**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 4](#_Toc450752436)

[1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ 5](#_Toc450752437)

[2 ПРОЕКТИРОВАНИЕ РЕЛЯЦИОННОЙ БАЗЫ ДАННЫХ 8](#_Toc450752438)

[2.1 Построение ER-диаграммы 8](#_Toc450752439)

[2.2 Построение схемы реляционной базы данных в третьей нормальной форме 10](#_Toc450752440)

[2.3 Описание физической модели данных 13](#_Toc450752441)

[3 ОПИСАНИЕ БД И ПРИЛОЖЕНИЯ ДЛЯ РАБОТЫ С НЕЙ 21](#_Toc450752442)

[3.1 Описание структуры базы данных 21](#_Toc450752443)

[3.2 Создание форм для ввода и отображения данных 25](#_Toc450752444)

[3.3 Формирование запросов и отчётов 42](#_Toc450752445)

[3.4 Описание интерфейса базы данных 61](#_Toc450752446)

[ВЫВОД 64](#_Toc450752447)

[СПИСОК ЛИТЕРАТУРЫ 65](#_Toc450752448)

[ПРИЛОЖЕНИЯ 66](#_Toc450752449)

# ВВЕДЕНИЕ

База данных – это совокупность структур, предназначенных для хранения больших объемов информации и программных модулей, осуществляющих управление данными, их выборку, сортировку и другие подобные действия.

Информация в базе данных хранится в одной или нескольких таблицах. Любая таблица с данными состоит из набора однотипных записей, расположенных друг за другом. Они представляют собой строки таблицы, которые можно добавлять, удалять или изменять.

Базы данных стали неотъемлемой частью практически всех компьютерных систем – от отрасли до отдельного предприятия. За последние несколько лет вырос уровень потребительских качеств систем управления базами данных (СУБД): разнообразие поддерживаемых функций, удобный для пользователя интерфейс, сопряжение с программными продуктами, в частности с другими СУБД, возможности для работы в сети и т.д. СУБД позволяет сводить воедино информацию из самых разных источников (электронные таблицы, другие базы данных) и помогает быстро найти необходимую информацию, донести ее до окружающих с помощью отчетов, графиков или таблиц.

Для разработки базы данных в данной курсовой работе выбрана СУБД MySQL. Важной особенностью MySQL является простота использования. Для начала работы с MySQL не требуется сложной процедуры конфигурации. MySQL Server начнет работать соответствующим образом сразу. По умолчанию выбираются значения, соответствующие минимальному использованию ресурсов диска и памяти. Для получения оптимальной производительности и для специальных условий (например, для проверки входа в систему), конечно же, потребуется дополнительная настройка. Чтобы помочь выполнить такую настройку, предлагаются соответствующие примеры файлов типовой конфигурации.

# 1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

В курсовой работе необходимо разработать базу данных “Аптека”. Данная база должна включать в себя просмотр информации о лекарствах, производителях, продажах, работниках, акциях, поставках. Выборку этой информации по различным критериям, ввод новых значений, а также изменение уже существующих.

Цель создания текущей базы данных:

* учет прихода и ухода товара;
* возможность его реализации;
* печать накладных и чеков;
* управление данными товара и информацией, связанной с ним.

В ходе проведённого анализа предметной области были выделены следующие объекты, информацию о которых необходимо учитывать при проектировании и разработке базы данных.

1. **Лекарства.** Они характеризуются следующей информацией: штрих-код; наименование; производитель; лекарственная группа, к которой принадлежит товар; упаковка товара; материал, а именно какого рода лекарство – жидкость, таблетки и т.п.; форма отпуска (без рецепта, с рецептом, строгого учета); инструкция по применению.
2. **Производители,** которые характеризуются следующей информацией: название, страна, город, адрес, телефон
3. **Работники.** О работниках должна содержаться следующая информация: Ф.И.О.; телефон; должность в аптеке; № контракта, по которому он работает; идентификационный номер налогоплательщика (ИНН).

Еще лекарства характеризуются ценой, ранжированной по дате. Новую цену можно назначить через неделю после задания текущей цены.

1. **Скидки (акции).** Скидки устанавливаются на определённый товар. Существует начало и конец акции, в течении которой действует скидка. Скидка представляет собой определённый процент. В базе необходимо хранить также краткое описание акции.
2. **Поставка лекарств.** Лекарства поставляются в аптеку по накладным, в которых содержится следующая информация: сотрудник, который принял товар; поставщик; лекарство; количество; оптовая цена лекарства; дату поступления и срок годности.
3. **Продажа лекарств** характеризуется следующей информацией: дата и время проведения операции, имя кассира, тип оплаты (картой или наличными), а также информацию о проданном лекарстве: название, количество, по какой цене был продан, аванс, данный покупателем и общая сумма чека.
4. **Пользователи.** Так как в данной базе существует разграничение доступа, то пользователи необходимы. Существует 3 вида пользователей: кассир, администратор, директор. Директор имеет наивысший уровень допуска и может задавать пароли доступа. Также ведутся логи действий, которые потом можно посмотреть.

Для доступа к хранимым данным существую следующие ограничения:

* запрещено изменение накладных, чеков;
* удаление накладных и чеков возможно при вводе пароля;
* при создании чеков удаление товара из чека возможно после ввода пароля безопасности;
* удаление чеков и накладных возможно только после ввода пароля.

Кроме того накладываются следующие ограничения общего порядка:

* не может быть работников с одинаковым ИНН или № контракта;
* количество пользователей строго ограничено – по одному на каждый тип;
* невозможно удалить администратора БД;
* нельзя продать просроченный товар;
* нельзя продать товара больше, чем есть на складе;
* цену на товар можно устанавливать раз в неделю;
* все записи связаны, поэтому удаление, например, производителя повлечет удаление товара этого производителя.

В приложении должна формироваться следующая выходная документация: чек; отчет продаж; прайс лекарств.

В ходе анализа предметной области были определены основные виды запросов к безе данных:

* выборка данных, ориентируемая на объекты;
* селективная фильтрация данных, по различным характеристикам;
* подсчет суммы чека, накладной;
* изменение и удаление данных;
* подсчет товара по состоянию (просрочено, не просрочено, продано, всего);
* подсчет количества работников одной профессии;
* выборка данных для архивации;
* интерактивный поиск данных о товаре, для кассира;
* продажа товара (реализация).

# 2 ПРОЕКТИРОВАНИЕ РЕЛЯЦИОННОЙ БАЗЫ ДАННЫХ

# 2.1 Построение ER-диаграммы

Средством моделирования предметной области на этапе концептуального проектирования БД является модель «сущность – связь» или ER-диаграмма. Основными объектами данной диаграммы являются сущности и их взаимосвязь между собой.

На этапе анализа предметной области были определены основные объекты предметной области «Аптека», информацию о которых необходимо хранить в проектируемой базе данных.

Каждый из объектов предметной области представляет собой сущность ER-диаграммы. Основные объекты – это лекарства, сотрудники, поставщики. Они представлены на диаграмме (рисунок 2.1) стержневыми сущностями. Поставка и продажа характеризуют взаимосвязь между стержнями и представляют собой ассоциативные сущности.

Каждый объект характеризуется различными параметрами. Характеристики представлены на диаграмме как атрибуты сущностей.

