Dipartimento di Ingegneria e Scienza dellInformazione

– KnowDive Group –

Data integration in the Music Domain

Artists, Reviews and Concerts

Document Data: Reference Persons:

© 2018 University of Trento
Trento, Italy

KnowDive (internal) reports are for internal only use within the KnowDive Group. They describe preliminary or instrumental work which should not be disclosed outside the group. KnowDive reports cannot be mentioned or cited by documents which are not KnowDive reports. KnowDive reports are the result of the collaborative work of members of the KnowDive group. The people whose names are in this page cannot be taken to be the authors of this report, but only the people who can better provide detailed information about its contents. Official, citable material produced by the KnowDive group may take any of the official Academic forms, for instance: Master and PhD theses, DISI technical reports, papers in conferences and journals, or books.

Index:

1 Input Dataset description

The topic we choose for our Knowledge and Data Integration course assignment is Data Integration. Data integration brings together data from different systems and makes it more valuable to users. Our aim through this assignment is to provide a facility for music lovers. Anyone who likes music will have a list of favorite music artists. There are several systems that provide the best artist based on the reviews or provide the artist based on the genre. Our motivation is to provide the list of artist in a certain genre who is assigned with a score. So the user can choose another artist in the same genre based on the reviews.

We started to collect the data related to music events from all over the world, with the objective of providing the user with diversified results. We collected data from California [4], [6], Spain [7], and Lazio (Italy) [1]. Each dataset was presented in their local language. With our group member's support we managed to understand the metadata of the dataset. But it was not enough to satisfy the objective of our project. The data we had mostly focused on charity events or ticketed events organized by one person or a group. So we decided to define the possible queries we want to resolve from our integration model. Based on this stage, we tried to find the artist dataset aligned with the event dataset we found before. We decided to use the tool Rapidminer introduced as part of this course. We discussed our plan with Alessio, during our progress meeting. As per his suggestion, the best solution is to use website crawling for our situation rather than Rapidminer. Hence we were on to find the possible websites that can provide the possible data. In parallel, we were also focused on finding a matching ontology that fits our domain. Most of the available ontologies were focused on music types and event types. Hence we decided to create our own ontology and fixed with modification according to the comments from Subhashis. After we finalized the ontology, we decided to follow a new strategy to find a new dataset that focuses on the artist details than the event.

Hence we find the dataset 10K MTV's top music artists from a open github repository [5]. This included artist name, genre of music, personal website, Facebook, MTV, and Twitter links. Then we managed find the dataset of reviews of artists. This dataset has title of the album, artist name, review date, review type, score which is in between 1 to 10, best tracks and YouTube link.

2 ER model design and ontology description

At the very beginning of our project, when we made our proposition to work on reviews in the framework of artists' concerts, we were told that an upper ontology of events had already been defined by some members of the KnowDive group, and that we were going to start building our integration project upon this given ontology. When it came out this ontology was not connected to the scope of our research, and that we had to build our own ontology to address the integration task, we started looking on the web for templates to build our domain-specific ontology in Protégé. Unfortunately, the ontologies that we have found were neither good nor compatible with our task, which brought us to the conclusion that we needed to build an ontology that could better target our purpose and data. At this point a great thank should be addressed to Subhashis who helped us in the definition of the ER model and the ontology, and guided us towards the sources we used to build it.

2.1 First steps

The first step in building our ontology has been looking for similar ontologies that could map out data. We started considering one well-known ontology to map events, LODE [2], and one of its extensions called LODSE [3]. Starting from a given ontology gave us a glance on how our final structure should look like, what attributes should be included and which should not. Before discussing on whether these ontologies were good or not for our project, we moved to the step of defining our ER model.

2.2 ER model definition

The first step was to define the core entities; we pointed out three core concepts: Artist, Concert and Review. The other important entity is the Location. We soon realized that both the LODE and LODSE ontology were not mapping some of the attributes we needed, while adding some information that either were not relevant to us, or simply were not present in our datasets. After building the ER model on paper, checking with Subhashis its correctness and completeness, we moved to the next phase of creating the ontology in Protégé. A quick statement should be introduced here, saying that the process has been iterative: the first ER model was not good enough, some of the properties were missing and some classes that at first were supposed to be needed (e.g. MusicalGroup) could be removed.

2.3 Building the ontology

The first version of the ontology was soon discarded to give place to the new one, following the advice Subhashis gave us. There is no MusicalGroup and there is no more distinction between an Artist as a Performer (an entertainer who performs a dramatic or musical work for an audience) and the Artist as a Creative Person (a person whose creative work shows sensitivity and imagination), but we only kept the second definition in order to have just one representation. The Artist is now whomever can be put as the representative of a Group (namely we just need the Artist to define the Group, that is we can link the Reviews directly to the Artist (to avoid attribution problems), in the broad sense that he/she can conceptually also be the person who performs at a given concert.

