INTEGRASI BASIS DATA RELASIONAL DAN NOSQL PADA SISTEM INFORMASI TERDISTRIBUSI

INTEGRATION OF RELATIONAL AND NOSQL BASIS DATAS IN DISTRIBUTED SYSTEMS

Hasmi[1], Supriyono[2], Wahyu Aji Laksono[3], Daffa Ali darajat Satriyo Pamburitan[4], Ahmad Danish Raditya[5], Alfin Najah Hasan[6], Rizaldi Saputra[7], Ferdian Surya Agung[8*]

E-mail: 240605110238@student.uin-malang.ac.id[1], priyono@ti.uin-malang.ac.id[2], 240605110188@student.uin-malang.ac.id[3], 240605110193@student.uin-malang.ac.id[4], 240605110217@student.uin-malang.ac.id[5], 240605110233@student.uin-malang.ac.id[6], 240605110142@student.uin-malang.ac.id[7], 230605110075@student.uin-malang.ac.id[8]

[1,2,3,4,5,6,7,8]Teknik Informatika, Sains dan Teknologi, UIN Maulana Malik Ibrahim Malang

Abstrak

Meningkatnya kebutuhan terhadap pengolahan data yang skalabel dan berperforma tinggi pada sistem terdistribusi mendorong penggunaan arsitektur basis data hibrida yang menggabungkan paradigma relasional dan NoSQL. Penelitian ini mengusulkan pendekatan integrasi sistem manajemen basis data yang memanfaatkan keunggulan basis data relasional dalam menjaga konsistensi data dan mendukung query terstruktur, serta keunggulan NoSQL dalam menangani data berskala besar, tidak terstruktur, dan berkecepatan tinggi. Model yang diusulkan diimplementasikan dalam lingkungan sistem terdistribusi untuk mendukung pemrosesan data secara real-time serta fleksibilitas skema data. Serangkaian eksperimen dilakukan untuk mengevaluasi kinerja sistem, meliputi waktu eksekusi query, tingkat konsistensi data, dan skalabilitas pada berbagai beban kerja. Hasil penelitian menunjukkan bahwa pendekatan integrasi ini mampu meningkatkan kinerja dan fleksibilitas sistem secara signifikan dibandingkan penggunaan basis data relasional atau NoSQL secara terpisah. Selain itu, penelitian ini juga mengidentifikasi trade-off antara konsistensi dan ketersediaan data, serta memberikan rekomendasi strategi partisi dan sinkronisasi data yang optimal. Penelitian ini berkontribusi dalam pengembangan sistem manajemen basis data yang efisien dan adaptif untuk aplikasi modern, khususnya dalam konteks big data dan komputasi awan.

Kata kunci*: *Basis Data Hibrida, Basis Data Relasional, NoSQL, Sistem Terdistribusi, Integrasi Data, Skalabilitas

Abstract

The increasing demand for scalable and high-performance data processing in distributed systems has driven the adoption of hybrid database architectures that combine relational and NoSQL paradigms. This study proposes an integrated data management approach that leverages the strengths of relational database in maintaining database consistency and structured queries, alongside NoSQL database handling large-scale, unstructured, and high-velocity data. The proposed model is implemented within a distributed system environment to support real-time data processing and flexible schema management. A series of experiments are conducted to evaluate system performance, including query execution time, data consistency, and scalability under varying workloads. The results demonstrate that the integrated approach significantly improves system performance and flexibility compared to standalone relational or NoSQL systems. Furthermore, the study highlights the trade-offs between consistency and availability,

providing insights into optimal data partitioning and synchronization strategies This research contributes to the development of efficient and adaptive database management systems suitable for modern distributed applications, particularly in big data and cloud computing environments.

***Keywords**: Keywords: Hybrid Database, Relational Database, NoSQL, Distributed Systems, Data Integration, Scalability*

1. PENDAHULUAN

Transformasi digital yang terjadi dalam dekade terakhir telah mengubah lanskap arsitektur sistem informasi secara fundamental. Munculnya aplikasi berbasis data intensif seperti media sosial, Internet of Things (IoT), analitik big data, dan layanan cloud computing telah menciptakan tuntutan yang kompleks terhadap sistem manajemen basis data SMBD (Sistem Manajemen Basis Data) [1], [2], [3]. Basis data relasional yang telah mendominasi industri selama lebih dari empat dekade menghadapi keterbatasan signifikan dalam menangani volume data yang masif (skala petabyte), kecepatan pertumbuhan data yang eksponensial, serta keberagaman struktur data yang tidak dapat diprediksi sebelumnya[4].

Sistem basis data relasional (RDBMS) seperti MySQL, PostgreSQL, dan Oracle dirancang berdasarkan model data yang terstruktur dan skema yang rigid, dengan penekanan kuat pada properti ACID (Atomicity, Consistency, Isolation, Durability) untuk menjamin konsistensi transaksional. Model relasional yang dikembangkan oleh Codd [23], mendasarkan dirinya pada teori himpunan dan aljabar relasional, yang mengharuskan data diorganisir dalam tabel-tabel dengan skema yang dideklarasikan secara eksplisit sebelum data dapat disimpan (skema-on-write). Namun, karakteristik ini justru menjadi hambatan ketika dihadapkan pada kebutuhan skalabilitas horizontal, fleksibilitas skema, dan latensi rendah yang dibutuhkan oleh aplikasi modern. Keterbatasan fundamental meliputi kesulitan dalam partisi data horizontal (sharding), overhead join operations untuk query kompleks, dan rigiditas skema yang menghambat evolusi cepat struktur data [5], [6], [7].

