ISCSITR-International Journal of Data Engineering (ISCSITR-IJDE)
Vol. 7, Iss 1, January-June, 2026, pp. 1-7
https://iscsitr.com/journals/ISCSITR-IJDE
Journal ID: 7193-8452

Schema Evolution Handling in Semi-Structured Databases Using Graph-Based Transformation Rules

Diego Fernandez
Data Infrastructure Engineer, Argentina.

Chloe Martin
Data Warehouse Engineer, France.

Abstract

Schema evolution in semi-structured databases has become increasingly complex due to the growing diversity and dynamism of data models, particularly in NoSQL and graph-based systems. Traditional static schema approaches fail to capture the flexible nature of evolving data. This paper presents a graph-based transformation rule framework designed to manage schema evolution in semi-structured databases. It outlines the systematic modeling, mapping, and synchronization of schema elements using transformation rules embedded in graph semantics. The approach enhances flexibility, ensures data consistency, and facilitates automated evolution tracking across distributed systems.

Keywords:

schema evolution, semi-structured data, graph-based models, transformation rules, NoSQL, data modeling, versioning, data consistency, schema migration, graph databases.

How to cite this paper: Diego Fernandez, Chloe Martin. (2026). Schema Evolution Handling in Semi-Structured Databases Using Graph-Based Transformation Rules. *ISCSITR-International Journal of Data Engineering (ISCSITR-IJDE)*, 7(1), 1–7.

URL: https://iscsitr.com/index.php/ISCSITR-IJDE/article/view/ISCSITR-IJDE_2026_07_01_001/ISCSITR-IJDE_2026_07_01_001

Published: 07th January 2026

Copyright © 2026 by author(s) and International Society for Computer Science and Information Technology Research (ISCSITR). This work is licensed under the Creative Commons Attribution International License (CC BY 4.0).

http://creativecommons.org/licenses/by/4.0/ **Open Access**

1. Introductıon

Semi-structured databases have gained prominence due to their ability to handle dynamic, schema-less, or evolving data models common in modern applications such as IoT, e-commerce, and social networks. Unlike traditional RDBMS, these systems do not require rigid schemas and thus are inherently adaptable to change. However, this flexibility introduces challenges in schema evolution, including version management, backward compatibility, and consistency.

Graph-based transformation rules present a promising solution to address these issues. These rules allow for abstract modeling of schema components as nodes and relationships as edges, providing a visual and computational structure to trace changes. With graph-based modeling, schema changes become a series of transformations that can be verified, reversed, or replicated.

2. Literature Review

Early research into schema evolution in semi-structured databases primarily focused on XML and object-oriented models. Buneman et al. emphasized the foundational need for keys and schema constraints in XML-based semi-structured data, highlighting the complexity introduced when structure and content evolve independently. Hidders and Paredaens introduced a graph-based language that allowed querying and transformation of XML, laying the groundwork for transformation rules as part of evolution strategies. Additionally, Fong et al. discussed schema evolution in object-oriented databases, offering structural comparison with relational schema transformations. These works established the importance of structural flexibility and formal rule definitions for evolving semi-structured models.

With the rise of NoSQL and graph databases, more attention shifted to distributed schema management. Liu et al. examined schema evolution strategies in NoSQL data stores, proposing hybrid approaches that accommodate both backward compatibility and efficient querying. Bugiotti et al. and Angles et al. contributed significantly by modeling NoSQL systems and graph databases, addressing the complexity of schema versioning, integration, and inference. Hartmann and Link introduced inconsistency resolution strategies in graph systems, while Vavoulas and Dalamagas proposed temporal RDF evolution techniques.

These studies collectively show a transition from centralized, XML-based solutions to scalable, graph-based schema evolution frameworks, emphasizing rule-based modeling, visual transformation languages, and versioned schema tracking as core mechanisms for managing change in heterogeneous, evolving data environments.

3. Graph-Based Transformation Rule Framework

Graph-based transformation rules provide a structural mapping between schema states over time. Each schema is represented as a labeled graph, where changes are recorded as transformation rules that modify node and edge properties.

The rule types may include node addition, node deletion, edge redefinition, attribute promotion, or version-based branching. These transformations can be sequentially applied and stored in a rule engine, enabling rollback, simulation, or predictive modeling of future schema states.

