Rancang Bangun *Real-Time Business Intelligence* Untuk Subjek Kegiatan Akademik pada Universitas Menggunakan *Change Data Capture*

Stephanie Pamela Adithama
Program Studi Teknik Informatika, Universitas Atma Jaya Yogyakarta
Jl. Babarsari 43, Yogyakarta 55281, Indonesia
E-mail: *stephanie_pamela@staff.uajy.ac.id*

Abstract. *The running of academic activities in university continuously adds more data to the existing operational system. The data are not ready for the university strategic decision making, preparing reports for accreditation purposes and academic units. Real-time business intelligence application using data warehouse can become a solution for data analysis. The process of creating a data warehouse includes designing data warehouse, retrieving academic data from multiple data sources, extracting, transforming, loading (ETL) process, creating cube; and generating report. ETL processes are conducted by using a Pull Change Data Capture approach so that data changes during a certain period can be transferred in real-time. The higher the frequency of data change requests brings us closer to real-time and requires less time than loading all the data.*
Keywords: *real-time, business intelligence, data warehouse, academic, change data capture*

Abstrak. *Kegiatan akademik di universitas berjalan terus menerus dan semakin menambah banyak data pada sistem operasional yang sudah ada. Data tersebut masih belum dapat dimanfaatkan oleh pihak universitas dalam pengambilan keputusan strategis, pembuatan laporan untuk keperluan akreditasi dan unit-unit akademik. Aplikasi real-time business intelligence menggunakan data warehouse menjadi solusi untuk analisa data. Proses pembuatan data warehouse meliputi perancangan data warehouse; pengambilan data akademik dari sumber data; proses extraction, transformation, loading (ETL); pembuatan cube; dan pembuatan laporan. Proses ETL dilakukan menggunakan pendekatan Change Data Capture Pull agar perubahan data selama periode tertentu dapat dipindahkan secara real-time. Semakin tinggi frekuensi permintaan perubahan data akan semakin mendekati real-time dan semakin membutuhkan waktu yang singkat dibandingkan dengan me-load semua data.*
Kata Kunci: *real-time, business intelligence, data warehouse, akademik, change data capture*

1. Pendahuluan

Di universitas pada umumnya telah tersedia sistem yang digunakan untuk mengelola transaksi akademik di setiap fakultas dan pada unit-unit akademik. Penelitian ini menggunakan studi kasus di Universitas Atma Jaya Yogyakarta (UAJY), UAJY telah memiliki sistem yang digunakan untuk memenuhi kebutuhan transaksional data mahasiswa, dosen, kelas, nilai, jadwal, mata kuliah, krs, transkrip, lulusan, yang berhubungan dengan bidang akademik. Proses pembuatan keputusan dengan melakukan analisa terhadap data yang dihasilkan dari sistem-sistem transaksional ini, tentu saja kurang efisien karena membutuhkan waktu yang cukup lama, dan tidak cukup akurat.

Akreditasi untuk institusi pendidikan tinggi dilakukan oleh Badan Akreditasi Nasional Perguruan Tinggi (BAN-PT). Persyaratan untuk mendapatkan akreditasi dari BAN-PT, institusi perguruan tinggi diwajibkan untuk mengisi borang akreditasi dan untuk dapat mengisi borang tersebut, diperlukan berbagai macam data dan informasi. Kebutuhan akan data dan informasi,

bisa dipenuhi apabila perguruan tinggi telah memanfaatkan sistem informasi dan menggunakan *Business Intelligence* (BI) (Wilarso,2008).

BI merupakan salah satu bentuk implementasi yang mampu menjawab kebutuhan untuk menganalisis masalah-masalah dalam pengambilan keputusan. Secara ringkas, BI diartikan sebagai pengetahuan yang didapatkan dari hasil analisis data yang diperoleh dari kegiatan suatu organisasi (Kusnawi, 2008). BI meliputi topik-topik seperti *data warehousing*, *Online Analytical Processing* (OLAP), *Extract Transform Loading* (ETL), *data mining*, dan multidimensionalitas (Turban et.al., 2004). Kesuksesan bisnis membutuhkan analisis data dilakukan secara *real-time* (Azvine et.al., 2006). *Real-time* BI akan menjelajahi *data warehouse* yang bervolume besar dan secara cepat mendatangkan data dalam operasi bisnis. *Real-time* BI mengoptimalkan proses pengambilan keputusan dengan mengurangi untuk menghilangkan latensi dan menyediakan data kontekstual yang kaya dan secara langsung dapat ditindaklanjuti (Sandu, 2008; Botan et.al., 2009).

Di UAJY perubahan data mahasiswa, dosen, kelas, nilai, jadwal, mata kuliah, krs, transkrip dilakukan setiap semester. Perubahan data lulusan dilakukan setiap bulan ketika yudisium dan empat bulan sekali ketika ada wisuda. Perubahan data calon mahasiswa baru dapat terjadi setiap waktu karena proses penerimaan mahasiswa baru dilakukan melalui beberapa jalur, yang sudah dimulai sejak bulan September sampai Juli. Proses penerimaan yang berkelanjutan ini dan penambahan data pendaftar yang tidak dapat diprediksi menyebabkan kebutuhan data-data akademik khususnya data calon mahasiswa baru perlu segera ditindaklanjuti untuk analisis dan pembuatan laporan. Proses ETL tradisional dijalankan secara periodik pada interval tertentu, contohnya bulanan atau mingguan. Pendekatan ini telah dilakukan oleh perusahaan bertahun-tahun, namun kondisi sekarang ini membutuhkan cara ETL baru yang lebih efisien dan *real-time* (Attachmate Corp, 2005; Attunity, 2006). Solusi (*Change Data Capture*) CDC didesain dan diintegrasikan dengan proses ETL untuk memaksimalkan efisiensi ETL, meminimalisasi penggunaan sumber daya dengan memindahkan perubahan data saja, dan meminimalisasi latensi dalam pengiriman perubahan data pada konsumen (Jörg & Deßloch, 2008).

Penelitian ini melakukan perancangan *data warehouse* dan pengembangan sistem *real-time* BI berbasis *web* menggunakan pendekatan CDC *Pull*. Konsep *real-time* diwujudkan dengan mengintegrasikan CDC dengan *tool* ETL menggunakan model CDC *Pull* yang secara periodik meminta perubahan data, setiap waktu menerima sekumpulan *record* yang merepresentasikan semua perubahan yang ditangkap sejak siklus permintaan terakhir. Skenario ini mirip dengan ETL tradisional, perbedaannya skenario ini menangkap dan memindahkan hanya data yang berubah saja selama periode tertentu.

