INTERNAL
Information System Journal

INTERNAL (Information System Journal)
Volume 4 No. 2 | Desember 2021 : 102-112
DOI: 10.32627
http://jurnal.masoemuniveristy.ac.id/index.php/internal

p-ISSN: 2621-9433
e-ISSN: 2656-0259

Analisis Dynamic ETL Incremental Load untuk Data Integration Datawarehouse

Zulkifli Arsyad
Teknik Komputer dan Informatika, Politeknik Negeri Bandung, Indonesia
zulkifli.arsyad@jtk.polban.ac.id

Info Artikel

Sejarah artikel :
Diterima 18 November 2021
Direvisi 16 Desember 2021
Disetujui 24 Desember 2021
Diterbitkan 30 Desember 2021

ABSTRACT

Data integration is a combination of techniques and businesses that are used to collect data from different sources into useful and valuable information ETL process that includes extracting data from various data sources, transforming data to form and calculate data and load data on target storage, to support data warehouse need. Based on organizations and industries that have implemented data warehouse, the problem that generally arises regarding data load is the difficulty in integrating different data sources, how to form data from various data formats into uniform data, how to integrate data delta between data sources and target storage in an incremental load process so that this data synchronization process can be carried out continuously and relatively faster. ETL process requires a platform that can facilitate data integration needs, in order to run this process. SSIS (SQL Server Integration Service) is a Data Integration platform to build an enterprise-level data integration and solutions for data transformation. Integration Service can extract and change data (transform) from various sources such as XML data files, flat files, APIs, and relational data sources, and then load into one or several destination data. According to the problem related to data load, we will examine how the solution model uses SSIS for the ETL process. This paper proposed an ETL Architecture model by completing the ETL process for full & incremental load extraction and the original data layer.

Keywords : *Data Integration; ETL; SSIS; data warehouse; multiple source.*

ABSTRAK

Data *Integration* merupakan kombinasi teknik dan bisnis yang digunakan untuk menggabungkan data dari sumber yang berbeda menjadi informasi yang bermakna dan berharga. Untuk kebutuhan data *warehouse* , proses ETL ini yang mencakup ekstrak data dari berbebagai sumber data, melakukan transformasi data untuk membentuk dan mengkalkulasi data serta load data pada target *storage*. Dari beberapa organisasi dan industri yang mengimplementasikan datawarehouse ini persoalan yang umumnya muncul mengenai load data, kesulitan dalam mengintegrasikan sumber data berbeda, serta bagaimana membentuk data dari berbagai format data yang perlu untuk di seragamkan, serta bagaimana mekanise integrasi terkait delta data antara sumber data dengan target *storage* dalam proses *incremental load* agar proses sinkronisasi data ini dapat dijalankan secara terus menerus dan relatif lebih cepat. Untuk menjalankan proses ETL ini dibutuhkan *platform* yang dapat memfasilitasi kebutuhan integrasi

data. SSIS (*Sql Server Integration Service*) merupakan *platform* Data *Integration* untuk membangun suatu enterprise level data integration serta solusi atas transformasi data. Layanan Integrasi ini dapat mengekstraksi dan mengubah data (*transform*) dari berbagai macam sumber seperti file data XML, flat file,API, dan sumber data relasional, dan kemudian memuat ke dalam satu atau beberapa *destination* data. Dari persoalan terkait load data ini akan di kaji bagaimana model solusi menggunakan SSIS untuk proses ETL.

Kata Kunci : *Data Integration; ETL; SSIS; data warehouse; multiple source*.

PENDAHULUAN

Data Integration merupakan kombinasi teknik dan bisnis yang digunakan untuk menggabungkan data dari sumber yang berbeda menjadi informasi yang bermakna dan berharga. *Data Integration* adalah praktik yang berkaitan dengan pengelolaan data yang berjalan antara aplikasi, penyimpanan data, sistem, dan organisasi secara tradisional [14]. Dalam membangun suatu sistem data *warehouse*, *multiple data source & platform* menjadi suatu hal yang menarik untuk di analisis lebih jauh bagaimana mekanisme ekstraksi data, *transform* data (mengubah bentuk data menjadi data yang diinginkan) serta *load* data . Data *warehouse* saat ini harus mampu mengatasi tipe baru, volume baru, tingkat kualitas data baru, persyaratan kinerja baru, metadata baru, dan persyaratan pengguna baru [15].

