Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Московский государственный технический университет имени Н.Э. Баумана»
(МГТУ им. Н.Э. Баумана)

Курсовой проект по дисциплине “Базы данных”

Тема: “Настольное приложение базы данных автосалона”

Выполнил:
студент группы ИУ7-73
Юрочко Ю.В.

Руководитель:
Просуков Е.А.

Москва, 2013

Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования

«Московский государственный технический университет имени Н.Э. Баумана»
(МГТУ им. Н.Э. Баумана)

УТВЕРЖДАЮ
Заведующий кафедрой _____
_____(_____)
«_____»_____2013 г.

З А Д А Н И Е
на выполнение курсовой работы

по дисциплине «Базы данных»
Студент Юрочко Ю. В. гр. ИУ7-73
Руководитель Просуков Е. А.

График выполнения проекта: 25% к 4 нед., 50% к 8 нед., 75% к 11 нед., 100% к 14 нед.

1. Тема курсовой работы

«Настольное приложение базы данных автосалона»

2. Техническое задание

Разработать базу данных автосалона и настольное приложение для работы с базой данных, предназначенное для сотрудников автосалона. Базовый функционал приложения:

1. Презентация товара(каталог автомобилей и запчастей с возможностью поиска товара по различным критериям);
2. Возможность удаления и редактирования любой записи из каталогов;
3. Ведение статистики продаж автомобилей и запчастей;
4. Составление контрактов продаж;

3. Оформление курсовой работы

3.1. Расчетно-пояснительная записка на 32 листах формата А4;
3.2. Функциональные схемы IDEF0(1-3 схемы);
3.3 Схема IDEF1X (или эквивалентная ей ER - диаграмма);
3.4 Диаграмма потока данных (или контекстная диаграмма);
3.5 Архитектура приложения (схема);

Дата выдачи задания «09» сентября 2013 г.

Руководитель курсовой работы _____/ Просуков Е.А. /
Студент _____/ Юрочко Ю.В. /

Реферат

Объектом исследования и автоматизации является автосалон, а именно процесс подбора автомобилей(автозапчастей) продавцом-консультантом.

Цель работы – разработка настольного приложения, работающего с базой данных автосалона.

В процессе разработки проводился анализ задач, которые выполняются продавццами-консультантами, чтобы понять, какую именно функциональность требуется реализовать в приложении.

В результате анализа были выделены несколько главных функций, которые требуется реализовать в будующем приложении, например, поиск по каталогу, составление контракта продажи.

Созданное, в рамках этой работы, приложение можно использовать как тестовую версию в любом автосалоне. По мере тестирования приложения сотрудниками, можно расширять его, добавлять новые возможности, требуемые в конкретном автосалоне.

Содержание

Введение

В рамках данной работы создается настольное приложение, работающее под управление ОС Linux. Приложение представляет собой графический интерфейс для работы с базой данных PostgreSql и позволяет пользователю осуществлять поиск по содержимому базы данных, вставлять, удалять и модифицировать данные.

Приложение нацелено на использование сотрудниками автосалона. Оно поможет продавцам-консультантам быстро, эффективно и без временных затрат удовлетворить потребности клиентов в какой-то продукции, будь то автомобиль или автозапчасти.

С помощью данного приложения можно просматривать все продажи, которые были осуществлены салоном по автомобилям и автозапчастям, а также найти все продажи в какой-то конкретный день. Это может потребоваться, если, например, клиенту доставили не тот автомобиль, который должны были. Тогда можно найти этот автомобиль в списке продаж, просмотреть его характеристики и выяснить, действительно ли это не тот автомобиль, или же сам клиент что-то не понял. Это примитивный пример того, что ведения списка продаж действительно важно и может пригодиться в реальном автосалоне.

Еще одной возможностью приложения является то, что оно помогает продавцу составлять контракт продаж, для этого требуется только заполнить некоторые данные. Контракт сохранится в отдельную папку, содержащую все контракты, а проданная продукция, например, автомобиль, будет автоматически добавлена в список продаж.

1. Аналитическая часть

1.1 Описание предметной области

В настоящее время существует большое количество автосалонов, ообенно в крупных городах. Количество автомобилей, предлагаемых конкретным автосалоном зависит от марки продаваемых автомобилей, класса, к которому они относятся и других факторов. Цена также может сильно отличаться.

В самом автосалоне работет большое число работников. В данной работе больше всего интересны продавцы-консультанты, которые могут иметь некоторое деление, например по отдельным автомобилям, или по автомобилям и запчастям. Представители этого класса сотрудников должны знать все о продукции автосалона, а именно: продавец-консультант, специализирующийся на продаже автомобилей, должен знать все характеристики каждого автомобиля; консультант по продаже автозапчастей и других материалов для автомобилей должен знать все о каждой единице продукции. Эти знания необходимы для того, что любой человек, пришедший в автосалон, мог получить полную и всеохватывающую консультацию по интересующему его автомобилю или запчасти и приобрести ее. Если сами консультанты ничего не знают о продаваемых автосалоном автомобилях, то они ничего не смогут посоветовать клиенту.

Если автосалон продает большое число автомобилей, а у каждого автомобиля может быть несколько комплектаций, то, в итоге, продавцу-консультанту необходимо держать в голове огромный объем информации. Для того, чтобы избежать этого, необходимо иметь каталог, в котором будут отражены все имеющиеся автомобили и полные характеристики по каждому из них. Все, что будет описано в данном разделе о автомобилях, в такой же степени относится и к автозапчастям.

Если вы храните каталог в бумажном виде, а количество машин, продаваемых вашим автосалоном, велико, то в результате ваш каталог – многолистовая книга, работать с которой абсолютно неудобно. Для избежания этого необходимо иметь электронную версию каталога ничем не отличающуюся от бумажной (все данные абсолютно идентичны и не урезаны).

Современные автомобили могут иметь различные комплектации. Это означает, что один и тот же автомобиль может выпускаться с различными двигателями, отличаться встроенным набором опций, иметь различную внутренную отделку салона и другое. Поэтому необходимо, чтобы каждый автомобиль имел отдельную запись в каталоге, даже если он отличается от других автомобилей только лишь отсутствием

кондиционера, к примеру.

Сейчас, если вы покупаете новый автомобиль, то получаете не просто средство передвижения – вы получаете результат нынешнего научного прогресса. Современные машины оборудованы множеством датчиков, имеют кондиционер, навигацию, спутниковое телевидение, бортовой компьютер и многое другое. Все зависит от марки и класса покупаемого автомобиля. Для того, чтобы продавец-консультант знал все о характеристиках любого автомобился, необходимо отразить эти характеристики в каталоге. Таким образом, каталог должен иметь поля, описывающие все характеристики, которые присутствуют или отсутствуют в конкретном автомобиле.

Когда клиент приходит в автосалон, бывает такое, что он не уверен до конца какой автомобиль хочет приобрести. Для этого и нужен продавец-консультант, он имеет знания обо всех машинах в салоне и может посоветовать клиенту тот или иной вариант. Но для этого ему необходимо найти нужное авто в каталоге, чтобы ознакомить покупателя со всеми характеристиками и опциями, присутствующими в данной модели. Если число записей в каталоге большое, то на поиск уйдет большое количество времени. Поэтому необходимо иметь удобный и простой поиск, который смог бы искать автомобиль по марке, модели, году выпуску и другим атрибутам.

Финальной стадией в покупке новенького автомобиля является оформление всех документов. Эта скучная бумажная работа, без которой никак не обойтись. Но чтобы сделать этот процесс простым и быстрым, можно автоматизировать его, сделать так, чтобы отчеты составлялись автоматически с малейшим вмешательством продавца-консультанта.

Таким образом, если реализовать все вышеизложенные предложения, то работа в автосалоне станет более автоматизированной, а для продавцов-консультантов намного удобнее, быстрее и приятнее.

Каталоги с автомобилями, автозапчастями и другими деталями, которые поставляет автосалон, удобнее всего хранить в базе данных, а работу с базой данных реализовать в виде настольного приложения, которым будут пользоваться продавцы-консультанты.

1.2 Определение требований к структуре базы данных

Прежде чем начинать разработку автоматизированной системы автосалона с вышеизложенными возможностями, необходимо определить цели проектирования базы данных, тип и объем данных, с которым придется работать, способы их

использования, а также любые ограничения, налагаемые на эти данные.

При определении требований к системе также важно выяснить объем и тип информации, для которой предназначена база данных. Каков бы ни был текущий размер системы, необходимо определить объем данных, которым будет управлять система. При изучении объема данных следует определить их реальное количество и тенденцию роста.

При сборе сведений о требованиях к системе нужно определить круг пользователей, число пользователей, работающих с данными, и задачи, которые они собираются решать.

Также необходимо описать бизнес - правила, которым будут подчиняться данные в системе и, согласно которым, будет производится обработка и защита данных.

1.2.1 Опередеделение целей создания системы

После анализа предметной области можно сформулировать следующие цели создания автоматизированной системы:

1) Централизовать информацию о автомобилях, автозапчастях и продажах автосалона для более эффективного и удобного управления внутренними делами автосалона.

2) Поддерживать каталоги, с которыми оперируют работники автосалона, а именно: каталог автомобилей, каталог автозапчастей, каталоги продаж.

3) Предоставить возможность сотрудникам автосалона добавлять, удалять и изменять данные в каталогах.

