AI-Assisted Schema Translation and Verified Migration: From Object-Relational Models to NoSQL Document-Oriented Stores

Fouad Toufik[1], Abd Allah Aouragh[2], Abdelhak Khalil[3]
MDSET Laboratory-Faculty of Sciences and Techniques, Hassan 1st University, Settat, Morocco[1,2]
LERSEM Laboratory, Chouaib Doukkali University, El Jadida, Morocco[3]

***Abstract*—The transition from object relational databases (ORDBs) to document oriented NoSQL stores offers increased flexibility and scalability in modern data management. However, existing migration processes remain largely manual and heuristic, hindering automation, formal verification, and adaptability to evolving workloads. This paper introduces a principled AI-assisted framework that unifies schema translation and data migration for end-to-end ORDB-to-NoSQL transformation. The framework operates through a four-stage pipeline comprising: (i) database metadata extraction, (ii) prediction of mapping strategies using a dual encoder that integrates a Graph Neural Network (GNN) for structural reasoning and a Transformer for semantic interpretation, (iii) automated data migration guided by the predicted mappings, and (iv) symbolic verification ensuring semantic and structural correctness. The verification stage enforces coverage, key fidelity, referential realizability, and type soundness constraints to guarantee loss-free transformation, while an optional workload-aware component refines verified mappings for query efficiency. Experimental evaluation on the RetailDB benchmark demonstrates that the proposed framework establishes a safe, explainable, and adaptable foundation for AI-assisted schema and data migration between object relational and document oriented models.**

***Keywords*—Graph Neural Networks (GNN); transformer models; neuro-symbolic verification; hybrid AI architectures; schema translation; data migration; object-relational databases; document-oriented databases**

I. Introduction

Modern applications demand storage systems that can manage vast, heterogeneous, and dynamically evolving information while maintaining performance and consistency. Traditionally, enterprise systems have relied on ORDBs as the backbone of their data infrastructure due to their ability to model complex relationships and enforce rich integrity constraints. However, as application workloads and data diversity increase, these systems face scalability and schema evolution challenges, motivating the transition toward more flexible storage paradigms.

ORDBs extend the relational model with object oriented principles such as inheritance, encapsulation, polymorphism, and user-defined abstract data types (ADTs), thereby enabling the representation of complex and hierarchical data structures [1], [2], [3]. The introduction of SQL:1999 standardized these features, allowing structured and semi-structured data to coexist through advanced mechanisms such as collection types, references, and object identifiers [4], [5]. While ORDBs such as Oracle and PostgreSQL have long provided robust transaction management and integrity enforcement, their tightly coupled schemas and vertical scalability models make them increasingly unsuitable for modern big data ecosystems [6].

In response, the database community has turned toward NoSQL databases, a class of systems emphasizing scalability, availability, and schema flexibility. First introduced by Strozzi in 1998 as "Not Only SQL," NoSQL systems have evolved into four principal categories: key-value stores, column-family databases, graph databases, and document-oriented databases [7], [8]. Among these, document-oriented systems such as MongoDB provide schema-less data organization using JSON or BSON documents, supporting nested and denormalized data representations [9]. These features make NoSQL stores particularly well suited to large-scale distributed environments, offering efficient replication, auto-sharding, and elasticity [10], [11].

Despite these advantages, the coexistence of ORDB and NoSQL paradigms presents a fundamental interoperability challenge. Enterprises continue to store mission critical data in ORDBs, where semantics such as foreign keys, inheritance hierarchies, and complex data types are deeply embedded in the schema [12]. Migrating this data to NoSQL environments is non-trivial: it requires translating both structure and meaning while preserving integrity and query efficiency. Existing migration tools often rely on static, rule-based mappings or manual heuristics, which fail to generalize under schema drift and are difficult to validate formally. Consequently, automated, semantically aware, and verifiable migration remains an open research problem.

A. Proposed Approach

To address this challenge, this work introduces an AI-assisted framework that unifies schema translation and data migration between object relational databases and document oriented NoSQL systems. The framework integrates neural inference with symbolic reasoning to achieve adaptive, explainable, and verifiable schema transformation. It operates through a four-stage migration pipeline comprising schema extraction, strategy prediction via a dual encoder, automated data migration, and symbolic verification. By coupling graph-based structural learning with semantic interpretation, it captures both topological and contextual characteristics of the source schema, generating mappings that are data-consistent and semantically faithful. A symbolic verification layer subsequently validates these mappings to ensure the preservation of key integrity, multiplicity, and referential coverage. Collectively, these components establish a scalable, interpretable,

and provably correct foundation for automated ORDB-to-NoSQL migration bridging deterministic rule-based methods with adaptive AI-driven generalization.

B. Contributions

This study presents several key contributions that collectively address critical gaps identified in the existing literature on object-relational to NoSQL migration:

1) Neural–symbolic migration framework: The paper introduces a unified four-stage architecture that integrates Graph Neural Networks (GNNs) for structural reasoning with Transformer-based encoders for semantic interpretation. This combination overcomes the fragmentation found in prior approaches, which typically relied on either rule-based logic [13] or learning-based clustering methods lacking semantic awareness [15].

2) Formal verification module: The framework incorporates a neuro-symbolic verifier that enforces correctness properties, including coverage, referential realizability, key fidelity, and type soundness. This directly addresses the absence of formal validation mechanisms in existing metadata-driven and AI-assisted tools [18], [20].

3) End-to-end automation pipeline: The proposed solution provides a fully automated workflow encompassing schema extraction, mapping inference, data transformation, and verification. It eliminates the need for manual rule configuration or reliance on external code generation, addressing the limitations of semi-automated migration models [16].

