МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное АВТОНОМНОЕ образовательное учреждение высшего образования

**Национальный исследовательский ядерный университет «МИФИ»**



**Факультет Кибернетики и информационной безопасности**

**Кафедра кибернетики (№ 22)**

Направление подготовки 09.04.04 Программная инженерия

**Пояснительная записка**

к научно-исследовательской работе студента на тему:

Исследование и реализация методов описания объектно-реляционного отображения для объектов мульти-модельной базы данных

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Группа | | М16-504 | |  | | |
| Студент | |  | |  | Лазарев С.С. | |
|  | | (подпись) | | (ФИО) | | |
| Руководитель | |  | |  | проф. Вольфенгаген В.Э. | |
|  | | (подпись) | | (ФИО) | | |
| Научный консультант | |  | |  | Исмаилова Л.Ю. | |
|  | | (подпись) | | (ФИО) | | |
| Оценка руководителя |  | | Оценка консультанта | | |  |
|  | (0-15 баллов) | |  | | | (0-15 баллов) |
|  |  | |  | | |  |
| Оценка за оформление |  | | Оценка за структуру | | |  |
|  | (0-10 баллов) | |  | | | (0-10 баллов) |

**Москва 2016**

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«Национальный исследовательский ядерный университет «МИФИ»**



**Факультет кибернетики и информационной безопасности**

**КАФЕДРА КИБЕРНЕТИКИ**

**Задание на НИР**

Студенту гр. М16-504 \_\_\_\_\_\_\_Лазарев Сергей Сергеевич\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(группа) (фио)

**ТЕМА НИР**

**Разработка и реализация методов описания**

**объектно-реляционного отображения для объектов**

**мультимодельной базы данных**

**ЗАДАНИЕ**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| №  п/п | Содержание работы | Форма  отчетности | Срок исполне­ния | Отметка о выполнении  Дата, подпись рук. |
|  | **Аналитическая часть** |  |  |  |
|  | Подбор и анализ литературы и имеющегося программного обеспечения по теме работы. Анализ отобранных источников, подготовка библиографии. | Библиография, раздел ПЗ | 19.09.2016 |  |
|  | Анализ средств связи с реляционной БД для объектной системы (на основе системы ВКО). | Раздел ПЗ | 03.10.2016 |  |
|  | *Оформление расширенного содержания пояснительной записки (РСПЗ)* | Текст РСПЗ | 20.10.2016 |  |
|  | **Теоретическая часть** |  |  |  |
|  | Разработка алгоритмов представлений объектов реляционной базы и функций, осуществляющих интерфейс с реляционной БД. Разработка модели объектно-реляционного отображения, обеспечивающих создание однородной среды обработки объектов. | Модель, алгоритмы, рабочие материалы | 25.10.2016 |  |
|  | Разработка поддерживающих алгоритмов для архитектуры хранения объектов ВКО на основе отобранного класса моделей. Учет в модели набора СУБД для поддержки реализации (включить MySQL). Разработка алгоритиов загрузки объектов в БД и выгрузки набора объектов в заданном формате. | Модель, алгоритмы, рабочие материалы | 31.10.2016 |  |
|  | **Инженерная часть** |  |  |  |
| 3.1. | Проектирование средств специализации архитектуры БД для представления типовых случаев объект­но-реляционного отображения. | Модель, рабочие материалы | 04.11.2016 |  |
| 3.2. | Проектирование сервисных процедур, механизмов отладки и интерфейсов программиста | Модель, рабочие материалы | 07.11.2016 |  |
|  | **Технологическая и практическая часть** |  |  |  |
|  | Реализация средств специализации архитектуры БД для представления типовых случаев объект­но-реляционного отображения (не менее 5). Реализация сервисных процедур, механизмов отладки и интерфейсов программиста. | Программный код, листинги отладки | 14.11.2016 |  |
|  | Разработка представительного примера работы с БД на примере задачи синтеза документации для тестовой предметной области. | Интерфейсы, программный код | 28.11.2016 |  |
|  | Доработка и отладка. Пробная эксплуатация системы на данных разработанного примера. Изучение влияния конфигурации базы данных на эффективность работы системы. | Доработки программного кода | 05.12.2016 |  |
|  | Подготовка документации по ГОСТ и help-системы. Повторная пробная эксплуатация. | Документация; help-система | 12.12.2016 |  |
|  | *Оформление пояснительной записки (ПЗ) и иллюстративного материала для доклада*. | Текст ПЗ, презентация | 15.12.2016 |  |

**ЛИТЕРАТУРА**

|  |  |
| --- | --- |
|  | Вольфенгаген В.Э. *Методы и средства вычислений с объектами. Аппликативные вычислительные системы*. – М.: JurInfoR Ltd., АО “Центр ЮрИнфоР”, 2004. – xvi + 789 c. |
|  | Вольфенгаген В.Э. *Конструкции языков программирования*. *Приемы описания.* – М.: АО “Центр ЮрИнфоР”, 2001. –276 c. |
|  | Вольфенгаген В.Э., А.А. Борзяк, А.С. Доронин, Л.Ю. Исмаилова, С.В. Косиков, В.В. Навроцкий *Безопасность информационной системы в условиях возникновения семантической нестабильности* // Безопасность информационных технологий. - 2013. -- N 1, с. 90-93. URL: <http://www.pvti.ru/articles_36.htm> (дата обращения: 30.05.2013). |
|  | Исмаилова Л.Ю., Косиков С.В. *Парадигмы аппликативного программирования: от логики через редукцию к программированию. Раздел в книге: Парадигма функционального программирования*. / Под ред. к.т.н. Исмаиловой Л.Ю., Москва: АО «Центр ЮрИнфоР», 2012. - с. 60-94 |
|  | Ismailova LYu. *Applicative computations and applicative computational technologies.* Life Sci J 2014; 11(11):177-181] (ISSN:1097-8135). [http://www.lifesciencesite.com](http://www.lifesciencesite.com/) (дата обращения: 16.07.2014) |
|  | К. Дейт. Введение в системы баз данных (восьмое издание). - М., Вильямс, 2006. |

|  |  |
| --- | --- |
| Дата выдачи задания:    « » 2016г. | Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(проф. Вольфенгаген В.Э.)  (подпись) (фио)  Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_( \_\_Лазарев С.С.\_\_\_\_\_\_)  (подпись) (фио) |

**Реферат**

Пояснительная записка содержит 23 страницы, из них 4 рисунка, 1 таблица, 25 ссылок на источники.