2. Tinjauan Pustaka

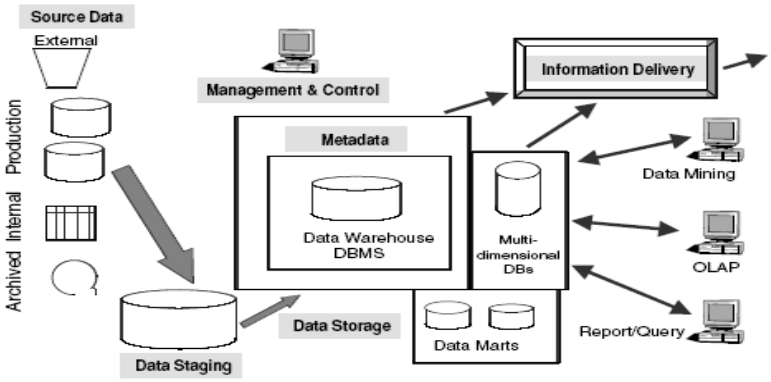
2.1. *Data Warehouse*

Menurut pelopor konsep dan istilah *data warehouse*, William Inmon, *data warehouse* adalah sebuah koleksi data yang berorientasi subjek, terintegrasi, *non-volatile*, dan *time-variant* dalam rangka mendukung keputusan-keputusan manajemen (Inmon, 2005).

Ponniah (2010) menyebutkan karakteristik dari *data warehouse* adalah sebagai berikut: (1) Berorientasi Subjek: *Data warehouse* didesain untuk menganalisa data berdasarkan subjek-subjek tertentu dalam organisasi, bukan pada proses atau fungsi aplikasi tertentu. (2) Terintegrasi: Sumber data yang ada dalam *data warehouse* tidak hanya berasal dari data operasional (*internal source*) tetapi juga berasal dari data di luar sistem (*external source*). (3) *Time-variant*: Sistem operasional mengandung data yang bernilai sekarang sedangkan data dalam *data warehouse* mengandung data tidak hanya data terkini tetapi juga data masa lampau. (4) *Non-volatile*: Pada *data warehouse* hanya ada dua kegiatan memanipulasi data yaitu *loading* data (mengambil data) dan akses data.

Komponen-komponen *data warehouse* menurut Ponniah (2010) digambarkan pada Gambar 1. Komponen sumber data berada di sebelah kiri. Komponen *data staging* menyediakan tempat dengan satu set fungsi untuk membersihkan, mengubah, menggabungkan,

mengkonversi, mencegah duplikasi data, dan menyiapkan data sumber untuk penyimpanan dan penggunaan dalam *data warehouse*, disebut *Extraction, Transformation, and Loading* (ETL). Di tengah, terdapat komponen penyimpanan data yang mengelola *data warehouse*. Komponen *information delivery* berada di sebelah kanan, menyediakan informasi dari *data warehouse* bagi pengguna.



Gambar 1. Komponen *Data Warehouse* (Ponniah, 2010)

Pembuatan *data warehouse* didasarkan pada model data multidimensi. Model ini menampilkan data dalam bentuk kubus (*cube*), sebuah data dapat dipandang dari berbagai sudut. Komponen model multidimensional yang umum ditemukan dalam perancangan *data warehouse* (Prasetyo et.al., 2010): (1) Dimensi: kategori yang independen dari multidimensional basis data, mengandung *item* yang digunakan sebagai kriteria *query*. (2) Tabel Fakta: Mempunyai dua tipe kolom, yaitu kolom yang menyimpan nilai-nilai numerik (*measure*) dan kolom yang *menyimpan foreign key* yang mengacu ke tabel dimensi. (3) *Measure*: *C*erminan dari fakta dan mengandung informasi kolom bertipe numerik yang akan dianalisa. (4) Hirarki: merupakan bentuk kesatuan dari dimensi. Sebuah dimensi bisa terbentuk dari multilevel, yang mempunyai *parent-child relationship*.

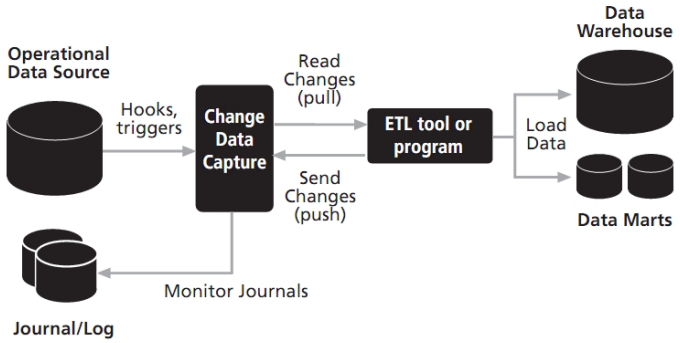
Model dimensional yang sering digunakan pada *data warehouse* (Prasetyo et.al., 2010): (1) *Star schema*, terdiri dari satu atau lebih tabel fakta dan satu atau lebih tabel dimensi. Tabel fakta merupakan pusat dari *star schema*, karena fungsinya sebagai pengikat dari tabel-tabel dimensi. (2) *Snowflake schema,* merupakan pengembangan dari *star schema*, tabel dimensi dinormalisasi secara sebagian atau keseluruhan untuk mengurangi nilai duplikat pada tabel.

2.2. *Change Data Capture* (CDC)

CDC adalah pendekatan inovasi untuk integrasi data, berdasarkan identifikasi, menangkap, dan mengirimkan perubahan yang dibuat oleh data sumber. Dengan memproses hanya perubahannya, CDC membuat proses integrasi data lebih efisien dan mengurangi biaya dengan mengurangi latensi (Attunity, 2006). CDC harus diintegrasikan dengan *tool* ETL sehingga proses ETL dapat efisien. Integrasi CDC dengan *tool* ETL yang ada menyediakan pendekatan terintegrasi untuk mengurangi jumlah informasi yang dikirimkan sambil meminimalisasi kebutuhan sumber daya dan memaksimalkan kecepatan dan efisiensi (Tank et.al., 2010).

