Visualization of Inferred Versioned Schemas from NoSQL Databases[*]

Alberto Hernández Chillón
alberto.hernandez1@um.es
Cátedra SAES
University of Murcia

Diego Sevilla Ruiz
dsevilla@ditec.um.es
Faculty of Computer Science
University of Murcia

Jesús García Molina
jmolina@um.es
Faculty of Computer Science
University of Murcia

September 20, 2016

Abstract

While the concept of database schema plays a central role in Relational Database systems, most NoSQL systems are schemaless: these databases are created without having to formally define its schema, that remains implicit in the stored data. This lack of schema definition offers a greater flexibility; more specifically, it eases both the recording of non-uniform data and data evolution. However, this comes at the cost of losing some of the benefits provided by schemas (such as type checking and integrity checking). In [1], a MDE-based reverse engineering approach for inferring the schema of aggregate-oriented NoSQL databases is presented. The obtained schemas can be used to build database utilities that tackle some of the problems encountered using implicit schemas. However, in the NoSQL context, the definition of a unique schema is not enough to accurately describe the stored data: different versions of the same object type coexist in the database, leading to different schema versions. In this work[1] a modeling transformation chain is designed in order to obtain these schema versions from an input database, and a visualization tool for these schemas is defined using Sirius. Sirius provides us with a workbench integrated in Eclipse able to associate graphical representations to metamodel elements, and the automatic generation of an embedded editor in Eclipse, all of this

[*]Work partially supported by the Cátedra SAES of the University of Murcia (`http://www.catedrasaes.org`), a research lab sponsored by the SAES company (`http://www.electronica-submarina.com/`).

[1]A complete description of this M.Sc. project may be found on the GitHub repository project: `https://github.com/Soltari/NoSQLVisualizationTools`.

with a suitable distribution method such as plugins and update sites. The visualization tool consists of several viewpoints according to our requirements. The most important designed viewpoint is the tree viewpoint. This representation allows the user to navigate between domain concepts such as *Schema version*, *Entity*, and *Entity version*, and to analyze these concepts with more viewpoints. The tree viewpoint also provides an inverted index and a dictionary. From this viewpoint it is possible to navigate to other diagrams such as the global schema viewpoint, which shows all the *Entities*, *Entity versions* and *Properties* in an intuitive way. Now for each inferred version schema, a couple of viewpoints are provided to analyze all the objects contained on the schema. Finally, a simple viewpoint is designed to inspect standalone *Entities*.

References

[1] Diego Sevilla Ruiz, Severino Feliciano Morales, and Jesús García Molina. Inferring Versioned Schemas from NoSQL Databases and its Applications. In *ER 2015*, pages 467–480, September 2015.

