Open Information System: Conceptual schema

Thomas Perale – 0546990
Maximilien Romain – 0543411
Felipe Rojas – 0542569
Lehal Sherik – 0543118

December 9, 2017

1 Project subject

Our OWL have seven classes, Category, Ebook, Purchase, Publisher, Author, User, Publisher and Role.

The *User* class represent both an admin or a customer of the platfrom, it have seven attributes:

- userId
- information
- name
- email
- password
- lastName

A *User* of the system can either have the *AdminRole* or just have the *ClientRole*, relation like *hasAdminRole* or *hasClientRole* from the *User* domain in the *AdminRole*/*ClientRole* range. Both *AdminRole*/*ClientRole* are subclasses of *Role* they are used to be more specific what permission a *User* have but are characterized by the attributes:

- roleId
- description

Having one of these role imply the *User* can interract with the *Ebook* has shown with the relations *hasPurchased* and *manage*. *Ebook* are the main component of the platform because it's simply what we sell on it. They are defined by:

- ISBN
- title

- year
- version

But also a remote *Author* class bound to *Ebook* by the *hasWritten* class who have got the following attributes:

- authorId
- firstName
- lastName

Also defined with the *Publisher* class bound from *Ebook* by the relation *isPublishedBy*, it is made of:

- publisherId
- name

And finally *Categorie* class bound by *Ebook* by the *hasCategory* relation, categories are just a way to do search on ebooks by genre so the *Category* class is defined by:

- categoryId
- description

To enable customers to read ebooks they must first create *Purchase* as you can see in our ontology a *User* make by the *isMaking* relation *Purchase*. *Purchase* are a way to modelize ebooks the user paid to have for each *Purchase* the user have to make a transaction and they can contain more than one *Ebook* has we can see with the *isPartOf* relation.

Other privilege a customer have is to rate the *Ebook* he bought modelized by the *Rating* class. *User* and *Rating* are bound together by the *hasRated* relation and make *Ebook* in relation with it with *hasRating*. The attributes for *Rating* are:

- ratingId
- number

2 Good Relations Description

This project implements the below three main classes of the Good Relations standards.

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.

3 Rules description

First rule, If the 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 it's subcategory.

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

Is relevant because an ebook could have more than one cateogory 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 to be purchased by an 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 didn't purchase the book can't rate it.

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

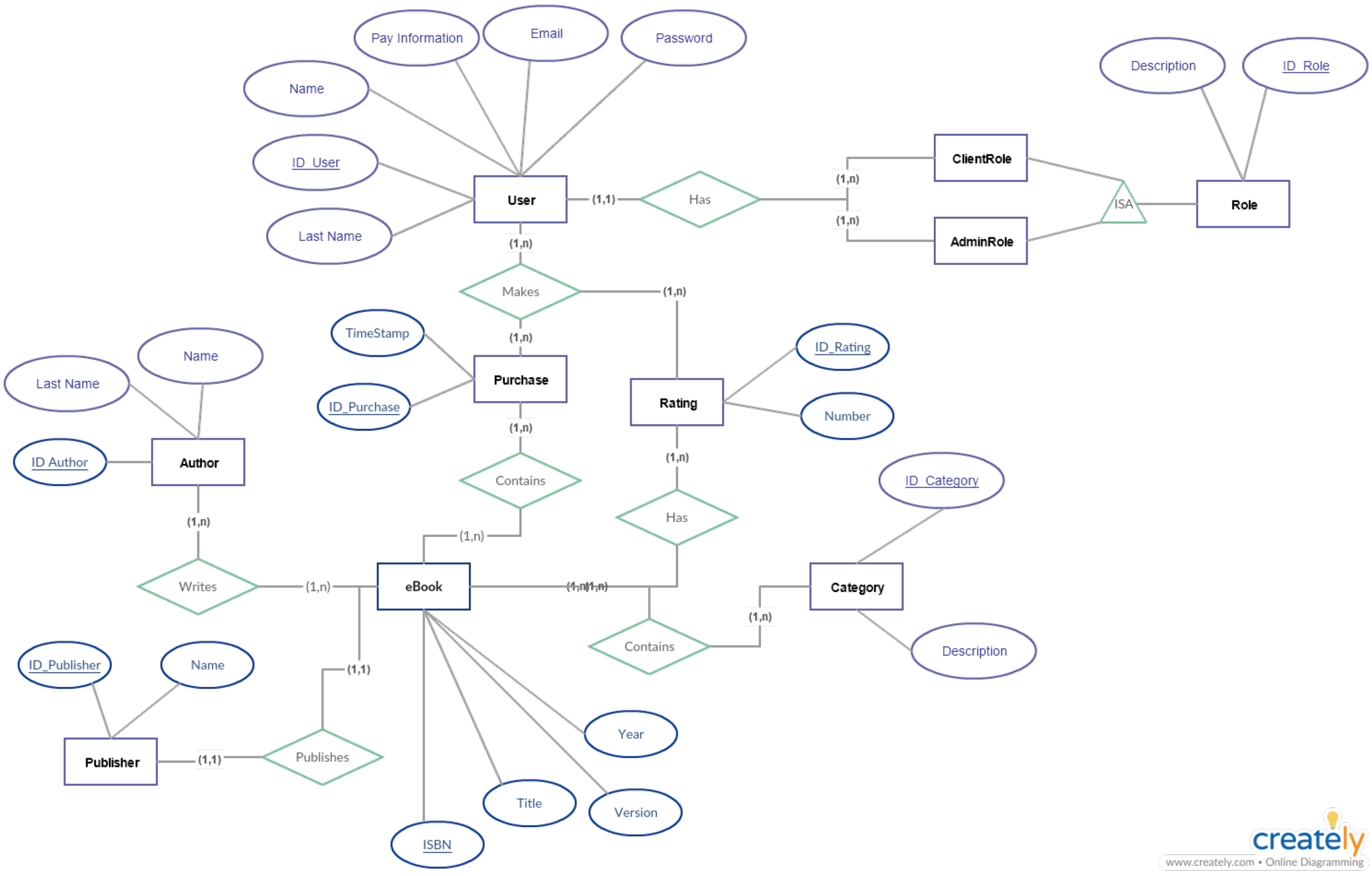
This rule is relevant since only users that has purchased an eBook would be able to rate that eBook.

