Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer e-ISSN: 2548-964X
Vol. 2, No. 4, April 2018, hlm. 1366-1371 http://j-ptiik.ub.ac.id

Analisis Peningkatan Performa Proses ETL (*Extract, Transform,* Dan *Loading*) Pada *Data Warehouse* Dengan Menerapkan *Delta Extraction* Menggunakan *Historical Table*

Arif Bimo Winnetou[1], Satrio Agung Wicaksono[2], Aryo Pinandito[3]

Program Studi Sistem Informasi, Fakultas Ilmu Komputer, Universitas Brawijaya
Email: [1]arif.bimo24@gmail.com, [2]satrio.agung@ub.ac.id, [3]aryo.pinandito@gmail.com

Abstrak

Dalam jangka waktu tertentu, umumnya instansi pendidikan menghasilkan data dengan kapasitas yang sangat besar dari kegiatan operasionalnya. Dari data tersebut akan muncul kebutuhan untuk mendapatkan informasi tertentu yang berguna sebagai bahan evaluasi kinerja atau bahkan sebagai bahan untuk mengambil keputusan manajemen kedepannya. *Data warehouse* merupakan salah satu solusi dari kebutuhan tersebut. Dengan bertambahnya kapasitas dan kompleksitas suatu *data warehouse* yang dimiliki maka teknik *full extract* dalam proses *Extract Transform Loading* (ETL) menjadi tidak memadai bahkan pada kasus tertentu proses ini tidak bisa diaplikasikan, salah satunya dikarenakan teknik ini menyebabkan *cost* yang besar. *Delta extraction* dapat menjadi salah satu solusi dari permasalahan tersebut dikarenakan prosesnya yang hanya meng-*extract* data yang mengalami perubahan saja pada proses ETL. Teknik *delta extraction* ini akan diuji coba pada DBMS MySQL dengan menggunakan *historical table* dan *trigger*. Dari hasil uji coba beban didapatkan kesimpulan bahwa pengimplementasian *trigger* memberikan beban terhadap tabel *parent*-nya, namun waktu tunggu yang dibutuhkan masih dapat ditoleransi.

Kata kunci: *Data, Data warehouse, ETL, Cost, Delta extraction*

Abstract

In a certain period of time, most educational institutions produce data with a huge capacity from their operational activities. From those data will appear the need to obtain certain information which is useful as a material of performance evaluation or even as a material of management decision. Data warehouse is one of solutions of these needs. When the capacity and complexity of data warehouse has been increasing, then full extract technique in Extract Transform Loading (ETL) process becomes inadequate, even in others case this technique can not be applied, because this technique causes a large cost. Delta extraction can be one of the solutions of this problem because the process is to extract only the data which has been changing in ETL process. Delta extraction technique will be tested on MySQL DBMS using historical table and trigger. From the result of load test, it can be concluded that trigger gives the load to its parent table, but the waiting time that is required is still tolerable.

Keywords: *Data, Data warehouse, ETL, Cost, Delta extraction*

1. PENDAHULUAN

Dalam jangka waktu tertentu, umumnya instansi pendidikan menghasilkan data dengan kapasitas yang sangat besar dari kegiatan operasionalnya. Dari data tersebut akan muncul kebutuhan untuk mendapatkan informasi tertentu yang berguna sebagai bahan evaluasi kinerja atau bahkan sebagai bahan untuk mengambil keputusan manajemen kedepannya. *Tools* yang dapat menangani masalah dan memenuhi kebutuhan tersebut adalah *data warehouse* (Firmansyah, 2016). *Data warehouse* dapat mengelola data dengan jumlah yang sangat besar dan mampu menyediakan informasi secara akurat serta cepat dari satu atau beberapa sumber data (*database).*

Lalu, akan muncul suatu masalah jika dalam mengelola *data warehouse* masih menggunakan pendekatan tradisional untuk proses ETL-nya. Maksudnya adalah proses ETL tersebut masih menggunakan skenario

refreshment yang mengulang proses *initial loading*, atau yang biasa dikenal dengan istilah *Full extract* (Mekterović & Brkić, 2015). Dengan kata lain, teknik ini melakukan proses ETL seluruhnya kembali dari awal. Dengan bertambahnya kapasitas dan kompleksitas *data warehouse* yang dimiliki maka teknik *full extract* menjadi tidak memadai bahkan pada kasus tertentu proses ini tidak bisa diaplikasikan. Selain itu teknik ETL dengan skenario *refreshment* seperti ini juga menyebabkan *cost* yang besar. *Cost* yang difokuskan disini adalah waktu yang dibutuhkan dalam melakukan proses ETL.

