Проведено аналіз існуючих на сьогоднішній день ORM фреймворків для відображення OWL\POJO. Запропоновано модель універсального ORM фреймворку об'єктно-семантичного двонаправленого відображення, функціонуючого на основі HTTP REST SPARQL

Ключові слова: онтологічні системи, ORM, OWL, RDF, SPARQL

Проведен анализ существующих на сегодняшний день ORM фреймворков для отображения OWL\POJO. Предложена модель универсального ORM фреймворка объектно-семантического двунаправленного отображения, функционирующего на основе HTTP REST SPARQL

Ключевые слова: онтологические системы, ORM, OWL, RDF, SPARQL

An analysis of currently existing ORM frameworks for OWL\POJO mapping has been conducted. A model of a universal ORM framework of the object-semantic bidirectional mapping, functioning on the basis of HTTP REST SPARQL has been proposed

Keywords: ontology system, ORM, OWL, RDF, SPARQL

УДК 519.7:007.52

РАЗРАБОТКА МОДЕЛИ И АРХИТЕКТУРЫ УНИВЕРСАЛЬНОЙ БИБЛИОТЕКИ ДЛЯ ОТОБРАЖЕНИЯ OWL\POJO

А.Н. Богдан
Кафедра искусственного интеллекта
Харьковский национальный университет радиоэлектроники
пр. Ленина, 14, г. Харьков, Украина, 61166
Контактный тел.: 067-120-19-64
E-mail: dickens3101@gmail.com

1. Введение

Роль ORM библиотек при создании объектно-ориентированных приложений хорошо известна. Они позволяют нивелировать разрыв между ООП, в котором классы представляют объекты реально мира (на их основе легко и естественно строится логика приложения) и реляционной моделью (в такой форме эффективно сохранять данные о конкретных персистентных экземплярах сущностей моделируемой предметной области в базе данных, откуда потом легко запросить нужный фрагмент информации с сохранением свойств экземпляров и отношений между ними) [1]. При отказе от выражения персистентных сущностей моделируемой предметной области в виде объектов, программы становятся сильно связными, трудночитаемыми, «захламленными» низкоуровневым кодом, на который к тому же накладываются ограничения реляционной модели, не свойственные ООП.

ORM библиотеки автоматизируют написание модулей, отвечающих за преобразование запросов к объектам программы в SQL, а результатов – обратно в объекты внутри программы. ORM избавляет программиста от написания большого количества кода, часто однообразного и подверженного ошибкам, тем самым значительно повышая скорость разработки. Кроме того, большинство современных реализаций ORM позволяют программисту при необходимости самому задать код SQL-запросов к постоянным объектам, который будет использоваться при запросах к ним или сохранении изменений. Благодаря этому работа с БД через ORM становится более быстрой, использует меньше памяти, чем при полностью автоматической генерации универсального кода запросов/транзакций [2].

С аналогичной проблемой сталкиваются и разработчики интеллектуальных приложений на основе онтологических БЗ [3]. Только вместо обработки более низкоуровневой реляционной модели они сталкиваются с семантическим разрывом при необходимости преобразования на этот раз, с одной стороны, более высокоуровневых онтологических отношений в термины ООП. Например, множественное наследование (от которого уже отказались большинство современных ОО языков), наследование свойств, принадлежность одного и того же свойства ничем не связанным между собой нескольким классам, новые типы отношений, такие как: owl:SameAs, owl:InverseOf и т.д. С другой стороны (по аналогии с реляционной моделью) – необходимость более низкоуровневого персистентного представления онтологии, как набора RDF-триплетов, в сущности ООП.

Представляет интерес создание программного инструментария для устранения/уменьшения такого разрыва. Целью данной статьи является разработка базовой архитектуры такого инструментария.

2. Выделение базовых принципов построения фреймворка

Существующие в настоящий момент библиотеки, связывающие мир ООП и SemanticWeb можно разделить на две категории: 1) средства объектно-семантического двунаправленного отображения (по аналогии с ORM) и 2) односторонние генераторы Java скелета классов по OWL/RDF схемам и наоборот. Вторая категория, по сути, является урезанной версией первой.

