Jurnal Riset Sistem Informasi Dan Teknik Informatika (JURASIK)
Volume 9, Nomor 2, Agustus 2024, pp 1042-1049
Terakreditasi Nomor 204/E/KPT/2022 | ISSN: 2527-5771/EISSN: 2549-7839
https://tunasbangsa.ac.id/ejurnal/index.php/jurasik

Penyimpanan Audit Log Dengan Menggunakan Apache Kafka

Achmad Ardiansyah[1]
Program Studi Teknik Informatika, Universitas Budi Luhur, Indonesia
Email: ahd.ardiansyah@gmail.com[1]

Abstract

Recently, there are many computer programs that being use for many purposes, so computer program will be entering complex programing era. If a system or a program become complex, it will need a new method to audit logs from the systems for an assurance if something is happening. For Example Error or Output that should not be happening but found after it release. This experiment will be using Apache kafka as a tool to make sure the system is making an audit logs without interrupting the main process. After some experiment using some of test case scenario, the result is audit logs using kafka is simple to use and reliable, also this method will not interrupt the main process of the program.

Keywords*: Apache Kafka, Audit Log, MongoDB*

Abstrak

Pada masa ini sangat banyak program komputer yang digunakan untuk berbagai kebutuhan, sehingga program komputer memasuki masa yang sangat kompleks. Jika sebuah sistem atau program mulai kompleks pastinya membutuhkan sebuah metode untuk melakukan Audit Log dari proses sistem sebagai jaminan apabila terjadi sesuatu, misalnya error atau output yang tidak semestinya muncul saat release. Penelitian ini akan menggunakan Apache Kafka sebagai alat yang dapat memastikan untuk melakukan Audit Log tanpa mengganggu proses utama. Setelah pengujian dilakukan hasil yang ditemukan bahwa Audit Log dengan menggunakan Apache Kafka mudah digunakan dan dapat diandalkan, serta tidak mengganggu proses program utama.

Kata kunci*: Apache kafka, Audit Logs, MongoDB*

1. PENDAHULUAN

Kemajuan dalam pemrograman sudah memasuki sebuah tahap baru yang sangat kompleks. Semakin besar atau semakin kompleks sebuah program maka akan memungkinan terjadinya *error*. Agar *programmer* dapat memeriksa program dengan membaca sejarah log. Sejarah log atau Audit log semakin baik jika sejarahnya semakin lama dan bisa disimpan ke dalam sebuah database tertentu. Log adalah rekaman yang menyimpan informasi tentang aktivitas sistem komputer tersebut.

Data Log yang disimpan adalah data yang berupa String. String adalah gabungan dari beberapa huruf atau merupakan gabungan dari beberapa kata. String sudah ada dari jaman Axel Thue's *foundational paper* di tahun 1906 [1] . String ini akan di proses menyesuaikan dengan format JSON (JavaScript Object Notation) agar data dapat disimpan di dalam MongoDB.

JSON menggunakan pasangan *key-value* dan memudahkan *developer* dalam membaca data tersebut. Format JSON ini juga digunakan sebagai *open standard file*

formats yang digunakan sebagai *human-readable text* untuk menyimpan dan mengirimkan data [2].

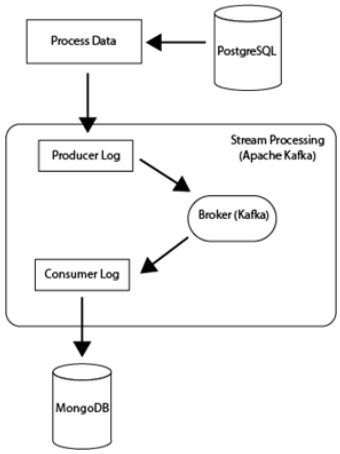
Penelitian ini akan membahas efektivitas penyimpanan log dengan menggunakan *Stream Processing* yang dapat atau tidaknya mengganggu proses program yang sedang berlangsung. Dengan metode ini juga diharapkan cocok untuk digunakan oleh perusahaan yang memiliki sistem program yang kompleks dan besar sehingga bisa memudahkan *Error Tracing* bagi *developer*.

2. METODOLOGI PENELITIAN

Metode penelitian ini akan dilakukan dengan beberapa langkah yang dapat dijelaskan dengan menggunakan Gambar 2.