Solusi untuk menangani masalah tersebut adalah dengan menerapkan proses ETL yang menggunakan skenario *refreshment* bertahap, sesuai dengan perubahan yang terjadi di sumber data setelah sinkronisasi terakhir. Dengan kata lain, *user* hanya perlu memperbarui data yang mengalami perubahan setelah proses *reload* yang sebelumnya sudah dilakukan. Teknik atau pendekatan ini dikenal dengan istilah *Incremental Loading* (Mekterović & Brkić, 2015). Teknik tersebut juga memiliki istilah lain yaitu *delta extraction* yang prosesnya juga hanya meng-*extract* data yang mengalami perubahan saja pada proses ETL. Data yang sudah di-*load* dan tidak mengalami perubahan maka tidak akan di-*extract* kembali dan tidak perlu dihapus terlebih dahulu sebelum proses *load (incremental update)*. Hal ini dapat meningkatkan performa *data warehouse* dibandingkan dengan teknik proses ETL keseluruhan data (SAP, 2016). Sementara itu, *delta extraction* dapat dilakukan dengan berbagai cara, salah satunya menggunakan *historical table* (Mekterović & Brkić, 2015).

Historical table ini nantinya akan menampung histori atau perubahan data yang terjadi pada tabel tertentu. Selain itu, juga diperlukan *trigger* untuk memasok data ke *historical table. Trigger* adalah sekumpulan perintah *Structured Query Language* (SQL) yang otomatis akan melakukan suatu aksi saat operasi spesifik, seperti perubahan data dalam tabel terjadi (Tech Target, 2017). Namun, pemakaian *trigger* tersebut akan meningkatkan beban yang perlu ditanggung oleh tabel *parent*-nya. Maka dari itu, perlu dilakukan pengujian untuk mengetahui apakah akibat penggunaan *trigger* tersebut sangat membebani tabel *parent*-nya atau tidak yang dihitung dalam satuan waktu.

Penelitian ini dilakukan dengan menggunakan DBMS MySQL dikarenakan masih banyaknya instansi pendidikan yang menggunakan DBMS tersebut, salah satunya adalah Universitas Budi Luhur, Jakarta (Warnars, 2010). Selain itu MySQL merupakan salah satu *database* relasional yang banyak digunakan (Zhang, et al., 2015). Universitas Brawijaya sebagai salah satu instansi pendidikan yang memiliki sangat banyak data, akan dilibatkan sebagai data referensi dan sebagai data latih bagi penelitian ini. Hasil dari penelitian ini diharapkan dapat menjadi bahan rekomendasi untuk mempercepat proses ETL.

2. PENELITIAN TERKAIT

Dalam penelitiannya yang berjudul "*Extracting Delta for Incremental Data warehouse Maintenance*" Prabhu Ram dan Lyman Do menjelaskan bahwa dengan melakukan *delta extraction* dapat mereduksi waktu yang dibutuhkan untuk proses *extract* daripada proses *extract* biasa. Dan jika ingin menangkap status perubahannya juga, maka *trigger* bisa menjadi metode yang dapat digunakan dalam proses *delta extraction* (Ram & Do, 2000).

Halim Elamy et al. dalam penilitiannya yang berjudul "*Building Data warehouse with Incremental Maintenance for Decision Support*" berhasil melakukan proses *loading* dan *updating* pada tabel fakta secara *incremental* yang mampu meningkatkan performa *query*. Penulis juga berhasil membuktikan bahwa *trigger* dapat berjalan otomatis saat terjadi pembaharuan data dan dapat langsung mencerminkannya pada *data warehouse* yang dituju (Elamy, et al., 2005).