Keterbatasan ini memicu munculnya gerakan NoSQL (Not only SQL) yang menawarkan pendekatan alternatif dalam pengelolaan data. Basis data NoSQL hadir dengan berbagai paradigma yang disesuaikan dengan karakteristik beban kerja spesifik: (1) Basis data dokumen (MongoDB, Couchbase) menyimpan data dalam format semi-terstruktur seperti JSON atau BSON, memungkinkan skema dinamis dan nested objects;[8] (2) Basis data graf (Neo4j, OrientDB) menggunakan struktur nodes dan edges untuk merepresentasikan data yang sangat terhubung dengan kompleksitas traversal O(1) untuk relationship queries; (3) Basis data key-value (Redis, Memcached) menyediakan akses O(1) untuk simple key lookups dengan penyimpanan in-memory untuk latensi ultra-rendah; (4) Basis data kolom lebar (HBase, Cassandra) mengorganisir data dalam column families untuk efficient analytical queries pada massive datasets [9], [10], [11].

Masing-masing paradigma NoSQL (Not only SQL) mengadopsi trade-off yang berbeda dalam spektrum teorema CAP (Consistency, Availability, Partition tolerance) yang dirumuskan oleh Brewer [24]. Teorema CAP menyatakan bahwa distributed data store hanya dapat menjamin dua dari tiga properti: Consistency (semua nodes melihat data yang sama pada waktu yang sama), Availability (setiap request mendapat response sukses atau error), dan Partition tolerance (sistem tetap beroperasi meskipun terjadi network partitions). Sistem NoSQL (Not only SQL) umumnya memilih AP (Availability + Partition tolerance) atau CP (Consistency + Partition tolerance), berbeda dengan RDBMS yang traditionally fokus pada CA (Consistency + Availability) dalam konteks non-distributed.

Evolusi kebutuhan aplikasi telah mengantarkan pada konsep polyglot persistence, yaitu pendekatan arsitektural yang menggunakan kombinasi berbagai jenis basis data dalam satu

ekosistem aplikasi [12], [13]. Pendekatan ini diusulkan oleh Sadalage dan Fowler [25] sebagai respons terhadap kompleksitas data modern. Sebuah aplikasi e-commerce kontemporer, misalnya, dapat menggunakan: PostgreSQL untuk order transactions dan payment processing (memerlukan ACID guarantees), MongoDB untuk product catalog dengan varying attributes per category, Redis untuk session management dan shopping cart dengan expiration berbasis TTL, Neo4j untuk recommendation engine based on user behavior graphs, dan Cassandra untuk user activity logs dengan high write throughput. Pendekatan ini memaksimalkan keunggulan masing-masing paradigma, namun secara bersamaan memunculkan kompleksitas signifikan dalam hal: integrasi data consistency across systems, unified skema management, cross-basis data transactions, data migration dan synchronization, serta pengembangan utilitas basis data yang bersifat cross-platform.

Salah satu tantangan fundamental dalam lingkungan polyglot persistence adalah tidak adanya representasi skema yang terpadu (unified skema representation) yang mampu mengakomodasi keberagaman model data[14]. Basis data relasional memiliki skema yang eksplisit dan dideklarasikan sebelum penyimpanan data (skema-on-write), sementara sebagian besar sistem NoSQL (Not only SQL) bersifat skemaless atau skema-on-read, di mana skema implisit ada dalam data dan kode aplikasi. Kondisi skemaless memungkinkan structural variation: objects yang belong to same entity type dapat have different sets of attributes. Misalnya, dalam collection User, beberapa documents mungkin memiliki field {name, email, phone}, sementara yang lain hanya {name, email, address}. Kondisi ini menyulitkan pengembangan perangkat lunak yang dapat bekerja secara generik melintasi berbagai platform.

Tabel 1. Perbandingan Karakteristik Paradigma Basis Data

Paradigma	Model Data	CAP Trade-off	Skalabilitas	Use Case Utama
Relasional	Tables, Rows, Columns	CA (tradisional)	Vertical	Transactional systems, ACID compliance
Dokumen	JSON/BSON documents	CP atau AP	Horizontal	Content management, catalogs, user profiles
Graf	Nodes, Edges, Properties	CA atau CP	Moderate	Social networks, fraud detection, recommendations
Key-Value	Key → Value pairs	AP	Horizontal	Session storage, caching, real-time analytics
Kolom Lebar	Column families	AP	Horizontal	Time-series data, IoT, event logging

Penelitian ini mengusulkan pendekatan integrasi berbasis unified metamodel yang dinamakan U-Skema, dirancang khusus untuk merepresentasikan skema logis dari basis data relasional dan empat paradigma NoSQL (Not only SQL) utama [15], [16], [17]. A unified metamodel for NoSQL and relational basis datas Towards unified modeling for NoSQL solution based on mapping approach;

SkiQL: A unified skema query language U-Skema mencakup abstraksi-abstraksi penting dalam pemodelan data logis seperti tipe entitas, tipe relasi, atribut, agregasi, referensi, dan hierarki pewarisan. Berbeda dengan proposal sebelumnya, U-Skema secara eksplisit merepresentasikan

structural variation yang merupakan karakteristik inheren dari sistem skemaless. Implementasi menggunakan Eclipse Modeling Framework (EMF) dan Ecore metamodeling language memberikan keuntungan strategis dalam hal tooling ecosystem dan interoperabilitas.