Gambar 2. Metode Penelitian

Tahap persiapan merupakan tahap pengumpulan data dan dokumen agar dapat memahami proses penelitian yang akan dilakukan, serta memahami pokok pembahasan agar dapat merancang skema yang akan digunakan untuk penelitian. Tahap selanjutnya berupa tahapan perancangan Sistem. Bagian ini merupakan mendeskripsikan detail rancangan sistem yang akan dibuat. Penelitian ini akan menggunakan rancangan yang dijelaskan dengan Gambar 3.



Gambar 3. Rancangan Sistem

Rancangan pada gambar 3 merupakan alur yang akan dilakukan oleh suatu program. Dalam rancangan ini, akan terjadi suatu proses data. Proses data ini sangat penting sehingga dibutuhkannya penyimpanan data log. Pada penelitian ini akan menggunakan Metode *Stream Processing* dengan broker berupa Kafka. Setelah melakukan proses data maka program akan membentuk sebuah Json yang akan dikirim melalui *Producer* Log ke Broker Kafka sebagai penyimpanan

sementara, dan kemudian akan di *consume* oleh *Consumer Log* di program lain dan disimpan ke dalam MongoDb. Json dipilih karena memiliki kelebihan yaitu ukuran data yang sangat kecil yang dimana dengan susunan data umum mencapai 154 bytes, yang dimana besar datanya sendiri hanya 55 bytes [3]. Sehingga meminimalisir ukuran data besar saat disimpan oleh broker (Kafka). MongoDB dipilih dikarenakan Audit Log akan membutuhkan performa tinggi dan dapat bekerja efektif pada volume data yang besar, hal ini dapat diraih oleh *database* NoSQL (Not Only SQL (Structured Query Language)) [4]. Kafka sebagai broker dapat menyimpan data yang sangat besar berdasarkan lama waktu data yang ingin disimpan. Kafka juga dapat mendukung kemampuan banyak *consumer*, sehingga jika terdapat lebih dari 1 *consumer* maka data yang diproses akan lebih banyak dan lebih cepat [5].

PostgreSQL dipilih sebagai tempat penyimpanan data transaksi sebagai bukti proses program. Selain itu, PostgreSQL merupakan *open source object – relational database system* (ORDBMS). Bahasa yang digunakan untuk berinteraksi menggunakan *standard SQL* [6]. Structured Query Language (SQL) ini di publish oleh ANSI dan International Organization for Standardization (ISO) sebagai standar Bahasa pemrograman deklaratif untuk bahasa *database* [7].

Setelah semua kebutuhan tersebut dibuat menjadi sistem, kemudian sistem ini akan diuji keberhasilannya dalam menyimpan Log yang bersamaan dengan kegiatan proses program. Diharapkan dengan menggunakan *Stream Processing* ini tidak akan mengganggu proses yang terjadi di dalam program. Detail sejarah proses program juga dapat dilihat melalui MongoDB.

3. HASIL DAN PEMBAHASAN

Penelitian dimulai dengan membuat *database lokal* dengan menggunakan *database* PostgreSQL yang dimana sebagai tempat penyimpanan data dari sebuah proses. Pada proses ini yang akan diteliti, saat kondisi *error*. Apakah data tetap disimpan atau tidak dan apakah proses pengiriman melalui Kafka dapat mengganggu proses utama program. Jika dalam kondisi *error* data dapat disimpan dan juga tidak mengganggu proses utama maka metode ini sangat cocok diaplikasikan dalam penyimpanan log audit dengan menggunakan Apache kafka.

Pada Gambar 4 merupakan daftar tabel-tabel pada PostgreSQL yang digunakan dalam penelitian untuk memastikan data berhasil atau tidaknya tersimpan. Sedangkan pada MongoDB hanya menambahkan *Collection* berupa cl_log_create_order.

Kemudian, program dibuat menggunakan bahasa pemrograman java dengan framework spring-boot. Terdiri dari 2 *Project*, *project* pertama sebagai pembuat atau penerima *request* data *create order*, yang setiap proses berhasil atau *error* akan mengirimkan *producer* dan *project* yang kedua sebagai *consumer* yang akan menerima data proses berhasil atau *error* dan disimpan ke dalam MongoDB.

