

Visualization of Inferred Versioned Schemas from NoSQL Databases

Alberto Hernández Chillón

alberto.hernandez1@um.es

Cátedra SAES-UMU
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



November 15, 2016

UNIVERSIDAD DE
MURCIA



About the authors



- Member of the Cátedra SAES team since 2014
- M.Sc. in Computer Science from the UM
- MDE, automatic code generation, NoSQL databases

- Associate professor at the Faculty of Computer Science of the UM
- M.Sc and Ph.D. in Computer Science from the UM
- NoSQL databases, distributed systems, testing



- Professor at the Faculty of Computer Science of the UM since 1984
- Head of the Modelum Group
- MDE, DSL, Software modernization, reverse engineering

Index

- 1 Introduction and context
- 2 Inference process
- 3 Schema visualization
- 4 Conclusions and future work

Index

- 1 Introduction and context
- 2 Inference process
- 3 Schema visualization
- 4 Conclusions and future work

Introduction and context



NoSQL systems

APACHE
HBASE



Cassandra



CouchDB
relax



mongoDB

HYPERTABLE INC



Neo4j

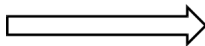
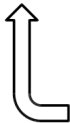


redis

Usefulness of an explicit schema

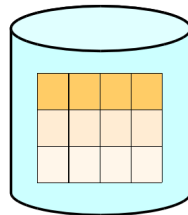
```
CREATE TABLE Persons (  
  PersonID int,  
  LastName varchar(255),  
  FirstName varchar(255),  
  Address varchar(255)  
);
```

Restricted
by the schema



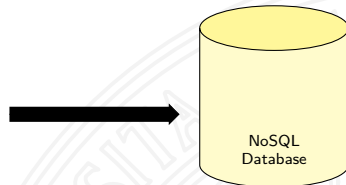
Data inserted/queried
Fit the schema

PersonID	LastName	FirstName	Address



What kind of NoSQL systems? - Schemaless

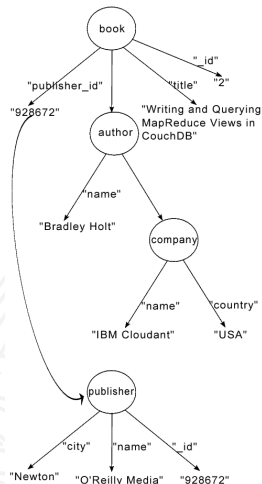
```
{
  "person_id": "123",
  "type": "Person",
  "lastName": "Rush",
  "firstName": "Christopher",
  "address": "C/Gran Via, 13, Madrid"
},
{
  "person_id": "456",
  "type": "Person",
  "lastName": "England",
  "firstName": "Wayne",
  "address": {
    "street": "Av. Pinos, 24",
    "city": "Murcia"
  }
},
{
  "person_id": "789",
  "type": "Person",
  "lastName": "Hoover",
  "firstName": "Quinton",
  "address": "Ronda Norte, 15, Murcia",
  "age": 35
}
```



- Data non-uniformity
- Different versions for the same data

What kind of NoSQL systems? - Aggregations

```
{
  "_id": "2",
  "type": "book",
  "title": "Writing and Querying MapReduce
    Views in CouchDB",
  "publisher_id": "928672",
  "author": {
    "_id": "101",
    "type": "author",
    "name": "Bradley Holt",
    "company": {
      "_id": "324",
      "type": "company",
      "name": "IBM Cloudant",
      "country": "USA"
    }
  },
},
{
  "_id": "928672",
  "type": "publisher",
  "name": "O'Reilly Media",
  "city": "Newton"
}
```



Proposed objective

- A *DataVersity* report (2015) indicated essential functionality required in the near future:
 - **Model visualization**
 - Code generation from schemas
 - Metadata management
- Our goal is to design and implement a tool which will allow us to visualize NoSQL schemas:
 - Taking into account concepts such as versions...
 - ...having in mind the underlying inference process...
 - ...among other things

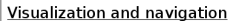
Index

- 1 Introduction and context
- 2 Inference process
- 3 Schema visualization
- 4 Conclusions and future work