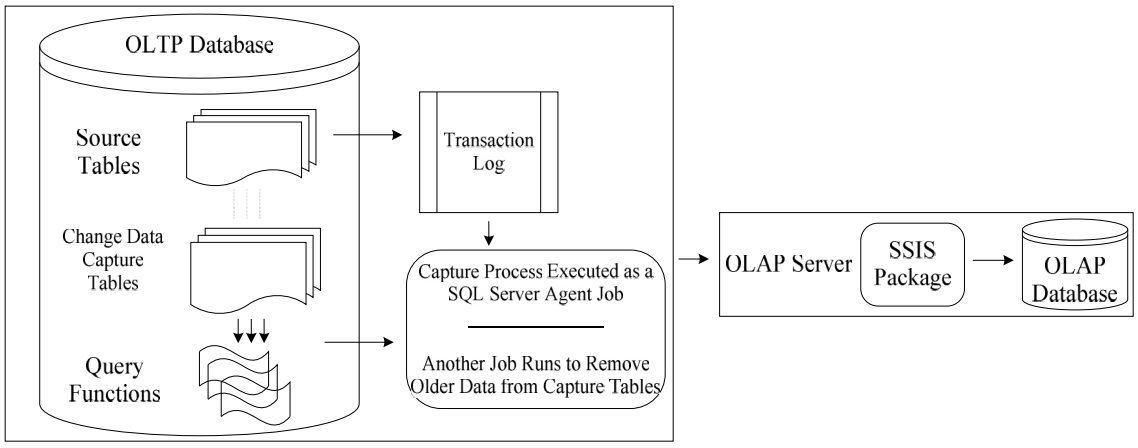
Terdapat dua model skenario CDC pada Gambar 2 yang terintegrasi dengan *tool* ETL (Attachmate Corp, 2005): (1) Model CDC *Pull*: *Tool* ETL secara periodik meminta perubahan data, setiap waktu menerima sekumpulan *record* yang merepresentasikan semua perubahan yang ditangkap sejak siklus permintaan terakhir. Permintaan perubahan data dapat dalam frekuensi tinggi atau rendah. Skenario ini mirip dengan ETL tradisional, perbedaannya skenario ini menangkap dan memindahkan hanya data yang berubah saja. (2) Model CDC *Push*: Mekanisme pengiriman mengirim perubahan data ke *tool* ETL segera setelah perubahan terjadi. Metode ini membutuhkan *tool* ETL menggunakan *listeners* yang menunggu *event* perubahan

dan *publisher* yang digunakan untuk mengirim dan memberikan notifikasi perubahan secara *real-time*.



Gambar 2. Integrasi CDC Pada ETL (Attachmate Corp, 2005)

Pada SQL Server 2008, CDC menangkap dan merekam aktivitas *insert*, *update*, dan *delete* pada *database* OLTP dan menyimpannya dalam bentuk yang mudah digunakan oleh aplikasi, seperti SSIS *package*. *Package* ini digunakan untuk mengambil data dan menyimpannya pada *server* OLAP. Gambar 3 merepresentasikan gambaran komponen utama pada arsitektur CDC di SQL Server 2008. Diagram ini dibagi menjadi dua bagian, bagian atas merepresentasikan *server* OLTP dan bagian bawah merepresentasikan *server* OLAP.



Gambar 3. Arsitektur CDC di SQL Server 2008 (McGehee, 2008)

Komponen-komponen CDC pada Gambar 3 dijelaskan sebagai berikut (McGehee, 2008): (1) Tabel sumber: ketika SQL Server pertama kali di*install*, secara *default* CDC dimatikan sehingga langkah pertama adalah mengaktifkan CDC pada level *database*, kemudian CDC harus diaktifkan pada level tabel. Setiap tabel yang CDC-nya aktif disebut dengan tabel sumber. (2) Tabel CDC: Setiap tabel sumber yang CDC-nya aktif, diciptakan tabel CDC yang berhubungan, yang digunakan untuk menyimpan perubahan yang dibuat di tabel sumber, bersama dengan beberapa metadata yang digunakan untuk menelusuri perubahan. (3) Fungsi *Query* CDC: setiap tabel sumber yang CDC-nya aktif, beberapa fungsi *query* CDC diciptakan untuk mengakses tabel CDC. (4) *Capture* dan *Cleanup Jobs*: dua SQL Server Agent *jobs* juga diciptakan, *Capture job* secara umum berjalan terus menerus dan digunakan untuk memindahkan perubahan data ke tabel CDC dari *transaction log*. *Cleanup job* dijalankan terjadwal untuk menghapus data lama pada tabel CDC sehingga tidak membengkak terlalu besar.

Cara kerja CDC sesuai dengan Gambar 3, adalah (McGehee, 2008): ketika ada kegiatan *insert*, *update*, atau *delete* yang terjadi pada tabel sumber, perubahan ini ditulis pada *transaction log* pada *database*. Hal ini normal, terjadi pula walaupun CDC tidak diaktifkan. Perbedaannya,

ketika CDC diaktifkan pada tabel sumber, SQL Server *Agent Capture job* membaca perubahan pada *transaction log* dan memindahkannya ke tabel CDC yang tepat. *Insert* dan *delete* masing-masing menghasilkan satu baris pada tabel CDC, dan *update* menghasilkan dua baris: satu baris data sebelum dan satu baris data setelah perubahan. Perubahan ini terus diakumulasikan pada tabel CDC sampai dihapus oleh *Cleanup Job*. Perubahan data dari tabel CDC diekstrak dengan menjalankan statemen yang sesuai menggunakan fungsi *query* yang relevan, yang mengizinkan untuk mengekstrak semua perubahan yang dibuat pada tabel sumber sejak terakhir kali SSIS *package* dieksekusi, dan kemudian memindahkannya ke *database* pada *server* OLAP.

3. Metodologi Penelitian

Metodologi yang digunakan dalam pembangunan apikasi *real-time* BI untuk subjek kegiatan akademik pada universitas terdiri dari empat tahap utama yaitu: (1) Analisis. Pada tahap ini dilakukan analisis kebutuhan dan menganalisis data-data yang dibutuhkan untuk merancang *data warehouse*. Dilakukan dengan melakukan pengamatan pada Buku Pedoman Akreditasi Institusi Perguruan Tinggi, Buku Pedoman Akreditasi Program Studi Sarjana, dan permintaan-permintaan kebutuhan data akademik dari fakultas-fakultas, rektorat, dan unit-unit. (2) Perancangan. Pada tahap ini dilakukan perancangan *data warehouse* yaitu: merancang *star* skema, merancang tabel dimensi dan tabel fakta, menentukan pemetaan tabel dan kolom pada data sumber yang akan diambil. (3) Pengkodean. Pembangunan sistem dilakukan melalui beberapa tahap, yaitu: proses ETL dengan CDC dan *loading* data sumber ke *staging* area, membangun tabel dimensi dan tabel fakta pada *data warehouse*, *loading* data dari *staging* area ke *data warehouse*, membangun *cube*, menambahkan *measure*, mendefinisikan kalkulasi, membuat laporan BI berbasis *web*. (4) Pengujian. Pada tahap ini dilakukan proses pengujian fungsionalitas sistem yang telah dibangun.

