TABLE SIAKAD.PROPINSI (
  K_PROPINSI VARCHAR(9) NOT NULL,
  K_NEGARA VARCHAR(9) NOT NULL,
  CONTENT VARCHAR(60) NULL,
  PRIMARY KEY(K_PROPINSI, K_NEGARA),
  FOREIGN KEY(K_NEGARA)
    REFERENCES SIAKAD.NEGARA(K_NEGARA)
);
```

Syntax SQL diatas adalah DDL untuk membuat tabel propinsi dengan kunci utama adalah K_PROPINSI dan K_NEGARA yang memiliki referensi pada tabel negara

8. Tabel Kota

```
CREATE TABLE SIAKAD.KOTA (
  K_KOTA VARCHAR(9) NOT NULL,
  K_NEGARA VARCHAR(9) NOT NULL,
  K_PROPINSI VARCHAR(9) NOT NULL,
  CONTENT VARCHAR(60) NULL,
  PRIMARY KEY(K_KOTA, K_NEGARA, K_PROPINSI),
  FOREIGN KEY(K_PROPINSI, K_NEGARA)
    REFERENCES SIAKAD.PROPINSI(K_PROPINSI, K_NEGARA)
);
```

Syntax SQL diatas adalah DDL untuk membuat tabel kota dengan kunci utama adalah K_KOTA, K_PROPINSI dan K_NEGARA yang memiliki referensi pada tabel propinsi dan negara.

9. Tabel Sekolah

```
CREATE TABLE SIAKAD.SEKOLAH (
  K_SEKOLAH VARCHAR(9) NOT NULL,
  K_PROPINSI VARCHAR(9) NOT NULL,
  K_NEGARA VARCHAR(9) NOT NULL,
  K_KOTA VARCHAR(9) NOT NULL,
  NAMA VARCHAR(50) NULL,
  ALAMAT VARCHAR(60) NULL,
  TELP VARCHAR(15) NULL,
  PRIMARY KEY(K_SEKOLAH, K_PROPINSI, K_NEGARA, K_KOTA),
  FOREIGN KEY(K_KOTA, K_NEGARA, K_PROPINSI)
    REFERENCES      SIAKAD.KOTA(K_KOTA,      K_NEGARA,
K_PROPINSI)
);
```

Syntax SQL diatas adalah DDL untuk membuat tabel sekolah dengan kunci utama adalah K_SEKOLAH, K_KOTA, K_PROPINSI dan K_NEGARA yang memiliki referensi pada tabel kota, propinsi dan negara.

10. Tabel Kerja

```
CREATE TABLE SIAKAD.KERJA (
  K_KERJA VARCHAR(9) NOT NULL,
  CONTENT VARCHAR(45) NULL,
  PRIMARY KEY(K_KERJA)
);
```

Syntax SQL diatas adalah DDL untuk membuat tabel kerja dengan kunci utama adalah K_KERJA.

11. Tabel Mahasiswa

```
CREATE TABLE SIAKAD.MAHASISWA (
  NIM VARCHAR(15) NOT NULL,
  K_JENJANG VARCHAR(9) NOT NULL,
  K_JURUSAN VARCHAR(9) NOT NULL,
  K_FAKULTAS VARCHAR(9) NOT NULL,
  K_PROGRAM_STUDI VARCHAR(9) NOT NULL,
  K_SELEKSI VARCHAR(9) NOT NULL,
  NAMA VARCHAR(60) NULL,
  ANGKATAN INTEGER NULL,
  PRIMARY KEY(NIM),
  FOREIGN KEY(K_SELEKSI)
    REFERENCES SIAKAD.SELEKSI(K_SELEKSI),
  FOREIGN  KEY(K_PROGRAM_STUDI,  K_FAKULTAS,  K_JURUSAN,
K_JENJANG)
    REFERENCES      SIAKAD.PROGRAM_STUDI(K_PROGRAM_STUDI,
K_FAKULTAS, K_JURUSAN, K_JENJANG)
);
```

Syntax SQL diatas adalah DDL untuk membuat tabel mahasiswa dengan kunci utama adalah NIM yang memiliki referensi pada tabel program studi, jenjang, jurusan dan fakultas.

12. Tabel Mahasiswa Biodata

