brModeloWeb: Ten Years of History and a New Extension for the Logical Design of NoSQL Databases

Milton Bittencourt de Souza Neto[1], **Ronaldo dos Santos Mello**[1]

[1]Department of Informatics and Statistics – Graduate Program in Computer Science – Federal University of Santa Catarina, Florianópolis, SC

milton.bsn@posgrad.ufsc.br, r.mello@ufsc.br

Abstract. *The brModelo initiative began in 2005 with the purpose of providing semiautomated solutions to assist Database modeling education. Supported by the UFSC Database Group, the initiative comprises a set of tools developed over these 21 years, of which brModeloWeb is the most recent. Available since 2016, brModeloWeb reaches 10 years of existence with more than half a million registered users from universities and professionals that adopt it for both educational and real-world database design. This paper celebrates the successful adoption of brModeloWeb and presents a new extension for NoSQL logical design, introduced in the latest release, along with performance improvements. A usability evaluation conducted during a workshop at the Brazilian Database School (ERBD 2026) indicated that the tool was well received.*

1. Introduction

Database modeling is a fundamental activity in the context of an information system [Elmasri and Navathe 2022]. A well-modeled database provides an adequate abstraction of the data, while also ensuring efficient data storage and access. To achieve this goal, a database design methodology must be adopted to define suitable logical and physical schemas from a conceptual abstraction of the domain data [Batini et al. 1992].

brModelo is an initiative of the Database Group at Federal University of Santa Catarina, dedicated to the design and development of open-source tools to support the teaching of database modeling. It adopts the classical database modeling methodology, comprising the conceptual, logical, and physical design phases, along with the notations introduced by Carlos Alberto Heuser [Heuser 2024]. The most recent version of the tool is named *brModeloWeb*. It celebrates today ten years of wide adoption by academics and professionals, both for educational purposes and for real-world database projects[1].

This paper presents the latest version of *brModeloWeb*, whose main novelty is the support for the logical design of NoSQL databases based on the *aggregate* concept [Sadalage and Fowler 2013], along with usability and performance improvements. Different from related work, this is the only free and open-source tool that provides a conceptual to an aggregate-based NoSQL logical schema conversion.

The remainder of the paper is organized as follows. Section 2 discusses the related work. Section 3 presents an overview of the new functionalities. Section 4 reports a usability evaluation and Section 5 is dedicated to the conclusion.

[1]Available at `https://github.com/brmodeloweb/brmodelo-app`. Application: `https://www.brmodeloweb.com`. Demonstration video of the new version: `https://www.loom.com/share/a197a298af1f4141aef26c7de9e6bd40`.

2. Related Work

Some available tools similar to brModeloWeb offer support for modeling NoSQL databases. **Moon Modeler**[2] is a desktop tool focused on document-oriented databases, with support for MongoDB, Cosmos DB, and Amazon DocumentDB. It supports reverse engineering and the generation of physical scripts, but it is proprietary, paid (US$ 99), and does not cover conceptual modeling. **Hackolade Studio**[3] is the most comprehensive solution in this segment, with both Web and desktop versions, support for multiple NoSQL DBMSs (document, key-value, and column-oriented) and relational DBMSs, conceptual modeling, reverse engineering, and physical schema generation. However, it is also a commercial software (US$ 175/month). **DbSchema**[4] is also a proprietary and paid (starting at US$ 29.40/month) general-purpose desktop tool that covers relational databases and MongoDB, offering reverse engineering and script generation.

Table 1 shows a tools comparison. The analyzed tools provide reverse engineering and physical schema generation for several DBMSs, but they are all proprietary and paid, and none of them offers semiautomatic conversion from the conceptual model to the logical model, which is the educational focus of brModeloWeb. With the extension proposed in this work, brModeloWeb is the only free, open-source solution that covers the logical design of NoSQL databases, i.e., from a conceptual schema to an aggregate-based logical schema. This paper focuses on brModeloWeb logical modeling capabilities, as follows.