NoSQL Schema metamodel



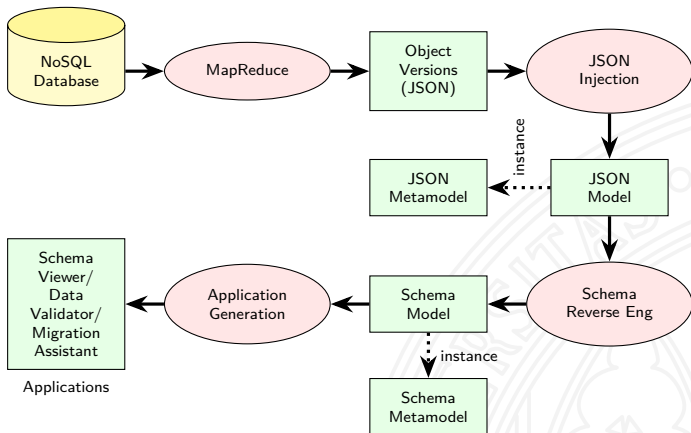
Extended NoSQL Schema metamodel



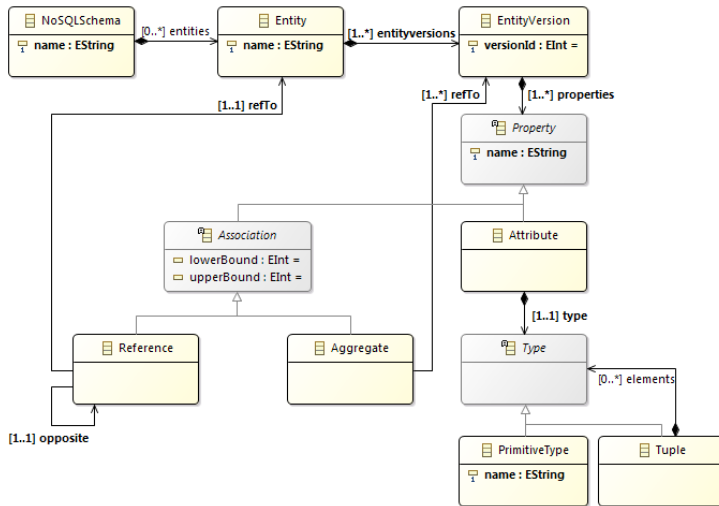
Viewpoint design



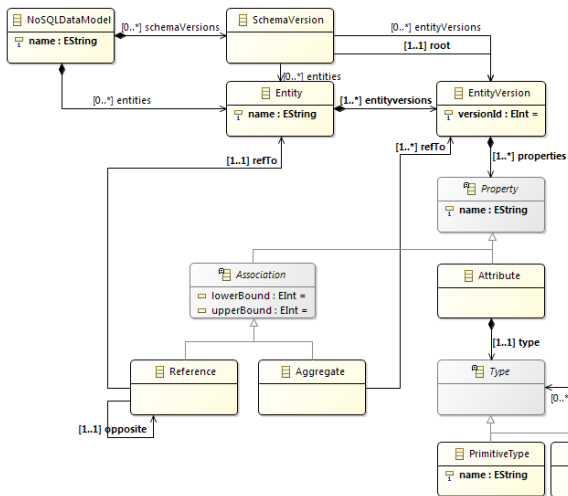
Inference process (I)