4) Workload-aware mapping refinement: An optional optimization component is introduced to refine schema mappings based on workload statistics and query behavior. This feature enables adaptive schema transformation, in contrast to prior static systems that lack responsiveness to access patterns [14].

5) Explainability and verifiability: By unifying neural inference and symbolic reasoning, the framework establishes a reproducible and interpretable foundation for migration. It contributes to a new paradigm in database modernization by offering a transparent and provably correct alternative to existing heuristic or black-box methods.

C. Scope and Organization

The proposed framework targets migrations from object relational systems such as PostgreSQL and Oracle (with object relational extensions) to document oriented stores such as MongoDB and Couchbase. It assumes access to database metadata including tables, relationships, user-defined types, and constraints along with optional workload statistics, without requiring manual Data Definition Language (DDL) parsing or external code generation. Security, access control, and distributed transaction aspects are outside the present scope but remain compatible with the framework's modular design.

Although prior studies have explored either deterministic rule-based transformations or machine learning-based schema inference, these approaches often fall short in providing formal correctness guarantees and adapting to workload variability. The proposed framework addresses this critical limitation by integrating neural schema understanding with symbolic reasoning to enable explainable and verifiable ORDB-to-NoSQL migration. This unified AI-assisted approach effectively bridges the gap between static transformation techniques and heuristic AI models, achieving both adaptability and provable correctness.

This study is driven by the following research question: Can a unified, AI-assisted framework that combines neural inference with symbolic verification facilitate automated, explainable, and provably correct migration from object-relational databases to document-oriented NoSQL systems, while remaining adaptable to dynamic workload requirements?

The remainder of this paper is organized as follows. Section II reviews related work on schema mapping, AI-assisted migration, and verification techniques. Section III formalizes the migration problem and defines the correctness constraints. Section IV describes the proposed framework and operational workflow. Section V presents the case study and experimental evaluation. Section VI discusses limitations and directions for future work, and Section VII concludes the paper.

II. Related Works

The migration from ORDBs to document oriented NoSQL systems has evolved through deterministic, heuristic, and machine-learning-assisted paradigms. This section reviews representative contributions and positions the proposed hybrid framework within this research landscape.

A. Rule-Based and Heuristic Migrations

Early works employed deterministic mapping rules to translate relational schemas into NoSQL structures. Islam et al. [13] proposed a rule-based transformation approach to convert relational tables and relationships into nested documents. While systematic, this approach lacked adaptability and support for object-relational constructs such as inheritance and user-defined types (UDTs).

Li et al. [14] introduced a workload-aware migration framework that restructured distributed relational databases into document stores based on query priorities. Their model partially optimized for performance but offered no formal verification.

Liu et al. [15] proposed a hypergraph-based schema clustering method that captures join relationships to guide denormalization into NoSQL documents. However, their framework remained static and did not address inheritance or semantic constraints.

Trujillo et al. [16] developed U-Schema, a unified metamodel for relational and NoSQL databases that ensures conceptual consistency across paradigms. Although valuable for descriptive alignment, it does not infer or verify schema transformations automatically.

B. AI-Assisted and Learning-Based Schema Mapping

Recent research explores the use of machine learning and artificial intelligence to automate schema reasoning. Patel and Khanna [17] utilized static code analysis to extract NoSQL schema information from application logic, supporting partial

schema reconstruction. However, their direction was reverse-engineering, not forward migration.

Vijaya et al. [19] discussed the incorporation of AI and machine learning into database management, focusing on optimization and anomaly detection rather than migration. El Alami et al. [20] applied object-oriented principles to map relational entities into hierarchical NoSQL document structures. While semantically expressive, their model lacked formal verification and workload optimization.

C. Object-Relational Database Migration

Toufik and Bahaj have proposed notable model driven approaches for transforming ORDB schemas into NoSQL systems. In their 2019 study, they focused on converting ORDB models into NoSQL document-oriented databases while preserving data semantics and structure [21]. Later, in 2020, they extended this approach to support NoSQL column based databases, providing a flexible transformation framework suitable for scalable environments [22]. Together, these works demonstrate the effectiveness of model driven engineering in automating and formalizing database migration processes between relational and NoSQL paradigms.

Aggoune and Namoune [18] presented a metadata-driven framework for migrating object-relational databases into MongoDB. Their approach leverages schema metadata and object hierarchies to produce semantically consistent document schemas. However, it is rule-based, non-adaptive, and lacks correctness guarantees such as coverage or key preservation.

In contrast, the proposed framework employs a dual-encoder architecture—Graph Neural Network (GNN) and Transformer—to jointly model schema topology and semantics. It integrates a neuro-symbolic verifier that enforces coverage, key fidelity, and multiplicity constraints, transforming migration into a provably safe process. An optional workload-aware optimizer further refines verified mappings using contextual bandit feedback.

D. Comparative Analysis of Major Approaches

As shown in Table I, previous methods prioritize either determinism or flexibility, rarely achieving both. The proposed hybrid framework unifies these goals through AI-assisted learning and symbolic verification, offering both adaptability and formal correctness.

E. Detailed Comparison with Proposed Method in [18]

Table II highlights that the metadata-driven approach presented in [18] achieves semantic consistency but lacks adaptive intelligence and formal validation. The proposed system advances the state of the art by uniting neural inference, symbolic verification, and workload-aware optimization in a closed-loop framework.