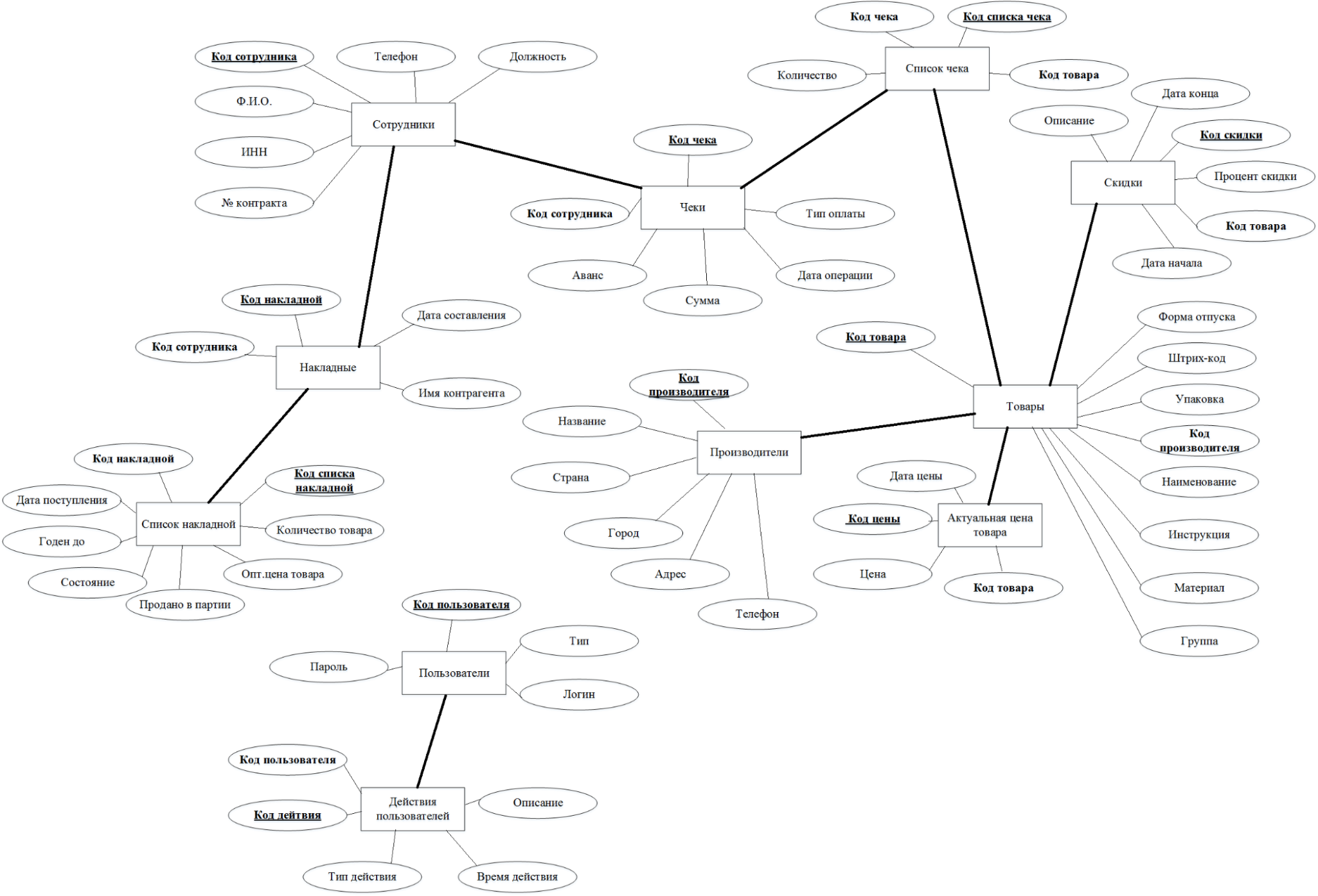


Рисунок 2.1 – ER-диаграмма

# 2.2 Построение схемы реляционной базы данных в третьей нормальной форме

Реляционная база данных представляет собой множество взаимосвязанных таблиц, каждая из которых содержит информацию об объектах определенного вида. Каждая строка таблицы содержит данные об одном объекте, а столбцы таблицы содержат различные характеристики этих объектов.

На этапе построения ER-диаграммы были выделены основные сущности предметной области: работник, производитель, товар, скидка, актуальная цена товара, чек, список чека, накладная, список накладной, пользователь, действие пользователя.

Каждая из этих сущностей согласно реляционной модели преобразуется в отношение (таблица 2.1 – 2.11).

Таблица 2.1 – Отношение «Работник»

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Код работника | Ф.И.О. работника | ИНН работника | № контракта работника | Телефон работника | Должность работника |
|  |  |  |  |  |  |

Таблица 2.2 – Отношение «Производитель»

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Код производителя | Название  производителя | Страна | Город | Адрес | Телефон |
|  |  |  |  |  |  |

Таблица 2.3 – Отношение «Товар»

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Код товара | Название товара | Код производителя | Группа товара | Упаковка товара | Материал товара | Форма отпуска товара | Штрих-код | Инструкция |
|  |  |  |  |  |  |  |  |  |

Таблица 2.4 – Отношение «Скидка»

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Код  скидки | Код  товара | Процент скидки | Дата начала  скидки | Дата конца скидки | Описание скидки |
|  |  |  |  |  |  |

Таблица 2.5 – Отношение «Актуальная цена товара»

|  |  |  |  |
| --- | --- | --- | --- |
| Код цены товара | Код товара | Дата цены | Цена  товара |
|  |  |  |  |

Таблица 2.6 – Отношение «Накладная»

|  |  |  |  |
| --- | --- | --- | --- |
| Код накладной | Код работника | Дата составления | Ф.И.О. контрагента |
|  |  |  |  |

Таблица 2.7 – Отношение «Список накладной»

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Код списка накладной | Код накладной | Количество товара | Оптовая цена товара | Дата поступления товара | Товар годен до | Код товара |
|  |  |  |  |  |  |  |

Таблица 2.8 – Отношение «Чек»

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Код чека | Код работника | Дата реализации | Тип оплаты | Сумма чека | Аванс покупателя |
|  |  |  |  |  |  |

Таблица 2.9 – Отношение «Список чека»

|  |  |  |  |
| --- | --- | --- | --- |
| Код списка чека | Код чека | Код товара | Количество товара |
|  |  |  |  |

Таблица 2.10 – Отношение «Пользователи»

|  |  |  |  |
| --- | --- | --- | --- |
| Код пользователя | Тип пользователя | Логин пользователя | Пароль пользователя |
|  |  |  |  |

Таблица 2.11 – Отношение «Действие пользователя»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Код действия пользователя | Код пользователя | Тип действия | Время действия | Описание действия |
|  |  |  |  |  |

# 

# 2.3 Описание физической модели данных

Каждая из спроектированных таблиц соответствует свойствам реляционного отношения и соответственно находится в 1 нормальной форме.

В каждой из таблиц присутствует первичный ключ. Отношение находится в третьей нормальной форме тогда, когда находится в 2 нормальная форма и все не ключевые атрибуты взаимно независимые.

С учётом особенностей выбранной СУБД, преобразуем реляционную модель базы данных в физическую модель (таблицы 2.12 – 2.20).

Таблица 2.12 – «Работник»

|  |  |  |  |
| --- | --- | --- | --- |
| Содержание поля | Название поля | Тип | Ограничения |
| Код работника | E\_ID | int(9) | primary key  unsigned not null auto\_increment |
| Ф.И.О. работника | E\_NAME | char(50) | not null |
| ИНН работника | E\_INN | int(9) | unsigned  not null  unique |
| Содержание поля | Название поля | Тип | Ограничения |
| № контракта работника | E\_CONTRACT | bigint(11) | unsigned  not null  unique |
| Телефон работника | E\_TEL | char(15) | -------- |
| Должность работника | E\_POSITION | enum('Кассир','Уборщик',  'Менеджер',  'Администратор',  'Фармацевт') | not null |

Таблица 2.13 – «Производитель»

|  |  |  |  |
| --- | --- | --- | --- |
| Содержание поля | Название поля | Тип | Ограничения |
| Код производителя | M\_ID | int(9) | primary key  unsigned auto\_increment  not null |
| Название  производите ля | M\_NAME | char(50) | not null |
| Страна | M\_COUNTRY | char(15) | not null |
| Город | M\_CITY | char(15) | -------- |
| Адрес | M\_ADDR | char(20) | -------- |
| Телефон | M\_TEL | char(15) | -------- |

Таблица 2.14 – «Товар»

|  |  |  |  |
| --- | --- | --- | --- |
| Содержание поля | Название поля | Тип | Ограничения |
| Код товара | P\_ID | int(9) | primary key  unsigned auto\_increment not null |
| Название товара | P\_NAME | char(50) | not null |
| Код производителя | M\_ID | int(9) | foreign key  unsigned  not null  on update cascade on delete cascade |
| Группа товара | P\_GROUP | enum('Без группы'…) | not null |
| Упаковка товара | P\_PACK | enum('Без упаковки',…) | not null |
| Материал товара | P\_MATERIAL | enum('Неизвестно',…) | not null |
| Форма отпуска товара | P\_FORM | enum('с рецептом','без рецепта', 'строгого учета') | not null |
| Штрих-код | P\_CODE | BIGINT(13) | UNSIGNED NOT NULL |
| Инструкция | P\_INSTR | text | default null |
| Всего пришло | P\_IN | mediumint(7) | unsigned  default 0 |
| Всего ушло | P\_OUT | mediumint(7) | unsigned  default 0 |

Таблица 2.15 – «Скидка»

|  |  |  |  |
| --- | --- | --- | --- |
| Содержание поля | Название поля | Тип | Ограничения |
| Код скидки | D\_ID | mediumint(6) | primary key unsigned auto\_increment not null |
| Код товара | P\_ID | int(9) | foreign key  unsigned  not null  on update cascade on delete cascade |
| Процент скидки | D\_PRICE | int(4) | unsigned not null |
| Дата начала  скидки | D\_BDATE | date | ---------- |
| Дата конца скидки | D\_EDATE | date | ------------- |
| Описание скидки | D\_TEXT | text | ------------ |

Таблица 2.16 – «Актуальная цена товара»

|  |  |  |  |
| --- | --- | --- | --- |
| Содержание поля | Название поля | Тип | Ограничения |
| Код цены товара | PAP\_ID | int(9) | primary key  unsigned auto\_increment |
| Код товара | P\_ID | int(9) | foreign key  unsigned  not null  on update cascade on delete cascade |
| Дата цены | PAP\_DATE | datetime | not null  default current\_timestamp |
| Цена  товара | PAP\_PRICE | float(7,2) | unsigned  not null |

Таблица 2.17 – «Накладная»

|  |  |  |  |
| --- | --- | --- | --- |
| Содержание поля | Название поля | Тип | Ограничения |
| Код накладной | W\_ID | int(9) | primary key  unsigned auto\_increment |
| Код работника | E\_ID | int(9) | unsigned  not null  foreign key  on update cascade on delete cascade |
| Дата составления | W\_DATE | date | ------------ |
| Ф.И.О. контрагента | W\_AGENT\_NAME | char(50) | not null |

Таблица 2.18 – «Список накладной»

|  |  |  |  |
| --- | --- | --- | --- |
| Содержание поля | Название поля | Тип | Ограничения |
| Код списка накладной | WL\_ID | int(9) | primary key  unsigned auto\_increment |
| Код накладной | W\_ID | int(9) | foreign key  unsigned  not null  on update cascade on delete cascade |
| Количество товара | WL\_VALUE | mediumint(5) | unsigned  not null |
| Оптовая цена товара | WL\_TRADE\_PRICE | float(7,2) | unsigned  not null |
| Дата поступления товара | WL\_BDATE | date | not null |
| Товар годен до | WL\_EDATE | date | not null |
| Код товара | P\_ID | int(9) | foreign key  unsigned  not null  on update cascade on delete cascade |
| Количество проданных продуктов | WL\_COUNT | int(5) | unsigned default 0 |
| Состояние | WL\_OVERDUE | enum('Не просрочено','Просрочено','Скоро истекает срок годности','N/A','Продано') | default 'N/A' |