2. TINJAUAN PUSTAKA

2.1. Metamodel Generik untuk Basis Data

Penelitian metamodel generik dimulai sejak era DB-Main [26] Hainaut et al yang mengusulkan Generic Entity-Relationship (GER) sebagai representasi platform-independent untuk evolusi basis data. Proyek DB-Main berfokus pada tiga elemen: (i) metamodel GER sebagai ekstensi model ER; (ii) pendekatan transformasional untuk reverse dan forward engineering; (iii) history list untuk merekam perubahan skema. Namun, DB-Main difokuskan pada basis data relasional dan sistem konvensional, tidak menangani karakteristik NoSQL (Not only SQL) seperti variasi struktural dan tipe relasi graf.

Atzeni et al. [27] memperkenalkan metamodel universal dengan 13 meta-constructs untuk merepresentasikan berbagai formalisme data. Metamodel ini mencakup Abstract untuk modeling autonomous concepts, AbstractAttribute untuk references, dan Generalization untuk hierarchies. Perbedaan fundamental dengan U-Skema: metamodel universal dirancang untuk instantiate data models (SuperMetamodel/Data Model/Skemas), sedangkan U-Skema adalah unified metamodel (Ecore/U-Skema/Skemas). Selain itu, implementasi metamodel universal tidak memanfaatkan framework MDE existing seperti EMF, dan tidak mencakup relationship types serta structural variations[18].

GeRoMe [28] mengusulkan pendekatan berbasis role dengan 48 peran covering berbagai model data. Namun, pendekatan ini kurang adopsi karena bahasa pemrograman modern tidak mendukung konsep role secara native. SOS dan NoAM adalah metamodel untuk aggregate-based stores (document, key-value, column). SOS basis data terdiri dari collections containing Structs dan Attributes dengan nested capability. NoAM basis data adalah collections containing blocks dengan key-value pairs. Kedua metamodel ini tidak mencakup graf, tidak merepresentasikan variasi struktural eksplisit, dan lebih dekat physical level daripada logical.[19]

Proyek Typhon mengembangkan TyphonML untuk polyglot basis data systems. Perbedaan dengan U-Skema: (i) TyphonML bukan language pada unified metamodel; (ii) structural variation tidak dipertimbangkan; (iii) TyphonML include logical dan physical aspects untuk setiap paradigm; (iv) relationship type tidak included; (v) Aggregates tidak direpresentasikan sebagai separate concept. Tabel 2 merangkum perbandingan approaches existing.

Tabel 2. Perbandingan Approaches Metamodel Generik

Aspect	DB-Main	Universal MM	SOS/NoAM	TyphonML	U-Skema
Scope	Relational, OO	Any metadata	Aggregate NoSQL	Polyglot DB	Relational + NoSQL
Structural Variations	No	No	No	No	Yes (Explicit)
Relationship Types	FK only	Generic	No	Limited	4 types

Aspect	DB-Main	Universal MM	SOS/NoAM	TyphonML	U-Skema
MDE Framework	Custom	Custom	N/A	Custom	EMF/Ecore
Graph Support	No	No	No	Yes	Yes (Full)
Skema Extraction	Per system	Not addressed	N/A	Not detailed	MapReduce

2.2. Ekstraksi Skema dari Basis Data NoSQL

Berbagai pendekatan telah diusulkan untuk mengekstrak skema dari basis data NoSQL yang bersifat skemaless [20], [21]. Klettke et al. [29] mengusulkan teknik ekstraksi skema untuk JSON-based NoSQL data stores yang mengidentifikasi structural outliers menggunakan MapReduce untuk process large datasets. Wang et al. [30] mengembangkan framework comprehensive untuk skema management dalam document stores yang include components untuk skema extraction, summarization, dan query optimization dengan sampling-based approach. Colazzo et al. [31] proposed approach untuk typing massive JSON datasets yang analyze structure dan produce type descriptions menggunakan structural patterns.

Comyn-Wattiau dan Akoka [32] mengusulkan model-driven reverse engineering approach untuk Neo4j yang analyze CREATE Cypher statements untuk extract EER conceptual skema. Graph model formed oleh nodes dan edges, kemudian model-to-model transformation generate EER skema. Sevilla Ruiz et al. [33] developed approach untuk inferring versioned skemas dari MongoDB menggunakan MapReduce operations untuk efficiently process large collections dan identify different structural versions. Hernández Chillón et al. [34] extended work untuk generate object-document mappers secara otomatis dan developed visualization approaches untuk aggregate-oriented skemas.