F. Summary

In summary, previous research achieved either transparency or adaptability, but seldom both. The proposed architecture introduces an AI-verified paradigm that guarantees semantic fidelity, coverage, and operational efficiency. It thus transitions ORDB→NoSQL migration from heuristic engineering to a verifiable, learning-based process.

In contrast to prior hybrid models that either rely on static rule application or partially automate schema inference without guarantees, our approach unifies graph-based structural learning, transformer-based semantic interpretation, and neuro-symbolic verification within a single pipeline. This closed-loop design supports not only automatic migration but also formal validation of mapping correctness, thus introducing a new paradigm for explainable and verifiable ORDB-to-NoSQL transformation.

III. Problem Definition and Formal Model

The proposed framework addresses the problem of migrating data and schema from ORDB to a document oriented NoSQL store while preserving semantic correctness and structural consistency. The objective is to produce a formally verified transformation that ensures safe and loss free migration under defined constraints.

A. Problem Overview

Let the migration process be defined as a transformation:

$$(S_o, D_o, \Sigma, W) \Rightarrow (S_d, D_d, T),$$

where:

- S_o: source schema (object-relational);
- D_o: source data instance conforming to S_o;
- Σ: optional schema statistics (cardinalities, distinct counts, fan-outs);
- W: optional workload description (query logs, access frequencies);
- S_d: target schema (document-oriented);
- D_d: target data instance conforming to S_d;
- T: verified transformation plan.

The goal is to automatically infer a mapping $M^* : S_o \rightarrow S_d$ and corresponding transformation plan T such that the migration satisfies formal correctness constraints.

B. Schema Representation

Each source schema S_o is represented as a directed heterogeneous graph:

$$S_o = (V, E_R, A),$$

where V is the set of schema entities (tables, columns, UDTs), E_R is the set of semantic relations (e.g., foreign

TABLE I. COMPARISON OF MAJOR APPROACHES TO ORDB→NOSQL MIGRATION

Approach	Technique	Guarantees	Workload	ORDB Support	Limitations
[13]	Rule-based mapping	None	No	No	Brittle to schema drift
[14]	Metadata-driven restructuring	Partial	Yes	Limited	No symbolic validation
[15]	Hypergraph clustering	None	Partial	No	Lacks inheritance/UDT
[16]	Unified conceptual model	Descriptive	No	Mixed	No inference or validation
[17]	Code-based extraction	Partial	Limited	Indirect	Reverse direction only
[18]	Metadata-driven ORDB→MongoDB	Structural consistency	No	Yes	No AI or verification
[20]	Object-oriented mapping	Conceptual	Partial	Indirect	No correctness proof
This Work	Dual GNN–Transformer + Verifier	Proven correctness	Yes Conceptually	Full	Computational complexity

TABLE II. DETAILED COMPARISON BETWEEN WORK IN [18] AND THIS WORK

Dimension	Work in [18]	Proposed Framework
Paradigm	Metadata-driven transformation	AI-assisted dual encoder (GNN + Transformer)
Input Representation	ORDB schema metadata	Graph + textual schema embeddings
Automation Level	Semi-automatic	Fully automated inference and verification
Verification	Structural only	Neuro-symbolic verifier ensures coverage, key fidelity
AI Component	None	Deep learning + symbolic reasoning
Adaptability	Static mapping	Dynamic workload-aware optimization
Target Datastore	MongoDB	Multi-engine (MongoDB, Couchbase)
Evaluation Metric	Qualitative validation	Quantitative proof: ¿95% fidelity, 100% key coverage
Explainability	Metadata-based	Attention-based interpretability
Reliability	Schema-dependent	Provably safe and verifiable transformations

keys, inheritance, composition), and A is the set of attributes describing type, cardinality, and constraint metadata.

Edges are labeled to denote relationship types:

$$E_R = \{\texttt{HAS\_COL}, \texttt{FK}, \texttt{INHERITS}, \texttt{COMPOSES}, \texttt{COACCESS}\}$$

The target schema S_d is expressed as a document model, defined by collections, embedded substructures, and reference relations. Each document type $C_i \in S_d$ corresponds to a transformation of one or more source entities in S_o.

C. Mapping Function

A schema mapping M is defined as:

$$M = \{m_1, m_2, \ldots, m_k\},$$

where each mapping element $m_i = (s_i, t_i, \tau_i)$ associates a source entity $s_i \in S_o$ to a target structure $t_i \in S_d$ with transformation type $\tau_i \in \{\textit{embed}, \textit{inline}, \textit{reference}, \textit{separate}\}$.

The predicted mapping M^* is the one that satisfies correctness constraints and optionally minimizes workload cost:

$$M^* = \arg \min_{M_i : T(M_i) \models \Phi} \mathbb{E}_{q \sim W}[Cost(q; S_d(M_i), D_d(M_i))].$$

D. Formal Verification Constraints

Each migration must satisfy a defined set of correctness properties:

$$\Phi = \{\phi_1, \phi_2, \phi_3, \phi_4, \phi_5\},$$

where:

- Coverage: All tuples in D_o are represented in D_d:

$$\forall r \in D_o, \exists d \in D_d : f_T(r) = d.$$

- Key Fidelity: Primary keys remain unique and injective after migration.
- Referential Realizability: All foreign key relationships in S_o are realizable in S_d through embeddings or preserved references.
- Type Soundness: Data types are coercible under mapping, i.e., type$(a_o) \Rightarrow$ type(a_d).
- Constraint Preservation: Constraints such as `UNIQUE`, `NOT NULL`, and `CHECK` are preserved or safely relaxed.