Таблица 2.19 – «Чек»

|  |  |  |  |
| --- | --- | --- | --- |
| Содержание поля | Название поля | Тип | Ограничения |
| Код чека | C\_ID | int(9) | primary key  unsigned auto\_increment |
| Код работника | E\_ID | int(9) | unsigned  not null  foreign key  on update cascade on delete cascade |
| Дата реализации | C\_DATE | datetime | not null |
| Тип оплаты | C\_PAYTYPE | enum('Наличные',  'Карточка') | not null |
| Сумма чека | C\_SUM | FLOAT(7,2) | UNSIGNED DEFAULT 0 |
| Аванс покупателя | C\_PREPAYMENT | FLOAT(7,2) | UNSIGNED DEFAULT 0 |

Таблица 2.20 – Отношение «Список чека»

|  |  |  |  |
| --- | --- | --- | --- |
| Содержание поля | Название поля | Тип | Ограничения |
| Код списка чека | CL\_ID | int(9) | primary key  unsigned auto\_increment |
| Код чека | C\_ID | int(9) | unsigned  not null  foreign key  on update cascade on delete cascade |
| Код товара | P\_ID | int(9) | unsigned  not null  foreign key  on update cascade on delete cascade |
| Количество товара | CL\_VALUE | mediumint(5) | unsigned not null |

Таблица 2.21 – «Пользователи»

|  |  |  |  |
| --- | --- | --- | --- |
| Содержание поля | Название поля | Тип | Ограничения |
| Код пользователя | U\_ID | smallint(3) | primary key  unsigned auto\_increment not null |
| Тип пользователя | U\_TYPE | enum('Кассир','Директор',  'Администратор') | not null unique |
| Логин пользователя | U\_NAME | char(20) | not null  unique |
| Пароль пользователя | U\_PASS | char(32) | not null  unique |

Таблица 2.22 – «Действие пользователя»

|  |  |  |  |
| --- | --- | --- | --- |
| Содержание поля | Название поля | Тип | Ограничения |
| Код действия пользователя | UA\_ID | int(9) | primary key  unsigned auto\_increment |
| Код пользователя | U\_ID | smallint(3) | unsigned  not null  foreign key  on update cascade on delete cascade |
| Тип действия | UA\_TYPE | enum('Insert','Update',  'Delete','Select') | not null |
| Время действия | UA\_DATETIME | datetime | not null  default current\_timestamp |
| Описание действия | UA\_DESCRIPTION | char(200) | not null |

# 3 ОПИСАНИЕ БД И ПРИЛОЖЕНИЯ ДЛЯ РАБОТЫ С НЕЙ

# 3.1 Описание структуры базы данных

На основе созданной физической модели создаются структуры таблицы базы данных “Аптека” с помощью СУБД MySQL. Созданные таблицы в режиме конструктора представлены на рисунках 3.1 – 3.11

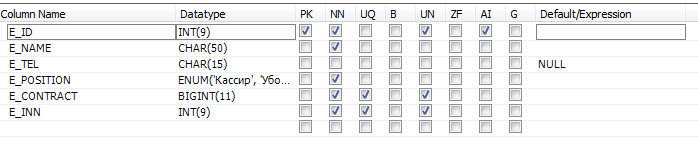


Рисунок 3.1 - Структура таблицы "Работники"

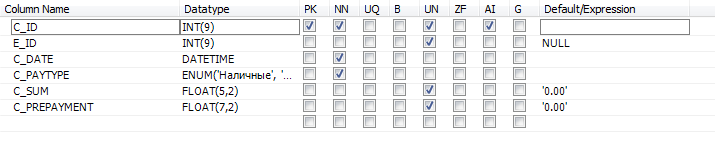


Рисунок 3.2 - Структура таблицы "Чеки"

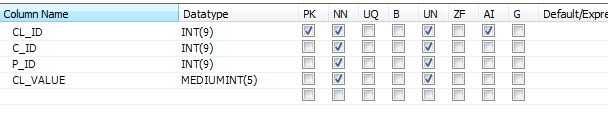


Рисунок 3.3 - Структура таблицы "Список чека"

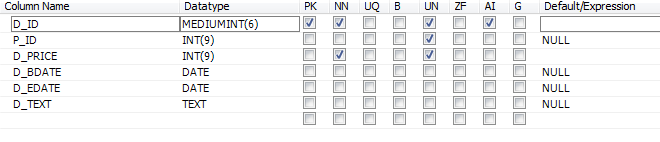


Рисунок 3.4 - Структура таблицы "Скидка"

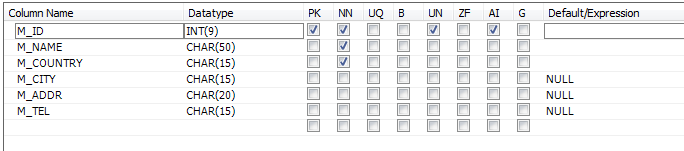


Рисунок 3.5 - Структура таблицы "Производители"

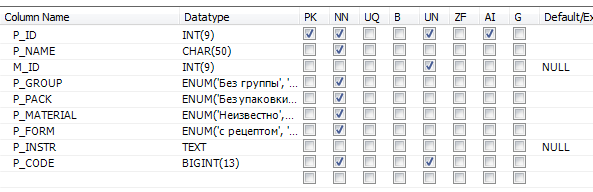


Рисунок 3.6 - Структура таблицы "Товары"

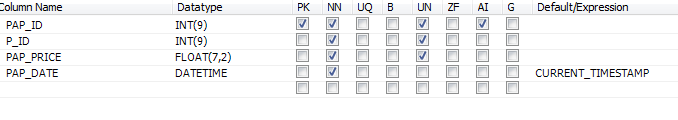


Рисунок 3.7 - Структура таблицы "Акт. цена товара"

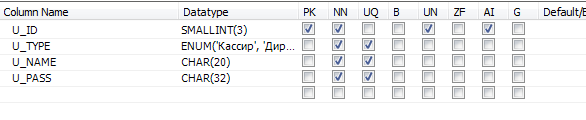


Рисунок 3.8 - Структура таблицы "Пользователи"



Рисунок 3.9 - Структура таблицы "Действия пользователя"

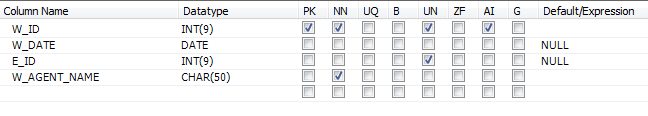


Рисунок 3.10 - Структура таблицы "Накладные"

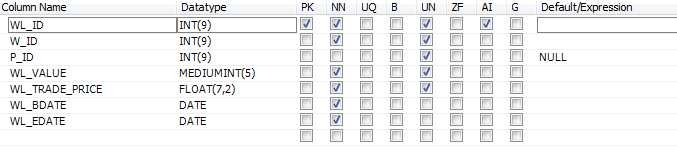


Рисунок 3.11 - Структура таблицы "Список накладной"

Между созданными таблицами были установлены соответствующие связи, которые отображены на рисунке 3.12 в виде схемы данных.

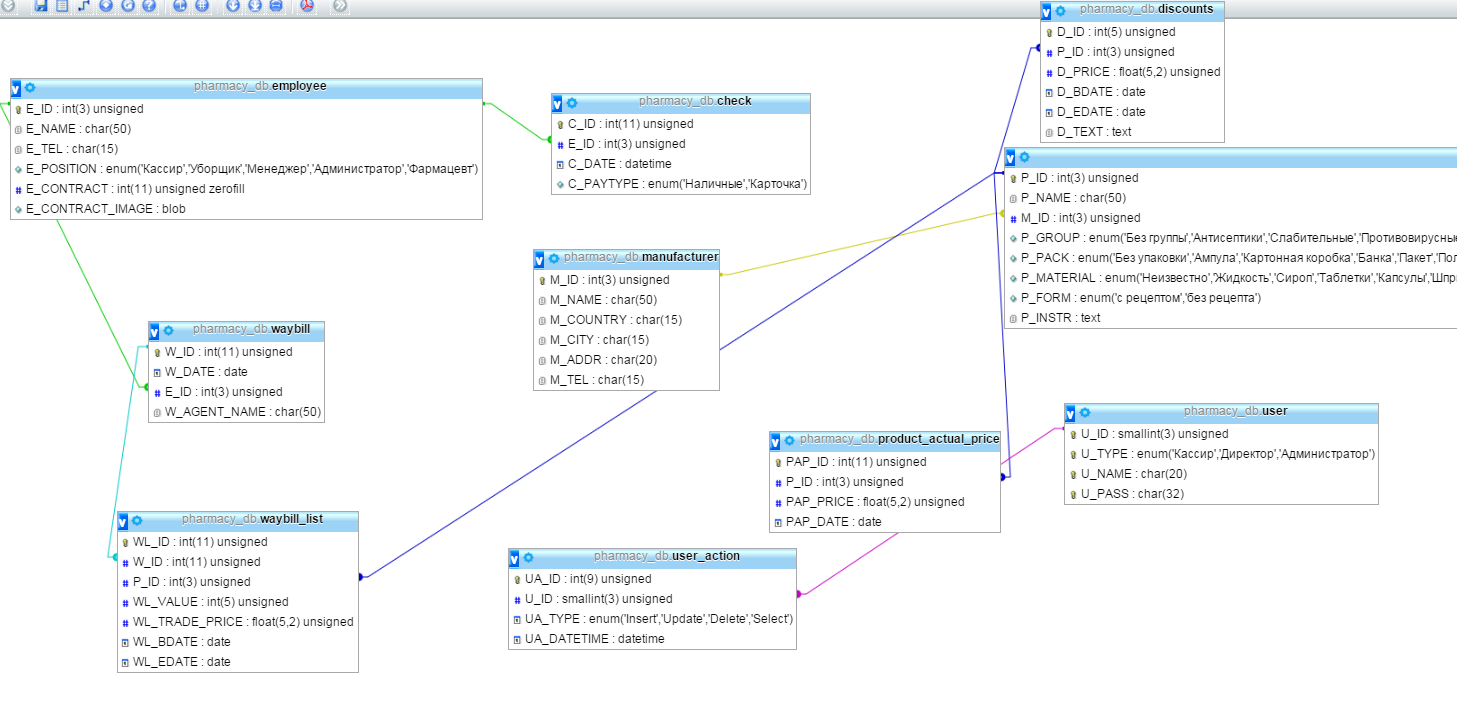


Рисунок 3.12- Схема данных

# 3.2 Создание форм для ввода и отображения данных

Форма — это объект базы данных, который можно использовать для создания пользовательского интерфейса для приложения базы данных.