Для начала необходимо показать, как происходит работа со знаниями онтологии без использования

ORM. Приведем пример на основе Jena API. Создание экземпляра класса «Статья» со свойствами «автор», «предметная область», «название»:

```
String NS = "http://purl.org/dc/elements/1.1/";
OntModel m = createModel();
OntClass articleCls = m.createClass(NS +"Article");
Individual i = articleCls.createIndividual(
          "http://examples.com/core/");
Property title = m.getProperty(NS + "title");
Literal    l = m.createTypedLiteral("Introduction to Jena");
           i.setPropertyValue(title,l);
Property creator = m.getProperty(NS + "creator");
           l = m.createTypedLiteral("Philip McCarthy");
           i.setPropertyValue(creator,l);
Property subject = m.getProperty(NS + "subject");
           l = m.createTypedLiteral("jena, rdf, "+
                        "java, semantic web");
           i.setPropertyValue(subject,l);
           m.write(System.out, "N3");
```

Как видно из примера, подход очень напоминает низкоуровневую работу с реляционной моделью БД. Приведем краткий анализ и сравнение ORM-библиотек для OWL-моделей.

So(m)mer

So(m)mer – это простой базовый инструмент для создания Java-объектов на основе RDF модели с довольно скудным перечнем возможностей (табл. 1). Он использует аннотации на подобии Hibernate, EJB3.0 и JDO [4]. Для отображения RDF в Java-объекты So(m)mer использует рефлексию, в частности, библиотеку манипуляции байт-кодом Javassist. Главной целью данного инструмента было сохранить его простым и таким, вероятно, он и останется.

Elmo

Довольно большой и развитый инструментарий для построения семантических веб-приложений. Он использует Sesame как средство доступа к RDF-хранилищу, поддерживает возможность сохранения и SPARQL запросы [5]. Он уже предлагает некоторые из аннотаций, позволяющие задать расширенные семантические отношения между сущностями модели, например inverseOf. Хотя исходный код немного обширный, в его ядре находится инструмент связывания между Java и RDF, который не намного сложнее, чем So(m)mer. Elmo обладает довольно развитым и документально описанным набором инструментов, который используется уже на протяжении нескольких лет [6].

Очевидно, что он был создан, чтобы сделать работу с хранилищем Sesame RDF для объектно-ориентированного Java проще.

AliBaba

AliBaba является следующим этапом кода Elmo. В то время как Elmo обеспечивает абстракцию над RDF, AliBaba предоставляет расширение к RDF репозиторию Sesame. Elmo предоставляет более OWL-подобные аннотации, а AliBaba был расширен для обработки более сложных OWL онтологий [7]. AliBaba все еще находится в бета-версии и пока доступно не так много документации.

AliBaba и Elmo отличается от других ORM фреймворков, тем, что они позволяют классам иметь несколько супер-классов, а объектам иметь несколько типов. На данный момент AliBaba не предоставляет возможностей использования аннотации inverseOf и других специфических свойств OWL и из его небольшой документации, доступной на данном этапе развития, не ясно, будут ли эти аннотации реализованы или нет. Хотя AliBaba активно развивается, он еще не очень подходит для преобразования данных из объектно-ориентированного кода Java в RDF / XML-формат и наоборот.

Empire

Empire предоставляет стандартный, широко известный фреймворк, сохраняющий состояния Java объектов для их использования в семантических веб-проектах, где данные хранятся в формате RDF [8]. Обеспечивая реализацию JPA, Empire использует его для абстракции деталей RDF, это снижает длительность обучения для новых разработчиков, а также помогает обеспечить прямой путь для переноса или расширения существующих традиционных веб-приложений, под использование семантических веб-технологий. В данном инструментарии реализованы только основы, а именно: ядро JPA фреймворка, EntityManager, запросы, EntityManagerFactory, и так далее.

