Managing semantic evolution in databases: from theory to implementation

Pedro Ivo Siqueira Nepomuceno[1]
Advisor: Kelly Rosa Braghetto[1]

[1]Instituto de Matemática, Estatística e Ciência da Computação
Universidade de São Paulo
São Paulo – SP – Brazil

{pedro.siqueira, kellyrb}@ime.usp.br

Abstract. *Long-lived databases undergo continuous semantic change: categories are renamed, standards revised, boundaries merged or split. Unlike schema evolution, these value-level shifts lack principled solutions, leaving users to reconcile discrepancies by hand. This thesis formalizes semantic evolution as time-stamped operations over an interval-based storage model, and derives two approaches that answer queries transparently: query rewriting and data preprocessing. Implemented in MellowDB, a MongoDB-based Python middleware, both proved production-ready on 30 years of Brazilian mortality records: preprocessing excels in read-oriented scenarios, whereas query rewriting becomes preferable as writes dominate.*

Degree	Master
Author	Pedro Ivo Siqueira Nepomuceno
Program	Graduate Program in Computer Science Institute of Mathematics, Statistics and Computer Science (IME), University of São Paulo (USP)
Advisor	Prof. Dra. Kelly Rosa Braghetto
Defense Date	June 12th, 2025
Examining Committee	Profa. Dra. Kelly Rosa Braghetto (IME - USP) Profa. Dr. Caetano Traina (ICMC - USP) Prof. Dr. Daniel Cardoso Moraes de Oliveira (IC - UFF)

Highlights

1. Theoretical framework to define semantic heterogeneity and evolution operations
2. Models and algorithms to deal with semantic evolution were presented and evaluated
3. Brazilian mortality data used as an example of heterogeneity and thesis applicability
4. A prototypical system, MellowDB, was developed to evaluate performance
5. Common users do not need to be aware of or manually deal with semantic heterogeneity

1. Context and Problem

Long-lived databases are subject to continuous semantic change: categories are renamed, classification standards are revised, and administrative boundaries are merged or split over time. These shifts alter the interpretation of attribute values without modifying the database schema, creating semantic heterogeneity. For instance, querying population data for the state of Rio de Janeiro across all available years would silently miss records prior to 1975, when the former state of Guanabara was merged into it, unless the user is aware of this administrative change and explicitly accounts for both names in the query. Unlike schema evolution, which has been extensively studied and is supported by mature tools, semantic evolution within a single longitudinal data collection remains underexplored. Without a supporting framework, users must manually reconcile semantic discrepancies when formulating queries and interpreting results, a process that is error-prone and largely invisible to standard database tooling.

2. Objective

This work aims to establish a formal and operational foundation for managing semantic evolution in databases, addressing the lack of principled solutions for semantic heterogeneity that arises within single longitudinal data collections as meanings, groupings, and classifications change over time. To achieve this, it reviews existing solutions for schema and semantic evolution and formalizes semantic evolution as a sequence of explicit, time-stamped operations, alongside storage models that preserve the full semantic history of every record. Two approaches, query rewriting and data preprocessing, are then designed and implemented to enable users to query semantically heterogeneous data without manual adaptation, and their performance is empirically assessed and compared on a real-world dataset.

3. Contribution

The main contributions of this work are: a formal value-level model for semantic evolution, distinct from schema evolution, comprising three explicitly defined operations (translation, merging, and splitting) and their formal properties; a storage model based on interval-encoded semantic versions that preserves raw records and full semantic history with controlled overhead; two approaches for handling semantic heterogeneity transparently at query time (query rewriting) and at insertion time (data preprocessing); and MellowDB, a Python middleware prototype implementing both approaches on top of MongoDB, publicly available at `https://github.com/pisn/semantic_heterogeneous_database`. The real-world dataset used in the evaluation, comprising 30 years of Brazilian mortality records along with all 170 semantic evolution operations and analysis scripts, is publicly available at `https://doi.org/10.17632/ytfdw568kk.1`.