4) Обеспечить удобный поиск по каталогам, имеющий разделение на поиск по различным характеристикам.

1.2.2 Определение объемов и типов данных

1) Проанализировав предметную область можно выделить следющие категории данных:

- Данные для входа;
- Краткая информация об автомобилях;
- Расширенная информация об автомобилях;
- Продажи автомобилей;
- Информация о запчастях ;

- Продажи запчастей;

2) Для каждой категории данных необходимо учитывать следующие сведения:

Категория	**Сведения**
Данные для входа	Логин, пароль, должность, имя, фамилия
Краткая информация об автомобилях	Марка, модель, год выпуска, мощность, цена, тип двигателя, наличие
Расширенная информация об автомобилях	Тип кузова, вид привода, тип руля, цвет кузова, состояние, наличие кондиционера, наличие телевидения, наличие навигации, расход топлива
Продажи автомобилей	Дата продажи, время продажи, информация о продавце, марка, модель, цена, полная информация о проданном авто, номер продажи
Информация о запчастях	Наименование, цена, фирма, тип, описание
Продажи запчастей	Дата продажи, время продажи, информация о продавце, цена, количество, полная информация о проданной детале

3) Примерный объем данных для каждой из категорий:

Данные для входа – количество данных этой категории равно количество сотрудников, работающих с системой. Примерно – 10.

Краткая информация об автомобилях – количество равно количеству продаваемых автосалоном автомобилей. Примерно – 100.

Расширенная информация об автомобилях – количество равно количеству записей в категории “Краткая информация об автомоилях” и является ее расширением. Примерно – 100.

Продажи автомобилей – количество равно числу проданных автосалоном автомобилей и определяется длительностью существования автосалона, поэтому по умолчанию принимаем за 100.

Информация о запчастях – количество равно предлагаемых салонов автозапчастей и расходных материалов. Примерно – 300.

Продажи запчастей – аналогично категории “Продажи автомобилей” - по умолчанию 100.

4) Тенденция роста

Тенденцию прироста числа в категории автомобилей и запчастей выявить трудно. Все зависит от выхода новых моделей автомобилей, ситуации на рынке и состояния экономики. Поэтому тенденцию для этих категорий не указываем.

Количество сотрудников автосалона будет расти с ростом автосалона, примерно – 2 сотрудника в год.

Количество продаж автомобилей и запчастей будет расти с каждым днем, ежегодно – увеличение, примерно, на 10%.

1.2.3 Определение способа использования данных

1) Категории пользователей: продавцы автомобилей, продавцы запчастей, администраторы системы.

2) В автосалоне не более 5 рабочих мест для продацов автосалона, поэтому потенциально в один момент времени с базой данных могут работать 5 человек.

Администратор системы в автосалоне один, он также может работать с базой данных, если это требуется.

3) Задачи, выполняемые каждой категорией пользователей:

Категория	Задачи
Продавцы автомобилей	Работа с каталогом автомобилей, поиск автомобилей в каталоге, изменение данных об автомобилях, добавление новых и удаление уществующих автомобилей, оформление заказов при покупке автомобиля
Продавцы автозапчастей	Работа с каталогом автозапчастей, поиск запчастей в каталоге, изменение данных о запчастях, добавление новых и удаление уществующих деталей, оформление заказов при покупке запчасти
Администраторы	Поддержание системы в работоспособном состоянии, устранение неполадок, установка

	системы на новые компьютеры, регистрация новых пользователей (выдача логина и пароля)

1.2.4 Определение бизнес - правил

- Данные для входа в систему выдаются администратором
- Данные для входа в систему содержат пароль и логин, которые выдаются строго сотруднику, на которого эти данные зарегистрированы.
- Данные для входа обязательно включают должность сотрудника, его имя и фамилию.
- Каждый сотрудник может занимать одну из следующих должностей: продавец автомобилей, продавец автозапчастей.
- Краткая информация об автомобиле включает марку авто, модель, год выпуска, мощность, цену, тип двигателя и наличие в автосалоне. Наличие всех перечисленных сведение обязательно.
- Год выпуска автомобиля не может превышать 2013 год и не может быть меньше 1920 года.
- Мощность автомобиля не может быть меньше 50 лошадиных сил и больше 1001.
- Цена автомобиля в краткой информации об автомобиле не может быть отрицательным числом.
- Заполнение полей наличия кондиционера, телевидения и навигации в расширенной информации об автомобиле не является обязательным.
- Расширенная информация об автомобиле обязательно включает в себя тип кузова, вид привода, тип руля, цвет, состояние автомобиля и расход топлива.
- Расход топлива в расширенной информации об автомобиле не может быть отрицательным числом.
- Список продаж автомобилей обязательно включает дату продажи, время продажи, информацию о продавце, марку авто, модель авто, цену, полную информацию об авто и номер продажи.
- Список продаж автозапчастей обязательно включает дату продажи, время продажи, информацию о продавце, цену, количество и полную информацию о запчасти.
- Информация о запчасти включает наименование, цену, фирму-производителя, вид запчасти, описание.
- Фирма-производитель, вид запчасти и описание в информации о запчасти могут отсутствовать.

- Описание в информации о запчасти содержит краткую характеристику конкретной детали.
- В каталоге могут быть автомобили одной марки или модели, а так же автомобили, отличающиеся друг от друга одной характеристикой. Аналогично с каталогом автозапчастей.
- Тип двигателя в краткой информации об автомобиле должен быть одним одним из следующего списка: инжектор, бензин-турбина, дизель, турбодизель, битурбо, компрессор.
- Тип кузова в расширенной информации об автомобиле должен быть одним из следующего списка: Седан, Кабриолет, Авант, Хетч-бек, Минивен, Купе, Джип.
- Привод в расширенной информации об автомобиле должен быть одним из следующего списка: Передний, Задний, Полный.
- Руль в расширенной информации об автомобиле должен быть либо Левый, либо Правый.
- Состояние автомобиля в расширенной информации об автомобиле должно быть одним из следующего списка: Отличное, Хорошее, Среднее, Битый.
- Продавец автомобилей может работать только с каталогом автомобилей, может модифицировать его.
- Продавец автозапчастей может работать только с каталогом автозапчастей, может модифицировать его.
- В заказе на некоторую запчасть указывается количество экземпляром, которое будет продано.

1.3 Разработка логической модели

1.3.1 Определение сущностей, связей между сущностями и атрибутов сущностей

В результате определения категорий данных и полученных бизнес - правил получаем следующие таблицы:

- data_enter – таблица данных для входа;
- general_car_info – краткие данные об автомобилях;
- full_car_info – расширенные данные об автомобилях;
- details_info – информация о запчастях;
- sales_car – список продаж автомобилей;
- sales_detail – список продаж автозапчастей;

Таблицы general_car_info и full_car_info имеют связь один к одному – один автомобиль имеет одно расширенное описание. Таблица full_car_info имеет внешний ключ, который ссылается на таблицу general_car_info.

Для определения столбцов таблиц обратимся к требованиям к системе, разработанным при анализе предметной области. Для каждой категории данных определена информация, которая входит в эту категорию. Эта информация определяет столбцы.

Таблица	Столбцы
data_enter	login, password, post, name, surname
general_car_info	id, firm, model, year, power, price, eng_type, present
full_car_info	id, id_gen_car, body_type, drive_type, rudder_type, color, condition_type, conditioner, tv, navigation, outgo
sales_car	id, data, time, seller_info, firm, model, price, id_full_info, sale_number
details_info	id, name, price, firm, type, info
sales_detail	id, data, time, tech_info, price, detail_count, id_info

1.3.2 Определение ограничений, накладываемых на данные

На базу данных накладываеются следующие ограничения:

Таблица data_enter:

- столбец login должен содержать значение и быть уникальным;
- столбец password должен содержать значение и быть уникальным;
- столбец post должен содержать значение одно из значение: seller (продавец автомобилей), tech(продавец автозапчастей);
- столбец name должен содержать значение;
- столбец surname должен содержать значение;

Таблица general_car_info:

- столбец id должен содержать значение; является первичным ключом с автоинкрементом;
- столбец firm должен содержать значение;
- столбец model должен содержать значение;
- столбец year должен содержать значение, которое не меньше 1920 и не больше 2013;

- столбец power должен содержать значение, которое не меньше 50 и не больше 1001;
- стоблец price должен содержать значение, которое является положительным числом.;
- столбец eng_type должен содержать значение из следующего списка: инжектор, бензин-турбина, дизель, турбодизель, битурбо, компрессор;
- столбец present должен содержать значение либо Да, либо Нет;

Таблица full_car_info:

- столбец id должен содержать значение; является первичным ключом с автоинкрементом;
- столбец id_gen_car должен содержать значение; является внешним ключом и ссылается на столбец id таблицы general_car_info;
- столбец body_type должен содержать значение из следующего списка: Седан, Кабриолет, Авант, Хетч-бек, Минивен, Купе, Джип;
- столбец drive_type должен содержать значение из следующего списка: Передний, Задний, Полный;
- столбец rudder_type должен содержать значение из следующего списка: Левый, Правый;
- столбец color должен содержать значение;
- столбец condition_type должен значение из следующего списка: Отличное, Хорошее, Среднее, Битый;
- столбец conditioner может не содержать значение;
- столбец tv может не содержать значение;
- столбец navigation может не содержать значение;
- столбец outgo должен содержать значение, которое является положительным числом;

Таблица details_info:

- столбец id должен содержать значение; является первичным ключом с автоинкрементом;
- столбец name должен содержать значение;
- столбец price должен содержать значение, которое является положительным числом;
- столбец firm может не содержать значение;
- столбец type может не содержать значение;
- столбец info может не содержать значение;

Таблица sales_car:

- столбец id должен содержать значение; является первичным ключом с автоинкрементом;
- столбец data должен содержать значение;
- столбец time должен содержать значение;
- столбец seller_info должен содержать значение;
- столбец firm должен иметь значение;
- столбец model должен содержать значение;
- столбец price должен содержать значение, которое является положительным числом;
- столбец id_full_info должен содержать значение; это значение равно значению столбца id таблицы full_car_info проданного автомобиля;
- столбец sale_number должен содержать значение;

Таблица sales_detail:

- столбец id должен содержать значение; является первичным ключом с автоинкрементом;
- столбец data должен содержать значение;
- столбец time должен содержать значение;
- столбец tech_info должен содержать значение;
- столбец price должен содержать значение, которое является положительным числом;
- столбец detail_count должен содержать значение, которое является положительным числом;
- столбец id_info должен содержать значение; это значение равно значению столбца id таблицы details_info проданной детали;

1.4 Графические материалы

1.4.1 Схема базы данных

На рисунке 1 представлена ER-диаграмма разрабатываемой базы данных.

