CM4330

Semantic Web and Ontology Modelling

Assignment

Ontology for banking system

Faculty of Information Technology

University of Moratuwa

2020

Table of Contents

Table of Figures

1. Introduction

Web Ontology is knowledge representing language and a formal way to describe taxonomies and classification networks, essentially defining the structure of knowledge for different domain areas. In a nutshell, ontologies are frameworks for representing shareable and reusable knowledge across a domain. Their ability to describe relationships and their high interconnectedness make them the bases for modeling high-quality, linked and coherent data. There are different tools are available for implement web ontologies and Protégé is one of them. Protégé is an open source ontology development editor built using Java which provides the facility of creating, modifying and uploading ontologies with full support of OWL.

Considering the domain, we have selected is Banking area and to develop this ontology several areas has been discovered. In the initial stage we have explored the hierarchy and relations belong to bank domain. Hence this is wider area, we have selected limited area for implement this ontology.

This report is included detailed description about the Banking ontology, the concept and the relationships among different individuals and their properties as well. To properly implement this ontology, we have sub-divided and ordered class in appropriate manner and the rest of paper explain it well.

2. Bank Ontology

2.1 Class

In general, a class can be considered as the key component of OWL ontology and also it can be defined as a set of individuals who possess precise requirement to be member of that class. In our project "ontology for banking system" we have defined eight main classes. Since the class "Owl-thing" represent all the individuals, all the classes are sub-classes of OWL-thing class. Figure 2.1 provides a detail view of the class hierarchy in the Bank ontology.

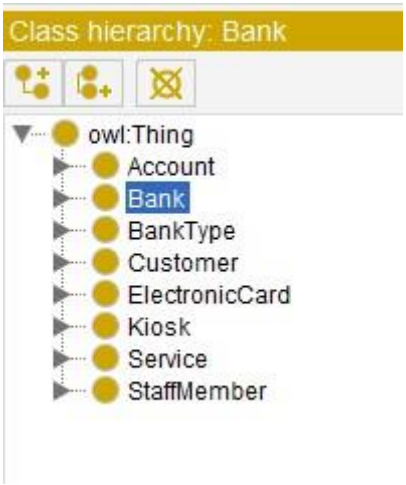

Figure 2.1: Class Hierarchy of Bank Ontology

Complete Class Hierarchy

As you can see in the figure the main class in the OWL: Thing class. All the individuals are instances of this Bank class. Not only that but also it explains the basic classifications of a Bank by displaying the categories as list of classes and a named class to express variety of services, accounts and persons. The Bank class consists of object properties which make relationships with its sibling classes. Following figure 2.2 is displayed it well.

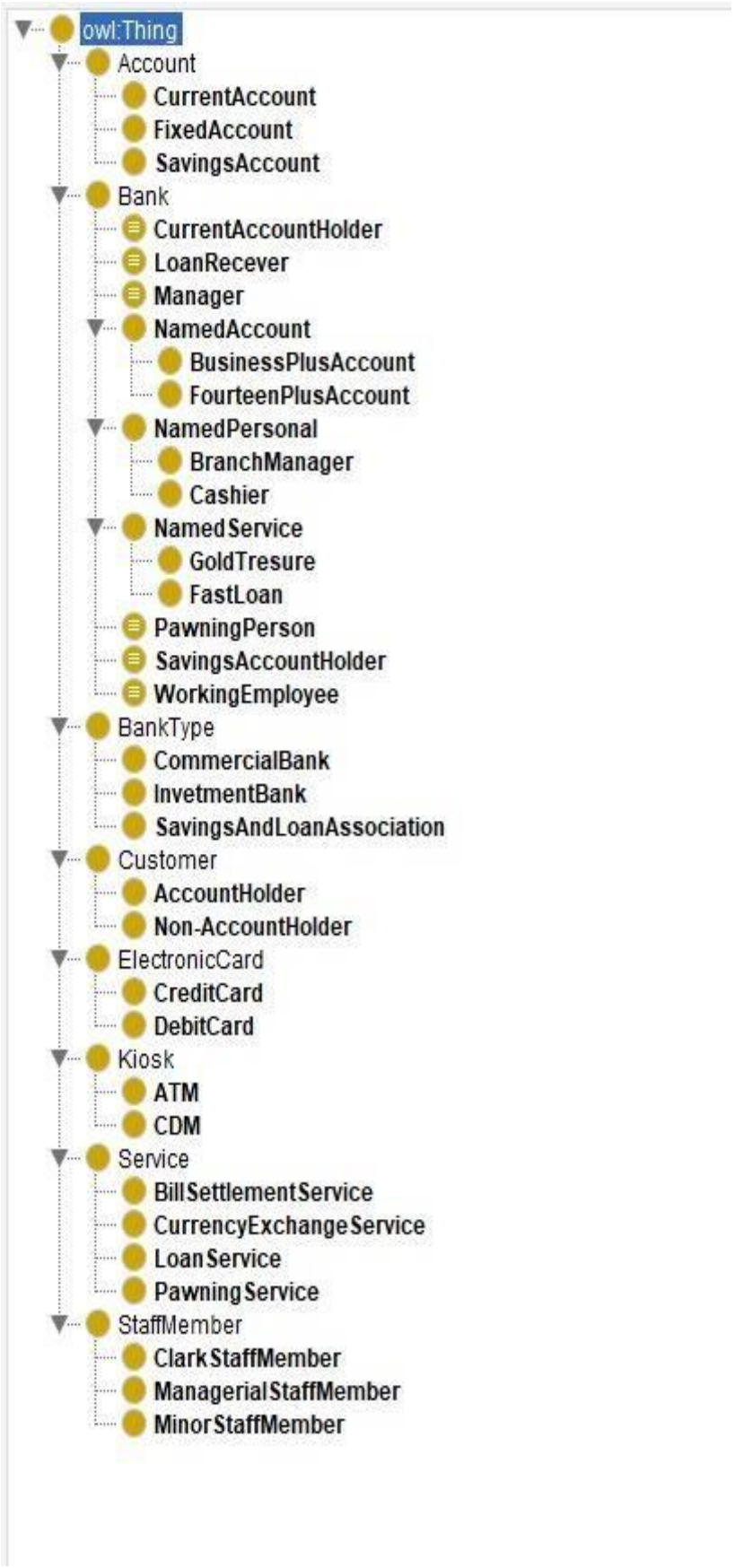

Figure 2.2: Complete class hierarchy of Bank Ontology

Bank Class

Bank class hold the subcategories of services, account and persons who engaged with the bank. In here Bank class consist nine subclasses called, NamedService, NamedAccount, NamedPersonal, LoanRecever, WorkingEmployee, Manger class, SavingAccountHolder, PawningPerson and CurrentAccountHolder class. NamedService class is included two subclasses knows as FastLoan and GoldTresure class. NamedAccount has BusniessPlussAccount subclass and ForteenPlusAccount subclass. NamedPerson class is included two subclasses called BranchManger and Cashier class.

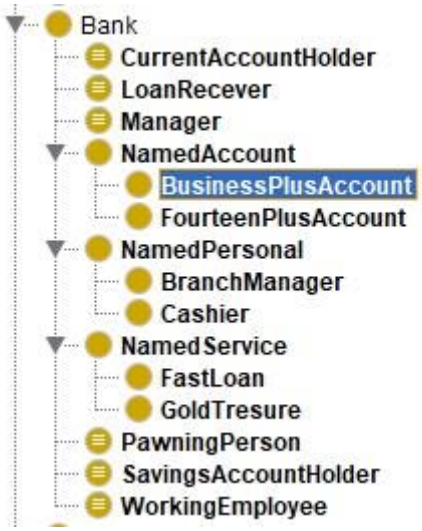

Figure 2.3: Bank class

StaffMember Class

StaffMember class is a main class in this ontology and it is comprised of three subclasses called ClarkStaffMember class, ManagerialStaffMember class and MinorStaffMember class. And also this class has two instances Employee_01 and Employee_2.