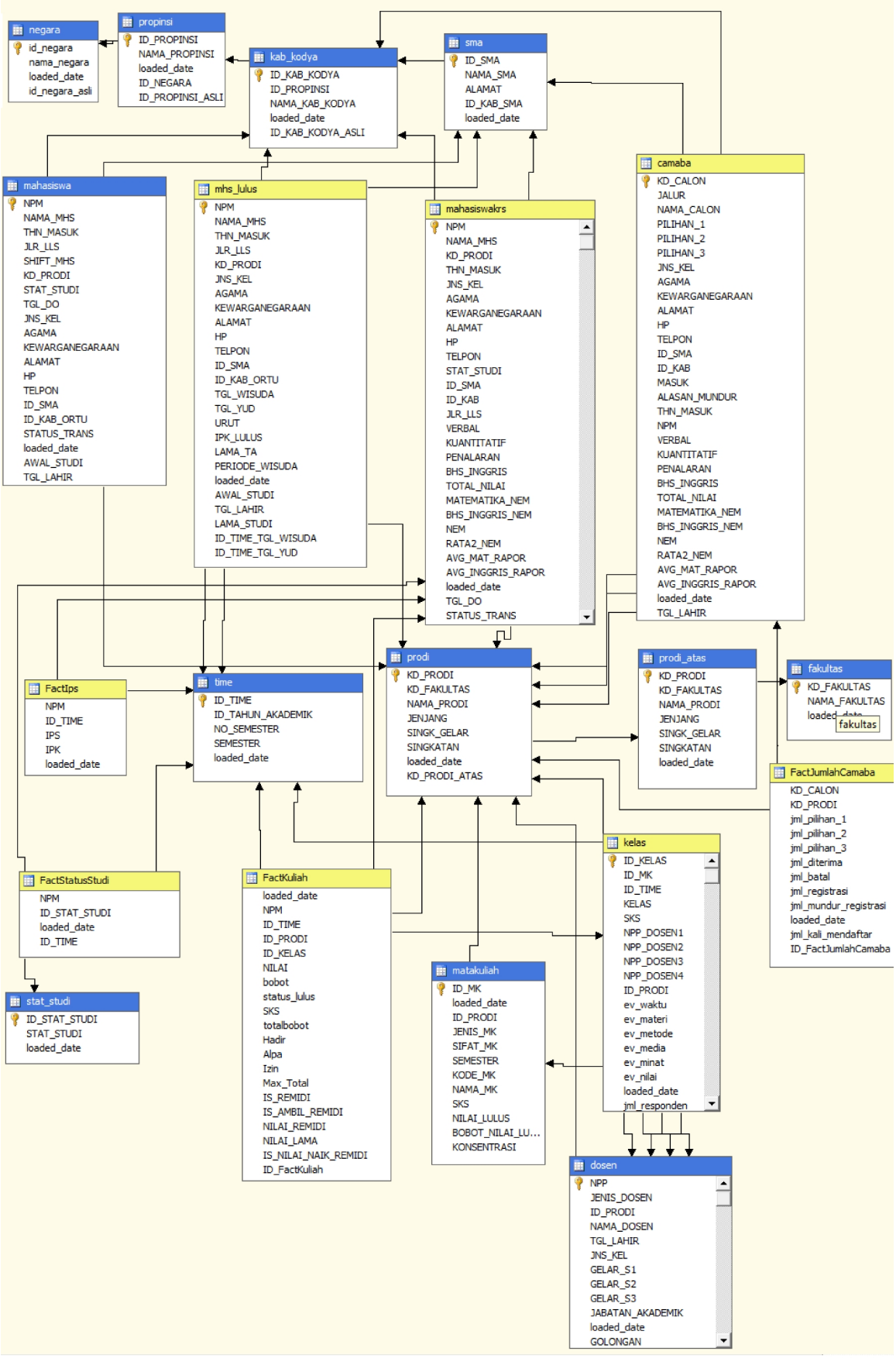
4. Pembahasan dan Pengujian

4.1. Pembahasan

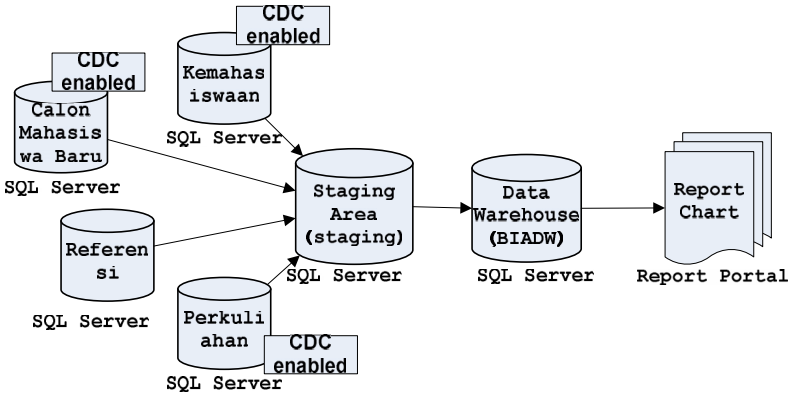
Analisis terhadap sumber data dilakukan untuk mengetahui informasi yang harus disediakan oleh sistem, fungsi-fungsi yang dapat ditangani seperti: (1) Melihat profil jumlah dan kinerja calon mahasiswa dilihat dari berbagai dimensi. (2) Melihat profil jumlah dan kinerja lulusan dilihat dari berbagai dimensi. (3) Melihat profil, jumlah, dan kinerja semua mahasiswa dilihat dari berbagai dimensi. (4) Melihat profil perkuliahan dari berbagai dimensi. (5) Melihat profil transaksi status studi semua mahasiswa dilihat dari berbagai dimensi. Dari fungsi-fungsi tersebut dibuat sebuah skema model data. Gambar 4 merupakan *star* dan *snowflake* skema yang digunakan untuk membangun *data warehouse.*

Pendekatan yang digunakan dalam penelitian ini menggunakan model CDC *Pull* yang mirip dengan ETL tradisional, namun proses hanya dilakukan untuk data sumber yang berubah saja, bukan keseluruhan data sumber. SSIS *package* akan meminta perubahan secara periodik yaitu akan dieksekusi per menit, setiap hari. Sekumpulan *record* yang diterima merepresentasikan perubahan yang ditangkap sejak permintaan perubahan terakhir. CDC dipasang untuk menangkap perubahan yang terjadi pada *database* sumber dan pada tabel-tabel di *database* sumber yang terkait. Dalam proses pembuatan *data warehouse*, komponen-komponen diaplikasikan menjadi tahapan seperti model pada Gambar 5.

Ketika SQL Server 2008 pertama kali di*install*, CDC dimatikan secara *default* sehingga langkah pertama yang harus dilakukan adalah mengaktifkan CDC pada level *database* dan pada level tabel. *Database-database* sumber yaitu *database* perkuliahan, kemahasiswaan, dan calon mahasiswa baru diaktifkan CDC-nya. Ketika sudah berhasil diaktifkan, pada *system tables* akan ditemukan beberapa tabel cdc yang otomatis ditambahkan, yang berisi semua perubahan pada tabel sumber. Pada tabel CDC tersebut terdapat kolom-kolom yang sama persis dengan tabel sumbernya, ditambah dengan lima kolom baru yaitu: **__$start_lsn , __$end_lsn , __$seqval , __$operation , __$update_mask**. Kolom **__$operation** merupakan kolom yang penting karena mengandung nilai yang berhubungan dengan operasi DML, yaitu: Delete = 1, Insert = 2, nilai sebelum Update = 3, nilai sesudah Update = 4.