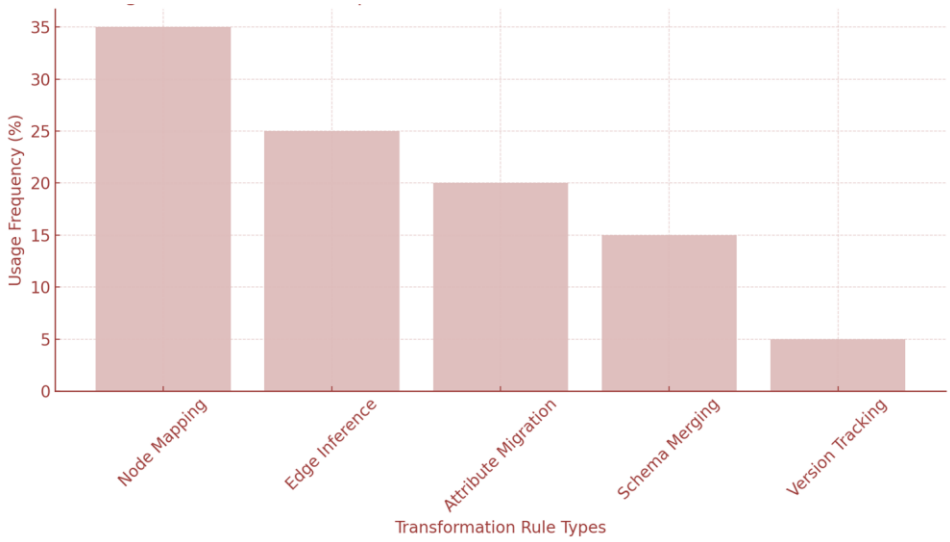

Figure 1: Usage Distribution of Graph-Based Transformation Rules in Schema Evolution

Figure 1, the relative usage of various transformation rules in practice. Node mapping and edge inference dominate, reflecting the priority of preserving data relationships during schema evolution.

4. Rule Definition and Classification

Rule-based schema transformation involves defining operations such as node splitting,

attribute reshaping, and inter-schema referencing. These rules can be classified into syntactic (structure-changing), semantic (meaning-based), and behavioral (data operation-based).

Table 1: Rule Definition and Classification

Rule Type	Description	Example Use Case
Node Mapping	Remaps schema entities to new versions	Renaming “user” to “customer”
Edge Reclassification	Changes relationships or types	Changing 1-N to N-N relationships
Attribute Promotion	Moves nested fields to top-level nodes	Moving "address.city" to "city"
Schema Merging	Combines schemas into a unified model	Merging user and profile schemas
Temporal Tracking	Stores schema versions and changes	Time-stamped schema management

These rules are encoded using graph grammars or declarative models that support both top-down and bottom-up schema evolution mechanisms.

5. Handling Evolution in Distributed Environments

In distributed semi-structured systems like document stores or graph databases, schema evolution must be synchronized across shards or nodes. The graph-based rule system supports transformation propagation, ensuring that changes made in one node reflect in others.

A transformation tracking engine records the sequence of operations and applies them to new instances or existing replicated datasets. This decentralization of schema evolution avoids data inconsistency and version drift, key challenges in multi-node deployments.

Table 2: Transformation Synchronization Example

Node ID	Schema Version	Rule Applied	Status
N1	v1.1	Attribute Migration	Applied
N2	v1.0	Pending Migration	Pending Sync
N3	v1.1	Node Addition	Applied

Such synchronization tables are critical for audit, debugging, and compliance in systems governed by regulatory standards.

6. Visualization of Schema Evolution

To manage schema evolution graphically, the system must provide a visualization dashboard. Each version of the schema is represented as a graph snapshot, with transitions animated through rule sequences.

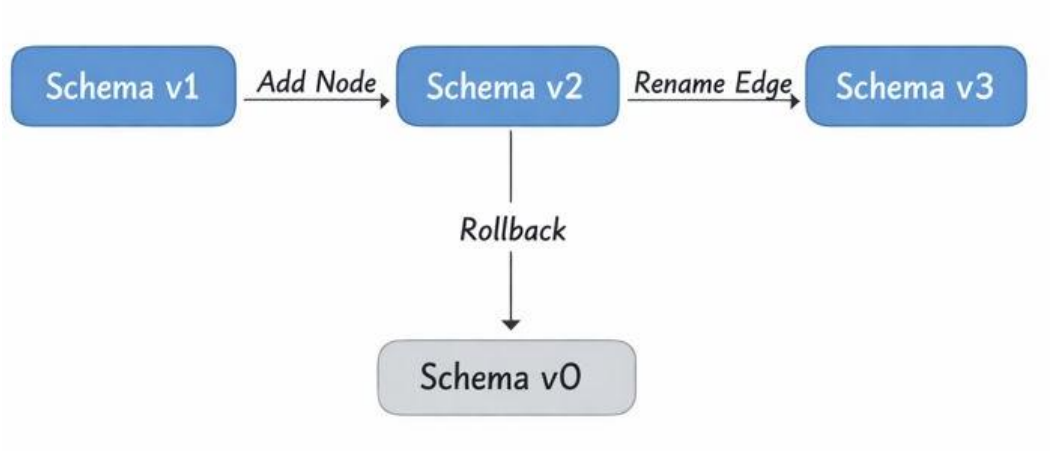

Figure 2: Schema Version Evolution with Forward Changes and Rollback Mechanism

This enables developers to trace transformations, perform impact analysis, and rollback to previous schemas as needed.