Additional Figures

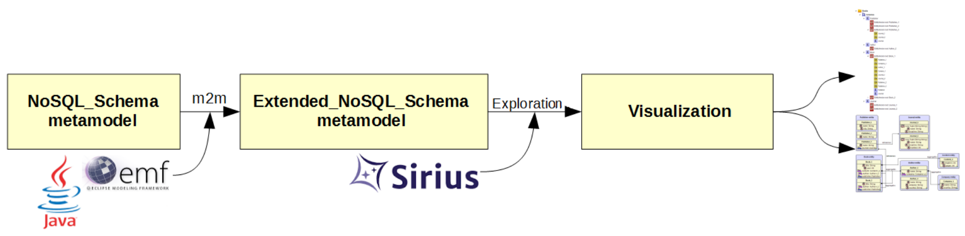
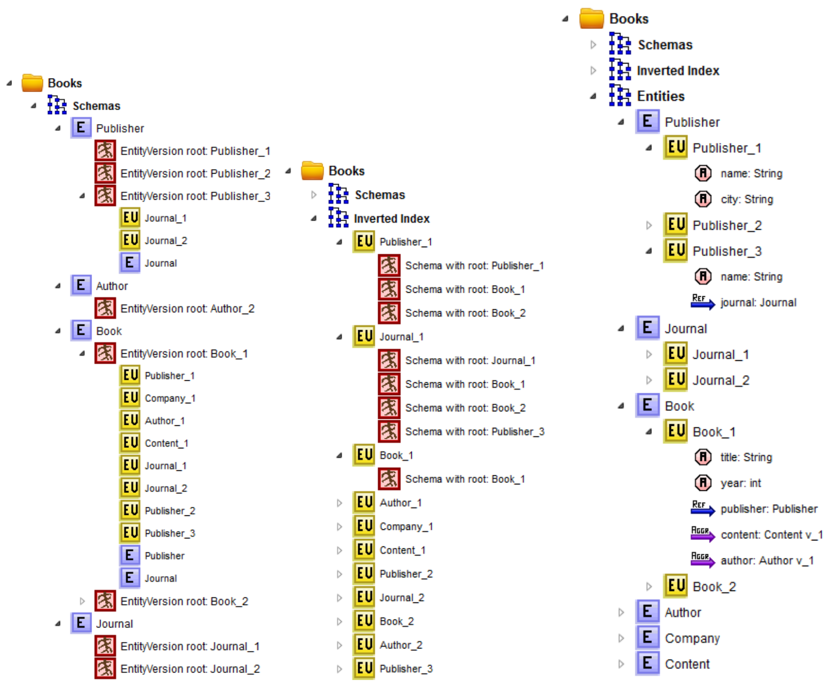


Figure 1: The process is composed of the definition of a *NoSQL_Schema* metamodel and an *Extended_NoSQL_Schema* metamodel to identify *Schema versions*. With a *m2m* transformation and some input models from the first metamodel, models of the second metamodel are generated. Now with Sirius these models may be visualized with several viewpoints.



```
Books
  Schemas
    E Publisher
      EntityVersion root: Publisher_1
      EntityVersion root: Publisher_2
      EntityVersion root: Publisher_3
        EU Journal_1
        EU Journal_2
        E Journal
    E Author
      EntityVersion root: Author_2
    E Book
      EntityVersion root: Book_1
        EU Publisher_1
        EU Company_1
        EU Author_1
        EU Content_1
        EU Journal_1
        EU Journal_2
        EU Publisher_2
        EU Publisher_3
        E Publisher
        E Journal
      EntityVersion root: Book_2
    E Journal
      EntityVersion root: Journal_1
      EntityVersion root: Journal_2
```

```
Books
  Schemas
  Inverted Index
    EU Publisher_1
      Schema with root: Publisher_1
      Schema with root: Book_1
      Schema with root: Book_2
    EU Journal_1
      Schema with root: Journal_1
      Schema with root: Book_1
      Schema with root: Book_2
      Schema with root: Publisher_3
    EU Book_1
      Schema with root: Book_1
    EU Author_1
    EU Company_1
    EU Content_1
    EU Publisher_2
    EU Journal_2
    EU Book_2
    EU Author_2
    EU Publisher_3
```

```
Books
  Schemas
  Inverted Index
  Entities
    E Publisher
      EU Publisher_1
        A name: String
        A city: String
      EU Publisher_2
      EU Publisher_3
        A name: String
        REF journal: Journal
    E Journal
      EU Journal_1
      EU Journal_2
    E Book
      EU Book_1
        A title: String
        A year: int
        REF publisher: Publisher
        AGGR content: Content v_1
        AGGR author: Author v_1
      EU Book_2
    E Author
    E Company
    E Content
```

Figure 2: This tree viewpoint is divided into three folders. The first one stores *Schema versions* sorted by *Entity*. The second folder stores *Entity versions*. The last folder contains a dictionary of *Entities*, *Entity Versions* and *Properties*. Each element may be expanded to show its components.