Pada 6 (enam) organisasi besar temuan menunjukkan bahwa penyebaran ETL paling sering memperkenalkan masalah load data yang disebabkan oleh (1) bagian ETL (*Extract Transform Load*) yang gagal dan tidak mengirimkan data tepat waktu, (2) sistem informasi terkunci selama eksekusi ETL, dan (3) pengguna tidak dapat menemukan data dalam target karena kesalahan dalam cara kunci utama ditransformasikan. Lebih jauh lagi, akurasi, ketepatan waktu, kepercayaan, dan masalah konsistensi representasional juga ditemukan disebabkan oleh ETL teknologi [1]. Temuan ini memang perlu di lihat dari beberapa perspektif sebagai penyebab dari persoalan yang ditemukan, seperti jumlah data yang di sinkronkan, waktu sinkron, mekanisme transformasi dan ekstrak data, serta perspektif lainnya yang dirasa memiliki pengaruh terkait aksesibilitas data ini.

Pada proses ETL ini, mekanisme sinkronisasi data terkait data yang telah di perbaharui pada sumber data perlu untuk di sinkronkan ke *destination* data berdasarkan periode tertentu.untuk itu perlu untuk di pertimbangkan mekanisme ekstraksi delta data antara sumber data dengan *destination* data sehingga hanya data yang terbaru saja yang diintegrasikan ke *destination data*. Delta data ini dikenal sebagai istilah *incremental load* dimana terdapat delta perbedaan data antara data yang terdapat pada destination data dengan sumber data berdasarkan aktifitas *insert*, *update* dan *delete*. Tanggal ekstrak terakhir serta perubahan data pada sumber data disimpan sehingga hanya data berdasarkan tanggal perubahan terakhir serta taggal ekstrak yang akan di integrasikan ke *destination data*. Namun, dari beberapa perusahaan yang mengimpementasikan data *warehouse* ini, format data yang berbeda menjadi kendala mengingat tidak semua data memiliki *timestamp* sebagai acuan untuk *update* data terakhir, serta tidak semua tabel memiliki *index* untuk atribut-atribut yang dibutuhkan pada saat *query where condition*. namun jika dilakukan *flush* and *pull* data pada sumber tertentu berkendala pada *time* limit saat

proses ETL mengingat data yang cukup besar jika diambil keseluruhan data yang akan diintegrasikan.

SSIS adalah *platform* Data *Integration* untuk membangun suatu *enterprise* level *data integration* serta solusi atas transformasi data. Layanan Integrasi ini untuk memecahkan masalah bisnis yang rumit dengan menyalin atau mengunduh file, memuat data *warehouse*, membersihkan dan *mining* data, dan mengelola data SQL Server [2].

Layanan Integrasi bertujuan untuk memecahkan masalah bisnis yang kompleks dengan menyalin atau mengunduh file, memuat gudang data, membersihkan dan menambang data, dan mengelola data SQL Server[8]. Layanan Integrasi ini dapat mengekstraksi dan mengubah data (*transform*) dari berbagai macam sumber seperti file data XML, flat file, API, dan sumber data relasional, dan kemudian memuat ke dalam satu atau beberapa *destination* data.

Pada penelitian ini akan dilakukan analisa kajian terkait mekanisme ETL dengan multiple data *source* menggunakan *platform* SSIS yang di fokuskan pada multiple format data untuk menyelesaikan persoalan terkait *incremental load* data.

METODE

Extract Transform Load

ETL (*Extract Transform Load*) adalah proses yang digunakan untuk memungkinkan perusahaan memindahkan data dari berbagai sumber, memformat ulang dan membersihkannya, dan kemudian memuat data ke area lain untuk analisis atau sistem operasional untuk mendukung proses bisnis organisasi [10]. Proses ETL terdiriri dari [11] :

1. *Extract*, ETL Melakukan Ekstraksi data untuk mendapatkan data dari berbagai sumber dan memuatnya ke dalam *warehouse.*
2. *Transform,* adalah untuk mengubah data yang diekstraksi sesuai dengan peran yang dirancang sebelumnya dan memproses data redundan dan ambigu sehingga format data yang awalnya bervariasi dapat disatukan.
3. *Load,* adalah bahwa semua data setelah transformasi akan ditambahkan secara bertahap atau menyeluruh ke database target [3] secara singkat proses ETL mengekstrak data dari sistem sumber, mengubah data sesuai dengan aturan bisnis, dan memuat hasilnya ke dalam datawarehouse.

