OPTIMASI DATA LOG AUDIT MASTER & AUDIT PAYLOAD MENGGUNAKAN METODE KRAFT

Rezza Anugrah Mutiarawan[1]*, **Ade Davy Wiranata**[2], **Iswahyudi**[3]
Universitas Utpadaka Swastika[1], Universitas Muhammadiyah Prof. Dr. Hamka[2],
Universitas Islam Negeri Kiai Ageng Muhammad Besari Ponorogo[3]
rezza.mutiarawan@gmail.com[1], adedavy@uhamka.ac.id[2],
iswahyudi.briliant@gmail.com[3]

ABSTRAK

Saat ini, banyak aplikasi yang digunakan untuk berbagai keperluan memerlukan tingkat integritas data yang tinggi, yang sering kali menyebabkan kompleksitas program semakin meningkat. Seiring bertambahnya kompleksitas sistem, mekanisme audit log yang efektif menjadi sangat penting untuk menjaga integritas data. Salah satu solusinya adalah dengan merekam aktivitas pengguna dalam sistem, sehingga semua tindakan dapat dimonitor secara menyeluruh. Dalam penelitian ini, KRaft digunakan sebagai alat untuk mengelola proses audit log. Implementasi KRaft bertujuan untuk memastikan bahwa proses audit tidak mengganggu operasi utama dari program. Berdasarkan skenario pengujian yang dilakukan, hasilnya menunjukkan bahwa penerapan KRaft untuk audit master data dan audit payload data dapat dilakukan dengan mudah dan tidak mengganggu proses utama aplikasi.

Kata Kunci: Audit Log, Kafka KRaft, Kafka Topic, MySQL

ABSTRACT

Currently, many applications serving various purposes require a high level of data integrity, which often results in increased program complexity. As systems become more complex, an effective audit logging mechanism is essential to maintain data integrity. One solution is to record user activities within the system, enabling comprehensive monitoring of all actions. In this study, KRaft is utilized as a tool to manage the audit logging process. The implementation of KRaft is intended to ensure that the auditing procedure does not interfere with the program's core operations. Based on the testing scenarios conducted, the results demonstrate that implementing KRaft for both audit master data and audit payload data is straightforward and does not disrupt the main processes of the application.

***Keywords**: Audit Log, Kafka KRaft, Kafka Topic, MySQL*

1. PENDAHULUAN

Perkembangan dalam dunia pemrograman semakin pesat, dengan tingkat kompleksitas yang terus meningkat. Seiring dengan bertambahnya ukuran dan kompleksitas suatu program, risiko terjadinya error juga semakin besar. Untuk mendeteksi serta menganalisis kesalahan yang muncul, programmer perlu merujuk pada log atau audit log. Audit log yang memiliki riwayat panjang dan tersimpan

dalam database akan sangat membantu dalam proses pemantauan serta analisis sistem.

Log merupakan rekaman aktivitas dalam sistem komputer yang menyimpan berbagai informasi terkait operasional program (Lu et al., 2025). Data yang dicatat dalam log umumnya berupa tipe data string, yaitu kumpulan karakter yang dapat terdiri dari huruf, angka, atau simbol tertentu. Konsep *string* sendiri telah dikenal sejak penelitian Alex Thue pada tahun 1906. Agar lebih mudah dikelola dan diproses, data log dalam bentuk string dapat dikonversi ke dalam format JSON (*JavaScript Object Notation*) sebelum disimpan dalam database (Munro, 2024).

JSON memiliki struktur berbasis pasangan *key-value*, yang membuatnya lebih mudah dibaca dan dipahami oleh developer (Wei et al., 2025). Format ini juga telah menjadi standar terbuka yang banyak digunakan dalam penyimpanan dan pertukaran data dalam bentuk teks yang dapat diinterpretasikan oleh manusia.