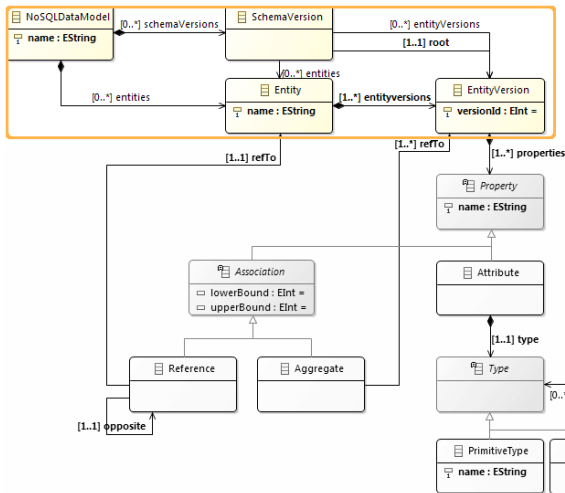
Initial metamodel



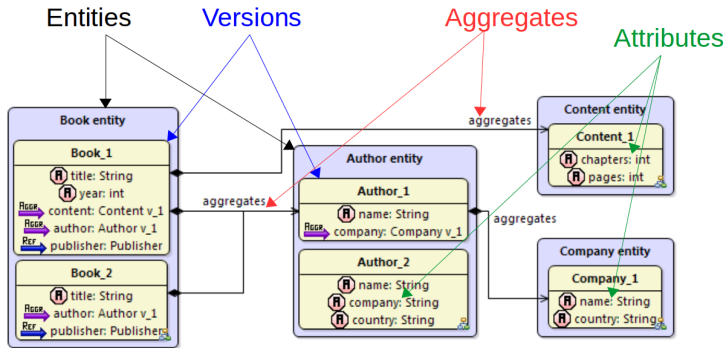
Polished metamodel (I)



Polished metamodel (II)

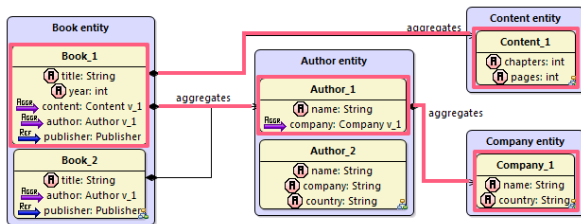


Metamodel elements



Schema versions (I)

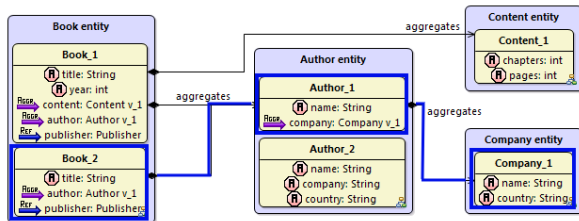
Schema version for Book_1



```
...
{
  "_id": "53",
  "type": "Book",
  "title": "Book_1",
  "hasContent":
  [{
    "_id": "64",
    "type": "Content",
    "chapters": 3,
    "pages": 17
  }],
  "hasAuthors":
  [{
    "_id": "155",
    "type": "Author",
    "name": "Author_155",
    "hasCompany":
    [{
      "_id": "123",
      "type": "Company",
      "name": "Company_123",
      "country": "Country_1"
    }]
  }]
}
```

Schema versions (II)

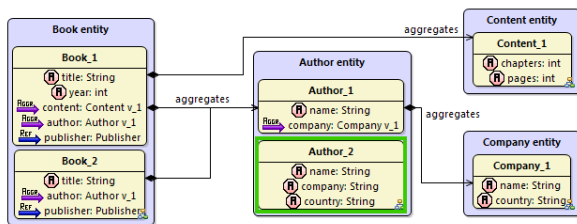
Schema version for Book_2



```
...
{
  "_id": "89",
  "type": "Book",
  "title": "Book_2",
  "hasAuthors":
  [{
    "_id": "77",
    "type": "Author",
    "name": "Author_77",
    "hasCompany":
    [{
      "_id": "61",
      "type": "Company",
      "name": "Company_61",
      "country": "Country_2"
    }]
  }]
}
```

Schema versions (III)

Schema version for Author_2



```
...  
{  
  "_id": "99",  
  "type": "Author",  
  "name": "Author_2",  
  "company": "Company_3",  
  "country": "Country_3"  
}  
...
```

Index

- 1 Introduction and context
- 2 Inference process
- 3 Schema visualization
- 4 Conclusions and future work

Why using Sirius?

