Jurnal Teknologi Informasi dan Ilmu Komputer (JTIIK) DOI: 10.25126/jtiik.202296833
Vol. 9, No. 7, Desember 2022, hlm. 1741-1746 p-ISSN: 2355-7699
Akreditasi KEMENRISTEKDIKTI, No. 36/E/KPT/2019 e-ISSN: 2528-6579

SISTEM PROPAGASI ANOTASI PADA METADATA *LINEAGE* UNTUK MANAJEMEN *DATA WAREHOUSE*

Dion Ricky Saputra[1], Welly Purnomo*[2], Nanang Yudi Setiawan[3]

[1,2,3]Universitas Brawijaya, Malang
Email: [1]dion_ricky@student.ub.ac.id, [2]wepe@ub.ac.id, [3]nanang@ub.ac.id
*Penulis Korespondensi

(Naskah masuk: 22 Desember 2022, diterima untuk diterbitkan: 27 Desember 2022)

Abstrak

ETL (*extract*, *transform*, dan *load*) merupakan proses yang dilibatkan dalam pembuatan dan manajemen *data warehouse*. Desain ETL dibuat menyesuaikan struktur sumber data dan *data warehouse*. Dengan adanya ketergantungan tersebut maka perubahan di sumber data bisa berdampak besar terhadap desain ETL. Ketika perubahan tersebut terjadi, pengelola ETL akan berkomunikasi dengan pemilik data untuk mengetahui rincian perubahan struktur data dalam rangka memperbaiki desain ETL. Aliran komunikasi ini akan semakin meningkat sejalan dengan jumlah sumber data yang digunakan. Semakin banyak sumber data yang diproses maka komunikasi tersebut berpotensi menjadi *bottleneck*. Informasi perubahan struktur data ini dapat dikomunikasikan melalui anotasi yang dilekatkan pada sumber data. Anotasi tersebut kemudian dipropagasikan sehingga dapat digunakan untuk memperbaiki rancangan ETL. Dengan menggunakan anotasi, harapannya aliran komunikasi antara pengelola ETL dengan pemilik data dapat berkurang. Permasalahan tersebut menunjukkan seberapa penting dikembangkannya sistem propagasi anotasi. Sistem propagasi anotasi tersusun atas tiga komponen yaitu ekstraksi metadata, propagasi anotasi, dan *adapter*. Pengujian sistem dilakukan menggunakan teknik *blackbox* dan *user acceptance testing* bersama pengguna akhir. Pengujian *blackbox* menghasilkan 30 kasus uji yang hasilnya valid. Hasil evaluasi *user acceptance testing* menunjukkan bahwa rata-rata pengguna menyatakan sangat setuju dengan sistem yang dikembangkan.

Kata kunci: *anotasi, lineage, propagasi anotasi, metadata, manajemen data*

ANNOTATION PROPAGATION SYSTEM ON LINEAGE METADATA FOR DATA WAREHOUSE MANAGEMENT

Abstract

ETL is a process of extracting, transforming, and loading data that is involved in creation and management of a data warehouse. Since ETL is deeply connected to the structure of the source data, if a small changes happens to that structure then the whole workflow might stop. Because one data source can be used by more than one ETL workflow, the impact of schema changes to the ETL design are enormous. When such incident happens, the ETL designer will ask the data owner for the details of the schema changes. The communication traffic between the ETL designer and the data owner will increase as the number of sources that are being used is increasing. Therefore potentially becoming a bottleneck. Information regarding schema changes of a data source can be attached as an annotation. This annotation will be then propagated so that the ETL manager can update the workflow according to the recorded changes.Using this technique, the communication traffic between the ETL designer and the data owner can be minimized. This problem highlights the need of an annotation propagation system. The system itself consists of three components: metadata extraction, adapter, and annotation propagation. To test the system, blackbox and user acceptance testing is used. The blackbox testing resulting with 30 test case which are all valid. The user acceptance testing is done with the end-user directly operating the system, and after analyzing the results it shows that on average the user is accepting the system.