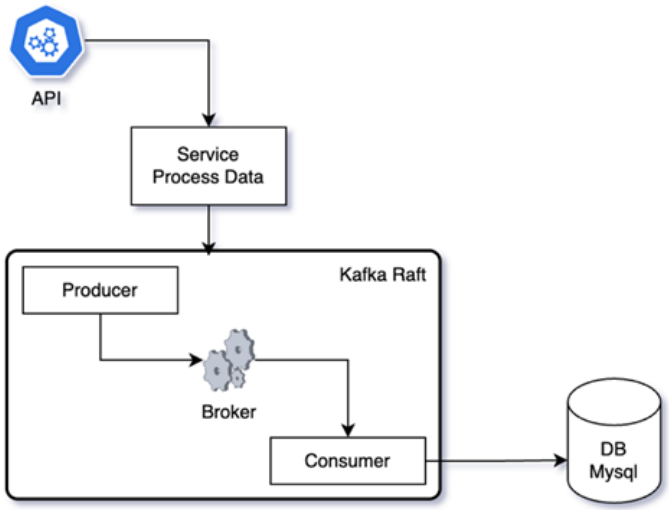
Penelitian ini berfokus pada efektivitas penyimpanan log dengan pendekatan *stream processing*, yang bertujuan untuk mengevaluasi sejauh mana metode ini dapat mengurangi gangguan terhadap proses utama sistem (*main process*). Dengan pendekatan ini, diharapkan solusi yang dihasilkan dapat diimplementasikan secara efektif pada perusahaan dengan sistem berskala besar dan kompleks, sehingga mempermudah developer dalam melakukan *error tracing* secara efisien.

2. METODE

Metode penelitian ini digambarkan dengan beberapa langkah seperti pada gambar di bawah ini Gambar 1.

Gambar 1. Metode Penelitian

Tahap persiapan adalah tahap untuk mengumpulkan data dan object agar bisa memahami proses penelitian yang akan dilakukan, serta memahami pokok bahasan agar dapat merancang skema yang akan digunakan dalam penelitian ini (Coe et al., 2025). Tahap selanjutnya yaitu tahap perancangan sistem. Pada tahap perancangan sistem menjelaskan detail rancangan dari sebuah sistem yang akan dibuat dalam proses penelitian. Penelitian ini akan menggunakan rancangan yang dijelaskan pada gambar 2.



Gambar 2. *Flow Data*

Rancangan sistem pada gambar 2 merupakan *flow* data yang akan dilakukan oleh suatu program. Pada rancangan ini proses *flow* data sangat penting, sehingga dibutuhkannya penyimpanan data log. Pada penelitian ini akan menggunakan metode *stream processing* dengan KRaft (*Kafka Raft*) (Chatterjee et al., 2025). Pada saat melakukan proses data, saat itu data dikirim ke Kafka melalui *Producer* dan meneruskan ke broker untuk proses penyimpanan data sementara, lalu data yang berada di broker di distribusikan ke *consumer* sebagai program yang memproses data dan menyimpan ke database MySQL.

MySQL merupakan salah satu database yang dipilih sebagai tempat penyimpanan data dari serangkaian aktivitas yang dilakukan dan sebagai bukti dari proses program (Nissan, 2025). MySQL adalah sistem manajemen basis data (RDBMS) yang bersifat *open-source* dan merupakan RDBMS yang ringan, cepat dan cocok untuk berbagai program aplikasi. SQL atau *Sructured Query Language* adalah bahasa yang digunakan untuk berinteraksi.

Setelah semua kebutuh diatas dibuat menjadi sistem, maka program dilakukan pengujian dalam proses penyimpanan pada saat terjadinya aktivitas proses program. Dengan menggunakan *Streaming Processing* diharapkan tidak menggangu main proses atau proses yang sedang berjalan di dalam program. *History* data dari sebuah proses yang sudah terjadi dapat dilihat melalui database MySQL.