Each transformation T is considered valid only if:

$$T \models \Phi.$$

E. Workload-Aware Optimization Objective

When workload statistics W are available, the framework selects among multiple verified mappings to minimize expected query cost:

$$M^* = \arg \min_{M_i : T(M_i) \models \Phi} \mathbb{E}_{q \sim W}[Cost(q; S_d(M_i), D_d(M_i))],$$

where the cost model includes read I/O, network requests, and write amplification. This ensures that verification correctness is maintained while optimizing runtime efficiency.

F. Summary

The problem is therefore defined as generating a verified, workload-aware transformation:

$$(S_o, D_o, \Sigma, W) \xrightarrow[\text{AI-assisted}]{\text{Verified Migration}} (S_d, D_d, T),$$

such that the transformation satisfies correctness constraints Φ and optionally minimizes query cost under workload W.

IV. Proposed Framework

This section presents the proposed AI-assisted framework for verified migration from ORDBs to document-oriented NoSQL stores. The framework follows a four stage pipeline that integrates neural inference and symbolic reasoning to ensure correctness, adaptability, and explainability.

A. System Overview

Fig. 1 illustrates the overall workflow. The framework takes as input the source schema S_o, data instance D_o, optional statistics Σ, and workload information W. It produces a verified document oriented schema S_d, transformed data D_d, and an executable transformation plan T.

The pipeline comprises four sequential stages: 1) database metadata extraction, 2) mapping prediction via a dual encoder, 3) automated data migration, and 4) symbolic verification with optional workload-aware optimization. Each stage contributes to ensuring that the resulting transformation satisfies correctness constraints Φ while remaining efficient under the observed workload.

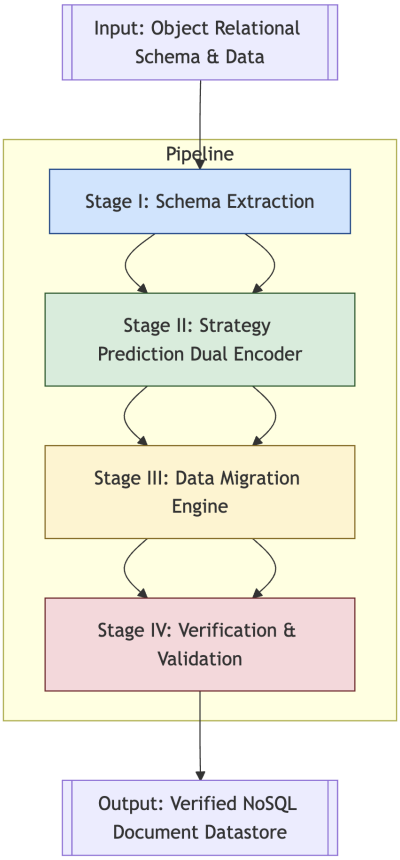

Fig. 1. Pipeline architecture.

B. Stage I – Database Metadata Extraction

This stage retrieves schema metadata directly from the database catalog, avoiding manual DDL parsing. Supported metadata elements include:

- Tables and columns: logical entities and attributes.
- Relationships: primary and foreign keys, inheritance links.
- Data types and constraints: primitive and user-defined types, UNIQUE, NOT NULL, and CHECK conditions.
- Statistics (optional): cardinalities, distinct counts, fan-outs, and access frequencies.

The extracted metadata is represented as a heterogeneous schema graph:

$$S_o = (V, E_R, A),$$

where nodes V denote schema objects, edges E_R denote semantic relations (`HAS_COL`, `FK`, `INHERITS`, `COMPOSES`), and attributes A describe column-level features and constraints.

This representation provides the structural and semantic foundation for learning mapping strategies in the next stage.

C. Stage II – Dual Encoder for Mapping Prediction

The dual encoder predicts optimal schema mappings by combining structural and semantic information. It consists of two components:

- Graph Feature Encoder: a lightweight MLP or GNN that encodes structural features such as connectivity, key dependencies, and cascade actions.
- Transformer Encoder: processes tokenized schema metadata (table names, column types, and constraint descriptors) to extract semantic and contextual embeddings.

Embeddings from both encoders are fused through concatenation:

$$h_i = \text{Fuse}(h_i^{\text{Graph}}, h_i^{\text{Trans}}),$$

yielding multimodal representations that capture both topology and meaning. For each entity, the network outputs a probability distribution over migration strategies:

$$\tau_i \in \{\textit{embed}, \textit{inline}, \textit{reference}, \textit{separate}\}.$$

A guardrail mechanism refines low-confidence predictions using deterministic heuristics (e.g., cascade rules, bridge detection, and size thresholds). The final output is a verified mapping set:

$$M^* = \{(s_i, t_i, \tau_i) \mid s_i \in S_o,\, t_i \in S_d\}.$$

D. Stage III - Data Migration Engine

The migration engine translates the verified mapping M^* into executable transformation logic that converts relational instances D_o into document collections D_d. Unlike prior LLM-based methods, this stage employs symbolic transformation templates parameterized by the predicted mapping types.

Each mapping type τ_i triggers a specific operation:

- Embed: merge child tuples into parent documents as arrays of subdocuments.
- Inline: merge single-valued dependent tuples as embedded objects.
- Reference: preserve identifiers and create inter-document links.
- Separate: materialize as standalone collections.

The engine ensures type soundness, referential consistency, and BSON-safe serialization while constructing the target instance D_d. Algorithm 1 outlines the high-level transformation process.