**Ключевые слова: базы данных, мульти-модельные базы данных, объектно-реляционное отображение, ORM.**

*Целью данной работы является* исследование и реализация методов описания объектно-реляционного отображения для объектов мульти-модельной базы данных.

*В первом разделе* приводится обзор существующих мульти-модельных баз данных.

*Во втором разделе* приводится описание объектно-реляционного отображения и модели описания редактора БД.

*В третьем разделе* приводится описание разработанной системы и разработанных алгоритмов интерпретации описания редактора БД и приведены результаты тестирования.

Оглавление

[**Реферат** 4](#_Toc469923381)

[**Введение** 6](#_Toc469923382)

[**1.** **Базы данных** 7](#_Toc469923383)

[**1.1.** **Мульти-модельные БД.** 7](#_Toc469923384)

[**1.1.1.** **OrientDB и ArangoDB** 7](#_Toc469923385)

[**1.1.2.** **Couchbase и FoundationDB** 8](#_Toc469923386)

[**1.1.3.** **MarkLogic и CrateIO** 8](#_Toc469923387)

[**1.2.** **Цели и задачи НИР** 10](#_Toc469923388)

[**2.** **Разработка модели объектно-реляционного отображения** 11](#_Toc469923389)

[**2.1.** **Объектно-реляционное отображение** 11](#_Toc469923390)

[**2.1.1.** **Достоинства и недостатки ORM** 12](#_Toc469923391)

[**2.1.2.** **ORM в Java** 12](#_Toc469923392)

[**2.2.** **Модель задания описания редактора БД** 13](#_Toc469923393)

[**3.** **Разработка методов описания объектно-реляционного отображения** 15](#_Toc469923394)

[**3.1.** **Результаты проектирования** 15](#_Toc469923395)

[**3.2.** **Программная реализация** 17](#_Toc469923396)

[**3.3.** **Тестирование разработанного программного модуля** 19](#_Toc469923397)

[**Заключение** 22](#_Toc469923398)

[**Литература** 23](#_Toc469923399)

# **Введение**

В настоящее время изменяются представления о роли информации в жизни человека и общества. Она становится ресурсом развития информационного общества, что влечет за собой необходимость хранения больших объемов информации, в связи с чем возникает проблема, связанная с ее организацией и систематизацией.

В науке потребность в создании информационных систем связана с тем, что широкий доступ к необходимой информации дает возможность эффективно управлять информационными потоками. Именно поэтому актуальной задачей становится разработка междисциплинарных методик, позволяющих осуществлять аналитическую обработку информации, и поиск способов ее структурирования для эффективного хранения и обработки. Оптимальными представляются технологии баз данных (БД), которые позволяют создавать структурированные массивы данных, которые хранятся и которыми управляют с помощью компьютерных технологий, используемых для создания и функционирования эффективных информационных систем.

Системы управления базами данных (СУБД) играют одну из важных ролей в организации современных информационных систем. Тематика СУБД поистине безгранична.

Технологическая среда изменяется очень быстро, в связи с чем расширяются представления о сферах деятельности, в которых применимы базы данных. Растущие информационные потребности общества отчетливо выявляют ограничения существующих технологий СУБД, и задача исследовательского сообщества – устремить свои усилия на эти новые направления. Спектр возможностей и потребностей весьма широк – от теоретических изысканий в области создания новых моделей и алгоритмических основ до реализации прототипов совершенно новых, революционных систем.

# **Базы данных**

## **Мульти-модельные БД.**

Большинство СУБД организованы относительно одной модель данных, которая определяет, как данные могут быть организованы, как они хранятся и как ими можно управлять. Мульти-модельные базы данных созданы для поддержки нескольких моделей данных с помощью единого внутреннего интерфейса. Документы, графы, реляционные и ключ-значение модели являются примерами моделей данных которые могут поддерживаться мульти-модельной базой данных.

Основное отличие между доступными мульти-модельными базам данных связано с их архитектурой. Мульти-модельные БД могут поддерживать различные модели данных. С помощью многоуровневой архитектуры каждая модель данных обрабатывается свои собственным компонентом.

### **OrientDB и ArangoDB**

OrientDB – СУБД NoSQL с открытым исходным кодом написанным на языке Java. Она является мульти-модельной БД, поддерживающей графы, документы, модель «ключ-значение» и объектные модели, но отношения управляют как в графовых БД с прямыми подключениями между записями. Она поддерживает следующие схемы:

* Less (слабоструктурированные данные);
* Full (строго задает обязательные поля);
* Mixed (смешанная: обязательные + необязательные поля).

Данная СУБД обладает сильной системой профилирования безопасности, основанной на пользователях и ролях, и поддерживает запросы на языке SQL. OrientDB использует несколько способов индексации, основанных на B-дереве и хэш-индексе. Каждая запись имеет суррогатный ключ, который идентифицирует позицию записи внутри массива, связь между записями хранится либо как единственное значение позиции записи внутри ссылающегося домена или как B-деревья записей позиций, которые предоставляет возможность быстрого прохождения связей «один-ко-многим» и быстрое добавление и удаление новых связей.

СУБД ArangoDB предоставляет гибкие модели хранения документов, графов и данных в формате ключ-значение. Работа с базой осуществляется через SQL-подобный язык запросов AQL или через специальные расширения на языке JavaScript. Средства для хранения данных соответствуют требованиям ACID (атомарность, согласованность, изолированность, надежность), поддерживают транзакции и обеспечивают как горизонтальную, так и вертикальную масштабируемость. Управление СУБД может производиться через web-интерфейс или консольный клиент ArangoSH. Проект написан на языках C и JavaScript.

