Vrije Universiteit Brussel

Open Information Systems 2017 - 2018

Final report - Group 8

ROMAIN Maximilien - 0543411

(maxromai@ulb.ac.be)

ROJAS Felipe - 0542569

(felipe.rojas@vub.be)

PERALE Thomas - 0546990

(thomas.perale@vub.be)

LEHAL Sherik - 0543118

(sherik.lehal@vub.be)

December 22, 2017

1 Teamwork

We tried to divide the workload equaly for everyone and doing a lot of teamwork to satisfy the requirements of the project also with a lot of communication through Facebook

2 ER Diagram

The main objective of the project was to create an online bookstore, where the users can buy and rate ebooks from the database which has sorted the books according to their categories. All ebooks are identified by their ISBN number. The books also have other attributes such as Year, Title, Publisher which is a different class linked to the main ebook class by "isPublishedby" relation. The ebook class also has a relation to the Category class, which allows users to search different books from different categories. It is also possible to have subcategories inside another category. The author class contains the name, ID of the authors, linked to the ebook class. This further adds more to detail the ebooks. The user class contains various attribues such as the user ID, name, Payment method etc. This class is linked to the ebook and Purchased class . The purchase class as a weak entity, since it will need the foreign keys of class user and class eBook to create his primary key, exists to keep a history of all purchases made from the database. The user class also has an attribute, a role class. It specifies wheather the user is an admin, or a normal user. An admin is able to manage all the data. The rating class is linked to the ebook class and contains all the rating given to books by the users, after they havepurchased the book. In short, this allows us to create a system, where the users can login and purchase a book. The user can also check history of their purchases. After purchasing, the user can then rate the book to review it.

3 Ontologies

As our application was relatively straightforward, essentially allowing only ebook purchases, the ontology was also straightforward in seeking to

focus on the main elements of our application: its buyers and ebooks. As a result, our ontology has mainly focused on developing interactions between users and ebooks. It can be seen by the integration of relationships between the user and books to verify that he has bought it or a relationship to link a user to the ratings he gives to ebooks. The ebooks, the other main element of ontology, is surrounded by other classes with whom it has relations of belonging, especially authors, publishers and categories. These classes make sense because they allow users of our ontology to use them to search and filter specific ebooks among a variety of attributes that we are used to see in similar applications.

4 Good Relations

In order to adapt the system to the Good Relations standard and ensure the consistency we select three different classes of this standard, explained below. in which we didn't have to modify a lot our conceptual schema and ontologies, so with ensure our consistency by just adding this classes and connecting them to the ones that already exist, which are eBook and Purchase(explained below).

ProductOrService: This class has two subclasses which are eBook and ProductOrServiceInstance(Good Relation Subclass). Making eBook a subclass of this standard we can deduce that an eBook is a product, and that it would have instances of it.

PaymentMethod: Implementing this class with our Purchase class will be easier to standarized the payments methods of our project. This class has two payment methods, one is through the good relation individual named PayPal and other with a a subclass of payment method named PaymentMethodCreditCard that has two good relations individuals named MasterCard and Visa.

DeliveryMethod: This class is also with the class Purchase, since when a client purchase an eBook, our system should Deliver our product, there is where Good Relations standard is implemented. For the Delivery Method we implement also a subclass of Good Relations standard named DeliveryModeDirectDownload since our product is eBooks.

5 Rules description

First rule : If a book belongs to a category and that category is also a subcategory to another category, it implies that the book belongs to both the main category and its subcategory.

Ebook(?x), Category(?y), Category(?z), subCategoryOf(?y, ?z), hasCategory(?x, ?y), differentFrom(?y, ?z) → hasCategory(?x, ?z)

This rule is relevant because an ebook could have more than one category and one of those categories could be a subcategory of one of these, which means that the ebook will have a category and a subcategory. Example : if a book belongs to *"war"* category and the *"war"* category belongs to *"action"* category, it is implied that the book belongs to both "war" and "action" categories.

Second rule : if a user makes a purchase and an ebook is part of the purchase, then we can imply that the user *"HasPurchased"* an ebook.

User(?x), Purchase(?y), Ebook(?z), isMaking(?x, ?y), isPartOf(?z, ?y) → hasPurchased(?x, ?z)

This rule is relevant because a purchase needs at least one ebook purchased by the user, meaning that a user purchased an eBook.

