Министерство образования и науки Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ Н.Г. ЧЕРНЫШЕВСКОГО»

Кафедра Математического и компьютерного моделирования

РАЗРАБОТКА ЭЛЕКТРОННОЙ БИБЛИОТЕКИ СРЕДСТВАМИ SQLITE И PYSIDE

Автореферат бакалаврской работы

студентки 5 курса 561 группы

направление 09.03.03 - Прикладная информатика

механико-математического факультета

Куницина Светлана Романовна

Научный руководитель
зав. кафедрой, д.ф. – м. н. Ю.А. Блинков

Зав. кафедрой
зав. кафедрой, д.ф. – м. н. Ю.А. Блинков

Саратов 2017

ВВЕДЕНИЕ

Qt – одна из ведущих платформ для разработки приложений с графическим пользовательским интерфейсом (GUI) под большинство существующих ныне операционных систем. Также Qt – одноименный набор библиотек, лежащий в основе платформы. Платформа развивается компанией Trolltech и ориентирована на язык программирования C++, однако, ввиду ее удобности и популярности, сторонними разработчиками создаются привязки библиотеки к некоторым иным объектно-ориентированным языкам.

В данном учебнике будет рассмотрена работа с PySide – привязкой библиотеки Qt к языку программирования Python. В этом сочетании нашли соединение многие положительные элементы как библиотеки, так и языка. Достаточно только сказать, что ООП-идеология Qt шире ОО возможностей языка C++, поэтому Qt представляет надстройку над синтаксисом C++. Отсюда следует необходимость использования специального препроцессора для Qt-программы, дабы получить компилирующийся C++ код. Язык же Python интерпретируемый, динамически типизируемый и легко расширяемый. PySide как раз и предоставляет соответствующее нуждам Qt расширение языка Python, которое позволяет ООП-идеологию Qt использовать в Python-е как «родную». С другой стороны, можно сказать даже больше: PySide расширяет ООП-идеологию библиотеки Qt высокоуровневыми возможностями языка Python.

Использование языка Python позволяет на порядок ускорить процесс разработки приложения для Qt по сравнению с альтернативной разработкой на C++. Достигается это за счет интерпретируемости Python-программы (то есть отсутствует цикл компиляции-сборки), также за счет более высокого уровня языка, что позволяет в несколько раз сокращать объем пишущегося кода.

Кроссплатформенность как библиотеки Qt, так и интерпретатора Python, позволяет переносить разработанные на PySide приложения из одной операционной системы в другую без каких-либо изменений.

Целью бакалаврской работы является разработка электронной библиотеки средствами SQLite и PySide.

1 Архитектура модель/представление в Qt

В Qt4 введен новый набор классов для просмотра элементов использующих архитектуру модель/представление для управления взаимосвязью между данными и методом их представления пользователю. Разделение функциональности, введенное в этой архитектуре, дает разработчикам большую гибкость в настройке отображения элементов и предоставляет стандартный интерфейс модели, позволяющий широкому диапазону источников данных использовать существующие представления элементов. В этом документе мы дадим краткое описание парадигмы модель/представление, выделим концепции и опишем архитектуру системы представления элементов. Каждый компонент данной архитектуры будет описан и будут предоставлены примеры, показывающие как использовать предоставленные классы.

Архитектура модель-представление-контроллер (Model-View-Controller, MVC) является шаблоном проектирования, берущим начало от Smalltalk, который часто используется для создания пользовательских интерфейсов. В книге Design Patterns Гаммы (Gamma) и других написано:

> «MVC состоит из трех типов объектов. Модель - объект приложения, представление - его экранное представление и контроллер - определяет реакцию пользовательского интерфейса на пользовательский ввод. До MVC при разработке пользовательского интерфейса эти объекты смешивались вместе. MVC разделяет их, для увеличения гибкости и возможности повторного использования.»

Если объединить объекты представления и контроллера, то в результате получится архитектура модель/представление. Это все еще отделяет способ хранения данных от способа их представления пользователю, но обеспечивает простую структуру, основанную на тех же принципах. Данное разделение дает возможность показать пользователю одни и те же данных в различных представлениях и реализовать новые типы представлений без изменения базовой структуры данных. Чтобы обеспечить гибкость управления пользовательским вводом, мы представляем концепцию делегата (delegate). Преимущество наличия делегата в этой структуре состоит в том, что это да-

ет возможность для настройки представления и редактирования элементов данных.