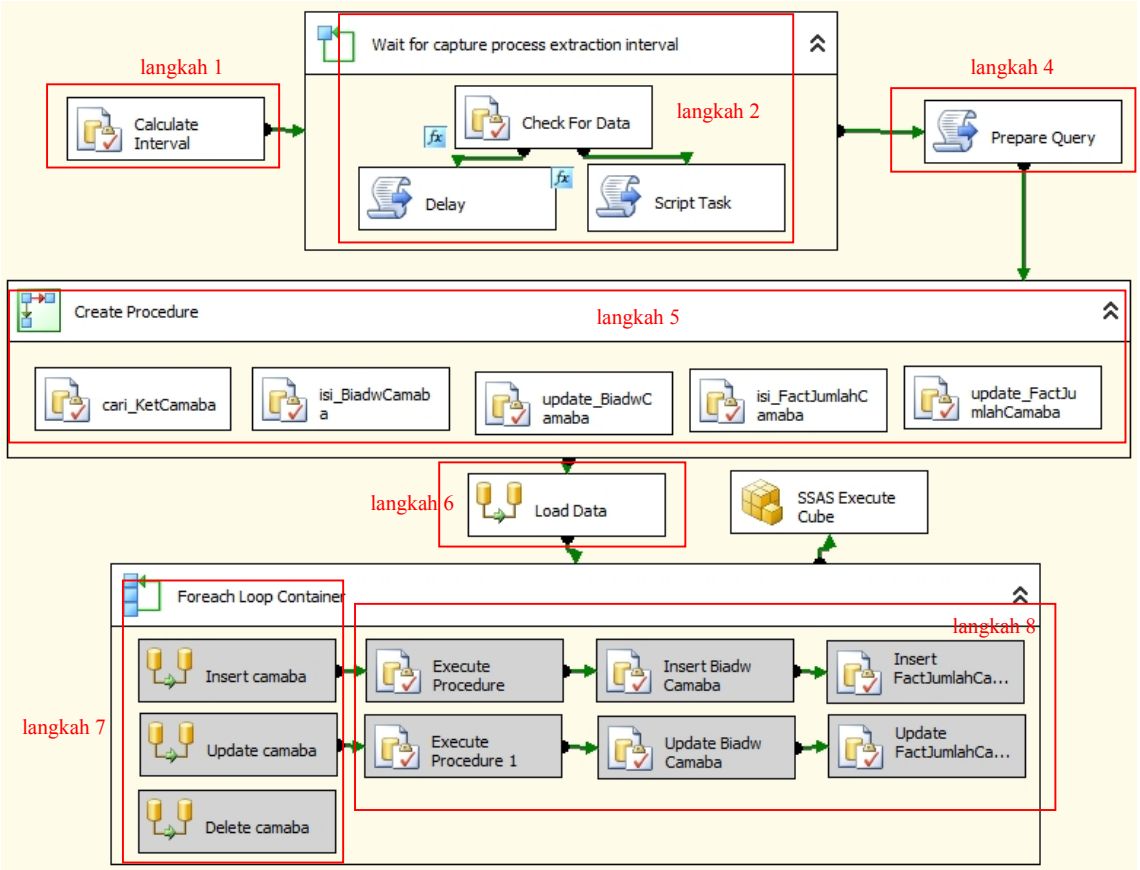


Gambar 4. *Star* dan *Snowflake* Skema BI Akademik



Gambar 5. Rancangan Tahap-Tahap Pembangunan

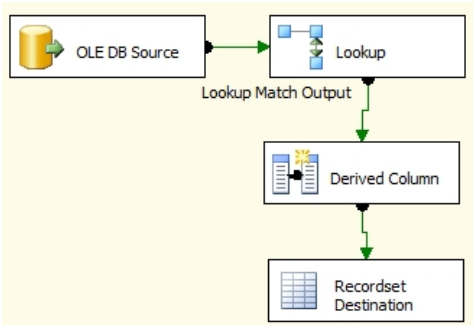
Control flow pada Gambar 6 merupakan cuplikan *flow* untuk menangani calon mahasiswa baru, yang akan memindahkan perubahan pada tabel **MHS_PENDAFTAR** dari *database* sumber menuju ke tabel **camaba** di *database staging* dan tabel **camaba** di *data warehouse*.



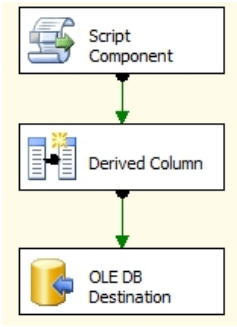
Gambar 6. *Control Flow* pada SSIS *Package*

Langkah-langkah untuk membuat *control flow* pada *SSIS package* yang terintegrasi dengan CDC secara umum adalah sebagai berikut: (1) Menentukan interval perubahan data, menghitung interval waktu mulai dan waktu selesai untuk perubahan pada sumber data yang ingin diambil. Studi kasus ini menggunakan interval tetap dan berasumsi bahwa *package* dijalankan setiap hari tanpa pengecualian. Waktu mulai untuk interval ini adalah waktu sekarang dikurangi satu menit. Waktu selesai untuk interval ini adalah waktu sekarang, yang