Third rule : if a user is rating an eBook or an eBook has been rated implies that the user hasPurchased an ebook. The rule is relevant because users who did not purchase books cannot rate them.

User(?x), Rating(?y), Ebook(?z), hasRated(?x, ?y), hasRating(?z, ?y) → hasPurchased(?x, ?z)

This rule is relevant since only users that has purchased an eBook would be able to rate that eBook.

The last rule implies that if a user has the role "admin", they can manage the ebooks database on the system.

User(?admin), Ebook(?book), AdminRole(?role), hasAdminRole(?admin, ?role) → manage(?admin, ?book)

This rule is relevant because an admin should be the only type of user

that can manage eBooks.

6 Sparql endpoint & Mapping

To be able to implement the sparql endpoint in the project, we used the tool *D2RQ*[1]. D2RQ is Open Source software proposes a language of association between ontologies and databases, and an endpoint *Sparql* allowing query the database through *Sparql* queries. We used for the project, one of the specifications of the tool, allowing to map the ontology to the database directly by using the database, creating a mapping file. "This file called the default mapping, maps each table to a new RDFS class that is based on the table's name, and maps each column to a property based on the column's name"[2]

This mapping can be done by using the following command in the D2RQ repertory :

```
./generate-mapping -u user -p password -o mapping.ttl
    jdbc:mysql://localhost:PORT/databaseName
```

It is then possible to run the server to be able to use the provided Sparql endpoint and to make queries with the Sparql syntax providing by the mapping.

```
./d2r-server mapping.ttl
```

6.1 Mapping discussion

Since we did not use the mapping from the ontology in our project, we will discuss some mapping we could have done for the mapping.

6.1.1 D2RQ Mapping tool

D2RQ provide a tool to map the ontology / database by hand for every classes, attibutes or relation from the ontology. This tool is the *D2RQ Mapping Language* allowing more control over the mapping.

[1] `http://d2rq.org`

[2] `http://d2rq.org/generate-mapping`

There is a example for the Ebook Class and the attributes *author* and *title* :

```
@prefix : <http://www.example.com/ontologies/myontology> .
# Namespace of the mapping file; does not appear in mapped data
@prefix map: <file:///Users/d2r/mapping.ttl#> .

# Other namespaces
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

map:Ebook a d2rq:ClassMap;
    d2rq:dataStorage map:Database;
    d2rq:class :Ebook;
    .
map:title a d2rq:PropertyBridge;
    d2rq:belongsToClassMap map:Ebook;
    d2rq:property :title;
    d2rq:column "Ebook.Title";
    d2rq:datatype xsd:string;
    .
map:author a d2rq:PropertyBridge;
    d2rq:belongsToClassMap map:Ebook;
    d2rq:property :author;
    d2rq:column "Ebook.Author";
    d2rq:datatype xsd:string;
    .
```

6.1.2 Protégé Mapping

The regular way to map the ontology to the database is to use protégé with the ontop reasoner.

Let's take the example of mapping the Ebook Class from the ontology with the Ebook Table from the database. We want for example to be able to retrieve all the information from the database for the Ebook. We then have

to need the SQL query to the Triples Template using the ontop mapping editor.

We then need to map the SQL query that retrieve all the information from the Ebook table :

```
SELECT isbn, title, author_id, publisher_id
FROM ebook_eBook
```