3. HASIL DAN PEMBAHASAN

Penelitian ini di awali dengan pembuatan database MySQL di *environment* lokal, yang berfungsi sebagai media penyimpanan data dari suatu proses. Fokus penelitian adalah menganalisis kondisi ketika terjadi kesalahan (*error*), untuk menilai apakah data tetap dapat tersimpan dan apakah proses pengiriman melalui kafka mempengaruhi jalannya proses utama dalam program. Apabila data tetap

tersimpan meskipun terjadi *error* dan tidak menggangu proses utama, maka pendekatan ini dinilai layak untuk diterapkan dalam sistem pencatatan *log* audit menggunakan KRaft. MySQL adalah sistem manajemen database relasional (RDBMS) *open-source* yang berbasis SQL dan bekerja dengan *model client-server* (Muppala, 2025).

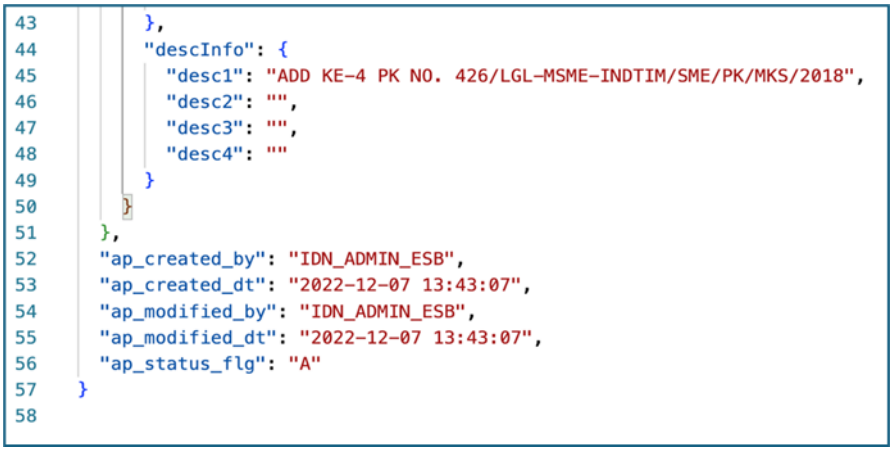
Gambar 4 menampilkan daftar tabel pada MySQL yang digunakan untuk memverifikasi keberhasilan atau kegagalan penyimpanan data. Program pada penelitian ini dikembangkan menggunakan bahasa pemrograman Java dengan *framework Spring Boot*. Terdapat dua aplikasi utama: aplikasi pertama berfungsi sebagai pengelola permintaan aktivitas transaksi (aplikasi-*publisher-log*), yang akan mengirimkan data ke *producer* setiap kali proses berjalan sukses atau gagal, sedangkan aplikasi kedua bertindak sebagai *consumer* atau *subscriber* (aplikasi-*consumer-log*), yang menerima data hasil proses dan menyimpan ke dalam database MySQL.

```
CREATE TABLE AUDIT_PAYLOAD_1 (
    AP_RQ_ID VARCHAR(100) NOT NULL,
    AP_PARENT_TRANS_SEQ_ID VARCHAR(100),
    AP_TRANS_SEQ_ID VARCHAR(100),
    AP_BUSINESS_SERVICE_ID VARCHAR(100),
    AP_AUDIT_TYPE VARCHAR(50),
    AP_PAYLOAD TEXT,
    AP_CREATED_BY VARCHAR(50),
    AP_CREATED_DT DATETIME DEFAULT CURRENT_TIMESTAMP,
    AP_MODIFIED_BY VARCHAR(50),
    AP_MODIFIED_DT DATETIME DEFAULT NULL ON UPDATE CURRENT_TIMESTAMP,
    AP_STATUS_FLG CHAR(1),
    INDEX idx_ap_trans_seq_id (AP_TRANS_SEQ_ID)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;



CREATE TABLE AUDIT_MASTER1 (
    AM_RQ_ID VARCHAR(100) NOT NULL,
    AM_PARENT_TRAN_SEQ_ID VARCHAR(100),
    AM_TRAN_SEQ_ID VARCHAR(100),
    AM_BUSINESS_SERVICE_ID VARCHAR(100),
    AM_AUDIT_TYPE VARCHAR(50),
    AM_IPADDRESS VARCHAR(50),
    AM_CONSUMER_ID VARCHAR(100),
    AM_SLA_EXCEED_TIME INT,
    AM_SLA_QOS_DURATION INT,
    AM_MESSAGE_IN_TIME DATETIME,
    AM_MESSAGE_OUT_TIME DATETIME,
    AM_CREATED_BY VARCHAR(50),
    AM_CREATED_DT DATETIME DEFAULT CURRENT_TIMESTAMP,
    AM_MODIFIED_BY VARCHAR(50),
    AM_MODIFIED_DT DATETIME DEFAULT NULL ON UPDATE CURRENT_TIMESTAMP,
    AM_STATUS_FLG CHAR(1),
    PRIMARY KEY (AM_RQ_ID),
    INDEX idx_am_tran_seq_id (AM_TRAN_SEQ_ID)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
```