```
CREATE TABLE SIAKAD.MAHASISWA_BIODATA (
  NIM VARCHAR(15) NOT NULL,
  K_KOTA VARCHAR(9) NOT NULL,
  K_NEGARA VARCHAR(9) NOT NULL,
  K_PROPINSI VARCHAR(9) NOT NULL,
  K_SEKOLAH VARCHAR(9) NOT NULL,
  TMP_LAHIR VARCHAR(60) NULL,
  TGL_LAHIR DATE NULL,
  ALAMAT_ASAL VARCHAR(60) NULL,
  ALAMAT_KOS VARCHAR(60) NULL,
  NO_TELP_RMH VARCHAR(25) NULL,
  NO_TELP_KOS VARCHAR(25) NULL,
  NO_HP VARCHAR(25) NULL,
  PRIMARY KEY(NIM),
  FOREIGN KEY(NIM)
    REFERENCES SIAKAD.MAHASISWA(NIM),
  FOREIGN KEY(K_SEKOLAH, K_PROPINSI, K_NEGARA, K_KOTA)
    REFERENCES    SIAKAD.SEKOLAH(K_SEKOLAH,    K_PROPINSI,
K_NEGARA, K_KOTA)
);
```

Syntax SQL diatas adalah DDL untuk membuat tabel mahasiswa biodata dengan kunci utama adalah NIM yang memiliki referensi pada tabel mahasiswa, sekolah, kota, propinsi dan negara.

13. Tabel Orang Tua Mahasiswa

```
CREATE TABLE SIAKAD.MAHASISWA_ORTU (
  NIM VARCHAR(15) NOT NULL,
  NAMA_AYAH VARCHAR(60) NULL,
  K_KERJA_AYAH VARCHAR(9) NOT NULL,
  NAMA_IBU VARCHAR(60) NULL,
  PENGHASILAN_AYAH DOUBLE NULL,
  K_KERJA_IBU VARCHAR(9) NOT NULL,
  PENGHASILAN_IBU DOUBLE NULL,
  ALAMAT VARCHAR(60) NULL,
  K_KOTA VARCHAR(9) NOT NULL,
  K_PROPINSI VARCHAR(9) NOT NULL,
  K_NEGARA VARCHAR(9) NOT NULL,
  NO_TELP_RMH VARCHAR(25) NULL,
  NO_HP VARCHAR(25) NULL,
  PRIMARY KEY(NIM),
  FOREIGN KEY(NIM)
    REFERENCES SIAKAD.MAHASISWA(NIM),
  FOREIGN KEY(K_KOTA, K_NEGARA, K_PROPINSI)
    REFERENCES SIAKAD.KOTA(K_KOTA, K_NEGARA, K_PROPINSI),
  FOREIGN KEY(K_KERJA_AYAH)
    REFERENCES SIAKAD.KERJA(K_KERJA),
  FOREIGN KEY(K_KERJA_IBU)
```

Syntax SQL diatas adalah DDL untuk membuat tabel orangtua mahasiswa dengan kunci utama adalah NIM yang memiliki referensi pada tabel mahasiswa, kerja, kota, propinsi dan negara

14. Tabel Nilai

```
CREATE TABLE SIAKAD.NILAI (
  K_NILAI VARCHAR(9) NOT NULL,
  ANGKA DOUBLE NULL,
  HURUF VARCHAR(1) NULL,
  MINIMUM DOUBLE NULL,
  MAKSIMUM DOUBLE NULL,
  PRIMARY KEY(K_NILAI)
);
```

Syntax SQL diatas adalah DDL untuk membuat tabel nilai dengan kunci utama adalah K_NILAI

15. Tabel Mata Kuliah

```
CREATE TABLE SIAKAD.MATA_KULIAH (
  ID_MK INTEGER NOT NULL GENERATED ALWAYS AS IDENTITY
(START WITH 1 INCREMENT BY 1),
  K_MK VARCHAR(10) NOT NULL,
  THN_MK INTEGER NOT NULL,
  K_JENJANG VARCHAR(9) NOT NULL,
  K_JURUSAN VARCHAR(9) NOT NULL,
  K_FAKULTAS VARCHAR(9) NOT NULL,
```

```
  K_PROGRAM_STUDI VARCHAR(9) NOT NULL,
  NAMA VARCHAR(100) NOT NULL,
  SKS INTEGER NULL,
  PRIMARY KEY(ID_MK),
  FOREIGN KEY(K_PROGRAM_STUDI, K_FAKULTAS, K_JURUSAN,
K_JENJANG)
    REFERENCES   SIAKAD.PROGRAM_STUDI(K_PROGRAM_STUDI,
K_FAKULTAS, K_JURUSAN, K_JENJANG)
);
```

Syntax SQL di atas adalah DDL untuk membuat tabel mata kuliah dengan kunci utama adalah ID_MK yang memiliki referensi pada tabel program studi, jenjang, jurusan, dan fakultas

16. Tabel KHS