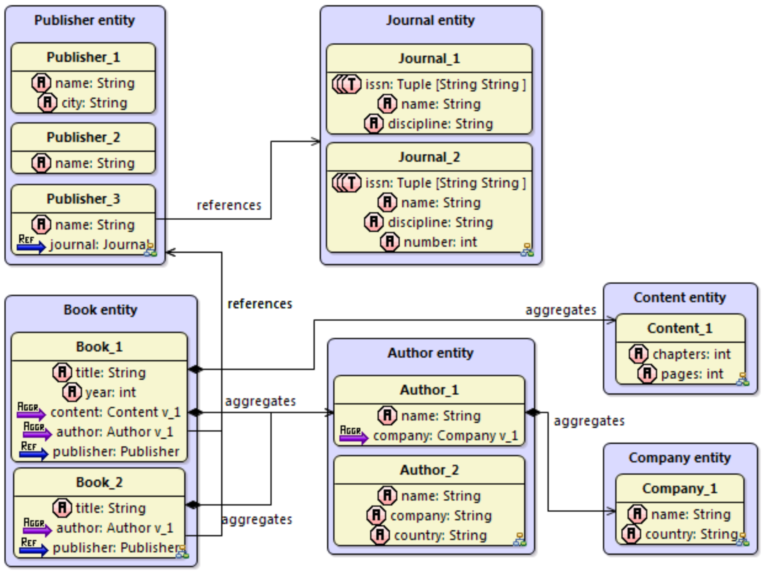


Figure 3: This global schema viewpoint shows a diagram with several box-like entities, nested elements (aggregates), elements properties and some connections. Each one of the elements is represented with a distinctive icon.