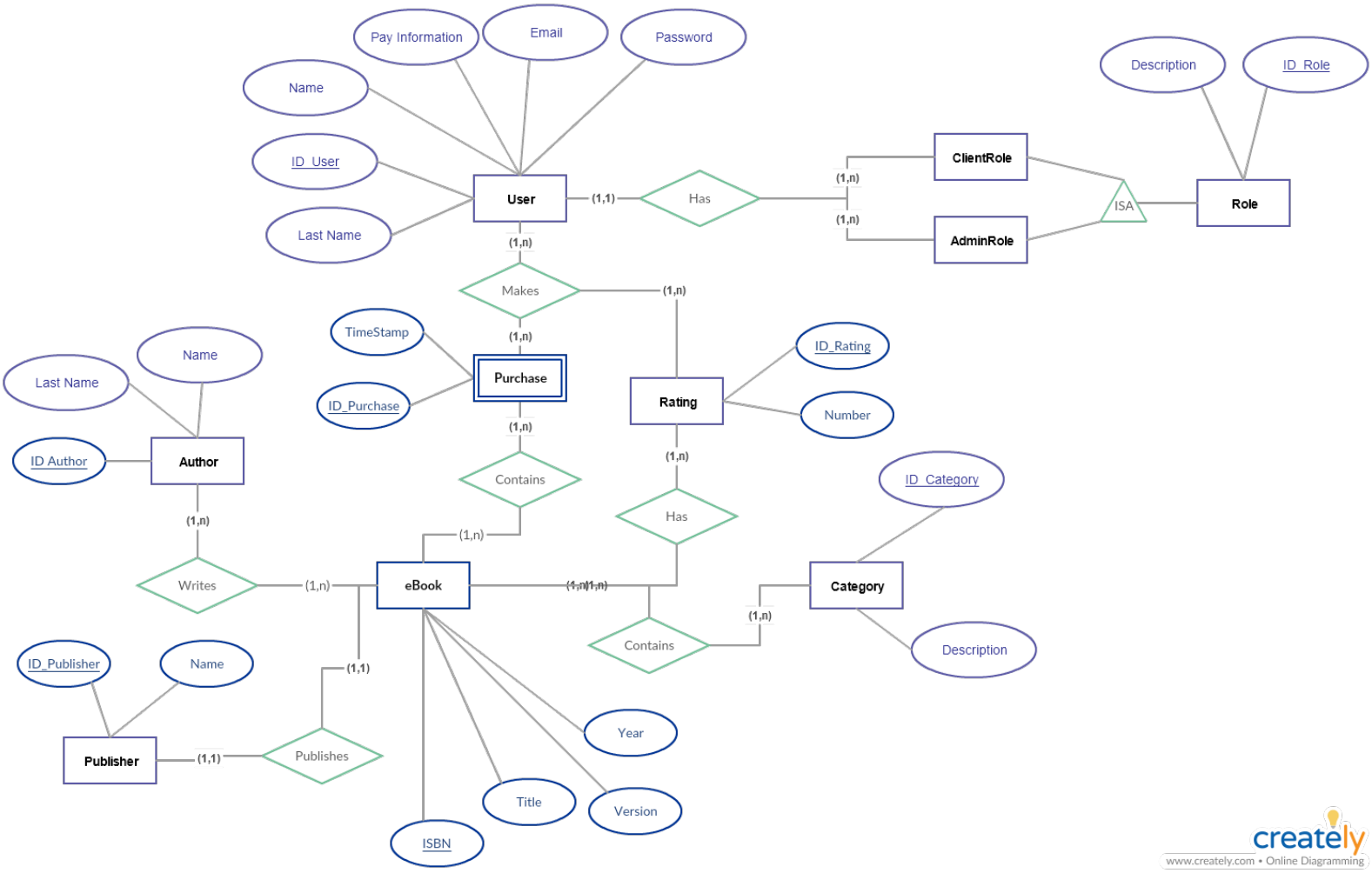
With the Target syntax in protégé :

ebookStore:{isbn} a ebookStore:Ebook , ebookStore:Author , ebookStore:Publisher ; ebookStore:title {title} .