Для просмотра информации о поставках была создана форма с вкладками (рисунок 3.13).

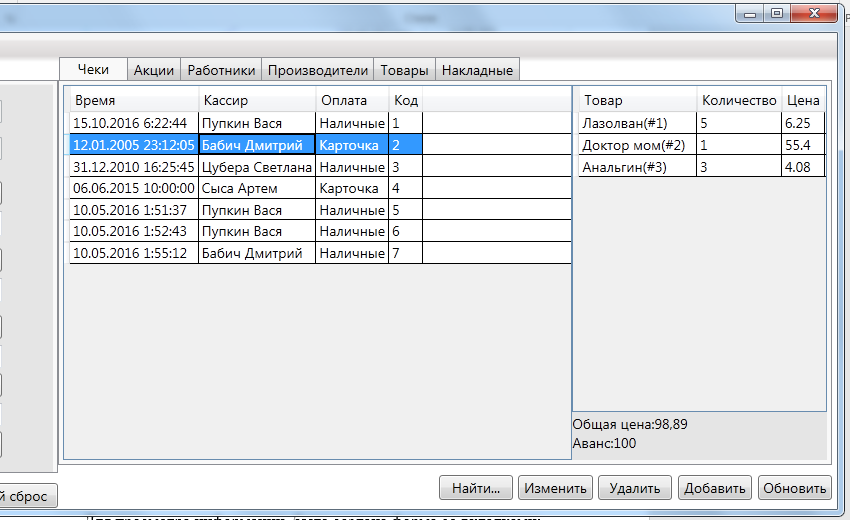


Рисунок 3.13- Форма просмотра

Во вкладке «Чеки» можно просмотреть информацию о чеках, а также о товарах в них. Также эта форма отображает акции для товаров в чеке, сумму чека и аванс (рисунок 3.14).

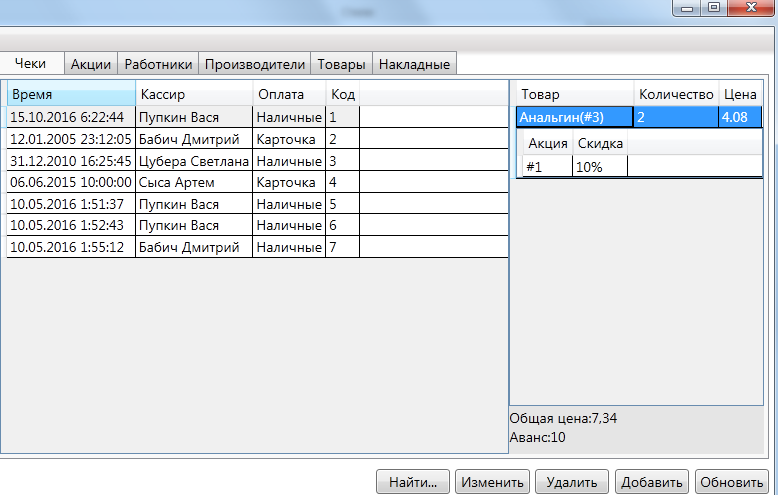


Рисунок 3.14- Вкладка с информацией о чеках

Следующая вкладка предоставляет возможность просмотреть информацию о существующих акциях (рисунок 3.15).

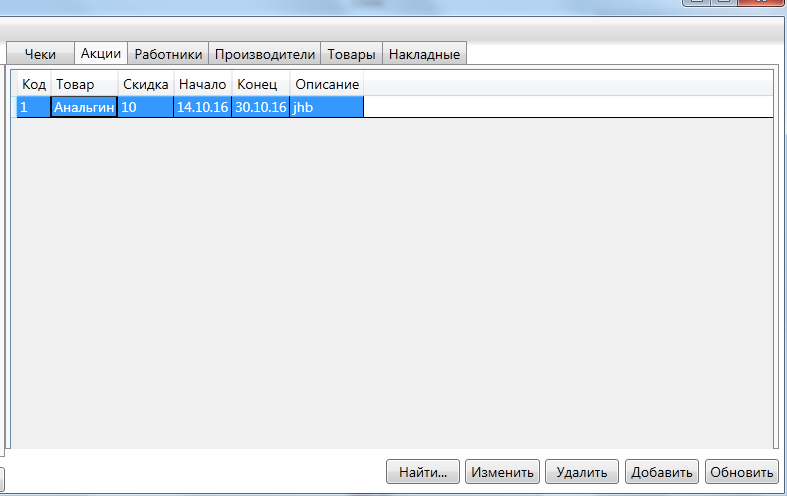


Рисунок 3.15- Вкладка скидок

При выборе складки «Работники» будет отображаться информация о работниках аптеки (рисунок 3.16).

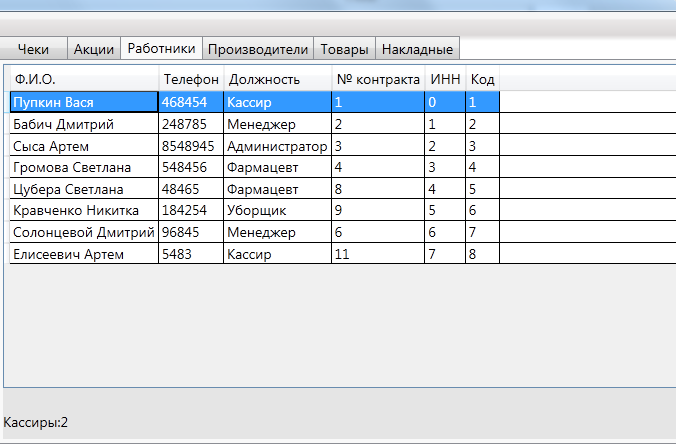


Рисунок 3.16 - Вкладка «Работники»

Для просмотра информации о производителях продаваемых лекарств (рисунок 3.17).

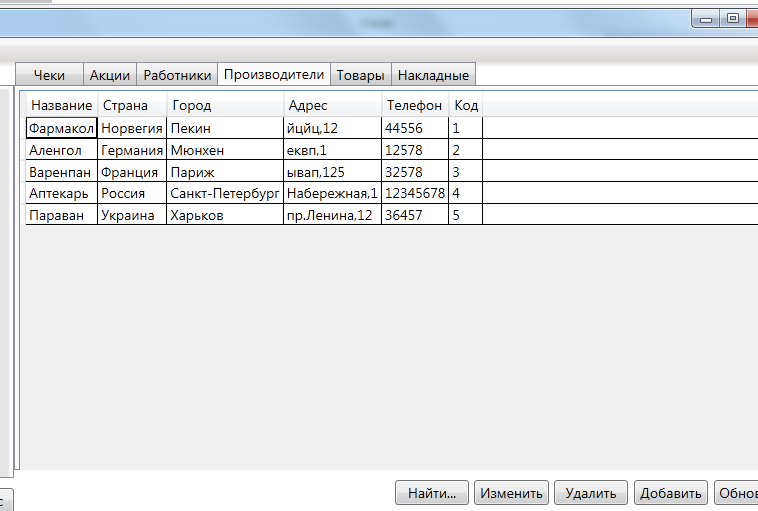


Рисунок 3.17- Вкладка «Производители»

Для просмотра информации о лекарствах, их количестве, цене необходимо выбрать вкладку «Товар» (рисунок 3.18).

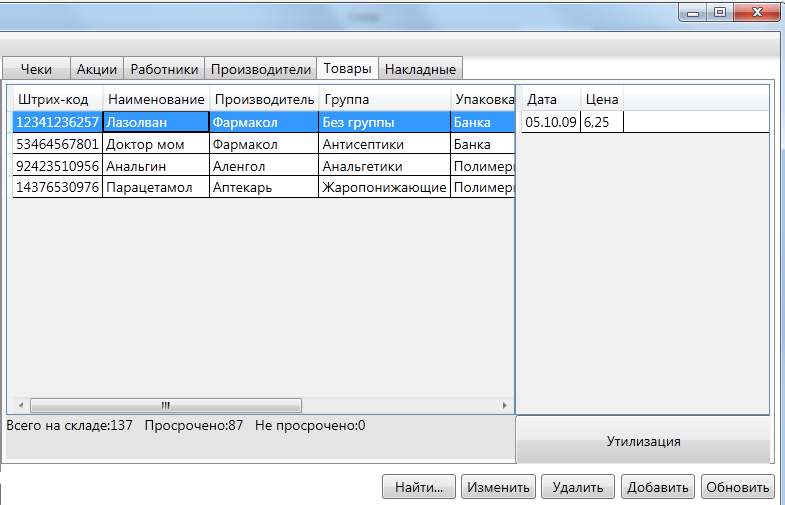


Рисунок 3.18- Вкладка «Товары»

На данной форме можно просмотреть просроченные товары. Для их утилизации необходимо выбрать функцию Удалить. Затем появится вкладка с выбором возможности утилизации (рисунок 3.19).