Figure 2.4: StaffMember class

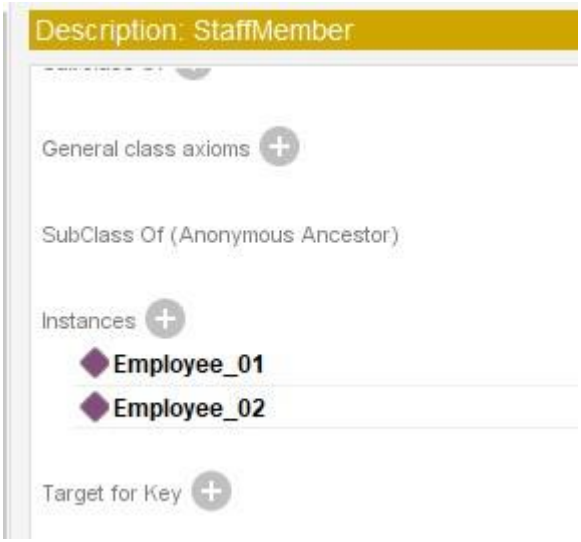

Figure 2.5: Instances of StaffMember class

Account Class

Under Account class there are three main types of subclasses are included. Those are CurrentAccount class, FixedAccount class and SavingAccount class. Account_1, Account_2 and Account_3 are the instances of this class.

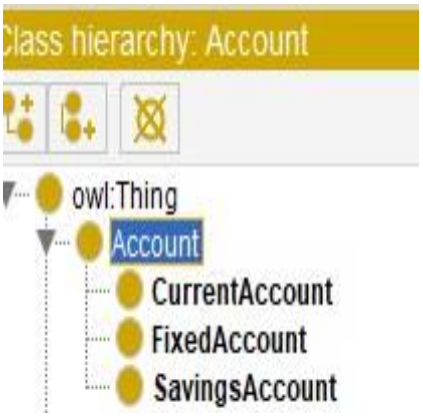

Figure 2.6: Account class

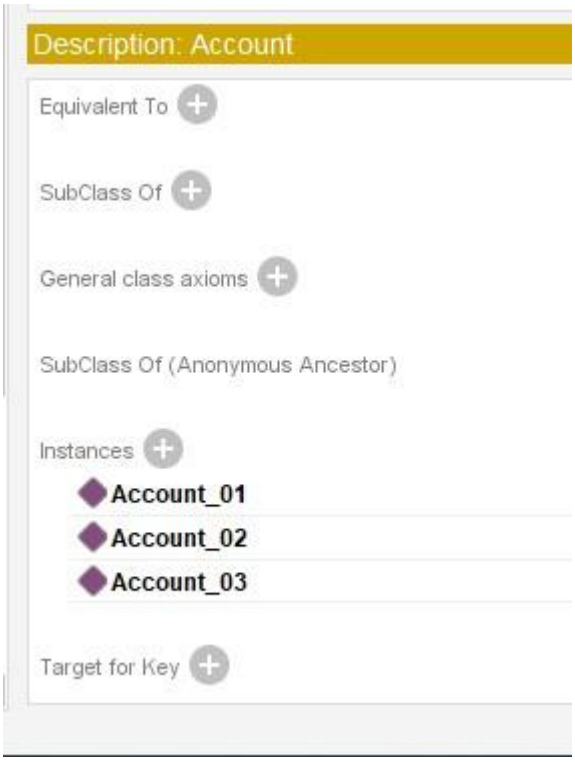

Figure 2.7: Instances of Account class

BankType Class

Considering the bank type there are lots of type bank are available in Sri Lanka. In here we have only implemented this BankType class under three main subclasses and they are as follows. CommericalBank class, InvestmentBank Class and SavigAndLoanAssociation class.

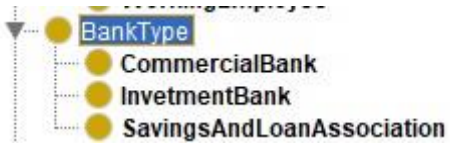

Figure 2.8: BankType Class

Customer Class

Considering the service provided by the bank we can categories this class into two parts. AccountHolder and Non-AccountHolder. The people who are maintain an account is included into AccountHolder class. The customer who come to the bank for apply loan, pawning, bill payment is included into to the Non-AccountHolder class. This customer class has four instances

Figure 2.9: Customer class

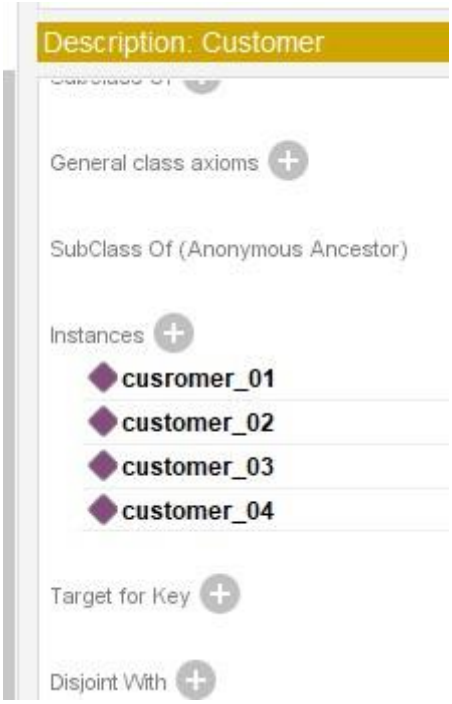

Figure 2.10: Instances of Customer class

ElectronicCard Class

In Bank domain there are two main card systems are available. By considering them we have implemented our ElectronicCard class under two main subclasses. They are CreditCard class and Debit Card class.

Figure 2.11: ElectronicCard class

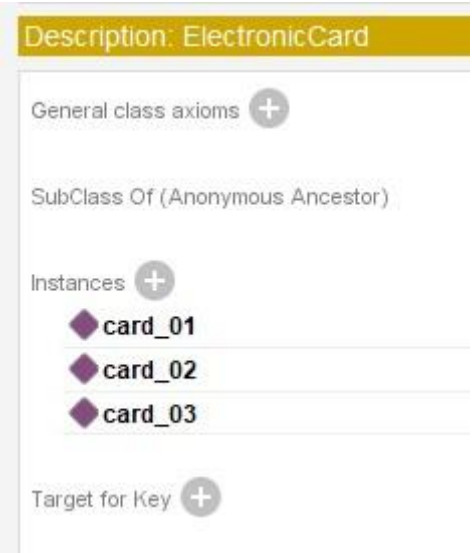

Figure 2.12: Instances of Electronic class

Kiosk Class

ATM and CDM kiosk are available all bank island wide for facilitating their customers with cash exchange facility. So, we have included kiosk also as another main class with subclasses of ATM and CDM as you see below. This class has two instances service_point 1 and service_point2.