3. LANDASAN KEPUSTAKAAN

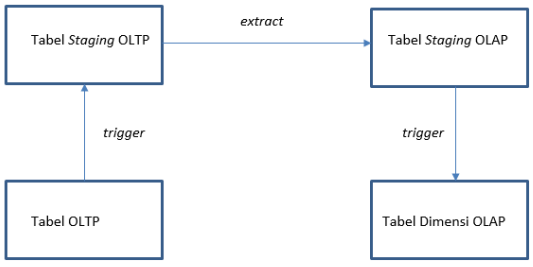
3.1. *Data warehouse*

Data warehouse adalah *tools* yang dapat dijadikan sebagai pendukung pengambilan keputusan yang berupa kolam data. Biasanya suatu data telah disediakan secara terstruktur dan siap diproses (yaitu: *Online Analytical Processing [OLAP], data mining,* laporan dan bentuk aplikasi pendukung pengambilan keputusan lainnya). *Data warehouse* adalah penyimpanan data historis terintegrasi yang berasal dari beberapa sumber data yang digunakan untuk kepentingan analisis sehingga mampu menjadi komponen pendukung pengambilan keputusan (Turban, et al., 2010).

Data warehouse melakukan pengekstrakan, pembersihan, penyesuaian, dan pengiriman sumber data ke dalam *dimensional data store*, yang kemudian dapat dilakukan analisis dan penerapan *query* tertentu sebagai bahan pendukung pengambilan keputusan (Kimball & Caserta, 2004).

3.2. *Delta extraction* Menggunakan *Historical Table*

Metode ini membutuhkan tabel *staging* yang dijadikan sebagai *historical table* untuk tempat sementara hasil *capture* perubahan data yang terjadi (Mekterović & Brkić, 2015). Satu *historical table* di sisi *Online Transaction Processing* (OLTP) dan satu lagi *historical table* di sisi OLAP. Tabel transaksi di OLTP akan dipasang *trigger* untuk menangkap perubahan yang terjadi lalu hasil tangkapannya dimasukkan ke *historical table* yang ada di OLTP. *Historical table* tersebut akan dijadikan sebagai *source* dalam proses *extract* yang *destination table*-nya adalah *historical table* yang berada di OLAP. *Historical table* OLAP tersebut akan dipasang *trigger* yang akan menangkap perubahan data yang terjadi, lalu hasil tangkapannya dimasukkan ke tabel dimensi atau tabel fakta yang ada di OLAP. Dengan begitu maka tabel dimensi atau tabel fakta yang dituju sudah memiliki data yang bersih sesuai kebutuhan (Ram & Do, 2000). Gambar 1. merupakan ilustrasi dari penjelasan diatas.



Gambar 1. Ilustrasi proses *delta extraction* menggunakan *historical table*

Sumber: (Ram & Do, 2000)

4. METODOLOGI

4.1. Studi Pustaka

Pada tahap ini akan dilakukan pengumpulan dan membaca jurnal, *e-book*, buku, naskah penelitian, dan informasi dari internet sebagai bahan referensi untuk melakukan penelitian ini.

4.2. Analisis Kebutuhan dan Pengumpulan Data

Langkah selanjutnya setelah melakukan studi literatur adalah analisis kebutuhan dan pengumpulan data. Tahap analisis kebutuhan dan pengumpulan data ini merupakan tahapan dimana akan dilakukan analisis kebutuhan dan pengumpulan data untuk sistem yang akan dibuat nantinya. Tahap ini dilakukan untuk mengetahui apa saja pelaporan yang dibutuhkan, seperti apa keadaan *environment database* OLTP yang sudah ada saat ini, dan bagaimana keadaan *environment data warehouse* yang diharapkan. Adapun proses yang dilakukan ketika analisis kebutuhan dan pengumpulan data, yaitu dengan wawancara. Wawancara ini dilakukan kepada pihak TIK Universitas Brawijaya. Hasil dari tahap ini akan digunakan sebagai acuan untuk proses perancangan sistem.

4.3. Perancangan OLTP dan *Data warehouse*

Pada tahap ini akan dilakukan perancangan OLTP dan *data warehouse* serta sistem *delta extraction* yang nantinya akan dibangun. Hasil dari tahap ini dijadikan sebagai dasar dari implementasi OLTP dan *data warehouse* yang akan dibangun. Dibawah ini merupakan rincian dari perancangan yang dilakukan:

1. Perancangan *Database* OLTP

   Pada tahap ini akan dilakukan perancangan *physical data model* dari *database* OLTP sesuai dengan lingkungan sistem yang sudah ada di Universitas Brawijaya dengan mengacu pada hasil tahap analisis kebutuhan dan pengumpulan data.

2. Perancangan *Data warehouse*

   Pada tahap ini akan dilakukan perancangan *data warehouse* mengacu pada analisis kebutuhan dari pengguna.