The entities been defined, we then moved to the step of mapping relationships and attributes. This step has also been made iteratively, both looking at the datasets and when importing the data in Karma. The datasets metadata provided by phase one have been used to identify the attributes we needed to map our events.

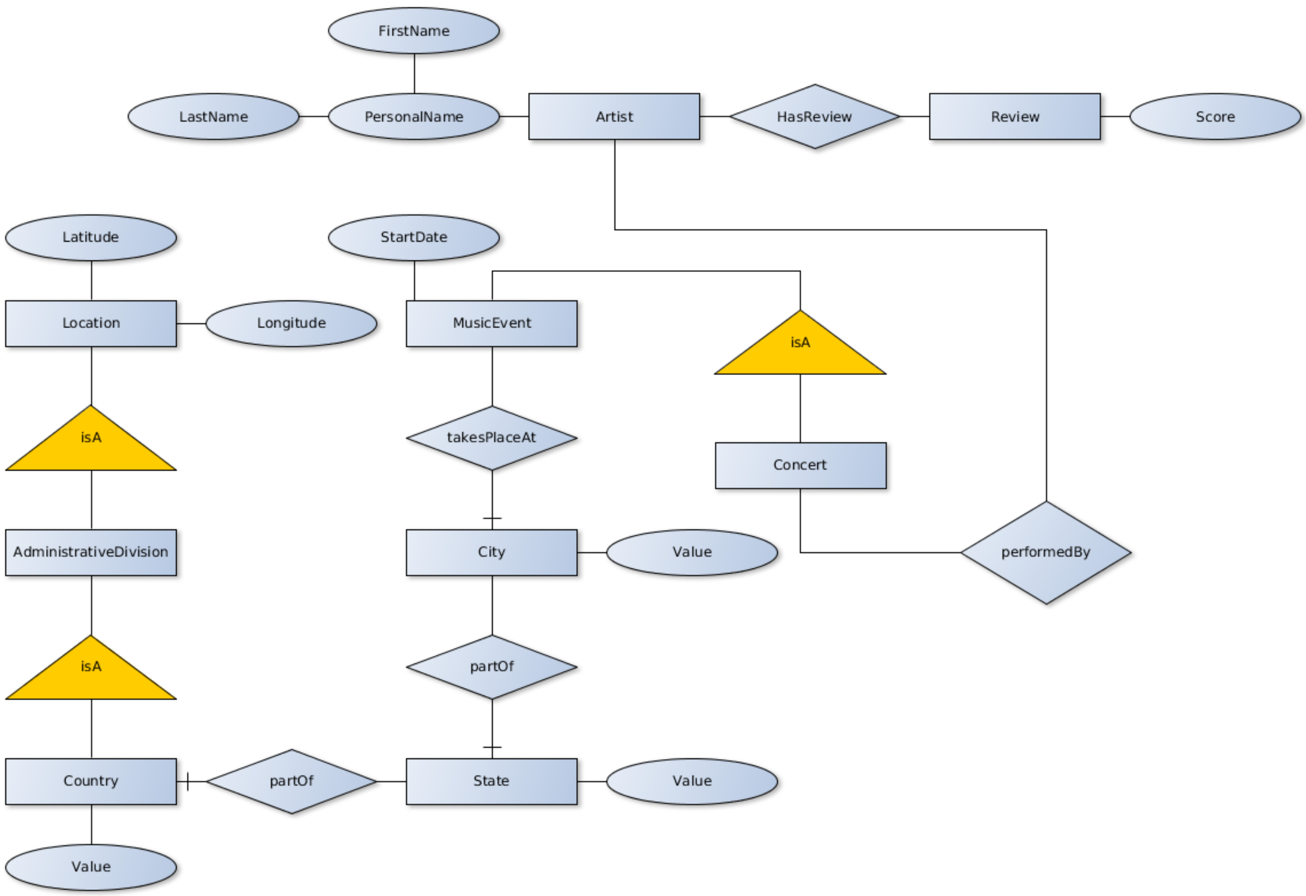
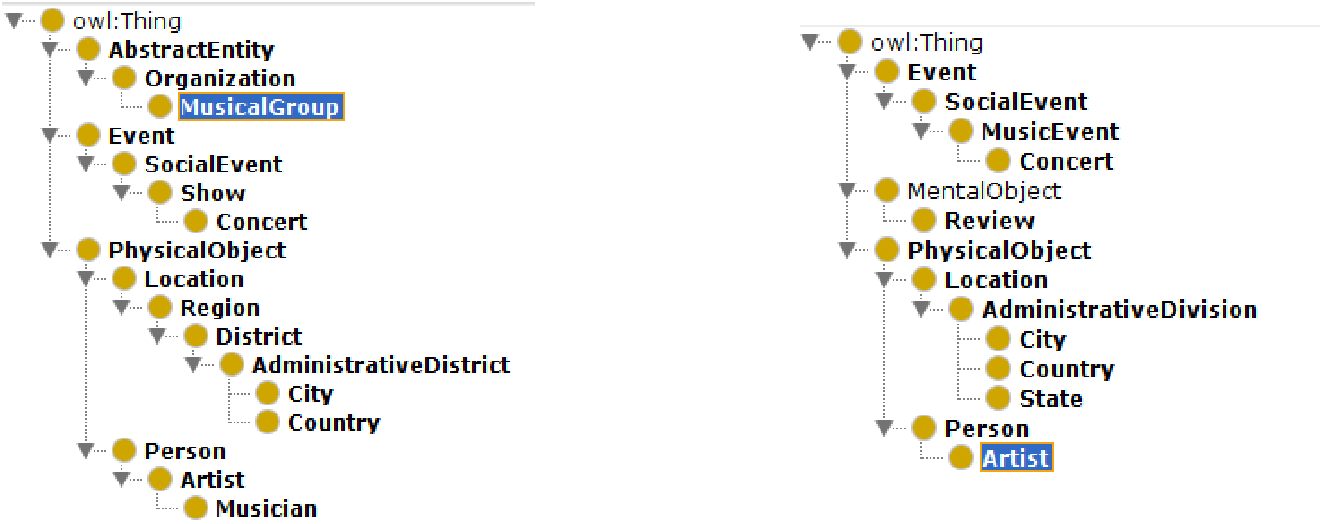