4. Advances over the State of the Art

Semantic heterogeneity has been studied primarily in the context of database integration, using ontology-based and taxonomy-mapping approaches to reconcile divergent representations across multiple sources [Ventrone 1991, Hakimpour and Geppert 2005, Mergen and Heuser 2006]. Schema evolution research offers related conceptual foundations, with tools such as PRIMA [Moon et al. 2008] and InVerDa [Herrmann et al. 2017]

addressing structural changes through query rewriting and delta code generation, and NoSQL-oriented work extending these ideas to schemaless environments [Klettke et al. 2016, Moller et al. 2019]. All these contributions, however, either target schema evolution or address heterogeneity across multiple databases, leaving value-level semantic evolution within a single longitudinal data collection without a formal, implementable solution. This work fills that gap by introducing explicit time-stamped semantic evolution operations, an interval-based storage model, and two empirically validated approaches that automate semantic reconciliation transparently to users.

5. Solution Overview

The proposed solution is a formal framework for managing semantic evolution in databases, implemented as MellowDB, a Python middleware built on top of MongoDB. Semantic changes are represented through discrete, time-stamped operations (translation, merging, and splitting), stored in a versioning structure that preserves raw records while maintaining full semantic history. Two approaches handle heterogeneity transparently: query rewriting, which adapts queries at execution time, and data preprocessing, which resolves discrepancies at insertion time by materializing semantic versions. In both, users query data without any awareness of historical changes. A key design challenge was controlling storage overhead in the preprocessing approach. Making all semantic versions of a record available would naively multiply storage by the number of versions. This was addressed through an interval-based representation, where a single processed record spans all consecutive versions it remains unaffected by, dramatically reducing overhead. Ensuring correctness under this design required recursive splitting and reprocessing of affected records whenever a new evolution operation is inserted between preexisting versions. On the query rewriting side, chained evolutions were handled through a stack-based algorithm that expands query clauses across semantic boundaries and normalizes results to the latest version. Experimental results confirm that both approaches achieve production-level performance, automating semantic reconciliation at a computational cost that is feasible for real-world use. This indicates that the framework is a viable foundation for future integration into database management systems.

6. Evaluation

The framework was evaluated using a real-world dataset containing 30 years of mortality records from the Brazilian public health system, with one record per cause of death, municipality, and year (6,179,083 records in total). The dataset exhibits two types of semantic heterogeneity: a disease classification standard change in 1996 (from ICD-9 to ICD-10) and municipality name changes and territorial splits over time, totaling 170 semantic evolution operations. Performance was assessed across five workload scenarios inspired by the YCSB benchmark: read-only, read-heavy (95% reads), balanced (50/50), write-heavy (95% writes), and write-only, both with and without indexing. Each experiment was repeated 10 times. The results in Figure 1 show that data preprocessing outperforms query rewriting in all scenarios except write-only. Numerically, preprocessing adds an average overhead of 0.01s per insertion, while rewriting adds 4.32s per query without indexes and 0.13s with indexes. The proportion of heterogeneous operations (15% versus 30%) had minimal impact, indicating that overhead is driven by the chosen strategy rather than the frequency of semantic events. Regarding storage, the interval-based representation kept overhead well below what a naive per-version duplication would produce.

Overall, both approaches proved computationally feasible for production use, with the optimal choice depending primarily on the read-to-write ratio of the target workload.