```
CREATE TABLE SIAKAD.KHS (
  NIM VARCHAR(15) NOT NULL,
  THN_AKADEMIK INTEGER NOT NULL,
  ID_MK INTEGER NOT NULL,
  IS_GANJIL CHAR(1) NOT NULL,
  IS_PENDEK CHAR(1) NOT NULL,
  K_NILAI VARCHAR(9) NOT NULL,
  PRIMARY  KEY(NIM,  THN_AKADEMIK,  ID_MK,  IS_GANJIL,
IS_PENDEK),
  FOREIGN KEY(NIM)
    REFERENCES SIAKAD.MAHASISWA(NIM),
  FOREIGN KEY(ID_MK)
    REFERENCES SIAKAD.MATA_KULIAH(ID_MK),
  FOREIGN KEY(K_NILAI)
    REFERENCES SIAKAD.NILAI(K_NILAI)
);
```

Syntax SQL di atas adalah DDL untuk membuat tabel KHS dengan kunci utama adalah NIM, THN_AKADEMIK, ID_MK, IS_GANJIL dan IS_PENDEK yang memiliki referensi pada tabel mahasiswa, mata kuliah dan nilai.

Kode DDL diatas akan di *deploy* pada basis data SIAKAD dengan *schema* DB2ADMIN.

LAMPIRAN C SPESIFIKASI JUMLAH DATA MAHASISWA

Berikut adalah data jumlah mahasiswa berdasarkan Dirjen Pendidikan Tinggi Republik Indonesia per Fakultas dan Program Studi:

1. Fakultas Peternakan

Prodi	Jenjang	Jumlah Mahasiswa
Peternakan	S1	2287
Ilmu Ternak	S2	37
Ilmu Ternak	S3	82

2. Fakkultas Ekonomi Bisnis

Prodi	Jenjang	Jumlah Mahasiswa
Profesi Akuntan	Profesi	56
Ekonomi Islam	S1	277
Ekonomi Pembangunan	S1	1325
Manajemen	S1	1518
Ilmu Ekonomi	S2	84
Manajemen	S2	221
Ilmu Ekonomi	S3	53
Ilmu Manajemen	S3	151

3. Fakultas Hukum

Prodi	Jenjang	Jumlah Mahasiswa
Ilmu Hukum	S1	2388
Ilmu Hukum	S2	64
Kenotariatan	S2	261
Ilmu Hukum	S3	45

4. Fakultas Ilmu Administrasi

Prodi	Jenjang	Jumlah Mahasiswa
Administrasi Pendidikan	S1	0
Akuntansi	S1	1258
Ilmu Administrasi Negara	S1	2788
Ilmu Administrasi Niaga	S1	2697
Ilmu Gizi	S1	543
Ilmu Perpustakaan	S1	50
Pariwisata	S1	23
Perpajakan	S1	928
Akuntansi	S2	305
Ilmu Administrasi Negara	S2	156
Ilmu Administrasi Niaga	S2	102

MM Pendidikan Tinggi	S2	28
Ilmu Administrasi	S3	196
Ilmu Akuntansi	S3	94

5. Fakultas Ilmu Budaya

Prodi	Jenjang	Jumlah Mahasiswa
Antropologi	S1	178
Pendidikan Bahasa Inggris	S1	442
Pendidikan Bahasa Dan Sastra Indonesia	S1	239
Pendidikan Bahasa Jepang	S1	245
Sastra Cina	S1	129
Sastra Inggris	S1	979
Sastra Jepang	S1	704
Sastra Perancis	S1	243
Seni Rupa Murni	S1	74
Ilmu Linguistik	S2	26

6. PTIIK/Fakultas Ilmu Komputer

Prodi	Jenjang	Jumlah Mahasiswa
Pendidikan Teknologi Informasi	S1	0
Sistem Informasi	S1	831
Teknik Informatika	S1	3202
Ilmu Komputer	S2	0

7. Fakultas Ilmu Sosial dan Ilmu Politik

Prodi	Jenjang	Jumlah Mahasiswa
Ilmu Hubungan Internasional	S1	1168
Ilmu Komunikasi	S1	1720
Ilmu Pemerintahan	S1	589
Ilmu Politik	S1	600
Psikologi	S1	1197
Sosiologi	S1	780
Ilmu Komunikasi	S2	39
Ilmu Sosial	S2	54
Sosiologi	S2	8
Ilmu Sosiologi	S3	15

8. Fakultas Kedokteran

Prodi	Jenjang	Jumlah Mahasiswa