Figure 2.13: Kiosk class

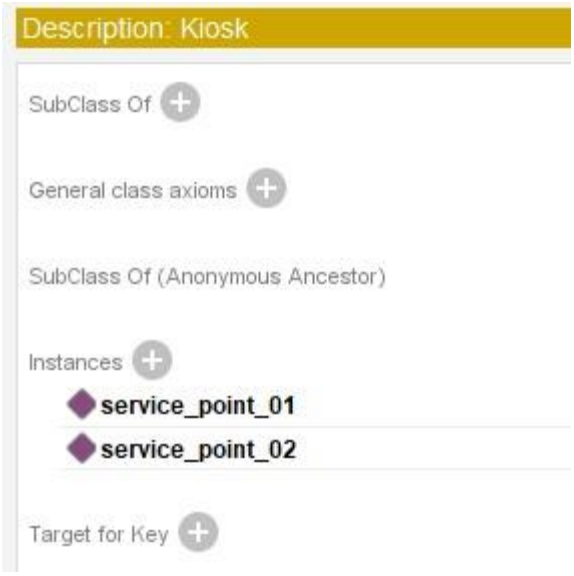

Figure 2.14: Instance of Kiosk class

Service Class

There are different types of Service are provided by a Bank. In here we have only focused in few services for implement this ontology. Under Service class there are four main subclasses called BillSettlemenService class, CurrencyEXchnageService class, LoanService Class and PawiningService class.

Figure 2.15: Service class

2.1.1 Sibling Class

Sibling class can be explained as descendent classes who possessing the same parent class who inherits the same set of properties and same set of characteristics of a single parent. Those descendent classes may have their own properties and characters but also, they inherit same set of features as explained above. All the subclass relations do possess these sibling relationships. So, set of classes with same parent are sibling classes of each other. Example for it in related to Service class will be given below.

Figure 2.16: Sibling class

BillSettlementService class, CurrencyExchangeService class, LoanService class and Pawning class are the sibling class of Service class.

2.1.2 Disjoint Classes

If two classes are said to be disjoint classes, then a particular individual cannot be an instance of more than one of those classes. For an example in the bank ontology, the sub classes of StaffMember class which are ClarkStaffMember, ManagerialStaffMember and MinorStaffMember are said to be disjoint classes. This is a main concept that we have taken.

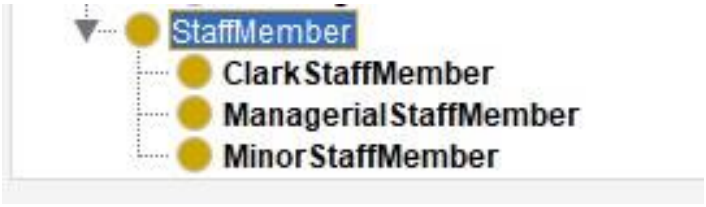

Figure 2.17: Disjoint class

2.1.3 Primitive and Defined classes

Primitive Class

A primitive class is a class which consists of necessary conditions only, which means there are no sets of necessary and sufficient conditions. Simply a necessary condition provides a requirement to be a member of a class that means if an individual is a member of a class, then that individual should have to fulfill a requirement. Hence, this does not mean that a certain individual who fulfills that requirement is a member of that particular class.

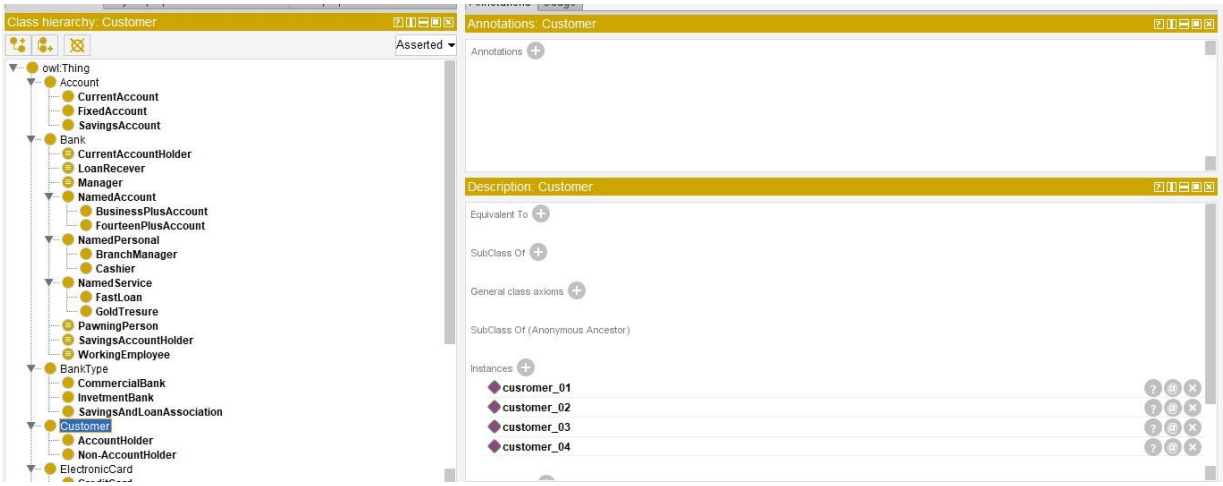

Figure 2.18: Primitive class

Defined Class

A defined class is a class with at least one set of necessary and sufficient conditions. This type of classes has an exact definition and any individual which satisfy that will be a member of that class. In a simple manner that means, if an individual fulfills the necessary conditions and at the same time has sufficient conditions to satisfy the class restrictions, then that particular class can be identified as a defined class.

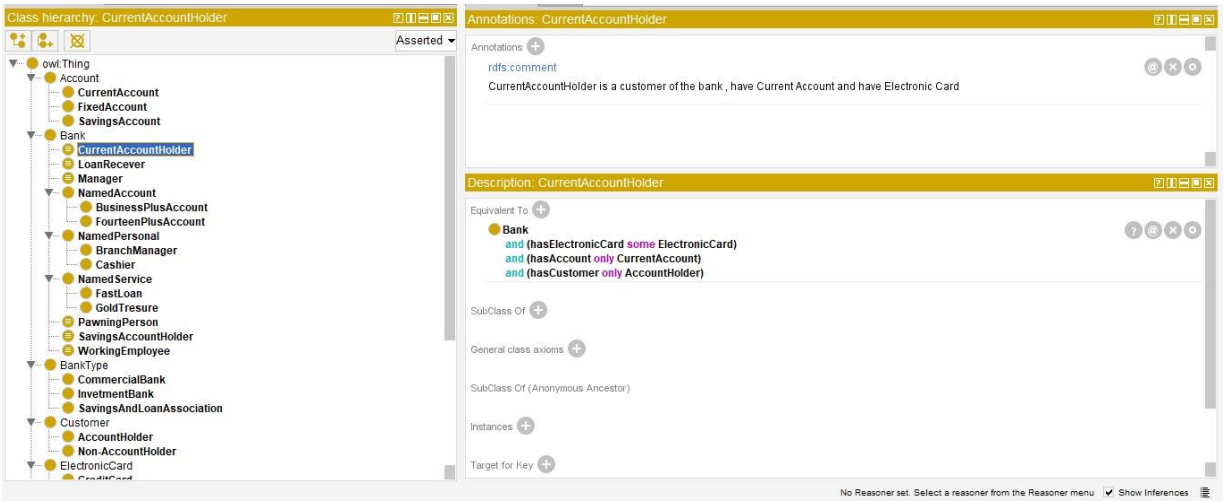

Figure 2.19: Defined class

CurrentAccountHolder class

This is a defined class in this ontology and it is included the customer who has a current account in the bank. To include an instance for this class, they should be an account holder in a particular bank and should have only current account also. And also, they should have at least any kind of electronic card.