nanti akan dipicu oleh SQL Server *Agents Jobs* yang akan menjalankan *package* setiap menit. (2) Menentukan kesiapan data pada interval yang telah dipilih sebelumnya, diperlukan karena data yang tidak sinkron akan menyebabkan kegagalan pada pemrosesan perubahan data. (3) Membuat *table-valued function* untuk mengambil perubahan data, hanya perlu diciptakan satu kali, dan berfungsi untuk mengakses tabel CDC, untuk mengambil perubahan data. (4) Mempersiapkan *query* untuk mengambil perubahan data, menggunakan *Script Task* untuk membangun statemen SQL yang akan digunakan untuk melakukan *query* untuk mengambil perubahan. Statemen SQL ini akan menggunakan fungsi *table-valued function* yang telah dibuat pada langkah sebelumnya. (5) Membuat prosedur, menambahkan *Sequence Container* dan menambahkan *Execute SQL Task* ke dalamnya. Prosedur-prosedur ini merupakan prosedur untuk transformasi dan pembersihan data. (6) Mengambil perubahan data, menambahkan *Data Flow task* seperti pada Gambar 7, yang akan mengambil *record-record* perubahan data, melakukan beberapa perubahan, dan akan menyimpannya dalam *recordset*, sebelum dapat digunakan pada langkah selanjutnya. (7) Memisahkan operasi *insert*, *update*, *delete*, dan memprosesnya pada tabel tujuan. Menambahkan *Data Flow task* seperti pada Gambar 8 untuk masing-masing operasi. *Data Flow task* ini akan mengambil nilai-nilai pada *variable package*, melakukan beberapa perubahan, dan akan melakukan operasi *insert*, *update*, atau *delete* pada *database* tujuan. *Script Component* ini akan mengambil nilai-nilai yang disimpan pada *variable package* dan akan mengeluarkannya melalui *Output Columns*. (8) Mengeksekusi prosedur, menambahkan *Execute SQL Task* untuk mengeksekusi prosedur yang diperlukan untuk melengkapi proses *insert* dan *update*. *Execute SQL Task* juga mengeksekusi prosedur yang melakukan proses *insert* dan *update* ke *data warehouse.* (9) Mengatur *data flow task* yang aktif (*enable*) sesuai dengan operasi yang terjadi pada *record*. Pada operasi *insert*, *data flow task insert* akan aktif (*enable*) jika operasi CDC adalah 'I', yang artinya *record* merupakan operasi *insert* dan belum ada pada tabel tujuan. Pada operasi *update*, *data flow task update* akan aktif (*enable*) jika operasi CDC adalah 'U', yang artinya *record* merupakan operasi *update*. Pada operasi *delete*, *data flow task delete* akan aktif (*enable*) jika operasi CDC adalah 'D', yang artinya *record* merupakan operasi *delete*.



Gambar 7. Desain *Data Flow* Pada Langkah 6



Gambar 8. Desain *Data Flow* Pada Langkah 7

4.2. Pengujian

Konsep *real-time* pada penelitian ini merupakan model *Pull*, dimana *tool* ETL secara periodik meminta perubahan data. Permintaan perubahan data dapat dalam frekuensi tinggi atau rendah, semakin tinggi frekuensi permintaan perubahan data maka akan semakin mendekati *real-time*. Skenario ini menangkap dan memindahkan hanya data yang berubah saja. SSIS *package* akan meminta perubahan secara periodik yaitu akan dieksekusi per menit dan dilakukan oleh SQL Server *Agent Jobs*. Skenario pengujian: (1) Mengatur SQL Server *Agent Job* agar mengeksekusi *package* setiap menit setiap hari. (2) Meng-*insert*-kan 11 data pada menit ke-0, dengan asumsi perubahan pada tabel MHS_PENDAFTAR dengan selang waktu hanya satu menit tidak akan terlalu banyak. Data yang di-*insert*-kan seperti pada Gambar 9, mempunyai isi yang sama persis dengan KD_CALON '10110340' dan '12113008', tetapi mempunyai KD_CALON yang berbeda-beda, agar lebih mudah dalam mencocokkan

kesesuaian data. Selain itu di-*insert*-kan juga data pendukung ke tabel-tabel yang berhubungan. Operasi *insert* tersebut akan dipindahkan secara otomatis dari *transaction log* ke tabel CDC yang secara otomatis terbentuk saat CDC diaktifkan, yaitu tabel cdc.dbo_mhs_pendaftar_CT seperti pada Gambar 10. Pada kolom __$operation nilai 2 menandakan operasi *insert*, dan kolom-kolom setelahnya menyimpan data-data dari baris yang di-*insert*-kan.

```
select * from Mission.dbo.MHS_PENDAFTAR
where KD_CALON in ('12110000','12113008', '10110340','00110000','10110000','20110000','30110000','40110000','5
```

Results | Messages

	KD_CALON	KD_JALUR	ID_ORTU	NM_CALON	PILIHAN_1	PILIHAN_2	PILIHAN_3	JNS_KEL	TMP_LAHIR	TGL_LAHIR	ALAMAT_SURA
1	12113008	11	NULL	RUTH KARLINA	09	10	00	NULL	NULL	NULL	JALAN PEMUDA
2	10110340	5	2762	DWI KURNIAWAN	09	00	00	L	JEMBER	1989-01-14 00:00:00.000	NAYAN MAGUW
3	00110000	5	2762	DWI KURNIAWAN	09	00	00	L	JEMBER	1989-01-14 00:00:00.000	NAYAN MAGUW
4	10110000	5	2762	DWI KURNIAWAN	09	00	00	L	JEMBER	1989-01-14 00:00:00.000	NAYAN MAGUW
5	20110000	5	2762	DWI KURNIAWAN	09	00	00	L	JEMBER	1989-01-14 00:00:00.000	NAYAN MAGUW
6	30110000	5	2762	DWI KURNIAWAN	09	00	00	L	JEMBER	1989-01-14 00:00:00.000	NAYAN MAGUW
7	40110000	5	2762	DWI KURNIAWAN	09	00	00	L	JEMBER	1989-01-14 00:00:00.000	NAYAN MAGUW
8	50110000	5	2762	DWI KURNIAWAN	09	00	00	L	JEMBER	1989-01-14 00:00:00.000	NAYAN MAGUW
9	60110000	5	2762	DWI KURNIAWAN	09	00	00	L	JEMBER	1989-01-14 00:00:00.000	NAYAN MAGUW
10	70110000	5	2762	DWI KURNIAWAN	09	00	00	L	JEMBER	1989-01-14 00:00:00.000	NAYAN MAGUW
11	80110000	5	2762	DWI KURNIAWAN	09	00	00	L	JEMBER	1989-01-14 00:00:00.000	NAYAN MAGUW
12	90110000	5	2762	DWI KURNIAWAN	09	00	00	L	JEMBER	1989-01-14 00:00:00.000	NAYAN MAGUW
13	12110000	11	NULL	RUTH KARLINA	09	10	00	NULL	NULL	NULL	JALAN PEMUDA

Gambar 9. Data Baru di *Database* Sumber Tabel MHS_PENDAFTAR

```
SELECT * FROM Mission.cdc.dbo_mhs_pendaftar_CT
```