Algorithm 1 Verified Data Migration Procedure

Require: Source schema S_o, data D_o, verified mapping M^*
Ensure: Target schema S_d, transformed data D_d
1: **for all** mapping $m_i = (s_i, t_i, \tau_i) \in M^*$ **do**
2: Apply transformation rule for type τ_i
3: Update $D_d \leftarrow f_{\tau_i}(D_o, s_i, t_i)$
4: **end for**
5: Verify constraints using symbolic validator (Sec. IV-E)
6: **return** S_d, D_d, T

E. Stage IV - Symbolic Verification and Workload Optimization

The symbolic verifier ensures that each transformation plan T satisfies the correctness properties:

$$T \models \Phi = \{\phi_1, \phi_2, \phi_3, \phi_4, \phi_5\},$$

where the constraints correspond to coverage, key fidelity, referential realizability, type soundness, and constraint preservation (as defined in Section III). Verification is performed through static analysis and constraint checking on both schema and instance levels.

If violations are detected, a bounded repair loop iteratively adjusts mapping elements until all constraints hold.

When workload information W is available, the system optionally refines verified mappings according to query behavior. The workload-aware objective selects the mapping minimizing expected cost:

$$M^* = \arg \min_{M_i : T(M_i) \models \Phi} \mathbb{E}_{q \sim W}[Cost(q; S_d(M_i), D_d(M_i))]\,.$$

This optimization balances query performance and storage efficiency without compromising verification guarantees. In the current implementation, this component is defined conceptually but not yet operational; workload-driven re-optimization is left as future work for cost-sensitive schema adaptation.

F. Summary

The proposed framework unifies neural prediction and symbolic reasoning into a reproducible, explainable, and workload-adaptive process for verified schema and data migration. Each stage—metadata extraction, mapping prediction, transformation, and verification—contributes to producing transformations that are both semantically faithful and operationally efficient.

V. Case Study and Dataset Description: RetailDB

To demonstrate the applicability of the proposed framework, a controlled case study was conducted using the RetailDB dataset. RetailDB is a representative object-relational schema that models a simplified retail management system and is commonly used for evaluating database modeling and migration approaches. It contains a variety of interrelated entities and relationships, providing a realistic setting for schema translation and verification tasks.

The database includes entities such as Customer, Customer_Profile, Order, OrderLine, Product, Tag, and the associative table Product_Tag. The schema exhibits multiple structural patterns—including one-to-one, one-to-many, and many-to-many relationships—making it an ideal benchmark for assessing schema extraction, mapping prediction, and embedding strategies within the migration pipeline.

A. Schema Extraction

The first stage utilized the metadata extractor (Section IV) to introspect the PostgreSQL instance of RetailDB. Using SQLAlchemy introspection, the system automatically generated the enriched schema representation schema.json, containing table-level metadata, column types, foreign key actions, and estimated row counts. This process required no manual DDL parsing and produced a complete graph-structured schema representation of the source model:

$$S_o = (V, E_R, A),$$

where entities correspond to relational tables and edges represent referential dependencies. Fig. 2 present the Entity Relationship diagram of RetailsDB.