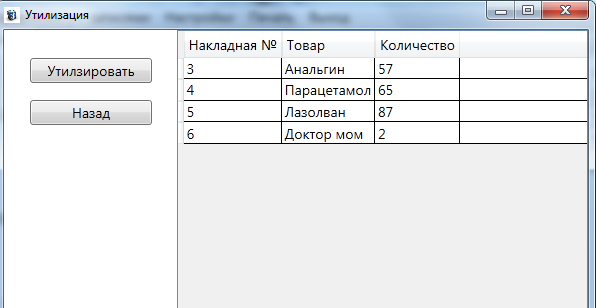


Рисунок 3.19- Вкладка утилизации товара

При нажатии на кнопку “Утилизировать” просроченный товар удаляется из базы данных, при нажатии на кнопку “Назад” мы переходим обратно к форме просмотра.

Во вкладке “Накладные” (рисунок 3.20) можно просмотреть информацию о накладных и товарах в них, а также то, в каком состоянии товары из определенной партии. На форме подсчитывается общая сумма товаров в накладной.

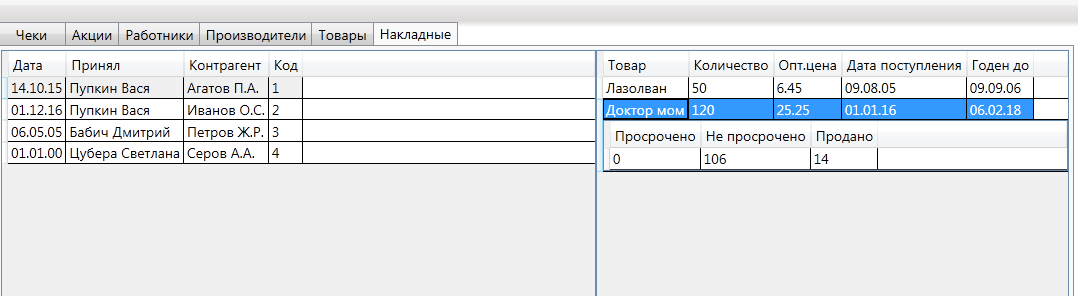


Рисунок 3.20- Вкладка просмотра накладных

Кроме того, на данной форме располагаются кнопки “Найти…”, “Изменить”, “Удалить”, “Добавить”, “Обновить” – эти кнопки действуют определенным образом в зависимости от выбранной вкладки, а именно:

“Обновить” – действует для всех таблиц, ручное обновление информации, в случае надобности можно обновить выбранную вкладку;

“Добавить” – вызывает формы добавления данных в базу в зависимости от выбранной вкладки, о них чуть позже;

“Удалить” – позволяет удалить данные из выбранной таблицы, а именно – удаляется выделенная в таблице просмотра запись. Удаление доступно для акций, работников, производителей, товаров. Для удаления чеков или накладных необходимо ввести пароль в появляющееся диалоговое окно (рисунок 3.21).

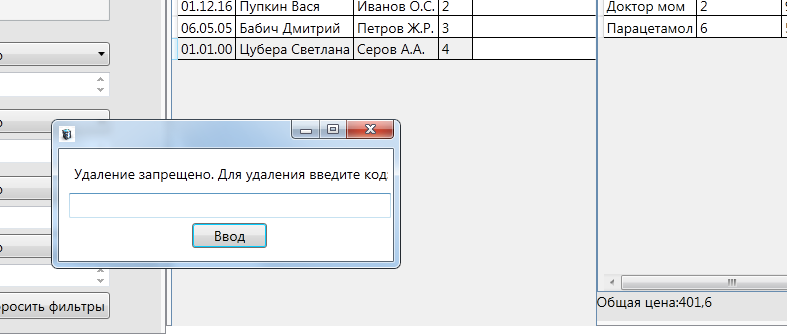


Рисунок 3.21- Вид окна для ввода пароля

“Изменить” – позволяет изменять данные в формах просмотра, и, соответственно, в базе данных, об них будет сказано вместе с формами добавления ниже;

“Найти…” – позволяет осуществить селективный поиск данных, управление диапазонами значений находится в левой части формы на рисунке 3.22.

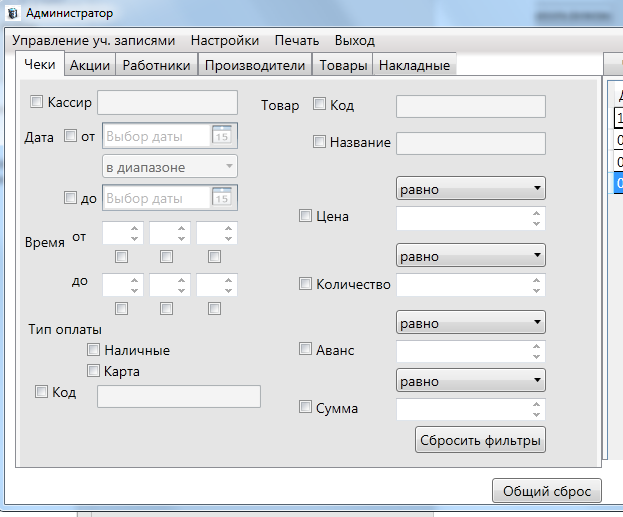


Рисунок 3.22- Селективный поиск

Как видно из рисунка в приложении реализовано множество фильтров: по конкретной дате, по диапазону дат, по типу оплаты, по цене, названию, наличию предоплаты. Кнопка “Сбросить фильтры” возвращает их исходное состояние на данной вкладке, кнопка “Общий сброс” позволяет сбросить все фильтры на всех вкладках.

Для реализации административных функций была создана форма управления учетными записями (рисунок 3.23).

Здесь можно добавлять (рисунок 3.24), изменять (рисунок 3.25), а также удалять пользователей соответственно названными кнопками. Кнопка “Обновить” позволяет вручную обновить список пользователей.

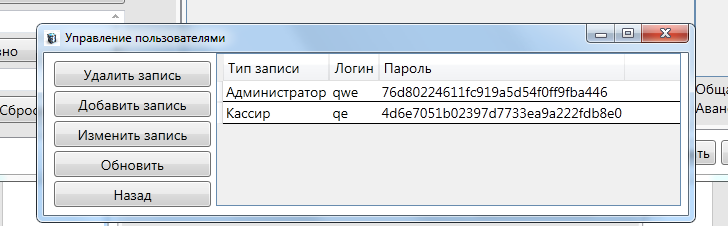


Рисунок 3.23- Управление записями

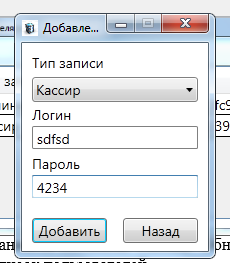


Рисунок 3.24- Добавление пользователя

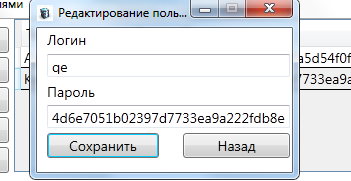


Рисунок 3.25- Изменение пользователя

Для добавления информации о новой акции необходимо использовать соответствующую вкладку – рисунок 3.2: При вводе присутствует проверка введенных значений, например, дата начала не может быть датой конца, нельзя не выбрать товар и т.п.

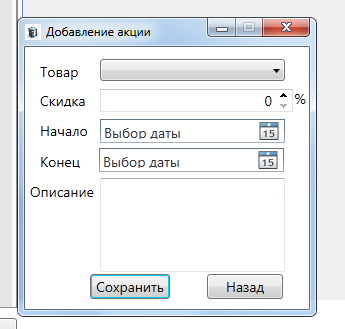


Рисунок 3.26- Добавление акции

Для добавления нового сотрудника используется форма с рисунка 3.27 (пример с неправильным вводом).

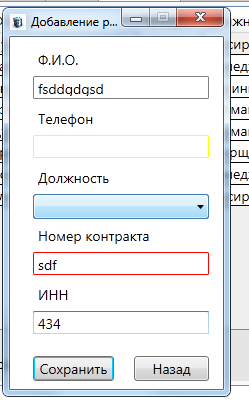


Рисунок 3.27- Добавление работника

Желтым подсвечиваются поля, не обязательные к заполнению, красным – обязательные.

Добавление информации о новом производителя осуществляется при помощи формы, представленной на рисунке 3.28. Если поля подсвечиваются серым, значит ввод был осуществлен правильно.

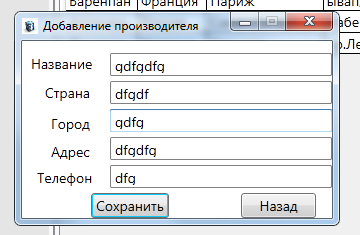


Рисунок 3.28- Добавление производителя

Добавление нового товара, пример нажатия кнопки “Сохранить” при пустых полях (рисунок 3.29).

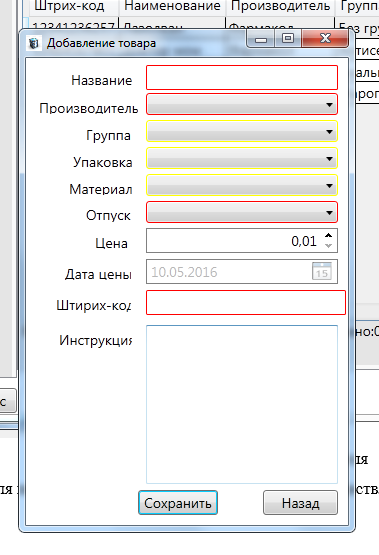


Рисунок 3.29- Добавление товара

Для добавления новой поставки лекарств используется форма «Добавление накладной» (рисунок 3.30).

