Terakreditasi SINTA Peringkat 2
Surat Keputusan Dirjen Penguatan Riset dan Pengembangan Ristek Dikti No. 10/E/KPT/2019
masa berlaku mulai Vol. 1 No. 1 tahun 2017 s.d Vol. 5 No. 3 tahun 2021

Terbit online pada laman web jurnal: **http://jurnal.iaii.or.id**

JURNAL RESTI
(Rekayasa Sistem dan Teknologi Informasi)
Vol. 3 No. 2 (2019) 113 - 123 ISSN Media Elektronik: 2580-0760

Rancang Bangun Engine ETL *Data Warehouse* dengan Menggunakan Bahasa Python

I Made Suwija Putra[1], Dewa Komang Tri Adhitya Putra[2]
[1,2] Program Studi Teknologi Informasi, Fakultas Teknik, Universitas Udayana, Bali
[1]putrasuwija@unud.ac.id, [2]dewa.adhitya21@gmail.com

Abstract

Big companies that have many branches in different locations often have difficulty with analyzing transaction processes from each branch. The problem experienced by the company management is the rapid delivery of massive data provided by the branch to the head office so that the analysis process of the company's performance becomes slow and inaccurate. The results of this process used as a consideration in decision making which produce the right information if the data is complete and relevant. The right method of massive data collection is using the *data warehouse* approach. *Data warehouse* is a relational database designed to optimize queries in Online Analytical Processing (OLAP) from the transaction process of various data sources that can record any changes in data that occur so that the data becomes more structured. In applying the data collection, *data warehouse* has extracted, transform, and load (ETL) steps to read data from the Online Transaction Processing (OLTP) system, change the form of data through uniform data structures, and save to the final location in the *data warehouse*. This study provides an overview of the solution for implementing ETL that can work automatically or manually according to needs using the Python programming language so that it can facilitate the ETL process and can adjust to the conditions of the database in the company system.

Keywords: Manually, Automation, ETL, Python, *Data Warehouse*

Abstrak

Perusahaan besar yang memiliki banyak cabang dengan lokasi berbeda sering sekali mengalami kesulitan dalam menganalisis proses transaksi dari setiap cabang. Masalah yang dialami dari pihak manajemen perusahaan terletak pada kurang cepatnya penyampaian data yang massive diberikan oleh cabang ke kantor pusat sehingga proses analisa kinerja perusahaan menjadi lambat dan kurang akurat. Hasil proses analisa yang digunakan sebagai pendukung dalam pengambilan keputusan akan menghasilkan informasi yang tepat jika data yang tersedia lengkap dan relevan. Metode pengumpulan data massive yang baik dan handal digunakan adalah dengan menggunakan pendekatan teknologi *data warehouse*. *Data warehouse* merupakan basis data relasional yang didesain lebih kepada mengoptimalkan *query* dalam Online Analytical Processing (OLAP) dari proses transaksi berbagai macam sumber data dan dapat mencatat segala perubahan data yang terjadi sehingga menjadi lebih terstruktur. Dalam penerapan pengumpulan datanya, *data warehouse* mempunyai tahapan *extract*, *transform*, dan *load* (ETL) untuk dapat membaca data dari sistem Online Transaction Processing (OLTP), merubah bentuk data melalui penyamaan struktur data, dan menyimpan ke lokasi akhir di *data warehouse*. Penelitian ini memberikan gambaran solusi implementasi ETL yang bisa bekerja secara otomatis maupun manual sesuai dengan kebutuhan dengan menggunakan bahasa pemrograman Python sehingga bisa memudahkan dalam proses ETL dan bisa menyesuaikan dengan kondisi *database* di sistem perusahaan.

Kata kunci: Manual, Otomatisasi, ETL, Python, *Data Warehouse*

1. Pendahuluan

Perkembangan kebutuhan informasi bagi perusahaan dewasa ini semakin meningkat. Informasi memiliki peran yang sangat penting dalam menentukan kelangsungan hidup perusahaan. Informasi dihasilkan dari pengolahan berbagai data yang memiliki fakta dan arti yang kemudian disimpulkan. Dengan demikian ketersediaan dan penyajian data menjadi penting guna menghasilkan informasi yang lengkap, akurat, konsisten dan relevan [1]. Beberapa perusahaan besar yang memiliki banyak cabang tersebar di berbagai

Diterima Redaksi : 21-04-2019 | Selesai Revisi : 05-07-2019 | Diterbitkan Online : 01-08-2019

lokasi sering sekali mengalami kesulitan dalam mengumpulkan informasi mengenai proses transaksi dari setiap cabang. Proses pengambilan data transaksi dari masing – masing cabang ke kantor pusat dalam beberapa kasus sering dilakukan secara manual lewat email atau sms. Hal ini menyebakan kurangnya reliabelnya data yang diberikan oleh masing – masing kantor cabang (tidak adanya sinkronisasi data antara pusat dan kantor cabang)[2]. Ada beberapa penyebab permasalahan ini, salah satunya karena sumber daya manusia yang ada di masing – masing kantor cabang kurang memadai dalam hal pembuatan laporan transaksi yang rinci, real time dan mudah dianalisa oleh pusat atau belum adanya teknologi yang tepat diterapkan oleh perusahaan tersebut.

Alasan yang terakhir ini bisa dicarikan solusi yaitu dengan menggunakan teknologi yang dapat mengumpulkan data traksaksi yang bersifat historis atau lampau, kemudian menyimpannya secara terstruktur dalam sebuah media penyimpanan yang dinamakan dengan *data warehouse*. *Data warehouse* diharapkan dapat menjadi sumber kebutuhan informasi eksekutif dalam pengambilan keputusan bisnis dengan cepat [3].

Data warehouse merupakan teknologi menyimpan data yang berasal dari basis data relasional, dimana dalam pembuatannya lebih menitikberatkan dalam kemudahan melakukan query dan analisa proses dari data transaksi yang ada di OLTP. *Data warehouse* mengandung data historis dari proses transaksi dan bisa juga data dari sumber lainnya. *Data warehouse* memisahkan beban kerja analisis dari beban kerja transaksi sehingga memungkinkan konsolidasi data dari berbagai macam sumber dalam organiasasi [4].