Results | Messages

	__$start_lsn	__$end_lsn	__$seqval	__$operation	__$update_mask	KD_CALON	KD_JALUR	ID_ORTU	NM_CALON	PILIHAN_1
1	0x000002D2000001810008	NULL	0x000002D2000001810007	2	0x3FFFFFFFFF	00110000	5	2762	DWI KURNIAWAN	09
2	0x000002D20000018A0008	NULL	0x000002D20000018A0007	2	0x3FFFFFFFFF	10110000	5	2762	DWI KURNIAWAN	09
3	0x000002D2000001930008	NULL	0x000002D2000001930007	2	0x3FFFFFFFFF	20110000	5	2762	DWI KURNIAWAN	09
4	0x000002D20000019C0009	NULL	0x000002D20000019C0008	2	0x3FFFFFFFFF	30110000	5	2762	DWI KURNIAWAN	09
5	0x000002D2000001A50009	NULL	0x000002D2000001A50007	2	0x3FFFFFFFFF	40110000	5	2762	DWI KURNIAWAN	09
6	0x000002D2000001AE0008	NULL	0x000002D2000001AE0007	2	0x3FFFFFFFFF	50110000	5	2762	DWI KURNIAWAN	09
7	0x000002D2000001B70008	NULL	0x000002D2000001B70007	2	0x3FFFFFFFFF	60110000	5	2762	DWI KURNIAWAN	09
8	0x000002D2000001C00008	NULL	0x000002D2000001C00007	2	0x3FFFFFFFFF	70110000	5	2762	DWI KURNIAWAN	09
9	0x000002D2000001C90008	NULL	0x000002D2000001C90007	2	0x3FFFFFFFFF	80110000	5	2762	DWI KURNIAWAN	09
10	0x000002D2000001D20008	NULL	0x000002D2000001D20007	2	0x3FFFFFFFFF	90110000	5	2762	DWI KURNIAWAN	09
11	0x000002D2000001DB0009	NULL	0x000002D2000001DB0007	2	0x3FFFFFFFFF	12110000	11	NULL	RUTH KARLINA	09

Gambar 10. Isi Tabel CDC cdc.dbo_mhs_pendaftar_CT

Selanjutnya, (3) SQL Server *Agent Job* pada menit ke-1 akan langsung mengeksekusi *package* CDC, dan memproses perubahan data yang dibuat sebelumnya. Baris-baris pada tabel CDC cdc.dbo_mhs_pendaftar_CT akan di-*load* ke *database* staging tabel camaba (Gambar 11), dan diteruskan ke *database* biadw tabel camaba (Gambar 12) dan FactJumlahCamaba (Gambar 13), sekaligus memproses *cube*-nya. (4) Pada Gambar 14, *report* pada saat itu juga akan langsung tertampil mahasiswa baru yang tadi ditambahkan.

```
select * from staging.dbo.camaba
where KD_CALON in ('12110000','12113008', '10110340','00110000','10110000','20110000','30110000','40110000','
```

Results | Messages

	KD_CALON	JALUR	NAMA_CALON	PILIHAN_1	PILIHAN_2	PILIHAN_3	JNS_KEL	AGAMA	ID_SMA	ID_KAB	MASUK	ALASAN_MUNDUR
1	00110000	UNGGULAN NEM	DWI KURNIAWAN	09	NULL	NULL	L	Islam	92475	392	09	
2	10110000	UNGGULAN NEM	DWI KURNIAWAN	09	NULL	NULL	L	Islam	92475	392	09	
3	20110000	UNGGULAN NEM	DWI KURNIAWAN	09	NULL	NULL	L	Islam	92475	392	09	
4	30110000	UNGGULAN NEM	DWI KURNIAWAN	09	NULL	NULL	L	Islam	92475	392	09	
5	40110000	UNGGULAN NEM	DWI KURNIAWAN	09	NULL	NULL	L	Islam	92475	392	09	
6	50110000	UNGGULAN NEM	DWI KURNIAWAN	09	NULL	NULL	L	Islam	92475	392	09	
7	60110000	UNGGULAN NEM	DWI KURNIAWAN	09	NULL	NULL	L	Islam	92475	392	09	
8	70110000	UNGGULAN NEM	DWI KURNIAWAN	09	NULL	NULL	L	Islam	92475	392	09	
9	80110000	UNGGULAN NEM	DWI KURNIAWAN	09	NULL	NULL	L	Islam	92475	392	09	
10	90110000	UNGGULAN NEM	DWI KURNIAWAN	09	NULL	NULL	L	Islam	92475	392	09	
11	12110000	REGULER III	RUTH KARLINA	09	10	NULL	-	NA	92448	1	NULL	
12	12113008	REGULER III	RUTH KARLINA	09	10	NULL	-	NA	92448	1	NULL	
13	10110340	UNGGULAN NEM	DWI KURNIAWAN	09	NULL	NULL	L	Islam	92475	392	09	

Gambar 11. Data Baru di *Database Staging* Tabel Camaba

```
select * from biadw.dbo.camaba
where KD_CALON in ('12110000','12113008', '10110340','00110000','10110000','20110000','30110000','401100
```

Results | Messages

	KD_CALON	JALUR	NAMA_CALON	PILIHAN_1	PILIHAN_2	PILIHAN_3	JNS_KEL	AGAMA	ID_SMA	ID_KAB	MASUK	ALASAN_M
1	00110000	UNGGULAN NEM	DWI KURNIAWAN	09	NULL	NULL	L	Islam	92475	392	09	
2	10110000	UNGGULAN NEM	DWI KURNIAWAN	09	NULL	NULL	L	Islam	92475	392	09	
3	10110340	UNGGULAN NEM	DWI KURNIAWAN	09	NULL	NULL	L	Islam	92475	392	09	
4	12110000	REGULER III	RUTH KARLINA	09	10	NULL	-	NA	92448	1	NULL	
5	12113008	REGULER III	RUTH KARLINA	09	10	NULL	-	NA	92448	1	NULL	
6	20110000	UNGGULAN NEM	DWI KURNIAWAN	09	NULL	NULL	L	Islam	92475	392	09	
7	30110000	UNGGULAN NEM	DWI KURNIAWAN	09	NULL	NULL	L	Islam	92475	392	09	
8	40110000	UNGGULAN NEM	DWI KURNIAWAN	09	NULL	NULL	L	Islam	92475	392	09	
9	50110000	UNGGULAN NEM	DWI KURNIAWAN	09	NULL	NULL	L	Islam	92475	392	09	
10	60110000	UNGGULAN NEM	DWI KURNIAWAN	09	NULL	NULL	L	Islam	92475	392	09	
11	70110000	UNGGULAN NEM	DWI KURNIAWAN	09	NULL	NULL	L	Islam	92475	392	09	
12	80110000	UNGGULAN NEM	DWI KURNIAWAN	09	NULL	NULL	L	Islam	92475	392	09	
13	90110000	UNGGULAN NEM	DWI KURNIAWAN	09	NULL	NULL	L	Islam	92475	392	09	

Gambar 12. Data Baru di *Database* biadw Tabel Camaba

```
select * from biadw.dbo.FactJumlahCamaba
where KD_CALON in ('12110000','12113008', '10110340','00110000','10110000','2011000
```