Table 1. Comparison between the analyzed tools and brModeloWeb

Criterion	Moon Modeler	Hackolade	DbSchema	brModeloWeb
Web version	×	✓	×	✓
Conceptual modeling	×	✓	×	✓
Aggregate modeling	✓	✓	✓	✓
Conceptual → logical map	×	×	×	✓
Reverse engineering	✓	✓	✓	×
Physical schema generation	✓	✓	✓	✓
Open source	×	×	×	✓
Free	×	×	×	✓

3. The New Version of brModeloWeb

The new version of *brModeloWeb* has been entirely reimplemented using the *React* library, replacing the *AngularJS* framework. The change was motivated by the end of official support for *AngularJS* in January 2022[5], which entailed risks related to application maintenance and security. The adoption of *React* provided better performance when editing complex diagrams, greater architectural modularity, and a lower barrier to external contributions, considering its wide adoption in the Web industry[6]. In addition, the

[2] https://www.datensen.com/moon-modeler-for-databases.html

[3] https://hackolade.com/

[4] https://dbschema.com/

[5] AngularJS Blog, 2022. Available at https://blog.angular.io/discontinued-long-term-support-for-angularjs-cc066b82e65a.

[6] State of JavaScript 2025. Available at https://2025.stateofjs.com/en-US/libraries/.

platform maintains compatibility with legacy models through automatic migration mechanisms. Among the new functionalities, the NoSQL module described next stands out.

The NoSQL module of *brModeloWeb* adopts the *aggregate model* as an intermediate logical representation between a conceptual schema and the physical schema of a NoSQL database. The concept of aggregates was introduced by Sadalage and Fowler [Sadalage and Fowler 2013] and formalized by Lima [Lima 2016], being suitable for high-level data abstraction across three NoSQL models: document-oriented, column-oriented and key-value. Its core concepts are *collection, block, attribute, aggregation relationship, reference relationship*, and *disjointness constraint*.

As illustrated in Figure 1, a collection represents a real-world object, generally corresponding to an entity in the conceptual model, and is the main unit of the model, mandatorily containing an identifier attribute. A block represents a nested structure inside a collection or another block, used to model embedded objects.

Attributes of collections and blocks have an associated cardinality in the (min, max) format, allowing the representation of optional, mandatory, and multivalued structures. The model defines two special types of attributes: *identifiers*, which play a role analogous to that of the primary key, and *references* (_REF suffix), which represent associations between collections without imposing referential integrity constraints. The model also includes the disjointness constraint, which indicates mutually exclusive blocks within a collection, used in the mapping of disjoint specializations from the conceptual model. An example is the disjointness between *Painting* and *Sculpture* blocks in Figure 1. The supported data types are *text, number, boolean, date, block*, and *reference*, a set aligned with the abstraction level of the aggregate model.

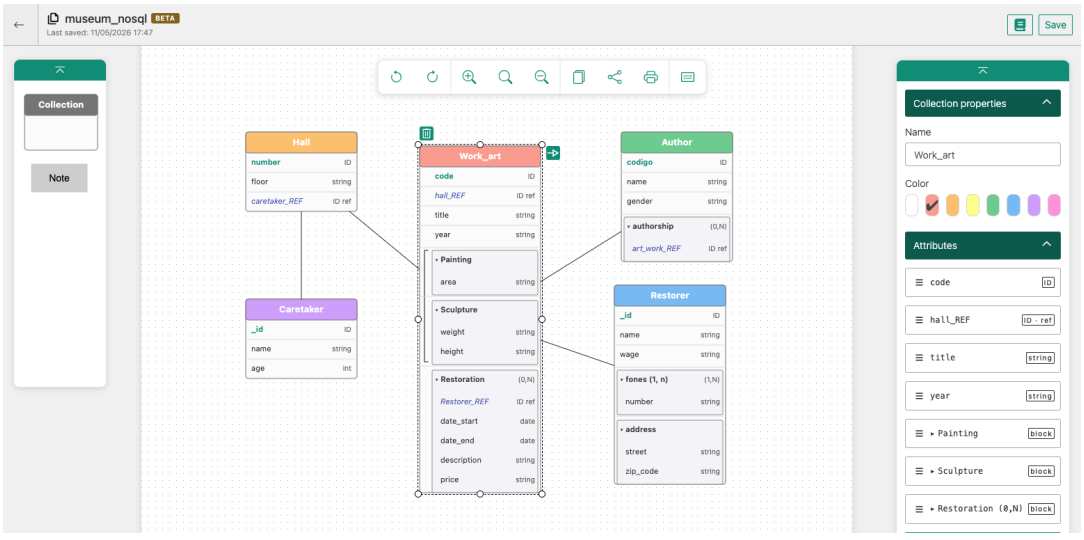

Figure 1. User interface of the NoSQL module illustrating the aggregate model concepts in a Museum domain

The module provides an automatic converter from the extended entity-relationship (EER) conceptual model to the aggregate model, organized into three phases executed in sequence: *(i) entities, (ii) specializations* and, *(iii) relationships*. This structure mirrors the relational converter already consolidated in the tool, maintaining uniformity between the conversion processes. In the first phase, each entity gives rise to a collection. Compos-

ite attributes are converted into nested blocks, and multivalued attributes can be modeled *(i)* as a nested block, *(ii)* as a separate collection that references the original collection, or *(iii)* as an attribute with explicit cardinality. Entities without a declared key automatically receive an identifier.