Gap penelitian yang significant: (1) Lack of unified representation across paradigms; (2) Structural variation not explicitly addressed in most approaches; (3) Incomplete relationship semantics; (4) Lack of formal mappings; (5) Limited scalability considerations; (6) Absence of comprehensive validation. Research kami addresses these gaps dengan comprehensive unified metamodel, formal canonical mappings, scalable extraction strategy, dan comprehensive validation.

3. METAMODEL U-SKEMA

3.1. Definisi Formal

U-Skema didefinisikan sebagai tuple 7-elemen yang merepresentasikan logical skema dari basis data:

U-Skema = (E, R, V, F, T, parent, root)

Di mana:

- E = {e_1, e_2, ..., e□} adalah finite set entity types
- R = {r_1, r_2, ..., r□} adalah finite set relationship types (khusus graf)
- V: (E ∪ R) → P(StructuralVariation) memetakan skema type ke set variations
- F: StructuralVariation → P(Feature) memetakan variation ke set features
- T adalah type system dengan T = T_Primitive ∪ T_Collection ∪ {Null}
- parent: (E ∪ R) → (E ∪ R) ∪ {⊥} mendefinisikan inheritance (⊥ = no parent)
- root: E → {true, false} mengindikasikan apakah entity type merupakan root

StructuralVariation didefinisikan sebagai sv = (id, count, firstTS, lastTS, features) di mana id adalah unique identifier, count adalah jumlah objek dengan struktur tersebut, firstTS dan lastTS adalah timestamps untuk objek pertama dan terakhir, dan features adalah set properti yang membentuk variasi struktural tersebut.
Feature hierarchy mencakup Attribute (scalar/structured properties), Aggregate (embedded objects), Reference (links ke independent entities), dan Key (identifier attributes). Formally: Feature = Attribute ∪ Aggregate ∪ Reference ∪ Key. U-Skema mendukung empat jenis relasi berbeda: agregasi (embedding dengan lifecycle dependency), referensi (link ke independent entity), graph relationship (first-class relationship dengan attributes), dan generalisasi (IS-A relationship).

4. STRATEGI EKSTRAKSI SKEMA

4.1. Proses Two-Stage

Ekstraksi U-Skema models melibatkan proses two-stage untuk NoSQL stores. Stage 1 menggunakan operasi MapReduce untuk infer logical skema yang implicit dalam data. Stage 2 applies forward mapping rules untuk create U-Skema model dari variation skemas yang telah diekstrak. Untuk relational basis datas, hanya Stage 2 diperlukan karena skemas sudah declared. Design goals mencakup: scalability (menangani basis data sangat besar), completeness (extract semua variasi struktural), accuracy (identifikasi entities/relationships yang benar), dan performance (waktu ekstraksi yang reasonable)[22].

4.2. Stage 1: Inferensi Skema MapReduce

Operasi Map mengekstrak raw skema untuk setiap stored object. Raw skema adalah intermediate representation dalam format JSON-like yang mendeskripsikan data structure. Algoritma menerapkan transformation rules: (1) setiap nilai primitif diganti dengan symbol representing typenya (String → 's', Number → 0, Boolean → true); (2) embedded objects diproses secara rekursif; (3) arrays ditransformasi menjadi array of type representations; (4) references diidentifikasi menggunakan heuristics (property name matches entity type, values match _id values).
Operasi Reduce mengkoleksi semua identical raw skemas dan menghasilkan single variation skema untuk setiap structural variation. Reduce phase melakukan: eliminating duplicate raw skemas, counting occurrences dari setiap variation, dan recording temporal information (first dan last timestamps). Untuk graph systems, process diulangi untuk obtain raw skemas dari relationship types, extracting structure dari arc properties dan connections mereka.

4.3. Stage 2: Konstruksi Model U-Skema

Stage kedua menganalisis variation skemas untuk membangun U-Skema model menggunakan Builder pattern. Proses meliputi: (1) Initialize USkemaModel dengan basis data name; (2) Identify entity types dari variation skemas; (3) Create StructuralVariations dengan unique identifiers; (4) Build Features (Attributes, Aggregates, References, Keys) dari properties; (5) Establish relationships connecting References ke target EntityTypes dan linking Keys ke constituent Attributes; (6) Handle RelationshipTypes khusus untuk graph basis datas.

4.4. Pemetaan Kanonik

Canonical mapping didefinisikan sebagai pemetaan yang memenuhi dua kondisi: (1) Forward-complete: rules correctly map semua characteristics dari data model ke U-Skema concepts; (2) Trivially bidirectional: original basis data skema dapat reproduced dari U-Skema model. Untuk setiap paradigma (Graph, Document, Key-Value, Columnar, Relational), canonical mapping rules dispesifikasikan secara formal menggunakan notasi: u ↔ m || {property relations}, di mana u adalah element dari U-Skema dan m adalah element dari data model individual.

Property relations dapat berupa: $p_1 = p_2$ (properties have same value), p ← v (value v assigned to property p), $e_1 \leftrightarrow e_2$ (enclosed mapping between elements), atau map(e, t) (obtain target element type t mapping to source e). Mapping rules defined untuk: Graph basis datas (nodes, edges, multi-label entities, relationship types), Document basis datas (collections, embedded objects, references), Key-Value basis datas (flattened object-key pattern), Columnar basis datas (column families, UDTs), dan Relational basis datas (tables, primary keys, foreign keys).