Penggunaan *data warehouse* pada perusahaan bertujuan untuk memperbaiki proses pendataan pada suatu perusahaan sehingga perusahaan dapat mencatat segala perubahan data yang terjadi sehingga menjadi lebih terstruktur. Dalam penerapannya, *data warehouse* merupakan sebuah wadah sinkronisasi data transaksi dengan *data warehouse*, yang didalamnya akan terjadi proses penyamaan struktur data sehingga data transaksi dapat diterima oleh *data warehouse* [5]. Proses sinkronisasi tersebut dalam *data warehouse* disebut dengan proses ETL (*extract*, *transform*, dan *load*) yang merupakan sebuah jembatan antara *data warehouse* dengan data transaksi yang dimiliki perusahaan.

ETL merupakan proses mengumpulkan, menyaring, mengolah, dan menggabungkan data-data yang relevan dari berbagai sumber untuk disimpan ke dalam *data warehouse*. ETL juga dapat digunakan untuk mengintegrasikan data dari sistem yang sudah ada sebelumnya. Hasil proses ETL adalah data yang memenuhi kriteria *data warehouse*, seperti data historis, terpadu, terangkum, statis, dan memiliki struktur yang dirancang untuk keperluan proses analisis [6]. Proses ETL tersebut memerlukan sebuah tabel proses atau file JSON yang dimana berfungsi sebagai tabel padanan antara tabel OLTP dengan tabel *data warehouse*. Proses *load* ke dalam *data warehouse* akan mengambil dari tabel proses atau file JSON yang sudah berisikan data dari basis data sumber dan hal ini akan dijalankan sesuai dengan kapan eksekusi algoritma ETL itu dilakukan.

ETL secara umum bisa dijalankan dengan 2 cara yaitu bisa secara manual dan bisa secara otomatis melalui penjadwalan. Cara manual adalah menjalankan proses ETL dengan *user administrator* sebagai operator yang memustuskan untuk mejalankan *engine* ETL sedangkan cara otomatis adalah dengan membuat penjadwalan eksekusi proses ETL, hal ini memudahkan bagi *user administrator data warehouse* dalam merencanakan, mengelola dan melaksanakan proses pengambilan data dari OLTP [7].

Penelitian sebelumnya mengenai penerapan teknologi *data warehouse* di sebuah perusahaan seperti yang dilakukan oleh Rahmad Syah dengan judul publikasi Rancang Bangun *Data Warehouse* Untuk Analisis Strategi Produksi Penjualan Usulan : PT.XYZ [8]. *Data warehouse* yang dibangun menggunakan proses ETL yang masih melibatkan *administrator* sebagai operator yang mengerjakan proses kerja ETL atau menjadwalkan proses ETL dan arsitektur *data warehouse* dibangun dalam penelitian ini menggunakan pendekatan *top down*.

Saat ini banyak *tool* atau *engine* ETL komersial yang disediakan oleh beberapa *developer software*. *Tool* ETL yang terkenal menyediakan antarmuka pengguna grafis (GUI) dimana *user* bisa menentukan aliran data dari sumber ke tujuan secara visual. Meskipun ini mudah digunakan dan mudah memberikan visual proses ETL, ada juga kelemahan yang berhubungan dengan pemrograman grafis semacam ini dari program ETL. Beberapa masalah yang ditemui adalah sulit untuk mengekspresikan solusi mereka yang sesuai dengan komponen standar yang tersedia di *editor* grafis [9].

Oleh karena itu penelitian ini akan mencoba membangun *tool engine* ETL menggunakan bahasa pemrograman Python, dengan sifat *open source* sehingga bisa menyesuaikan dengan kondisi *database* sumber dan tujuan yang diinginkan oleh perusahaan. Cara eksekusi proses juga bisa secara manual dan otomatis. Pendekatan arsitektur pembangunan *data warehouse* dalam penelitian ini dilakukan secara *bottom up* artinya data warehouse dibentuk dari kumpulan *data mart*.

2. Metode Penelitian

Penelitian Rancang Bangun Engine ETL menggunakan Bahasa Python ini menggunakan konsep metode penelitian kuantitatif yaitu hasil yang didaptkan dari penelitian ini merupakan sebuah realitas dan variable

bisa diidentifikasi dan diukur dengan dengan alur penelitian seperti pada Gambar 1 di bawah ini.

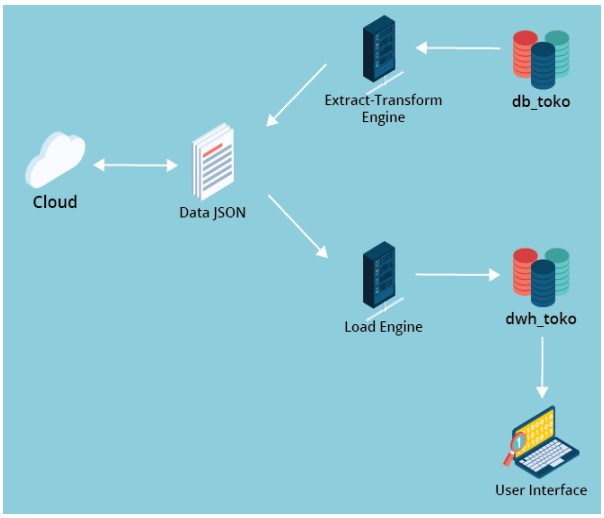
Penelitian dimulai dengan mengidentifikasi masalah yang ditemukan seperti diuraikan dalam sub bab pendahuluan. Berangkat dari itu, studi literatur diperlukan untuk memberikan solusi dari permasalahan. Dengan pengetahuan yang cukup, langkah selanjutnya adalah mendesain rancangan data multidimensional pada *data warehouse*, alur mesin ETL dan *graphic user inteface* yang kemudian diimplemenasikan ke dalam sebuah program yang dibangun dengan bahasa Python untuk mesin ETl dan bahasa PHP untuk sistem OLAP.

Tahapan selajutnya adalah melakukan uji coba terhadap sistem yang sudah dibangun, hingga tidak terjadi kesalahan proses ETL serta mendokumentasikan sistem.

Gambar 1 Alur penelitian

2.1 *Gambaran Umum Sistem*

Sistem ETL yang dibangun dalam penelitian ini mengambil contoh kasus pada toko elektronik komputer. Terdapat tahapan-tahapan dalam proses ETL mulai dari tahap insialisasi *database* sumber hingga menjadi data yang bisa dianalisis sesuai kebutuhan di dalam *data warehouse*. Gambaran umum sistem pada penelitian ini dapat dilihat pada Gambar 1.