Figure 1: The final ER model



(a) The first ontology

(b) The refined ontology

Figure 2: The two versions of the ontology

2.4 Labels, definitions and WordNet

While we were taking the LODE ontology as an example to create the entities structure of our ontology, another important element was to unambiguously define what those entities refer to (the definition), their names (synonyms) and their properties (range, type, ..). We thoroughly exploited WordNet [8] as the means to select what the best terms to use were and their IsA hierarchy; the structure that our ontology reflects is the one of WordNet hypernyms/hyponyms.

2.5 Missing upper-level ontologies

The ontology we have defined is not meant to thoroughly cover the whole scenario of music concerts and artists, conversely it has been built as a domain-specific ontology that can be used for our purpose. The reasons why we have not used any (or few) elsewhere defined upper/core ontologies are two: first, the formal modelling of the problem is not in the scope of our project; second, for the scope of our integration task there was no need to link ontologies whose level was higher than our domain-specific description, since all the datasets we have can be directly mapped with no ambiguity to our entities. This is also why we decided to discard all properties of the data that were not strictly related to our final queries (links, social accounts, concert tickets availability, etc.). The importance of top and core level ontologies is that they improve the reusability of the whole ontology, which should be the first task we would do whenever reusing this project as a starting point for further refinements.

3 Integration process description

While half of our group was still working on sharpening the final ontology, and looking for other data that could be integrated in the system, the other half started playing with Karma to address the integration task. One of the first issues risen was the difficulty in using the tool itself: every time you restart it you are required to import the data again and then apply the ontology and the saved .ttl models. To fasten this process all the datasets have been imported in a MySQL database using MySQL workbench. Each dataset has been turned into a database with its tables, and views were exposed when needed: indeed it is easier to make joins directly in the database and then importing the data in Karma by simply using the import database table tool.

3.1 Data filtering

Some datasets had to be filtered, since they contained information that were not linked to the scope of our project. For instance, the dataset of events of the Pais Vasco contained more than one type of event. We simply filtered the records (rows) that were not relevant to the integration (e.g. theatre events) before importing the data in Karma using tools like LibreOffice.

3.2 Integration in Karma

We started working with two main test datasets, importing the MusicOntology we had built with Protégé, and starting linking the columns to the entities. It took us little time to see that some of the attributes where missing (e.g. Review.score), so we iteratively modified the ontology by adding these properties. A good advantage was provided by Karma: when modifying the ontology, if you add some attributes, the old models you had saved before can still be applied; in other words, after

import_data -> link_data_to_ontology -> export_.ttl_model

if you modify the ontology and then do

import_data -> apply .ttl_model,

the model is still applied successfully.

We then added entities and linked all the columns to the corresponding classes using the ontology attributes. We created URIs for the most relevant entities, linked the classes using the ontology properties. Some columns have been filtered and normalized to fit the attributes definitions.

