Pengembangan Model Migrasi *Database Relational* ke NoSQL Memanfaatkan *Metadata* SQL

Muhammad Nur Yasir Utomo
Program Studi Teknik Komputer dan Jaringan, Jurusan Teknik Elektro, Politeknik Negeri Ujung Pandang
Email: yasirutomo@poliupg.ac.id

Abstract

Penyimpanan data merupakan isu krusial pada teknologi Big Data karena membutuhkan teknologi penyimpanan data yang profisien agar dapat menyimpan data (terstruktur dan tidak terstruktur) secara cepat dalam jumlah besar. Hal ini sudah tidak bisa lagi dipenuhi oleh model database relational (SQL) yang saat ini masih banyak digunakan. Kelemahan tersebut dapat diatasi dengan menggunakan database NoSQL, namun sayangnya proses migrasi data dari relational/SQL database ke NoSQL masih sulit dilakukan karena perbendaan skema dan format penyimpanan data. Berdasarkan masalah tersebut, maka penelitian mengenai migrasi database relational ke NoSQL sangat diperlukan. Penelitian ini mencoba mengajukan pengembangan model perangkat lunak untuk migrasi database relational ke NoSQL menggunakan pendekatan aturan migrasi dan data transformasi yang memanfaatkan metadata SQL. Berdasarkan eksperimen yang telah dilakukan dengan menggunakan 1 (satu) database dengan 5 (lima) table, model yang dikembangkan berhasil melakukan migrasi database SQL ke NoSQL dengan kecepatan rerata 0.978 detik.

Keywords: *migrasi database, sql ke nosql, basis data relational, database nosql, relational db ke nosql*

I. PENDAHULUAN

Teknologi digital saat ini telah menyentuh hampir seluruh aspek dari kehidupan manusia mulai dari perbankan, pendidikan sampai dengan komunikasi. Tiap aspek dari penggunaan teknologi digital ini menghasilkan data dalam jumlah besar yang biasa juga disebut dengan *Big Data*. *Big Data* memiliki potensi untuk menghasilkan informasi baru melalui proses analisis data seperti *Data Mining* dan *Machine Learning* [1]. Namun demikian, membangun infratruktur penyimpanan *Big Data* bukanlah pekerjaan yang mudah, utamanya dalam hal penyimpanan data.

Penyimpanan data merupakan isu krusial pada teknologi *Big Data* [2]. *Big Data* membutuhkan teknologi penyimpanan data yang profisien agar dapat menyimpan data secara cepat dalam jumlah besar. Penyimpanan *Big Data* juga harus bisa menyimpan data dalam berbagai berbentuk baik terstruktur dan tidak terstruktur [3]. Sayangnya, kebutuhan tersebut gagal dipenuhi oleh hampir semua jenis *database relational* berbasis *Structured Query Language* (SQL) seperti MySQL, PostgreSQL, Oracle SQL yang banyak digunakan saat ini [2], [4]. Hal inilah yang mendorong lahirnya *database Not Only SQL* (NoSQL) [5].

NoSQL merupakan *database* yang dapat menyimpan data baik terstruktur dan tidak terstruktur dalam jumlah besar untuk *Big Data* [6]. Salah satu model *database* NoSQL yang banyak digunakan adalah *Document Oriented data* MongoDB [7]. Masalahnya, MongoDB menyimpan data dengan bentuk *key-value pair* sedangkan bentuk *database* yang banyak digunakan saat ini masih berbentuk tabel (kolom dan baris) pada *relational database*. Hal ini menyebabkan migrasi data berbentuk *relational database* menjadi sulit dilakukan.