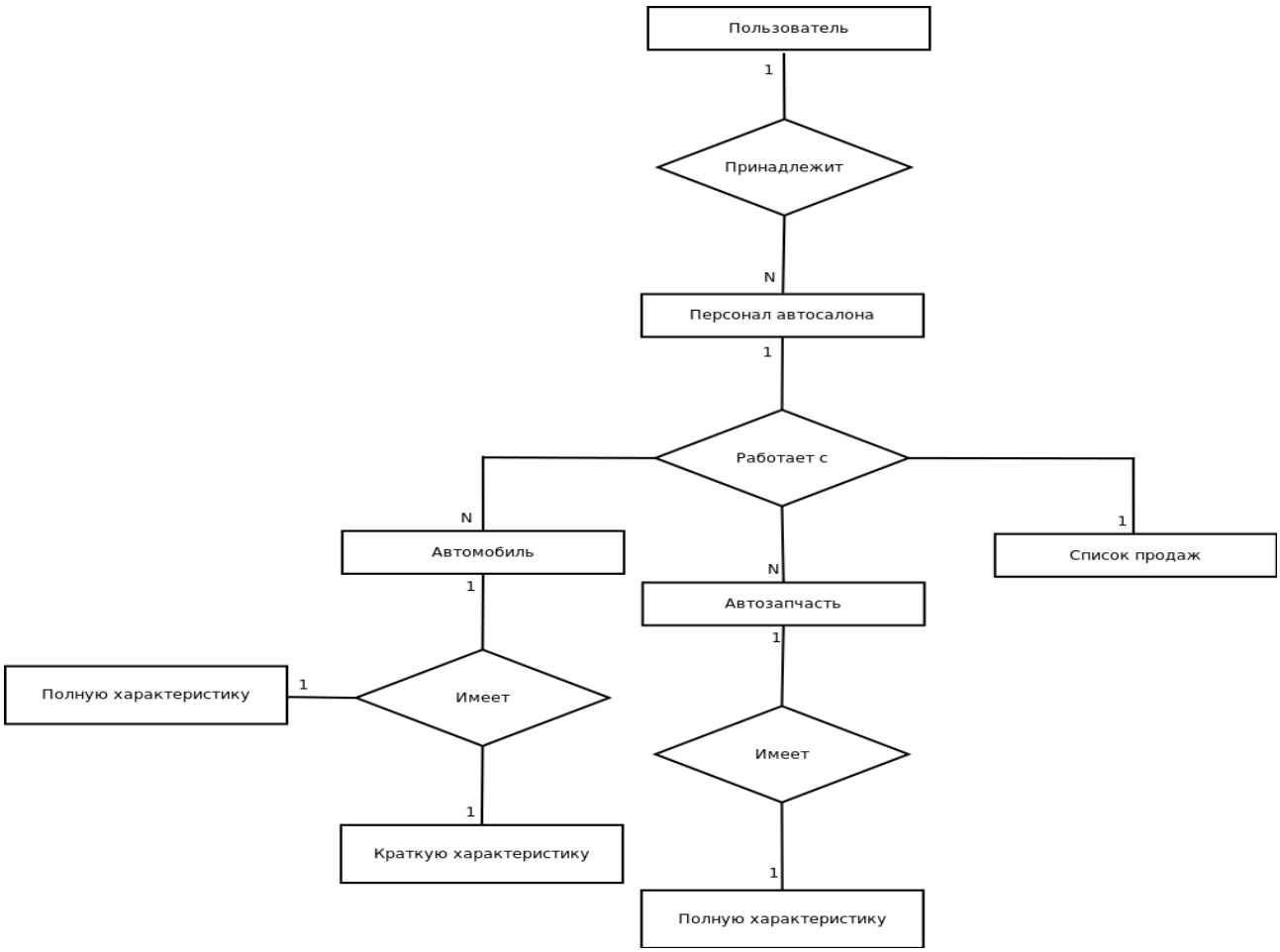


Рисунок 1. ER-диаграмма базы данных

1.4.2 Функциональная схема IDEF0

На рисунке 2 представлена общая схема IDEF0 (A0).

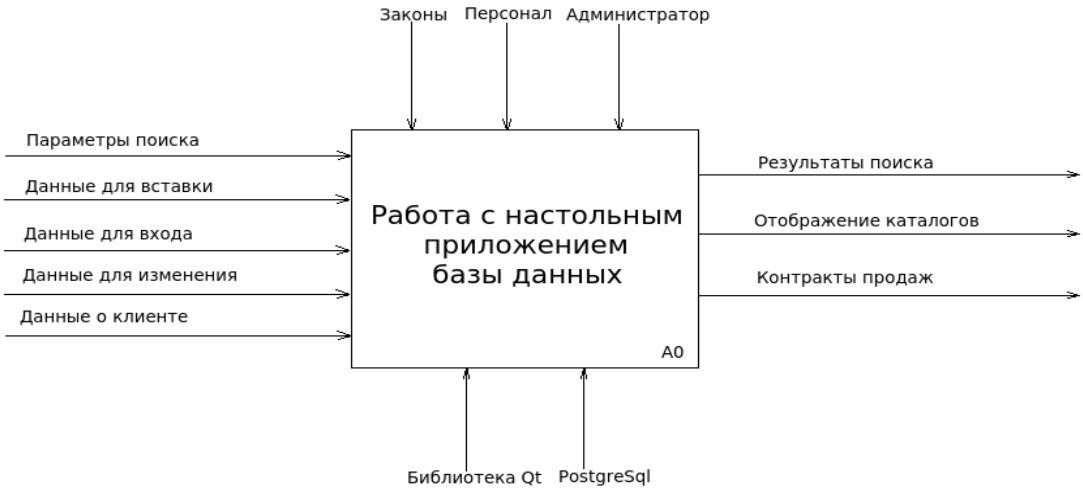


Рисунок 2. IDEF0 (Блок A0)

На рисунке 3 приводится детализация A0, отображается основная концепция работы приложения в графическом виде.