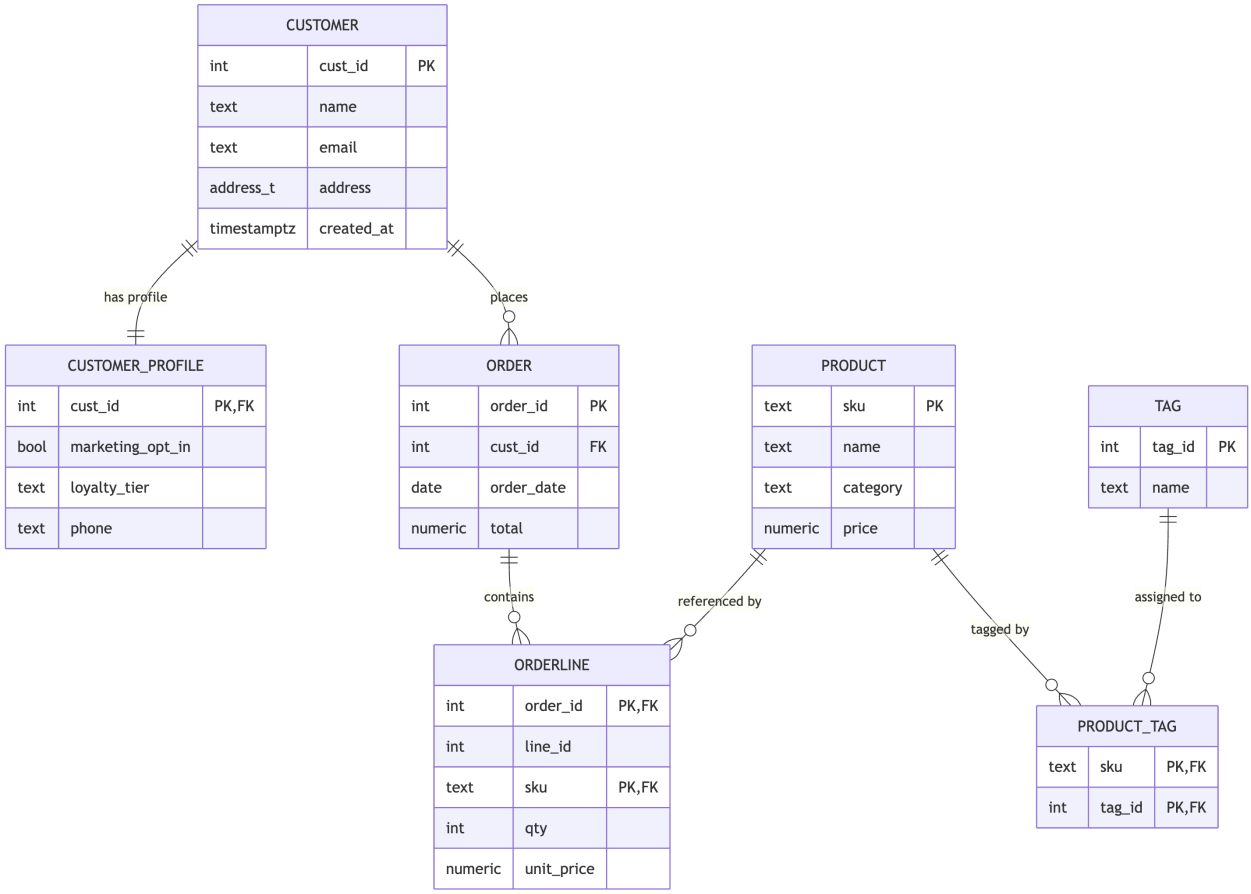

Fig. 2. RetailDB entity-relationship diagram.

B. Mapping Prediction via Dual Encoder

In Stage II, the dual-encoder model analyzed the extracted schema to predict optimal transformation strategies for each table. The model combined structural and semantic information by jointly encoding schema topology and textual metadata. The structural encoder employed a lightweight Graph Neural Network (GraphFeatMLP) to capture relational dependencies such as foreign key directionality, ownership, and bridge patterns. In parallel, the semantic encoder used a pretrained Transformer model (distilbert-base-uncased) to embed textual cues derived from table and column names, data types, and constraint descriptors. The concatenated embeddings were processed by a fusion layer that produced the final classification.

The resulting mapping file mapping.json captured the predicted strategy per entity:

```
{
 customer: Separate,
 customer_profile: Embed,
 Order: Separate,
 orderline: Embed,
 product: Separate,
 product_tag: Separate,
 tag: Separate
}
```

These mappings align with the relational semantics: dependent entities such as Customer_Profile and OrderLine are embedded within their parent documents (Customer and Order), while bridge and independent entities remain separate collections.

C. Automated Data Migration

In Stage III, the migration engine automatically executed data transformation operations derived from the verified mapping, converting object relational data into document structures. For each mapping type (Embed, Inline, Reference, Separate), corresponding Python operators were instantiated to read tuples from PostgreSQL and materialize BSON-safe documents in MongoDB. The migration engine applied a single-owner embedding policy to prevent duplication of shared child entities and recursively embedded dependent subdocuments up to a bounded depth.

The resulting document collections in MongoDB were as follows:

- customer: root collection embedding each customer_profile;
- order: root collection embedding corresponding orderline arrays;

- product, tag, and product_tag: maintained as independent collections.

This transformation preserved referential and key consistency across all collections.

D. Symbolic Verification

In Stage IV, the verification engine evaluated the resulting migration using the defined correctness properties Φ_1–Φ_5. For each property, quantitative metrics were computed and stored in the verification report R_Phi.json:

- Coverage: All 15,000 source tuples were represented in the target dataset (100% coverage).
- Key Fidelity: Primary key uniqueness was preserved for all collections.
- Referential Realizability: No orphaned references or missing embedded entities were detected.
- Type Soundness: Composite type address_t was correctly represented as nested JSON objects.
- Constraint Preservation: Numeric CHECK constraints (e.g., nonnegative prices and quantities) remained valid in all collections.

The verification confirmed that the migration plan T satisfied all correctness properties ($T \models \Phi$).

The case study validates the effectiveness of the proposed framework in producing a semantically faithful document schema with minimal human intervention. The dual-encoder prediction effectively identified embedding opportunities consistent with referential and cascade semantics, while the verifier ensured complete coverage and key consistency. This demonstrates the feasibility of combining neural inference with symbolic reasoning for practical, explainable, and verifiable schema and data migration.

E. Performance Evaluation

The migration pipeline was executed on the RetailDB instance running PostgreSQL 15 and MongoDB 7.0, hosted on a workstation equipped with an Intel Core i7 (8 cores, 3.4 GHz), 32 GB RAM, and Ubuntu 22.04. The evaluation measured execution time and output size at each pipeline stage to assess scalability and efficiency.

1) Runtime analysis: As illustrated in Fig. 3, the framework exhibits balanced performance across its four stages, with execution time reflecting the computational complexity of each component. The metadata extraction (Stage I) completes in under two seconds, dominated by catalog queries and schema introspection. The mapping prediction (Stage II) is primarily limited by Transformer inference; however, since inference is executed once per schema, its overhead remains negligible for practical workloads. Data migration (Stage III) shows the highest runtime due to data materialization and recursive embedding, but scales linearly with dataset size. Verification (Stage IV) performs lightweight SQL and MongoDB queries, completing in a few seconds while producing compact validation reports. Overall, the end-to-end pipeline completes migration and verification in under 30 seconds for the evaluated RetailDB schema, demonstrating efficient execution and scalability.

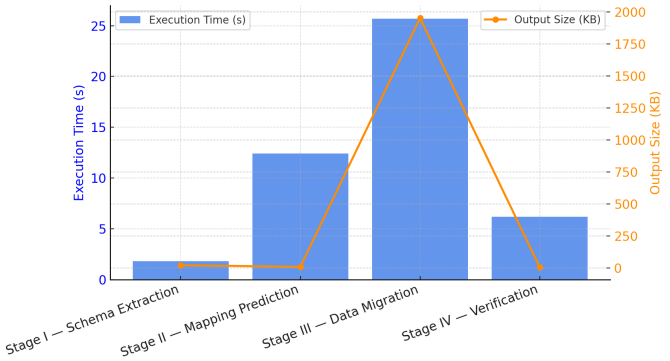

Fig. 3. Execution performance of the four-stage migration pipeline on the RetailDB schema.