Masalah ini sudah coba diselesaikan oleh beberapa penelitian, Ramzan [3] menggunakan pendekatan transformasi data dan data *cleansing* untuk membuat model migrasi *database relational*/SQL ke NoSQL. Pendekatan ini diklaim bisa bekerja dengan baik, namun demikian pendekatan yang dilakukan penelitian [3] mengubah bentuk data asli melalui proses data *cleasing* sehingga isi data hasil migrasi telah berubah dari yang asli. Penelitian lain juga coba dilakukan oleh Sanket [8] yang menggunakan pendekatan data *adapter*. Model yang diajukan penelitian ini yaitu dengan membuat *middleware* komunikasi yang dapat melakukan dua query sekaligus untuk *relational database* dan NoSQL [8]. Sayangnya, pendekatan ini tidak menyelesaikan masalah migrasi jika data yang ada sudah ada didalam *relational database*,

pendekatan ini hanya bekerja jika data baru ditambahkan ke *database*. Berdasarkan kekurangan-kekurangan penelitian sebelumnya, maka penelitian lanjutan mengenai migrasi *database relational* ke NoSQL sangat diperlukan.

Penelitian ini mencoba mengajukan pengembangan model perangkat lunak untuk migrasi *database relational*/SQL ke NoSQL menggunakan pendekatan data transformasi yang memanfaatkan *metadata* SQL. Penelitian ini menggunakan MariaDB/MySQL sebagai objek penelitian ini karena merupakan *database relational* yang paling banyak digunakan saat ini [9] dan MongoDB dari NoSQL karena merupakan *database* NoSQL yang paling populer [10]. Penelitian ini akan memanfaatkan *metadata* dari MySQL sebagai dasar untuk membuat model data pada MongoDB.

II. KAJIAN LITERATUR

NoSQL adalah *database* yang fleksibel untuk dapat menampung data dengan *volume* yang besar, progres perkembangan yang cepat (*velocity*) dan dalam bentuk bervariasi (*variety*). Saat ini ada beberapa *database* NoSQL yang sudah dikenal luas seperti MongoDB, GraphDB, ElasticSearch dll. *Database* NoSQL menyimpan data terstruktur ataupun tidak terstruktur dalam format *key-value pair*, karena sifat dan strukturnya tersebut, NoSQL tidak mendukung kueri SQL. Sehingga data dari *database* relasional (SQL) tidak bisa begitu saja dimasukkan ke *database* NoSQl, migrasi data perlu dilakukan.

Migrasi data dari *database relational* ke NoSQL merupakan tugas untuk mentransfomasi data yang sebelumnya tersimpan dalam bentuk relasi tabel ke bentuk *key-value pair* pada NoSQL [9], [7]. Tugas ini sangat penting untuk menyediakan infratruktur penyimpanan *Big Data*, tanpa menggangu sistem berjalan yang masih menggunakan *database relational* (MySQL, Oracle SQL dan SQL Server). Migrasi data dapat dilakukan dengan mengekstrak data pada *relational database* kemudian ditransformasikan melalui proses data transformasi ke bentuk NoSQL (*document-oriented data model*) [2]. Hasil migrasi data merupakan data dengan model *key-value pair* yang siap digunakan untuk berbagai proses analisis atau penggalian informasi baru melalui proses *Data Mining* dan *Machine Learning* [3].

Hingga saat ini penelitian mengenai pengembangan model migrasi *database relational* ke NoSQL masih terus berkembang. Evolusi penyelesaian masalah migrasi *database* berkembang mulai dari pendekatan data *adapter* [11], data *cleansing* [3], hingga saat ini bergeser pada pemanfaatan *metadata relational database* [7].

Transformasi data dari *relational database* ke NoSQL menggunakan pendekatan data *adapter* dilakukan oleh Solanke pada tahun 2017 [11], penelitian menggukan data *adapter* yang diposisikan sebagai *middleware* antar aplikasi dan *database*. Data *adapter* ini dapat menerjemahkan permintaan data untuk SQL dan NoSQL. Hasil penelitian ini bekerja sangat baik untuk infrastruktur *database Hybrid*, namun tidak menyelesaikan masalah mengenai penanganan data *existing* pada SQL *database* yang ingin di migrasikan ke NoSQL.