### **Couchbase и FoundationDB**

Couchbase Server, ранее извстный как Membase, является программным пакетом мульти-модельной документо-ориентированной NoSQL базой данных с открытым исходным кодом. Данный пакет оптимизирован для интерактивных приложений, которые могут обслуживать одновременно множество пользователей путем создания, хранения, поиска, агрегирования, обработки и представления данных. Используемая в Couchbase Server модель данных позволяет определять документы в формате JSON, снимая с разработчика необходимость определения схемы хранения. Запросы и индексация данных могут выполняться в соответствии с парадигмой MapReduce.

Каждый узел Couchbase состоит из службы данных, индекс службы, службы запросов и компоненты менеджера кластера. Начиная с версии 4.0, три службы могут быть распределены между узлами кластера, если в этом есть необходимость.

Менеджер кластера контролирует конфигурацию и поведение всех серверов в кластере Couchbase. Он настраивает и контролирует поведение между узлами похожее на управление потоками репликации и перебалансировки операций. Она также обеспечивает метрические агрегация и консенсусные функции для кластера, и RESTful интерфейс управления кластером. Менеджер кластера использует язык программирования Erlang и Open Telecom Platform.

FoundationDB был мульти-модель NoSQL базы данных с архитектурой распределения ресурсов. Продукт был разработан вокруг базы данных, которая называется «ядро», с дополнительными функциями, которые называются «слои». Ядро предоставляет упорядоченное хранилище ключ-значение с транзакциями. Транзакции могут считывать или записывать несколько ключей, хранящихся на любом компьютере в кластере полностью поддерживая свойства ACID. Операции используются для реализации различных моделей данных с помощью слоев.

### **MarkLogic и CrateIO**

Корпорация MarkLogic – это американская компания, которая разрабатывает и предоставлет базу данных MarkLogic. MarkLogic считается мульти-модельной NoSQL базой данных, в которой возможно хранить, управлять, искать JSON и XML-документы, а также семантические данные.

MarkLogic развилась от XML базы данных также изначально хранить документы в формате JSON и троек RDF, модель данных для семантики. В дополнение к наличию гибкой модели данных, MarkLogic использует распределенную, масштабируемую архитектуру, которая может обрабатывать сотни миллиардов документов и сотни терабайт данных. В отличие от других NoSQL баз данных, MarkLogic поддерживает согласованность ACID для транзакций, и сосредоточены на создании корпоративных функций в каждом выпуске, в том числе надежной модели безопасности, сертифицированной в соответствии с общими критериям, и корпоративного класса высокой доступности и аварийного восстановления. MarkLogic предназначен для работы на территории или в облаке на веб-сервисах Amazon.

CrateIO является полностью ориентированным на поиск документов хранилищем данных. CrateIO является открытым исходным кодом, написанный на Java и на основе архитектуры распределения ресурсов. CrateIO предназначена для высокой масштабируемости и включает в себя компоненты из Facebook Presto, Apache Lucene, Elasticsearch и Netty.

Язык CrateIO является SQL, но он использует документ-ориентированный подход баз данных типа NoSQL. CrateIO использует SQL-парсер от Facebook Presto, его собственный анализ запросов и распределенный двигатель запросов. Elasticsearch и Lucene используется для обнаружения транспортного протокола и кластера, а Netty для асинхронных управляемых событиями рамок сетевых приложений.

CrateIO включает в себя встроенный интерфейс администрирования. Интерфейс командной строки (Crate Shell - CraSh) позволяет производить интерактивные запросы. Клиент Python Crate является наиболее передовой и предлагает интеграции SQLAlchemy.

Таблица 1. Сравнение мульти-модельных БД

| БД | Ключ-значе-ние | SQL | Докумен-ты | Графы | Объекты | Текст | Пространствен-ные данные |
| --- | --- | --- | --- | --- | --- | --- | --- |
| OrientDB | + | + | + | + | + | + | + |
| ArangoDB | + | - | + | + | - | - | + |
| Couchbase | + | + | + | - | + | + | + |
| FoundationDB | + | + | + | + | - | - | - |
| CloudBoost.io | + | + | + | + | + | - | - |
| MarkLogic | + | + | + | + | - | + | - |
| CrateIO | + | + | + | - | + | - | - |

## **Цели и задачи НИР**

Научно-исследовательская работа заключается в исследовании методов описания объектно-реляционного отображения для объектов мульти-модельной базы данных:

* в качестве исследовательской цели: анализ существующих мульти-модельных баз данных;
* в качестве учебной цели: изучение вычислителя концептуальных описаний, способов реализации объектно-реляционного отображения;
* в качестве целей проектирования: реализация методов описания объектно-реляционного отображения.

Для достижения указанных целей необходимо решить следующие задачи:

* исследовательской задачи: исследование систем управления мульти-модельными базами данных;
* учебные задачи: разработка алгоритмов представлений объектов реляционной базы и функций, осуществляющих интерфейс с реляционной БД, модели объектно-реляционного отображения, обеспечивающих создание однородной среды обработки объектов.
* задачи проектирования: разработка поддерживающих алгоритмов для архитектуры хранения объектов ВКО на основе отобранного класса моделей.

# **Разработка модели объектно-реляционного отображения**

## **Объектно-реляционное отображение**

ORM – объектно-реляционное отображение – технология программирования, которая связывает базы данных с концепциями объектно-ориентированных языков программирования, создавая виртуальную объектную базу данных. ORM предоставляет набор объектных представлений сущностей базы данных для определённого языка программирования. Таким образом, программист работает только с объектной моделью сущностей, абстрагируясь от структуры базы данных, что в большинстве случаев позволяет не возвращаться к старому коду при изменении структуры базы данных. Посредством ORM можно частично абстрагироваться от самой базы данных и её структуры.

ORM используется для упрощения процесса сохранения объектов в реляционную базу данных и их извлечения, при этом ORM сама заботится о преобразовании данных между двумя несовместимыми состояниями. Большинство ORM-инструментов в значительной мере полагаются на метаданные базы данных и объектов, так что объектам ничего не нужно знать о структуре базы данных, а базе данных — ничего о том, как данные организованы в приложении. ORM обеспечивает полное разделение задач в хорошо спроектированных приложениях, при котором и база данных, и приложение могут работать с данными каждый в своей исходной форме.