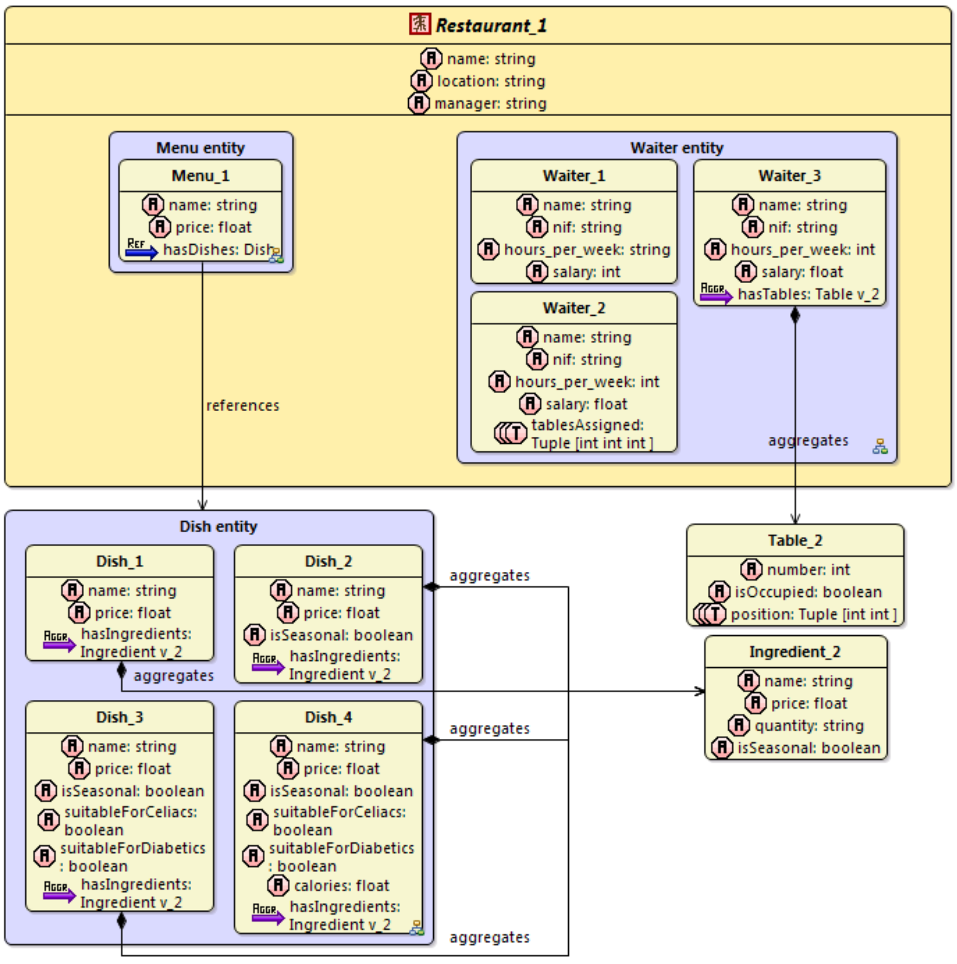


Figure 4: This specific schema viewpoint shows all the elements which compose a *Schema version*. A *Schema version* contains an *Entity Version root*, several *Entities* and *Entity versions*. The same distinctive icons are used for all viewpoints.

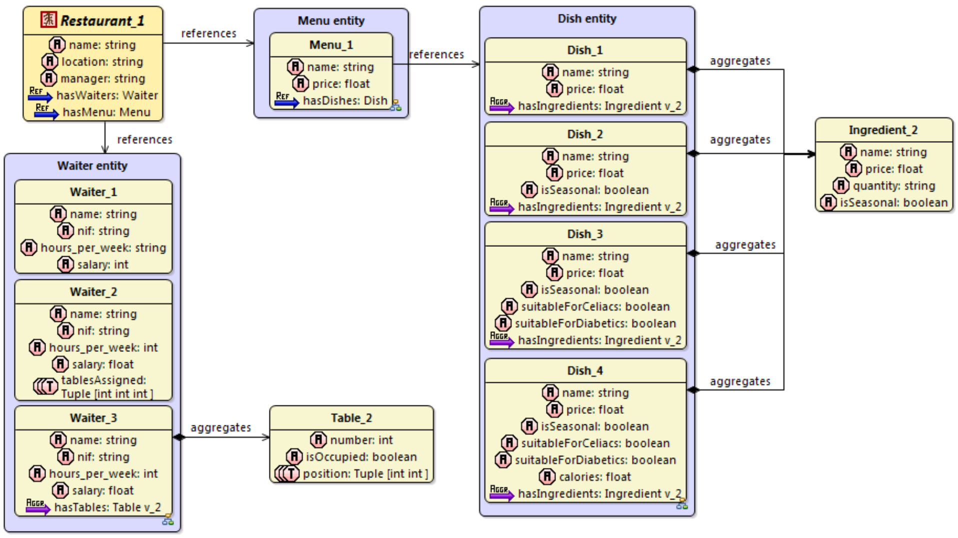


Figure 5: This schema viewpoint is equivalent to the last one but changing the way elements are nested. It organizes the elements with the same layout as the global schema viewpoint.

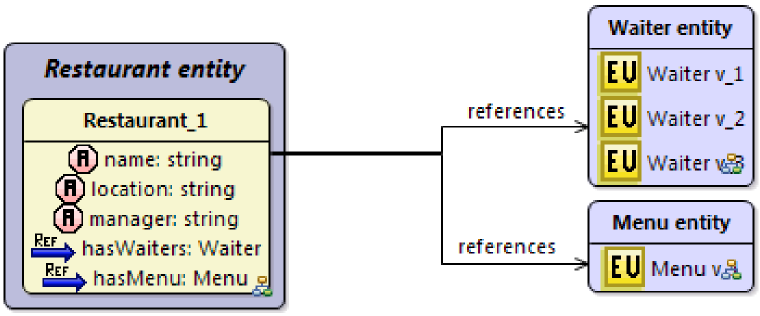


Figure 6: This last viewpoint shows a detailed *Entity* with all its *Entity versions* and all the connections with other elements that these *Entity versions* have.