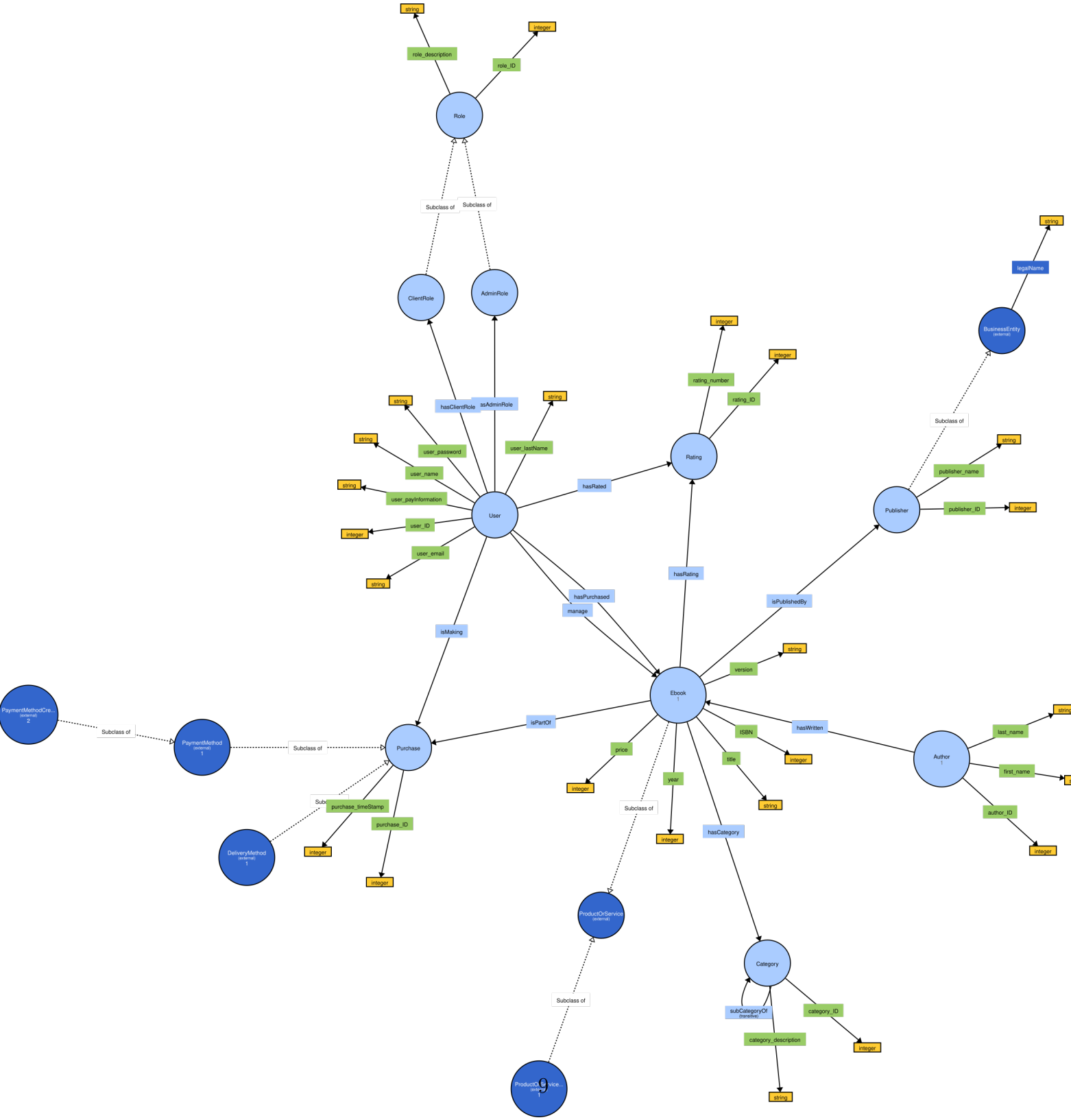
We have therefore associated the attributes from the database to the rdf triple. *Author* and *Publisher* are in yellow because they are referecing to another Class. We can then make the SPARQL query :

```
PREFIX : <http://www.semanticweb.org/ebookStore.owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT DISTINCT ?p ?title ?author ?publisher
WHERE {
  ?p a :Ebook ; :title ?title ; :Author ?author ;
  :Publisher ?publisher .
}
```

7 ER Diagram



8 WebOwl


string
integer
role_description
role_ID
Role
Subclass of
Subclass of
ClientRole
AdminRole
string
integer
integer
rating_number
rating_ID
Rating
string
hasClientRole
hasAdminRole
string
user_password
user_lastName
string
user_name
hasRated
string
user_payInformation
User
integer
user_ID
string
user_email
string
hasPurchased
manage
hasRating
isMaking
isPublishedBy
string
version
Ebook
1
isPartOf
hasWritten
price
ISBN
integer
integer
title
year
integer
string
Subclass of
hasCategory
integer
PaymentMethodCre...
(external)
2
Subclass of
PaymentMethod
(external)
1
Subclass of
Purchase
Sub
purchase_timeStamp
purchase_ID
integer
integer
DeliveryMethod
(external)
1
ProductOrService
(external)
Subclass of
Category
subCategoryOf
(transitive)
category_ID
category_description
integer
string
ProductOrService...
(external)
1
Subclass of
BusinessEntity
(external)
string
legalName
Subclass of
Publisher
string
publisher_name
publisher_ID
integer
Author
1
string
last_name
first_name
string
author_ID
integer