Gambar 1. Gambaran Umum Sistem ETL *Data warehouse* dari database OLTP

Data pada *database* sumber OLTP dalam penelitian ini bernama *database* db_toko. Dari *database* ini dilakukan proses *extract* dan *transform*. Proses *extract* yaitu mengambil seluruh data yang diperlukan sesuai dengan inisialiasai tabel-tabel yang bersesuaian dengan tabel yang ada di *data warehouse* (dwh_toko). Data yang berhasil di-*extract* kemudian dilakukan penyamaan struktur dengan struktur pada *data warehouse* dan pembersihan yang disebut proses *transform*. Kedua proses tersebut dilakukan oleh satu *engine* yang bernama Extract-Transform Engine. *Engine* ini menghasilkan data *extract* yang telah di-*transform* dan tersimpan ke dalam file format JSON contoh seperti yang diuraikan pada sub bab 2.5. File JSON tersebut akan dibaca oleh *service load* di *data warehouse*. File JSON juga disimpan ke dalam *cloud server* secara otomatis, hal ini berfungsi sebagai pengamanan serta *backup file* dari hasil Extract-Transform Engine agar jika terjadi permaslahan pada server *data warehouse*, perusahaan masih punya backup data di luar.

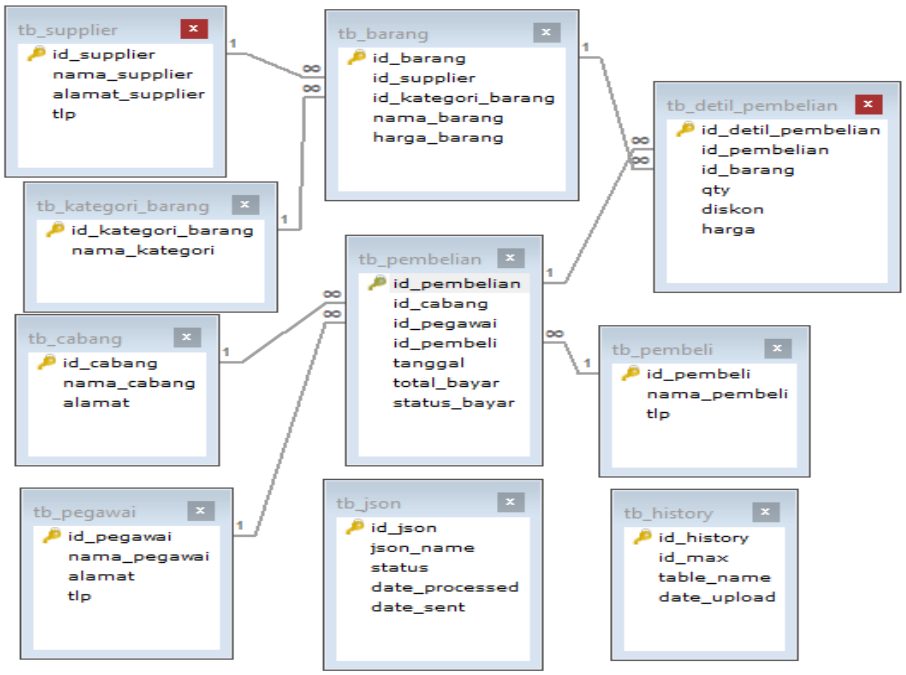
Langkah selanjutnya yaitu membaca file JSON yang berisi data hasil proses *extract* dan *transform* dengan memanggil *engine service* yang bernama Load Engine. *Engine* ini akan membaca isi *file* dan mempadankan data tersebut ke dalam struktur dimensional pada *data warehouse* yang sebelumnya sudah dirancang dengan menggunakan pendekatan Kimball (*bottom -up*).

Pendekatan Kimball dalam pembangunan *Data warehouse* dimulai dengan membangun tabel yang terdiri dari tabel fakta dan dimensi. Tabel fakta berisikan metrik yang biasa data recordnya berasal dari tabel transaksi di *relational database* dan tabel dimensi berisikan atribut yang data recordnya berasalah dari tabel master di *relational database*. Pendekatan ini tidak memperhatikan aturan normalisasi [10].

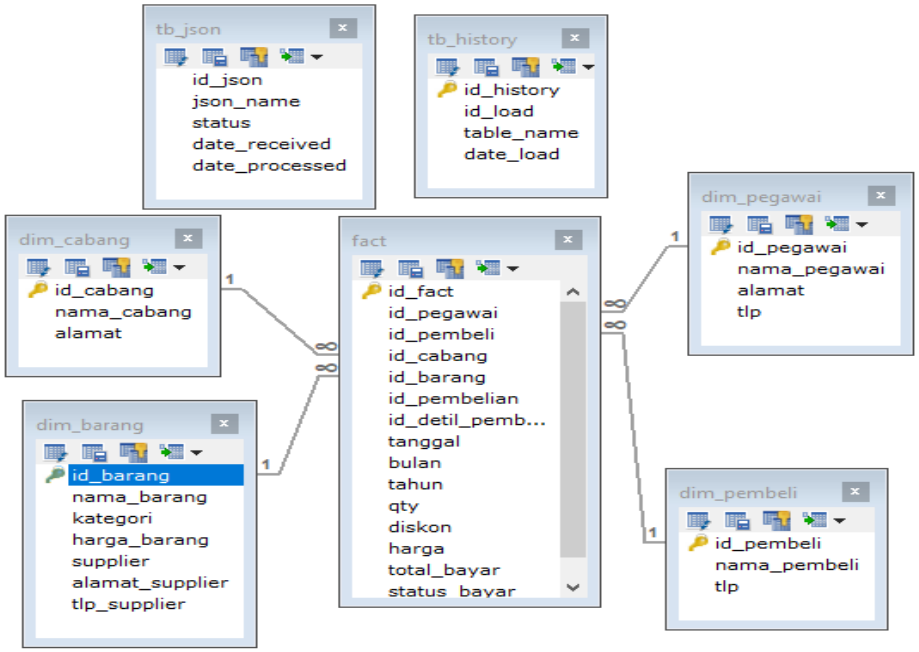
Proses ETL yang dibangun bisa dijalankan dengan 2 cara yaitu cara manual, dimana proses ETL akan berjalan jika *user administrator database* menekan tombol proses yang terdapat pada *interface engine* ETL. Cara yang kedua adalah proses otomatis yang dibuat khusus dengan konsep proses yang terjadwal sehingga proses ETL akan berjalan otomatis sesuai dengan jadwal yang sudah ditentukan sebelumnya.

Data yang telah berhasil disimpan pada *data warehouse* kemudian sudah siap untuk dianalisis sehingga bisa menghasilkan informasi dalam bentuk laporan sesuai dengan kebutuhan *user* pimpinan. Laporan-laporan tersebut seperti:

a. Laporan transaksi penjualan berdasarkan per bulan.
b. Laporan transaksi penjualan barang berdasarkan tiap cabang.
c. Laporan pegawai terhadap transaksi penjualan yang ditangani.