Этот инструментарий еще вовсе не был документирован и для ознакомления с ним не хватает источников информации. Исходные коды находятся в открытом доступе, разработка продолжается, нет ни одного релиза. На данный момент Empire не поддерживает ни одну из желаемых аннотаций как, например inverseOf и на этом этапе развития инструмента нельзя этого изменить. Окончательной версии еще не существует. Вероятно, его преимущество заключается в основном в реализации JPA, возможно не в ближайшее время, но по крайней мере это является целью данного инструмента.

Jenabean

Jenabean почти недокументированный инструмент, который использует широко известный Jena фреймворк [9]. Исходный код обновляется время от времени. Это не очень надежный инструмент для пользования, довольно трудно что-либо узнать о работе с ним из-за отсутствия документации. Его единственное преимущество заключается в том, что он использует Jena фреймворк, что является многообещающим, так как Jena уже является известным и используемым разработчиками фреймворком.

RDFBean

RDFBean очень развитый инструмент с широким набором функций и некоторой простотой. Он поддерживает широкий спектр хранилищ, таких как последние Sesame, Virtuoso, поддержка Jena считается экспериментальной и еще не подтверждена, также поддерживает реляционные базы данных в качестве хранилища [10]. Доступ к хранилищу Sesame может быть быстрым за счет хранения онтологии в памяти или родном хранилище, так что можно быстро создавать модель в памяти и затем размещать её в родном хранилище.

RDFBean обладает небольшой документацией, что очень поможет начинающим пользователям в ознакомлении с его основными возможностями.

RDFBean отличный инструмент, который может предложить много свойств OWL. Он должен быть довольно простым, поскольку не использует настройку встроенных прокси или манипуляторов байт-кода. Доступен экспорт как в RDF/XML файлы, так и в RDF-хранилище. RDFBean до сих пор разрабатывается и поддерживается, он был использован в нескольких производственных системах.

RDFBeans

RDFBeans не очень развитый инструмент, он не предоставляет специальных аннотаций, которые являются основными для обеспечения отображения классов Java в RDF [11].

Код компилируется без проблем с использованием Maven. Документация не очень насыщенная, но полна примеров кода о том, как использовать различные функции RDFBeans. Неизвестно поддерживает ли он какой-либо экспорт в RDF / XML-файл, но, исходя из документации, RDF2Go должен поддерживать.

Topaz

Topaz является библиотекой, которая позволяет сохранять и делать запросы к RDF объектам. Он поддерживает использование языка запросов, который является родным для используемых за основу RDF хранилищ и свой собственный язык запросов Object (OQL), который используется для запроса на определенные объекты [12].

В настоящее время Topaz поддерживает только RDF хранилище Mulagara. А для хранения небольших данных он предлагает отдельное хранилище Fedora (Blob хранилище). Но оно не поддерживает запросов SPARQL. Topaz поддерживает основные аннотации, необходимые для маппинга и преобразования предиката (свойства) в объект. Он также учитывает мощность, коллекций, контейнеров и проблемы, связанные с ними.

Elmo, пожалуй, самый развитый и продвинутый инструмент в настоящее время, он предлагает богатую функциональность и вспомогательные инструменты, для помощи в решении некоторых конкретных задач, эти преимущества, ставят его на первое место. Основным недостатком Elmo является жесткая привязка к Sesame API.

Таким образом, если есть необходимость использовать его для другого типа репозитория, то нужен класс-прослойка (через jdbc) от желаемого репозитория к Sesame, например, Virtuoso Sesame provider. Что значительно замедляет работу всего приложения. Результаты сравнения проанализированных библиотек сведены в табл. 1.

Так как большинство RDF-репозиториев сейчас поддерживают доступ по HTTP REST, то логично и обосновано разработать универсальный ORM-фреймворк, функционирующий на основе HTTP REST SPARQL.