Gambar 3. *Query Table*

```
{
  "ap_rq_id": "1330432674_SIBS_20221207-134308",
  "ap_parent_trans_seq_id": "7A184EB52E290YA6",
  "ap_trans_seq_id": "2212071343085600",
  "ap_business_service_id": "DocumentAdd1.0",
  "ap_audit_type": "DP_AUDIT_IN",
  "ap_payload": {
    "nonFinSvcHeaderInfo": {
      "bankId": 11,
      "branchId": "47007"
    },
    "docInfo": {
      "docId": "MA==",
      "docLevel": "LL",
      "collateralId": "MTgyODAxMQ==",
      "docCode": "PK ADD",
      "docNo": 426,
      "docType": "1",
      "noOfCopies": 0,
      "docDate": "2022-11-30",
      "docRegDate": "",
      "docFollowUpDate": "",
      "docketCode": "",
      "docStatus": "07",
      "docStatusDate": "2022-12-07",
      "frequency": 0,
      "freqCode": "",
      "lodgementDate": "",
      "startDate": "",
      "waivedBy": "",
      "nextDueDate": "",
      "custodianInfo": {
        "docCustodian": "01",
        "otherCustodian": "",
        "branchCustodian": 44000
      },
      "refClause": "",
      "invoiceNo": "",
      "remarkInfo": {
        "remark1": "",
        "remark2": "",
        "remark3": ""
      },
```

Gambar 4. *Data Json Request Body Audit-Payload 1*



```
      },
      "descInfo": {
        "desc1": "ADD KE-4 PK NO. 426/LGL-MSME-INDTIM/SME/PK/MKS/2018",
        "desc2": "",
        "desc3": "",
        "desc4": ""
      }
    }
  },
  "ap_created_by": "IDN_ADMIN_ESB",
  "ap_created_dt": "2022-12-07 13:43:07",
  "ap_modified_by": "IDN_ADMIN_ESB",
  "ap_modified_dt": "2022-12-07 13:43:07",
  "ap_status_flg": "A"
}
```

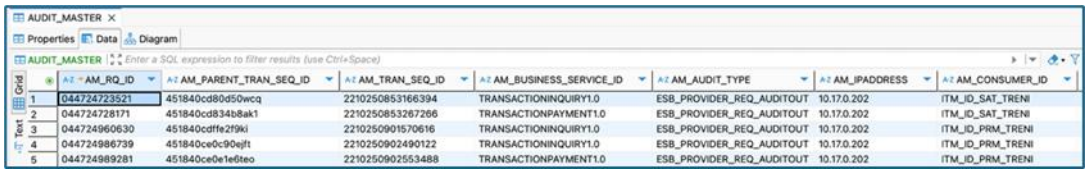
Gambar 5. *Data Json Request Body Audit-Payload 2*

```
{
  "ap_rq_id": "044724723521",
  "ap_parent_trans_seq_id": "451840cd80d50wcq",
  "ap_trans_seq_id": "2210250853166394",
  "ap_business_service_id": "TRANSACTIONINQUIRY1.0",
  "ap_audit_type": "ESB_PROVIDER_REQ_AUDITOUT",
  "ap_source_ip": "10.17.0.202",
  "ap_source_app_id": "ITM_ID_SAT_TRENI",
  "ap_payload": null,
  "ap_payload_rsp": null,
  "ap_request_dt": "2022-10-25 08:53:16",
  "ap_response_dt": "2022-10-25 08:53:16",
  "ap_created_by": "IDN_ADMIN_ESB",
  "ap_created_dt": "2022-10-25 08:53:16",
  "ap_modified_by": "IDN_ADMIN_ESB",
  "ap_modified_dt": "2022-10-25 08:53:16",
  "ap_status_flg": "A"
}
```