Keywords: *annotation, lineage, annotation propagation, metadata, data management*

1. PENDAHULUAN

Data yang berada di *data warehouse* berasal dari beragam sumber data baik didalam maupun diluar organisasi. Data yang pada mulanya berada di sumber data akan dipindahkan melalui proses ekstraksi, transformasi, dan *loading* atau disebut juga dengan proses ETL. Aktivitas ekstraksi dan transformasi data

pada proses ETL sangat bergantung pada struktur data di sumbernya, sehingga sedikit perubahan yang terjadi pada sturuktur data akan menimbulkan efek yang besar terhadap desain proses ETL. Terlebih lagi, satu sumber data mungkin digunakan pada lebih dari satu ETL sehingga semakin meningkatkan kompleksitas penanganan perubahan struktur data.

Pada saat terjadi insiden kegagalan eksekusi ETL yang diakibatkan oleh perubahan skema, pengelola ETL akan berkomunikasi dengan pemilik data untuk mengetahui rincian perubahan yang terjadi dalam rangka memperbaiki ETL. Namun aliran komunikasi ini akan semakin meningkat seiring dengan bertambahnya jumlah sumber data yang digunakan. Kondisi ini semakin diperparah apabila perusahaan menggunakan data yang berasal dari luar organisasi. Pada akhirnya, komunikasi ini malah menjadi penghambat dalam proses bisnis penanganan perubahan.

Penyampaian informasi perubahan skema data tersebut dapat dioptimalkan menggunakan bantuan anotasi. Anotasi dapat diisi dengan keterangan perubahan struktur data yang terjadi pada sumber. Kemudian berdasarkan informasi yang direkam tersebut, pengelola ETL dapat memperbaiki rancangan ETL menyesuaikan dengan perubahan yang terjadi. Saat ini, belum ada alat ETL yang memungkinkan pemindahan anotasi dari sumber data. Oleh karena itu diperlukan adanya sistem yang dapat mempropagasikan anotasi.

Penelitian yang dilakukan oleh Bhagwat et al (2005), Chiticariu et al (2005), Geerts et al (2006), dan Lu et al (2016) mengembangkan sistem manajemen anotasi yang dapat mengelola dan mempropagasikan anotasi sesuai jalur transformasinya. Permasalahan yang ada pada sistem-sistem tersebut adalah propagasi anotasi yang terbatas. Keterbatasan yang dimaksud adalah penggunaan satu jenis *database* yang spesifik, misalnya Postgres pada penelitian Bhagwat et al (2005) dan Chiticariu et al (2005) dan Hadoop pada penelitian Lu et al (2016). Keterbatasan tersebut muncul dikarenakan tidak adanya informasi yang jelas terkait jalur perpindahan dan transformasi data.

Propagasi anotasi sulit dilakukan karena adanya lapisan transformasi sebelum data masuk ke *data warehouse*. Dimana lapisan transformasi itu seringkali tidak dapat ditelusuri jalurnya sehingga mengakibatkan terputusnya hubungan logika yang seharusnya ada antara sumber data dengan *data warehouse*.

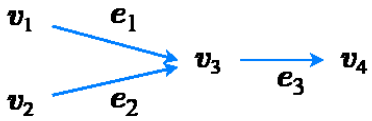
Jalur transformasi yang dilalui saat data dipindahkan merupakan definisi dari metadata *lineage*. *Lineage* dapat merekam alur perjalanan data mulai dari sumber yang paling utama (genesis) hingga ke tujuannya baik di *data warehouse* maupun penyimpanan sementara seperti *staging*. Dengan adanya *lineage*, permasalahan yang ditimbulkan akibat tidak adanya jalur perpindahan data untuk mempropagasikan anotasi dapat diatasi.