Gambar 2. Rancangan *database* percobaan db_toko



Gambar 3. Rancangan data multidimensional percobaan *data warehouse* dwh_toko

2.2 Rancangan Database OLTP

Data transaksi pada *database* OLTP db_toko menyimpan seluruh data transaksional pembelian di toko elektronik komputer. OLTP ini menggunakan konsep relasional *database*. Rancangan *database* db_toko dapat dilihat pada Gambar 2.

Database db_toko terdapat 10 tabel. Tabel tersebut yaitu tb_barang yang menyimpan data barang, tb_kategori_barang yang menyimpan kategori-kategori barang, tb_supplier yang menyimpan data para suplier barang, tb_pembeli yang menyimpan data para pembeli barang di toko, tb_cabang yang menyimpan data cabang yang dimiliki toko, dan tb_pegawai yang menyimpan data pegawai yang bersentuhan langsung terhadap transaksi di sistem. Data transaksi penjualan pada *database* ini tersimpan pada tabel tb_pembelian dan tb_detil_pembelian. Tabel lainnya yaitu tabel history yang digunakan untuk menyimpan log data yang sudah berhasil dibawa ke *data warehouse*, serta tabel json yang menyimpan konversi data menjadi *script* json.

2.3 Rancangan Data warehouse

Data hasil proses ETL nantinya akan disimpan ke dalam *data warehouse*. Rancangan struktur *data warehouse* yang dibangun yaitu menggunakan skema bintang. Skema tersebut terdiri dari tabel fakta dan tabel dimensi akan mengelilingi tabel fakta atau yang disebut dengan database multidimensional[11]. Rancangan *data warehouse* yang digunakan dapat dilihat pada Gambar 3.

Rancangan *data warehouse* yang berbentuk skema bintang dalam kasus ini terdiri dari 7 tabel dengan masing-masing 1 tabel fakta dan 4 tabel dimensi. Tabel tersebut yaitu fact, tb_fakta, dim_pegawai, dim_pembeli, dim_cabang, dan dim_barang. Tabel history dan tabel json digunakan masing-masing untuk menyimpan history data yang masuk dan menyimpan script json. Rancangan tersebut dapat dimanfaatkan untuk membentuk *data mart* yang sesuai dengan kebutuhan pengguna.

2.4 Tabel Bantu

Tabel bantu yang dimaksud di penelitian ini adalah tabel tambahan yang digunakan untuk menyimpan data riwayat (*history*) dari proses *extract-transform* ataupun *load* yang telah dilakukan. Terdapat 2 tabel bantu yaitu tabel tb_history dan tb_json. Tabel tb_history berfungsi menyimpan data log dari proses yang telah berhasil di-*extract, transform* ataupun *load*. Sedangkan tb_json berfungsi menyimpan nama *file* JSON yang berisikan hasil proses *extract* dan *transform* untuk dicatat dan dialokasikan untuk di upload ke *cloud* sebagai *backup file*.

2.5 Format JSON

File JSON dalam mesin ETL ini dipergunakan sebagai penyimpanan data dari hasil proses *extract-transform database* OLTP kemudian dipakai oleh mesin *Load Engine* untuk membawah data sehingga bermuara menuju ke masing-masing tabel dimensi dan tabel fakta yang ada di *data warehouse*. Di bawah ini adalah salah contoh format isian file JSON yang diambil dari data barang di OLTP yang akan bermuara ke dimensi barang di *data warehouse*.

JSON Data Barang

```
[
  {
    "id_barang": <<id_barang1>>,
    "nama_supplier": "<<nama_suplier1>>",
    "nama_kategori": "<<kategori1>>",
    "nama_barang": "<<nama_barang1>>",
    "harga_barang": <<harga1>>,
    "alamat_supplier": "<<alamat2>>"
  },

  {
    "id_barang": <<id_barang2>>,
    "nama_supplier": "<<nama_suplier2>>",
    "nama_kategori": "<<kategori2>>",
    "nama_barang": "<<nama_barang2>>",
    "harga_barang": <<harga2>>,
    "alamat_supplier": "<<alamat2>>"

  }
]
```

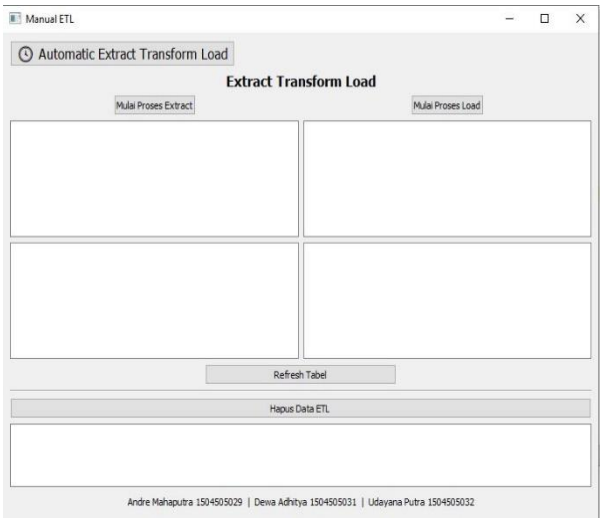
3 Hasil dan Pembahasan

Hasil dari penelitian ini adalah mengimplementasikan dan menguji mesin ETL yang telah dirancang pada bab sebelumnya untuk bisa memberikan kesimpulan apakah *engine* ETL sudah bisa bekerja sesuai dengan harapan. *Engine* ETL berkerja dengan 2 pilihan yaitu bisa secara manual proses atau secara otomatis dengan penjadwalan yang ditentukan sebelumnya.