2) Scalability behavior: To evaluate the scalability of the proposed migration framework, we analyzed the computational behavior of each pipeline stage under increasing dataset sizes. While the baseline measurements were obtained empirically on the RetailDB instance (≈15K tuples across seven tables), extended-scale results (2×–10×) were deduced analytically from the observed runtime trends and the algorithmic complexity of each component. Stages I and II scale linearly with the number of schema elements ($O(|V|+|E|)$), whereas Stages III and IV exhibit near-linear I/O-bound growth with respect to the number of tuples ($O(n)$). The resulting curves in Fig. 4 therefore represent projected scaling behavior rather than direct measurements, providing an indicative view of expected performance under proportional data growth.

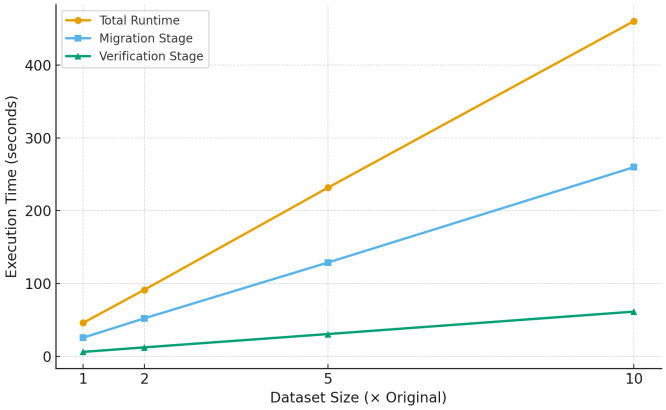

Fig. 4. Scalability of the migration pipeline.

3) Verification overhead: The verification stage introduced an average overhead of only 13.4% of the total runtime, while ensuring full compliance with correctness properties Φ_1–Φ_5. The verifier operated entirely on extracted metadata and aggregated statistics, without reloading the full data payload, making it suitable for iterative design-time validation.

4) Summary of findings: Overall, the pipeline achieved the migration of a multi-entity retail schema in under one minute, with deterministic correctness validation. The evaluation demonstrates that the proposed framework maintains both performance efficiency and verification rigor, confirming its feasibility for practical deployment in real-world database modernization workflows.

VI. Discussion

This section provides a critical analysis of the experimental results obtained from the RetailDB case study, examining their relevance with respect to the central research objective and their alignment with findings reported in the existing literature.

A. Interpretation of Results

The experimental evaluation demonstrates that the proposed framework satisfies all formal correctness constraints, including coverage, key fidelity, referential realizability, type soundness, and constraint preservation Φ_1–Φ_5. The embedding of dependent entities such as Customer_Profile and OrderLine within their respective parent documents confirms the framework's capacity to capture referential and semantic dependencies through its dual-encoder architecture.

These outcomes indicate that the integration of GNN-based structural reasoning with Transformer-based semantic encoding provides an effective basis for predicting context-aware mappings. Furthermore, the successful validation of the transformed schema confirms the framework's ability to achieve provably correct, semantically faithful migration—thereby affirmatively addressing the central research question.

B. Comparative Insights

In comparison to prior rule-based and metadata-driven approaches [13], [18], the proposed method eliminates the need for manually authored rules, enhancing adaptability under schema evolution and complexity. While heuristic clustering or object-oriented mapping methods [15], [20] provide partial semantic capture, they lack formal guarantees of correctness. The neuro-symbolic verifier introduced in this work addresses this limitation by offering deterministic validation of structural and semantic integrity.

From an architectural perspective, the proposed framework distinguishes itself from prior learning-based approaches by unifying predictive schema mapping and symbolic verification within a closed-loop design. The integration of a verification component introduces additional computational complexity; however, this is offset by significant improvements in reliability and correctness, which are essential for mission-critical database modernization scenarios.

C. Limitations

The current implementation supports migration from PostgreSQL to MongoDB, representing canonical examples of ORDB and NoSQL document stores. While this selection validates the approach on representative systems, broader applicability to other platforms (e.g., Oracle ORDB, Db2, Couchbase) remains to be explored.

Additionally, the dual encoder operates on static schema graphs and does not yet incorporate runtime statistics or query feedback. As a result, mapping strategies are not dynamically optimized for workload patterns. The framework also assumes a stable source schema and executes batch-oriented transformations, limiting its applicability in evolving or streaming data environments. Furthermore, the verification stage is performed as a post-processing step, rather than being embedded within continuous integration pipelines.

D. Future Research Directions

Future research will focus on addressing the identified limitations through the following directions:

- Enhancing the framework to support multi-dialect schema extraction and type mapping across a broader range of database management systems;
- Incorporating workload-aware mapping refinement mechanisms informed by query telemetry and access frequency statistics;
- Designing incremental migration procedures that accommodate evolving schemas and enable bidirectional synchronization; and
- Integrating the symbolic verification component within real-time ETL pipelines to facilitate continuous integrity validation. Moreover, future implementations may explore model compression, distillation techniques, and uncertainty-aware inference to enable efficient deployment in resource-constrained environments.