Gambar 6. *Data Json Request Body Audit-Master*

Pada aplikasi kedua, sistem bertugas menerima data dalam format JSON (Seperti ditunjukan pada Gambar 4,5 dan Gambar 6) yang dikirim melalui KRaft. Data tersebut kemudian diproses dan disiapkan untuk disimpan kedalam basis data MySQL. Dalam hal ini Query kemudian membantu mengambil data dari berbagai tabel, menyusunnya, lalu menampilkannya sesuai perintah (Wanniarachchi & Misra, 2025) (Wisnani, 2010).

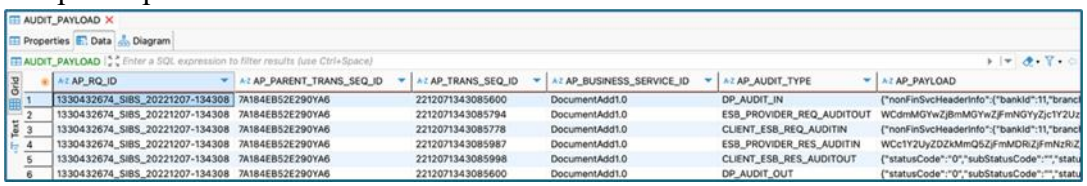
Setelah dilakukan pengujian revisi agar lebih ringkas dan jelas, diperoleh hasil data seperti yang ditampilkan berikut ini.



	AM_RQ_ID	AM_PARENT_TRAN_SEQ_ID	AM_TRAN_SEQ_ID	AM_BUSINESS_SERVICE_ID	AM_AUDIT_TYPE	AM_IPADDRESS	AM_CONSUMER_ID
1	044724723521	451840cd80d50wcq	2210250853166394	TRANSACTIONINQUIRY1.0	ESB_PROVIDER_REQ_AUDITOUT	10.17.0.202	ITM_ID_SAT_TRENI
2	044724728171	451840cd834b8ak1	2210250853267266	TRANSACTIONPAYMENT1.0	ESB_PROVIDER_REQ_AUDITOUT	10.17.0.202	ITM_ID_SAT_TRENI
3	044724960630	451840cdffe2f9ki	2210250901570616	TRANSACTIONINQUIRY1.0	ESB_PROVIDER_REQ_AUDITOUT	10.17.0.202	ITM_ID_PRM_TRENI
4	044724986739	451840ce0c90ejft	2210250902490122	TRANSACTIONINQUIRY1.0	ESB_PROVIDER_REQ_AUDITOUT	10.17.0.202	ITM_ID_PRM_TRENI
5	044724989281	451840ce0e1e6teo	2210250902553488	TRANSACTIONPAYMENT1.0	ESB_PROVIDER_REQ_AUDITOUT	10.17.0.202	ITM_ID_PRM_TRENI

Gambar 7. Tampilan Data *Audit Master*

Tabel Audit_Master merupakan tabel utama yang menyimpan tahap demi tahap data pada service dari suatu transaksi.