5. IMPLEMENTASI DAN VALIDASI

5.1. Implementasi Per Paradigma

Implementasi dilakukan untuk enam sistem basis data populer. Neo4j (graf) menggunakan modified strategy dengan preliminary serialization stage menggunakan Spark connector, karena graph basis datas umumnya tidak menyediakan facilities untuk efficiently process entire graph. MongoDB (dokumen) mengikuti standard two-stage process dengan special handling untuk aggregation hierarchies melalui recursive processing. Redis (key-value) memerlukan preliminary stage untuk join properties dari each entity variation dengan simple MapReduce assuming flattened object-key pattern encoding[23].

HBase (kolom lebar) applies standard strategy dengan MapReduce identifying default dan aggregated column families. Cassandra menggunakan API untuk retrieve declared skema metadata, kemudian build U-Skema directly dari metadata karena skema already declared. MySQL (relational) implementation straightforward karena skemas already declared; process directly constructs U-Skema dari JDBC metadata dengan mapping tables ke EntityTypes, columns ke Attributes, primary keys ke Keys, dan foreign keys ke References.

5.2. Setup Eksperimen

Eksperimen dilakukan pada Intel Core i7-6700 CPU @ 3.40 GHz dengan 48 GB RAM dan SSD storage. Empat dataset berbeda ukuran dibuat berdasarkan running example 'User Profiles' dengan ukuran Small (100k User, 50k Movie), Medium (200k User, 100k Movie), Large (400k User, 200k Movie), dan Larger (800k User, 400k Movie). Tabel 3 menunjukkan detail ukuran datasets.

6. HASIL EKSPERIMEN DAN VALIDASI

6.1. Setup Eksperimen

Eksperimen dilakukan pada Intel Core i7-6700 CPU @ 3.40 GHz dengan 48 GB RAM dan SSD storage. Empat dataset berbeda ukuran dibuat berdasarkan running example 'User Profiles'. Tabel 3 menunjukkan ukuran datasets yang digunakan.

Tabel 3. Ukuran Dataset untuk Eksperimen Performance

Size	User	Movie	Watched/ Favorite	Total Nodes (Graph)	Total Relationships (Graph)
Small	100,000	50,000	3/user	250,000	550,000
Medium	200,000	100,000	5/user	500,000	1,700,000
Large	400,000	200,000	10/user	1,000,000	6,400,000
Larger	800,000	400,000	20/user	2,000,000	24,800,000

6.2. Performance dan Skalabilitas

Untuk mengekspresikan skalabilitas proses inferensi skema secara meaningful, kami menggunakan baseline query agregasi yang menghitung rata-rata watched movies per user. Query ini representatif untuk query yang menghasilkan laporan periodik. Tabel 4 menunjukkan hasil performance untuk semua basis data implementations.

Tabel 4. Hasil Performance: Waktu Inferensi dan Rasio terhadap Query Baseline

Basis data	Metric	Small	Medium	Large	Larger
Neo4j	Query (ms)	686	1,213	3,165	12,016
	Inference (ms)	11,821	20,177	41,814	109,724
	Ratio	17.23x	16.63x	13.21x	9.13x
MongoDB	Query (ms)	295	380	840	2,366
	Inference (ms)	5,187	6,730	10,226	23,452
	Ratio	17.58x	17.71x	12.17x	9.91x
HBase	Query (ms)	931	1,942	6,419	24,023
	Inference (ms)	5,615	6,840	11,526	49,042
	Ratio	6.03x	3.52x	1.80x	2.04x (Best)
Redis	Query (ms)	1,002	2,833	10,091	43,888
	Inference (ms)	11,487	22,013	61,505	252,794
	Ratio	11.46x	7.77x	6.10x	5.76x

Key findings dari performance analysis: (1) Ratio consistently menurun seiring basis data size increases, demonstrating good scalability; (2) HBase menunjukkan best performance dengan ratio 2.04x untuk larger basis data; (3) Neo4j dan MongoDB menunjukkan higher ratios kemungkinan karena query baseline benefits dari built-in optimizations; (4) Semua implementations maintain reasonable performance (< 18x slower than aggregate query untuk small basis datas, improving ke 2x-10x untuk larger basis datas).

Tabel 5. Hasil Validasi Correctness untuk Semua Basis data Systems

Basis data	Dataset	Round-Trip Validation	All Variations Exist Query	Correctness Count Query
Neo4j	User Profiles (Synthetic)	Passed	Passed	Passed
	Movies (Real-world)	Passed	Passed	Passed
MongoDB	User Profiles (Synthetic)	Passed	Passed	Passed
	EveryPolitician (Real-world)	Passed	Passed	Passed
Redis	User Profiles (Synthetic)	Passed	Passed	Passed
	EveryPolitician (Real-world)	Passed	Passed	Passed
HBase	User Profiles (Synthetic)	Passed	Passed	Passed
	EveryPolitician (Real-world)	Passed	Passed	Passed
Cassandra	User Profiles (Synthetic)	N/A (Skema Declared)	Passed	Passed
MySQL	Sakila (Real-world)	N/A (Skema Declared)	Passed	Passed

Validation results confirm: (1) Extracted skemas equivalent ke manually defined models (round-trip validation); (2) All structural variations correctly identified (all variations exist query); (3) Object counts accurate untuk each variation (correctness count query); (4) Validation successful across both synthetic dan real-world datasets; (5) Approach applicable ke all paradigms dengan consistent accuracy.