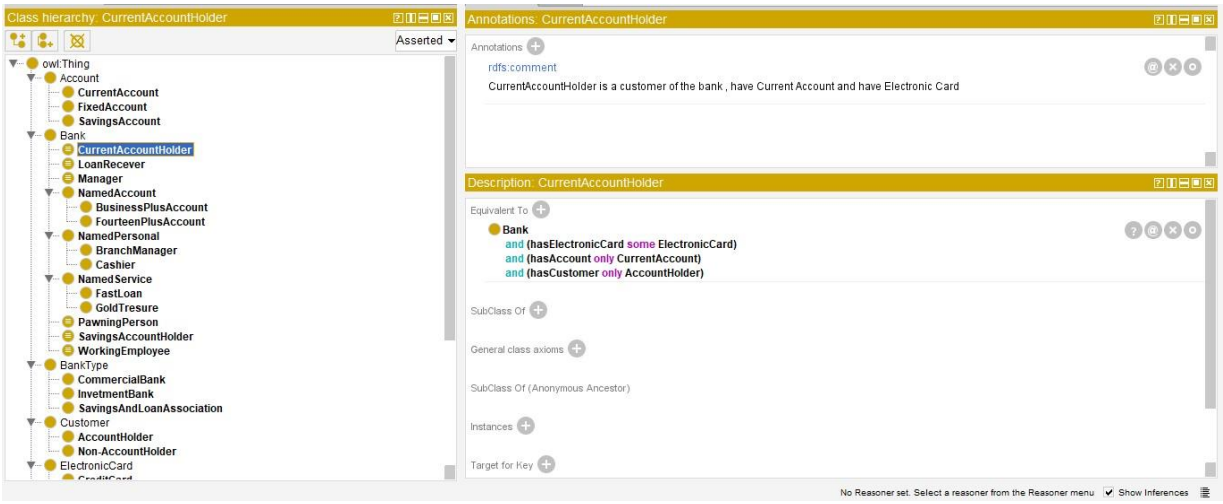

Figure 2.20: CurrentAccountHolder class

LoanReceiver

Loan Receiver is a person who is received a loan from a particular bank and he should be a customer of the bank and also, he should open an at least one account in that bank. And he should only get the loan service from the bank.

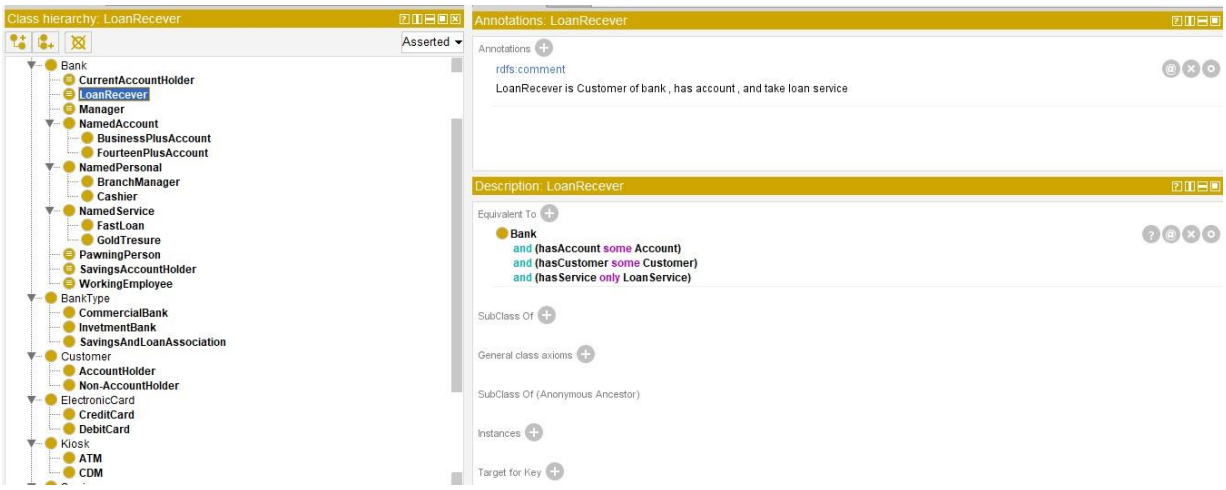

Figure 2.21: LoanReceiver class

Manager

Manger is the person who has responsibility to manage all activates in a bank and should being a only a managerial staff member.

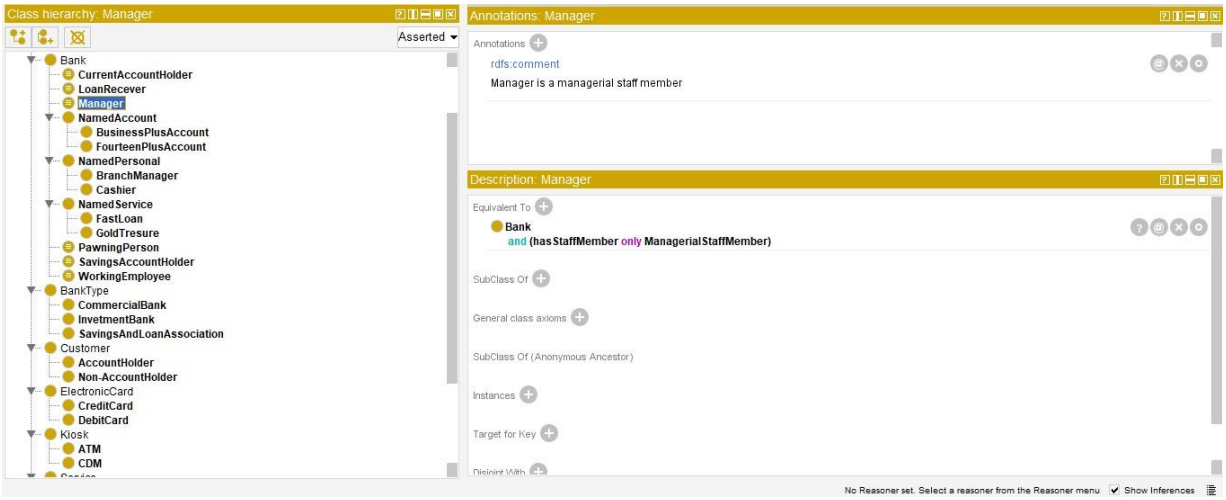

Figure 2.22: Manager class

Pawning person

Pawning person means who is came to bank to get the service of pawning. And at least that person should be a customer of the bank and get the only pawning service.