Реляционные базы представляют данные в табличном формате, в то время как объектно-ориентированные языки представляют их как связанный граф объектов. Основные проблемы и несоответствия возникают во время сохранения этого графа объектов в реляционную базу или его загрузки:

* реляционная модель может быть намного детальнее, чем объектная;
* реляционные СУБД не имеют ничего похожего на наследование;
* в СУБД определен только один параметр для сравнения записей — первичный ключ. В то время как ООП предоставляет как проверку идентичности объектов, так и их равенства;
* для связи объектов СУБД использует понятие внешних ключей, в объектно-ориентированных языках связь между объектами может быть только однонаправленной. Если же нужно организовать двунаправленные отношения, то придется определить две однонаправленные ассоциации. Кроме того, нет возможности определить кратность отношения, глядя на модель предметной области;
* принцип доступа к данным в ООП сильно отличается от доступа к данным в БД. Для доступа к данным в ООП используются последовательные переходы от родительского объекта к свойствам дочерних элементов и инициализации объектов по необходимости. Такой подход считается не эффективным способом извлечения данных из реляционных баз данных. Как правило, количество запросов к БД должно быть сведено к минимуму, необходимые сущности должны по возможности загружаться сразу с использованием JOIN-ов.

### **Достоинства и недостатки ORM**

Использование ORM в проекте избавляет разработчика от необходимости работы с SQL и написания большого количества кода, часто однообразного и подверженного ошибкам. Весь генерируемый ORM код предположительно хорошо проверен и оптимизирован, поэтому не нужно в целом задумывается о его тестировании. Это несомненно является плюсом, но в тоже время не стоит забывать и о минусах. Основной из них — это потеря производительности. Это происходит потому, что большинство ORM предназначены для обработки широкого спектра сценариев использования данных, гораздо большего, чем любое отдельное приложение когда-либо сможет использовать.

Работа с БД посредством грамотно написанного SQL-кода будет намного эффективнее, но не стоит забывать и о таком параметре, как время — то, что с легкостью пишется с использованием ORM за неделю, можно реализовывать ни один месяц собственными усилиями. Кроме того, большинство современных ORM позволяют программисту при необходимости самому задавать код SQL-запросов. Без сомнений, для небольших проектов использование ORM будет куда более оправдано, чем разработка собственных библиотек для работы с БД.

Применение ORM для решения простых задач значительно усложнит процесс, а для решения слишком специфических может оказаться недостаточным. Ведь гибкость настройки необходимых SQL-запросов прямо зависит от возможности конкретной ORM-системы.

### **ORM в Java**

Hibernate — библиотека для языка программирования Java, предназначенная для решения задач объектно-реляционного отображения. Она представляет собой свободное программное обеспечение с открытым исходным кодом, распространяемое на условиях GNU Lesser General Public License. Данная библиотека предоставляет легкий в использовании каркас для отображения объектно-ориентированной модели данных в традиционные реляционные базы данных.

Целью Hibernate является освобождение разработчика от значительного объёма сравнительно низкоуровневого программирования по обеспечению хранения объектов в реляционной базе данных. Разработчик может использовать Hibernate как в процессе проектирования системы классов и таблиц, так и для работы с уже существующей базой данных.

Hibernate не только решает задачу связи классов Java с таблицами базы данных и типов данных Java с типами данных SQL, но и также предоставляет средства для автоматической генерации и обновления набора таблиц, построения запросов и обработки полученных данных и может значительно уменьшить время разработки, которое обычно тратится на ручное написание SQL- и JDBC-кода. Hibernate автоматизирует генерацию SQL-запросов и освобождает разработчика от ручной обработки результирующего набора данных и преобразования объектов, максимально облегчая перенос приложения на любые базы данных SQL.

Hibernate обеспечивает прозрачную поддержку сохранности данных для стандартных Java-объектов.

## **Модель задания описания редактора БД**

При разработке модели описания редактора необходимо учитывать ряд следующих свойств, упрощающих разработку редактора:

1. удобочитаемость;
2. понятность;
3. информативность;
4. упрощенность.

Учитывая все эти факты была разработана следующая модель описания редактора:

<База

сервер: «имя\_сервера»

база\_данных: «имя\_базы\_данных»

выполнить: [

<Запрос

имя: «имя\_запроса»

запрос: «запрос\_на\_языке\_SQL»

>

…

<Запрос

имя: «имя\_запроса»

запрос: «запрос\_на\_языке\_SQL»

>

]>

Как видно из представленной модели в описании учитываются основные параметры, которые используются при подключении к базе данных, а также стартовый набор необходимых запросов.

Модель представлена как своеобразный шаблон описания на ЯКО. Стоит определить используемые условные обозначения:

* <…> – объект;
* […] – список значений;
* База – имя объекта, описывающего основные аспекты работы с базой данных;
* сервер – ключевое слово, которое дает понять, что в данном месте необходимо указать имя сервера, на котором хранится необходимая база данных и к которой необходимо подключиться;
* база\_данных – в данном поле необходимо указать имя базы данных, хранящейся на сервере, из которой необходимо получить информацию;
* Запрос – имя объекта, являющегося свойством объекта «База», описывающего набор запросов, которые необходимо выполнить после подключения к базе данных;
* имя – имя исполняемого запроса;
* запрос – в данном поле прописывается запрос на языке SQL.

Стоит отметить ряд моментов, которые позволят облегчить дальнейшую работу с создаваемым редактором. Свойство «имя» объекта «Запрос» обязательно для указания, но не обязательно должно иметь значение. Однако для упрощения работы с редактором рекомендуется указывать значение данного параметра. В объекте «Запрос» необходимо указывать минимальный набор SQL-запросов, хотя бы тех, которые получают таблицы, хранимые в базе данных.

Как видно из написанного данная модель удовлетворяет всем четырем заявленным свойствам: оно легко читается, понятно любому пользователю, несет достаточную информацию о базе данных и довольно проста.