Penelitian lanjutan kemudian dilakukan oleh El Alami pada tahun 2017 [7]. Penelitian ini memanfaatkan *metadata* dari *database* SQL untuk membuat migrasi secara otomatis. Hasil pembacaan metadata menjadi dasar transformasi data melalui model aturan yang didefinisikan oleh peneliti. Kekurangan hasil penelitian ini adalah pada seleksi informasi *metadata* yang berakibat pada lambatnya proses migrasi. Pemanfaat *metadata* dalam melakukan proses migrasi data pada *database* SQL ke NoSQL juga dilakuakn oleh dilakukan oleh Ramzan pada tahun 2019 [3] dengan menambahkan pendekatan data *cleansing*. Penelitian ini mengembangkan model migrasi dengan dua tahap yaitu tahap data transformasi yang memanfaatkan *metadata* dan tahap data *cleansing*. Penelitian ini mengklaim bahwa model migrasi yang diajukan berhasil melakukan migrasi dan memperbaiki kualitas data. Namun demikian, perubahan isi data pada tahap data *cleansing* membuat data yang dimigrasikan bukan lagi data asli sehingga ada perbedaan pada isi data dalam *relational database* dan hasil migrasi pada NoSQL.

Berdasarkan penelitian sebelumnya yang telah dipaparkan, perkembangan penelitian tentang migrasi data SQL ke NoSQL saat ini mengarah pada pemanfaatan metadata SQL. Namun, penelitian-penelitian yang ada masih meninggalkan celah pengembangan. Model migrasi yang dapat memindahkan data dengan cepat dan akurat tanpa mengubah isi data yang asli masih perlu dikembangkan. Oleh karenanya, penelitian mengenai migrasi

database SQL ke NoSQL menjadi sangat penting dan menarik untuk diteliti.

III. METODE PENELITIAN

Penelitian ini dilakukan dengan empat tahap yaitu pengumpulan data eksperimen, membuat aturan migrasi, perancangan model migrasi dan evaluasi performa. Keempat tahap tersebut dapat dijabarkan sebagai berikut:

3.1. Data Eksperimen dan Setup

Bahan penelitian merupakan data dari website Addresseek.com yang merupakan bagian dari IDalamat.com. Kedua website tersebut merupakan jaringan portal informasi alamat [4] dengan jumlah data alamat lebih dari 300.000 data yang tersimpan dalam *relational database* MySQL. Sebagai data eksperimen, penelitian ini hanya menggunakan 200 data alamat saja ditambah dengan beberapa tabel lain seperti *parent* kategori, sub kategori, halaman statis dan *user*.

Pada penelitian ini, struktur *database* dan tabel eksperimen dijadikan ujicoba dalam membangun sistem migrasi *database relational* ke *database non-relational* atau NoSQL. Adapun pengujian dilakukan dengan menggunakan perangkat dengan spesifikasi 8GB RAM, 2 Core CPU, dan 1TB HDD *Storage.*

3.2. Membuat Aturan Migrasi

Aturan migrasi adalah regulasi yang mengatur sebuah bentuk skema pada *database relational* (SQL) ke bentuk atau skema pada *database non-relational* (NoSQL) [12] saat proses migrasi berlangsung. Aturan migrasi ini dibuat dengan sebisa mungkin mencari kesetaraan bentuk antara SQL dan NoSQL [13]. Adapun aturan yang dibuat berdasarkan aturan-aturan sesuai Table berikut:

Tabel 1. Aturan Kesetaraan Migrasi.