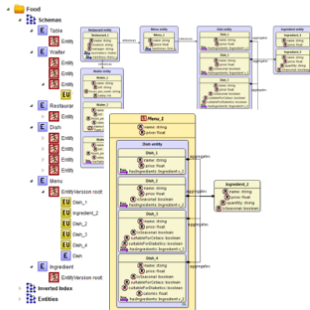
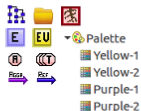
- Views we need:
 - General tree view
 - Global schema view
 - Schema version views
 - Entity detail view
- Ability to create different viewpoints and navigability mechanisms between them
- Automatic generation of an embedded editor in the Eclipse environment
- Extensibility

Design process



3 [Get overview diagram from Sirius schema design description \(NoSQL schema design\)](#)

- NoSQL Schema
 - NoSQL Schema
 - Tree Root
 - [self.name]
 - Schema Folder
 - Index Folder
 - Entity Folder
 - Popup Menu Show overview diagram
 - Overview Diagram
 - Entity Detail
 - NoSQL Schema Diagram (1)
 - NoSQL Schema Diagram (2)
 - Palette

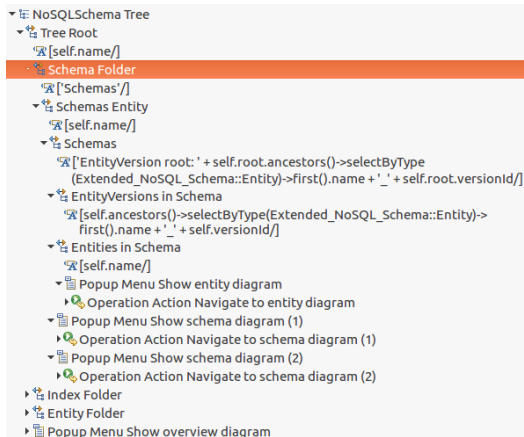
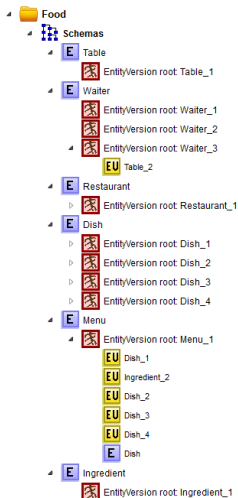


Schema visualization - Input model



Schema visualization - Tree viewpoint (I)

Schema versions grouped by Entity



Schema visualization - Tree viewpoint (II)

Schema versions grouped by Version



NoSQLSchema Tree

Tree Root

`[self.name/]`

Schema Folder

Index Folder

`['Inverted Index']`

Index EntityVersions

`[self.ancestors()->selectByType(Extended_NoSQL_Schema::Entity)->first().name + '_' + self.versionId/]`

Schemas Index

`['Schema with root: ' + self.root.ancestors()->selectByType(Extended_NoSQL_Schema::Entity)->first().name + '_' + self.root.versionId/]`

Popup Menu Show schema diagram (1)

Operation Action Navigate to schema diagram (1)

Popup Menu Show schema diagram (2)

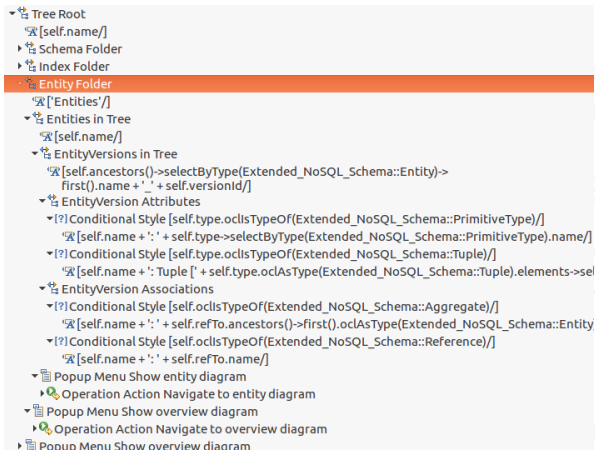
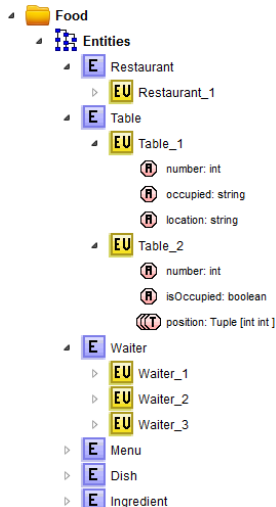
Operation Action Navigate to schema diagram (2)