Модельные индексы дают представлениям и делегатам информацию о размещении элементов, предоставляемых моделью, и независимую от структуры данных.

Элементы определяются номерами строк и столбцов и модельным индексом их родительских элементов.

Модельные индексы создаются моделями по требованию других компонентов, таких как представления и делегаты. SQLite — легковесная встраиваемая реляционная база данных. Исходный код библиотеки передан в общественное достояние. В 2005 году проект получил награду Google-O'Reilly Open Source Awards.

Слово «встраиваемый» означает, что SQLite не использует парадигму клиент-сервер, то есть движок SQLite не является отдельно работающим процессом, с которым взаимодействует программа, а предоставляет библиотеку, с которой программа компонуется и движок становится составной частью программы. Таким образом, в качестве протокола обмена используются вызовы функций (API) библиотеки SQLite. Такой подход уменьшает накладные расходы, время отклика и упрощает программу. SQLite хранит всю базу данных (включая определения, таблицы, индексы и данные) в единственном стандартном файле на том компьютере, на котором исполняется программа. Простота реализации достигается за счёт того, что перед началом исполнения транзакции записи весь файл, хранящий базу данных, блокируется; ACID-функции достигаются в том числе за счёт создания файла журнала.

Несколько процессов или потоков могут одновременно без каких-либо проблем читать данные из одной базы. Запись в базу можно осуществить только в том случае, если никаких других запросов в данный момент не обслуживается; в противном случае попытка записи оканчивается неудачей, и в программу возвращается код ошибки. Другим вариантом развития событий является автоматическое повторение попыток записи в течение заданного интервала времени.

В Qt имеется три уровня в модуле QtSql:

1. Уровень драйверов

2. Программный уровень
3. Уровень пользовательского интерфейса

К уровню драйверов относятся классы для получения данных на физическом уровне, такие, как:

- QSqlDriver
- QSqlDriverCreator<T*>
- QSqlDriverCreatorBase,
- QSqlDriverPlugin
- QSqlResult

QSqlDriver является абстрактным базовым классом, предназначенный для доступа к специфичным БД. Важно, что класс не должен быть использован «прямо», взамен нужно/можно воспользоваться QSqlDatabase. Хотя, если вы хотите создать свой собственный драйвер SQL, то можете наследовать от QSqlDriver и реализовать чисто виртуальные, и нужные вам виртуальные функции.

QSqlDriverCreator — шаблонный класс, предоставляющий фабрику SQL драйвера для специфичного типа драйвера. Шаблонный параметр должен быть подклассом QSqlDriver.

QSqlCreatorBase — базовый класс для фабрик SQL драйверов, чтобы возвращать экземпляр специфичного поскласса класса QSqlDriver, который вы хотите предоставить, нужно «перефразировать» метод createObject().

QSqlDatabase несет ответственность за загрузку и управление плагинов драйверов баз данных. Когда база данных добавлена (это делается функцией QSqlDatabase::addDatabase()), необходимый плагин драйвера загружается (используя QSqlDriverPlugin). QSqlDriverPlugin предоставляет собой абстрактный базовый класс для пользовательских QSqlDriver плагинов.

QSqlResult сам говорит о себе (как и все Qt-шные классы), этот класс предоставляет абстрактный интерфейс для доступа к данным специфичных БД. С практической точки зрения мы будем использовать QSqlQuery вместо QSqlResult, поскольку QSqlQuery предоставляет обертку («обобщенную») для БД-специфичных реализации QSqlResult.

2 Реализация БД «Библиотека»

Рассмотрим деятельность библиотеки, которая занимается выдачей книг читателям. В библиотеки, как показанщ на рисунке 2.1 также предусмотрено бронирование книг по запросу.