***Relational* DB (SQL)**	***Non-Relational* DB (NoSQL)**
Database	*Database*
Table	*Collection*
Row	*Document*
Column	*Field*
Primary Key	*_id provided by Mongodb*

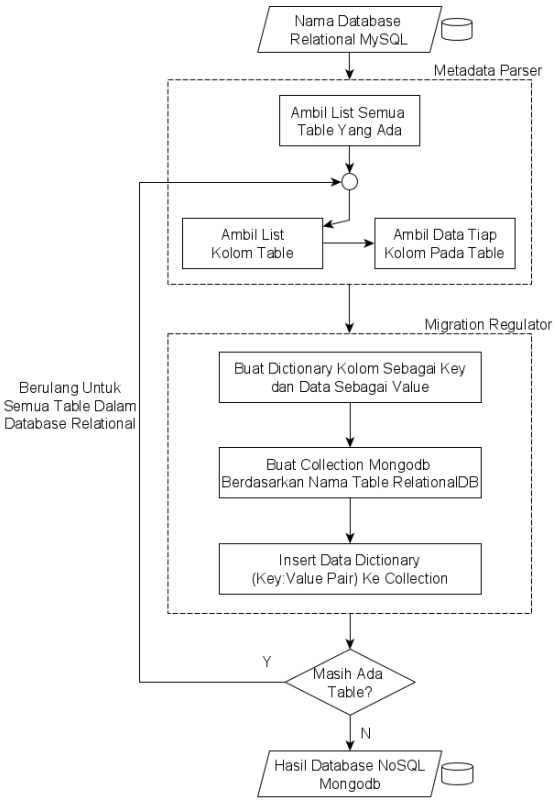
3.3. Model Migrasi yang Diajukan

Penelitian ini berfokus pada pengembangan model migrasi *database relational* ke *database* NoSQL menggunakan pendekatan transformasi data yang memanfaatkan *metadata* SQL dari *database relational*. Realisasi dari fokus penelitian tersebut diperlihatkan melalui perancangan sistem model yang digambarkan pada diagram Gambar 1.

Perancangan model pada Gambar 1 dapat dijabarkan menjadi dua langkah besar yaitu:

3.3.1. Metadata Parser

Proses *metadata paerser* berfungsi untuk mengekstrak skema dan data dari *database* ujicoba. Terdapat tiga langkah utama yang dilakukan pada tahap ini yaitu:



Gambar 1. Model Migrasi yang Diajukan

1. Ekstrasi informasi *metadata* tabel

Tahap pertama yaitu pengambilan informasi tabel-tabel yang ada pada *database*. *Query* `show tables` digunakan untuk mendapatkan informasi tersebut:

Tabel 2. *Query* Ektraksi *Metadata Table List*

Query
`> USE [nama_database];`
`> SHOW TABLES;`

Hasil dari *query* `show tables` akan menghasilkan daftar tabel-tabel yang ada dalam *database*.

2. Ekstrasi informasi kolom-kolom sebuah tabel

Untuk tiap tabel yang didapatkan pada tahap pertama, ekstrasi *metadata* kolom-kolom pada table kemudian dilakukan dengan perintah:

Tabel 3. *Query* Ektraksi Kolom Tabel

Query
`Untuk tiap table yang ada:` `> SHOW COLUMNS FROM [nama_table];`

Hasil dari perintah `show colomns` merupakan informasi kolom apa saja yang ada pada sebuah tabel.

3. Ekstrasi data sebuah tabel

Langkah ketiga yang dilakukan setelah informasi tabel dan kolomnya diketahui adalah ekstraksi data tabel. Pengambilan data pada sebuah tabel menggunakan perintah:

Tabel 4. *Query* Ektraksi Kolom Tabel

Query
`Untuk tiap table yang ada:` `> SELECT * FROM [nama_table];`

Hasil dari *query* `select` ini berupa data pada sebuah tabel dengan urutan yang sama dengan urutan kolom hasil langkah kedua.

