University of Liege

2023-2024

INFO9014-1 - Knowledge Representation and Reasoning project

Milestone 3

Wilfried Mvomo Eto - s226625

May 16, 2024

Context

As seen in milestone 1, the database has been normalized, optimizing its structure for performance reasons, albeit at the expense of obscuring its true conceptual meaning. To address the challenge stemming from the physical centralization of this database, an ontology has been developed to provide a more flexible and semantically-centered schema. This initiative aims to facilitate the transformation of our relational database into RDF triples.

However, the mapping process is not straightforward due to the complexities involved. For example, while an OWL Player class is designated as the domain for the OWL `playerTeamCode` data property, the corresponding 'players' database table might have a foreign key reference to another table containing team information. Additionally, properties of type `xsd:string` can pose challenges, as they may correspond to composite data spread across multiple tables in the database. This includes information about the country, city, and stadium, which may be stored in distinct tables. These intricacies contribute to the difficulty in achieving a direct mapping between the ontology and the relational database.

Despite these challenges, our commitment to achieving more precise and flexible semantic modeling remains steadfast. This will be accomplished using R2RML. By leveraging the benefits of data normalization in conjunction with the semantic richness of ontology, we believe we can fully harness the potential of our data while ensuring its long-term integrity and coherence.

1 R2RMLMapping

The entire ontology is mapped in a fragmented way based on the different classes and object properties obtained in our ontology, the different .csv files containing the different data from the tables in our database, the Github project R2RML-f to generate ontology instances and other dependencies. The database is mapped to the ontology following Table 1.

Namespace Prefix: `foot: <http://www.foot-info.org/ontology#>`			
Relational database	**RDF**	**Triples Map**	**Notes**
PLAYERS	foot:PlayerIdentity	PlayerIdentityMap	Player data
MANAGERS	foot:ManagerIdentity	ManagerIdentityMap	Manager data
REFEREES	foot:RefereeIdentity	RefereeIdentityMap	Referee data
TEAMS	foot:Team	TeamMap	Team data
COUNTRIES	foot:OriginCountry	OriginCountryMap	Country data
CITIES	foot:Location	LocationMap	City data
CONTINENTS	foot:QualificationZone	QualificationZoneMap	Continent data
PLAYER-STATISTICS	foot:FairPlayIndexPlayer	FairPlayIndexPlayerMap	Player statistics
TEAM-STATISTICS	foot:FairPlayIndexTeam	FairPlayIndexTeamMap	Team statistics
STADIUM	foot:SportsFacility	SportFacilityMap	Stadium
-	foot:PlayerPenalties	PlayerPenaltiesMap	Player penalties
-	foot:TeamPenalties	TeamPenaltiesMap	Team penalties
-	-	PlayerContinentMap	Player - Qualification zone
-	-	RefereeContinentMap	Referee - Qualification zone
-	-	ManagerContinentMap	Manager - Qualification zone
-	-	TeamCountryMap	Team-Country-Qualification

Table 1: Connection between the Relational Database (RDB) entities, the ontology (RDF), and the Triples maps

We choose to use URI Fragments with content negotiation when publishing data for several reasons, considering the specific advantages they offer in different contexts. URI fragments are particularly well-suited for ontologies and vocabularies due to their stability. By assigning fragments to specific concepts or entities within our ontology, we ensure that the identifiers remain stable over time, even if other parts of the URI change. This stability is crucial for maintaining the integrity and coherence of our semantic model, enabling reliable linking and referencing across datasets and applications.

On the other hand, content negotiation is often more beneficial for managing large datasets. Content negotiation allows us to serve different representations of our data based on user preferences or client capabilities. With large datasets, users may have diverse needs and preferences regarding the format, structure, or level of detail they require. Content negotiation enables us to accommodate these varying needs efficiently, providing users with the most suitable representation of the data while minimizing bandwidth usage and improving overall performance.