3. Perancangan Proses *Delta extraction* dan ETL

   Pada tahap ini akan dilakukan perancangan proses *delta extraction* dan proses ETL yang direpresentasikan ke dalam penjelasan singkat dan konseptual ETL.

4.4. Pembangkitan Data

Pada langkah ini akan dilakukan perancangan dan pembangkitan data dengan lingkup jenjang sarjana dengan mengacu pada hasil tahap analisis kebutuhan dan pengumpulan data. Data ini dibuat untuk nantinya dimasukkan kedalam *database* OLTP.

Data ini akan digunakan untuk tahap selanjutnya yaitu implementasi sistem.

4.5. Implementasi Sistem

Pada tahap ini akan dilakukan eksekusi atau implementasi terhadap rancangan *database*, dan proses ETL, serta data yang sudah dibuat. Implementasi yang dilakukan diantara lain:

1. Implementasi *Database* OLTP
   Pada langkah ini akan dilakukan pembangunan *database* OLTP dan meng-*import* data penelitian yang sudah dibuat.

2. Implementasi *Data warehouse*
   Pada langkah ini akan dilakukan pembangunan *data warehouse* pada *server* OLAP.
3. Implementasi proses *Delta extraction* dan Proses ETL
   Nantinya hasil dari tahap ini akan digunakan sebagai bahan untuk dilakukan pengujian dan analisis hasil.

4.6. Pengujian dan Hasil Analisis

Pada tahap ini akan dilakukan pengujian untuk menilai kelayakan dari hasil implementasi yang telah dibangun. Selain itu juga akan dilakukan analisis dari hasil pengujian tersebut untuk menunjukkan hasil akhir yang didapat. Terdapat 2 pengujian yang akan dilakukan pada penilitian ini, yaitu pengujian beban OLTP dan pengujian performa ETL.

1. Pengujian beban OLTP
   Pada pengujian ini akan dilakukan analisis dari dampak hasil implementasi *trigger* terhadap beban yang harus ditanggung oleh *database* OLTP yang dihitung dalam satuan waktu. Karena pengimplementasian *trigger* akan memberikan beban lebih terhadap tabel induknya maka perlu dilakukan suatu pengujian yang bertujuan untuk mengidentifikasi apakah ada penurunan kecepatan yang signifikan pada OLTP setelah diimplementasikan *trigger*. Pengujian ini akan menghasilkan perbandingan kecepatan eksekusi transaksi antara tabel pada OLTP yang diimplementasikan *trigger* dengan tabel pada OLTP yang tidak diimplementasikan *trigger*.
2. Pengujian performa ETL
   Pada pengujian ini akan dilakukan analisis performa dari proses ETL yang telah diimplementasikan dalam *data warehouse.* Performa ETL yang diuji pada penelitian ini dilihat dari segi waktu untuk eksekusi *query*-nya. Pengujian ini akan menghasilkan perbandingan kecepatan eksekusi proses ETL antara *data warehouse* yang diimplementasikan *delta extraction* dengan *data warehouse* yang tidak diimplementasikan *delta extraction*.

4.7. Kesimpulan

Pada tahap ini akan menghasilkan kesimpulan dan saran dari hasil analisis yang ditemukan. Tahap ini merupakan penjabaran jawaban dari rumusan masalah dan tujuan penelitian ini.

5. PENGUJIAN DAN HASIL ANALISIS

5.1. Pengujian Beban pada Tabel ISIAN_KUESIONER_MHS

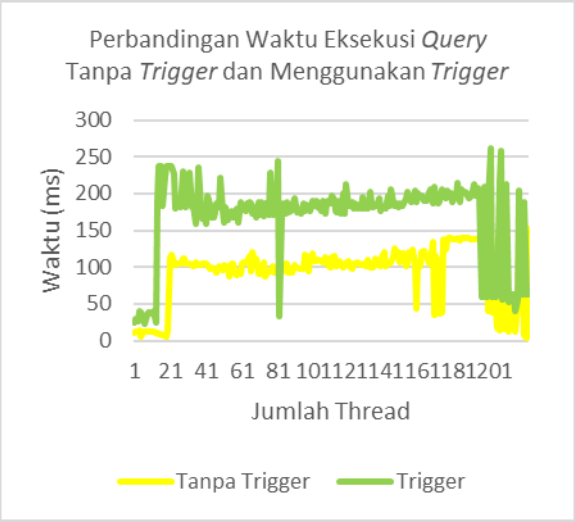
Data rata-rata waktu eksekusi *query* dari hasil pengujian dengan jumlah transaksi 200 *insert* dan 20 *update* disajikan dalam Tabel 1. Dapat dilihat bahwa rata-rata waktu eksekusi *query* yang dihasilkan oleh tabel ISIAN_KUESIONER_MHS saat dipasangkan *trigger* lebih besar (169,4 ms) dibandingkan saat tidak dipasangkan *trigger* (94,4 ms), dan itu berarti saat dipasangkan *trigger* transaksi yang dilakukan rata-rata selesai lebih lama. Tetapi waktu eksekusi *query* saat dipasangkan *trigger* juga masih dalam batas toleransi waktu tunggu *user*. Grafik perbandingannya disajikan dalam Gambar 2.