	AP_RQ_ID	AP_PARENT_TRANS_SEQ_ID	AP_TRANS_SEQ_ID	AP_BUSINESS_SERVICE_ID	AP_AUDIT_TYPE	AP_PAYLOAD
1	1330432674_SIBS_20221207-134308	7A184EB52E290YA6	2212071343085600	DocumentAdd1.0	DP_AUDIT_IN	{"nonFinSvcHeaderInfo":{"bankId":11,"branch
2	1330432674_SIBS_20221207-134308	7A184EB52E290YA6	2212071343085794	DocumentAdd1.0	ESB_PROVIDER_REQ_AUDITOUT	WCdmMGYwZjBmMGYwZjFmNGYyZjc1Y2Uz
3	1330432674_SIBS_20221207-134308	7A184EB52E290YA6	2212071343085778	DocumentAdd1.0	CLIENT_ESB_REQ_AUDITIN	{"nonFinSvcHeaderInfo":{"bankId":11,"branch
4	1330432674_SIBS_20221207-134308	7A184EB52E290YA6	2212071343085987	DocumentAdd1.0	ESB_PROVIDER_RES_AUDITIN	WCc1Y2UyZDZkMmQ5ZjFmMDRiZjFmNzRiZ
5	1330432674_SIBS_20221207-134308	7A184EB52E290YA6	2212071343085998	DocumentAdd1.0	CLIENT_ESB_RES_AUDITOUT	{"statusCode":"0","subStatusCode":"","statu
6	1330432674_SIBS_20221207-134308	7A184EB52E290YA6	2212071343085600	DocumentAdd1.0	DP_AUDIT_OUT	{"statusCode":"0","subStatusCode":"","statu

Gambar 8. Tampilan Data *Audit-Payload*

Tabel *Audit_Payload* merupakan tabel yang merincikan tahap demi tahap detail payload pada service dari suatu transaksi (Zagi, 2025).

Tabel 1. Tabel *Audit_Payload*

Skenario Test	**Target Program**	**Tahapan Pelaksanaan**	**Result**	**Status**
Menyimpan *log* ke MySQL melalui KRaft tanpa menggangu proses *insert* ke DB	Penyimpanan log ke MySQL berjalan tanpa kendala.	Lakukan pengiriman data *create* transaksi, kemudian periksa hasil penyimpanan data tersebut di MySQL	Ketika sistem berjalan normal, data ke MySQL dan pencatatan *log* dilakukan di MySQL	Success
Mengirim Request ke aplikasi *publisher* dengan aplikasi-*consumer-log* tidak aktif.	Penyimpanan data ke MySQL berhasil dilakukan, dan log aktivitas di tampung sementara.	Tahapan diawali dengan menghentikan layanan aplikasi-*consumer-log*, kemudian dilanjutkan dengan pengiriman *request create* transaksi ke sistem aplikasi-*publisher-log*.	Proses insert data ke MySQL tidak terpengaruh oleh pengiriman data melalui KRaft, bahkan dalam kondisi aplikasi-consumer-log belum aktif.	Success
Request log di *consumer* ketika dinyalakan kembali.	Log yang belum di proses dan masih berada di kafka dapat di proses ulang oleh aplikasi-*consumer-log*	Menghidupkan kembali layanan *aplikasi-consumer-log* untuk melanjutkan proses konsumsi data	Proses konsumsi data dari *broker* berjalan sukses dan data log berhasil di simpan di MySQL untuk menghindari kehilangan informasi	Success
Pengiriman data ke kafka tidak menyebabakan proses insert memakan waktu yang lama.	Saat *request* transaksi dijalankan, pemrosesan data berlangung sangat cepat hanya dalam beberapa *milisecond*.	Mengirim permintaan *create* transaksi kemudian mengamati transaksi durasi yang tersimpan di MySQL	Dengan waktu eksekusi hanya 23 *milisecond* untuk request number ’ 044724723521’ dapat di simpulkan bahwa prose loging tidak memberikan dampak negatif terhadap proses utama.	Success

Kondisi *error* juga mengirimkan *log* dan *log* tersebut disimpan kedalam MySQL	Jika *request number* yang sama dikirimkan kembali, maka sistem seharunya mencatat alasan duplicate data log, already exist, beserta *request body* dan *response body*.	Melakukan pengiriman kembali data transaksi dengan nomor yang identik dengan pengujian sebelumnya, yakni ' 044724723521'	Pada *request number* '044724723521', meskipun program memproses request number yang sama., data tetap tercatat di MySQL dan menampilkan responseCode (409) dan responseMessage (Data *already exist)*.	Success