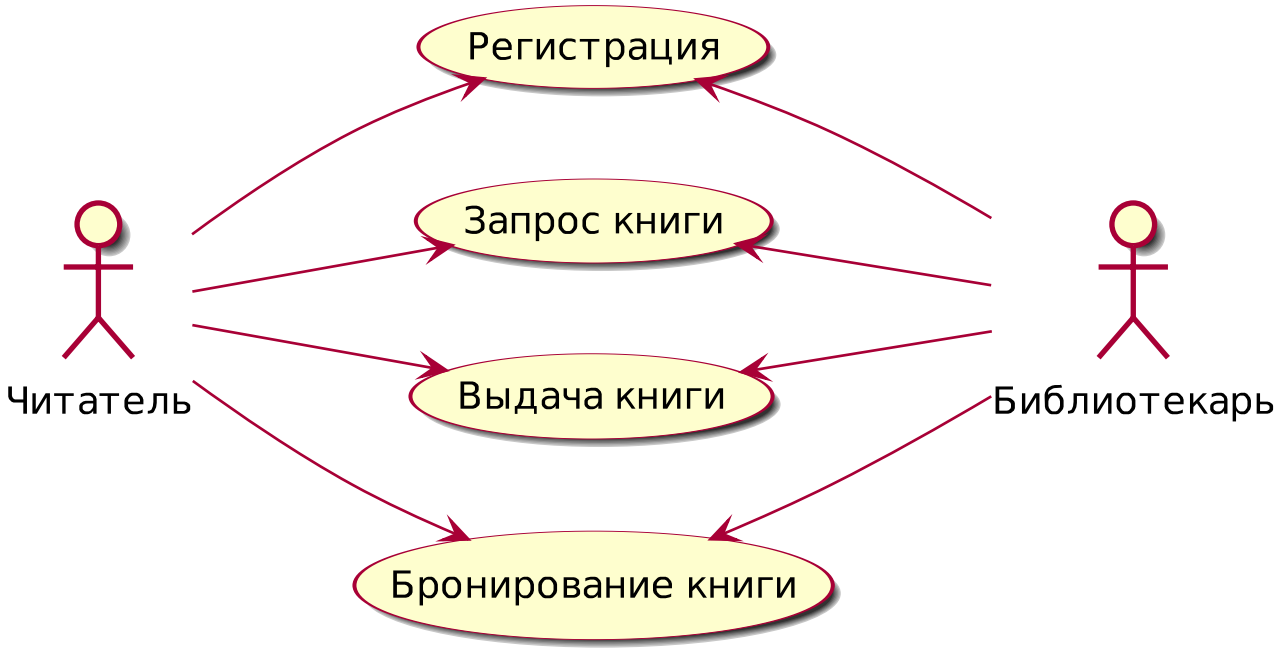

Рисунок 2.1 — Диаграмма основных прецедентов

Таблица «tblЧитатели» в соответствии с рисунком 2.2 содержит информацию о должностях.

tblЧитатели	
код	INTEGER PRIMARY KEY AUTOINCREMENT
номер_билета	TEXT UNIQUE
действителен	BOOLEAN DEFAULT TRUE
фамилия	TEXT
имя	TEXT
отчество	TEXT
телефон	TEXT
адрес	TEXT

Рисунок 2.2 — Таблица «tblЧитатели»

Таблица «tblГорода» в соответствии с рисунком 2.3 содержит информацию о должностях.

tblГорода	
код	INTEGER PRIMARY KEY AUTOINCREMENT
название	TEXT UNIQUE

Рисунок 2.3 — Таблица «tblГорода»

Таблица «tblИздательства» в соответствии с рисунком 2.4 содержит информацию о должностях.

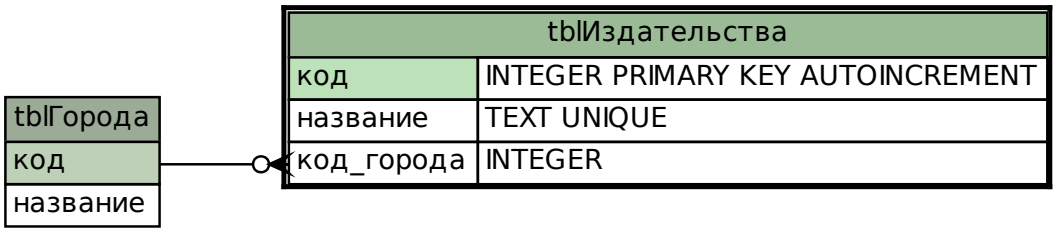

Рисунок 2.4 — Таблица «tblИздательства»

Таблица «tblИздание» в соответствии с рисунком 2.5 содержит информацию о должностях.

Рисунок 2.5 — Таблица «tblИздание»

Таблица «tblКниги» в соответствии с рисунком 2.6 содержит информацию о должностях.