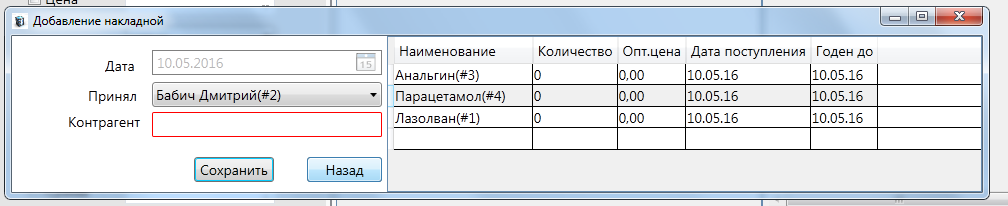


Рисунок 3.30- Добавление накладной

Изменение отличается только занесением уже существующих значений в поля для последующего изменения, если есть такая необходимость.

Кнопка “Сохранить” позволяет добавить или изменить данные, кнопка “Назад” возвращает нас обратно на форму администратора.

Кнопка “Выход” в форме администратора позволяет выйти в окно входа программы (рисунок 3.31).

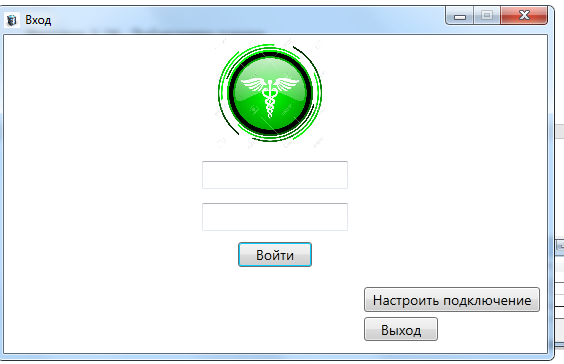


Рисунок 3.31- Окно входа

Для продажи лекарств реализована форма кассира, которая отображена на рисунке 3.32.

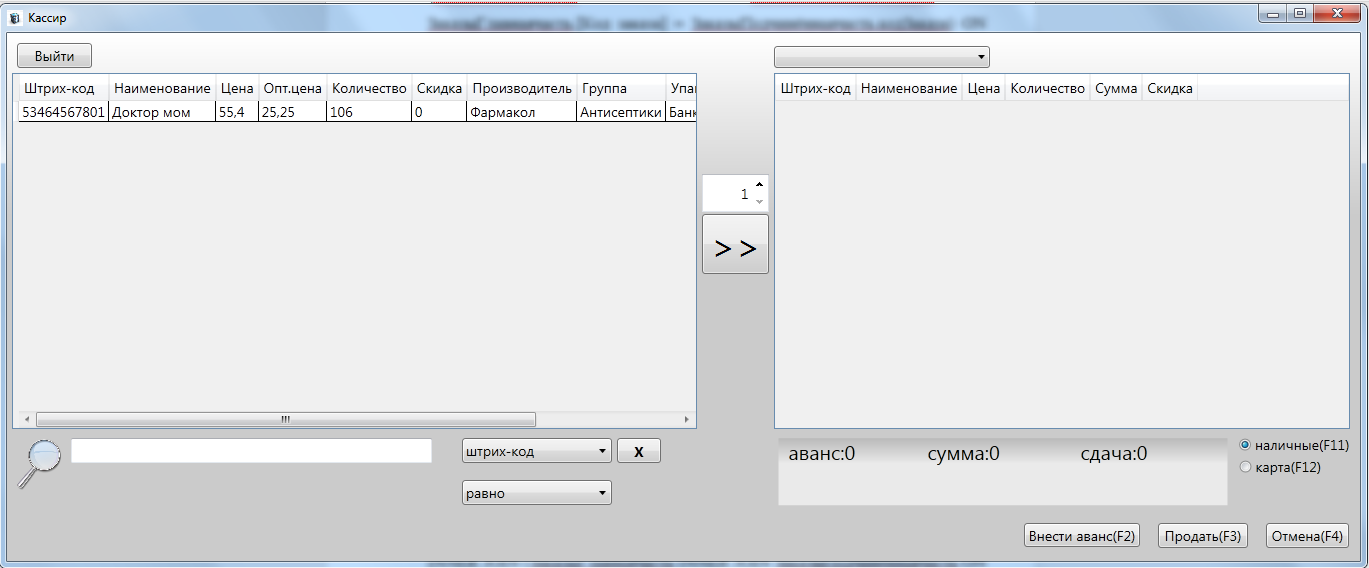


Рисунок 3.32- Окно кассира

В левой части окна находится таблица с информацией о товаре, ниже расположена строка поиска, по которой, выбирая нужный параметр, можно выбирать тот или иной товар. В левую таблицу выводятся только актуальные товары, т.е. не просроченные. Также вы не сможете продать больше чем хранится на складе.

Кнопка “Внести аванс” позволяет внести сумму денег, данную покупателем кассиру. Нельзя ввести сумму, меньшую от общей суммы чека (рисунок 3.33).

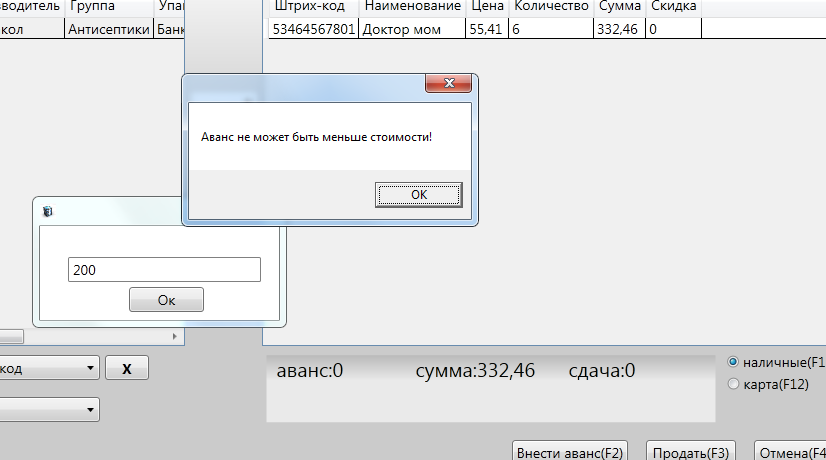


Рисунок 3.33- Окно кассира

Кнопка “Продать” позволяет продать имеющиеся в перечне справа товары покупателю, списав их со склада.

Кнопка “Отмена” позволяет вернуть товары на склад после ввода пароля безопасности.

Кнопка “Выйти” позволяет выйти из окна на окно входа.

Также при продаже возможна печать чека (рисунок 3.34).

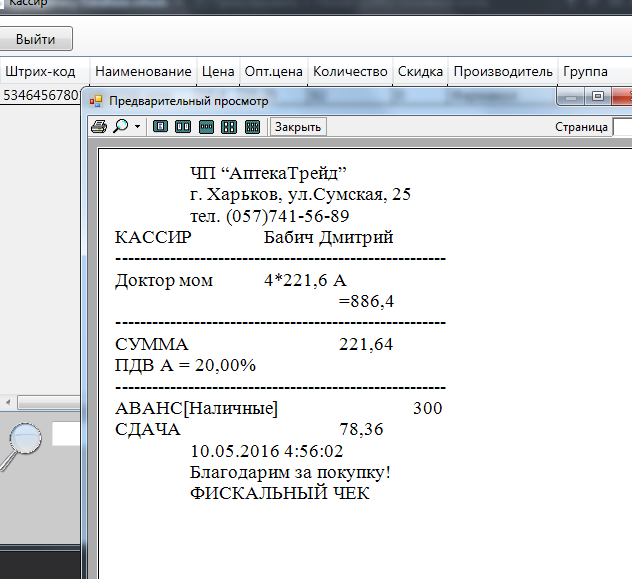


Рисунок 3.34- Печать чека после продажи

Данная функция может быть выполнена и в окне администратора (рисунок 3.35).

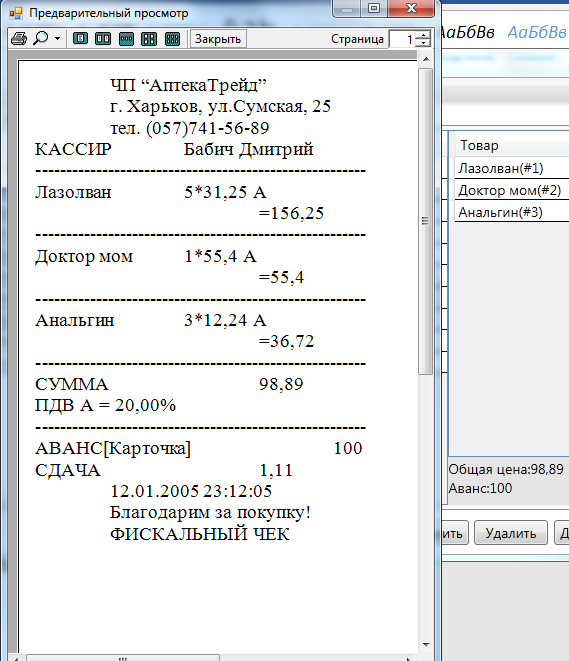


Рисунок 3.35- Печать чека

При необходимости возможна печать накладной в виде, как на рисунке 3.36.