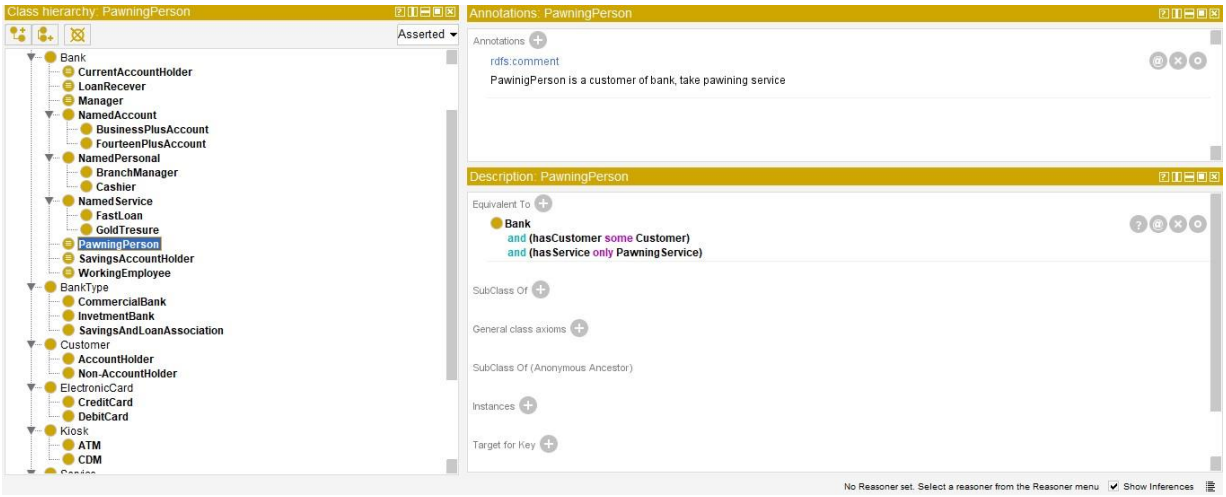

Figure 2.23: Pawning Person

SavingAccountHolder

To become saving account holder, that person should only have saving account and should be account holder and also may have at least one kind of electronic card also.

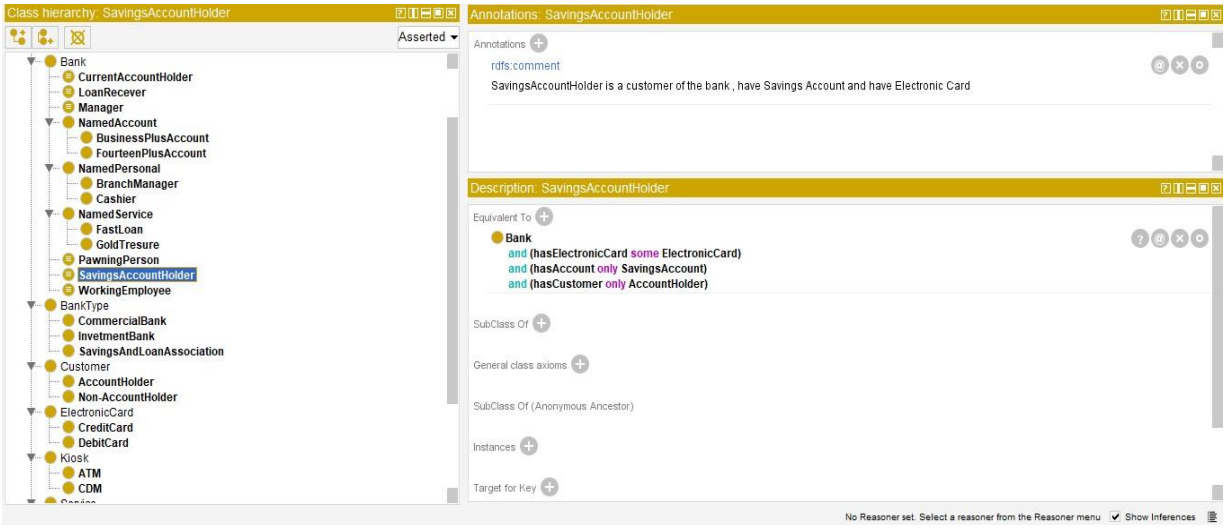

Figure 2.24: SavingAccountHolder

Working Employee

To being a working employee, the particular person should only in the ClarkStaffMember class.

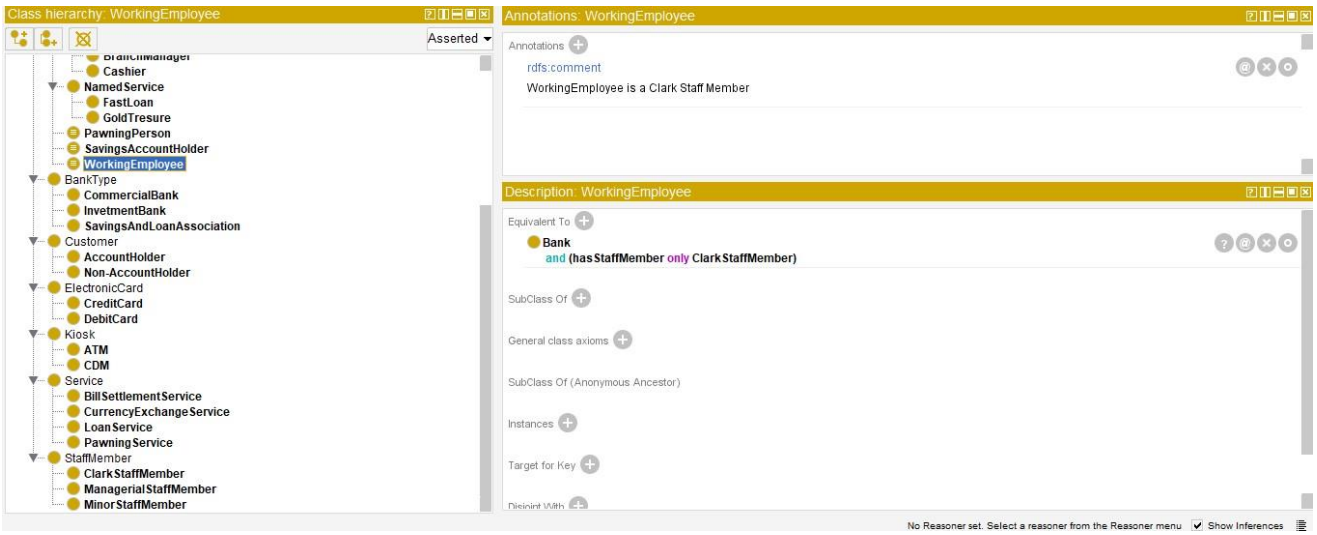

Figure 2.25: Working Employee class

Property Restrictions

OWL object properties are used to express the relationship between two individuals. Restrictions on the other hand, describes a class based on the relationships it's individuals have. Furthermore, this restricts how the members should be linked to another member of a class. This is also kind of a class. Property Restrictions are classified into three main sub-categories.

• Quantifier Restrictions

• Cardinality Restrictions

• hasValue Restrictions

2.1.4 Quantifier Restrictions

Describes the relationship between the individuals among classes.

Existential Quantifier

Existential Quantifier means for a given instance there should be at least one instance connected via a given property. Key word “some” is used to describe Existential Quantifiers in Protégé. In the bank ontology, Existential Quantifier are being used to describe dew classes. In saving account Holder class, it is used. This class has relationships with ElectronicCard class, SavingAccount class and AccountHolder class. In here, existential quantifier is used to describe that the being a saving account holder need at least one individual from Electronic card class. Following figure will show it clear.

Figure 2.26: Existential Quantifier

2.1.5 Cardinality Restrictions:

Cardinality Restrictions define how many distinct values a property can have forgiven subject. Minimum and maximum cardinality define using owl:minCadinality and owl:maxCardinality

keywords in owl expressions. It can specify a precise number by using the same minimum and maximum number.

Example: According to the bank ontology, one credit card has only one account. If one card belongs to one account, it cannot belongs to another account.

Figure 2.27: Usage of cardinality restriction

2.1.6 HasValue Restrictions:

HasValue allows to define classes based on the existence of particular property values, their values must be at least one matching property value.

Example: According to bank ontology, Account_01 is a type of Account in the hierarchy.