# **Разработка методов описания объектно-реляционного отображения**

## **Результаты проектирования**

Программный модуль разработан на языке Java на базе уже разработанного в системе «РОК» модуля «ВКО». Данная разработка велась в среде NetBeans 8.2.

Для функционирования системы необходимы следующие условия: установленная на рабочей машине JDK 7 или выше, возможность установление соединения системы с локальной или внешней базой данных (БД) (в зависимости от настроек).

Данная система предназначена для разработки систем, позволяющих обрабатывать хранящуюся в базе данных информация. Описание данных систем производится на языке ЯКО с использованием языка SQL и следующих система управления базами данных:

* MySQL;
* PostgreSQL;
* SQLite.

Результатом разработки является система, позволяющая манипулировать данными.

Основные параметры разрабатываемой системы задается пользователем. Результатом выполнения вычисления выводится пользователю на экран.

Архитектура системы приведена на рис.1.

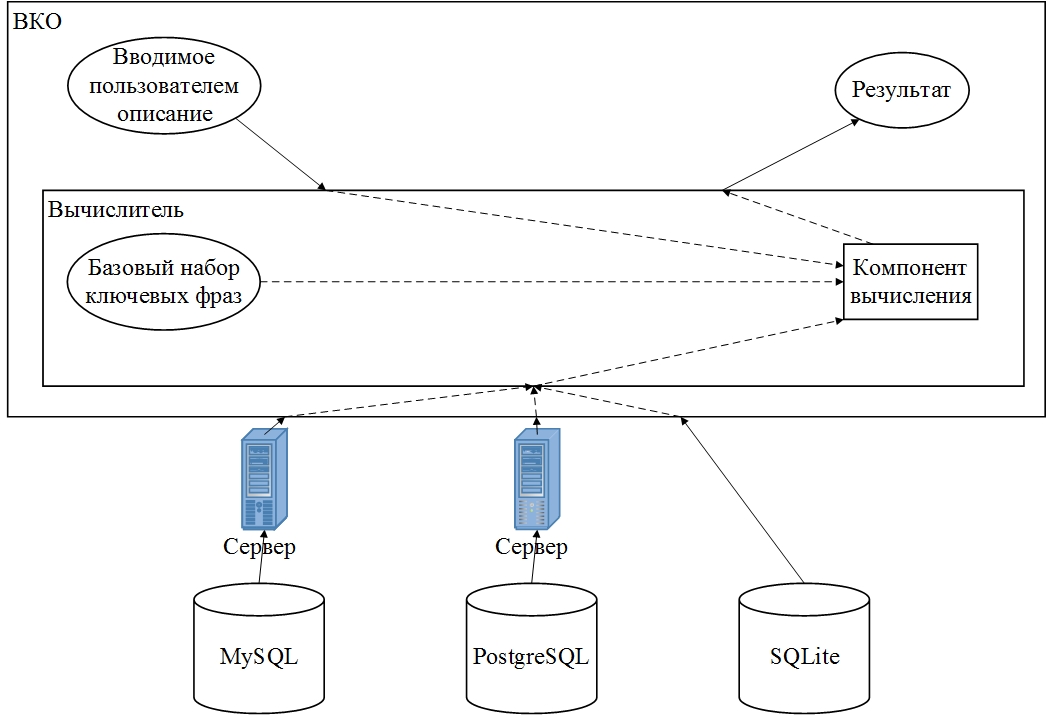


Рисунок 1. Архитектура

Система является дополнением разработанной в рамках системы «РОК» компонента «ВКО», предназначением которого является обработка описания некоторой системы, позволяющей манипулировать данными и разрабатываемой на декларативном языке ЯКО. В системе присутствует доступный базовый набор ключевых фраз, позволяющих создавать подобные системы по описанию:

* «База»

Объект, описывающий базу данных, к которой производится подключение, и минимальный набор исполняемых запросов: получение таблиц с их значениями из базы данных

* «сервер»

Свойство объекта «База», описывающий имя сервера, к которому необходимо подключиться. Указание данного свойства необязательно в случае использование SQLite.

* «база\_данных»

Свойство объекта «База», описывающий имя базы данных, из которой будут получаться данные.

* «выполнить»

Свойство объекта «База», описывающий список объектов «Запрос»

* «Запрос»

Объект, описывающий запрос к некоторой базе данных. В данном объекте рекомендуется описывать запрос на получение таблиц, хранящихся в базе данных.

* «имя»

Свойство объекта «Запрос», позволяющее задавать имя данному запросу, для упрощения навигации в создаваемой системе.

* «запрос»

Свойство объекта «Запрос», которое в качестве своего значения получает написанный разработчиком на SQL запрос.

Для нормального функционирования программы необходимо обеспечить её соединение с базой данных. Кроме связи с БД программа должна иметь возможность взаимодействовать с данными, хранящимися в ней.

При завершении интерпретации описания на выходе разработчик получает окно, содержащее различные вкладки. В данных вкладках располагаются данные полученные с помощью написанных запросов, а именем вкладки является значение, заданное свойству «имя» объекта «Запрос».

Разработчику при описании объекта «База» предоставляется возможность выбора СУБД с помощью набора элементов интерфейса, переключателей, которые могут находиться в одном из возможных состояний:

* MySQL;
* PostgreSQL;
* SQLite.

## **Программная реализация**

В рамках разработки был разработан набор классов, состоящий из двух классов, которые позволяют интерпретировать предоставляемое описание, подключиться к необходимой базе данных и построить необходимый интерфейс:

* MyDBConnector

Данный класс содержит в себе методы, которые необходимы для подключения к той или иной базе данных, используя передаваемые им:

* + имя сервера, на котором хранится база данных;
  + используемый диалект СУБД;
  + имя базы данных.

В случае использования диалекта SQLite имя сервера не указывается, так как база данных находится внутри проекта.

* DataBase

Данный класс содержит в себе набор функций, который позволяют построить пользовательский интерфейс и обработать предоставляемые разработчиком описание объекта «База» с его свойствами.