3.1 Manual ETL

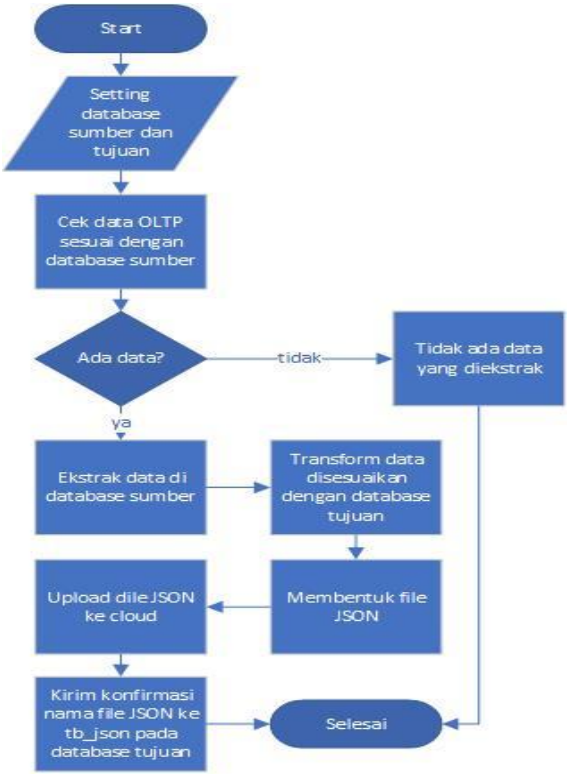
Manual ETL merupakan salah saru pilihan cara untuk menjalankan proses extract, transform dan load secara manual. Artinya engine ETL akan bekerja dengan bantuan user sebagai administrator untuk menjalankan service yang ada dalam Engine ETL. Tampilan interface pada pilihan Manual ETL dapat dilihat pada Gambar 4.

Gambar 4 merupakan tampilan *interface* pada *engine* ETL yang digunakan menjalankan proses *extract, transform* dan *load* secara manual. Terdapat button "Mulai Proses *Extract*", "Mulai Proses Load", "Refresh Tabel" dan "Hapus Data ETL". Sebelum menjalankan proses tersebut, *user* terlebih dahulu sudah menentukan database sumber (OLTP) dan tujuan (*Data warehouse*) yang dipakai sampai pada level tabel-tabel yang akan bersesuaian. Saat proses berlangsung *user* dapat melihat hasil dari proses *extract* dan *load* yang dilakukan.



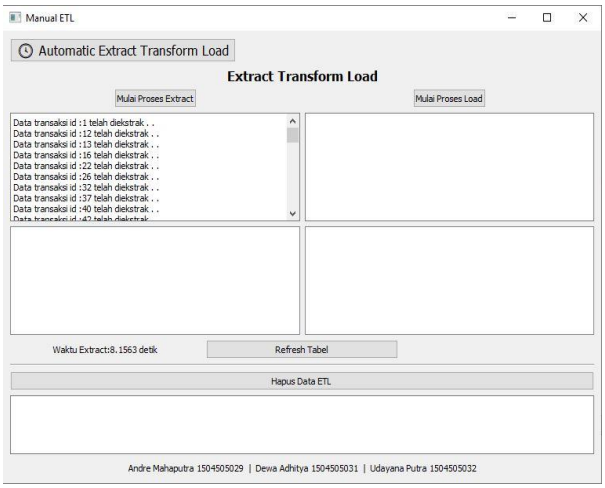
Gambar 4. GUI ETL (Tampilan Awal)

Proses ETL yang pertama adalah *extract* data yaitu mengambil data dari tabel yang ada di *database* sumber yang telah ditentukan sebelumnya. Kemudian proses *transform* (konversi dan pembersihan data) akan mengikuti aturan dari pasangan tabel antara tabel master dengan tabel dimensi dan tabel transaksi dengan tabel fakta di *data warehouse*. Kedua proses ini akan dieksekusi dengan memilih button "Mulai Proses Extract". Berikut adalah *flowchart extract* dan *transform* data yang mengambarkan alur tiap proses yang dijalani oleh Extract Transform Engine dapat dilihat pada Gambar 5.



Gambar 5. Flowchart Extract Transform Engine

Output dari Extract Transform Engine ini ditampilkan di dalam *interface* ETL. Hasil ini berisikan jumlah data yang berhasil diekstrak dan ditransformasikan dari tabel yang dipilih. Seperti yang terlihat pada Gambar 6.

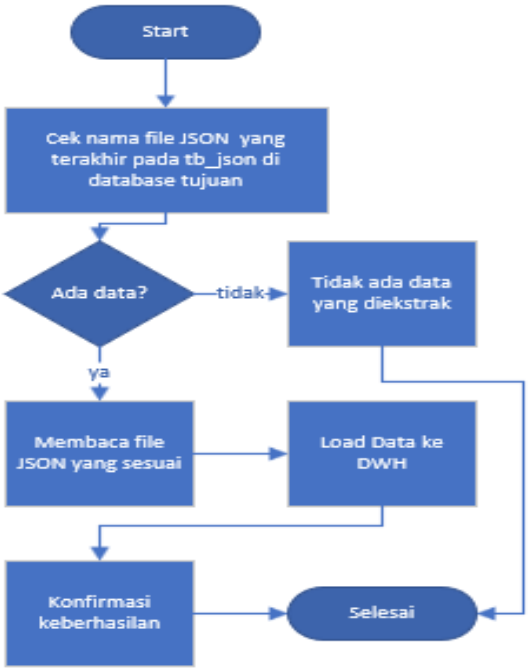


Gambar 6. *Interface* ETL *extract* data

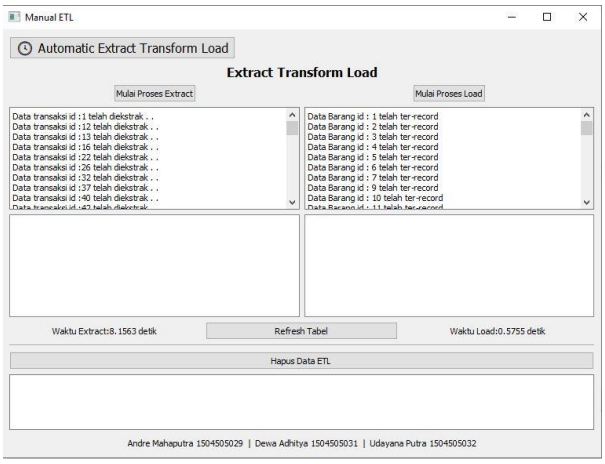
Terlihat pada Gambar 6, *output* yang ditampilkan pada *engine* ETL adalah sebuah *print text* "Data transaksi id: 1 telah diekstrak" yang menandakan bahwa proses *extract* dan *transform* data berhasil dilakukan untuk transaksi yang mempunyai id 1. Engine ETL juga menampilkan data waktu yang diperlukan saat melakukan proses *extract* dan *transform* seperti Gambar 6, terlihat waktu yang diperlukan untuk proses data. Contoh sebanyak 1000 data diperlukan waktu eksekusi sebesar 8.1563 detik. *Output* ini akan disimpan dalam bentuk file JSON yang secara otomatis akan diupload ke dalam *cloud server* sebagai cadangan data.