Penelitian ini mengembangkan sistem propagasi anotasi berbasis *lineage*. Anotasi dari sumber data dipropagasikan mengikuti jalur perpindahan data yang terekam dalam *lineage*. Dengan dikembangkannya sistem ini maka dapat mengurangi beban komunikasi antara pengelola ETL dengan pemilik data dan meminimalisir peluang terjadinya *bottleneck*.

2. PERMASALAHAN DALAM PROPAGASI ANOTASI

Sulitnya melakukan propagasi anotasi dengan tepat diakibatkan karena tidak adanya informasi jalur perpindahan data. Lapisan transformasi yang dilakukan pada data dengan mengubah struktur data dan menggabungkan satu *dataset* dengan *dataset* lainnya melalui operasi seperti JOIN atau UNION juga menjadi masalah. Terlebih lagi, transformasi yang dilakukan pada data tidak terbatas pada transformasi berbasis *query* saja. Operasi transformasi dapat dilakukan menggunakan beberapa cara seperti *copy-paste*, SPJU (*select-project-join-union*), hingga *black box* (Ikeda & Widom, 2009).

Permasalahan terkait tidak tersedianya informasi jalur perpindahan data ini dapat diselesaikan dengan adanya metadata *lineage*. *Lineage* atau yang disebut juga dengan *provenance*, *pedigree*, *parentage*, *genealogy*, dan *filiation* memiliki beragam makna dilihat dari penggunaannya. Menurut Buneman et al (2000), *lineage* merujuk pada deskripsi dari asal usul dan proses suatu data menuju ke *database*. Ikeda & Widom (2009) menyebutkan, *lineage* merujuk pada informasi darimana data berasal, bagaimana data ditransformasikan, dan bagaimana data tersebut diubah seiring waktu. Simmhan et al (2005) mendefinisikan *lineage* sebagai riwayat derivatif dari data atau *dataset* yang dimulai dari sumber aslinya. Pada konteks penelitian ini, definisi *lineage* adalah informasi terkait darimana data berasal dan bagaimana data tersebut ditransformasikan.



Gambar 1. Representasi *lineage* sebagai DAG

Biasanya *lineage* direpresentasikan dalam bentuk *directed acyclic graph* (DAG), atau graf berarah tanpa siklus. Gambar 1 menunjukkan *lineage* dari *dataset* v_4 dalam bentuk DAG dengan 4 *vertex* v_1, v_2, v_3, dan v_4 dan 3 *edge* e_1, e_2, dan e_3. Setiap *vertex* merepresentasikan suatu *dataset* misalnya tabel, *file*, atau *collection* (di basis data NoSQL). Setiap *edge* merepresentasikan operasi transformasi yang dilakukan pada data. Arah *edge* menunjukkan arah perpindahan data.

Dalam mempropagasikan anotasi suatu *dataset*, pertama perlu dicari tahu genesis dari *dataset*

tersebut. Genesis didefinisikan sebagai *dataset* sumber yang bukan merupakan turunan dari *dataset* lain. Pada contoh *lineage* Gambar 1 sebelumnya, dapat diketahui bahwa genesis dari v_4 adalah v_1 dan v_2. Setelah diketahui genesis dari *dataset*, maka selanjutnya anotasi dari semua genesis tersebut akan diambil dan digabungkan. Masih menggunakan contoh yang sama seperti sebelumnya, anotasi dari v_4 adalah anotasi v_1 dan v_2. Teknis penggabungan anotasi akan dijelaskan pada bagian perancangan.

Setiap *dataset* pada *lineage* pasti memiliki setidaknya satu genesis. Genesis dari *dataset* genesis adalah *dataset* itu sendiri. Sehingga, genesis dari v_1 adalah v_1 dan aturan yang sama berlaku pula untuk v_2.

Setelah diketahui cara mempropagasikan anotasi dengan menggunakan *lineage*, permasalahan selanjutnya adalah bagaimana cara mengetahui *lineage* tersebut. Pada sistem *data warehousing*, biasanya digunakan *tools* ETL terentu seperti: Talend, Pentaho, Apache Beam, atau Apache Airflow. Pada penelitian ini digunakan *tools* ETL Apache Airflow karena merupakan perangkat lunak *open source* yang sudah banyak digunakan oleh perusahaan-perusahaan di seluruh dunia (HG Insights, 2022).