7. Advantages and Limitations

The advantages of using graph-based transformation rules include increased flexibility, semantic clarity, and maintainability. These rules abstract structural complexity and allow consistent schema management even in loosely coupled systems.

However, rule definition can become complex for large schemas. There is also overhead in maintaining transformation logs and validating rule sequences. Integration with existing schema registry systems requires customization and domain-specific rule extensions.

8. Future Directions

Future research will focus on AI-assisted transformation rule generation, automatic anomaly detection during schema evolution, and real-time rule validation using graph neural

networks. Integration with CI/CD pipelines for schema changes in database-as-a-service (DBaaS) models will be critical. Another prospective direction is semantic-aware schema evolution using ontologies and knowledge graphs, enabling high-level transformations based on domain logic rather than just structural changes.

9. Conclusion

Graph-based transformation rules offer a powerful paradigm for handling schema evolution in semi-structured databases. They combine the expressiveness of graph models with the rigor of rule-based systems, allowing for dynamic, synchronized, and traceable schema transformations. As data systems become more decentralized and complex, the need for robust schema evolution frameworks will continue to grow. This study positions graph-rule systems as a foundational layer for next-generation database adaptability.

References

[1] Roddick, J.F.: A survey of schema versioning issues for database systems. Information and Software Technology, 37(7), 383–393 (1995)

[2] Rahm, E., Bernstein, P.A.: A survey of approaches to automatic schema matching. VLDB Journal, 10(4), 334–350 (2001)

[3] Bézivin, J., Jouault, F., Valduriez, P.: First experiments with the ATL model transformation language: Transforming XSLT into XQuery. In: Proc. of the 2nd OOPSLA Workshop on Generative Techniques in the Context of Model Driven Architecture (2004)

[4] Herrmannsdoerfer, M., Vermolen, S.C., Wachsmuth, G.: COPE – automating coupled evolution of metamodels and models. In: European Conference on Object-Oriented Programming, 52–76 (2009)

[5] Wimmer, M., Kappel, G.: Model transformation in practice. In: Proceedings of the ICMT, LNCS 6707, 4–19 (2011)

[6] Papazoglou, M.P., van den Heuvel, W.J.: Service oriented architectures: approaches, technologies and research issues. VLDB Journal, 16(3), 389–415 (2007)

[7] Klettke, M., Scherzinger, S., Heuer, A.: Schema extraction and structural outlier detection for JSON-based NoSQL data stores. In: BTW Conference (2015)

[8] Hartmann, S., Link, S., Yuan, H.: Managing Schema Evolution in NoSQL Data Stores. In: International Conference on Conceptual Modeling, 327–341 (2017)

[9] Cabot, J., Gómez, C., Clarisó, R.: Building flexible model-to-model transformations with RubyTL. Software and Systems Modeling, 10(3), 325–345 (2011)

[10] Steinberg, D., Budinsky, F., Merks, E., Paternostro, M.: EMF: Eclipse Modeling Framework. Addison-Wesley Professional (2008)

[11] Jouault, F., Allilaire, F., Bézivin, J., Kurtev, I.: ATL: A model transformation tool. Science of Computer Programming, 72(1–2), 31–39 (2008)

[12] Cuesta, C.E., Molina, J., Benavides, D., Trinidad, P., Ruiz-Cortés, A.: Emfatic and text-based modeling. In: European Conference on Model Driven Architecture Foundations and Applications, 249–260 (2008)

[13] Mens, T., Van Gorp, P.: A taxonomy of model transformation. Electronic Notes in Theoretical Computer Science, 152, 125–142 (2006)

[14] Atkinson, C., Kühne, T.: Model-driven development: A metamodeling foundation. IEEE Software, 20(5), 36–41 (2003)

[15] Czarnecki, K., Helsen, S.: Classification of model transformation approaches. In: OOPSLA Workshop on Generative Techniques (2003)

[16] Frankel, D.S.: Model Driven Architecture: Applying MDA to Enterprise Computing. Wiley Publishing (2003)