By leveraging both URI fragments and content negotiation, we ensure a comprehensive approach to data publishing that addresses the specific requirements and challenges of different types of data. This approach aligns with the principles of Linked Data, promoting the use of standard web technologies and best practices to maximize the accessibility, interoperability, and usability of our data resources. Our mappings, facilitated by the `mapping.ttl` files and their associated output mapping `mapping-output.ttl` where the resulting RDF data will be written, adhere to these principles, contributing to a more interconnected and accessible web of data. The base URI used when creating URIs for the resulting RDF resources: `http://www.example.org/foot-infos/resources/`; and the mapping will be achieved by using the .csv files (players.csv, managers.csv, referees.csv, cities.csv, conutries.csv, continents.csv, stdiums.csv, player_statistics.csv and team_statistics.csv) holding the data of our relational database.

We adhere to a naming convention for the triples maps that follows PascalCase, starting with the ontology name and ending with "TriplesMap" to maintain consistency with the conventions taught in our course. As for the IRIs assigned to each generated instance, we adopt an engineering approach where we concatenate our base IRI with the RDF class name and the RDB primary key, employing kebab-case. This approach allows for content negotiation and ensures clarity and coherence in our data representation.

Mapping our relational database to an ontology presented numerous challenges, chiefly regarding data consistency and integrity. One significant hurdle was the complexity involved in performing joins across various tables within the relational database during the ontology development phase. To address this, we required an enriched vocabulary to ensure the preservation of data information.

2 Mapping strategies

To create a mapping, it's essential to have data instances structured according to the ontology's framework. In our exploration, we identified five scenarios:

(Throughout the examples, we utilize the following prefixes:)

```
@prefix rr: <http://www.w3.org/ns/r2rml#> .
@prefix foot: <http://www.example.org/foot-infos/ont#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
```

- Direct mapping from a table (e.g `ManagerIdentityMap`) where all the information is contained in the table :

```
<#ManagerIdentityMap>
    rr:logicalTable [ rr:tableName "MANAGERS" ];
    rr:subjectMap [
        rr:template "managerIdentity#{ID}" ;
        rr:class foot:Manager;
    ] ;
    rr:predicateObjectMap [
        rr:predicate foot:managerFirstName ;
        rr:objectMap [ rr:column "FIRST_NAME"; rr:datatype xsd:string ] ;
    ] ;
    rr:predicateObjectMap [
        rr:predicate foot:managerLastName ;
        rr:objectMap [ rr:column "LAST_NAME"; rr:datatype xsd:string ] ;
    ] ...
```

that yields for instance :

```
<http://www.example.org/foot-infos/resources/managerIdentity#4>
    a                           foot:Manager ;
    foot:managerComesFrom       <http://www.example.org/foot-infos/resources/
        country#4> ;
    foot:managerDateOfBirth     "1955-07-21"^^xsd:date ;
    foot:managerFirstName       "Marcelo" ;
    foot:managerGenre           true ;
    foot:managerIsBorn          <http://www.example.org/foot-infos/resources/team#
        4> ;
    foot:managerLastName        "Bielsa" ;
    foot:manages                <http://www.example.org/foot-infos/resources/city#
        4> ...
```

- Example of an R2RML mapping snippet of `PlayerPenaltiesMap`:

```
<#PlayerPenaltiesMap>
    rr:logicalTable [ rr:tableName "PLAYER_STATISTICS" ];
    rr:subjectMap [
        rr:template "playerPanalties#{ID}" ;
        rr:class foot:PlayerPenalties;
    ] ;
    rr:predicateObjectMap [
        rr:predicate foot:playerPenaltyDate, rdfs:label ;
        rr:objectMap [ rr:column "STATISTICS_DATE" ; rr:datatype xsd:string ] ;
    ];
    rr:predicateObjectMap [
        rr:predicate foot:playerYellowCardNumber, rdfs:label ;
        rr:objectMap [ rr:column "YELLOW_CARDS" ; rr:datatype xsd:integer] ;
    ]; ...
```

that yields for instance:

```
<http://www.example.org/foot-infos/resources/playerIdentity#58>
    a                           foot:Player ;
    rdfs:label                  false , "Kim" , "Sophia" ;
    foot:playerComesFrom        <http://www.example.org/foot-infos/resources/country
        #58> ;
    foot:playerDateOfBirth      "1996-04-30"^^xsd:date ;
    foot:playerFirstName        "Sophia" ;
    foot:playerGenre            false ;
    foot:playerIsBornIn         <http://www.example.org/foot-infos/resources/city#
        58> ;
    foot:playerLastName         "Kim" ;
    foot:playsFor               <http://www.example.org/foot-infos/resources/team#
        58> ...
```