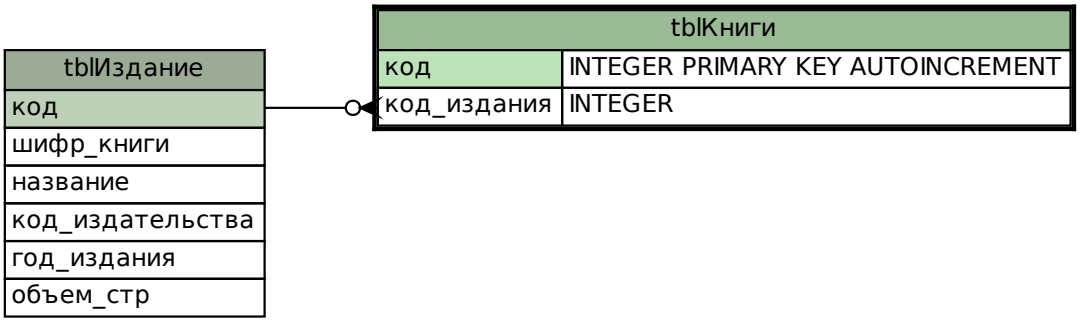

Рисунок 2.6 — Таблица «tblКниги»

Таблица «tblЖурнал» в соответствии с рисунком 2.7 содержит информацию о должностях.

Рисунок 2.7 — Таблица «tblЖурнал»

На рисунке 2.8 приведена полная ER (сущность – связь) диаграмма разработанной базы данных.

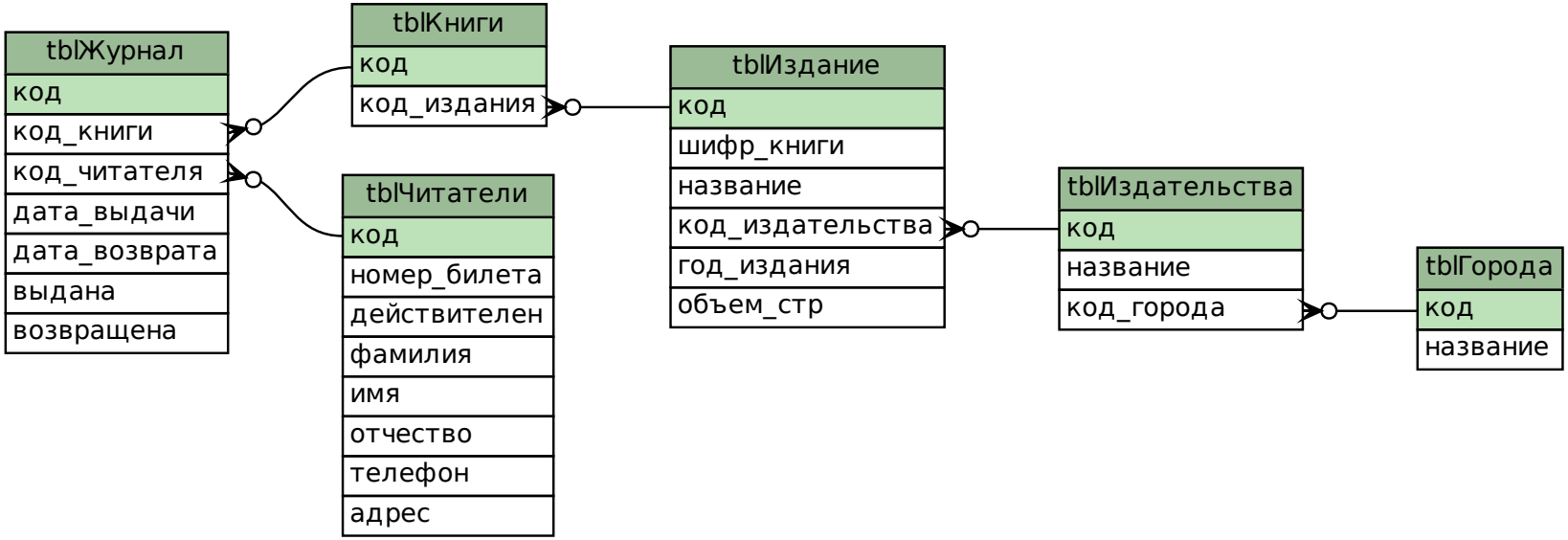

Рисунок 2.8 — Основная диаграмма

Правка таблицы «tblЧитатели» может быть произведена, как показано на рисунком 2.9.