In the second phase, the converter handles specializations of the EER model. Based on the specialization type, it offers the designer three conversion strategies (Figure 2): *(i)* one collection per specialization, in which the attributes of the supertype are replicated in the subtype collections; *(ii)* nested blocks with a disjointness constraint, in which each subtype becomes a block within the root collection, with mutual exclusivity recorded; and *(iii)* a single generic collection with a type discriminator attribute, which unifies the entire hierarchy into a single collection. The default strategy is context-sensitive: for disjoint specializations in the conceptual phase, strategy *(ii)* is suggested.

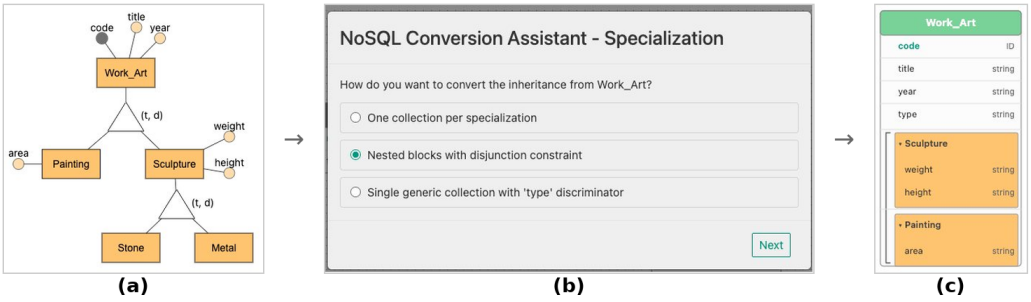

Figure 2. Converter from EER specialization to the aggregate model: (a) input conceptual schema; (b) conversion wizard with the three available strategies; (c) resulting aggregate schema considering the strategy of nested blocks with a disjointness constraint

In the third phase, relationships are converted based on the cardinality and participation constraint of each entity, as summarized in Table 2. When multiple strategies are applicable, the converter consults the designer through an interactive dialog (Figure 3). Relationship-specific attributes are preserved in all strategies, and self-relationships are processed before the others to avoid orphan references in case of later collection merging.

Table 2. Guideline for converting EER relationships to the aggregate model [Lima 2016]

Cardinality	Merge	Nesting	Reference
(1,1)-(1,1)	✓		
(1,1)-(0,1)	✓	✓	
(0,1)-(0,1)	✓	✓	✓
(1,1)-(1,n) / (1,1)-(0,n)		✓	✓
(0,1)-(0,n)		✓	✓
(m,n)-(m,n)			✓

The brModeloWeb converter adopts as a principle the designer's participation in decisions for which there is not a single possible solution. In NoSQL databases, the modeling strategy strongly depends on the expected access patterns. Thus, the same

conceptual structure may be materialized in different ways, depending on whether the predominant operations favor, for instance, aggregated reads or pointwise updates. In addition to broadening the flexibility of the transformation process, this approach has a pedagogical character by making explicit the modeling decisions that are often implicit in the implementation of the database.