Entity Folder

Popup Menu Show overview diagram

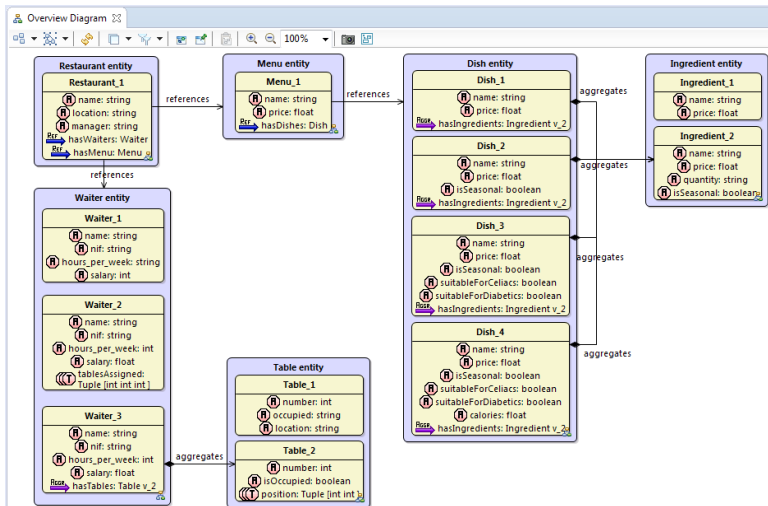
Schema visualization - Tree viewpoint (III)

List of Entities, Versions and Properties



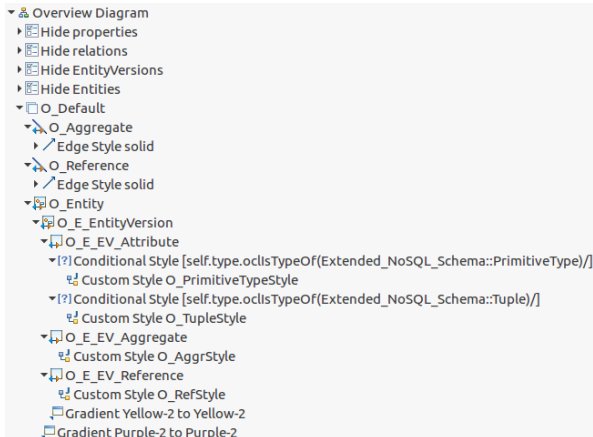
Schema visualization - Global schema (I)

Diagram of Entities, Versions and Properties



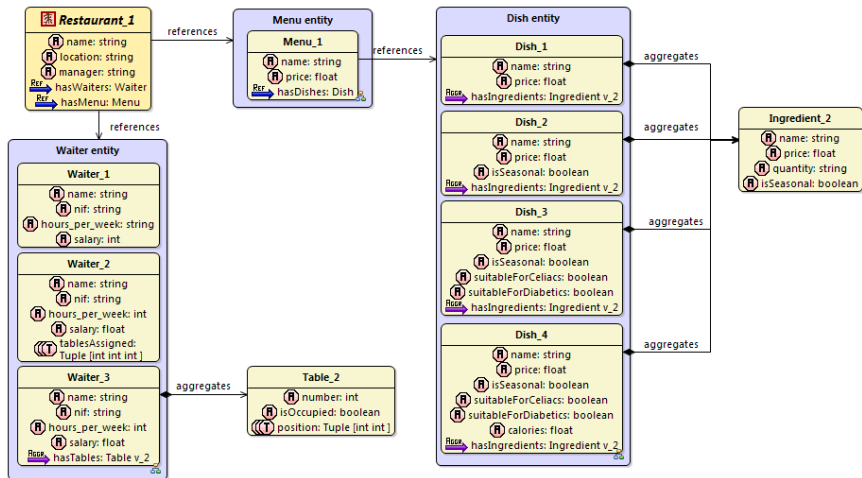
Schema visualization - Global schema (II)