Jurnal Riset Sistem Informasi Dan Teknik Informatika (JURASIK)
Volume 9, Nomor 2, Agustus 2024, pp 1042-1049
Terakreditasi Nomor 204/E/KPT/2022 | ISSN: 2527-5771/EISSN: 2549-7839
https://tunasbangsa.ac.id/ejurnal/index.php/jurasik

```
CREATE TABLE tb_detail_order (
  tbdo_id SERIAL PRIMARY KEY,
  tbdo_order_number VARCHAR NOT null,
  tbdo_recipient_name VARCHAR NOT null,
  tbdo_total_trx int not null,
  tbdo_create_date timestamp,
  tbdo_update_date timestamp
);

CREATE TABLE tb_detail_order_product (
  tbdop_id SERIAL PRIMARY KEY,
  tbdop_tbp_sku VARCHAR NOT null references tb_product (tbp_sku),
  tbdop_tbdo_id int not null,
  tbdop_amount int NOT null,
  tbdop_harga_akhir int not null,
  tbdop_discount int not null,
  tbdop_create_date timestamp,
  tbdop_update_date timestamp
);

CREATE TABLE tb_product (
  tbp_id SERIAL PRIMARY KEY,
  tbp_sku VARCHAR NOT null,
  tbp_harga int not null,
  tbp_supplier_name Varchar not null,
  tbp_item_name Varchar not null
);
```

Gambar 4. Query Tabel Yang digunakan

```
{
 "orderDate": 123212312,
 "orderDetail": [
  {
   "amount": 5,
   "sku": "A-1235"
  },
  {
   "amount": 2,
   "sku": "A-1236"
  },
  {
   "amount": 4,
   "sku": "A-1237"
  }
 ],
 "orderNumber": "ABV127",
 "recipientName": "Daniel C"
}
```

Gambar 5. Data Json Request Body Project Pertama (Create Order)

Pada Gambar 5 merupakan kode Json yang digunakan sebagai *request body* untuk diproses dan disimpan ke dalam PostgreSQL.

```
{
 "createDate": "Date",
 "duration": 0,
 "endPoint": "string",
 "orderNumber": "string",
 "process": "string",
 "requestBody": "string",
 "requestId": "string",
 "requestTime": "string",
 "responseBody": "string",
 "statusCode": 0
}
```

Gambar 6. Kode Json *Request Body Project* Kedua (Consumer Log)

Jurnal Riset Sistem Informasi Dan Teknik Informatika (JURASIK)
Volume 9, Nomor 2, Agustus 2024, pp 1042-1049
Terakreditasi Nomor 204/E/KPT/2022 | ISSN: 2527-5771/EISSN: 2549-7839
https://tunasbangsa.ac.id/ejurnal/index.php/jurasik

Project kedua akan menerima data-data pada Json (Gambar 6) dan diolah agar dapat dimasukkan ke dalam MongoDB, Data Json inilah yang diterima melalui Kafka.

Setelah diuji beberapa kali dengan beberapa kondisi, serta setelah memasukkan dan mengambil beberapa data dari *database* PostgreSQL, tampil data sebagai berikut.

tb_detail_order | Enter a SQL expression to filter results (use Ctrl+Space)

	tbdo_id	tbdo_order_number	tbdo_recipient_name	tbdo_total_harga_final	tbdo_create_date	tbdo_update_date
1	1	ABV124	Daniel	140,000	[NULL]	[NULL]
2	2	ABV125	Daniel	170,000	[NULL]	[NULL]
3	3	ABV126	Daniel C	140,000	2024-06-25 00:03:24.350	2024-06-25 00:03:24.350
4	4	ABV127	Daniel C	126,000	2024-06-25 00:10:19.240	2024-06-25 00:10:19.240
5	5	ABV128	Daniel C	112,000	2024-06-25 00:12:36.319	2024-06-25 00:12:36.319

Gambar 7. Tampilan tb_detail_order

Tb_detail_order merupakan tabel utama yang menyimpan total transaksi / total belanja, nomor order dan nama pemesan.

tb_detail_order_product | Enter a SQL expression to filter results (use Ctrl+Space)

	tbdop_id	tbdop_tbp_sku	tbdop_amount	tbdop_subtotal_harga	tbdop_discount	tbdop_tbdo_id	tbdop_create_date	tbdop_update_date
1	1	A-1234	12	60,000	0	1	[NULL]	[NULL]
2	2	A-1237	10	80,000	0	1	[NULL]	[NULL]
3	3	A-1234	10	50,000	0	2	[NULL]	[NULL]
4	4	A-1237	15	120,000	0	2	[NULL]	[NULL]
5	5	A-1234	12	60,000	0	3	2024-06-25 00:03:24.383	2024-06-25 00:03:24.383
6	6	A-1237	10	80,000	0	3	2024-06-25 00:03:24.397	2024-06-25 00:03:24.397
7	7	A-1235	5	50,000	0	4	2024-06-25 00:10:19.251	2024-06-25 00:10:19.251
8	8	A-1237	2	16,000	0	4	2024-06-25 00:10:19.252	2024-06-25 00:10:19.252
9	9	A-1236	4	60,000	0	4	2024-06-25 00:10:19.253	2024-06-25 00:10:19.253