URIs creation We began to add URIs to our entities using pytransform and column aggregations when needed. A good example is the artist URI, whose code can be applied to all datasets, and turns an artist name into a lowercase, underscore-separated domain/artist_name.

```
artist_name = getValue("name")
lowercase_name = artist_name.lower() # Normalize all names to lowercase
tokens = lowercase_name.split()
uri = 'http://music_event_domain.org/' # Add our domain to the uri
for token in tokens[:-1]:
    uri +=  token + '_' # Words are underscore-separated
uri += tokens[-1]
return uri
```

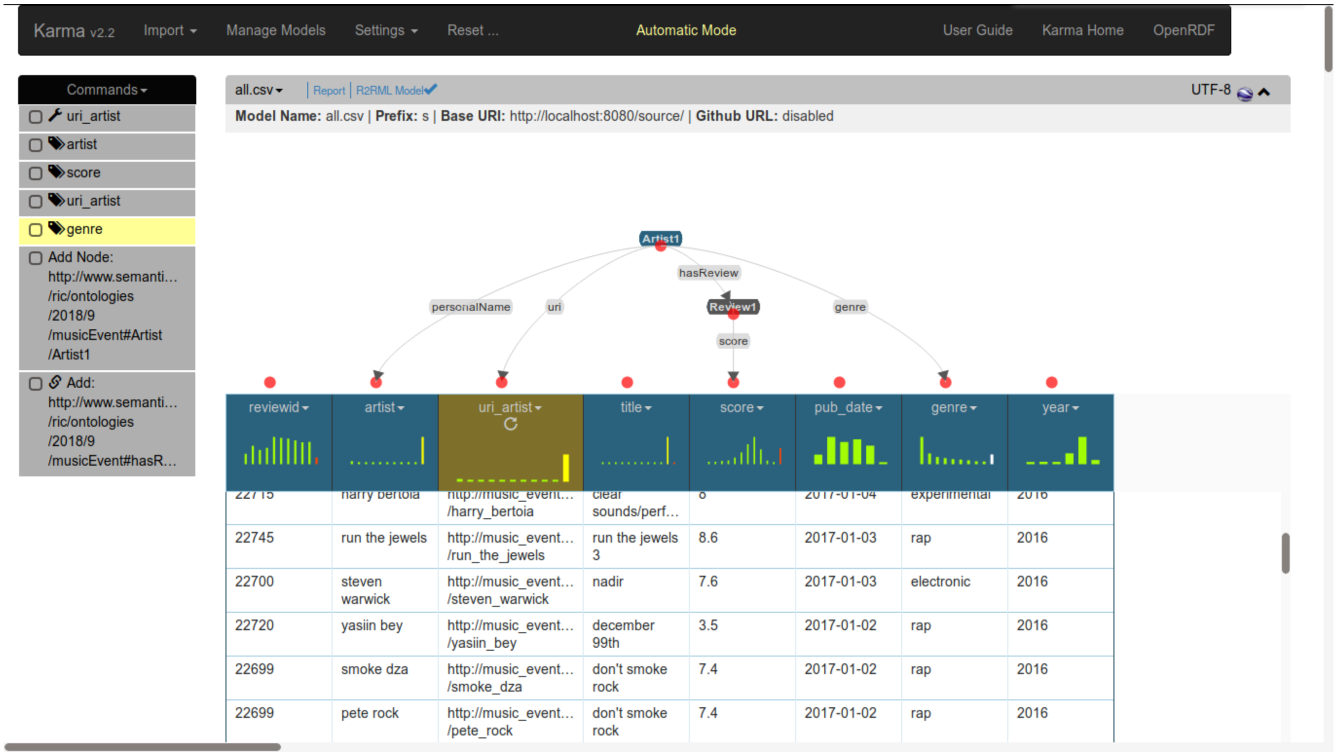


Figure 3: One of the datasets linkeds to the ontology

Columns normalization Some of the columns did not directly fit into the ontology, and had to be normalized. The most relevant example can be found in the MusicArtistAndLocation dataset, where we had to work on the *location* attribute to make it fit into the properties of the ontology **Location** entity.

```
import re
location = getValue("location")
if location != 'Not Defined':
    entities = re.split('-|,',location) # Isolate entities (city, state, ..)
    if len(entities) > 0:
         city = entities[0]
         return city
 return location
```

After we had acquired good confidence that our ontology was the right one for our purpose, the two people that were still working on it moved to defining the queries, whereas the other two who were working on Karma kept managing the integration process.