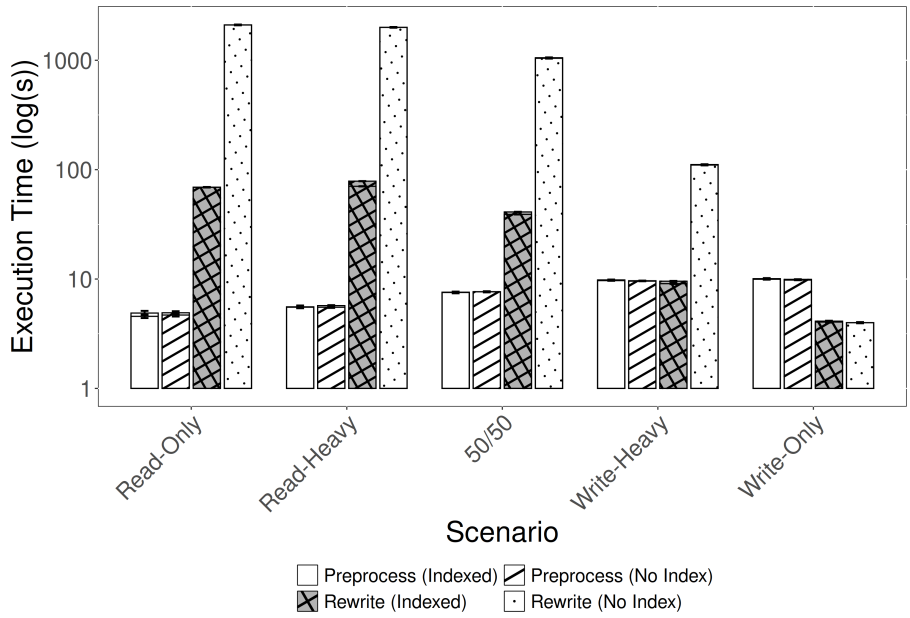

Figure 1. Performance comparison across workload scenarios with and without indexes (log scale). 500 operations in each experiment.

7. Publications

1. Nepomuceno, P. I. S.; Braghetto, K. R. Managing semantic evolution in databases: From theory to implementation. *Future Generation Computer Systems*, vol. 177, p. 108257, 2026. DOI: `https://doi.org/10.1016/j.future.2025.108257`
2. Nepomuceno, P. I. S.; Braghetto, K. R. Managing semantic evolutions in semi-structured data. In: *Proc. International Conference on Database and Expert Systems Applications (DEXA)*, 2023, p. 179–185. DOI: `https://doi.org/10.1007/978-3-031-39847-6_12`

8. Technical Production, Records and Patents

- **Software:** MellowDB – Python middleware library for managing semantic evolution in document-oriented NoSQL databases. Repository: `https://github.com/pisn/semantic_heterogeneous_database`
- **Dataset:** Brazilian public health mortality records (1979–2021), including 170 semantic evolution operations and analysis scripts. `https://doi.org/10.17632/ytfdw568kk.1`

Acknowledgements

This work was supported by the São Paulo Research Foundation (FAPESP) (grant #2023/00779-0) and the National Council for Scientific and Technological Development (CNPq) (grant #420623/2023-0).

References

Hakimpour, F. and Geppert, A. (2005). Resolution of semantic heterogeneity in database schema integration using formal ontologies. *Information Technology and Management*, 6:97–122.

Herrmann, K., Voigt, H., Behrend, A., Rausch, J., and Lehner, W. (2017). Living in parallel realities: Co-existing schema versions with a bidirectional database evolution language. In *Proceedings of the 2017 ACM International Conference on Management of Data*, SIGMOD/PODS, pages 1101–1116.

Klettke, M., Storl, U., Shenavai, M., and Scherzinger, S. (2016). NoSQL schema evolution and big data migration at scale. In *2016 IEEE International Conference on Big Data*, Big Data, pages 2764–2774.

Mergen, S. L. S. and Heuser, C. A. (2006). Data translation between taxonomies. In *International Conference on Advanced Information Systems Engineering*, pages 111–124. Springer.

Moller, M. L., Klettke, M., Hillenbrand, A., and Störl, U. (2019). Query rewriting for continuously evolving NoSQL databases. In *International Conference on Conceptual Modeling*, ER, pages 213–221.

Moon, H. J., Curino, C. A., Deutsch, A., Hou, C.-Y., and Zaniolo, C. (2008). Managing and querying transaction-time databases under schema evolution. *Proceedings of the VLDB Endowment*, 1(1):882–895.

Ventrone, V. (1991). Semantic heterogeneity as a result of domain evolution. *ACM SIGMOD Record*, 20(4):16–20.