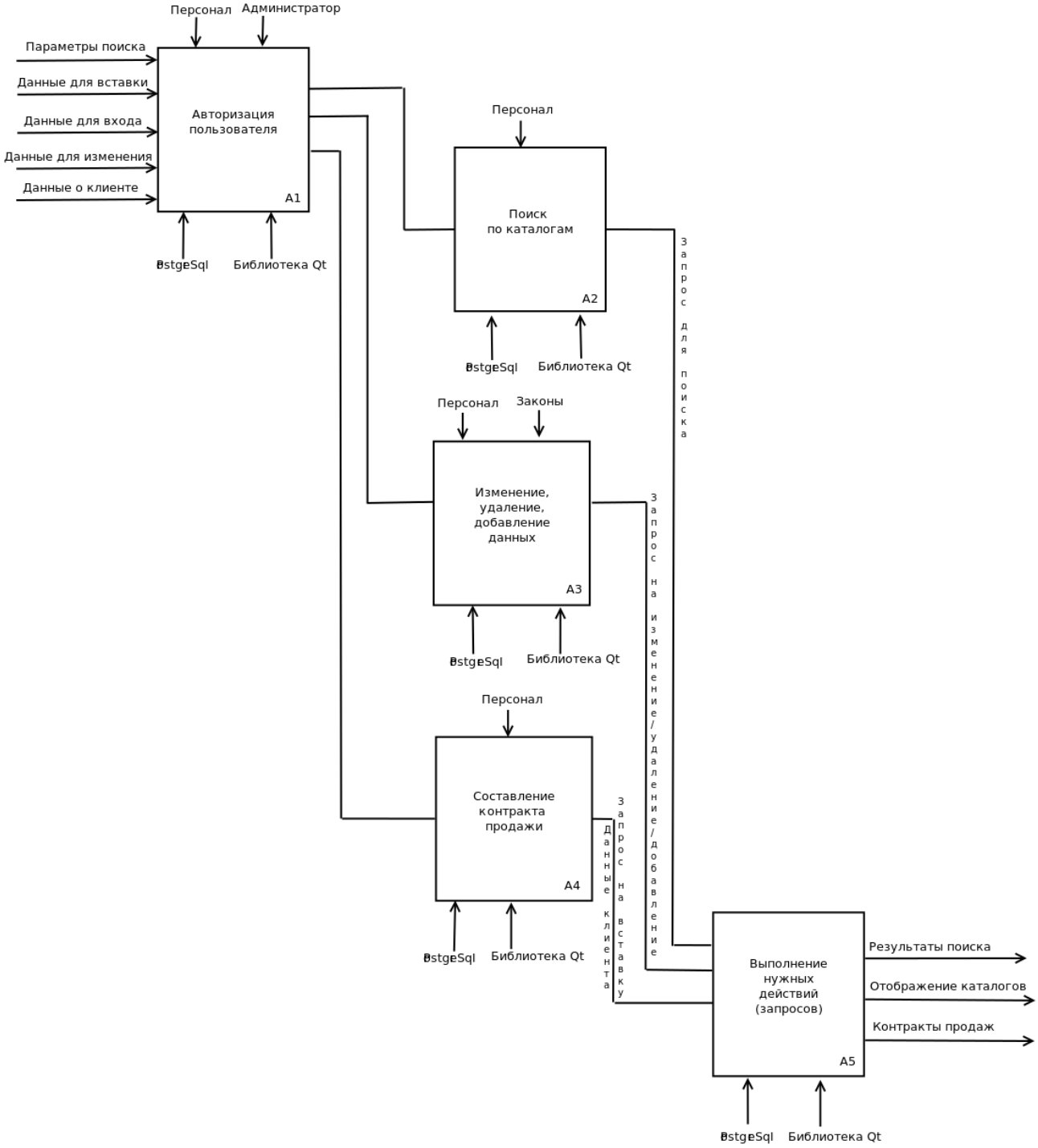


Рисунок 3. Детализация блока A0

В технологической части будет подробно разобрана работа каждого блока. Блоку A1 в этой части уделяется меньшее внимание, поэтому на рисунке 4 представлена детализация данного блока.

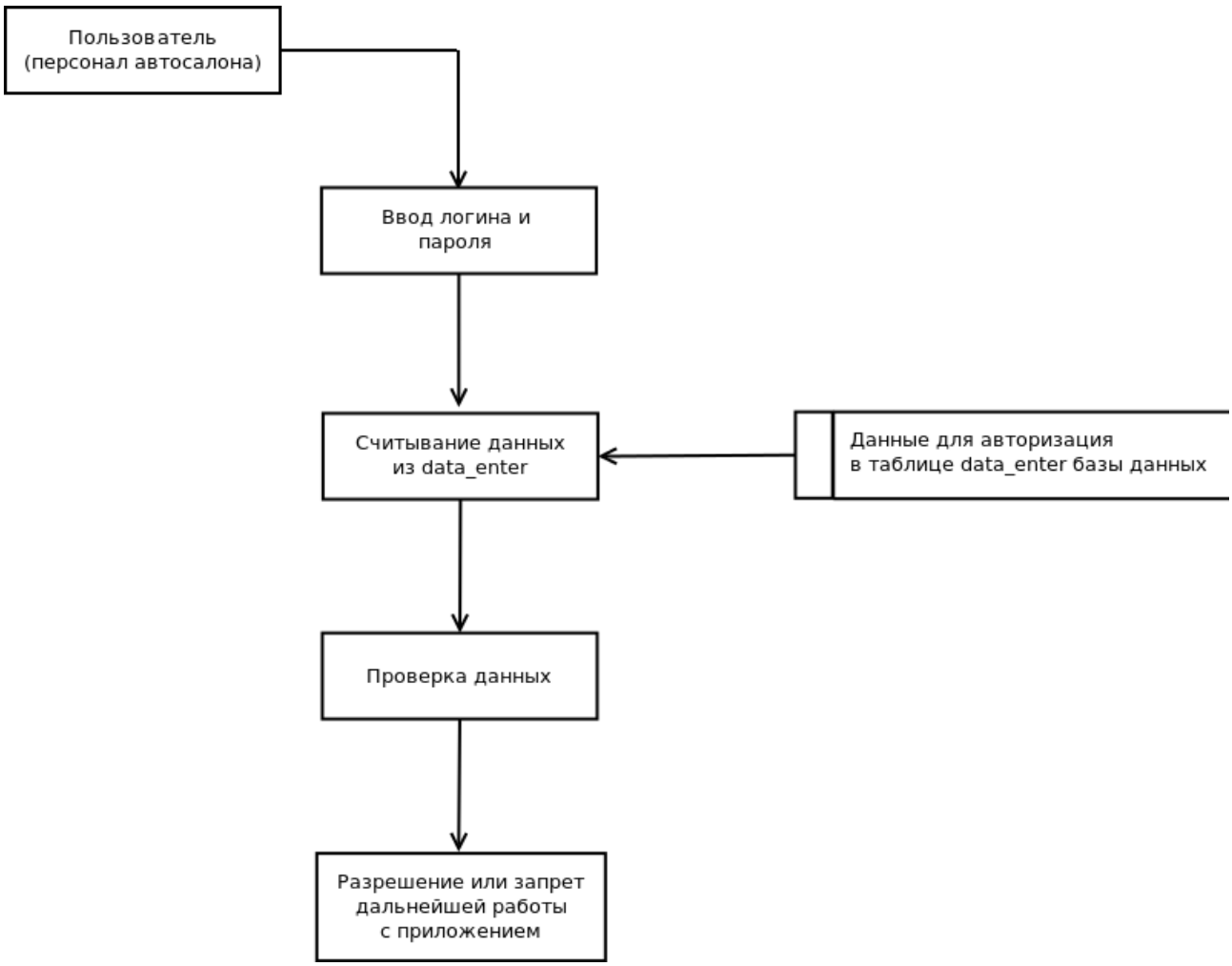


Рисунок 4. Детализация блока А1

2. Конструкторская часть

2.1 Выбор СУБД

Выбор производился между MySql и PostgreSql как самых распространеных, хорошо документированных и имеющих большие возможности.

MySql имеет все те же возможности, что и PostreSql, синтаксис языка в них практически одинаковый, средства работы с обоими СУБД просты и понятны. Но, в MySql проблемы с русским языком при использовании базы данных в приложениях, не сайтах. Также отсутствуют ограничения CHECK на проверки, что является проблемой при реализации приложения, создаваемого в рамках этой работы.

По этим причинам была выбрана PostgreSql как более развитая, позволяющая использовать ограничения. Поддержка русского языка в этой СУБД осуществляется без каких-либо дополнительных настроек.

2.2 Создание базы данных autocenter

```
DROP DATABASE autocenter
CREATE DATABASE autocenter
```

2.3 Определение типов данных для столбцов таблиц базы данных autocenter

Рекомендуемые типы данных для всех столбцов приведены в следующей таблице:

Таблица	Столбец	Тип данных
data_enter	login	varchar(20)
	password	varchar(20)
	post	varchar(20)
	name	varchar(20)
	surname	varchar(20)
general_car_info	id	integer
	firm	varchar(20)
	model	varchar(20)
	year	integer
	power	integer
	price	integer
	eng_type	varchar(20)
	present	varchar(3)

full_car_info	id	integer
	id_gen_car	integer
	body_type	varchar(20)
	drive_type	varchar(20)
	rudder_type	varchar(20)
	color	varchar(20)
	condition_type	varchar(10)
	conditioner	varchar(3)
	tv	varchar(3)
	navigation	varchar(3)
	outgo	float
details_info	id	integer
	name	varchar(40)
	price	integer
	firm	varchar(20)
	type	varchar(20)
	info	text
sales_car	id	integer
	data	varchar(11)
	time	varchar(5)
	seller_info	text
	firm	varchar(20)
	model	varchar(20)
	price	integer
	id_full_info	integer
	sale_number	integer
sales_detail	id	integer
	data	varchar(11)
	time	varchar(5)
	tech_info	text
	price	integer
	detail_count	integer
	id_info	integer

2.4 Сценарий создания таблиц в базе данных autocenter

Ниже приведены сценарии для создания каждой из таблиц базы данных. В этих же сценариях происходит добавление ограничений CHECK, первичных и внешних ключей и уникальности полей.