Karma integration results At the end of this process we have imported five datasets, with data and labels in different languages and different structures; one of the most relevant legacy this process has left us is the understanding of the importance and the need of using ontologies to address integration problems. Whenever you have a clear view of what the world is (even if it comes from experience and is built iteratively, as it is the case with this project), you become a step closer to master the problem you are tackling; eventually, whatever the data you acquire, you will most probably be able to map the relevant elements that make your project meaningful.

(a) Dataset in Spanish

(b) Dataset in English

Figure 4: Integration with labels in different languages

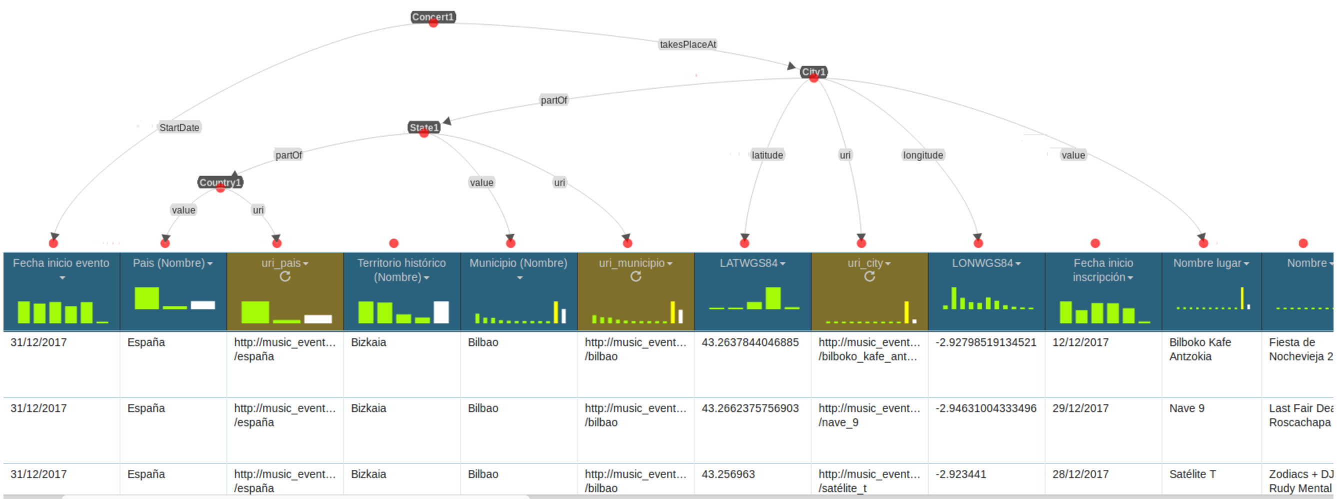


Figure 5: Integration of rich datasets

Risen issues An issue emerged when importing the data from the MySQL database; out of the thousands of records contained in the tables, only a little number of entries (some hundreds) were imported in Karma. To overcome this problem we simply exported the aggregated information gathered from the database to a .csv file and then import it again in Karma. In the project folder we stored both the exported .csv and the SQL schema of the database we used to create the aggregation; the latter, which is simply a join over different tables, is stored as the view *all* in the database.

4 Output datasets and query description

While completing the integration task, we started writing on paper the queries that were going to be answered by our project. Some of the queries had already been defined before the integration project started, whereas others have been pointed out by the data itself.

4.1 Queries description

The main objective of our integration task was to provide a model that could be inquired using three major entities: artists, reviews and concerts. Key elements for the artist are the genres of music he/she plays, the reviews with their score, and the places where he/she performed a concert.

RDF is a data model of graphs of subject, predicate, object triples. These elements are represented using URIs, which can be abbreviated as prefixed names. Objects can also be literals: strings, integers, booleans, etc.

SPARQL is the query language of the Semantic Web. It lets us: pull values from structured and semi-structured data, explore data by querying unknown relationships, perform complex joins of disparate databases in a single, simple query and transform RDF data from one vocabulary to another. Joins are seamlessly provided after the integration; defining URIs is of major importance, since the integration is done using these data as keys. URIs creation should be defined in a way that allows normalization over different sets of data, meaning that if we have an artist *(Joe Low)* in a dataset $D1$ and the artist *(joe, low)* in another dataset $D2$, the two resulting URIs should be identical (e.g. uri = joe_low).