Aktivitas yang terkait dengan ETL di data *warehouse* saat ini paling memakan waktu dan fokus pengembang dalam memaksimalkan integrasi serta transformasi [9]. Dalam proses ekstraksi terdapat 2 jenis tipe ekstraksi yaitu [4]:

1. *Full extraction data* dari sistem sumber benar-benar diekstraksi. Ekstraksi penuh mengekstrak semua data dari sistem sumber tidak diperlukan untuk melacak perubahan yang dibuat di sumber sistem sehubungan dengan ekstraksi sebelumnya.
2. *Incremental Load* , hanya mengambil perubahan yang dilakukan pada sistem sumber yang akan diekstraksi dengan ekstraksi sebelumnya yang telah di lakukan. *Change data capture* (CDC) adalah mekanisme yang digunakan untuk *incremental extraction*. Kebanyakan *tools* ETL lakukan tidak menggunakan mekanisme CDC ini, sebaliknya mereka ekstrak seluruh tabel dari sumber sistem di area pementasan dan bandingkan tabel ini dengan data atau tabel diekstrak dari ekstraksi sebelumnya ke identifikasi

p-ISSN: 2621-9433
e-ISSN: 2656-0259

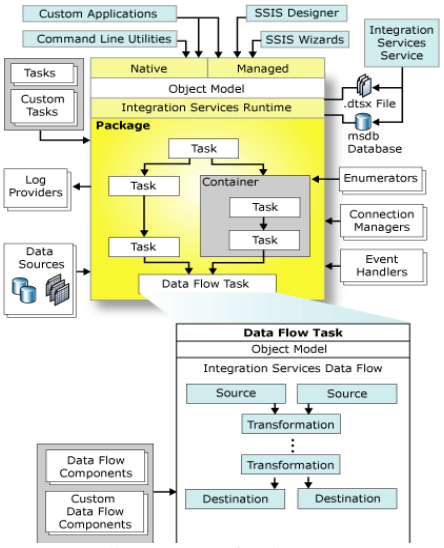
perubahannya. Perbandingan ini menghasilkan dampak performa yang besar proses ELT data *warehouse*

Dynamic ETL

Dalam pendekatan MDA (*model driven Architecture*), pengembangan perangkat lunak dapat sebagian atau sepenuhnya otomatis melalui penerapan transformasi model secara berurutan, mulai dari model yang mewakili spesifikasi sistem yaitu, model konseptual dan berakhir dalam model yang mewakili deskripsi rinci dari realisasi fisik, dari mana kode yang dapat dieksekusi pada akhirnya dapat dihasilkan [5]. MDA mendefinisikan pendekatan spesifikasi sistem TI yang memisahkan fungsionalitas sistem dari detail implementasi pada platform teknologi tertentu [12].

Pengembangan proses ETL sesuai dengan pendekatan MDA diusulkan dalam (Mun˜oz dkk. 2008; Mazo'n dan Trujillo 2008). Dengan demikian, model konseptual didefinisikan sebagai model *independen platform*—PIM yang kemudian diubah secara otomatis menjadi model spesifik *platform* — PSM (melalui serangkaian transformasi yang ditentukan secara formal) [5]. pendekatan MDA secara umum, didasarkan pada penyempurnaan model melalui transformasi model yang berurutan, namun proses ini biasanya juga mengharuskan model yang dibuat secara otomatis diperpanjang secara manual dengan detail tambahan [5]. Dalam mengimplementasikan Automati ETL ini, peneliti menggunakan *platform* Data *Integration* SSIS.

Microsoft Integration Services atau SQL *Server Integration Service* adalah *platform Data integration* yang dimiliki oleh *microsoft* untuk membangun integrasi data level *enterprise* dan solusi transformasi data. Layanan Integrasi ini untuk memecahkan masalah bisnis yang rumit dengan menyalin atau mengunduh file, memuat data *warehouse*, membersihkan dan mining data, dan mengelola objek dan data SQL *Server* [6]. Layanan Integrasi dapat mengekstraksi dan mengubah data dari berbagai macam sumber seperti file data XML, flat file, dan sumber data relasional, dan kemudian memuat data ke satu atau beberapa tujuan [6]. Berikut arsitektur dari *Integration Service.*