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)

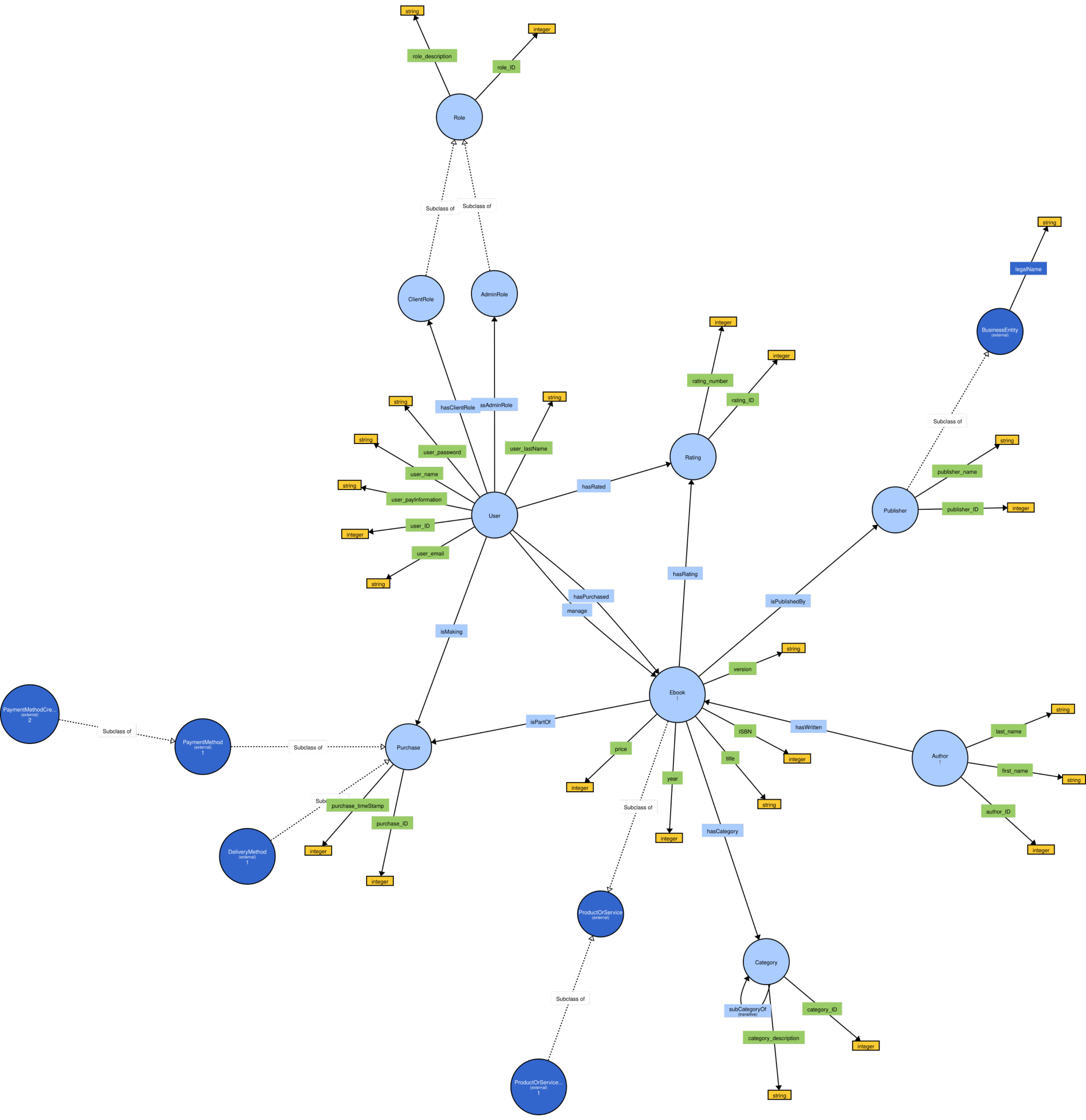
This rule is relevant because an Admin should be the only type of user that can manage the eBooks.

4 ER Schema



Pay Information
Email
Password
Name
ID_User
Last Name
User
(1,1)
Has
(1,n)
(1,n)
ClientRole
AdminRole
ISA
Role
Description
ID_Role
(1,n)
Makes
(1,n)
(1,n)
TimeStamp
ID_Purchase
Purchase
Rating
ID_Rating
Number
(1,n)
(1,n)
Contains
Has
ID_Category
Category
Description
Last Name
Name
ID Author
Author
(1,n)
Writes
(1,n)
(1,n)
eBook
(1,n)(1,n)
Contains
(1,n)
(1,1)
ID_Publisher
Name
Publisher
(1,1)
Publishes
Year
Title
Version
ISBN
creately
www.creately.com • Online Diagramming

5 WebOwl visualisation


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