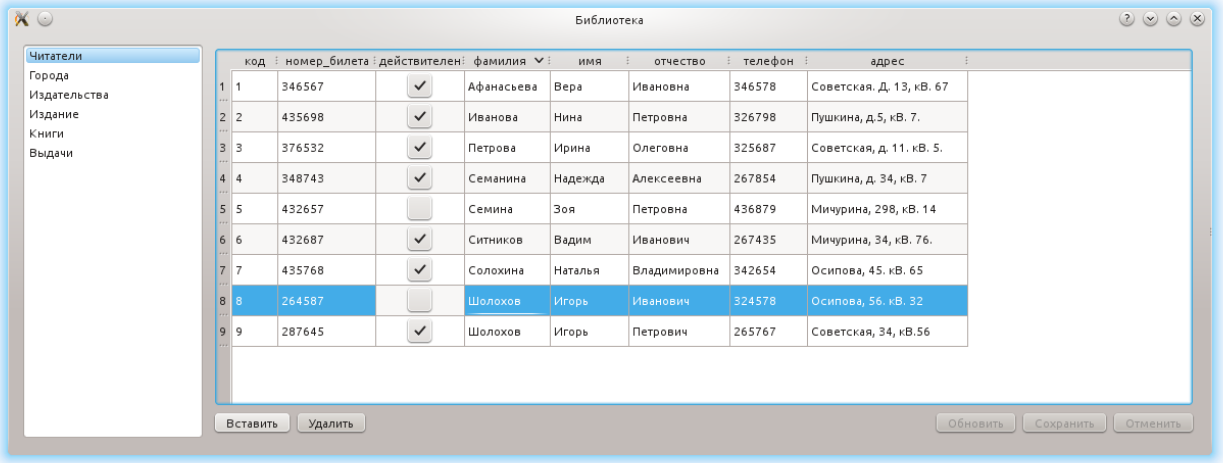

Рисунок 2.9 — Правка таблицы «tblЧитатели»

Правка таблицы «tblИздательства» может быть произведена, как показано на рисунком 2.10.

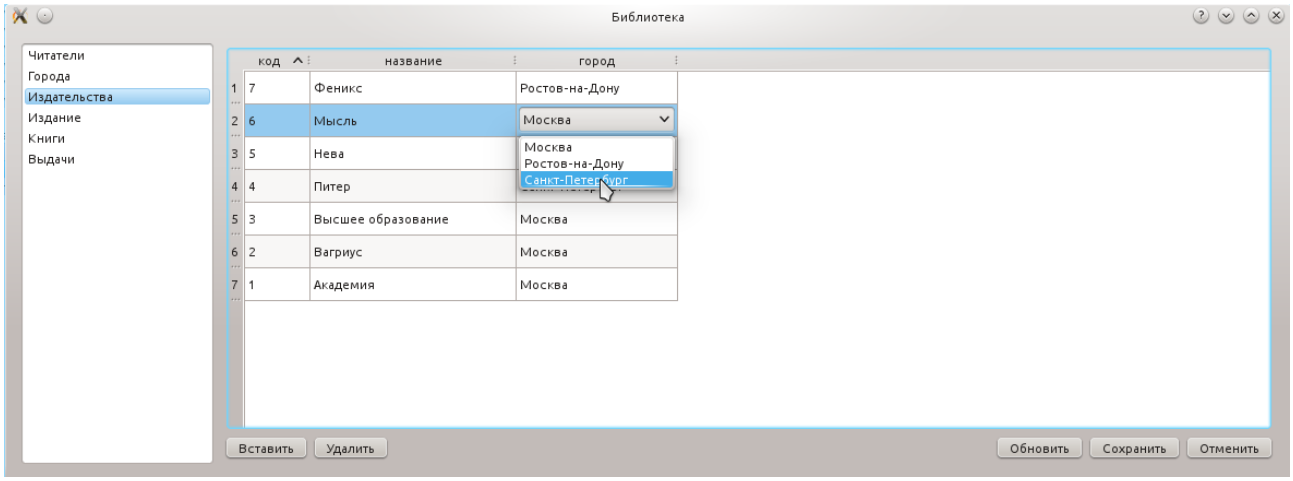

Рисунок 2.10 — Правка таблицы «tblИздательства»

Правка таблицы «tblИздание» может быть произведена, как показано на рисунком 2.11.

Правка таблицы «tblЖурнал» может быть произведена, как показано на рисунком 2.12.