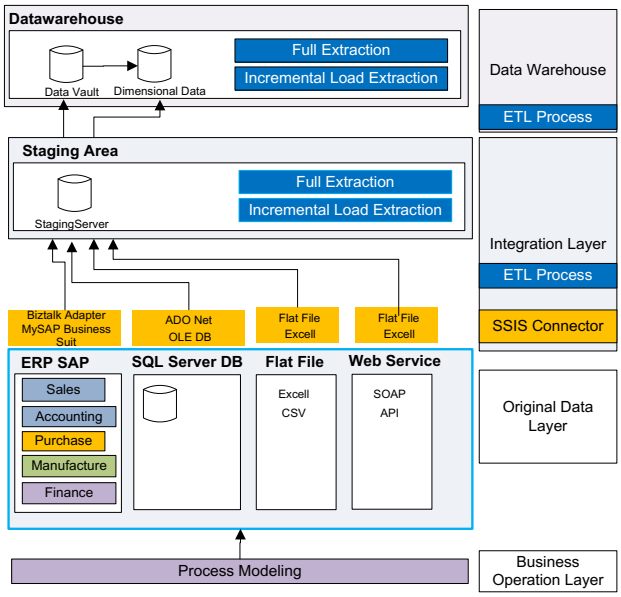
Gambar 1. Arsitektur SSIS

server Integration Service memiliki arsitektur yang memisahkan pergerakan data dan transformasi dari aliran kontrol paket dan manajemen[13]. *Data flow task* mengelola tugas aliran data yang didedikasikan untuk memindahkan dan mengubah SQL data dari sumber ke *destination*. *Data flow task* berisi komponen *data flow*, transformasi dan destinasi [2].

HASIL DAN PEMBAHASAN

ETL *Architecture*

ETL *architecture* dapat dilihat pada gambar berikut.

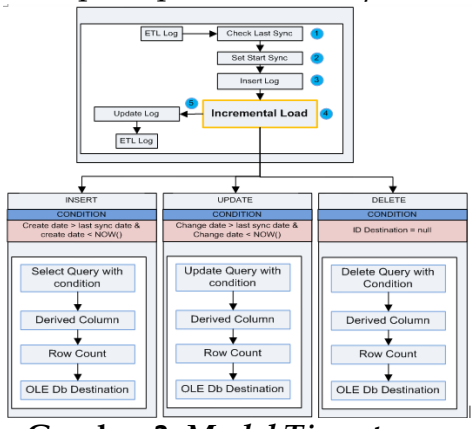


Gambar 2. Usulan Arsitektur ETL

Berdasarkan gambar 2, peneliti mengusulkan arsitektur ETL ini dengan melengkapi ETL proses untuk *full and incremental load extraction* serta *original data layer.* Pada *business operation layer* , sebelum dilakukan proses ETL untuk kebutuhan data *warehouse*, terlebih dahulu difahami bagaimana proses/*flow* itu berjalan secara bisnis. Hal tersebut dapat berpengaruh pada *design* ETL yang dibuat.

Model Timestamp

Model timestamp ini menggunakan attribute dengan tipe data *timestamp , date, date time* sebagai acuan komparasi pada proses *insert, update* serta *delete.*



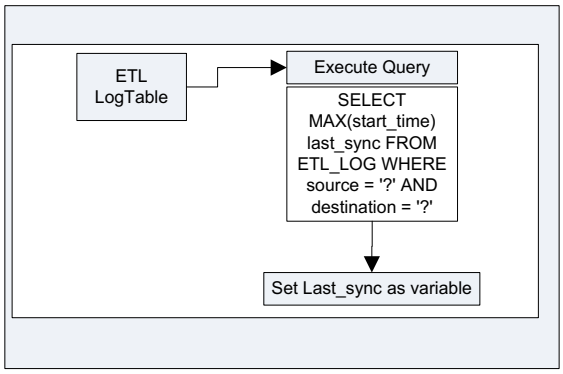
Gambar 3. *Model Timestamp*

p-ISSN: 2621-9433
e-ISSN: 2656-0259

Pada gambar 3 proses *insert* melakukan komparasi antara *create date* dengan *last sync date* yang tersimpan pada ETL Log, begitu juga dengan proses *update.*

Check *last Sync* pada ETL Log

Check *last Sync* pada ETL Log dapat dilihat pada gambar 4.