Ekstraksi *lineage* di Airflow sudah cukup dimudahkan karena representasi DAG yang melakukan proses ETL sudah disimpan di basis data Airflow. Tetapi DAG yang disimpan hanya memuat nama dan jenis *task* serta *parameter*-nya saja. Hanya dengan informasi ini masih belum cukup untuk dapat mengetahui *lineage* karena rekonstruksi *edge* membutuhkan informasi tentang *input* dan *output* apa yang dihasilkan dari setiap *task*.

Oleh sebab itu, sistem yang dikembangkan setidaknya harus dapat melakukan dua fungsi, yaitu fungsi ekstraksi *lineage* dan propagasi anotasi. Selanjutnya akan dibahas hasil analisis identifikasi komponen dan rancangan sistem untuk dapat mempropagasikan anotasi berdasarkan *lineage*.

3. IDENTIFIKASI KOMPONEN SISTEM

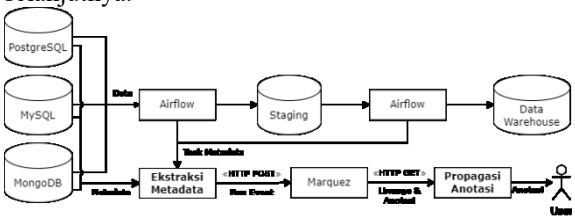
Sistem propagasi anotasi dikomposisikan oleh satu atau lebih komponen penyusun. Identifikasi komponen perlu dilakukan untuk menentukan secara spesifik kewajiban fungsionalitas setiap komponen sistem. Dalam melakukan identifikasi komponen, perlu diketahui bahwa sistem yang dikembangkan tidak dapat berdiri sendiri. Perlu adanya sistem pendukung yang memungkinkan sistem untuk berfungsi dengan baik.

Sistem pendukung tersebut adalah Airflow untuk menjalankan ETL dan Marquez untuk mengelola *lineage*. Marquez dipilih sebagai piranti manajemen *lineage* karena merupakan perangkat lunak *open source* dan menggunakan standar API OpenLineage yang juga didukung oleh banyak penyedia layanan *data governance*.

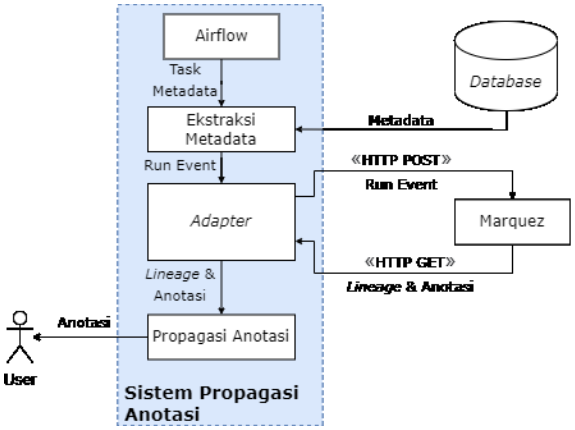
Berdasarkan ketergantungan terhadap dua sistem pendukung tersebut maka dapat dilakukan identifikasi komponen penyusun sistem. Setelah dianalisis, diketahui bahwa ada tiga komponen sistem yaitu ekstraksi metadata, propagasi anotasi, dan *adapter*. Komponen ekstraksi metadata berperan untuk mengambil dan memproses metadata eksekusi *job* ETL di Airflow dalam bentuk *task*. Selain itu komponen ini juga mengambil metadata dari *dataset* yaitu anotasi dan skema data seperti kolom serta tipe datanya. Komponen *adapter* bertugas untuk menjembatani interaksi antara komponen sistem dengan Marquez. Komponen propagasi anotasi, sesuai namanya, bertugas untuk mempropagasikan anotasi dengan terlebih dahulu melacak dan mengambil anotasi *dataset* berdasarkan *lineage*-nya.

