Введение

На сегодняшний день одной из основных проблем бизнес приложений является необходимость сохранения бизнес-данных, обеспечение быстрого и надежного доступа к ним, а также легкость обеспечения любых требуемых манипуляций с данными. Именно с этим связано высокая ценность разработчиков, способных создать программное средство, удовлетворяющее указанным требованиям. Огромное разнообразие требований заказчика их вариаций привело к возникновению чрезвычайно большого числа конкурирующих технологий обеспечения хранения, доступа и модификации данных. Каждая из них имеет свои преимущества и недостатки. Но не подлежит сомнению, что для работы с любой из этих технологий специалист должен обладать весьма высокой квалификацией. Тем более еще большие требования предъявляются к специалисту, работающему (или способному работать) сразу с несколькими технологиями. Данная работа предлагает альтернативное решение, значительно снижающее трудность обучения работе с технологией хранения, модификации и доступа к бизнес данным при этом сохраняя их гибкость.

Анализ предметной области

С появлением на рынке ПО различных баз данных стала очевидной необходимость обеспечения унифицированного интерфейса для взаимодействия с этими базами (создание, доступ, модификация, удаление). В качестве примера, иллюстрирующего необходимость такого интерфейса можно рассмотреть различия в семантике языков взаимодействия с базами данных. Каждая из них имеет свою специфику, поддерживает определенные возможности, которая не поддерживает другая. Это значительно усложняет и удорожает разработку приложений, взаимодействующих с базами данных. Кроме того переход с одной базы данных на другую в случае изменившихся требований заказчика или изменения функционала приложения становится неоправданно дорогим.

Отсюда вытекает требование обеспечивать совместимость с разными базами и, соответственно, простое добавление новой базы к уже поддерживаемым.

Разрабатываемая система взаимодействия с базой данных может быть использована в качестве backend’а для приложений автоматизированного контроля успеваемости (осуществления контрольных мероприятий) школьников/студентов. Данная система позволяет использовать различные виды баз данных для хранения сущностей необходимых для работы вышеуказанных приложений. При этом благодаря такой системе опытный пользователь приложения может выбирать желаемую базу данных для работы исходя из своих предпочтений, средств и требований к работе программы.

Описание ПО

Разрабатываемое ПО представляет собой интерфейс прикладного взаимодействия с любой из поддерживаемых баз данных и позволяет:

* создавать БД;
* открывать соединение с БД;
* загружать и сохранять сущности, требуемые заказчиком;
* обновлять схему базы данных;
* закрывать соединение;
* удалять БД.

При этом внешний разработчик, то-есть человек использующий разрабатываемую в данном проекте библиотека, должен полностью изолирован от проблем реализации взаимодействия с конкретной базой данных.

Разрабатываемой ПО представляет собой программную библиотеку (.jar файл), поставляемую заказчику, который разрабатывает программу на языке Java. Для получения возможности использовать эту библиотеку, заказчик должен подключить ее к своему проекту.

Функциональные требования

* Разрабатываемое ПО должно обеспечить общие функции для работы с базой данных. Любое стандартное взаимодействие с базой данных – сохранение, удаление, модификация данных должны быть реализовано максимально просто
* Работа с БД со стороны пользователя приложения должна быть унифицирована и не должна зависеть от типа используемой базы. Например, переход от работы с базой данных MySQL на базу данных H2 не должен приводить к каким-либо изменениям во внутреннем коду заказчика, либо эти изменения должны быть сведены к минимуму.
* Дополнение списка поддерживаемых баз данных не должно приводить к переписыванию архитектуры приложения.
* Конечный пользователь ожидает простое создание базы для сущностей, необходимых ему для своей работы. Стандартные функции взаимодействия с базой данных должны быть максимально простыми в использовании. Необходимо, чтобы дополнительные опции взаимодействия можно было ввести, но это не должно быть обязательным. Ожидается, что пользователю потребуется получение вышеуказанных сущностей из базы, и их добавление/сохранение.
* Прикладной интерфейс должен позволять определить тип используемой базы данных.

Нефункциональные требования

Применимость: данный прикладной интерфейс должен быть применим при написании приложений на языке программирования Java, осуществляющих автоматизированный контроль успеваемости школьников/студентов. Кроме того стоит отметить, что разрабатываемая библиотека может быть использована в любом приложении, которому требуется хранить и обрабатывать бизнес-данные

Надежность: при возникновении программных ошибок на стороне приложения должна обеспечиваться целостность данных в базе, а также осуществляться базовая обработка ошибок или оповещение приложений, использующих данное. Способ оповещения или тип оповещения должен однозначно идентифицировать возникающую ошибку. При этом приложение должно оставаться работоспособным.

Производительность: определяется производительностью используемой базы данных, а также количеством сущностей в базе.

При проектировании необходимо учитывать, что дальнейшее обновление системы не должно затрагивать уже существующий функционал заказчика, то-есть интерфейс взаимодействия приложения заказчика и базы данных может только расширятся.

Функциональная структура системы

API  
createDatabase()  
openDatabase()  
closeDatabase()

removeDatabase()  
loadData()  
saveData()

CMSSQL

…

CDatabase

CMySQL

Hibernate

Последовательность выполняемых действий

1. Реализовать настройку взаимодействия с БД с использованием Hibernate (автоматизированная настройка конфигурации)
2. Конкретизировать методы выполняющие действия, зависимые от БД
3. Реализовать простой механизм добавления новых сущностей
4. Реализовать простой механизм обновления таблиц в базах данных