Gambar 4. ***Check Last Syncronization***

Berdasarkan gambar 4, dapat di cek tanggal terakhir ETL dijalankan dengan menjalankan query berikut :

SELECT MAX(end_time) last_sync FROM ETL_LOG WHERE source = '?' AND destination = '?' AND state = 'success'

Query tersebut mengambil tanggal terakhir berakhirnya proses ETL dengan status sukses, setiap ETL dijalankan, akan diambil last_sync sebagai start_time proses ETL selanjutnya, yaitu :

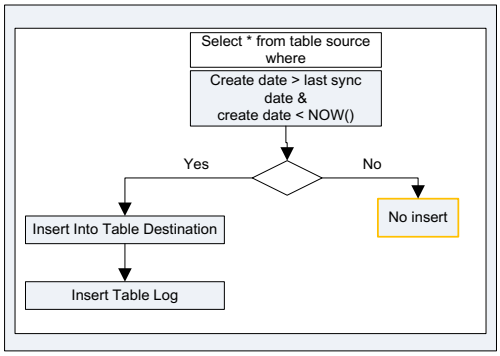
1. *Set start sync*
   Mengeset tanggal dimulainya proses ETL, hal tersebut diperlukan sebagai parameter dalam *query* pencarian
2. *Insert log,*
   setiap ETL dijalankan, informasi mengenai ETL akan disimpan pada tabel ETL log beserta tanggal dilakukannya proses ETL, adapun informasi yang tersimpan pada log tersebut adalah :
   1. *Source*, nama tabel / *view* / API sumber
   2. *Destination,* nama tabel destinasi
   3. *Start_time,* tanggal dimulai ETL
   4. *End_time,* tanggal berakhir ETL
   5. *Row,* jumlah data yang tersimpan
   6. *Flag,* kategori *Insert / Update /Delete*
   7. *ETL_name, nama ETL*
   8. *State,* status berhasil atau gagal.
3. *Update Log,* setelah proses incremental selesai langkah selanjutnya akan dilakuan update log untuk menyimpan total data yang telah di *insert/update/delete*, tanggal berakhir serta status.

Incremental Load

Incremental load mengambil perubahan yang dilakukan pada sistem sumber. Peneliti mengusulkan model *incremental* seperti pada gambar berikut.

p-ISSN: 2621-9433
e-ISSN: 2656-0259

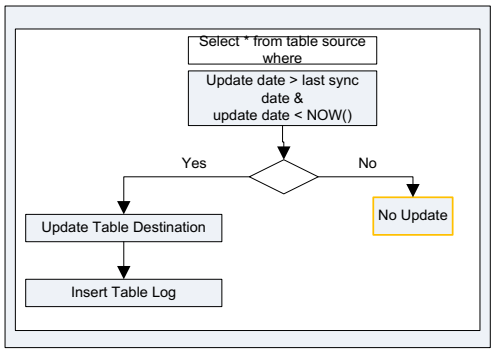
1. Kondisi *Insert*



Gambar 5. Kondisi Insert pada Incremental Load

Create date pada sistem sumber akan di komparasikan dengan tanggal terakhir dilakukannya proses ETL(*last_sync_date*) dan create date pada sistem sumber harus lebih atau sama dengan tanggal sekarang (NOW()).

2. Kondisi *Update*

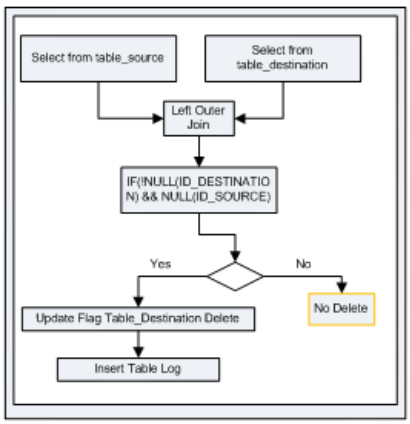


Gambar 6. Kondisi *Update* pada *Incremental load*

Update date pada sistem sumber akan di komparasikan dengan tanggal terakhir dilakukannya proses ETL untuk *update*. *Update* pada sistem sumber harus lebih atau sama dengan tanggal sekarang (NOW()).

3. Kondisi *Delete*

terdapat 2 bentuk *delete*, yaitu : (a) aksi *delete* hanya mengupdate status data tersebut namun datanya tetap ada, (b) *delete* permanen. Pada kondisi *delete* sebagai status maka metode *timestamp* bisa dilakukan, namun jika *delete* secara permanen maka akan dilakukan metode komparasi seperti pada model berikut:

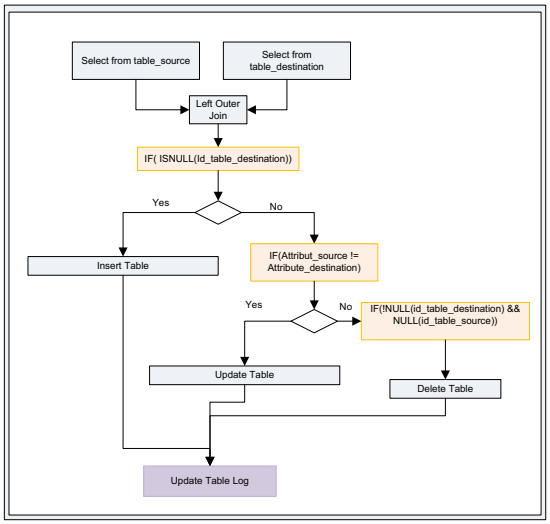


Gambar 7. Kondisi *Delete* pada *Incremental load*

p-ISSN: 2621-9433
e-ISSN: 2656-0259

Model *Compare and Consolidate*

Model *Compare and Consolidate* dapat dilihat pada gambar berikut.



Gambar 8. Model *Compare & Consolidate*

Jika tidak menemukan *timestamp / datetime* sebagai acuan proses *insert, update* dan *delete* maka alternatif lain untuk *incremental load* ini adalah menggunakan metode *compare & consolidate. Compare* adalah melakukan komparasi atribut data antara table sumber dengan table target. Jika terdapat perbedaan value maka terjadi proses update pada data tersebut. jika Berikut gambaran model untuk *compare & consolidate*.

Prototype Model Incremental Load

Pada implementasi ini akan dicobakan *data source* yang bersumber pada database SQL server yang merupakan tabel *sales.* Adapun struktur tabel dari *sales* adalah sebagai berikut :

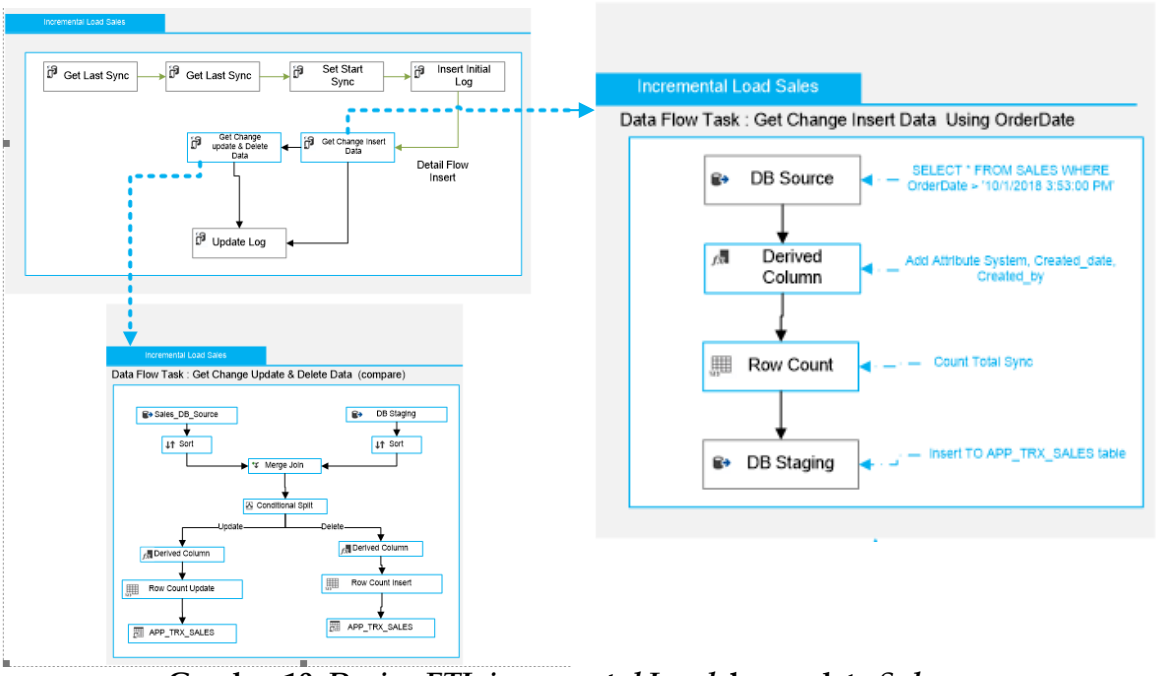
Column Name	Data Type	Allow Nulls
Region	varchar(50)	☑
Country	varchar(50)	☑
ItemType	varchar(50)	☑
SalesChannel	varchar(50)	☑
OrderPriority	varchar(50)	☑
OrderDate	varchar(50)	☑
OrderID	varchar(50)	☑
ShipDate	varchar(50)	☑
UnitsSold	varchar(50)	☑
UnitPrice	varchar(50)	☑
UnitCost	varchar(50)	☑
TotalRevenue	varchar(50)	☑
TotalCost	varchar(50)	☑
TotalProfit	varchar(50)	☑