Проведенный анализ позволяет выделить следующие базовые принципы построения фреймворка:

– доступ к графу базы знаний на основе протокола HTTP REST SPARQL;

– декларативная форма задания отображения OWL/Objects с использованием аннотаций;

– оптимизация вычислений с использованием ленивой загрузки фактов;

– дуальный принцип логического вывода неявных фактов (участвуют как сервер, так и клиент в зависимости от контекста).

Таблица 1

Сравнение ORM-подобных библиотек для RDF

	AliBaba	Elmo	Empire	JenaBean	RDFBean	So(m)mer	RDFBeans	Topaz
Наличие документации	+	+	-	-	+	-	+	+
Возможность сохранения изменений в базу знаний	+	+	+	+	+	-	+	-
Поддержка стандартных аннотаций	+	+	+	+	+	+	+	+
Поддержка специализированных OWL аннотаций	-	+	-	-	+	-	-	-
Поддержка SPARQL запросов	+	+	+	+	+	?	+	-
Поддержка SeRQL запросов	-	+	+	-	-	?	-	-
Обладает собственным языком запросов	-	-	-	-	+	-	-	+
Отсутствие привязки к хранилищу	-	-	-	-	-	?	-	-
Открытый исходный код	+	+	+	+	+	+	+	+

3. Диаграмма классов ORM фреймворка

Пример возможной реализации фреймворка представлен на рис. 1. StorageManager представляет собой класс реализующий работу с онтологическими хранилищами.

Метод connection() обеспечивает создание и использование подключения к хранилищу. Метод transaction() обеспечивает передачу данных между приложением и хранилищем.

Интерфейс Query обеспечивает работу с запросами к базе знаний, контролирует управление различными параметрами запросов.

Интерфейс Manager обеспечивает работу с хранилищем, позволяя добавлять в него новые данные, обновлять их или изменять каким-либо образом. Здесь же обеспечивается возможность сохранять изменения в базе знаний.

Интерфейс Annotations предоставляет некоторые полезные аннотации для работы с базой знаний.

Разрабатываемый ORM фреймворк функционирует на основе HTTP REST SPARQL, что позволяет подключаться к большинству RDF репозиториев, без жесткой привязки к какому-нибудь одному репозиторию.

В разрабатываемом фреймворке можно считывать из онтологий классы, преобразуя их в соответствующие типы объектов используемого языка программирования с сохранением всех свойств, связей и значений.

Кроме того, можно самостоятельно создавать подобные типы в виде экземпляров класса.