Gambar 8. Tampilan tb_detail_order_product

Tb_detail_order_product merupakan tabel yang merincikan detail barang yang dibelanjakan beserta sub total harganya dari jumlah barang yang dibeli dikalikan jumlah harganya.

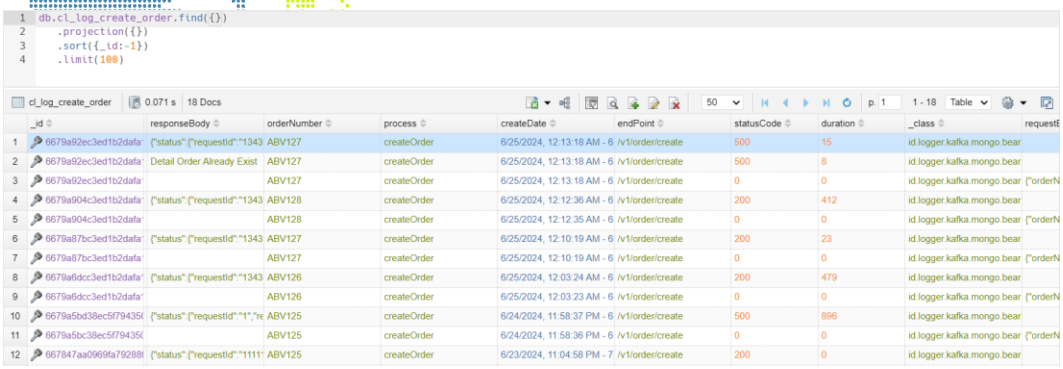
tb_product | Enter a SQL expression to filter results (use Ctrl+Space)

	tbp_id	tbp_sku	tbp_harga	tbp_supplier_name	tbp_item_name
1	1	A-1234	5,000	sulaiman	Apel
2	3	A-1235	10,000	sulaiman	Nanas
3	4	A-1236	15,000	sujandi	Pisang
4	5	A-1237	8,000	suep	Roti Coklat

Gambar 9. Tampilan tb_product

Tb_product merupakan referensi data berdasarkan SKU, untuk mengetahui harga, *supplier* dan nama itemnya. Dari Tabel ini yang akan menjadi referensi ketersediaan nama produk dan harga yang nanti digunakan untuk mengisi Tb_detail_order_product.

Jurnal Riset Sistem Informasi Dan Teknik Informatika (JURASIK)
Volume 9, Nomor 2, Agustus 2024, pp 1042-1049
Terakreditasi Nomor 204/E/KPT/2022 | ISSN: 2527-5771/EISSN: 2549-7839
https://tunasbangsa.ac.id/ejurnal/index.php/jurasik



Gambar 10. Tampilan data mongoDB dari *Collection* cl_log_create_order

Pada Gambar 10 merupakan *Collection* MongoDB yang berisikan Log hasil dari menjalankan Program. Setelah Program dibuat dan dapat berjalan sesuai rancangan maka selanjutnya adalah memasuki masa pengujian dengan beberapa *test scenario*. *Test Scenario* ini berfungsi untuk memastikan target dari penelitian dapat tercapai.