Таблица data_enter:

```
CREATE TABLE data_enter (
	login VARCHAR(20) NOT NULL,
	password VARCHAR(40) NOT NULL,
	post VARCHAR(20) check (post='seller' or post='tech'),
	name VARCHAR (20) NOT NULL,
	surname VARCHAR(20) NOT NULL, UNIQUE (login), UNIQUE (password)
);
```

Таблица general_car_info:

```
CREATE TABLE general_car_info (
	id serial primary key,
	firm VARCHAR(20) NOT NULL,
	model VARCHAR(20) NOT NULL,
	year int check (year>=1920 and year<=2013),
	power int NOT NULL check (power>=50 and power<=1001) ,
	price int NOT NULL check (price > 0),
	eng_type VARCHAR(20) NOT NULL check (eng_type='инжектор' or eng_type='бензин-турбина' or eng_type='дизель' or eng_type='турбодизель' or eng_type='битурбо' or eng_type='компрессор'),
	present VARCHAR(3) NOT NULL
);
```

Таблица full_car_info:

```
CREATE TABLE full_car_info(
	id serial primary key NOT NULL,
	id_gen_car int NOT NULL check (id_gen_car > 0),
	body_type VARCHAR(20) NOT NULL check (body_type='Седан' or body_type='Кабриолет' or body_type='Авант' or body_type='Хетч-бек' or body_type='Минивен' or body_type='Купе' or body_type='Джип'),
	drive_type VARCHAR(20) NOT NULL check (drive_type='Передний' or drive_type='Задний' or drive_type='Полный'),
	rudder_type VARCHAR(20) NOT NULL check (rudder_type='Левый' or rudder_type='Правый'),
	color VARCHAR(20) NOT NULL,
	condition_type VARCHAR(10) NOT NULL check (condition_type='Хорошее'
```

```
or condition_type='Отличное' or condition_type='Среднее' or condition_type='Битый'),
	conditioner VARCHAR(3),
	tv VARCHAR(3),
	navigation VARCHAR(3),
	outgo float NOT NULL check (outgo>0), foreign key(id_gen_car) references general_car_info(id)
);
```

Таблица details_info:

```
CREATE TABLE details_info (
	id serial primary key,
	name VARCHAR(40) NOT NULL,
	price int NOT NULL check(price > 0),
	firm VARCHAR(20),
	type VARCHAR(20),
	info text
);
```

Таблица sales_car:

```
CREATE TABLE sales_car (
	id serial primary key,
	data VARCHAR(11) NOT NULL,
	time VARCHAR(5) NOT NULL,
	seller_info text NOT NULL,
	firm VARCHAR(20) NOT NULL,
	model VARCHAR(20) NOT NULL,
	price int NOT NULL,
	id_full_info int NOT NULL check (id_full_info>0),
	sale_number int NOT NULL check(sale_number>0)
);
```

Таблица sales_detail:

```
CREATE TABLE sales_detail (
	id serial primary key,
	data VARCHAR(11) NOT NULL,
	time VARCHAR(5) NOT NULL,
	tech_info text NOT NULL,
	price int NOT NULL,
	detail_count int NOT NULL check(detail_count >= 1),
	id_info int NOT NULL check(id_info > 0)
);
```

2.5 Импорт данных в таблицы базы данных autocenter

Для импорта данных в таблицы базы данных созданы следующие файлы:

- test_data_for_data_enter_table — для таблицы data_enter;
- test_data_for_general_car_info_table — для таблицы general_car_info;
- test_data_for_full_car_info_table — для таблицы full_car_info;
- test_data_for_details_info_table — для таблицы details_info;
- test_data_for_sales_car_table — для таблицы sales_car;
- test_data_for_sales_detail — для таблицы sales_detail;

Для того, чтобы считать данные из файлов в таблицы базы данных, необходимо воспользоваться функцией `COPY` (т. к. используется СУБД PostgreSql):

- `copy data_enter from '/home/yura/Project/test_data_for_data_enter_table';`
- `copy general_car_info (firm, model, year, power, price, eng_type, present) from '/home/yura/Project/test_data_for_general_car_info_table';`
- `copy full_car_info (id_gen_car, body_type, drive_type, rudder_type, color, condition_type, conditioner, tv, navigation, outgo) from '/home/yura/Project/test_data_for_full_car_info_table';`
- `copy details_info (name, price, firm, type, info) from '/home/yura/Project/test_data_for_details_info_table';`
- `copy sales_car (data, time, seller_info, firm, model, price, id_full_info, sale_number) from '/home/yura/Project/test_data_for_sales_car_table';`
- `copy sales_detail (data, time, tech_info, price, detail_count, id_info) from '/home/yura/Project/test_data_for_sales_detail_table';`

Здесь `/home/yura/Project/` - путь к папке, в которой хранятся перечисленные выше файлы с данными для таблиц базы данных.

2.6 Шифрование данных из столбца password таблицы data_enter базы данных autocenter

Таблица data_enter содержит такие столбцы, как login и password. Данные из этих столбцов конфиденциальны. Поэтому хранить пароль в чистом виде нельзя, нужно его шифровать. СУБД PostgreSql предоставляет функцию *crypt* для этих целей. Таким образом, при добавлении данных в таблицу data_enter или считывание данных в

нее же из файла, необходимо использовать данную функцию. Ниже приведен код, который это реализует:

```
update data_enter set password=crypt(password, gen_salt('md5'));
```

2.7 Модули базы данных

Хранимые процедуры, функции или триггеры в данной реализации отсутствуют. Это связано с тем, что все запросы на обновление, получение, удаление или добавление данных реализуются непосредственно самим приложением. Такую возможность предоставляет Qt. В технологической части будет подробно описаны механизмы, позволяющие это сделать.

3. Технологическая часть

3.1 Модуль QPSQL

Настольное приложение написано на языке C++ с широким использованием библиотеки Qt. Все действия, в той или иной степени связанные с базой данных, будь то вставка данных или их удаление, выполняются с использованием модуля PostgreSql для Qt — QPSql. Для установки данного модуля в Linux (например Ubuntu) нужно установить пакет `libqt5sql5-psql`.

Этот модуль устанавливает связь между приложением и базой данных, и обеспечивает обмен данными. Каким образом устанавливается связь между приложением и конкретной базой данных будет описано ниже.

3.2 QSqlDatabase и установка соединения

Qt обеспечивает разработчика всеми средствами для работы с базой данных, а именно - предоставляет интерфейс для этого. Внутреннее устройство этих механизмом самому разработчику знать не нужно, ему нужен лишь конкретный функционал.

В конструкторской части была создана база данных autocenter. Для того, чтобы взаимодействовать именно с этой базой данной из приложения, нужно создать объект класса QSqlDatabase и в конструкторе указать модуль, который будет обеспечивать подключение к базе и обмен данными, и имя базы данных. В коде это выглядит так:

```
QSqlDatabase db = new QSqlDatabase(«QPSQL», «autocenter»);
```

После этого, нужно установить имя хоста, имя пользователя базы данных и его пароль. В данном приложении база данных расположена на том же компьютере, на котором запускается приложение, поэтому имя хоста - «localhost». Чтобы произвести все эти установки, нужно использовать методы класса QSqlDatabase и применять их к вышесозданному экземпляру класса данного класса, а именно — db. В коде это выглядит так:

```
db.SetHostName(«localhost»);
db.SetUserName(«user_name»);
db.SetPassword(«user_password»);
```

После всех проделанных выше манипуляций все запросы, которые будут выполняться в приложении, будут выполняться от пользователя с именем user_name. Установка имени пользователя и пароля осуществляется один раз.

3.3 QSqlQuery и его использование в приложении

В конструкторской части говорилось, что реализация данной системы не имеет ни одной хранимой процедуры, функции или триггера. Вместо этого для взаимодействия с данными базы данных широко используется класс библиотеки Qt — QSqlQuery.

Класс QSqlQuery - это интерфейс между приложением и данными, хранящимися в базе данных. Представим, что нам нужно получить все данные из таблицы general_car_info базы данных autocenter. Для этого нужно обьявить экземпляр класса QSqlQuery, связать его с базой данных, с которой будет производиться работа, и передать sql-запрос, который нужно выполнить, в метод exec() класса QSqlQuery.

Связывать экземпляр класса QSqlQuery с конкретной базой данных не обязательно. Если не указать имя базы данных, то будет использоваться база данных, с которй на текущий момент установлено соединение. Но, для безопасности и предотвращения ошибок, лучше указать имя базы данных явно. В коде данный пример выглядит так:

```
QSqlQuey query = new QsqlQuery(QSqlDatabase::database(«autocenter»));
query.exec(«select * from general_car_info order by id»);
```

После выполнения этих строчек кода будут получены все записи из таблицы general_car_info, упорядоченные по id.

Таким образом, QSqlQuery — мощный инструмент для взаимодействия с данными базы данных, который позволяет производить любые действия с ними.

3.4 Основная концепция приложения

Глобально, приложение разделено на две ветки: ветка продаж автомобилей и ветка продаж автозапчастей. Обе ветки имеют одинаковый функционал, а именно:

- просмотр каталогов;
- изменение данных в каталоге;
- добавление данных в каталог;
- удаление данных из каталога;
- просмотр списка продаж;
- составление контрактов продаж
- поиск по каталогу;

Разница состоит в том, что обе ветки работают с разными таблицами, а значит и с разными данными. На рисунке 5 показана архитектурная схема приложения.