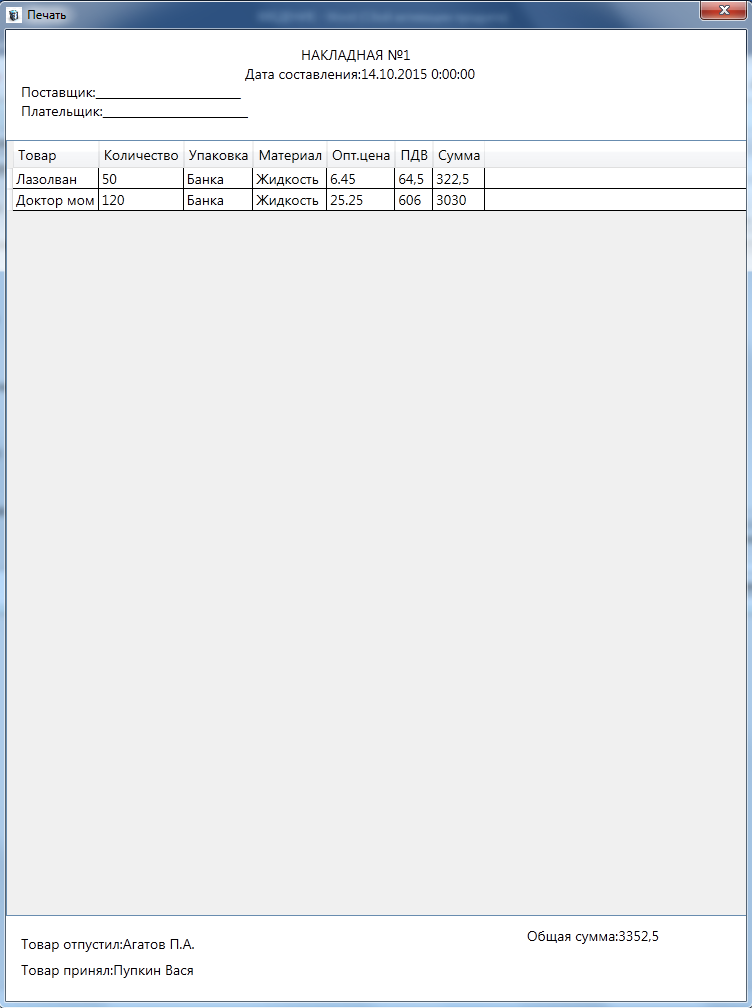


Рисунок 3.36- Печать накладной

# 3.3 Формирование запросов и отчётов

Создание новой записи в акциях, параметры передаются на программном уровне, считанные из форм. В следующих запросах то же самое.

INSERT INTO `discounts` (`P\_ID` ,`D\_PRICE` ,`D\_BDATE` ,`D\_EDATE` ,`D\_TEXT`)VALUES(@\_id,@\_price,@\_bdate,@\_edate,@\_text);

Данный запрос выбирает весь перечень товаров, в определенный элемент управления (рисунок 3.37):

SELECT P\_ID,P\_NAME FROM product;

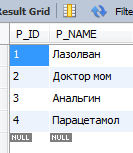


Рисунок 3.37- Результат запроса №1

Создание новой записи работника:

INSERT INTO `employee`(`E\_NAME`,`E\_TEL`,`E\_POSITION`,`E\_CONTRACT`,E\_INN)VALUES(@\_name,@\_tel,@\_pos,@\_contract,@\_inn);

Создание новой записи производителя:

INSERT INTO `manufacturer` (`M\_NAME`, `M\_COUNTRY`, `M\_CITY`, `M\_ADDR` ,`M\_TEL`)VALUES(@\_name,@\_country,@\_city,@\_addr,@\_tel);

Создание новой записи товара:

INSERT INTO `product`(`P\_ID`,`P\_NAME`,`M\_ID`,`P\_GROUP`,`P\_PACK`,`P\_MATERIAL`,`P\_FORM`,`P\_INSTR`,P\_CODE)VALUES(@\_id,@\_name,@\_manufacturer,@\_group,@\_pack,@\_material,@\_form,@\_instr,@\_code);

Создание новой цены продукта и кол-ва:

INSERT INTO `product\_actual\_price`(`P\_ID`,`PAP\_PRICE`,`PAP\_DATE`)VALUES(@\_id,@\_price,@\_date);

INSERT INTO product\_quantity(P\_ID)VALUES(@\_id);

Выборка всех производителей в какой-то элемент управления, результат отображен на рисунке 3.38:

SELECT M\_ID,M\_NAME FROM manufacturer;

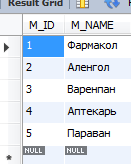


Рисунок 3.38- Результат запроса №2

Выборка текущей даты:

SELECT date(now()); результат – рисунок 3.39



Рисунок 3.39- Результат запроса №3

Создание списка накладной:

INSERT INTO `waybill\_list`(`W\_ID`, `P\_ID`, `WL\_VALUE`, `WL\_TRADE\_PRICE`, `WL\_BDATE`, `WL\_EDATE`)VALUES

(ЗНАЧЕНИЯ)…..

Создание накладной:

INSERT INTO `waybill`(W\_ID,W\_DATE,E\_ID,W\_AGENT\_NAME)VALUES(@\_id,@\_date,@\_employee,@\_agent);

Выборка кода списка накладной (рисунок 3.40):

SELECT WL\_ID FROM waybill,waybill\_list WHERE waybill.W\_ID=waybill\_list.W\_ID AND [waybill.W\_ID=@\_id](mailto:waybill.W_ID=@_id);



Рисунок 3.40- Результат запроса №4

Обновление состояния продукта:

UPDATE product\_overdue po,waybill\_list wl SET po.PP\_IS\_OVERDUE=IF((wl.WL\_EDATE-14)>DATE(NOW())-0,IF((SELECT wl.WL\_VALUE-product\_sold.PS\_COUNT FROM product\_sold WHERE product\_sold.WL\_ID=wl.WL\_ID)=0,'Продано','Не просрочено'),IF((WL\_EDATE)<DATE(NOW()),IF((SELECT wl.WL\_VALUE-product\_sold.PS\_COUNT FROM product\_sold WHERE product\_sold.WL\_ID=wl.WL\_ID)=0,'Продано','Просрочено'),'Скоро истекает срок годности'))WHERE po.PP\_IS\_OVERDUE<>'Просрочено' AND po.PP\_IS\_OVERDUE<>'Продано' AND po.WL\_ID=wl.WL\_ID AND po.PP\_ID>0 AND wl.WL\_ID>0;

Обновление количества продукта:

UPDATE product\_quantity SET PQ\_IN=PQ\_IN+@\_value WHERE P\_ID=@\_id;

Выборка работников (рисунок 3.41):

SELECT E\_ID,E\_NAME FROM employee;

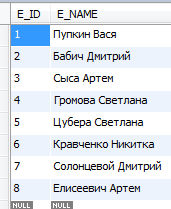


Рисунок 3.41- Результат запроса №5

Выборка пользователей для проверки (рисунок 3.42):

SELECT U\_TYPE FROM `user` WHERE U\_ID=3;

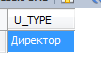


Рисунок 3.42- Результат запроса №6

Выбор суммы чека (рисунок 3.43):

SELECT C\_SUM FROM `check` where C\_ID=@\_curid;



Рисунок 3.43- Результат запроса №7

Выбор аванса чека (рисунок 3.44):

SELECT C\_PREPAYMENT FROM `check` where C\_ID=@\_curid;



Рисунок 3.44- Результат запроса №8

Количество товара на складе (рисунок 3.45):

SELECT IFNULL(SUM(waybill\_list.WL\_VALUE-product\_sold.PS\_COUNT),0) FROM product\_sold,waybill\_list WHERE product\_sold.WL\_ID=waybill\_list.WL\_ID AND [waybill\_list.P\_ID=@\_id](mailto:waybill_list.P_ID=@_id);



Рисунок 3.45- Результат запроса №9

Из него просроченного (рисунок 3.46):

SELECT IFNULL(sum(waybill\_list.WL\_VALUE-product\_sold.PS\_COUNT),0) FROM waybill\_list,product\_sold,product\_overdue WHERE product\_overdue.PP\_IS\_OVERDUE='Просрочено' AND product\_sold.WL\_ID=waybill\_list.WL\_ID AND product\_overdue.WL\_ID=waybill\_list.WL\_ID AND waybill\_list.P\_ID=1;



Рисунок 3.46- Результат запроса №10

Не просроченного (рисунок 3.47):

SELECT IFNULL(sum(waybill\_list.WL\_VALUE-product\_sold.PS\_COUNT),0) FROM waybill\_list,product\_sold,product\_overdue WHERE product\_overdue.PP\_IS\_OVERDUE='Не просрочено' AND product\_sold.WL\_ID=waybill\_list.WL\_ID AND product\_overdue.WL\_ID=waybill\_list.WL\_ID AND [waybill\_list.P\_ID=@\_id](mailto:waybill_list.P_ID=@_id);



Рисунок 3.47 - Результат запроса №11

Общая цена накладной (рисунок 3.48):

SELECT SUM(waybill\_list.WL\_TRADE\_PRICE\*waybill\_list.WL\_VALUE)

FROM waybill\_list WHERE waybill\_list.W\_ID=@\_curid;

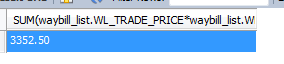


Рисунок 3.48 - Результат запроса №12

Количество работников одного типа (рисунок 3.49):

SELECT COUNT(E\_ID) FROM employee WHERE E\_POSITION=@\_pos;



Рисунок 3.49 - Результат запроса №13

Выборка данных для формы чеков (рисунок 3.50):

SELECT distinct product.P\_ID,CONCAT( product.P\_NAME,"(#",product.P\_ID,")") AS 'P\_NAME',cl.CL\_VALUE,product\_actual\_price.PAP\_PRICE FROM check\_list cl,product,`check` c,product\_actual\_price WHERE cl.P\_ID=product\_actual\_price.P\_ID AND cl.P\_ID=product.P\_ID AND cl.C\_ID=c.C\_ID AND product\_actual\_price.PAP\_PRICE=ifnull((select pap.PAP\_PRICE from product\_actual\_price pap where pap.P\_ID=cl.P\_ID and pap.PAP\_DATE<=c.C\_DATE order by pap.PAP\_DATE desc,pap.PAP\_PRICE desc limit 1),(select pap.PAP\_PRICE from product\_actual\_price pap where pap.P\_ID=cl.P\_ID order by pap.PAP\_DATE desc,pap.PAP\_PRICE desc limit 1)) AND c.C\_ID=1;