A SPARQL query comprises, in order:

```
Prefix declarations:    for abbreviating URIs
Dataset definition:     stating what RDF graph(s) are being queried
A result clause:        identifying what information to return from the query
The query pattern:      specifying what to query for in the underlying dataset
Query modifiers, slicing, ordering, and otherwise rearranging query results
```

which in SPARQL slang becomes

```
PREFIX ...  # prefix declarations
FROM ...    # dataset definition
SELECT ...  # result clause
WHERE {     # query pattern
    ...
}
ORDER BY    # query modifiers
```

4.2 CQ defined before integration

Above we described what the key elements of our resulting database should be; now we explain some of the queries we had defined:

```
get artists that belong to a given genre
get artists grouped by their genre
get artists based on their concerts location
get artists based on their review score
get artists sorted by their reviews scores
get artists that have performed at a given location
get concerts based on their genre
get concerts based on the artists genre
...
```

Here below we show some of the most important implementations that prove the integration process has worked successfully:

```
# Select concerts performed by a given artist
    PREFIX ontology:<http://www.semanticweb.org/ric/ontologies/2018/9/musicEvent#>
    SELECT ?concert
    WHERE {
        ?artist     ontology:personalName   'target_artist' .
        ?concert    ontology:performedBy    ?artist .
    }
```

The importance of this query is that we have the information about concerts in a dataset that is different to the one that contains the information about reviews. This query can be asked only to the integrated database, not to its base datasets.

```
# Select review of artists that performed in a given city
    PREFIX ontology:<http://www.semanticweb.org/ric/ontologies/2018/9/musicEvent#>
    SELECT ?review
    WHERE {
        ?artist     ontology:hasReview      ?review.
        ?concert    ontology:performedBy    ?artist.
        ?concert    ontology:takesPlaceAt   ?city.
        ?city       rdf:type                ontology:City
        ?city       ontology:value          'target_city'.
    }
```

This query is relevant because it shows how the information about the concert city is directly accessible after the integration task, whereas in the pre-integration datasets it was embedded and shadowed in other columns.

```
# Select concerts at a given location, performed by artists that belong to a given genre and have reviews w
    PREFIX ontology:<http://www.semanticweb.org/ric/ontologies/2018/9/musicEvent#>
    SELECT ?concert
    WHERE {
        ?review     ontology:score          'target_score'
        ?artist     ontology:genre          'target_genre'.
```

```
        ?concert    ontology:performedBy    ?artist .
        concert     ontology:takesPlaceAt   ?city.
        ?city       rdf:type                ontology:City
        ?city       ontology:value          'target_city'.
    }
```

This query represents the ultimate set of composite questions our database can answer. Each of the element of the query (city, review, genre) comes from a different input dataset.

5 Input/output datasets comparative analysis description

We start the comparison by listing the number of classes (entities), object properties (relationships) and data properties (attributes) of our ontology and of the datasets we have used. We than move to compute coverage and flexibility of our model. An important note should be added here: many of the datasets we have used contain albums information, which is the main reason why the coverage metric is low; one of the first natural improvements of this project would be to introduce the class Album into the ontology, and map the missing attributes to this class. Nevertheless, not being the Album information in the scope of this project, we decided to omit it from the current implementation.

Competency queries

- Number of entities: 4 (artist, concert, location, review)
- Number of relationships: 3 (performedBy, hasReview, takesPlaceAt)
- Number of attributes: 3 (Review.score, Artist.personalName, city)

MusicEvent ontology

- Number of entities: 14
- Number of relationships: 13
- Number of attributes: 10

Datasets

- Number of entities: 25
- Number of relationships: hard to define, definitely more than 20
- Number of attributes: 82 (60 excluding duplicates)

Since there was no formal ontology describing the input datasets, to define the entities and attributes we started from our ontology and then proceeded in two ways: for the classes we simply identified classes that are obviously present in the datasets (e.g. Album, Song); for the attributes, we took the sum of distinct columns in our dataset. Datasets classes that are missing from our ontology are: album, video, song, release and a huge number of other types of events like theatre, festival, presentation, which are present in the Pais Vasco Dataset. Some of the most interesting missing attributes are the ones linked to the song description and to the album tracks composition and tracks releases.

For the CQ/model:

$$Cov(c) = 1 - \frac{|Cc - Mc|}{|Cc|} = 2$$

$$Flex(c) = \frac{|Mc - Cc|}{|Mc|} = 0.7$$

$$Cov(p) = 1 - \frac{|Cp - Mp|}{|Cp|} = 2$$

$$Flex(p) = \frac{|Mp - Cp|}{|Mp|} = 0.8$$

For the databases/model:

$$Cov(dc) = 1 - \frac{|Dc - Mc|}{|Dc|} = 0.6$$

$$Flex(dc) = \frac{|Mc - Dc|}{|Mc|} = 0.8$$

$$Cov(dp) = 1 - \frac{|Dp - Mp|}{|Dp|} = 0.7$$

$$Flex(dp) = \frac{|Mp - Dp|}{|Mp|} = 0.5$$

For the CQ/databases:

$$Cov(dc) = 1 - \frac{|Cc - Dc|}{|Cc|} = -4.25$$

$$Flex(dc) = \frac{|Dc - Cc|}{|Dc|} = 0.84$$

$$Cov(dp) = 1 - \frac{|Cp - Dp|}{|Cp|} = -18$$

$$Flex(dp) = \frac{|Dp - Cp|}{|Dp|} = 0.95$$

5.1 Datasets metadata

```
..1. Santa Monica
Region
Sub region
Address
Zip code
Age group: adults, kids, teens,etc
Duration of event
Type of event : venue,library, etc (general)
Contact: email, person, phone
Instructions
URL : description
End date
Event ID
Event type: community, life long learning, workshops,etc
Venue ID
Image URL: decription
Cancel flag
Name of the venue
Sign up URL: description
Start date
```