Figure 2.28: Usage of HasValue restriction

2.2 Instances

Instance are the individual representation components of real-world objects and entities of given domain in owl.

Example: According to bank ontology, Customer_01 to Customer_04 belongs to the customer subclass. Those are instances of customer class.

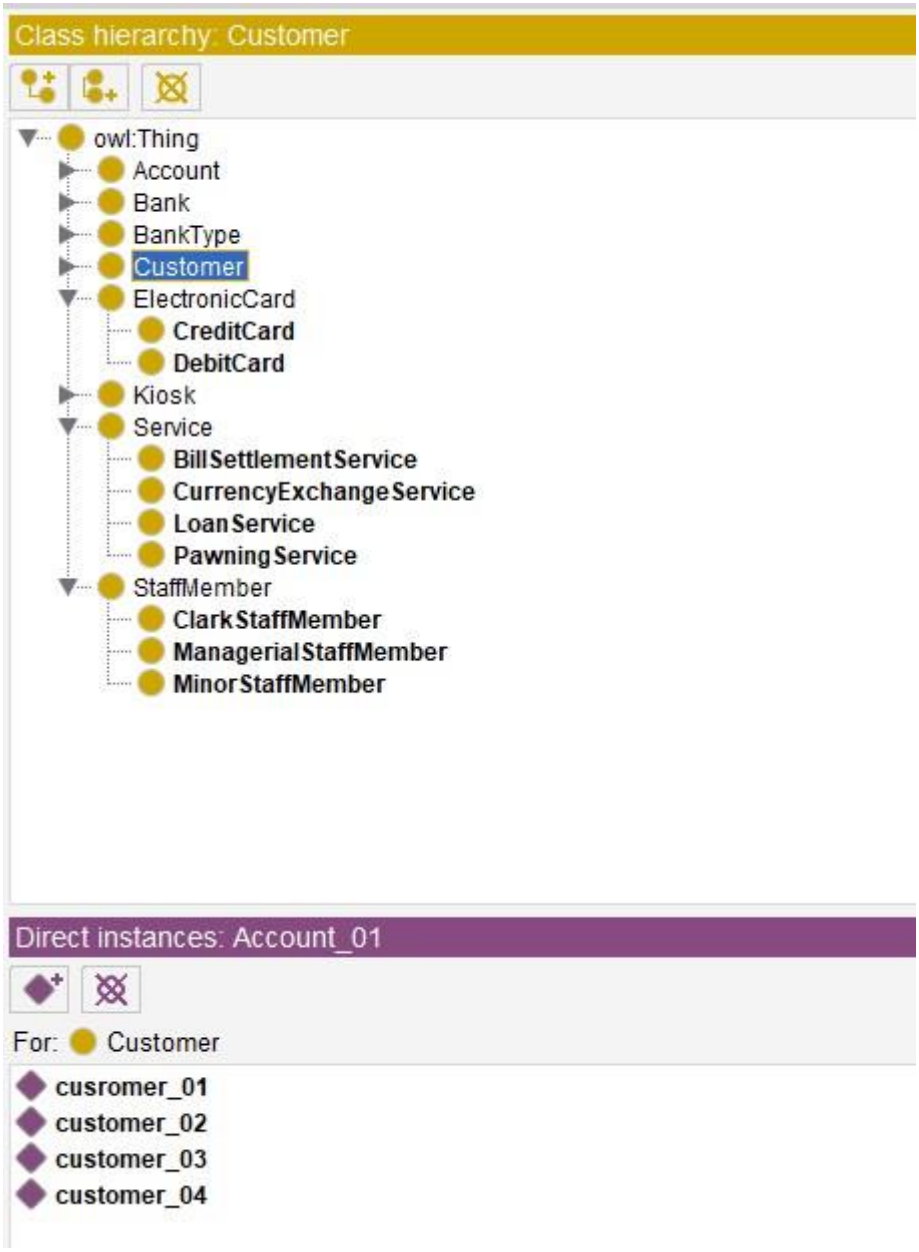

Figure 2.29: Representation of Individuals

2.2 Properties

Properties used to represent the relationship in ontologies while annotations of properties are used in adding information to individuals, classes, and properties. There are two types of properties named Object properties and Data type properties.

2.2.1 Object Properties

All relationship among individuals illustrated by under object properties. All object properties of the ontology represent under owl:topObjectProperty collapsible in the hierarchy structure.

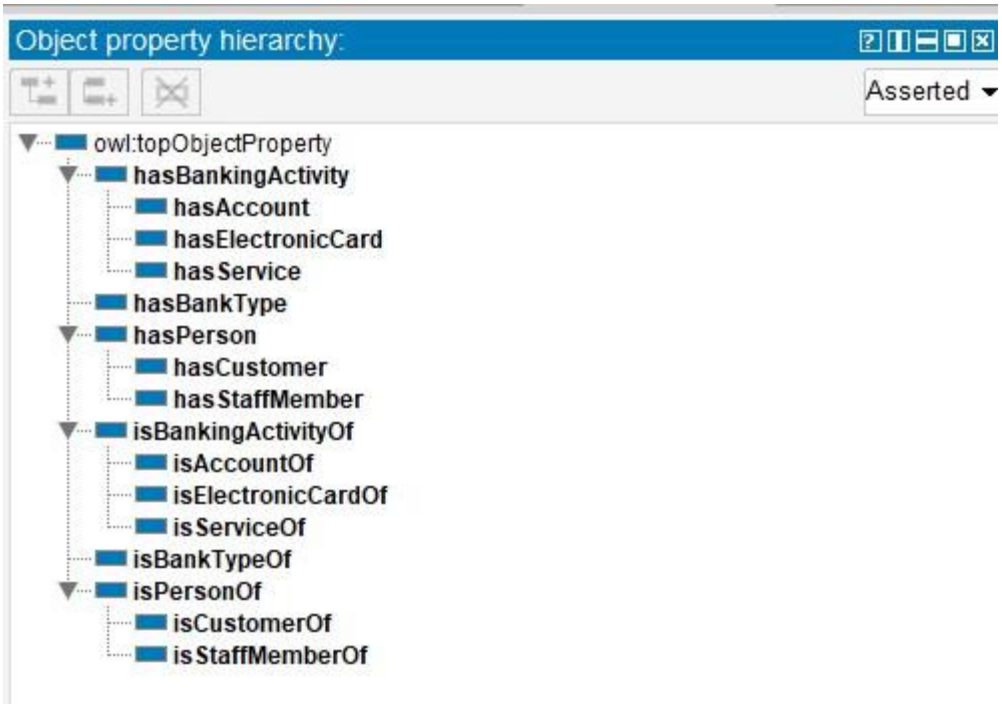

Figure 2.30: Object Properties

Every object property consists of a domain and range attribute. Relationship between property and individuals will be developed on this range. List of such linked relations is given in below table for the object property hierarchy.

Object Property	Domain	Range
hasBankingActivity	Bank	Service, ElectronicCard, Account
hasAccount	Bank	Account
hasElectronicCard	Bank, Account	ElectronicCard
hasService	Bank	Service
hasBankType	Bank	BankType

hasPerson	Bank	Customer, StaffMember
hasCustomer	Bank	Customer
hasStaffMember	Bank	StaffMember
isBankingActivityOf	Service, ElectronicCard, Account	Bank
isAccountOf	Account	Bank
isElectronicCardOf	ElectronicCard	Bank, Account
isServiceOf	Service	Bank
isBankTypeOf	BankType	Bank
isPersonOf	Customer, StaffMember	Bank
isCustomerOf	Customer	Bank
isStaffMemberOf	StaffMember	Bank

Table 1.1: Property Domain and Ranges

Inverse Properties

Some of the relationships in ontologies are not only existing for the forward direction but also an existing relationship in backward direction also, those properties are called inverse properties. But the thing is domain and range of the inverse property is depending on the corresponding property.