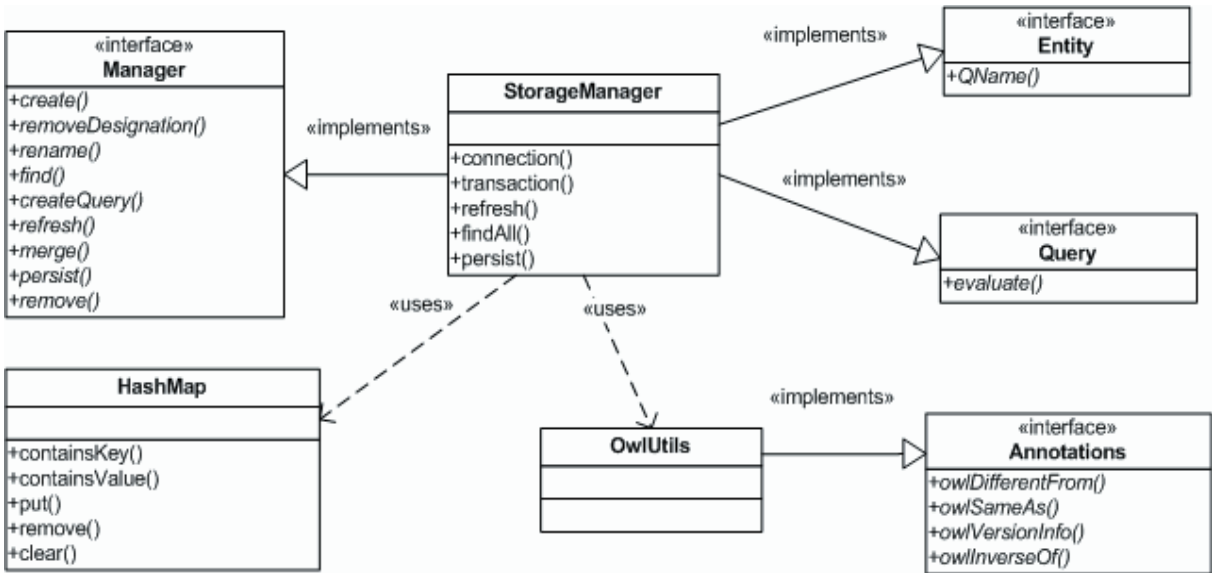

Рис. 1. Диаграмма классов

4. Создание примера класса с использованием ORM

Создание экземпляра класса «Статья» со свойствами «автор», «предметная область», «название»

```
@RDFNamespaces({
    "foaf = http://purl.org/dc/elements/1.1/",
    "individual = http://examples.com/core/"
})
@RDFBean("foaf:Article")
public class Article {
        private String title;
        private String creator;
        private String subject;
        @RDFSubject(prefix = "individual")
        @RDF("foaf:title")
            public String getTitle() { return title; }
            public void setTitle(String title) {
                this.title = title;
            }
        ...
}
```

Обращение к базе знаний может быть выполнено двумя методами: либо через формирование SPARQL запросов, либо через непосредственное обращение к данным хранящимся в онтологии по средствам объектов языка java.

Выборка объектов из онтологической базы знаний происходит путем преобразования объектного кода в SPARQL запрос. К примеру, мы хотим извлечь все статьи определенного автора опубликованные в 2012 году:

```
person.Article()
        .where("Year of publication", "2012")
        .find();
```

Таким образом, данный код преобразуется в SPARQL запрос, который при выполнении выбирает из базы знаний все статьи (articles), принадлежащие указанному автору (person), опубликованные в определенном году.

Результаты выборки заносятся в коллекцию объектов. Для этого создается объект определенного типа (например, типа Article) и из базы знаний извлекаются все свойства, которыми обладает объект и значения этих свойств. Кроме стандартных свойств, объект также хранит и неявные связи, выводимые на основе owl-свойств (к примеру, inverseOf). И по результатам поиска все объекты заносятся в коллекцию.

Для сохранения изменений или обновления состояния базы знаний используются методы persist() – для сохранения и refresh() – для обновления текущего состояния. Для удаления объектов из онтологии используется метод remove().

5. Выводы

В результате проведенного моделирования нового ORM фреймворка можно сделать вывод, что предложенный инструментарий значительно уменьшит семантический разрыв при преобразовании онтологических отношений в термины ООП. Данный фреймворк позволит избавить разработчика от привязки к определенному хранилищу и даст возможность самому определять хранилище, которое, по его мнению, является наиболее подходящим под ту или иную задачу. Дальнейшие исследования должны быть направлены на оценку производительности разрабатываемого фреймворка и поиск путей её повышения.

Литература

1. Шевченко, О.Ю. Модели распределенной онтологической базы знаний для интеллектуальных информационных систем [Текст] / О.Ю. Шевченко, О.Л. Шевченко // Журнал "Системи обробки інформації" – 2010. – №6. – С. 25 – 29.