Implementation of the Global schema



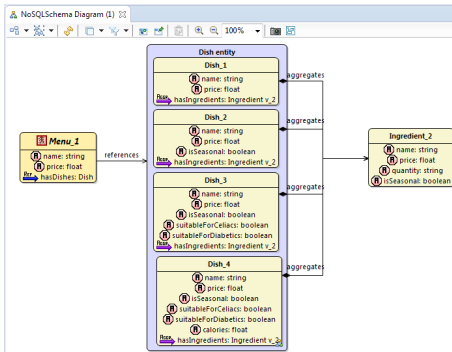
Schema visualization - Schema version (I)

Schema version root and its associations



Schema visualization - Schema version (II)

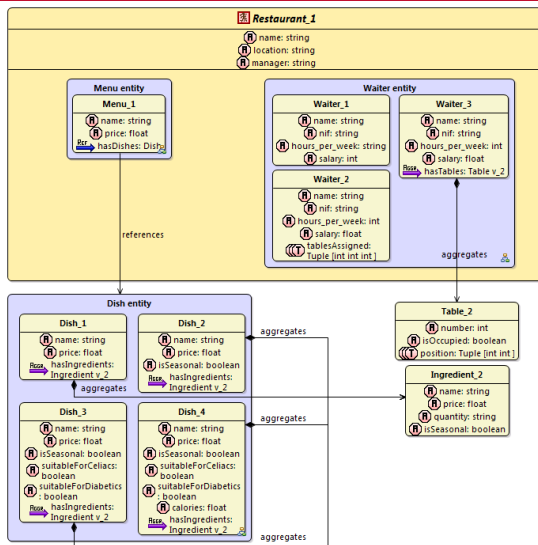
Implementation of the Schema view



- ▼ NoSQLSchema Diagram (1)
 - ▶ Hide properties
 - ▶ Hide relations
 - ▶ Hide EntityVersions
 - ▶ Hide Entities
 - ▶ S1_Default
 - ▶ S1_Aggregate
 - ▶ S1_Reference
 - ▶ S1_EntityVersionRoot
 - ▶ S1_EVR_Attribute
 - ▶ [?] Conditional Style [self.type.oclisTypeOf(Extended_NoSQL_Schema::PrimitiveType)/]
 - ▶ [?] Conditional Style [self.type.oclisTypeOf(Extended_NoSQL_Schema::Tuple)/]
 - ▶ S1_EVR_Aggregate
 - ▶ S1_EVR_Reference
 - ▶ Gradient Yellow-1 to Yellow-1
 - ▶ S1_Entity
 - ▶ S1_E_EntityVersion
 - ▶ S1_E_EV_Attribute
 - ▶ [?] Conditional Style [self.type.oclisTypeOf(Extended_NoSQL_Schema::PrimitiveType)/]
 - ▶ [?] Conditional Style [self.type.oclisTypeOf(Extended_NoSQL_Schema::Tuple)/]
 - ▶ S1_E_EV_Aggregate
 - ▶ S1_E_EV_Reference
 - ▶ Gradient Yellow-2 to Yellow-2
 - ▶ Gradient Purple-2 to Purple-2
 - ▶ S1_EntityVersion
 - ▶ S1_EV_Attribute
 - ▶ [?] Conditional Style [self.type.oclisTypeOf(Extended_NoSQL_Schema::PrimitiveType)/]
 - ▶ [?] Conditional Style [self.type.oclisTypeOf(Extended_NoSQL_Schema::Tuple)/]
 - ▶ S1_EV_Aggregate
 - ▶ S1_EV_Reference
 - ▶ Gradient Yellow-2 to Yellow-2