3.3.2. Penerapan Aturan Migrasi

Data yang telah berhasil diekstrak pada tahap satu hingga tiga kemudian di pindahkan ke database MongoDB dengan menerapkan aturan migrasi pada Table 1. Adapun langkah-langkahnya dapat dijabarkan sebagai berikut:

1. Pairing kolom dan data sebagai *key-value* data

Database NoSQL dalam hal ini MongoDB memiliki format data yang berbeda dengan data dari *relational database*. Jika di *relational*, data disimpan dalam bentuk tabel, di NoSQL data disimpan dalam bentuk *key-value* [14]. Oleh karena itu data yang diperoleh pada tahap sebelumnya perlu disinkronisasikan ke bentuk *key-value* data [15]. Proses ini memanfaatkan *dictionary* Python, ilustrasi *pairing* diperlihatkan tabel berikut:

Tabel 5. *Pairing* Kolom dan Data ke Format *Key-Value*

Data Kolom	***Row* Data**	**Hasil *Pairing***
`[ 'column_1' ]`	`( data_row_1_kolom_1 )`	`{ 'column_1': data_row_1_kolom_1 }`

2. Membuat *collection* pada *database* NoSQL MongoDB

Langkah berikutnya yang dilakukan adalah membuat *Collection* pada NoSQL dengan nama yang sama dengan nama_tabel dari *relational database* MySQL. Adapun *query* pembuatan *collection* tersebut dapat dilakukan sebagai berikut:

Tabel 6. Pembuatan *Collection* MongoDB

Query
`> db.createCollection(nama_collection )`

3. *Insert* data ke *collection*

Insert data merupakan tahap akhir dari proses migrasi. Proses ini hanya dapat dilakukan jika format data *key-value* dan *collection* sudah siap. Proses *insert* dapat dilakukan dengan perintah:

Tabel 7. *Insert* Data ke MongoDB

Query
`> [nama_collection].insert_many(data_key_value)`

Pada akhirnya data akan masuk pada *database* NoSQL MongoDB. Proses ini dilakukan berulang sebanyak jumlah tabel yang akan di migrasi ke *database* NoSQL.

3.4. Evaluasi Performa

Model migrasi *database* yang dikembangkan pada penelitian ini akan dievaluasi dengan dua kriteria yaitu kesesuaian hasil migrasi berdasarkan aturan migrasi dan kecepatan proses migrasi [3]. Kecepatan migrasi diukur menggunakan Persamaan 1 berikut:

$$\bar{T} = \sum_{t=1}^{n} \frac{pt_1 + pt_2 + \cdots + pt_n}{n} \qquad (1)$$

Persamaan diatas merupakan perhitungan kecepatan migrarasi tabel dimana pt merupakan kecepatan migrasi satu tabel pada percobaan ke-n. Untuk tiap tabel (t) perhitungan kecepatan dilakukan sebanyak lima kali $(n = 5)$, sehingga rerata kecepatan tiap tabel akan dibagi lima. Untuk mendapat

total kecepatan keselurah, rerata kecepatan tiap table kemudian di jumlahkan.

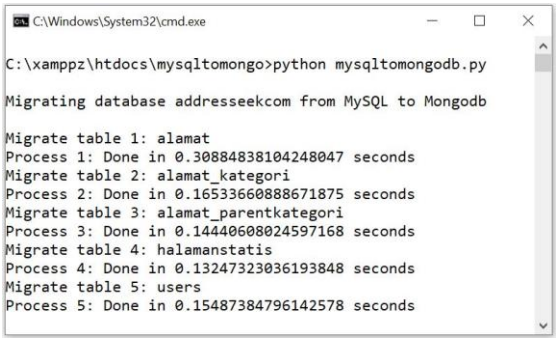
Pada akhirnya, menggunakan perhitungan kecepatan Persamaan 1, akan diperoleh nilai waktu kecepatan dari model migrasi *database* yang telah dikembangkan.