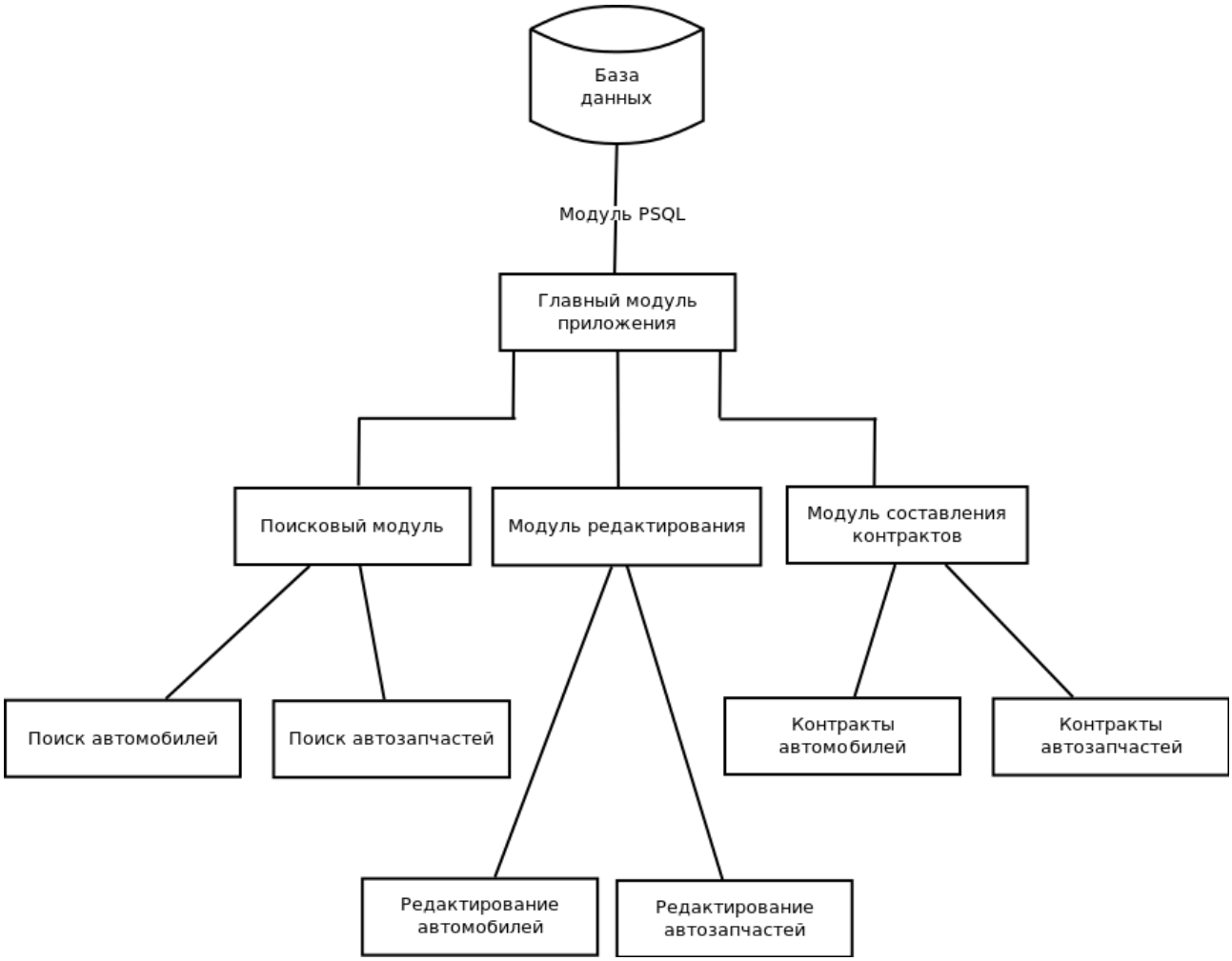


Рисунок 5. Архитектура приложения

Все данные о имеющейся продукции хранятся в каталогах, которые с приложении отражаются в виде сетки. На рисунке 7 показан внешний вид каталога ветки автомобилей. Для ветки автозапчастей каталог выглядит также, отличие заключается в содержимом каталога.

3.5 Получение данных из каталогов и заполнение QtableView

Библиотека предоставляет разработчику такой класс, как QSqlQueryModel,

который помогает получать целый поток данных и размещать их в нужном объекте путем привязывания экземпляра класса QSqlQueryModel к этому объекту.

В качестве такого объекта в данной программе выступает QTableView. Ниже показано каким образом привязать экземпляр класса QSqlQueryModel к этому объекту.

```
QSqlQueryModel *queryModel = new QsqlQueryModel();
tbl->setModel(queryModel);
```

Теперь, все результаты манипуляций, которые будут производится с элементом queryModel, будут отображаться в элементе tbl. Элемент tbl — это экземпляр класса QTableView, представляющий собой сетку с данными. Независимо от того, вернет queryModel одно или несколько данных, количество столбцов и строк в элементе tbl будет автоматически пересчитано на нужное.

Еще одной особенностью использования данного подхода является то, что в результате выполнения какого-то запроса внутри queryModel, названия столбцов в элементе tbl будут соответствовать названиям столбцов таблицы, из которой делается запрос.

Рассмотрим на примере получения данных из таблицы general_car_info базы данных autocenter. Запрос выглядит так:

```
select * from general_car_info order by id"
```

Этот запрос передается в метод setQuery() элемента queryModel и автоматически выполняется. Все данные, которые возвращаются в результате выполнения этого запроса будут отражены в элементе tbl, к которому привязан queryModel. Названия столбцов в tbl станут такими же, как и названия столбцов в таблице general_car_info.

Для того, чтобы установить свои значения столбцов, как это сделано в программе, нужно вызвать метод setHeaderData() для элемента queryModel, в который передать номер столбца и устанавливаемое название. Ниже приведен пример этого.

```
queryModel->setHeaderData(1, Qt::Horizontal, «Марка»)
```

Таким образом, QSqlQueryModel обеспечивает отображение потока данных, полученного в результате выполнения запроса, в таблице.

3.6 Добавление и удаление данных

Добавление и удаление данных выполняется с использованием класса QsqlQuery и метода exec(). В метод exec() передается запрос на добавление или удаление определенных данных в определенную таблицу, затем этот запрос

выполняется, тем самым выполняется добавление или удаление в самой таблице.

В данной программе имеются два каталога, добавление в которые осуществляется самими пользователями — продавцами. Это каталог автозапчастей и каталог автомобилей. Также есть каталог, отражающие продажи, но добавление в них происходит внутри программы без прямого участия пользователя (будет описано в разделе 3.9).

Рассмотрим на примере добавление авто в каталог автомобилей. Пользователь заполняет все поля, которые характеризуют такой объект, как автомобиль. После этого происходит проверка на корректность введенных данных и, если данные корректны, происходит формирование строки запроса. Пример строки запроса на добавление автомобиля в таблицу general_car_info выглядит так:

```
str = «insert into general_car_info (firm, model, year, power, price, eng_type, present) values ('audi','a6',2010,200,900000,'инжектор','Да')»
```

После того, как строка запроса сформирована, создается экземпляр класса QSqlQuery, а затем выполняется метод exec(), применимо к данному экземпляру, в который передается сформированная строка. Ниже приведен код .

```
QSqlQuery *qr = new QsqlQuery(QSqlDatabase::database(work_base))
qr.exec(str)
```

Что касается удаления, то она производится по той же технологии, разница состоит лишь в содержимом выполняемого запроса. Пример запроса на удаление приведен ниже.

```
str = «delete from general_car_info where id = 10»
```

Для удаления записи в строке запроса должен быть указан id удалемой записи. После выполнения запроса запись в нужной таблице с заданным id будет удалена.

3.7 Изменение данных

Пользователь может изменять следующие данные:

- данные об автомобилях:
- данные об автозапчастях;

Для изменения данных об автомобиле пользователь должен щелкнуть по id того автомобиля, характеристики которого хочет отредактировать, а затем нажать кнопку «Изменить». В результате этого откроется форма редактирования, содержащая текущие значения характеристик, которые можно редактировать.

После завершения редактирования происходит считывание и проверка на корректность новых данных, формируется строка запроса, которая передается в метод exec() экземпляра класса QSqlqQuery. Обновление применяется к той записи в базе данных, id которой равно id автомобиля, над которым производилось редактирование. Пример строки запроса на обновление приведен ниже.

```
update general_car_info set firm ='audi', model='a6', year=2011, power=200, price=900000, eng_type='инжектор', present='Да' where id=10
```

Аналогичные действия производятся для изменения данных об автозапчастях.

3.8 Организация поиска

Пользователь может производить поиск по каталогам:

- автозапчастей;
- автомобилей;
- продаж автозапчастей;
- продаж автомобилей;

Поиск по каталогам продаж осуществляется только по дате с той целью, чтобы найти все записи, проданные в конкретный день.

Поиск по каталогу автомобилей может осуществляться по следующим параметрам:

- марка;
- модель;
- год выпуска;
- мощность;
- цена;
- наличие;

Поиск по каталогу автозапчастей может осуществляться по следующим параметрам:

- наименование;
- фирма;
- типа детали;
- цена:

Особенностью поиска является то, что можно заполнять не все поля, а так же наличие поисковых интервалов. Это означает, что вы можете, например, заполнить

только модель и искать по этой модели, не заполняя остальные поля. Также можно указать интервал, например, от 2009 до 2012 года, и в результатах поиска окажутся только те автомобили, год выпуска которых попадет в этот интервал.

Достигается это путем анализа введенных данных и динамического формирования строки запроса. Образец строки запроса приведен ниже.

```
select * from general_car_info where firm='firm_value' and mark='mark_value' and year>min_year_value and year<max_year_value and power>min_power_value and power<max_power_value and price>min_price_value and price<max_price_value
```

После того, как пользователь заполнил поисковые данные, происходит проверка на корректность введенных данных. Если данные введены корректно, то начинается формирование строки запроса по образцу, приведенному выше. Если какое-то поле не заполнено, то оно либо пропускается, либо вместо него подставляется граничное значение, о которых рассказано ниже (для каталога автомобилей; для каталога автозапчастей просто пропускается). Например, если не заполнено поле «Модель», то поисковая строка буде выглядеть так:

```
select * from general_car_info where firm='firm_value' and year>min_year_value and year<max_year_value and power>min_power_value and power<max_power_value and price>min_price_value and price<max_price_value
```

Граничные значения, подставляемые в поисковую строку (если пользователь явно не указал значение), определены в ограничениях, описанных при создании таблиц базы данных. Например, для поля year максимальное значение равно 2013. Таким образом, если пользователь явно не введет максимальную границу года выпуска, то вместо нее будет подставлено значение 2013.