Title of event

2. Mixed CA

Region1
Region 2
Sub region1
Sub region 2
Concert date
Location
Contact name
Contact number
End time
Location address coordinate
Location address
Location city
Location state
Zip code
Artist
Organization presenting
Start time
Supervisor district
Year

3. Spain

Name
Type of event
Start date
End date
Country
Department
Department chief town
Registration start date
Registration end date
Is online event
Opening
Price
Event language
GPS coordinates (LAT, LONG)
Latitude
Longitude
Event place
Department chief town
Department code

```
Region
Region code
Country
Country code
Human readable URL
Physic URL
Data XML
XML metadata
Zip file

..4. Lazio (contributions, donations)
Ente                    Organizer
Soggetto beneficiario   Beneficiaries
Progetto                Project
Contributo              Donation

..5. Lazio (groups information)
Denominazione banda     Group name
Anno di fondazione      Foundation year
Indirizzo               Address
E-mail                  E-mail address
Sito internet           WebSite
Social Network          Link/ type
Presidente              President
Direttore               Director
n. componenti banda     n. of group members
Composizione banda      Group composition
Repertorio              Repertoire
```

5.2 Model evaluation

	Class Coverage	Class flexibility	Attribute Coverage	Attribute Flexibility
CQ-Model	$\simeq$ 0.8 (Ideal) [2]	$\simeq$ 0.2 (Ideal) [0.7]	$\simeq$ 0.8 (Ideal) [2]	$\simeq$ 0.5 (Ideal) [0.8]
DS-Model	$\simeq$ 0.8 (Ideal) [0.6]	$\simeq$ 0.2 (Ideal) [0.8]	$\simeq$ 0.8 (Ideal) [0.7]	$\simeq$ 0.5 (Ideal) [0.5]
CQ-DE	$\simeq$ 0.8 (Ideal) [-4.25]	$\simeq$ 0.2 (Ideal) [0.84]	$\simeq$ 0.8 (Ideal) [-18]	$\simeq$ 0.5 (Ideal) [0.95]

Schema Level

- Does the model including cycles in the class hierarchy ?
  No, we tried to avoid cycles using definitions taken from Wordnet.

- Does the Model uses any polysemous terms for its class or property name?
  No, every entity has been defined using a Wordnet class. Even the Artist class, which can by either described as a CreativePerson or as a Performer, has been disambiguated by setting its defintion.

- Is Multiple Domain / Range defined for any property ? Yes

- Does any class have more than one direct parent class ?
  No, we do not have multiple inheritance in our ontology.

- Does the Model include multiple classes which have same meaning ?
  No, we excluded any possibility of classes having the same meaning using Wordnet definitions.

- Is the class Hierarchy over specified?
  No, we just kept the most relevant inheritances (isA relationships) that were needed to fully identify the ontology structure.

- Does the model use isA as a object Property or relation?
  No, isA is enforced in the entity structure, not as a relation. We map the partOf relation as an onbject property intead.

- Does the model have any leaf class for which there is no relation with the rest of the model?
  No, all leaves are linked to at least one dataset.

- Did you use miscellaneous or others as one of the class name?
  No, all data properties that cannot be mapped onto the ontology are not integrated. The ontology is easily expandable to include most of the properties that are not currently mapped.

- Does the model have any chain of Inheritance in class hierarchy?
  Yes, for example we have City isA AdministrativeDivision isA Location; from Location the entity City inherits the attributes latitude and longitude.

- Do all properties have explicit domain and range declarations? Yes, except for the partOf relation which is used twice with different domain/range (City partOf State, State partOf Country) and enforced when integrating in Karma.

- Does the model have any classes or properties which are not used? No, we kept it as clean as possible.

- Are a collection of elements included as a group in a number of class/attribute ? No, we have no *Other* entity; all information that cannot be directly mapped onto the ontology are discarded by the integration process.

Linguistic Level

- Does all elements of the model (i.e. class and property) have human readable annotations? Yes, taken from WordNet and Oxford dictionary.