Класс DataBase включает в себя следующие функции:

* createNewDB

ListValue x JPanel

Данная функция предназначена для создания окна и соответственно для построения пользовательского интерфейса. С ее помощью задаются:

* имя сервера, на котором хранится база данных;
* имя базы данных, из которой получаются данные;

и выполняются описанные запросы, а также раздает задание остальным функциям, а именно createQuerryPanelMySQL, createQuerryPanelPSQL, createQuerryPanelMySQLL, а также конструктору класса MyBDConnector.

* createQuerryPanelMySQL

String x MyDBConnector 🡪 JScrollPane

Данная функция выполняется в том случае, если разработчик выбрал среди переключателей «MySQL».

* createQuerryPanelPSQL

String x MyDBConnector 🡪 JScrollPane

Данная функция выполняется в том случае, если разработчик выбрал среди переключателей «PostgreSQL».

* createQuerryPanelSQLL

String x MyDBConnector 🡪 JScrollPane

Данная функция выполняется в том случае, если разработчик выбрал среди переключателей «SQLite».

Посредству последних трех функций осуществляется создание вкладки с добавленным на нее скроллом. Используя данные функции на создаваемую панель помещается таблица с данными вместе с своей шапкой.

Стоит учесть тот факт, что при прокрутке страницы, если это необходимо, шапка остается видимой, что позволяет пользователю легче анализировать полученную информацию.

* MyBDConnector

String x String x String 🡪 MyDBConnector

Данная функция-конструктор осуществляет подключение к базе данных по передаваемым ей параметрам:

* имя сервера;
* диалект;
* имя базы данных.

Система реализована на языке Java, для работы с БД используется язык запросов SQL, диалектов СУБД: MySQL 5.5, PostgreSQL 9.6.1, SQLite 3.15.2.

## **Тестирование разработанного программного модуля**

Для проведения тестирования было созданы две базы данных в дополнение к уже существующей, что позволило провести тестирование для всех трех диалектов:

* MySQL;
* PostgreSQL;
* SQLite.

Также были подготовлены три описания на языке ЯКО с добавлением SQL для всех трех диалектов:

* MySQL

<База

сервер: "localhost"

база\_данных: "sheme"

выполнить: [

<запрос

имя: "objects"

запрос: "SELECT \* FROM objects">

<запрос

имя: "pragm"

запрос: "SELECT \* FROM pragm">

]>

* PostgreSQL

<База

сервер: "localhost"

база\_данных: "sheme"

выполнить: [

<запрос

имя: "object\_to\_object"

запрос: "SELECT \* FROM public.object\_to\_object">

<запрос

имя: "pragm"

запрос: "SELECT \* FROM public.pragm">

]>

* SQLite

<База

база\_данных: "sheme1"

выполнить: [

<запрос

имя: "object\_to\_object"

запрос: "SELECT \* FROM object\_to\_object">

<запрос

имя: "concept"

запрос: "SELECT \* FROM concept">

]>

В результате выполнения обработки данных описаний был получен пользовательский интерфейс с верным отображением результатов запросов (рисунки 2-4).