Example: According to bank ontology, isElectronicCardOf object property is inverse property of hasElectronicCard object property.

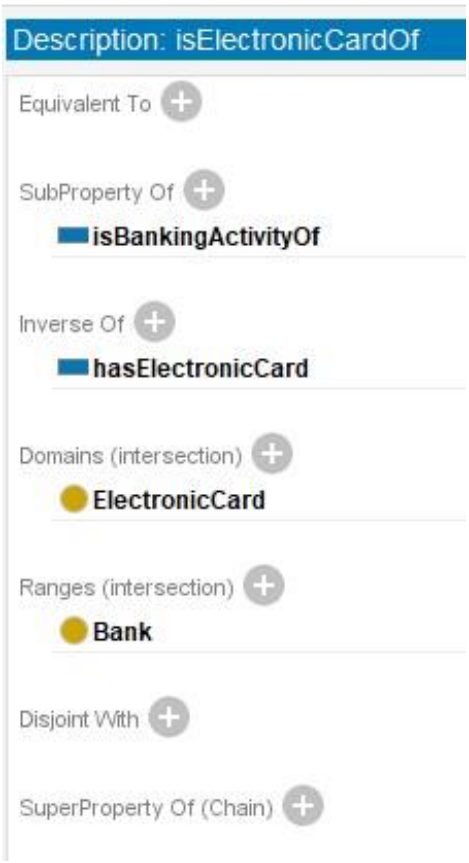

Figure 2.31: Inverse property example

Property Characteristics

The Object Property characteristics view displays the asserted characteristics for the selected object property. Characteristics are proven as a list of checkboxes. If the container is checked for a given characteristic that capacity that some ontology in the imports closure of the active ontology asserts that characteristic. If the container in unchecked for a given characteristic that skill that the characteristic is now not asserted at all in any ontology in the imports closure of the active ontology.

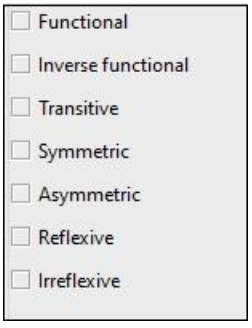

Figure 2.32: Property Characteristics

- **Functionals**

For a given individual, it can be having at most one relation to another individual by connected via the property.

Example: : According to bank ontology, functional charasteristics of hasElectronicCard property shows in following figure.

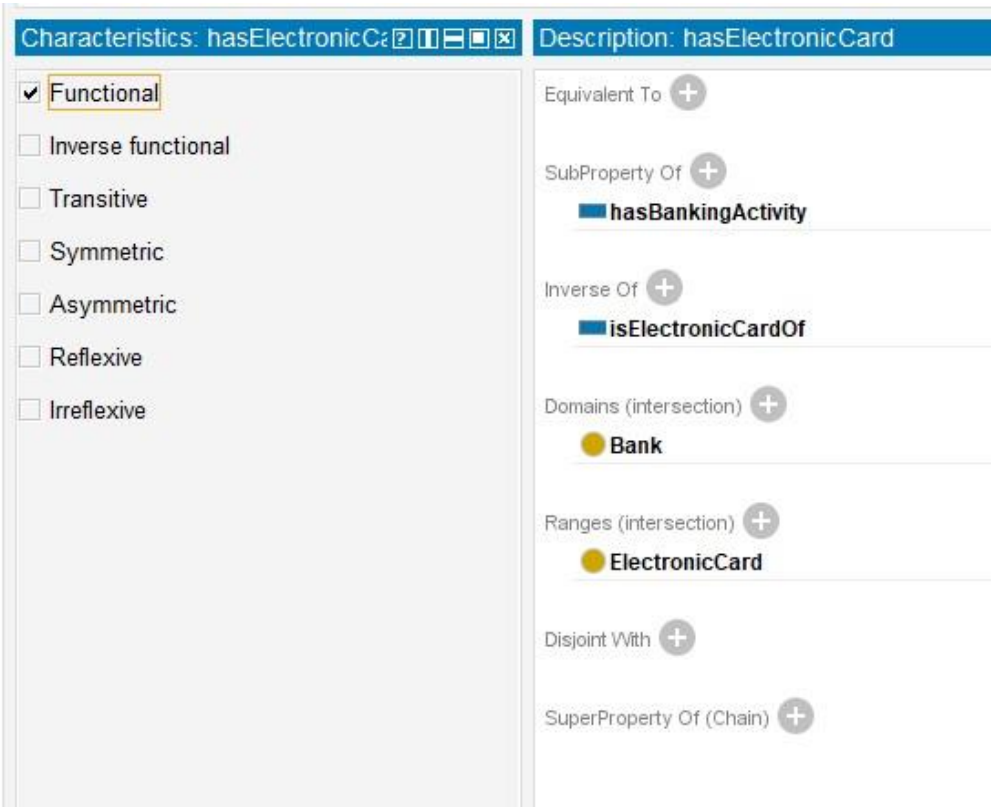

Figure 2.33: Functional Characteristic of hasElectronicCard property

- **Inverse Functionals**

The inverse property of functional property is called as inverse functional property.

Example:

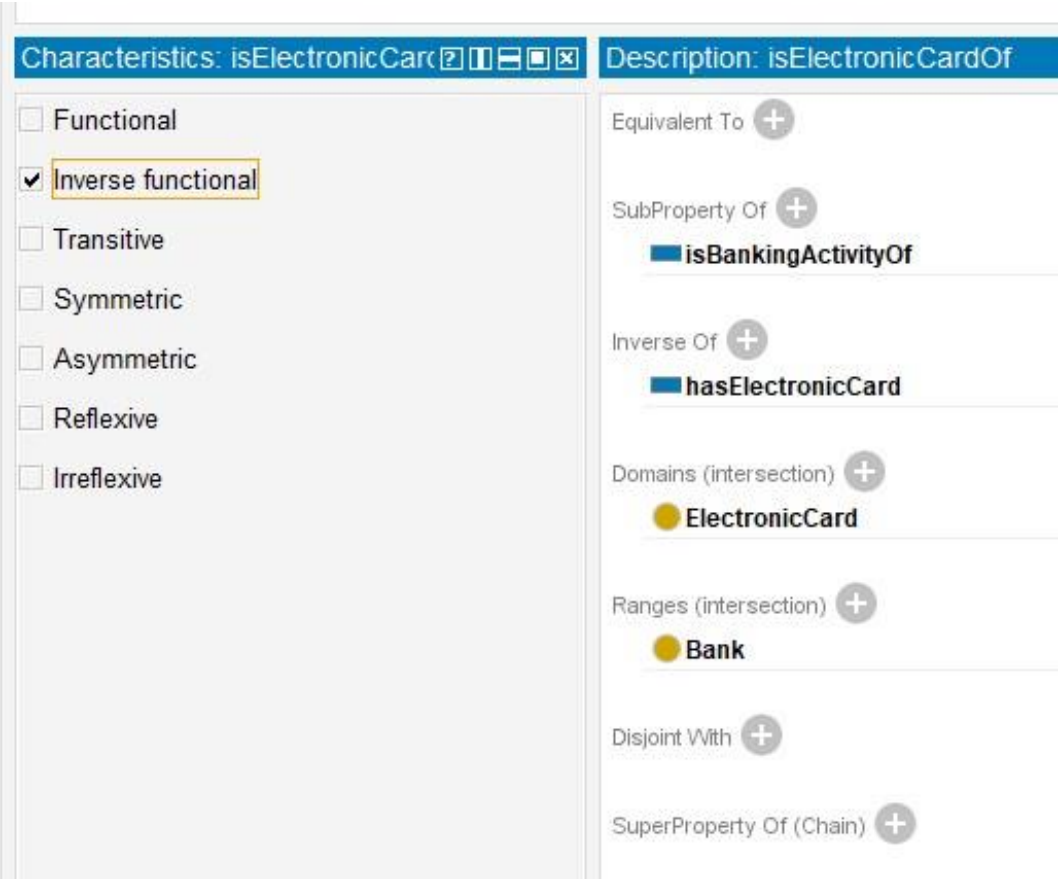

Figure 2.34: Inverse functional characteristic of isElectronicCardOf property

2.2.2 DataType Properties

To build a relationship between an individual and an XML schema value of RDF literal is use Datatype properties. The hierarchical structure for all datatype properties is illustrated under owl:topDataProperty.

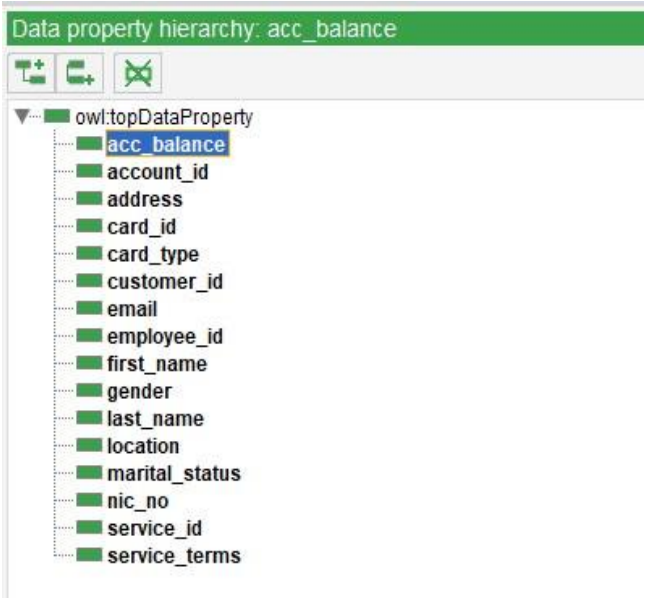

Figure 2.35: Datatype properties

2.3 Reasoner

Reasoner in semantic (protégé), plugin piece of software able to infer logical consequences from a set of asserted facts or axioms. After starting reasoner on already defined ontology, it will automatically validate the rules and do the all reasoner related actions (Ex: classify taxonomy, Compute inferred types,.)

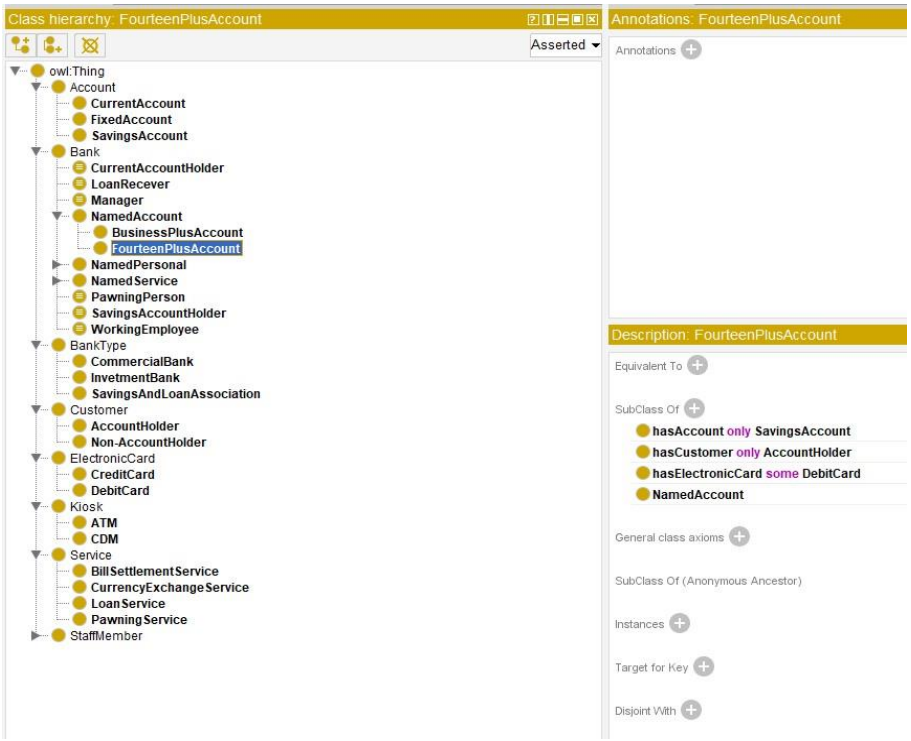

Figure 2.36: Bank ontology before start reasoning

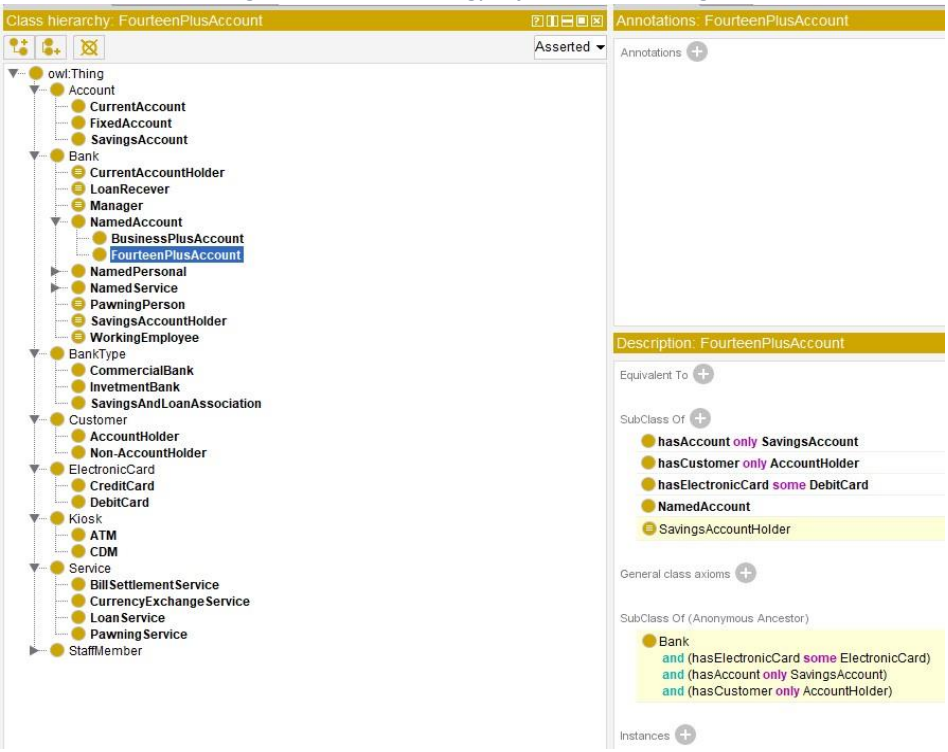

Figure 2.37: Bank ontology after start reasoning