Setelah menjalankan proses *extract* dan *transform* data. *User* baru bisa melakukan proses *load* ke *data warehouse* yaitu dengan memilih tombol "Mulai Proses Load" di sebelah kanan *interface engine* ETL seperti pada Gambar 4. Tombol ini berfungsi untuk memasukkan data ke dalam skema *data warehouse* yang sudah dibuat. Adapun langkah prosesnya digambarkan pada *flowchart engine load* data yang dapat dilihat pada Gambar 7.

Gambar 7 merupakan alur dari proses Load Engine pada *engine* ETL. Proses diawali dengan membaca file JSON yang sudah dihasilkan oleh Extract Transform Engine sebelumnya, kemudian file JSON dieksekusi oleh Load Engine dengan menterjemahkan data, format, dan mengecek versi data kemudian melakukan *insert* ke tabel fakta dan tabel dimensi di dalam *data warehouse*. *Output* pada proses Load Engine akan ditampilkan di dalam *interface engine* ETL seperti yang terlihat pada Gambar 8.



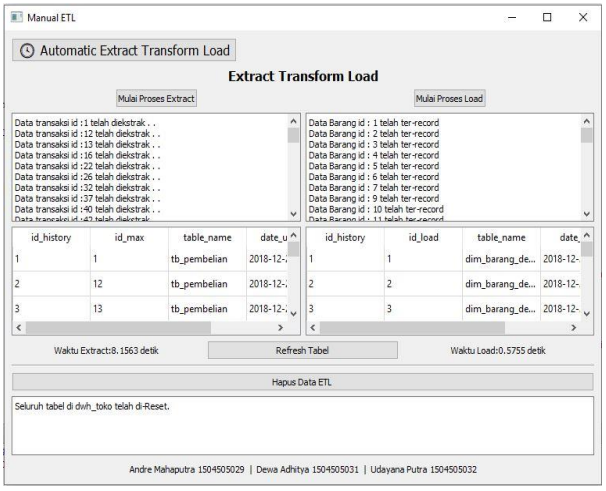
Gambar 7. Flowchart Load Engine



Gambar 8. *Interface* ETL *load* data

Gambar 8 merupakan tampilan *engine* ETL yang telah melakukan proses *load* data. *Output* yang ditampilkan pada *engine* ETL adalah sebuah *print text* "Data Barang id: 1 telah ter-record" yang menandakan bahwa proses *load* data pada record tersebut ke tabel *data warehouse* dengan id transaksi 1 telah berhasil dilakukan. *Engine* ETL juga menampilkan data waktu yang diperlukan saat melakukan proses *load* seperti pada Gambar 8 terlihat waktu yang diperlukan adalah 0.5755 detik untuk 1000 data.

Selain kedua *engine* tersebut di atas, juga terdapat fitur "Refresh Tabel" dan "Hapus Data ETL". Kedua fitur ini adalah fitur tambahan guna menunjang hal-hal yang bersifat teknis.



Gambar 9. Refresh Tabel dan Hapus Data ETL

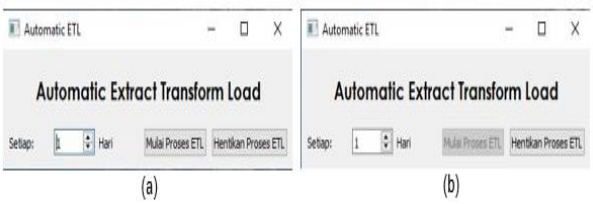
Gambar 9 merupakan tampilan *engine* ETL yang telah menampilkan *output* jika menekan button "Refresh Tabel". Output yang dihasilkan oleh proses "Refresh Tabel" adalah menampilkan data *history* untuk mengetahui data apa saja yang telah berhasil di *extract*, *transform* dan *load* dari sumber *database* pada proses terakhir dilakukan. terlihat pada Gambar 9, sebelah kiri menampilkan data *history* dari *database* OLTP yaitu tabel tb_history dan disebelah kanan menampilkan data *history* dari *data warehouse*.

Button "Hapus Data ETL" berfungsi menghapus atau mengkosongkan data pada *data warehouse*, kecuali data *history* yang tidak terhapus. Hal ini dengan asumsi bahwa data yang sebelumnya sudah dipindahkan ke arsip data. Hal tersebut berguna agar *data warehouse* yang aktif tidak mengalami *over load* dalam penyimpanan data karena faktor kapasitas penyimpanan sumber daya di setiap perusahaan berbeda-beda. Proses ini berhasil ditandai dengan menampilkan *output* berupa *print text* "Seluruh tabel di dwh_toko telah di-Reset" pada bagian bawah *interface* ETL seperti yang terlihat pada Gambar 9.

3.2 Automatic ETL

Automatic ETL merupakan sebuah fitur pengembangan alternatif dari Manual ETL di *engine* ETL yang digunakan untuk menjalankan proses *extract*, *transform* dan *load* secara otomatis. *Engine* ETL akan bekerja secara otomatis karena *service* akan menjalankan proses sesuai penjadwalan yang telah ditentukan.

Dalam *interface* ETL fitur ini bisa dijalankan dengan cara memilih *button* "Automatic Extract Transform Load" yang terdapat di pojok kiri seperti yang ada pada Gambar 4. Setelah itu akan muncul tampilan *interface* pada *engine* Automatic ETL, dapat dilihat pada Gambar 10 di bawah ini.



Gambar 10. Automatic ETL

Gambar 10 pada bagian (a) merupakan *interface* "Automatic *Extract Transform Load*" yang berisi konfigurasi mengenai jumlah hari. Input jumlah hari ini fungsinya adalah mengatur setiap berapa hari sekali yang diinginkan untuk melakukan proses *extract*, *transform* dan *load* secara otomatis. Gambar 10 pada bagian (b) merupakan tampilan *interface* Automatic ETL yang menandakan bahwa *service* proses ETL telah dijalankan, dan proses tersebut bekerja dibalik layar. User bisa juga menghentikan Proses ETL yang sudah dijadwalkan dengan memilih *button* "Hentikan Proses ETL" sehingga *engine* ETL akan berhenti bekerja secara otomatis.