4. RANCANGAN SISTEM

Dalam rangka mendukung berjalannya sistem, maka dibuatlah replika arsitektur *data warehouse* (Gambar 2). Basis data yang digunakan adalah PostgreSQL, MySQL, dan MongoDB. Detail alur pemrosesan metadata tersebut akan dibahas selanjutnya.



Gambar 2. Arsitektur *data warehousing*



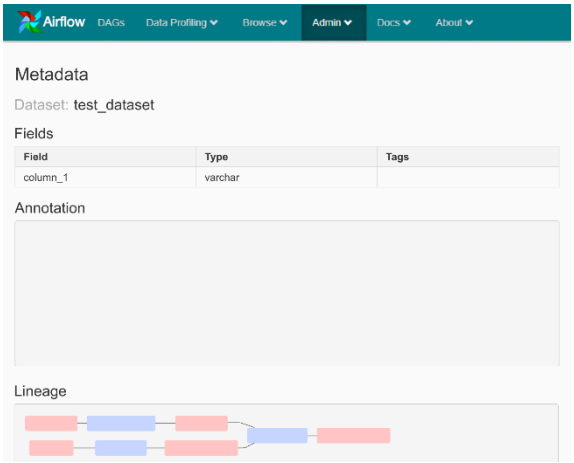
Gambar 3. Gambaran interaksi sistem propagasi anotasi

Pada Gambar 2 hanya ditunjukkan dua komponen saja tanpa memperlihatkan komponen *adapter*. Untuk memberikan gambaran interaksi yang lebih lengkap dan jelas maka dapat merujuk pada Gambar 3.

Aktivitas pertama yang dilakukan oleh sistem dimulai dari komponen ekstraksi metadata. Komponen ini dirancang untuk dijalankan secara otomatis dan terjadwal mengikuti jadwal ETL data. Komponen ini menyediakan *Operator* yang dapat digunakan oleh Airflow untuk berinteraksi dengan fungsi internal komponen. Saat dijalankan oleh

Airflow, komponen ekstraksi metadata akan mengambil dan memproses informasi seperti nama *task*, waktu eksekusi, status, masukan (*input*) yang diproses, dan keluaran (*output*) yang dihasilkan oleh suatu *task* dari Airflow. Apabila *task* tersebut melakukan ekstrasi atau transformasi data, maka pada waktu yang sama komponen ini juga akan mengambil metadata dari *dataset* yang menjadi *input* dan/atau *output*. Keluaran dari komponen ekstraksi metadata kemudian dikirim ke *adapter* untuk selanjutnya diserialisasi sehingga dapat dikirim melalui HTTP POST menuju ke Marquez.

Berbeda dengan komponen ekstraksi metadata, komponen propagasi anotasi tidak dieksekusi secara rutin atau terjadwal. Komponen ini hanya akan dijalankan ketika pengguna akhir (*user*) mengakses anotasi dari suatu *dataset* melalui antarmuka atau ketika ada sistem lain yang berinteraksi dengan menggunakan kode program. Ketika dijalankan, komponen mengambil *lineage* yang disimpan di Marquez melalui *adapter*. Seperti yang telah dibahas sebelumnya, anotasi dari *dataset* adalah gabungan dari anotasi semua genesisnya. Penggabungan anotasi yang dimaksud adalah penulisan anotasi dalam format JSON *string*, kemudian anotasi dari masing-masing genesis dimasukkan ke nama *key* yang sesuai berdasarkan nama *dataset*. Misalnya, jika genesis dari v_4 adalah v_1 dan v_2 maka anotasi dari v_4 akan memiliki dua *key* di tingkat paling luar yaitu ”dataset:v1” dan ”dataset:v2”.