Gambar 9. *Table Sales*

Untuk menggunakan metode *timestamp,* maka pilih atribut yang dapat menjadi acuan untuk *insert, update* dan *delete.* Dari tabel tersebut terdapat atribut *orderDate* sebagai acuan *insert*, adapun untuk *update* dan *delete* tidak ditemukan

p-ISSN: 2621-9433
e-ISSN: 2656-0259

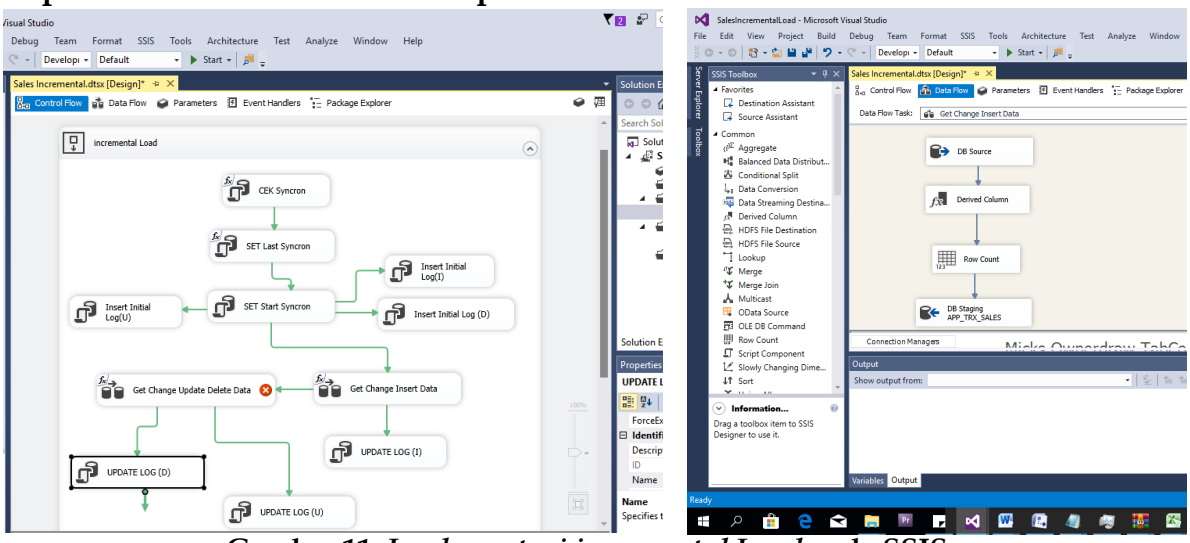
acuan sehingga digunakan metode *compare*. Pada metode *compare* ini akan dilakukan komparasi antara tabel atribut *source* dengan tabel atribut *destination*, jika terdapat perbedaan maka di identifikasi terjadi *update* data. Adapun untuk *delete* akan di lakukan komparasi ID, jika terdapat ID *primary table destination* namun ID di sumber tidak ada, maka diidentifikasi terjadi *delete* data. Berikut model *dinamic* ETL berdasarkan contoh kasus data sumber transaksi *sales*.



Gambar 10. ***Design ETL incremental Load*** **kasus data** ***Sales***

Dari model *flow* untuk *dynamic* ETL *incremental load* pada kasus data *sales* ini akan diimplementasikan pada *platform Data Integration* SSIS, perancangan model untuk *incremental load* ini memadukan teknik *timestamp* dan komprasi, *capture* berikut hasil implementasi dynamic ETL menggunakan SSIS.

Implementasi Incremental Load pada SSIS



Gambar 11. ***Implementasi incremental Load*** **pada SSIS**

Berdasarkan design ETL pada gambar 10 diimplementasikan menggunakan SSIS untuk *incremental load* yaitu mekanisme *insert , update* dan *delete* didetailkan pada proses *get Change update delete data* serta *insert data*.