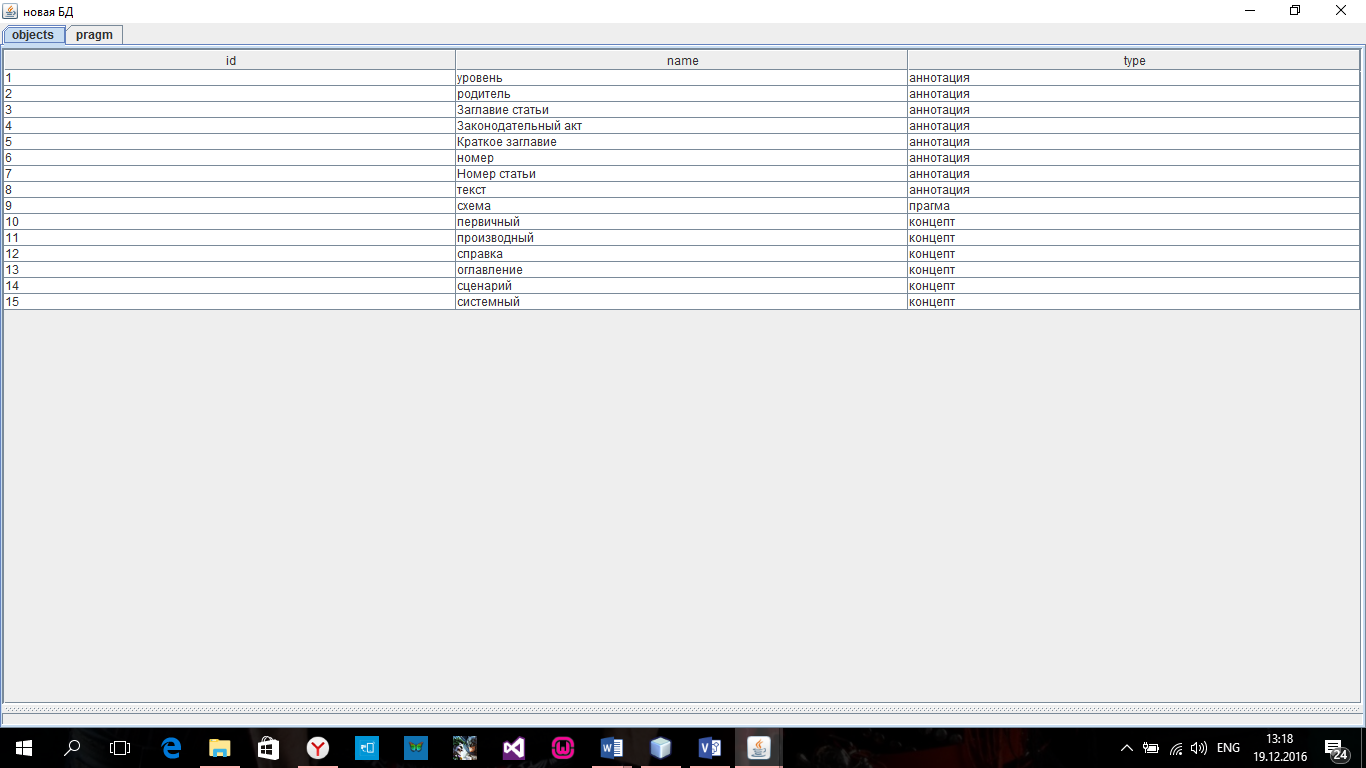


Рис. 2. Результат описания на диалекте MySQL

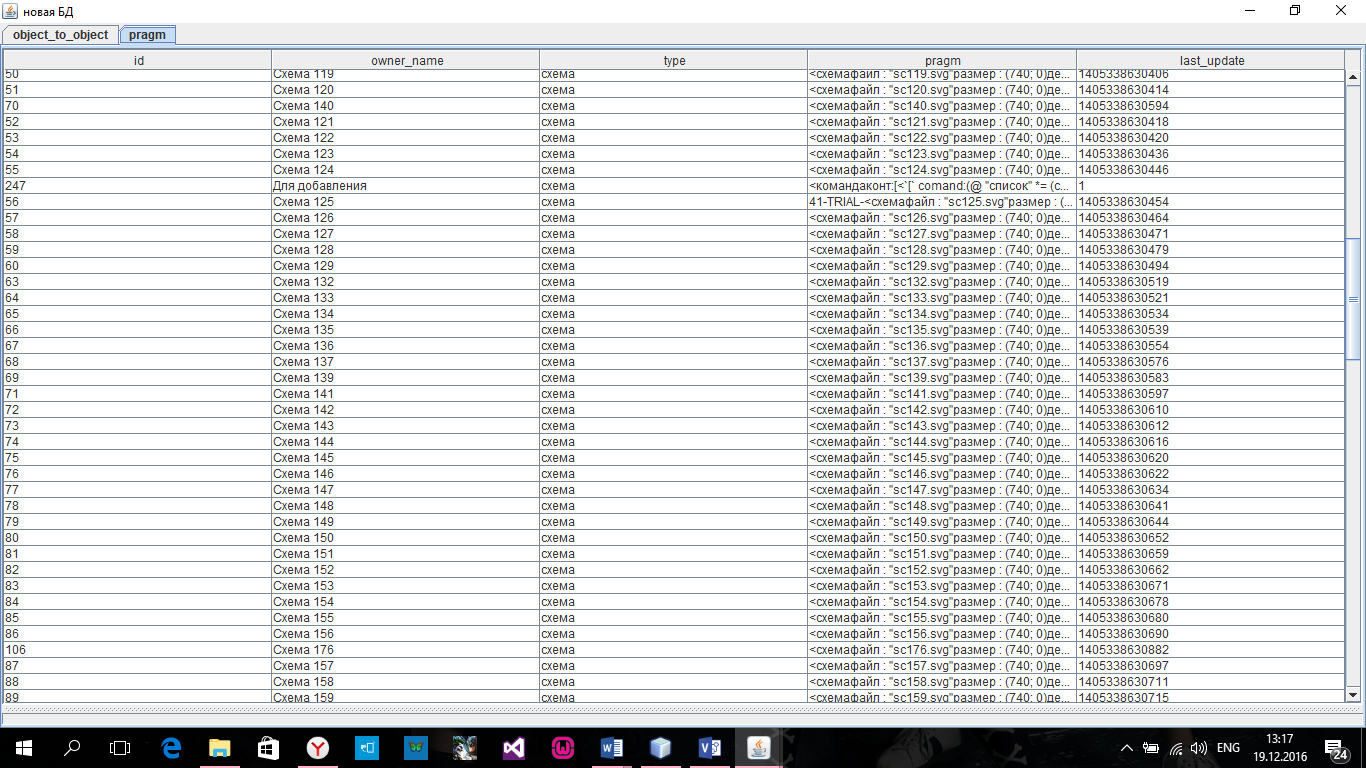


Рис. 3. Результат описания на диалекте PostgreSQL

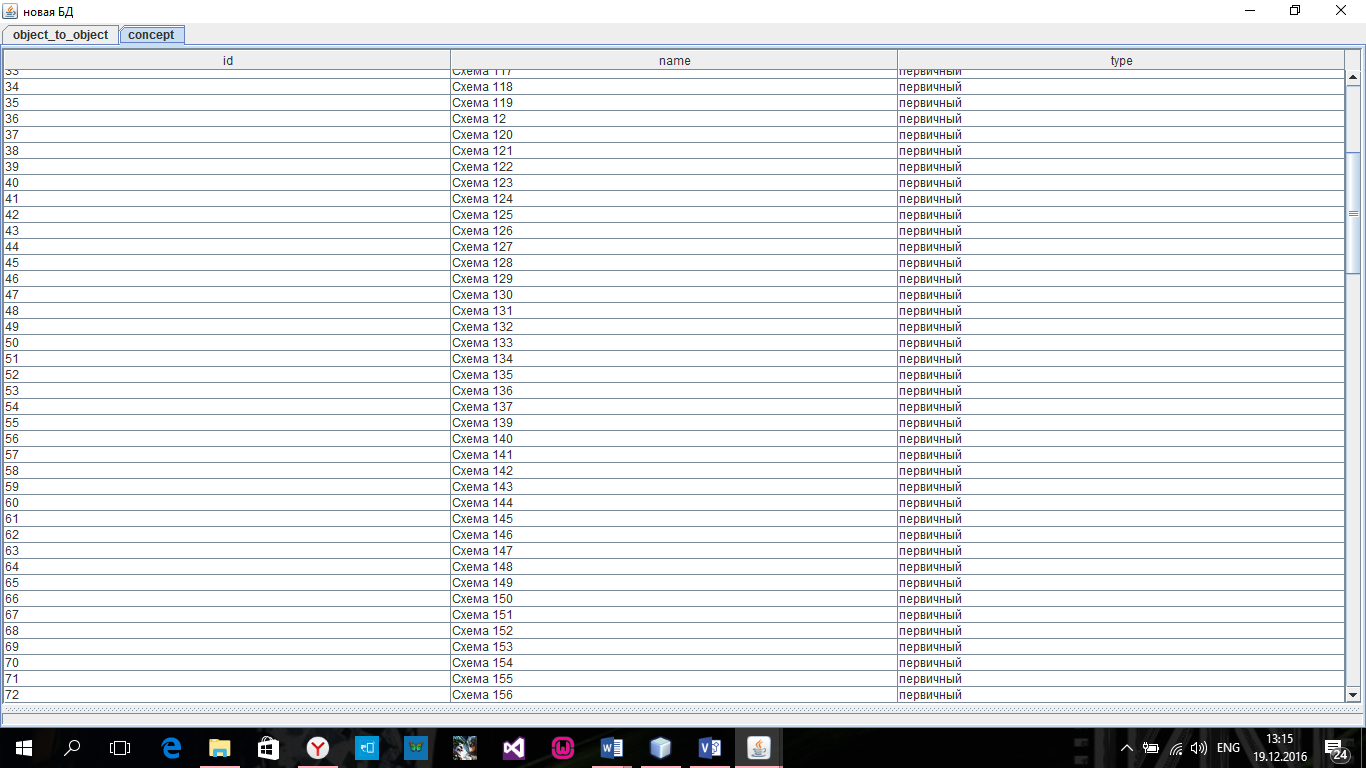


Рис. 4. Результат описания на диалекте SQLite

# **Заключение**

В ходе выполнения научно-исследовательской работы по теме разработка и реализация методов описания объектно-реляционного отображения для объектов мульти-модельной базы данных:

* была поставлена задача исследования систем управления мульти-модельными базами данных;
* был проведен анализ литературы;
* был проведен анализ выразительных возможностей выбранного языка;
* разработаны алгоритмы представлений объектов реляционной базы и функций, осуществляющих интерфейс с реляционной БД;
* разработан набор представительных примеров пространства данных;
* подготовлена пояснительная записка и графический материал.

# **Литература**

1. Вольфенгаген В.Э. Методы и средства вычислений с объектами. Аппликативные вычислительные системы. – М.: JurInfoR Ltd., АО “Центр ЮрИнфоР”, 2004. – xvi + 789 c.
2. Вольфенгаген В.Э. Конструкции языков программирования. Приемы описания. – М.: АО “Центр ЮрИнфоР”, 2001. –276 c.
3. Вольфенгаген В.Э., А.А. Борзяк, А.С. Доронин, Л.Ю. Исмаилова, С.В. Косиков, В.В. Навроцкий Безопасность информационной системы в условиях возникновения семантической нестабильности // Безопасность информационных технологий. - 2013. -- N 1, с. 90-93. URL: http://www.pvti.ru/articles\_36.htm (дата обращения: 30.05.2013).
4. Исмаилова Л.Ю., Косиков С.В. Парадигмы аппликативного программирования: от логики через редукцию к программированию. Раздел в книге: Парадигма функционального программирования. / Под ред. к.т.н. Исмаиловой Л.Ю., Москва: АО «Центр ЮрИнфоР», 2012. - с. 60-94
5. Ismailova LYu. Applicative computations and applicative computational technologies. Life Sci J 2014; 11(11):177-181] (ISSN:1097-8135). http://www.lifesciencesite.com (дата обращения: 16.07.2014)
6. К. Дейт. Введение в системы баз данных (восьмое издание). - М., Вильямс, 2006.