- Another R2RML mapping achieved, relies on the presence of the same foreign keys value. For instance the table `Players` and `Stadium` share the same teams foreign key and this enables to set up a link between the both via a relation `playsAtHome` according to the common valzue of `team_id`.

```

<#PlayerIdentityMap>
    rr:logicalTable [ rr:tableName "PLAYERS" ];
    rr:subjectMap [
        rr:template "playerIdentity#{ID}" ;
        rr:class foot:Player;
    ] ;
    rr:predicateObjectMap [
        rr:predicate foot:playerFirstName, rdfs:label ;
        rr:objectMap [ rr:column "FIRST_NAME" ; rr:datatype xsd:string] ;
    ] ;...
   rr:predicateObjectMap [
        rr:predicate foot:playsAtHome;
        rr:objectMap [
            rr:parentTriplesMap <#SportFacilityMap>;
            rr:joinCondition [
                rr:child "TEAM_ID"; rr:parent "TEAM_ID";
            ];
        ];
    ] .
```

That yields for instance:

```
    a                         foot:PlayerIdentity , foot:Player ;
        rdfs:label                "Jr." , true , "Neymar" ;
        foot:playerAttendsIn      <http://www.example.org/foot-infos/resources/
            qualificationZone#3> ;
        foot:playerComesFrom      <http://www.example.org/foot-infos/resources/
            country#45> ;
        foot:playerDateOfBirth    "1992-02-05"^^xsd:date ;
        foot:playerFirstName      "Neymar" ;
        foot:playerGenre          true ;
        foot:playerIsBornIn       <http://www.example.org/foot-infos/resources/
            city#45> ;
        foot:playerLastName       "Jr." ;
        foot:playsAtHome          <http://www.example.org/foot-infos/resources/
            stadium#45> ;
        foot:playsFor             <http://www.example.org/foot-infos/resources/
            team#45> .
...
```

- Mapping from joined tables (e.g TeamCoountryMap) where the information is contained across several tables.

```
<#TeamCountryView>
    rr:logicalTable [
        rr:sqlQuery """
SELECT␣teams.id,␣teams.name␣AS␣tname,␣countries.name␣AS␣country_name,␣continents
    .name␣AS␣continent_name
FROM␣teams
LEFT␣OUTER␣JOIN␣cities␣ON␣teams.city_id␣=␣cities.id
LEFT␣OUTER␣JOIN␣countries␣ON␣cities.country_id␣=␣countries.id
LEFT␣OUTER␣JOIN␣continents␣ON␣countries.continent_id␣=␣continents.id
␣␣␣␣␣␣␣␣""";
    ];
    rr:subjectMap [
        rr:template "teams#{ID}" ;
        rr:class foot:Team;
    ] ;
    rr:predicateObjectMap [
        rr:predicate foot:teamName, rdfs:label ;
        rr:objectMap [ rr:column "TNAME"; rr:datatype xsd:string ] ;
    ] ;
  ....
```

That yields for instance :

```
<http://www.example.org/foot-infos/resources/country#38>
    a                            foot:Country ;
    rdfs:label                   "Sweden" ;
    foot:countryConsistsOf      <http://www.example.org/foot-infos/resources/city
        #38> ;
    foot:countryName            "Sweden" ;
    foot:isIncludedIn           <http://www.example.org/foot-infos/resources/
        qualificationZone#1> ;
    foot:playerNativeCountry    <http://www.example.org/foot-infos/resources/
        playerIdentity#67> ;
    foot:refereeNativeCountry   <http://www.example.org/foot-infos/resources/
        refereeIdentity#58> .
...
```

Indeed, the query provides detailed information on the teams, including their names, as well as the names of the associated countries and continents, even though some of these relationships may not exist (thanks to **LEFT OUTER JOIN**). This type of mapping is very useful for integrating relational data into the Semantic Web, enabling this data to be represented and linked in a standardised and interoperable format.