IV. HASIL DAN PEMBAHASAN

Hasil penelitian ini dibagi menjadi dua yaitu hasil migrasi *database* dan performa kecepatan migrasi. Masing-masing hasil dapat dijabarkan sebagai berikut:

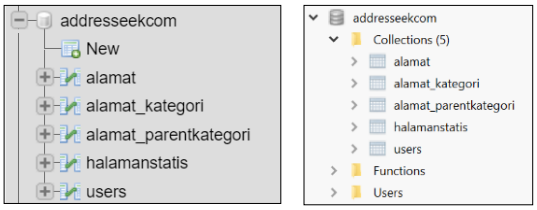
4.1. Hasil Migrasi

Model dibuat menggunakan bahasa pemograman Python. Eksekusi model dilakukan melalui *command line* seperti terlihat pada Gambar 2:



Gambar 2. Eksekusi Model Migrasi *Database*

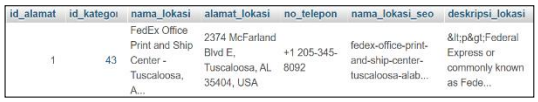
Hasil dari migrasi *database relational* (SQL) ke NoSQL yang dilakukan dengan model yang dikembangkan pada penelitian ini dapat dilihat pada gambar berikut:



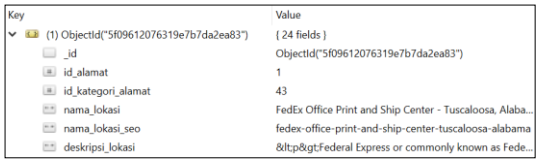
A. MySQL (SQL) B. MongoDB (NoSQL)

Gambar 3. Perbandingan *Database* Setelah Migrasi

Dapat terlihat bahwa semua tabel pada *database* MySQL berhasil di pindahkan ke *database* MongoDB. Tidak hanya tabel saja, kolom dan data pada setiap tabel juga berhasil dipindahkan seperti terlihat pada gambar berikut:



A. Kolom Table MySQL (SQL)



B. Hasil Migrasi Ke MongoDB (NoSQL)

Gambar 4. Perbandingan Kolom dan Data Setelah Migrasi

Gambar diatas memperlihatkan bahwa model yang dibuat telah berhasil melakukan migrasi data dari *database relational* / SQL (MySQL) ke database NoSQL (MongoDB).

4.2. Kecepatan Migrasi

Pengukuran kecepatan migrasi dilakukan sebanyak lima kali untuk tiap tabel. Untuk tiap pengukuran, hasil dicatat dan direratakan, pada akhirnya rerata kecepatan migrasi tiap table akan dijumlahkan dan dibagi dengan total table yang diolah. Hasil pengukuran kecepatan dari model migrasi yang diajukan terlihat oleh Table 8 berikut:

Tabel 8. Kecepatan Migrasi Model Migrasi

pt	**Table**					**Total**
	1	**2**	**3**	**4**	**5**	
1	0.309	0.165	0.144	0.132	0.155	0.906
2	0.265	0.139	0.141	0.147	0.188	0.880
3	0.519	0.155	0.147	0.152	0.167	1.140
4	0.286	0.163	0.179	0.131	0.220	0.979
5	0.287	0.181	0.208	0.154	0.155	0.987
$\overline{T}$	0.333	0.161	0.164	0.143	0.177	**0.978**

Table diatas memperlihatkan bahwa model migrasi *database relational* ke NoSQL yang dikembangkan dapat bekerja dengan cepat. Kecepatan rerata migrasi 5 tabel *database relational* hanya membutuhkan waktu rata-rata 0.978 detik.

V. KESIMPULAN

Sebuah model migrasi *database relational* ke NoSQL pada penelitian ini dikembangkan. Model didesain memiliki aturan migrasi untuk menerjemahkan bentuk *database* SQL ke NoSQL. Berdasarkan eksperimen yang telah dilakukan dengan data eksperimen berupa 1 (satu) *database* dengan 5 (lima) tabel, model yang dikembangkan mampu melakukan proses migrasi *database* dengan kecepatan rata-rata 0.978 detik untuk sekali migrasi.

Penelitian lanjutan dapat dilakukan dengan melengkapi aturan *join table* pada *database relational* menjadi *embedded data* pada *database* NoSQL.

UCAPAN TERIMA KASIH

Penelitian ini didukung oleh Addressek.com dan IDalamat.com dalam penyediaan data eksperimen.