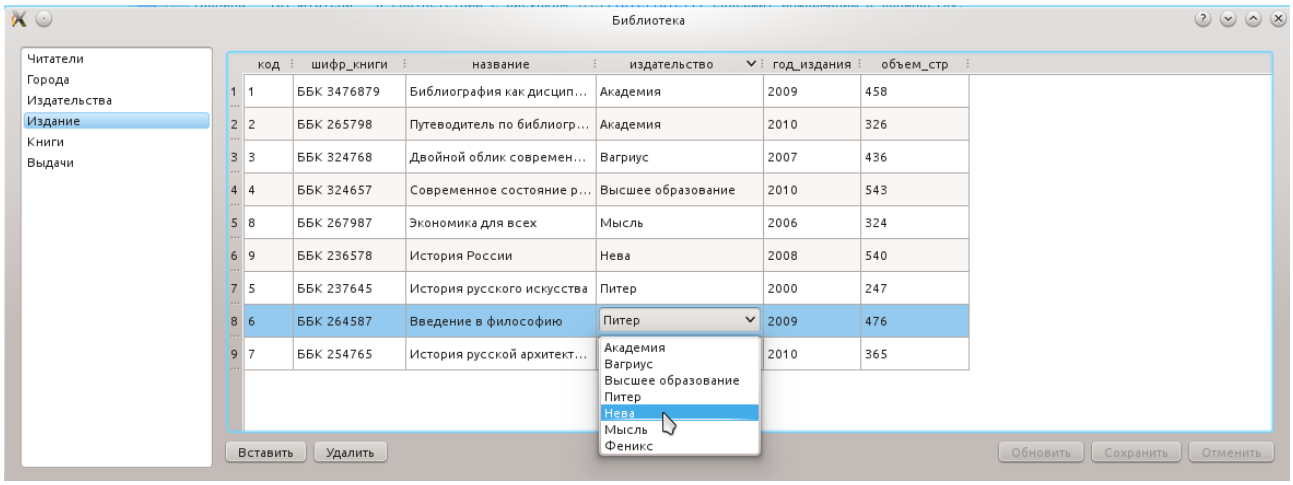

Рисунок 2.11 — Правка таблицы «tblИздание»

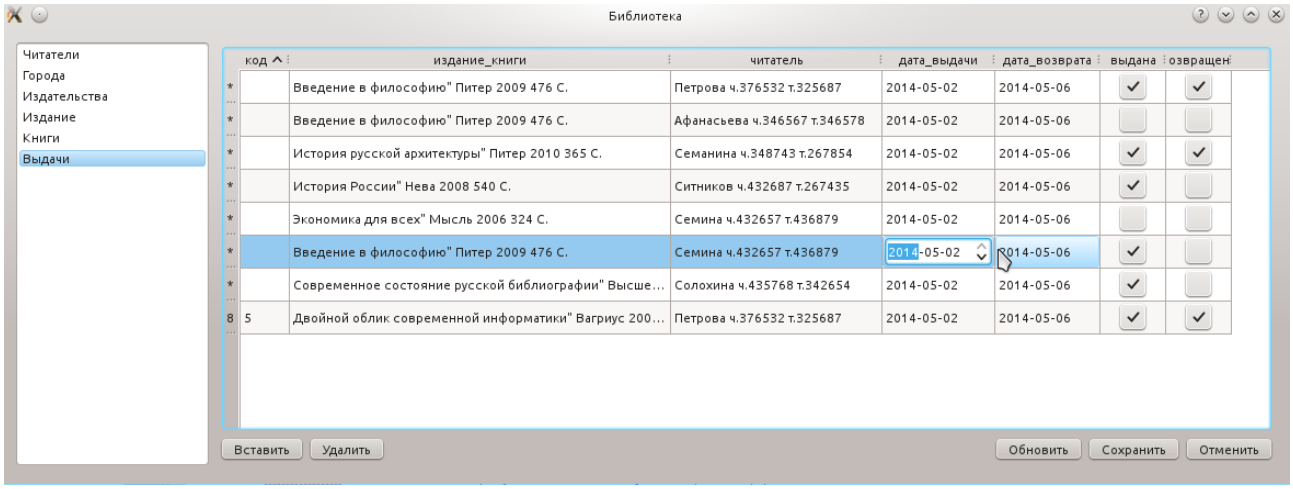

Рисунок 2.12 — Правка таблицы «tblЖурнал»

ЗАКЛЮЧЕНИЕ

В дипломной работе сделан анализ предметной области «библиотека». Для нее разработана структура БД в виде диаграмм сущность–связь и для СУБД SQLITE3 написан SQL для ее реализации. В качестве графического интерфейса пользователя для работы с информационной системой «библиотека» на PySide реализован интерфейс ко всем таблицам.