7. Multi-model database // Википедия – Реж. доступа: <https://en.wikipedia.org/wiki/Multi-model_database> – 20.09.2016
8. OrientDB // Википедия – Реж. доступа: <https://en.wikipedia.org/wiki/OrientDB> – 20.09.2016
9. Couchbase\_server // Википедия – Реж. доступа: <https://en.wikipedia.org/wiki/Couchbase_Server> – 20.09.2016
10. FoundationDB // Википедия – Реж. доступа: <https://en.wikipedia.org/wiki/FoundationDB> – 20.09.2016
11. MarkLogic // Википедия – Реж. доступа: <https://en.wikipedia.org/wiki/MarkLogic> – 20.09.2016
12. CrateIO // Википедия – Реж. доступа: <https://en.wikipedia.org/wiki/CrateIO> – 20.09.2016
13. ArangoDB // ArangoDB – Реж. доступа: <https://www.arangodb.com/why-arangodb> – 21.09.2016
14. NetBeans // NetBeans Docs & Support URL: <https://netbeans.org/kb/index.html>
15. Шилдт Г. Java. Полное руководство, 8-е изд.: Пер. с англ. – М.: ООО «И.Д. Вильямс», 2012. –1104 с.
16. Java Language and Virtual Machine Specifications URL:http://docs.oracle.com/javase/specs/ (05.03.2016)
17. Фленаган Д. Java в примерах, 2-е изд.: Пер. с англ. – СПб: Символ-Плюс, 2003. – 644с.
18. Буч Г., Рамбо Д., Якобсон И. Язык UML. Руководство пользователя. 2-е изд.: Пер. с англ. Мухин Н. – М.: ДМК Пресс, 2006. – 496 с.
19. Иванов Д. Ю., Новиков Ф. А. Основы моделирования на UML: Учеб. пособие. – СПб.: Изд-во Политехн. ун-та, 2010. – 249с.
20. Кузовкин А.В., Цыганов А.А., Щукин Б.А. Управление данными: учебник для студ. Высших учеб. Заведений – М.: Издательский центр «Академия», 2010. – 256 с.
21. Роберт Лафоре Структуры данных и алгоритмы в Java. Классика Computers Science. 2-е изд. — СПб.: Питер, 2013. — 704 с.: ил. — (серия «Классика Computer Science»)
22. Daniel Abadi The Beckman Report on Database Research / Daniel Abadi [и др.] // Communications Of The Acm - 2016 - VOL. 59 - NO. 2 - pp: 92-99
23. MySQL // MySQL :: MySQL Community Edition URL: http://www.mysql.com/products/community/ (от 18.10.16)
24. PostgreSQL // PostgreSQL :: PostgreSQL 9.6.1 Documentation URL: <https://www.postgresql.org/docs/9.6/static/release-9-6-1.html>
25. SQLite // SQLite :: SQLite Documentation URL: http://www.sqlite.org/docs.html