Gambar 4. Rancangan antarmuka sistem

Penggunaan beberapa jenis basis data yang berbeda mengharuskan adanya format penulisan anotasi yang seragam. Format anotasi diperlukan untuk menentukan bagaimana anotasi disimpan dan dikelola di basis data. Untuk mendukung banyak jenis basis data yang berbeda maka diputuskan penggunaan format JSON *string*. Tujuannya adalah agar anotasi dapat disimpan dalam bentuk teks tetapi tetap memiliki struktur yang dapat dibaca oleh sistem lain menggunakan *library parser* JSON.

Pengguna akhir dapat mengakses sistem propagasi melalui antarmuka seperti pada Gambar 4. Antarmuka tersebut menampilkan beberapa metadata seperti skema data yaitu nama kolom serta tipe datanya, anotasi yang telah dipropagasikan, dan *lineage* dari *dataset*.

5. IMPLEMENTASI SISTEM

Komponen ekstraksi metadata ketika dijalankan oleh Airflow akan memanggil kelas AirflowExtractor untuk memulai ekstraksi metadata. Kelas AirflowExtractor, selanjutnya disebut sebagai *manager*, kemudian mengambil daftar *task* yang telah dijalankan Airflow beserta TaskInstance-nya. TaskInstance ini berisi rincian status eksekusi *task* ETL. *Manager* kemudian memanggil kelas turunan dari *superclass* BaseMetadataExtractor untuk memproses setiap *task* yang sesuai. Kelas turunan dari BaseMetadataExtractor ini selanjutnya disebut sebagai MetadataExtractor. MetadataExtractor akan melakukan ekstraksi dengan caranya masing-masing, tetapi pada dasarnya metadata yang diambil berkaitan dengan identitas *task* seperti nama atau ID, status, *input*, *output*, dan metadata *dataset* seperti anotasi dan skema data. MetadataExtractor juga mungkin melakukan *parsing* SQL apabila transformasi yang digunakan adalah *query*. *Parsing* SQL dilakukan untuk dapat mengetahui dengan tepat *endpoint* (*vertex input* dan *output*) dari *edge* (operasi transformasi). Metadata yang telah diambil kemudian diproses lebih lanjut oleh *manager* sebelum akhirnya dikirim ke *adapter*.

Implementasi komponen *adapter* dalam bentuk *library* sehingga dapat digunakan oleh komponen ekstraksi metadata maupun komponen propagasi anotasi. Komponen ini menggunakan standar API OpenLineage, sehingga komponen ini juga dapat bertukar data dengan sistem lain yang menggunakan standar yang sama.

Komponen propagasi anotasi diimplementasikan kedalam beberapa *class* kode program, diantaranya adalah *class* LineageParser dan Annotation. *Class* LineageParser bertugas untuk membaca dan mengolah *lineage* agar dapat ditelusuri dengan mudah. Saat dijalankan, *class* Annotation akan meminta *lineage* dari *dataset* dari Marquez melalui *adapter*. Kemudian *lineage* tersebut akan diproses oleh LineageParser. *Class* Annotation kemudian melacak genesis dengan cara menelusuri *lineage*. Setelah diketahui genesis yang sesuai maka dilanjutkan dengan pengambilan anotasi dari Marquez. Anotasi-anotasi yang diperoleh kemudian digabungkan sebelum akhirnya dikembalikan sebagai *output*.

Antarmuka diimplementasikan dalam bentuk *plugin* Airflow. Konsep *plugin* pada Airflow ini dapat disebut sebagai ekstensi. Oleh karena itu, antarmuka sistem dapat diakses melalui *webserver* yang sama yang menangani Airflow. Dengan begitu tidak diperlukan adanya *webserver* tambahan hanya untuk menampilkan antarmuka sistem propagasi.