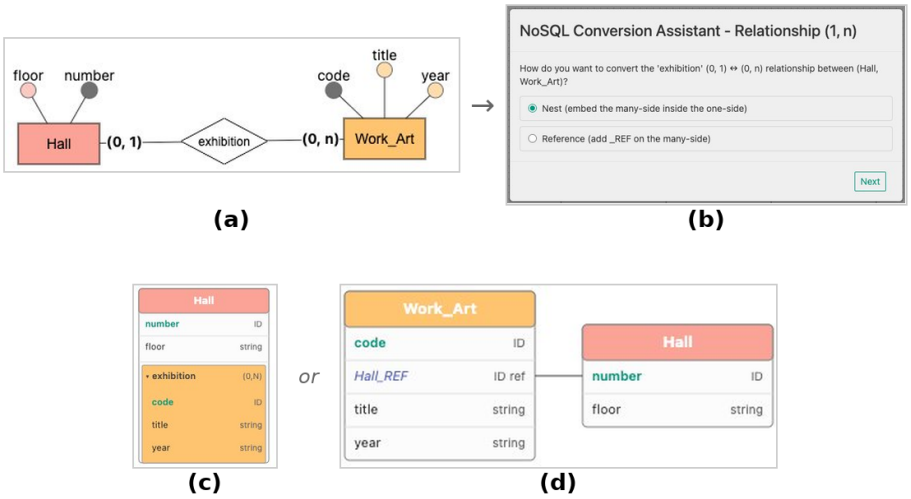

Figure 3. Converter from EER to the aggregate model applied to a 1-N relationship: (a) conceptual model with the *exhibition* relationship between Hall and Artwork; (b) conversion wizard with the two alternatives; (c) result considering the nesting strategy; (d) result considering the reference strategy.

4. Evaluation

The new version of *brModeloWeb* was evaluated during a database modeling workshop at the *Brazilian Database School (ERBD 2026)*. The activity consisted of designing an EER conceptual model followed by the derivation of a NoSQL logical model using the tool. The evaluation employed the *System Usability Scale* (SUS) questionnaire [Brooke 1996], complemented by open-ended questions.

The evaluation involved 18 participants, all of whom completed the proposed activity within the stipulated one-hour time frame. The tool obtained an average SUS score of 77.8 (median 76.3; standard deviation 8.9), a value above the threshold associated with the *Good* adjective on the scale proposed by [Bangor et al. 2009]. The highest-rated items were related to the intention of tool adoption and the low perception of complexity, while the lowest scores were associated with the need for prior learning and technical support. The open-ended responses reinforced these results, highlighting as positive points the intuitive interface and the decision dialogs during the conversion process.

The main limitations of the evaluation are the use of a convenience sample, restricted to a single event participants, and the absence of comparison with previous versions of the tool or competing solutions.

5. Conclusion

This paper presents the new version of *brModeloWeb*, which celebrates ten years of history! Originally focused on supporting the teaching of relational database design, the tool now offers a complete educational pipeline for NoSQL database design through a new module based on the aggregate model together with a semiautomatic EER converter that explicitly engages the designer in modeling decisions. By providing such a capability, *brModeloWeb* becomes, to the best of our knowledge, the first free, open-source tool to support a logical database design methodology for NoSQL stores, from a conceptual schema to an aggregate-based logical schema.

Beyond the immediate gains in performance and maintainability, the complete reimplementation in *React* opens the project to a new generation of contributors, as *AngularJS* had reached end of life and become a barrier for community engagement around an actively used educational tool. Future work includes: *(i)* physical schema generation for multiple NoSQL DBMSs (such as MongoDB, Cassandra, DynamoDB, and Firestore) a work in progress (schema export module); *(ii)* reverse engineering through a schema import module, closing the design loop currently covered only by competing proprietary tools; *(iii)* extension of the conceptual-to-logical converter to other NoSQL data models, such as graph-oriented databases, and; *(iv)* comparative evaluations with competing tools in regular undergraduate and graduate database courses.

References

Bangor, A., Kortum, P., and Miller, J. (2009). Determining what individual SUS scores mean: Adding an adjective rating scale. *Journal of Usability Studies*, 4(3):114–123.

Batini, C., Ceri, S., and Navathe, S. B. (1992). *Conceptual Database Design: An Entity-Relationship Approach*. Benjamin/Cummings.

Brooke, J. (1996). SUS: A Quick and Dirty Usability Scale. In *Usability Evaluation in Industry*, chapter 21, pages 189–194. Taylor and Francis.

Elmasri, R. and Navathe, S. B. (2022). *Fundamentals of Database Systems*. Pearson, 7 edition.

Heuser, C. A. (2024). *Projeto de Banco de Dados*. Clube de Autores, 7 edition. ISBN 978-65-01-22210-3.

Lima, C. (2016). Projeto Lógico de Bancos de Dados NoSQL Documento a Partir de Esquemas Conceituais Entidade-Relacionamento Estendido (EER). Master's thesis, PPGCC-UFSC. Available at: `https://repositorio.ufsc.br/xmlui/handle/123456789/167633`.

Sadalage, P. J. and Fowler, M. (2013). *NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence*. Addison-Wesley.