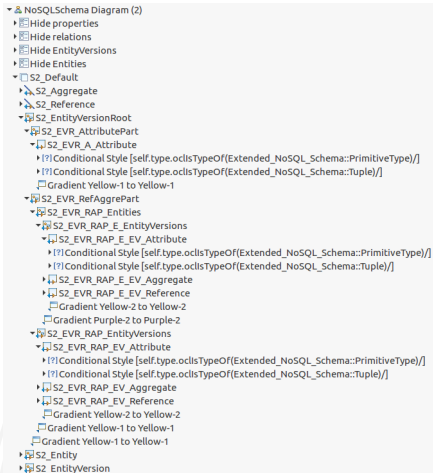
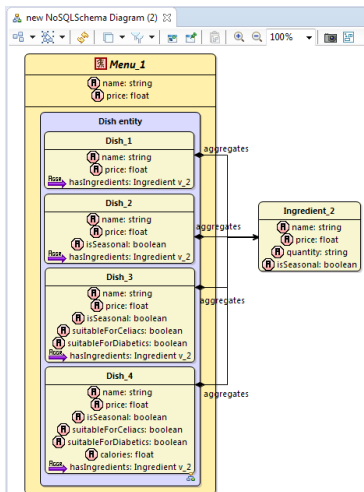
Schema visualization - Schema version (III)

Variation with embedded direct associations



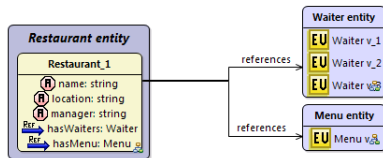
Schema visualization - Schema version (IV)

Implementation of the Schema view



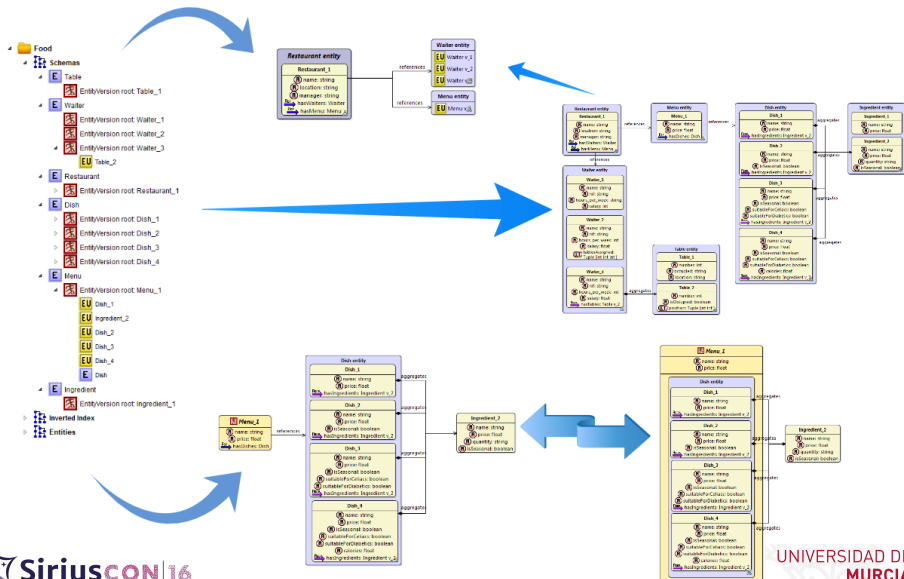
Schema visualization - Entity detail

Entity detail with its Versions associations



- Entity Detail
 - Hide Inner EntityVersions
 - Hide Outer EntityVersions
 - Hide Outer Entities
 - Hide properties
 - Hide relations
 - D_Default
 - D_Aggregate
 - D_Reference
 - D_Entity
 - D_E_EntityVersion
 - D_E_EV_Attribute
 - [?] Conditional Style [self.type.ocliTypeOf(Extended_NoSQL_Schema::PrimitiveType)/]
 - [?] Conditional Style [self.type.ocliTypeOf(Extended_NoSQL_Schema::Tuple)/]
 - D_E_EV_Aggregate
 - D_E_EV_Reference
 - Gradient Yellow-2 to Yellow-2
 - Gradient Purple-1 to Purple-1
 - D_ReferenceEntity
 - D_RE_ReferencedEntityVersion
 - Gradient Purple-2 to Purple-2
 - D_AggregatedEntityVersion
 - D_AEV_Attribute
 - [?] Conditional Style [self.type.ocliTypeOf(Extended_NoSQL_Schema::PrimitiveType)/]
 - [?] Conditional Style [self.type.ocliTypeOf(Extended_NoSQL_Schema::Tuple)/]
 - D_AEV_Aggregate
 - D_AEV_Reference
 - Gradient Yellow-2 to Yellow-2

Navigation between views



Index

- 1 Introduction and context
- 2 Inference process
- 3 Schema visualization
- 4 Conclusions and future work

To sum up (I)

About Sirius...