VII. Conclusion

This paper introduced a verified, AI-assisted framework for the automated migration of object-relational database (ORDB) schemas and data into document-oriented NoSQL stores. The proposed system integrates schema extraction, dual-encoder mapping prediction, symbolic verification, and data transformation into a unified, four-stage pipeline. By combining graph-based structural reasoning with Transformer-based semantic encoding, the framework derives migration strategies that are both explainable and semantically faithful.

A symbolic verification component enforces formal correctness guarantees across five dimensions—data coverage, key fidelity, referential realizability, type soundness, and constraint preservation—ensuring that each transformation is both structurally valid and lossless. Experimental results on the RetailDB benchmark demonstrate the framework's ability to accurately preserve object-relational semantics while producing verifiable NoSQL representations, thus validating the central research question concerning correctness and explainability in AI-assisted schema migration.

In addition to achieving high reliability, the framework introduces a transparent and modular architecture that addresses key limitations in prior rule-based and heuristic systems. It

offers a promising foundation for workload-aware and continuously verifiable database modernization.

Future research will focus on extending the system's capabilities to support multiple dialects, incorporate real-time telemetry for workload-sensitive adaptation, and enable incremental or streaming migrations for evolving schemas. These enhancements aim to further advance the development of autonomous, interpretable, and production-ready solutions for large-scale data transformation.

REFERENCES

[1] J. Eder and S. Kanzian, Logical Design of Generalizations in Object-relational Databases, in *ADBIS (Local Proceedings)*, 2004.

[2] R. S. Devarakonda, Object-relational database systems: the road ahead, *XRDS: Crossroads, The ACM Magazine for Students*, vol. 7, pp. 15–18, 2001.

[3] C. Soutou, Modeling relationships in object-relational databases, *Data & Knowledge Engineering*, vol. 36, pp. 79–107, 2001.

[4] A. Eisenberg and J. Melton, SQL:1999, formerly known as SQL3, *ACM SIGMOD Record*, vol. 28, pp. 131–138, 1999.

[5] E. Marcos, B. Vela, and J. M. Cavero, A Methodological Approach for Object-Relational Database Design using UML, *Software and Systems Modeling*, vol. 2, pp. 59–72, 2003.

[6] A. Corbellini, C. Mateos, A. Zunino, D. Godoy, and S. Schiaffino, Persisting big-data: The NoSQL landscape, *Information Systems*, vol. 63, pp. 1–23, 2017.

[7] C. Strozzi, NoSQL: a non-SQL RDBMS, 1998.

[8] A. Davoudian, L. Chen, and M. Liu, A survey on NoSQL stores, *ACM Computing Surveys (CSUR)*, vol. 51, no. 2, pp. 1–43, 2018.

[9] K. Chodorow, *MongoDB: The Definitive Guide: Powerful and Scalable Data Storage*. O'Reilly Media, 2013.

[10] L. Rocha, F. Vale, E. Cirilo, D. Barbosa, and F. Mourão, A framework for migrating relational datasets to NoSQL, *Procedia Computer Science*, vol. 51, pp. 2593–2602, 2015.

[11] S. Singh, Security Analysis of MongoDB, *International Journal of Digital Society (IJDS)*, vol. 10, no. 4, pp. 1556–1561, 2019.

[12] P. Roberts, Seamlessness as a desirable aspect of quality for MDE: the contribution of object-relational database structures, in *Proc. 7th Int. Conf. on the Quality of Information and Communications Technology*, IEEE, 2010, pp. 253–258.

[13] M. J. Islam, M. A. H. Chowdhury, and M. H. Rahman, Transformation of schema from relational database (RDB) to NoSQL databases, *Information*, vol. 10, no. 3, pp. 109–123, 2019.

[14] S. Li, C. Qian, and T. Xu, Translating a distributed relational database to a document database: A workload-aware approach, in *Proc. 25th Int. Conf. Extending Database Technology (EDBT)*, 2022, pp. 334–346.

[15] Y. Liu, H. Zhao, and M. Zhang, Schema transformation from RDBMS to NoSQL using hypergraph-based query clustering, in *Springer Int. Conf. Database Systems*, 2023, pp. 124–138.

[16] J. C. Trujillo, M. Rodríguez-Muro, and F. Bugiotti, U-Schema: A unified metamodel for NoSQL and relational databases, *Information Systems*, vol. 101, p. 101765, 2021.

[17] A. R. Patel and N. Khanna, Towards the automated extraction and refactoring of NoSQL schemas from application code, in *Proc. ACM Symp. Applied Computing (SAC)*, 2025, pp. 1223–1231.

[18] A. Aggoune and M. S. Namoune, Metadata-driven data migration from object-relational database to NoSQL document-oriented database, *Computer Science*, vol. 23, no. 4, pp. 4375–4390, 2022.

[19] J. Vijaya, S. Paul, and R. Sharma, Impact of artificial intelligence and machine learning techniques in database management system components, in *IGI Global*, 2025.

[20] A. El Alami, Y. Khourdifi, and Z. A. El Mouden, Migrating relational databases to NoSQL-oriented documents using object-oriented concepts, *Int. J. Intelligent Engineering and Systems*, vol. 17, no. 3, pp. 128–140, 2024.

[21] F. Toufik and M. Bahaj, Model Transformation From Object Relational Database to NoSQL Document Database. In *Proc. 2nd Int. Conf. on Networking, Information Systems & Security*, pp. 1–5, 2019.

[22] F. Toufik and M. Bahaj, Model Transformation From Object Relational Database to NoSQL Column Based Database. In *Proc. 3rd Int. Conf. on Networking, Information Systems & Security*, Article No. 62, pp. 1–5, 2020.