Tabel 1. Hasil Pengujian Sistem

Test Scenario	Harapan Program	Tahap Yang Dilakukan	Hasil	Keberhasilan / Status
Menyimpan Log ke Mongo Melalui kafka Tanpa Mengganggu proses *Insert* ke DB	Dapat Menyimpan Log ke Mongo tanpa hambatan	Kirim data *request create order* dan lihat data di PostgreSQL dan MongoDB	Dalam kondisi normal data berhasil masuk ke postgresql dan log disimpan ke mongoDB	Berhasil
Mengirim *Request* ke *Project Create Order* dengan *Project Consumer Log* tidak aktif	Data akan berhasil tersimpan ke dalam DB PostgreSQL dan data log tersimpan sementara di dalam Kafka Siap di *Consume*	Mematikan *Project Consumer Log*, Kemudian mengirim *Request Create Order* ke *Project Create Order*	Data Berhasil *Insert* ke DB PostgreSQL tanpa terganggu oleh pengiriman data melalui Kafka walau *Project Consumer* sedang tidak aktif	Berhasil
Request Log di *Consume* ketika di nyalakan kembali	Log yang masih tersimpan di Broker Kafka dapat di *Consume* kembali oleh *Project Log Consumer*	Mengaktifkan Kembali *Project Consumer Log*	Data dari Broker Berhasil di *Consume* dan disimpan ke dalam MongoDB, sehingga memastikan data Log tidak Hilang	Berhasil
Proses yang dilakukan tidak memakan waktu yang	Ketika *Request Create Order* dilakukan, proses data	Menjalankan *Request Create Order* dan melihat data	Berdasarkan *orderNumber* “ABV127” pada Gambar 10 hanya	Berhasil

Test Scenario	Harapan Program	Tahap Yang Dilakukan	Hasil	Keberhasilan / Status
lama ketika *insert* yang disebabkan oleh kirim data ke Kafka	hanya memakan waktu beberapa *milisecond*	*Duration* di MongoDB	memakan waktu 23 *millisecond* sehingga membuktikan proses logger tidak mempengaruhi proses utama	
Kondisi *Error* juga mengirimkan Log dan Log tersebut disimpan kedalam MongoDB	Ketika mengirimkan order Number yang sama dengan sebelumnya dan seharusnya tercatat alasan *error*, *request Body* dan *Response Body*	Mengirimkan *order Number* yang sama dengan sebelumnya yaitu "ABV127"	Data berhasil disimpan di dalam mongoDB walaupun kondisi program *create Order Error*, serta menampilkan Alasannya bisa dilihat pada *order number* "ABV127" pada gambar 10, menghasilkan *Error* 500 dengan alasan "*Detail Order Already Exist*"	Berhasil

4. SIMPULAN

Berdasarkan data percobaan dengan beberapa skenario tersebut menghasilkan beberapa kesimpulan. Seperti, audit log yang dikirimkan melalui Apache kafka secara *Realtime,* sehingga memudahkan analisa waktu kejadian pada MongoDB. Selain itu, Proses pembentukan dan pengiriman Audit Log tidak menghentikan atau mengganggu proses utama program yaitu *insert* atau *update* data di *database*, serta mencatat secara teliti jika terjadi *Error* pada program. Disisi lain jika *consumer* mengalami gangguan atau keadaan tidak aktif, broker (Apache kafka) akan menjadi penyimpanan sementara data yang dikirim oleh *producer* hingga *Consumer* sudah dapat melakukan kegiatan *consume* kembali. Hal ini meminimalisir data yang hilang pada saat proses berlangsung. Apache Kafka sangat *reliable* selama penelitian berlangsung.

DAFTAR PUSTAKA

[1] N. Mhaskar and W. F. Smyth, "String Covering: A Survey," *Fundam Inform*, vol. 190, no. 1, pp. 17–45, Oct. 2023, doi: 10.3233/FI-222164.
[2] M. Sharma, *MongoDB Complete Guide*, 1st ed. BPB Publications, 2021.
[3] G. P. Tiwary, E. Stroulia, and A. Srivastava, "Compression of XML and JSON API Responses," *IEEE Access*, vol. 9, pp. 57426–57439, 2021, doi: 10.1109/ACCESS.2021.3073041.

[4] B. Jose and S. Abraham, "Performance analysis of NoSQL and relational databases with MongoDB and MySQL," *Mater Today Proc*, vol. 24, pp. 2036–2043, 2020, doi: https://doi.org/10.1016/j.matpr.2020.03.634.
[5] N. Momtselidze and A. Tsitsagi, "Apache Kafka-Real-time Data Processing," 2015.
[6] A. Makris, K. Tserpes, G. Spiliopoulos, D. Zissis, and D. Anagnostopoulos, "MongoDB Vs PostgreSQL: A comparative study on performance aspects," *Geoinformatica*, vol. 25, no. 2, pp. 243–268, 2021, doi: 10.1007/s10707-020-00407-w.
[7] S. Juba and A. Volkov, *Learning PostgreSQL 11*, 3rd ed. Birmingham: Packt Publishing Ltd., 2019.