Berdasarkan hasil percobaan yang dilakukan dengan berbagai skenario, diperoleh beberapa kesimpulan penting. Pertama, audit *log* yang dikirim melalui KRaft bersifat *real-time,* sehingga memungkinkan analisis waktu kejadian langsung pada MySQL. Selain itu proses pembuatan dan pengiriman audit *log* tidak menghambat atau mengganggu jalannya proses utama, seperti *insert* atau *update* pada databases, serta tetap mampu mencatat kesalahan program dengan akurat. Di sisi lain ketika *consumer* mengalami gangguan atau tidak aktif, *broker* KRaft berfungsi sebagai penyimpan sementara bagi data yang dikirim oleh *producer,* hingga *consumer* dapat kembali melakukan proses konsumsi data. Mekanisme ini mampu mengurangi risiko kehilangan data selama proses berlangsung. Secara keseluruhan, KRaft menunjukan performa yang handal sepanjang proses penelitian.

4. SIMPULAN

Berdasarkan hasil percobaan yang dilakukan dengan berbagai skenario, diperoleh beberapa kesimpulan penting. Pertama, audit *log* yang dikirim melalui KRaft bersifat *real-time,* sehingga memungkinkan analisis waktu kejadian langsung pada MySQL. Selain itu proses pembuatan dan pengiriman audit *log* tidak menghambat atau mengganggu jalannya proses utama, seperti *insert* atau *update* pada databases, serta tetap mampu mencatat kesalahan program dengan akurat. Di sisi lain ketika *consumer* mengalami gangguan atau tidak aktif, *broker* KRaft berfungsi sebagai penyimpan sementara bagi data yang dikirim oleh *producer,* hingga *consumer* dapat kembali melakukan proses konsumsi data. Mekanisme ini mampu mengurangi risiko kehilangan data selama proses berlangsung. Secara keseluruhan, KRaft menunjukan performa yang handal sepanjang proses penelitian.

DAFTAR PUSTAKA

Chatterjee, N., Yadav, A. R., & Talwandi, N. S. (2025). Temporal Resilience in Stream Processing: Mitigating Late Data and Lag in Apache Kafka 4.0. *IJSAT-International Journal on Science and Technology*, *16*(2).

Coe, R., Waring, M., Hedges, L. V, & Ashley, L. D. (2025). *Research methods and methodologies in education*. SAGE Publications Limited.

Lu, Y., Karanjai, R., Alsagheer, D., Kasichainula, K., Xu, L., Shi, W., & Huang, S.-H. S. (2025). LogBabylon: A Unified Framework for Cross-Log File Integration and Analysis. *Proceedings of the 40th ACM/SIGAPP Symposium on Applied Computing*, 1953–1960.

Munro, M. C. (2024). Using JSON. In *Learn FileMaker Pro 2024: The Comprehensive Guide to Building Custom Databases* (pp. 339–375). Springer.

Muppala, M. (2025). *SQL Database Mastery: Relational Architectures, Optimization Techniques, and Cloud-Based Applications*. Deep Science Publishing.

Nissan, M. I. (2025). MemTraceDB: Reconstructing MySQL User Activity Using ActiviTimeTrace Algorithm. *ArXiv Preprint ArXiv:2509.05891*.

Wanniarachchi, D., & Misra, A. (2025). Mimic: Ai and ar-enhanced multi-modal, immersive, relative instruction comprehension. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, *9*(1), 1–34.

Wei, W., Na, L., Lei, Z., Fang, L., Hao, C., Xiuying, Y., Lei, H., Min, Z., Gang, W., & Jie, Z. (2025). An Extensive Study on Text Serialization Formats and Methods. *ArXiv Preprint ArXiv:2505.13478*.

Wisnani, Y. (2010). Queri Ganda Pada Sistem Temu-Kembali Informasi Berbasis Jaringan Inferensi. *Makara Journal of Science*, *8*(2), 5.

Zagi, L. M. (2025). Implementasi dan Pengujian Polimorfisme pada Malware Menggunakan Dasar Payload Metasploit Framework. *ArXiv Preprint ArXiv:2508.00874*.