3.3 Hasil Sistem OLAP

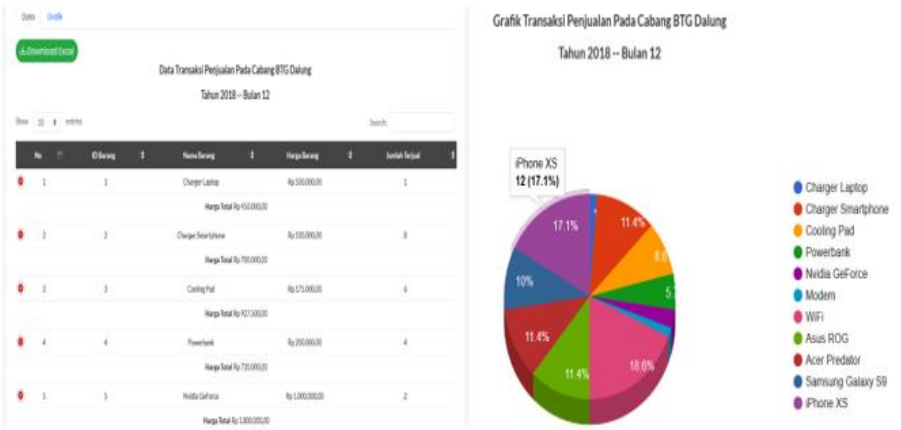
Penelitian ini juga membangun sistem analisa data yang disebut dengan OLAP (*Online Analytical Processing*) yang dimana ditujukan untuk pengguna manajemen level atas dalam perusahaan guna kebutuhan menunjang pengambilan suatu keputusan. Sistem ini menyediakan laporan-laporan seperti yang disebutkan pada sub bab 2.1. Gambar 11 merupakan tampilan awal pada aplikasi OLAP.

Gambar 11 merupakan tampilan *dashboard* pada aplikasi OLAP yang menampilkan total pemasukan pada seluruh cabang toko, informasi mengenai cabang dengan penjualan terbanyak, informasi mengenai barang terlaris dan informasi mengenai proses *load* ETL terakhir (versi data terakhir).

Aplikasi OLAP mengambil data dari *data mart* yang merupakan bagian penunjang dari *data warehouse*. Berikut ini adalah *data mart* yang tersedia pada penelitian yang merupakan hasil dari ETL yang sudah diproses dari sebuah database transakasi toko.

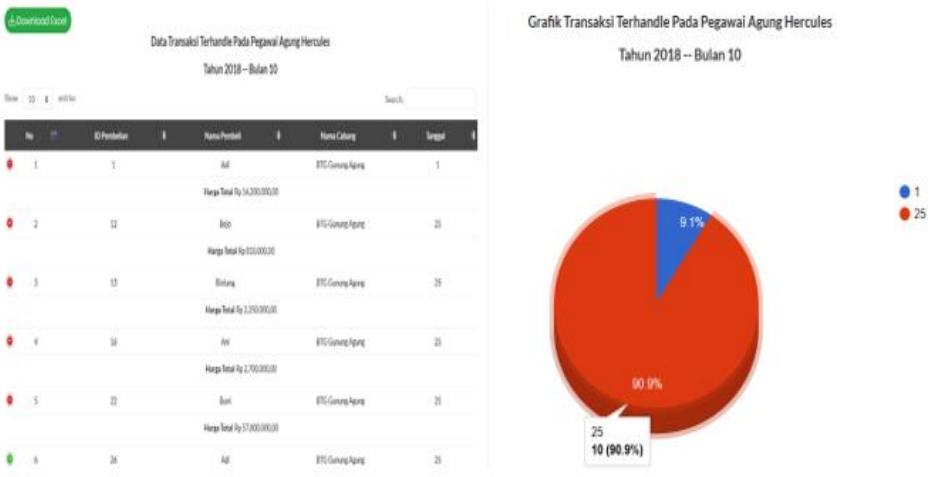
Gambar 11. Tampilan *dashboard* sistem OLAP yang berisikan rekapan data penjualan

Gambar 12. a. Hasil ETL berupa Data Mart Barang, b. Diagram Pie penjualan sebuah barang di beberapa cabang



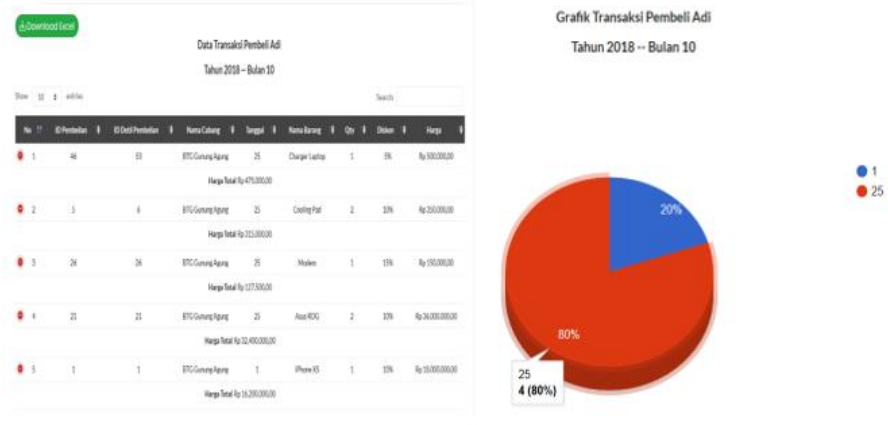
a. b.

Gambar 13. a. Hasil ETL berupa Data Mart Cabang, b. Diagram Pie penjualan barang di cabang tertentu



a. b.

Gambar 14. a. Hasil ETL berupa Data Mart Pegawai, b. Diagram Pie penjualan yang dilakukan oleh seorang pegawai



a. b.

Gambar 15. a. Hasil ETL berupa Data Mart Pembeli, b. Diagram Pie pemebelian yang dilakukan oleh seorang *customer*

a. Data Mart Barang

Data Mart Barang merupakan menu yang menampilkan informasi mengenai data penjualan barang dalam jangka wantu tertentu. Tampilan Data Mart Barang dapat dilihat pada Gambar 12.

Gambar 12 bagian (a) merupakan tampilan Data Mart Barang pada aplikasi OLAP yaitu menampilkan informasi mengenai penjualan barang pada waktu tertentu dengan memilih nama barang, tahun berapa, dan pada bulan ke berapa. Contoh pada tahun 2018 dan bulan ke 10 ada barang berupa Carger Laptop dengan harga Rp. 500.000 pada toko cabang xxx terjual sebanyak 19 dan total harga menjadi Rp. 9.500.000. Gambar 12 bagian (b) merupakan tampilan sebuah diagram pie yang memberikan informasi mengenai persentase penjualan Carger Laptop di toko cabang xxx pada tahun 2018 dan bulan ke 10. Dalam data mart ini juga terdapat fasilitas *export* data menuju ke format *excel.*

b. Data Mart Cabang

Data Mart Cabang merupakan menu yang menampilkan informasi mengenai data transaksi penjualan tiap cabang dalam jangka waktu tertentu. Tampilan Data Mart Cabang dapat dilihat pada Gambar 13.