REFERENSI

[1] M. N. Y. Utomo, A. E. Permanasari, E. Tungadi, and I. Syamsuddin, "Determining single tuition fee of higher education in Indonesia: A comparative analysis of data mining classification algorithms," in *Proceedings of 2017 4th International Conference on New Media Studies, CONMEDIA 2017*, pp. 113–117, 2017.

[2] S. Hamouda and Z. Zainol, "Document-Oriented Data Schema for Relational Database Migration to NoSQL," in Proceedings - *2017 International Conference on Big Data Innovations and Applications, Innovate-Data 2017*, 2018.

[3] S. Ramzan, I. S. Bajwa, B. Ramzan, and W. Anwar, "Intelligent Data Engineering for Migration to NoSQL Based Secure Environments," *IEEE Access*, vol. 7, pp. 69042–69057, 2019.

[4] M. N. Y. Utomo, A. Bastian, and A. Winursito, "Improving Speed Performance of Select Random Query in SQL Database," *INTEK: Jurnal Penelitian*, vol. 7, no. 1, pp. 26–31, 2020.

[5] E. Tungadi, I. Thalib, and M. N. Y. Utomo, "Machine Learning Penentuan Penerima Beasiswa Peningkatan Prestasi Akademik (PPA) Menggunakan Metode Jaringan Saraf Tiruan (JST)," *Seminar Nasional Hasil Penelitian & Pengabdian Kepada Masyarakat (SNP2M)*, pp. 391–396, 2018.

[6] Rosdiana, E. Tungadi, Z. Saharuna, and M. N. Y. Utomo, "Analisis Sentimen pada Twitter terhadap Pelayanan Pemerintah Kota Makassar," *Seminar Nasional Teknik Elektro dan Informatika*, pp. 87–93, 2019.

[7] A. El Alami and M. Bahaj, "Migration of a relational databases to NoSQL: The way forward," *International Conference on Multimedia Computing and Systems*, pp. 18–23, 2017.

[8] S. Ghule and R. Vadali, "Transformation of SQL system to NoSQL system and performing data analytics using SVM," *Proceedings - International Conference on Trends in Electronics and Informatics (ICEI 2017)*, pp. 883–887, 2018.

[9] W. Puangsaijai and S. Puntheeranurak, "A comparative study of relational database and key-value database for big data applications," *International Electrical Engineering Congress (iEECON 2017)*, pp. 8–10, 2017.

[10] D. Liang, Y. Lin, and G. Ding, "Mid-model design used in model transition and data migration between relational databases and NoSQL databases," *Proceedings - 2015 IEEE International Conference on Smart City / SocialCom / SustainCom (SmartCity)*, pp. 866–869, 2015.

[11] G. B. Solanke and K. Rajeswari, "SQL to NoSQL transformation system using data adapter and analytics," *Proceedings - 2017 IEEE International Conference on Technological Innovations in Communication, Control and Automation (TICCA)*, pp. 59–63, 2017.

[12] Y. T. Liao *et al.*, "Data adapter for querying and transformation between SQL and NoSQL database," *Future Generation Computer Systems*, vol. 65, pp. 111–121, 2016.

[13] W. Ali, M. U. Shafique, M. A. Majeed, and A. Raza, "Comparison between SQL and NoSQL Databases and Their Relationship with Big Data Analytics," *Asian Journal of Research in Computer Science*, vol. 4, no. 2, pp. 1–10, 2019.

[14] M. N. Y. Utomo, T. B. Adji, and I. Ardiyanto, "Prediksi Geolokasi Berbasis Teks untuk Data Media Sosial Berbahasa Indonesia Menggunakan Named Entity Extraction," *Universitas Gadjah Mada*, 2018.

[15] K. Ma, B. Yang, and A. Abraham, "Asynchronous Data Translation Framework for Converting Relational Tables to Document Stores," *International Journal of Computers and Applications*, vol. 38, no. 1, pp. 19–28, 2016.