7. DISKUSI

7.1. Aplikasi dalam Basis data Engineering

U-Skema memfasilitasi berbagai utilitas basis data engineering. Migrasi Basis data: mengurangi kompleksitas dari m × (m-1) migrators menjadi m + m (m extractors dan m generators) [24], [25]. Generic Query Language: foundation untuk SQL-extended language dengan capabilities:

dot notation untuk aggregated objects, arrow notation untuk reference dereferencing, collection filtering, dan variation-specific queries. Skema Visualization: common diagram generation untuk logical skemas across paradigms dengan variation-aware visualization dan relationship type differentiation. Synthetic Data Generation: platform-independent specification generates data untuk multiple paradigms dengan variation-aware generation dan relationship constraint preservation.

7.2. Limitasi dan Trade-offs

Limitasi teknis: (1) Beberapa nuances physical-level hilang dalam logical representation; (2) Full basis data scan required untuk skema extraction, costly untuk very large basis datas; (3) Periodic re-extraction atau incremental updates needed untuk skema maintenance; (4) Mapping antara certain model features tidak perfectly lossless; (5) Constraint representation currently limited. CAP theorem trade-offs: berbagai paradigma prioritize different CAP properties - Relational traditionally CA, Document/Column often AP, Graph typically CA atau CP, Key-Value generally AP. Integration approach must carefully balance based pada application requirements[26].

Implementation challenges: (1) Reference identification heuristics may miss non-standard patterns; (2) Variation explosion dalam highly dynamic skemas; (3) Performance overhead untuk real-time skema tracking; (4) Complex aggregation hierarchies meningkatkan extraction complexity; (5) Beberapa paradigm-specific optimizations tidak captured dalam unified representation. Meskipun limitations ini, pendekatan yang diusulkan provides solid foundation untuk developing basis data-independent tools dan significantly reduces complexity dalam polyglot persistence environments.

8. KESIMPULAN DAN PENELITIAN MASA DEPAN

8.1. Kesimpulan

Penelitian ini successfully mengusulkan dan mengimplementasikan U-Skema, unified metamodel pertama yang capable merepresentasikan logical skemas dari relational basis datas dan four major NoSQL paradigms dengan explicit support untuk structural variations dan rich relationship semantics. Kontribusi utama meliputi: (1) Formal specification U-Skema metamodel menggunakan Ecore; (2) Formal definition bidirectional canonical mappings untuk setiap paradigma; (3) Scalable MapReduce-based skema extraction strategy; (4) Comprehensive implementation across enam popular basis data systems; (5) Thorough validation dengan synthetic dan real-world datasets.

Hasil eksperimen mendemonstrasikan: (1) Scalability: inference time ratios improve dari 17,58× (small basis datas) ke 2,04×-9,91× (larger basis datas); (2) Accuracy: 100% validation success rate across semua paradigms; (3) Completeness: semua structural variations correctly identified; (4) Efficiency: reasonable extraction time relative ke complex aggregate queries; (5) Applicability: successful implementation across diverse paradigms dengan consistent methodology. U-Skema provides foundation untuk developing comprehensive tooling ecosystem untuk polyglot persistence environments.[27], [28]

8.2. Arah Penelitian Masa Depan

Arah penelitian jangka pendek: (1) Physical metamodel extension untuk representing indexes, partitioning, replication strategies; (2) Expressive skema mapping language untuk specialized transformations; (3) Complete generic query language implementation dengan DML dan SDL; (4) Skema evolution framework dengan change taxonomy dan impact analysis; (5) Incremental skema extraction untuk avoiding full re-extraction[14].

Arah penelitian jangka panjang: (1) Machine learning integration untuk platform selection recommendations dan automated partitioning strategy optimization; (2) Query optimization across heterogeneous basis datas; (3) Constraint representation extension untuk validation rules; (4) Cloud-native extensions untuk serverless architectures; (5) Comprehensive tool ecosystem development including visual designers, migration assistants, comparison tools, dan lineage tracking. Expected long-term benefits: reduced development time untuk multi-basis data applications, improved maintainability, enhanced portability, better alignment business requirements dengan technical implementations, dan broader adoption advanced basis data technologies.

9. DAFTAR RUJUKAN

[1] M. E. Bakir *et al.*, "A scalable, end-to-end IoT and remote sensing platform for precision rangeland and livestock management," *Comput. Electron. Agric.*, vol. 247, no. March, p. 111615, 2026, doi: 10.1016/j.compag.2026.111615.

[2] L. J. Mwinuka, M. Cafaro, L. Pereira, and H. Morais, "Big Data Energy Systems: A Survey of Practices and Associated Challenges," *Comput. Sci. Rev.*, vol. 60, no. February, p. 100927, 2025, doi: 10.1016/j.cosrev.2026.100927.