Gambar 13 bagian (a) merupakan tampilan Data Mart Cabang pada aplikasi OLAP yang menampilkan informasi mengenai data transaksi penjualan tiap cabang dengan cara memilih nama cabang, selanjutnya memilih waktu pada tahun berapa dan bulan ke berapa. Contoh data transaksi pada tahun 2018 dan bulan ke 10 yaitu toko cabang xxx. Gambar 13 bagian (b) merupakan tampilan sebuah diagram pie yang memberikan informasi mengenai persentase data transaksi penjualan pada toko cabang xxx, seperti pada tahun 2018 dan bulan ke 10 yaitu Carger Laptop yang memiliki persentase terbanyak yaitu 17.6% yang menandakan bahwa transaksi Carger Laptop pada toko cabang xxx merupakan yang paling banyak diantara barang lainnya. Dalam data mart ini juga terdapat fasilitas *export* data menuju ke format *excel*

c. Data Mart Pegawai

Data Mart Pegawai merupakan menu yang menampilkan informasi mengenai data transaksi yang dilayani oleh pegawai dalam jangka waktu tertentu. Tampilan Data Mart Pegawai dapat dilihat pada Gambar 14.

Gambar 14 bagian (a) merupakan tampilan Data Mart Pegawai pada aplikasi OLAP yaitu menampilkan informasi mengenai data transaksi yang dilayani oleh pegawai perbulan dengan memilih nama pegawai, selanjutnya memilih pada tahun berapa dan bulan ke berapa. Contoh adalah data transaksi sebanyak 10 pembeli oleh pegawai dengan nama Agung Hercules pada tahun 2018 bulan ke 10. Gambar 14 bagian (b) merupakan tampilan sebuah diagram pie yang memberikan informasi mengenai persentase data transaksi yang dilayani oleh seluruh pegawai yaitu pada tahun 2018 bulan ke 10. Dalam data mart ini juga terdapat fasilitas *export* data menuju ke format *excel.*

d. Data Mart Pembeli

Data Mart Pembeli merupakan menu yang menampilkan informasi mengenai data transaksi pembeli pada jangka waktu tertentu. Tampilan Data Mart Pembeli dapat dilihat pada Gambar 15.

Gambar 15 bagian (a) merupakan tampilan Data Mart Pembeli pada aplikasi OLAP yaitu menampilkan informasi mengenai data transaksi pembeli perbulan dengan memilih nama pembeli, pada tahun berapa dan bulan ke berapa. Contoh data transaksi pembeli bernama Adi pada tahun 2018 bulan ke 10. Gambar 15 bagian (b) merupakan tampilan sebuah diagram pie yang memberikan informasi mengenai persentase data transaksi seluruh pembeli pada tahun 2018 bulan ke 10. Dalam data mart ini juga terdapat fasilitas *export* data menuju ke format *excel.*

4 Kesimpulan

Dari hasil implementasi dan pengujian sistem yang telah dilakukan dalam penelitian ini dapat disimpulkan bahwa dengan menggunakan bahasa pemrograman Python dapat dibangun *engine* ETL yang bisa bekerja secara manual maupun otomatis dengan menggunakan penjadwalan di dalamnya. *Engine* ETL yang dibangun telah mampu menjalankan proses *extract*, *transform* dan *load* dari *database* sumber OLTP (db_toko) menuju database tujuan *data warehouse* (dwh_toko) dengan menghasilkan waktu proses yang cepat sehingga ketersediaan data pada *data warehouse* perusahaan tersebut selalu terjaga. Ketersediaan data dalam *Data warehouse* sangat bergantung kepada kinerja *engine* ETL yang yang digunakan apakah sudah bekerja dengan stabil pada proses mengambil, menyamakan struktur dan menaruh data ke dalam *data warehouse*, diharapkan dengan penerapan *engine* ETL ini dalam suatu perusahaan dapat memberikan solusi persoalan dalam ketersediaan data guna menunjang proses pengambilan sebuah keputusan yang selama ini masih terkendala di perusahaan yang banyak memiliki cabang sehingga perusahaan ke depan bisa lebih kompetitif dengan perusahaan lainnya.

Daftar Rujukan

[1] J. O'Brien and G. Marakas, *Management Information Systems*, Tenth Edit. 2011.

[2] S. Lim, "*Data warehouse* Untuk Pengelolaan Penjualan Pada Pt. Lippo Karawaci, Tbk.," *J. Ilm. SISFOTENIKA*, 2012.

[3] F. Y. Al Irsyadi, "Implementasi *Data warehouse* dan Data Mining untuk Penentuan Rencana Strategis Penjualan Batik (Studi Kasus Batik Mahkota Laweyan)," *KomuniTi*, 2014.

[4] T. Connolly and C. Begg, *Pearson Database Systems A Practical Approach to Design Implementation and*

Management 6th Global Edition. 2014.

[5] A. Simitsis, P. Vassiliadis, T. Sellis, C. Of, and I. Of, "Modeling and Optimization of Extraction-Transformation-Loading (ETL) Processes in *Data warehouse* Environments," 2004.

[6] K. Haryono, "Penerapan *data warehouse* dalam pengelolaan sistem keuangan daerah," *J. Warehouse.*, vol. 1, pp. 1–9, 2005.

[7] L. Muñoz, J.-N. Mazón, and J. Trujillo, "Automatic generation of ETL processes from conceptual models," in *Proceeding of the ACM twelfth international workshop on Data warehousing and OLAP - DOLAP '09*, 2009.

[8] R. Syah, "Rancang Bangun *Data warehouse* untuk Analisis Strategi Produksi Penjualan Usulan : PT.XYZ," *TECHSI - J. Penelit. Tek. Inform.*, 2014.

[9] C. Thomsen and T. B. Pedersen, "pygrametl: a powerful programming framework for extract-transform-load programmers," in *DOLAP 09 Proceeding of the ACM twelfth international workshop on Data warehousing and OLAP*, 2009, pp. 49–56.

[10] R. Kimball and J. Caserta, *The Data warehouse ETL Toolkit*. 2014.

[11] D. Schuff, K. Corral, and O. Turetken, "Comparing the understandability of alternative *data warehouse* schemas: An empirical study," *Decis. Support Syst.*, vol. 52, no. 1, pp. 9–20, Dec. 2011.