PENUTUP

Incremental Load merupakan teknik ekstraksi yang digunakan dengan mengambil perubahan yang dilakukan pada sistem sumber yang akan diekstraksi dengan ekstraksi sebelumnya yang telah di lakukan, hal tersebut dikarenakan mempertimbangkan efisiensi dalam data *integration.* Jika hanya mengambil perubahan pada sistem sumber tentunya diharapkan dapat meminimalisir *loading time* data yang akan di integrasikan, dalam penelitian ini terkait analisis *incremental loading* yang dikaitkan dengan *issue* yang terjadi di perusahaan, banyak ditemukan beberapa persoalan diantaranya sumber data yang bervariasi, tidak hanya dalam struktur tabel namun juga berfariasi dari sisi *platform,* tidak ditemukan acuan atribut *timestamp* untuk menjalankan *incremental load* sehingga metode komparasi dilakukan sebagai alternatif, tidak adanya index pada tabel sumber sehingga berdampak *loading time* yang relatif lama.

Oleh karena itu proses integrasi membutuhkan *tools* untuk dapat mengakomodasi setiap perbedaan *platform* dan jenis data yang akan diintegrasikan serta persoalan lainnya. *Tools* SSIS memiliki *connector* yang dapat terintegrasi dengan *RDBMS, flat file, web service, SAP, osisoft PI* dan *platform* lainnya. Sejauh yang peneliti amati dan coba, *tools* SSIS mencover untuk kebutuhan integrasi dengan baik. Adapun terkait persoalan *loading time* , ini merupakan *open issue* yang menjadi target penelitian selanjutnya untuk dapat memaksimalkan *loading* time saat ETL dijalankan.

Dalam pendekatan konsep *incremental load* di SSIS mengikuti konsep MDA (*model driven Architecture*), dimana pengembangan perangkat lunak dapat sebagian atau sepenuhnya otomatis melalui penerapan transformasi model secara berurutan, mulai dari model yang mewakili spesifikasi sistem yaitu, model konseptual dan berakhir dalam model yang mewakili deskripsi rinci dari realisasi fisik.

DAFTAR PUSTAKA

[1] A. B. O. J. G. Philip Woodall, "Data Quality Problems in ETL: The State of the Practice in Large Organisations," *IOSR Journal of Computer Engineering (IOSR-JCE),* vol. 16, no. 1, p. 18, 2016.

[2] D. Laudenschlager, 06 2018. [Online]. Available: https://docs.microsoft.com /en-us/sql/integration-services/sql-server-integration-services?view=sql-server-2017.

[3] R. H. Junya, "Heterogeneous Database Synchronization Mechanism Based on ETL and XML," *RISTI,* p. 154, 2015.

[4] K. Kakish and T. Kraft, "ETL Evolution for Real-Time Data Warehousing," in *Proceedings of the Conference on Information Systems Applied Research*, New Orleans Louisiana, 2012.

[5] R. Kimball, L. Reeves, M. Ross, and W. Thornthwaite, The Data Warehouse Lifecycle Toolkit: Export Methods for Designing,Developing and Developing and Deploying Data Warehouses, Indiana: Wiley Publishing Inc, 1998.

[6] B. t. D. Warehouse, Inmon, W. H., Indiana: Wiley Publishing Inc, 1996.

[7] A. Jain, S. Garg, and N. Sharma, "The Management of Conformed ETL Architecture," *International Journal of Computer Application,* vol. 118, p. 20, 2015.

[8] D. Laudenschlager, G. Milener, D. Mabee, M. B, and Craig Guyer, "SQL Server Integration Services," 2018. [Online]. Available: https://docs.microsoft.com/en-us/sql/integration- services/sql-server-integration-services?view=sql-server-2017. [Accessed: 09-Dec-2018]

[9] P. Ponniah, *FUNDAMENTALS DATA WAREHOUSING FUNDAMENTALS A Comprehensive Guide for*, vol. 6. 2001.

[10] P. Bindal and P. Khurana, "ETL Life Cycle," *Citeseer*, vol. 6, no. 2, pp. 1787–1791, 2015.

[11] Lv, Junya, and H. Ren, "Heterogeneous database synchronization mechanism based on ETL and XML," *RISTI [Revista Iber. Sist. e Tecnol. Inf.*, vol. no. 17A, p. 153+., 2016.

[12] M. Thesis and A. Mashkoor, "Investigating Model Driven Architecture," 2004.

[13] Microsoft Team, "Integration Services Programming Overview." [Online]. Available: https://docs.microsoft.com/en-us/sql/integration-services/integration-services-programming-overview?view=sql-server-ver15.

[14] N. D. M. Hwy, "DAMA International DAMA International Handbook Revised : October 2009," no. October 2009.

[15] K. Krishnan, *Data Warehousing in the Age of Big Data*. Morgan Kaufmann, 2013.