[3] F. Hochkamp, S. Reuver, and M. Rabe, "ScienceDirect Systematic review of data integration methods in the context of data preparation for enterprise information systems," *Procedia Comput. Sci.*, vol. 278, pp. 292–299, 2026, doi: 10.1016/j.procs.2026.02.465.

[4] Y. Hajjaji, W. Boulila, and I. R. Farah, "An improved tile-based scalable distributed management model of massive high-resolution satellite images," *Procedia Comput. Sci.*, vol. 192, pp. 2931–2942, 2021, doi: 10.1016/j.procs.2021.09.065.

[5] M. Fotache, M. I. Cluci, T. Taipalus, and G. Talaba, "The effects of database normalization on decision support system performance," *Inf. Syst.*, vol. 136, no. October 2025, p. 102636, 2026, doi: 10.1016/j.is.2025.102636.

[6] J. C. Yusmita, R. Arya, J. M. Wijaya, K. M. Suryaningrum, and R. R. Siswanto, "Optimizing Database Access Strategy: A Performance Analysis Comparison of Raw SQL and Prisma ORM," *Procedia Comput. Sci.*, vol. 269, pp. 1201–1210, 2025, doi: 10.1016/j.procs.2025.09.061.

[7] A. N. Ramadhan, K. N. Pane, K. R. Wardhana, and Suharjito, "Blockchain and API Development to Improve Relational Database Integrity and System Interoperability," *Procedia Comput. Sci.*, vol. 216, no. 2022, pp. 151–160, 2022, doi: 10.1016/j.procs.2022.12.122.

[8] G. Manyam, M. A. Payton, J. A. Roth, L. V. Abruzzo, and K. R. Coombes, "Relax with CouchDB - Into the non-relational DBMS era of bioinformatics," *Genomics*, vol. 100, no. 1, pp. 1–7, 2012, doi: 10.1016/j.ygeno.2012.05.006.

[9] C. Blanco *et al.*, "Security policies by design in NoSQL document databases," *J. Inf. Secur. Appl.*, vol. 65, p. 103120, 2022, doi: 10.1016/j.jisa.2022.103120.

[10] C. Blankenberg, B. Gebel-Sauer, and P. Schubert, "Using a graph database for the ontology-based information integration of business objects from heterogenous Business Information Systems," *Procedia Comput. Sci.*, vol. 196, pp. 314–323, 2021, doi: 10.1016/j.procs.2021.12.019.

[11] P. Suárez-Otero, M. J. Mior, M. J. Suárez-Cabal, and J. Tuya, "Data migration for column family database evolution," *Inf. Softw. Technol.*, vol. 187, no. April, 2025, doi: 10.1016/j.infsof.2025.107834.

[12] J. De Curtò, I. De Zarzà, and C. T. Calafate, "Integrating Polyglot Persistence with Large Language Models for Scalable Social Network Applications," *Procedia Comput. Sci.*, vol. 270, pp. 733–743, 2025, doi: 10.1016/j.procs.2025.09.193.