- ✓ Built in the Eclipse Modeling Framework
- ✓ Easily deployable along with the metamodel into plugins
- ✓ Different viewpoints, layers and customization options
- ✓ Less development time and easier to extend
- ✗ AQL might be quite tricky sometimes when defining actions
- ✗ Learning curve for creating complex examples may be too harsh
- ? Language/model maintenance and evolution
- ? Managing large input models

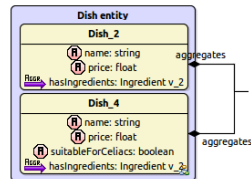
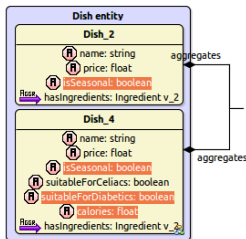
To sum up (II)

About the tool...

- We developed one of the first approaches for NoSQL visualization considering versions
- Tool to visualize NoSQL database schemas and schema versions with Sirius
 - Tree viewpoint and global schema viewpoint
 - Schema version and entity detail viewpoint
- ...but there is still a lot of work to do:
 - Polish the NoSQL_Schema metamodel
 - Improve the Sirius viewpoint definitions
 - Implement the editor functionality

Future work (I) - DB alterations

Removing attributes



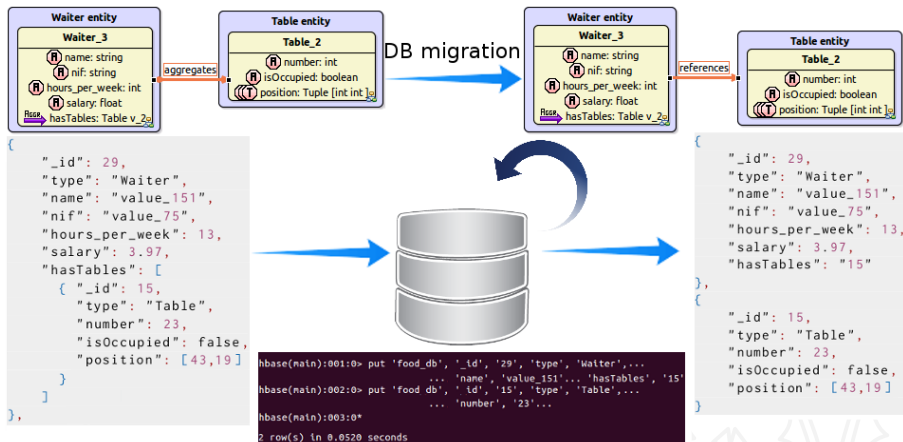
```
{
  "_id": 45,
  "type": "Dish",
  "name": "value_190",
  "price": 0.43,
  "isSeasonal": true,
  "hasIngredients": ...
},
{
  "_id": 50,
  "type": "Dish",
  "name": "value_77",
  "price": 53.84,
  "isSeasonal": false,
  "suitableForCeliacs": false,
  "suitableForDiabetics": true,
  "calories": 7.16,
  "hasIngredients": ...
},
```



Generate transformation scripts for the DB

```
{
  "_id": 45,
  "type": "Dish",
  "name": "value_190",
  "price": 0.43,
  "hasIngredients": ...
},
{
  "_id": 50,
  "type": "Dish",
  "name": "value_77",
  "price": 53.84,
  "suitableForCeliacs": false,
  "hasIngredients": ...
},
```

Future work (II) - DB migration



Generate migration scripts for the DB



alberto.hernandez1@um.es



<https://github.com/Soltari>



<https://linkedin.com/in/albertohc>