Такой же механизм характерен и для поиска в каталоге автозапчастей.

После того, как строка сформирована, она передается в метод setQuery() экземпляра класса QsqlQueryModel, где и выполняется, а затем данные, полученные в результате выполнения, отображаются в таблице, к которой был привязан данный экземпляр. Всего в программе 4 таких таблицы: для каталога автозапчастей, автомобилей, списка продаж автозапчастей, списка продаж автомобилей. На рисунке 8 показан внешний вид списка продаж.

На рисунке 6 Изображена блок-схема реализации поиска для автозапчастей. Для автомобилей аналогично вышеописанному.

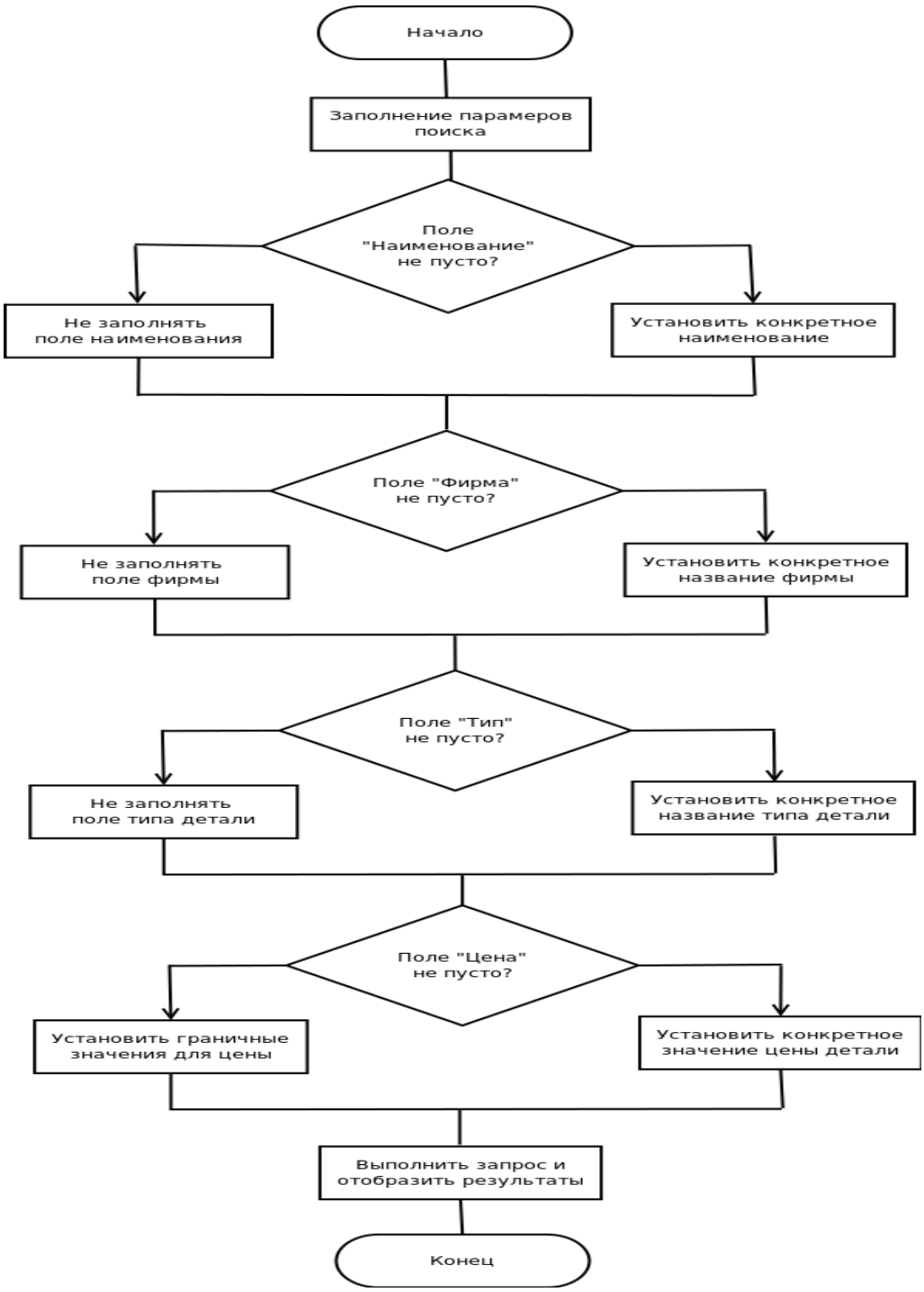


Рисунок 6. Блок-схема алгоритма поиска автозапчастей

3.9 Составление контрактов продаж

Когда продавец подобрал нужный автомобиль клиенту, или нужную запчасть, он должен оформить контракт продажи. Система позволяет сделать это автоматически, для этого требуется лишь несколько действия со стороны продавца.

Первое, что требуется выполнить — это щелкнуть по id выбранной продукции, будь то автомобиль или автозапчасти (рисунок 7, столбец id).

Второе — это нажать кнопку «Оформить заказ» (рисунок 7). После этого действия будет открыто окно, в котором от продавца требуется заполнить данные клиента (ФИО), дату его рождения и дату продажи. После заполнения и нажатия кнопки «Готово» (рисунок 9) будет произведена проверка корректности введенных данных и, если данные корректны, составлен контракт. Также, автоматически происходит добавление проданной продукции в каталог списка продаж этой продукции (автомобилей — в каталог продаж автомобилей, автозапчастей — в каталог продаж автозапчастей).

Технология добавления данных в таблицы была описана в разделе 3.6. Особенность в том, что все данные, необходимые для составления запроса на добавление в каталог продаж уже имеются, поэтому сам процесс добавления скрыт от пользователя и не требует его участия. Пример строки для запроса на добавления в список продаж автомобилей приведен ниже.

```
insert into sales_car (data, time, seller_info, firm, model, price, id_full_info, sale_number) values('13.10.2013','11:09,'Иван Иванов','audi','a6',900000,3,4)
```

Все контракты сохраняются в папку «contracts» и имеют фиксированные имена. Для контракта продажи авто - «car_contract_№», а для контракта продажи автозапчасти - «detail_contract_№». Здесь № - порядковый номер продажи в списке продаж. Пример содержимого обоих контрактов приведен в приложении.

Список продаж также хранится в каталоге, отображаемом в сетке. На рисунке 8 представлен внешний вид формы списка продаж.

При составлении контракта продажи продавец-консультант оперирует с формой составления контрактов. На рисунке 9 показан внешний вид этой формы.

3.10 Отслеживание ошибок в приложении

В приложении происходит постоянное отслеживание ошибок. При вводе данных проверяются проверки на корректность данных и, при обнаружении некорректных данных, пользователю выдается сообщение об этом. Например, если пользователь вводит неверные данные для авторизации, ему выдается следующее сообщение:

```
Ошибка","Авторизация не пройдена.
```

При выполнении метода exec() отслеживается, чтобы во время его выполнения не произошло никаких ошибок. Если такие ошибки возникают, то они

перехватываются и также выводятся пользователю в качестве информации. В коде это выглядит так:

```
if (!temp->exec(str_query))
        throw temp->lastError().text().toStdString().c_str();
```

Здесь str_query — строка запроса. При возникновении ошибки будет выведено сообщение, комментирующее эту ошибку.

Заключение

В результате проделанной работы были выполнена все задачи, поставленные в техническом задании. Настольное приложение было создано согласно всем требованиям. Оно позволяет продавцам-консультантам осуществлять поиск по каталогам, изменять записи, добавлять и удалять их.

Все записи хранятся на русском языке, но можно вносить данные и на английском, проблем с отображением не возникнет.

Контракты сохраняются в папку contracts и могут быть напечатаны без каких-либо проблем на любом принтере. Они не имеют строгой структуры, оформленной в соответствии с каким-нибудь стандартом, а создаются просто для демонстрации и имеют произвольную структуру, взятую «из головы».

Список использованной литературы

1. Брауде Э. Д. Технология разработки программного обеспечения. - СПб.: Питер, 2004.