[13] A. Gamal, S. Barakat, and A. Rezk, “Standardized electronic health record data modeling and persistence: A comparative review,” *J. Biomed. Inform.*, vol. 114, no. December 2020, p. 103670, 2021, doi: 10.1016/j.jbi.2020.103670.
[14] D. Y. Zebua, E. D. Lase, D. K. Lahagu, J. F. Cahyani, I. Tiur, and C. Zai, “Integrasi Sistem Informasi Layanan Kampus Menggunakan Basis Data Terintegrasi,” vol. 4, no. 3, pp. 2224–2230, 2026.
[15] C. J. F. Candel, D. Sevilla Ruiz, and J. J. García-Molina, “A unified metamodel for NoSQL and relational databases,” *Inf. Syst.*, vol. 104, no. January 2021, p. 101898, 2022, doi: 10.1016/j.is.2021.101898.
[16] A. Gueidi, H. Gharsellaoui, and S. Ben Ahmed, “Towards unified modeling for NoSQL solution based on mapping approach,” *Procedia Comput. Sci.*, vol. 192, pp. 3637–3646, 2021, doi: 10.1016/j.procs.2021.09.137.
[17] C. J. Fernández Candel, J. J. García-Molina, and D. Sevilla Ruiz, “SkiQL: A unified schema query language,” *Data Knowl. Eng.*, vol. 148, no. October, p. 102234, 2023, doi: 10.1016/j.datak.2023.102234.
[18] P. Lestari and L. Belluano, “TERDISTRIBUSI PADA PANGKALAN DATA PENDIDIKAN TINGGI,” vol. 9, no. April, pp. 42–48, 2017.
[19] A. Lesmono, K. Ghozali, and S. Indrawanti, “Integrasi Data pada Aplikasi Desktop,” vol. 11, no. 2, pp. 67–72, 2022.
[20] C. J. Fernandez-Candel, A. Cleve, and J. J. García-Molina, “Towards the Automated Extraction and Refactoring of NoSQL Schemas from Application Code,” vol. 236, no. October 2025, 2025.
[21] S. Bouaziz, A. Nabli, and F. Gargouri, “Extending ETL for NoSQL: A validated Framework for Document-Oriented Warehousing,” *Procedia Comput. Sci.*, vol. 270, pp. 4114–4123, 2025, doi: 10.1016/j.procs.2025.09.536.
[22] D. Y. Yudhyarta and H. Susanti, “Pengembangan Model Integrasi Basis Data dan Sistem Manajemen Informasi untuk Optimalisasi Kecerdasan Bisnis,” vol. 4, no. 2, pp. 6094–6101, 2025.
[23] R. Setiawan, “Integrasi Teknologi Blockchain untuk Kontrol Akses yang Aman dalam Basis Data Terdistribusi,” vol. 3, no. 2, pp. 379–384, 2025.
[24] L. Rocha, F. Vale, E. Cirilo, D. Barbosa, and F. Mourão, “A framework for migrating relational datasets to NoSQL,” *Procedia Comput. Sci.*, vol. 51, no. 1, pp. 2593–2602, 2015, doi: 10.1016/j.procs.2015.05.367.
[25] R. Yangui, A. Nabli, and F. Gargouri, “Automatic Transformation of Data Warehouse Schema to NoSQL Data Base: Comparative Study,” *Procedia Comput. Sci.*, vol. 96, no. September, pp. 255–264, 2016, doi: 10.1016/j.procs.2016.08.138.
[26] W. Handiwidjojo *et al.*, “INTEGRASI BASIS DATA SYARAT MUTLAK PEMBANGUNAN SISTEM INFORMASI E-GOVERNMENT,” vol. 2009, no. semnasIF, pp. 222–227, 2009.
[27] D. Schwalbe-Koda, “mkite: A distributed computing platform for high-throughput materials simulations,” *Comput. Mater. Sci.*, vol. 230, no. July, p. 112439, 2023, doi: 10.1016/j.commatsci.2023.112439.
[28] B. El Idrissi, S. Baïna, A. Mamouny, and M. Elmaallam, “RDF/OWL storage and management in relational database management systems: A comparative study,” *J. King Saud Univ. - Comput. Inf. Sci.*, vol. 34, no. 9, pp. 7604–7620, 2022, doi: 10.1016/j.jksuci.2021.08.018.
[29] E. A. Brewer, "Towards robust distributed systems," in Proc. Annu. ACM Symp. Princ. Distrib. Comput. (PODC), Portland, OR, USA, 2000, pp. 7.
[30] P. Sadalage and M. Fowler, NoSQL Distilled: A Brief Guide to the Emerging World of

Polyglot Persistence. Boston, MA, USA: Addison-Wesley Professional, 2012.
[31] J.-M. Hick and J.-L. Hainaut, "Strategy for database application evolution: The DB-MAIN approach," in Proc. Int. Conf. Conceptual Modeling (ER), Berlin, Germany: Springer, 2003, pp. 291–306.
[32] P. Atzeni, G. Gianforme, and P. Cappellari, "A universal metamodel and its dictionary," Trans. Large Scale Data Knowl. Centered Syst., vol. 1, pp. 38–62, 2009.
[33] D. Kensche, C. Quix, M. A. Chatti, and M. Jarke, "GeRoMe: A generic role based metamodel for model management," J. Data Semantics, vol. 8, pp. 82–117, 2007.
[34] M. Klettke, U. Störl, and S. Scherzinger, "Schema extraction and structural outlier detection for JSON-based NoSQL data stores," in Proc. Conf. Database Syst. Bus. Technol. Web (BTW), 2015, pp. 425–444.
[35] L. Wang et al., "Schema management for document stores," Proc. VLDB Endow., vol. 8, no. 9, pp. 922–933, 2015, doi: 10.14778/2777598.2777601.
[36] D. Colazzo, G. Ghelli, and C. Sartiani, "Typing massive JSON datasets," in Proc. Int. Workshop Cross-Model Language Design Implementation, vol. 541, 2012, pp. 12–15.
[37] I. Comyn-Wattiau and J. Akoka, "Model driven reverse engineering of NoSQL property graph databases: The case of Neo4j," in Proc. IEEE Int. Conf. Big Data, Boston, MA, USA, 2017, pp. 453–458.
[38] D. Sevilla Ruiz, S. Feliciano Morales, and J. García Molina, "Inferring versioned schemas from NoSQL databases and its applications," in Proc. 34th Int. Conf. Conceptual Modeling (ER), Stockholm, Sweden, 2015, pp. 467–480.
[39] A. Hernández Chillón, D. Sevilla Ruiz, J. García Molina, and S. Feliciano Morales, "A model-driven approach to generate schemas for object-document mappers," IEEE Access, vol. 7, pp. 59126–59142, 2019, doi: 10.1109/ACCESS.2019.2913433.
[40] Handiwidjojo, W., Sutedjo, B., Oetomo, D., Edukatif, S., Sistem, P., Universitas, I., & Duta, K. (2009). *INTEGRASI BASIS DATA SYARAT MUTLAK PEMBANGUNAN SISTEM INFORMASI E-GOVERNMENT*. *2009*(semnasIF), 222–227.