2. Шевченко, О.Ю. Метод інтелектуалізації платформ для Cloud Computing [Текст] / О.Ю. Шевченко, О.Л. Шевченко // Східно-Європейський журнал передових технологій. – 2011. – № 3/12. – С. 66 – 70.
3. Taylor Cowan Binding Java Objects to RDF / Портал группы-участника W3C SemanticWeb. Режим доступа: www / URL: http://semanticweb.com/binding-java-objects-to-rdf_b10682. – 25.05.2012. – Загл. с экрана.
4. Semantic Object (Medata) Mapper / Официальная страница проекта Sommer. Режим доступа: www / URL: http://java.net/projekts/sommer. – 25.05.2012. – Загл. с экрана.
5. About Elmo / Портал посвященный Sesame, связанным с ним приложениям и разработкам. Режим доступа: www / URL: http://www.openrdf.org/doc/elmo/1.5. – 25.05.2012. – Загл. с экрана.
6. SemanticWeb Wiki Elmo / Портал посвященный SemanticWeb. Режим доступа: www / URL:http://semanticweb.org/wiki/Elmo. – 25.05.2012. – Загл. с экрана.
7. AliBaba 2.0-beta7 – Home / Портал посвященный Sesame и связанным с ним приложениям и разработкам. Режим доступа: www / URL: http://www.openrdf.org/doc/alibaba/2.0-beta7. – 25.05.2012. – Загл. с экрана.
8. RDF for Fun and Profit: Using Empire / Портал ClarkParsia для разработчиков. Режим доступа: www / URL: http://weblog.clarkparsia.com/2010/05/07/rgf-for-fun-and-profit-using-empire. – 25.05.2012. – Загл. с экрана.
9. JenaBean. A library for persisting java beans to RDF / Портал GoogleDevelopers для разработчиков. Режим доступа: www / URL: http://code.google.com/p/jenabean. – 25.05.2012. – Загл. с экрана.
10. RDFBean Reference Documentation / Официальный сайт документации по библиотеке RDFBean. Режим доступа: www / URL: http://source.mysema.com/static/rdfbean/1.4.8/reference/html. – 25.05.2012. – Загл. с экрана.
11. RDFBeans framework / Портал посвященный RDFBeans фреймворку. Режим доступа: www / URL: http://rdfbeavs.sourceforge.net. – 25.05.2012. – Загл. с экрана.
12. Topaz Mission and Goals / Официальный сайт проекта Topaz. Режим доступа: www / URL: http://www.topazproject.org. – 25.05.2012. – Загл. с экрана.

Гібридні нейронні мережі засновані на ідеї поєднання спайк-нейронних мереж та принципів нечіткої логіки. У статті пропонується архітектура самонавчанної фаззі-спайк-нейронної мережі на основі дискретних динамічних ланок другого порядку

Ключові слова: нечітка кластеризація, спайк, фаззі-спайк-нейронна мережа

Гибридные нейронные сети основаны на идее объединения спайк-нейронных сетей и принципов нечеткой логики. В статье предлагается архитектура самообучающейся фаззи-спайк-нейронной сети на основе дискретных динамических звеньев второго порядка

Ключевые слова: нечеткая кластеризация, спайк, фаззи-спайк-нейронные сети

The hybrid neural network based on the idea of combining spiking neural networks and the principles of fuzzy logic. The paper presents the architecture of self-learning fuzzy spiking neural network based on discrete second-order critically damped response units

Keywords: fuzzy clustering, spike, fuzzy spiking neural networks

УДК 004.8:004.032.26

СФСНМ НА ОСНОВІ ЛАНОК ДРУГОГО ПОРЯДКУ ДЛЯ НЕЧІТКОЇ КЛАСТЕРИЗАЦІЇ

Д.М. Малишева
Кафедра штучного інтелекту
Харківський національний університет радіоелектроніки
пр. Леніна, 14, м. Харків, Україна, 61166
Контактний тел.: +38 (050) 781-75-98
E-mail: darly.malysheva@gmail.com

1. Вступ

Серед розмаїття засобів обчислювального інтелекту задля обробки даних за умов відсутності апріорної інформації, самонавчанні фаззі-спайк-нейронні мережі (СФСНМ) [1] привертають дедалі більше уваги через те що вони ближче до моделей реальних нейронних систем, ніж штучні нейронні мережі попередніх поколінь, швидкіші та обчислювально потужніші за них. Крім того, СФСНМ виявили нову галузь, де спайк-нейронні мережі можуть бути успішно застосовані, а саме – нечітку кластеризацію. У даній роботі пропонується архітектура самонавчанної фаззі-спайк-нейронної мережі на основі дискретних динамічних ланок другого порядку для нечіткої кластеризації.