Tabel 1. Hasil Kuesioner UAT

No.	Pernyataan	Frekuensi				Rata-rata Terbobot ($\bar{x}$)
		STS	TS	S	SS	
1.	Sistem dapat mengekstraksi *lineage* data dengan tepat			1	6	3,86
2.	Sistem dapat mengekstraksi anotasi data dengan tepat			4	3	3,43
3.	Sistem dapat mempropagasikan anotasi dengan tepat			2	5	3,72
4.	Sistem propagasi anotasi dapat digunakan sebagai bentuk dokumentasi data			2	5	3,72
5.	Sistem propagasi anotasi memiliki antarmuka yang mudah dipahami			3	4	3,56
6.	Sistem propagasi anotasi memiliki antarmuka yang mudah diakses			3	4	3,56
7.	Sistem propagasi anotasi memiliki data yang terbaru			4	3	3,43

Plugin yang dibuat terdiri dari *controller* yang ditulis menggunakan bahasa pemrograman Python dan *view* menggunakan HTML. Tampilan antarmuka yang menampilkan metadata *dataset* dapat dilihat pada Gambar 5.

Gambar 5. Antarmuka sistem

6. PENGUJIAN

Pengujian dilakukan dengan teknik *blackbox* dan *user acceptance testing* (UAT). Pengujian *blackbox* dilakukan dalam tiga tahap atau tingkatan yaitu pengujian unit, integrasi, dan sistem. Pengujian UAT dilaksanakan oleh pengguna akhir yang berinteraksi dengan sistem secara langsung.

Pada pengujian unit dilakukan pengujian pada tingkat *method* yang berkaitan dengan pengelolaan metadata dan SQL *parsing*. Hasil pengujian unit diperoleh 25 kasus uji yang semuanya memiliki status valid. Kemudian dilakukan pengujian integrasi untuk menguji interaksi antar komponen. Diperoleh tiga kasus uji yaitu dua kasus uji ekstraksi metadata untuk menguji integrasi komponen ekstraksi metadata dan Airflow; dan satu kasus uji pengambilan anotasi untuk menguji integrasi komponen propagasi anotasi, *adapter*, dan Marquez. Semua kasus uji pada pengujian integrasi memiliki status valid. Kemudian pengujian sistem dilakukan untuk menguji keseluruhan fungsionalitas sistem. Kasus uji pertama menguji sistem ketika terjadi perubahan jalur ETL sedangkan kasus uji kedua menguji sistem ketika terjadi perubahan anotasi. Kedua kasus uji pada pengujian sistem ini juga memiliki status valid. Dapat disimpulkan bahwa sistem telah berfungsi dengan baik karena seluruh pengujian *blackbox* yang dilakukan telah berstatus valid.

Pengujian *user acceptance testing* (UAT) dilakukan bersama dengan pengguna akhir, dimana pengguna akan melakukan beberapa skenario pengujian sistem kemudian mengisi kuesioner yang menyatakan kesetujuan terhadap pernyataan-pernyataan terkait sistem propagasi anotasi. Hasil kuesioner ditampilkan pada Tabel 1. Setiap pernyataan dapat dijawab dengan 4 opsi dalam skala *likert* yaitu sangat tidak setuju, tidak setuju, setuju, dan sangat setuju. Untuk menganalisis jawaban responden dihitunglah rata-rata terbobot dengan bobot 1 yaitu sangat tidak setuju hingga bobot 4 yaitu sangat setuju. Pada setiap pernyataan kuesioner, rata-rata terbobot yang diperoleh berada diatas 3,25 semuanya. Artinya, pengguna akhir merasa sangat setuju dengan pernyataan terkait performa dan fungsionalitas sistem.