Profesi Dokter	Profesi	519
Profesi Dokter Gigi	Profesi	201
Profesi Ners	Profesi	104

Farmasi	S1	310
Ilmu Keperawatan	S1	780
Kebidanan	S1	353
Kedokteran	S1	974
Pendidikan Dokter Gigi	S1	426
Ilmu Biomedik	S2	114
Kebidanan	S2	114
Keperawatan	S2	117
Manajemen Rumah Sakit	S2	59
Ilmu Kedokteran	S3	231
Anestesiologi Dan Reanimasi	Sp-1	25
Ilmu Bedah	Sp-1	44
Ilmu Bedah Orthopaedi Dan Traumatologi	Sp-1	32
Ilmu Kebidanan Dan Penyakit Kandungan	Sp-1	40
Ilmu Kesehatan Anak	Sp-1	51
Ilmu Kesehatan Kulit Dan Kelamin	Sp-1	16
Ilmu Penyakit Dalam	Sp-1	45
Ilmu Penyakit Jantung Dan Pembuluh Darah	Sp-1	37
Ilmu Penyakit Mata	Sp-1	41
Ilmu Penyakit Paru	Sp-1	27
Ilmu Penyakit Syaraf	Sp-1	26
Ilmu Penyakit Tht	Sp-1	17
Patologi Klinik	Sp-1	32
Radiologi	Sp-1	12
Urologi	Sp-1	1

9. Fakultas Kedokteran Hewan

Prodi	Jenjang	Jumlah Mahasiswa
Pendidikan Dokter Hewan	S1	908

10. Fakultas Matematika dan Pengetahuan Alam

Prodi	Jenjang	Jumlah Mahasiswa
Biologi	S1	349
Fisika	S1	667
Kimia	S1	425
Matematika	S1	409
Statistika	S1	470
Teknik Geofisika	S1	30
Biologi	S2	49
Fisika	S2	41
Kimia	S2	50
Matematika	S2	51

Statistika	S2	7
Biologi	S3	41

11. Fakultas Pertanian

Prodi	Jenjang	Jumlah Mahasiswa
Agribisnis	S1	2145
Agroteknologi	S1	3052
Agribisnis	S2	0
Agronomi	S2	68
Ekonomi Pertanian	S2	41
Pengelolaan Tanah Dan Air	S2	20
Ilmu Pertanian	S3	208

12. Fakultas Ilmu Perikanan dan Kelautan

Prodi	Jenjang	Jumlah Mahasiswa
Budidaya Perikanan	D1	23
Agrobisnis Perikanan	S1	793
Budidaya Perairan	S1	732
Ilmu Kelautan	S1	765
Manajemen Sumber Daya Perairan	S1	827
Pemanfaatan Sumber Daya Perikanan	S1	708
Teknologi Hasil Perikanan	S1	986
Budidaya Perairan	S2	94
Ilmu Perikanan dan Kelautan	S3	69

13. Fakultas Teknik

Prodi	Jenjang	Jumlah Mahasiswa
Arsitektur	S1	773
Perencanaan Wilayah Dan Kota	S1	568
Teknik Elektro	S1	893
Teknik Industri	S1	1318
Teknik Kimia	S1	52
Teknik Mesin	S1	810
Teknik Pengairan	S1	667
Teknik Sipil	S1	699
Arsitektur Lingkungan Binaan	S2	31
Teknik Elektro	S2	145
Teknik Mesin	S2	44
Teknik Pengairan	S2	58
Teknik Sipil	S2	151
Ilmu Teknik Mesin	S3	124
Ilmu Teknik Sipil	S3	59

14. Fakultas Teknologi Pertanian

Prodi	Jenjang	Jumlah Mahasiswa
Bioteknologi	S1	242
Teknik Lingkungan	S1	329
Teknik Pertanian	S1	835
Teknologi Industri Pertanian	S1	1270
Teknologi Pangan	S1	1045
Keteknikan Pertanian	S2	4
Teknologi Hasil Pertanian	S2	44
Teknologi Industri Pertanian	S2	20
Teknologi Industri Pertanian	S3	5

15. Pascasarjana

Prodi	Jenjang	Jumlah Mahasiswa
Kajian Wanita	S2	5
Pengelolaan Sumberdaya Lingkungan dan Pembangunan	S2	37
Wawasan Pertahanan Nasional	S2	20

16. Vokasi

Prodi	Jenjang	Jumlah Mahasiswa
Kesekretariatan	D3	434
Usaha Perjalanan Wisata	D3	154