Results | Messages

	KD_CALON	KD_PRODI	jml_pilihan_1	jml_pilihan_2	jml_pilihan_3	jml_diterima	jml_registrasi	loaded_date
1	10110340	09	1	0	0	1	1	2012-11-26 04:17:18.523
2	12113008	09	1	0	0	0	0	2012-11-26 04:18:44.163
3	12113008	10	0	1	0	0	0	2012-11-26 04:18:44.163
4	00110000	09	1	0	0	1	0	2013-04-16 23:35:04.797
5	10110000	09	1	0	0	1	0	2013-04-16 23:35:05.157
6	20110000	09	1	0	0	1	0	2013-04-16 23:35:05.500
7	30110000	09	1	0	0	1	0	2013-04-16 23:35:05.843
8	40110000	09	1	0	0	1	0	2013-04-16 23:35:06.210
9	50110000	09	1	0	0	1	0	2013-04-16 23:35:06.537
10	60110000	09	1	0	0	1	0	2013-04-16 23:35:06.850
11	70110000	09	1	0	0	1	0	2013-04-16 23:35:07.180
12	80110000	09	1	0	0	1	0	2013-04-16 23:35:07.527
13	90110000	09	1	0	0	1	0	2013-04-16 23:35:07.837
14	12110000	09	1	0	0	0	0	2013-04-16 23:35:08.027
15	12110000	10	0	1	0	0	0	2013-04-16 23:35:08.027

Gambar 13. Data Baru di *Database* biadw Tabel FactJumlahCamaba

PILIHAN 1	NAMA CALON
09	(Multiple Items)

	THN MASUK		
	2010	2012	Grand Total
KD CALON	Camaba Count	Camaba Count	Camaba Count
00110000	1		1
10110000	1		1
10110340	1		1
12110000		1	1
12113008		1	1
20110000	1		1
30110000	1		1
40110000	1		1
50110000	1		1
60110000	1		1
70110000	1		1
80110000	1		1
90110000	1		1
Grand Total	11	2	13

Gambar 14. Data Baru pada *Report*

Berdasarkan pengujian di atas, pada Tabel 1 dapat disimpulkan untuk mengeksekusi 11 perubahan data mulai dari proses *insert* ke tabel sumber hingga bisa ditampilkan di *browser* laporan memerlukan waktu satu menit, sehingga dengan semakin tingginya frekuensi periode permintaan perubahan data akan semakin mendekati *real-time*.

Tabel 1. Tabel Kesimpulan Pengujian

Menit ke-	sumber.MHS_ PENDAFTAR	staging.camaba	biadw.camaba biadw.FactJumlahCamaba	*Report*	Waktu
0	*insert* 11 data	-	-	-	-
1	-	*insert* 11 perubahan data dari sumber	*insert* dan *update* 11 perubahan data dari staging	perubahan tampil di *report*	1 menit
Total Waktu					**1 menit**

5. Kesimpulan

Konsep *real-time* diwujudkan dengan mengintegrasikan CDC dengan *tool* ETL menggunakan model CDC *Pull* secara periodik meminta perubahan data, yaitu setiap menit setiap harinya, sehingga mampu memberikan informasi *real-time* kepada manajemen universitas. SSIS *package* memproses perubahan data saja dengan menerima sekumpulan *record* yang merepresentasikan perubahan yang ditangkap sejak siklus permintaan terakhir, sehingga membutuhkan waktu yang singkat dibandingkan me-*load* semua data. Untuk mengeksekusi 11 perubahan data mulai dari proses *insert* ke tabel sumber hingga bisa ditampilkan di *browser* laporan memerlukan waktu satu menit. Dengan cara ini data pada *data warehouse* selalu sinkron dengan data pada sistem transaksionalnya dengan hanya satu menit penundaan, sehingga dengan semakin tingginya frekuensi periode permintaan perubahan data akan semakin mendekati *real-time*.

Referensi

Attachmate Corporation. 2005. *Capturing Changes to Host-Based Data Sources*, Attacmate Technical Paper, (Online), (http://www.attachmate.com/NR/rdonlyres/2F8775C9-9CA1-411E-95D9-5415D7EDDB7D/0/tp_capturing_changes.pdf, diakses 18 Januari 2013).

Attunity. 2006. *Real Time Business Intelligence Enabling Effective Decision Making: Strategic, Real Time Data Integration Platform With Change Data Capture*, Attunity White Paper, (Online), (http://i.zdnet.com/whitepapers/Attunity_Real_Time_Biz_Intelligence.pdf, diakses 18 Januari 2013).

Azvine, B., Cui, Z., Nauck, D.D. & Majeed, B. 2006. Real Time Business Intelligence for the Adaptive Enterprise. *The 3rd IEEE International Conference on Enterprise Computing, E-Commerce, and E-Services*, pp. 1-11.

Botan, I., Cho, Y., Derakhshan, R., Dindar, N., Haas, L.M., Kim, K. & Tatbul, N. 2009. Federated Stream Processing Support for Real-Time Business Intelligence Applications. *In BIRTE 2009*, pp. 14-31.

Inmon, W.H. 2005. *Building the Data Warehouse, Fourth Edition*. New York: John Wiley & Sons,Inc.

Jörg, T. & Deßloch, S. 2008. Towards Generating ETL Processes for Incremental Loading. *In Proceedings of the 2008 International Symposium on Database Engineering & Applications (IDEAS '08)*.

Kusnawi. 2008. Aplikasi Data Warehouse untuk Business Intelligence. *Jurnal Dasi, vol. 9, no. 1*, pp. 82-91.

McGehee, B. 2008. *Brad's Sure Guide to SQL Server 2008*. Tasmania: Simple-Talk Publishing.

Ponniah, P. 2010. *Data Warehousing Fundamentals*, *2nd edition*. Singapore: John Wiley & Sons Inc.

Prasetyo, M.A., Saikhu, A. & Sarwosri, 2010, *Pembuatan Aplikasi OLAP Untuk Pelaporan pada PT. Aneka Tuna Indonesia Menggunakan SQL Server 2005*, (Online), (http://digilib.its.ac.id/public/ITS-Undergraduate-9803-Paper.pdf, diakses tanggal 20

Januari 2014).

Sandu, D.I. 2008. Operational and Real-Time Business Intelligence. *Revista Informatica Economică, no.3(47)*, pp. 33-36.

Tank, D.M., Ganatra, A., Kosta, Y.P., Bhensdadia, C.K. 2010. Speeding ETL Processing in Data Warehouses Using High-Performance Joins for Changed Data Capture (CDC). *Advances in Recent Technologies in Communication and Computing (ARTCom), 2010 International Conference*, pp.365-368.

Turban, E., Aronson, J.E. & Liang, T.P. 2004. *Decision Support Systems and Intelligent Systems, 7th Edition*. New Jersey: Prentice-Hall.

Wilarso, I. 2008. Pemanfaatan Data Warehouse di Perguruan Tinggi Indonesia. *Jurnal Sistem Informasi MTI-UI, vol.4, no.1*, pp. 47-54.