7. KESIMPULAN

Berdasarkan hasil dari penelitian yang telah dilaksanakan, dapat diambil kesimpulan terhadap pengembangan sistem propagasi anotasi sebagai berikut:

1. Pada penelitian ini, sistem propagasi anotasi membutuhkan dua sistem pendukung yaitu Airflow untuk menjalankan ETL dan Marquez untuk mengelola *lineage*. Terdapat tiga komponen sistem yaitu komponen ekstraksi metadata, *adapter*, dan propagasi anotasi. Metadata diekstraksi dari Airflow secara periodik mengikuti jadwal ETL data. Kemudian metadata tersebut dikirim ke Marquez melalui *adapter* dan diambil oleh komponen propagasi anotasi untuk dapat mempropagasikan anotasi dengan tepat. Untuk mendukung interaksi pengguna dengan sistem, dirancang pula antarmuka sistem.
2. Sistem propagasi anotasi diimplementasikan dengan menggunakan bahasa pemrograman

Python kedalam tiga *library* yang terdiri dari komponen ekstraksi metadata, *adapter*, dan propagasi anotasi. Komponen ekstraksi metadata memanfaatkan sifat *polymorphism* pada pemrograman berbasis objek untuk dapat mengekstraksi metadata dengan dinamis. Komponen *adapter* menggunakan standar API OpenLineage. Kemudian komponen propagasi anotasi melakukan propagasi berdasarkan *lineage* dengan terlebih dahulu melacak genesis dari *dataset*.

3. Pengujian dilakukan dengan teknik *blackbox* dan *user acceptance testing* (UAT). Pengujian *blackbox* dilakukan dalam tiga tahap yaitu pengujian unit, integrasi, dan sistem. Pengujian unit menguji *method* yang berkaitan dengan pengelolaan metadata dan SQL *parsing*. Pada pengujian integrasi, diuji integrasi antara komponen ekstraksi metadata dengan Airflow dan komponen propagasi anotasi, *adapter*, dan Marquez. Pada pengujian sistem dilakukan pengujian perubahan ETL dan penambahan anotasi. Keseluruhan pengujian *blackbox* menghasilkan 30 kasus uji yang semuanya berstatus valid. Hasil analisis pengujian UAT menunjukkan bahwa rata-rata pengguna sangat setuju dengan performa dan fungsionalitas yang dimiliki oleh sistem. Oleh karena itu dapat disimpulkan bahwa sistem yang dikembangkan sudah dapat memenuhi kebutuhan pengguna.

Pada penelitian ini tata kelola anotasi pada sumber data harus dilakukan secara manual oleh *Database Administrator* (DBA). Proses bisnis ini mungkin justru malah menambah beban pekerjaan DBA. Implikasi penerapan sistem ini belum diketahui lebih lanjut. Penelitian selanjutnya disarankan agar dapat mengatasi masalah ini.

DAFTAR PUSTAKA

BHAGWAT, D., CHITICARIU, L., TAN, W.C., & VIJAYVARGIYA, G., 2005. An annotation management system for relational databases. VLDB Journal, 14(4), pp.373-396.

BUNEMAN, P., KHANNA, S., & TAN, W.C., 2000. Data provenance: Some basic issues. Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 1974(December), pp.87-93.

CHITICARIU, L., TAN, W.C., & VIJAYVARGIYA, G., 2005. DBNotes: A post-it system for relational databases based on provenance. Proceedings of the ACM SIGMOD International Conference on Management of Data, pp.942-944.

GEERTS, F., KEMENTSIETSIDIS, A., & MILANO, D., 2006. MONDRIAN: Annotating and querying databases through colors and blocks. Proceedings - International Conference on Data Engineering, 2006, p.82.

HG INSIGHTS, 2022. Apache Airflow. HG Insights, [online] Tersedia di: <https://discovery.hgdata.com/product/apache-airflow> [Diakses 29 Juni 2022]

IKEDA, R., WIDOM, J., 2009. Data lineage: A survey. Technical Report. Stanford InfoLab.

LU, Y., LI, Y., & ELTABAKH, M.Y., 2016. Decorating the cloud: enabling annotation management in MapReduce. VLDB Journal, 25(3), pp.399-424.

SIMMHAN, Y., PLALE, B., & GANNON, D., 2005. A survey of data provenance techniques. Computer Science Department, Indiana University, 47405, p.69.