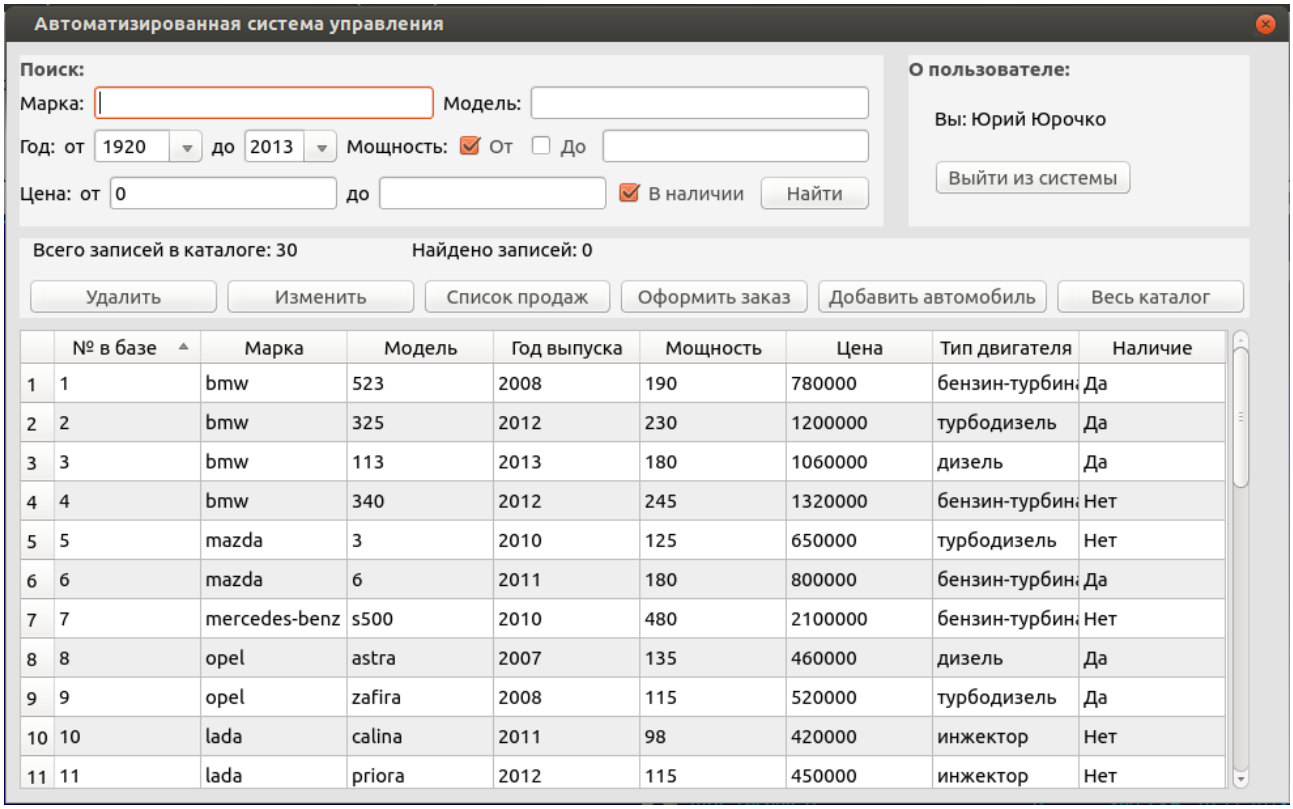
2. Фред Ролланд. Основные концепции баз данных. - М.: Вильямс, 2003.

3. Построение запросов и программирование на SQL/А. В. Маркин. - М.: Диалог-МИФИ, 2011.

4 .Мартин Грабер. Введение в SQL. - М.: Лори, 2008.

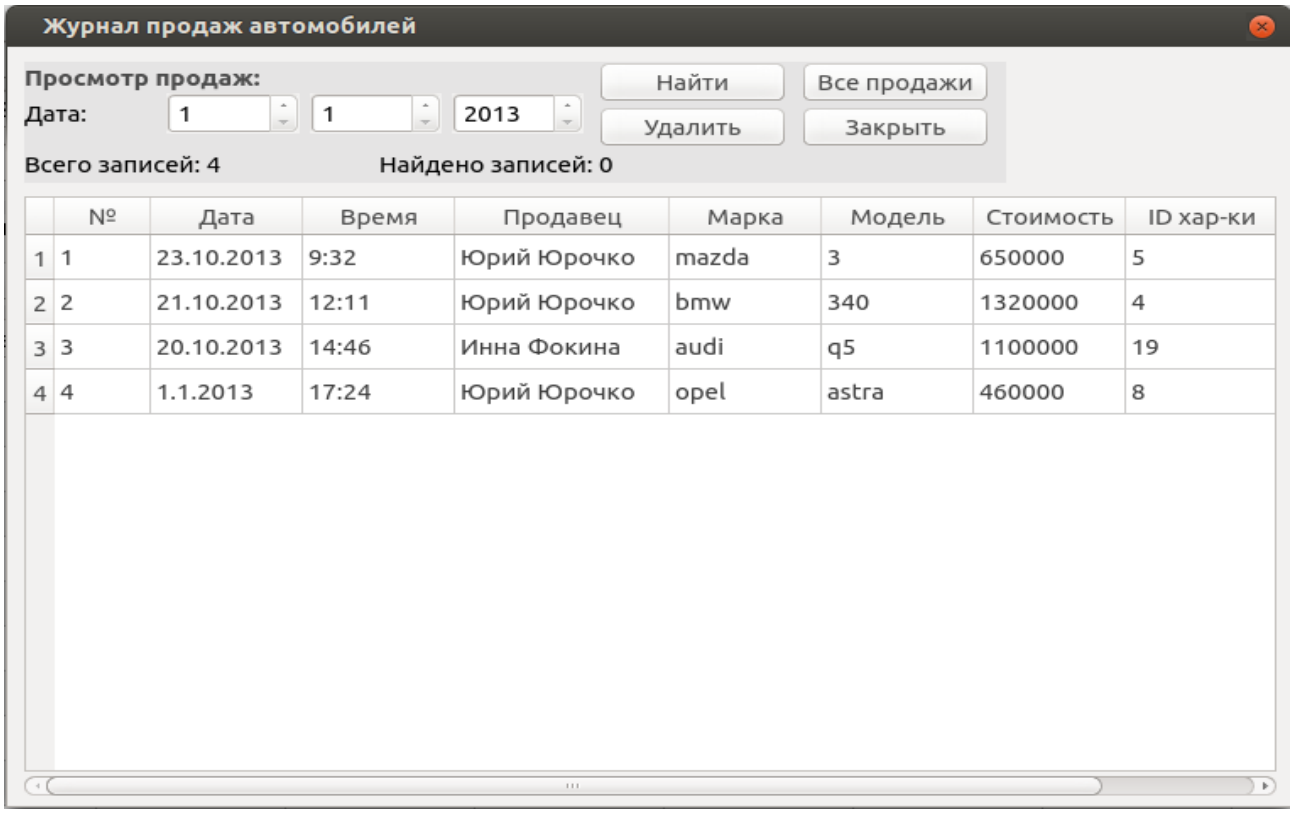
5. SQL. Программирование/Д. Кауфман и др. - М.: БИНОМ. Лаборатория знаний, 2002.

Приложения



	№ в базе	Марка	Модель	Год выпуска	Мощность	Цена	Тип двигателя	Наличие
1	1	bmw	523	2008	190	780000	бензин-турбин	Да
2	2	bmw	325	2012	230	1200000	турбодизель	Да
3	3	bmw	113	2013	180	1060000	дизель	Да
4	4	bmw	340	2012	245	1320000	бензин-турбин	Нет
5	5	mazda	3	2010	125	650000	турбодизель	Нет
6	6	mazda	6	2011	180	800000	бензин-турбин	Да
7	7	mercedes-benz	s500	2010	480	2100000	бензин-турбин	Нет
8	8	opel	astra	2007	135	460000	дизель	Да
9	9	opel	zafira	2008	115	520000	турбодизель	Да
10	10	lada	calina	2011	98	420000	инжектор	Нет
11	11	lada	priora	2012	115	450000	инжектор	Нет

Рисунок 7. Внешний вид каталога ветки продажи автомобилей



	№	Дата	Время	Продавец	Марка	Модель	Стоимость	ID хар-ки
1	1	23.10.2013	9:32	Юрий Юрочко	mazda	3	650000	5
2	2	21.10.2013	12:11	Юрий Юрочко	bmw	340	1320000	4
3	3	20.10.2013	14:46	Инна Фокина	audi	q5	1100000	19
4	4	1.1.2013	17:24	Юрий Юрочко	opel	astra	460000	8

Рисунок 8. Внешний вид списка продаж

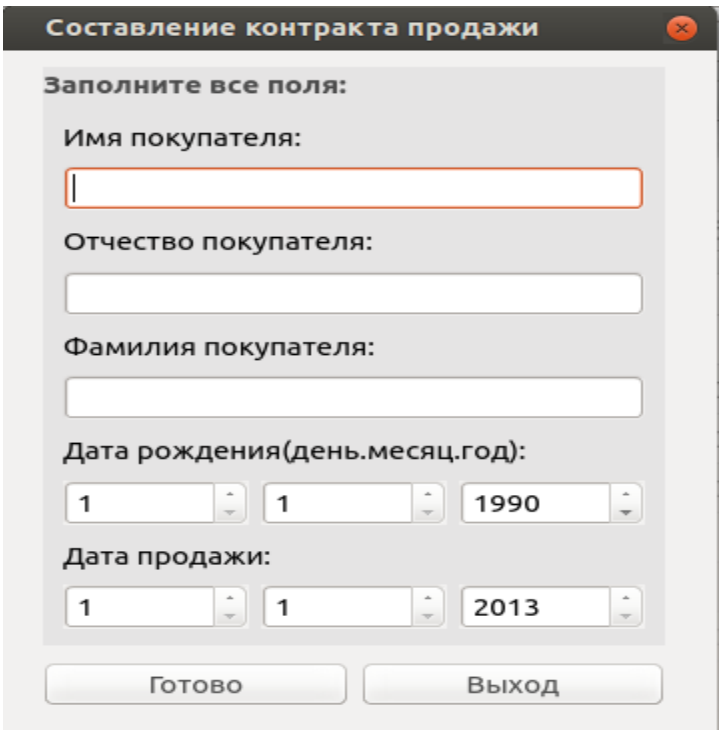


Рисунок 9. Форма составления контракта продажи