Tabel 1. Hasil rata-rata waktu eksekusi *query* tabel ISIAN_KUESIONER_MHS - 200 *insert* dan 20 *update*

Tanpa *Trigger*	*Trigger*
94,4 ms	169,4 ms

5.2. Pengujian Beban pada Tabel KHS

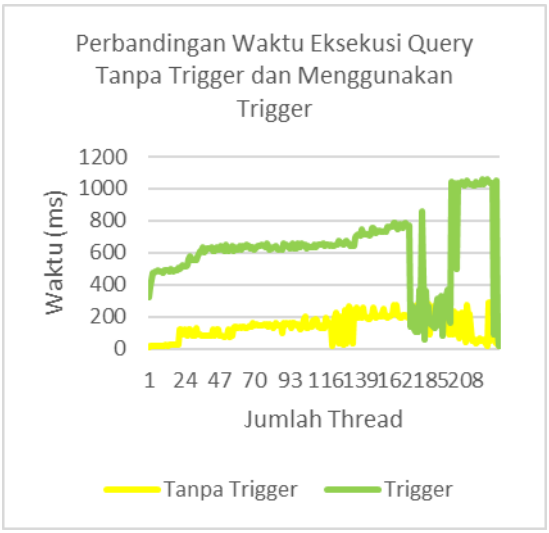
Data rata-rata waktu eksekusi *query* dari hasil pengujian dengan jumlah transaksi 200 *insert*, 20 *update*, dan 10 *delete* disajikan dalam Tabel 2. Dapat dilihat bahwa waktu eksekusi *query* yang dihasilkan oleh tabel KHS saat dipasangkan *trigger* lebih besar (627,8 ms) dibandingkan saat tidak dipasangkan *trigger* (142,6 ms), dan itu berarti saat dipasangkan *trigger* transaksi yang dilakukan rata-rata selesai lebih lama. Untuk lebih detilnya disajikan dalam Gambar 3.



Gambar 2. Hasil perbandingan rata-rata waktu eksekusi *query* tabel ISIAN_KUESIONER_MHS – 200 *insert* dan 20 *update*

Tabel 2. Hasil rata-rata waktu eksekusi *query* tabel KHS - 200 *insert*, 20 *update* dan 10 *delete*

Tanpa *Trigger*	*Trigger*
142,6 ms	627,8 ms



Gambar 3. Hasil perbandingan rata-rata waktu eksekusi *query* tabel KHS – 200 *insert*, 20 *update* dan 10 *delete*

5.3. Perhitungan Perubahan Nilai

Selanjutnya dilakukan perhitungan untuk melihat seberapa besar kenaikan waktu eksekusi *query* setelah diimplementasikan *trigger* pada tabel KHS dan ISIAN_KUESIONER_MHS. Dari perhitungan tersebut didapatkan angka kenaikan persentase yang dialami setelah pengimplementasian *trigger*, **paling kecil sebesar 79,4%** dan yang **paling besar sebesar 471,1%.**

5.4. Analisis Hasil Pengujian

Berikut beberapa hasil analisis yang didapatkan dari hasil pengujian yang telah dilakukan, antara lain:

1. Dapat diketahui bahwa jika rata-rata waktu hasil eksekusi *query* tiap operasi dari hasil pengujian beban pada tabel KHS disatukan, maka akan menghasilkan rata-rata waktu hasil eksekusi *query* saat tidak menggunakan *trigger* sebesar 113,68 ms atau 0,11368 detik. Sementara saat menggunakan *trigger* menghasilkan rata-rata waktu eksekusi *query* sebesar 500,91 ms atau 0,50091 detik. Lalu rata-rata waktu hasil eksekusi *query* keseluruhan operasi pada tabel ISIAN_KUESIONER_MHS saat tidak menggunakan *trigger* sebesar 63,56 ms atau 0,06356 detik, dan untuk saat menggunakan *trigger* sebesar 137,57 ms atau 0,13757 detik. Dapat diketahui bahwa setiap rata-rata waktu eksekusi *query* yang dihasilkan oleh masing-masing tabel, baik saat tidak menggunakan *trigger* maupun saat menggunakan *trigger*, seluruhnya masih berada dalam kategori waktu yang dapat ditoleransi *user*. Dapat diketahui pula bahwa terdapat perbedaan kecepatan eksekusi *query* yang dihasilkan oleh tabel KHS dan tabel ISIAN_KUESIONER_MHS yang cukup besar, hal itu dikarenakan berbedanya jumlah dan jenis operasi yang dilakukan oleh tiap tabel tersebut, dan juga berbedanya isi dari *trigger* yang dibutuhkan oleh masing-masing tabel tersebut. Jadi, jumlah dan jenis operasi, serta isi *trigger* yang dimiliki suatu tabel akan berpengaruh pada waktu eksekusi *query* yang dihasilkan.
2. Dari semua hasil data pengujian yang telah dilakukan, dapat diketahui bahwa teknik *delta extraction* dengan menggunakan *historical table* dapat meningkatkan kecepatan proses ETL, namun hal ini juga memiliki konsekuensi terhadap beban yang ditimbulkan dari pengimplementasian *trigger* dilihat dari segi waktu, dalam hal ini waktu eksekusi *query* yang dihasilkan dari pengimplementasian *trigger* akan **lebih lambat 246,3%**. Namun, kembali merujuk poin pertama, bahwa dalam hal ini waktu yang dihasilkan dari pengimplementasian *trigger* masih dapat

ditoleransi oleh *user*.

6. KESIMPULAN

Setelah dilaksanakannya semua langkah dalam penelitian ini, maka dapat ditarik beberapa kesimpulan untuk menjawab rumusan masalah, antara lain:

1. Pengimplementasian *trigger* memberikan dampak beban terhadap tabel *parent*-nya sehingga memperlambat kecepatan eksekusi *query*-nya, namun waktu tunggu yang dihasilkan dari waktu eksekusi *query*-nya masih berada dalam batas toleransi waktu tunggu *user*.
2. Kecepatan eksekusi proses ETL yang dihasilkan dengan teknik *delta extraction* lebih cepat secara signifikan jika dibandingkan dengan kecepatan eksekusi proses ETL dengan teknik *full extract.*

7. DAFTAR PUSTAKA

Elamy, A. H., Alhajj, R. S. & Far, B. H., 2005. Building *Data warehouse* with Incremental Maintenance for Decision Support.

Firmansyah, Muchlis, 2016. *Pembangunan Data Warehouse Keuangan Pada Instansi Pendidikan (Studi Kasus: Fakultas ABC)*, Malang: S1. Universitas Brawijaya.

Kimball, R. & Caserta, J., 2004. *The Data warehouse ETL Toolkit: Practical Techniques for Extracting, Cleaning, Conforming, and Delivering Data*. s.l.:Wiley India Pvt. Limited.

Mekterović, I. & Brkić, L., 2015. Delta View Generation for Incremental Loading of Large Dimensions in a *Data warehouse*.

Ram, P. & Do, L., 2000. Extracting Delta for Incremental *Data warehouse* Maintenance.

SAP, 2016. *Delta extraction.* [Online] Tersedia pada: http://help.sap.com/saphelp_dm40/helpdata/en/d0/4cc138944cfa06e10000000a11405a/content.html [Diakses pada 23 Januari 2017].

Tech Target, 2017. *Definition Trigger*. [Online] Tersedia pada: http://searchsqlserver.techtarget.com/definition/*trigger* [Diakses pada 25 Januari 2017].

Turban, E., Sharda , R., Delen, D. & King, D., 2010. *Business Intelligence: A Managerial Approach*. 2nd ed. s.l.:Prentice Hall.

Warnars, S., 2010. Tata Kelola *Database* Perguruan Tinggi yang Optimal dengan *Data warehouse*. *Telkomnika*.

Zhang, H. et al., 2015. *In-Memory Big Data Management and Processing: A Survey*.