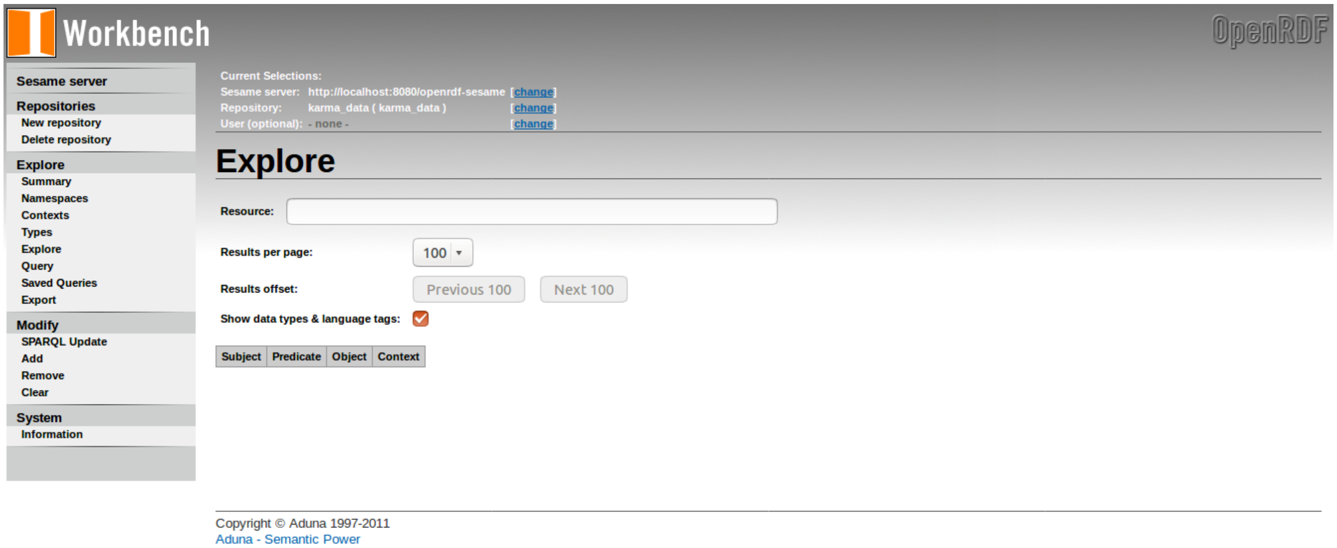
- Do all elements of the model follow the same naming convention? Yes.

Metadata Level

- Is provenance information (Creator, Version, Date) available for the final protege model? No.
- Is provenance information available for any property or class which is taken from some reference standard or ontology? Yes, all things that are taken from outside the project have been linked.

6 DB generation proposal

At the end of the integration task we exploited the Karma integrated Sesami triplestore to test our queries. A triplestore database is our main proposal for the final dataset. A MySQL database can also be instantiated, given that our ontology is based on an ER model that can be easily mapped to a relational database.



No actual improvement is needed, since the exported data can be directly imported in any triplestore database.

7 Final considerations and open issues

At the conclusion of this project we can say that the main task of mapping and querying artists and their concerts and reviews has been completely successfully. Although the number of entities and attribute we put in our ontology might seem to be limited, we kept working inside the scope of the project; mapping other information would have brought us to a larger solution that could have failed in addressing the main task

Linkable ontology The final result of this project is an ontology that can be easily extended; one first improvement would be to represent our person as a foaf:Person. Another possible integration would be to find a connection between the LODE entities and our ontology. We kept the ontology as open as possible, to let it easily expand to future improvements.

Other improvements One of the elements we saw was present in our datasets but that we have not considered in the scope of our project was the entity Album. Other interesting entities to be added are Song and other types of concerts.

References

[1] *Lazio dataset*, http://dati.lazio.it/catalog/it/dataset?tags=SPETTACOLI.

[2] *Lode: An ontology for linking open descriptions of events*, http://linkedevents.org/ontology/.

[3] *Lodse: Linking open descriptions of social events*, http://www.mdpi.com/2078-2489/9/7/164/pdf.

[4] *Los angeles dataset*, https://data.smgov.net/Public-Services/Calendar-Events/6aer-mv8t.

[5] *Mtv dataset*, https://gist.github.com/mbejda/9912f7a366c62c1f296c.

[6] *Santa monica dataset*, https://data.lacounty.gov/Arts-and-Culture/Free-Concerts-in-LA-County-Public-Sites/m2fm-jjhb.

[7] *Spain dataset*, https://datos.gob.es/en/catalogo?q=cultur&sort=metadata_modified+desc.

[8] *Wordnet*, http://wordnetweb.princeton.edu/perl/webwn.