Рисунок 3.50 - Результат запроса №14

Выборка данных для формы накладных (рисунок 3.51):

SELECT product.P\_ID,product.P\_NAME,waybill\_list.WL\_VALUE,waybill\_list.WL\_TRADE\_PRICE,waybill\_list.WL\_BDATE,waybill\_list.WL\_EDATE FROM `waybill`,`waybill\_list`,`product` WHERE `waybill\_list`.`W\_ID`=`waybill`.`W\_ID` AND `waybill\_list`.`P\_ID`=`product`.`P\_ID` AND waybill\_list.W\_ID=1;

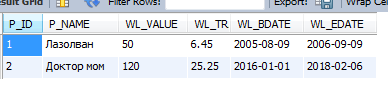


Рисунок 3.51 - Результат запроса №15

Удаление чека:

DELETE FROM `check` WHERE C\_ID=@\_id;

Удаление скидки:

DELETE FROM `discounts` WHERE D\_ID=@\_id

Удаление работника:

DELETE FROM `employee` WHERE E\_ID=@\_id

Удаление производителя:

DELETE FROM `manufacturer` WHERE M\_ID=@\_id

Удаление товара:

DELETE FROM `product` WHERE P\_ID=@\_id

Выборка товара по производителю (рисунок 3.52):

SELECT `product`.`P\_ID`,`product`.`P\_NAME`,`product`.`M\_ID`,`product`.`P\_GROUP`,`product`.`P\_PACK`,`product`.`P\_MATERIAL`,`product`.`P\_FORM`,`product`.`P\_INSTR`,`product`.`P\_CODE` FROM `product` WHERE M\_ID=@\_id;

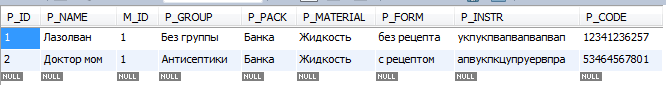


Рисунок 3.52 - Результат запроса №16

Выборка чеков по работнику (рисунок 3.53):

SELECT `check`.`C\_ID`,`check`.`C\_DATE`,`check`.`C\_PAYTYPE`,`check`.`E\_ID`,`check`.`C\_SUM`,`check`.`C\_PREPAYMENT` FROM `check` WHERE E\_ID=@\_id

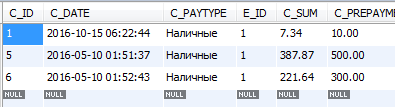


Рисунок 3.53 - Результат запроса №17

Выборка накладной по работнику (рисунок 3.54):

SELECT `waybill`.`W\_ID`,`waybill`.`W\_DATE`,`waybill`.`E\_ID`,`waybill`.`W\_AGENT\_NAME` FROM `waybill` WHERE E\_ID=1;

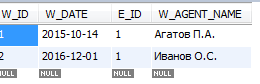


Рисунок 3.54 - Результат запроса №18

Для поиска применяется конструктор запросов, вот пример нескольких его строк:

AND `check`.C\_DATE>=@\_bdate","AND `check`.C\_DATE<=@\_edate"," AND `check`.C\_DATE BETWEEN @\_bdate AND @\_edate"," AND (`check`.C\_DATE>@\_edate OR `check`.C\_DATE<@\_bdate)

" AND HOUR(`check`.C\_DATE)>=@\_bhour"," AND HOUR(`check`.C\_DATE)<=@\_ehour"," AND HOUR(`check`.C\_DATE) BETWEEN @\_bhour AND @\_ehour"

SELECT DISTINCT `check`.C\_ID,`check`.C\_DATE,`check`.C\_PAYTYPE,`employee`.E\_NAME,`check`.C\_SUM,`check`.C\_PREPAYMENT FROM `check`,`employee`,check\_list,product,product\_actual\_price WHERE `check`.`E\_ID`=`employee`.`E\_ID` AND `check`.C\_ID=check\_list.C\_ID AND check\_list.P\_ID=product.P\_ID AND check\_list.P\_ID=product\_actual\_price.P\_ID

" AND (SELECT SUM(waybill\_list.WL\_VALUE-product\_sold.PS\_COUNT) FROM waybill\_list,product\_sold WHERE waybill\_list.P\_ID=p.P\_ID AND waybill\_list.WL\_ID=product\_sold.WL\_ID)",

" AND (SELECT waybill\_list.WL\_VALUE-product\_sold.PS\_COUNT FROM waybill\_list,product\_sold,product\_overdue WHERE product\_overdue.PP\_IS\_OVERDUE='Просрочено' AND product\_sold.WL\_ID=waybill\_list.WL\_ID AND product\_overdue.WL\_ID=waybill\_list.WL\_ID AND waybill\_list.P\_ID=p.P\_ID)",

" AND (SELECT IF(COUNT(waybill\_list.WL\_VALUE-product\_sold.PS\_COUNT)>0,waybill\_list.WL\_VALUE-product\_sold.PS\_COUNT,0) FROM product\_sold,product\_overdue,waybill\_list WHERE product\_sold.WL\_ID=product\_overdue.WL\_ID AND product\_overdue.PP\_IS\_OVERDUE<>'Просрочено' AND product\_overdue.PP\_IS\_OVERDUE<>'Продано' AND product\_sold.WL\_ID=waybill\_list.WL\_ID AND waybill\_list.P\_ID=p.P\_ID)"

Выборка данных акций для товаров в форме чека (рисунок 3.55):

SELECT CONCAT('#',D\_ID) AS 'D\_ID',CONCAT(D\_PRICE,'%') AS 'D\_PRICE' FROM discounts,check\_list,`check` WHERE discounts.P\_ID=check\_list.P\_ID AND `check`.C\_ID=@\_cid AND check\_list.P\_ID=@\_id AND check\_list.C\_ID=`check`.C\_ID AND `check`.C\_DATE>=discounts.D\_BDATE AND `check`.C\_DATE<=discounts.D\_EDATE;

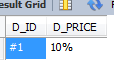
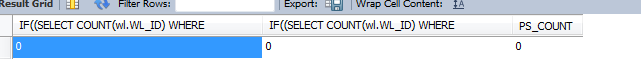


Рисунок 3.55 - Результат запроса №19

Выборка данных для формы накладных для отображения состояния и количества товара в этом состоянии (рисунок 3.56):

SELECT IF((SELECT COUNT(wl.WL\_ID) WHERE po.PP\_IS\_OVERDUE='Просрочено')>0,wl.WL\_VALUE-ps.PS\_COUNT,0),IF((SELECT COUNT(wl.WL\_ID) WHERE po.PP\_IS\_OVERDUE='Не просрочено')>0,wl.WL\_VALUE-ps.PS\_COUNT,0),ps.PS\_COUNT FROM waybill\_list wl,product\_sold ps,product\_overdue po WHERE ps.WL\_ID=wl.WL\_ID AND po.WL\_ID=wl.WL\_ID AND wl.W\_ID=1 AND wl.P\_ID=1;

Рисунок 3.56 - Результат запроса №20

Выборка суммы и аванса чека для печати (рисунок 3.57):

SELECT C\_PREPAYMENT,C\_SUM FROM `check` WHERE C\_ID=@\_id

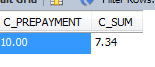


Рисунок 3.57 - Результат запроса №21

Выборка самого чека и связанных с ним данных для печати (рисунок 3.58):

SELECT distinct product.P\_ID,product.P\_NAME,cl.CL\_VALUE,round(ifnull((select pap.PAP\_PRICE from product\_actual\_price pap where pap.P\_ID=cl.P\_ID and pap.PAP\_DATE<=c.C\_DATE order by pap.PAP\_DATE desc,pap.PAP\_PRICE desc limit 1),(select pap.PAP\_PRICE from product\_actual\_price pap where pap.P\_ID=cl.P\_ID order by pap.PAP\_DATE desc,pap.PAP\_PRICE desc limit 1))\*cl.CL\_VALUE\*(select if((select count(d.D\_ID))>0,exp(sum(log(1-d.D\_PRICE\*0.01))),1) from discounts d where d.P\_ID=cl.P\_ID and d.D\_BDATE<=c.C\_DATE and d.D\_EDATE>=c.C\_DATE),2),(SELECT if((SELECT COUNT(d.D\_ID))>0,d.D\_PRICE,0) from discounts d where d.P\_ID=cl.P\_ID) FROM check\_list cl,product,`check` c,product\_actual\_price WHERE cl.P\_ID=product\_actual\_price.P\_ID AND cl.P\_ID=product.P\_ID AND cl.C\_ID=c.C\_ID AND product\_actual\_price.PAP\_PRICE=ifnull((select pap.PAP\_PRICE from product\_actual\_price pap where pap.P\_ID=cl.P\_ID and pap.PAP\_DATE<=c.C\_DATE order by pap.PAP\_DATE desc,pap.PAP\_PRICE desc limit 1),(select pap.PAP\_PRICE from product\_actual\_price pap where pap.P\_ID=cl.P\_ID order by pap.PAP\_DATE desc,pap.PAP\_PRICE desc limit 1)) AND c.C\_ID=1

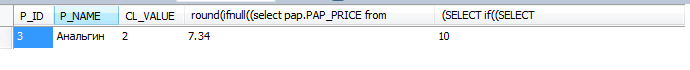


Рисунок 3.58 - Результат запроса №22

Выборка данных для печати накладной (рисунок 3.59):

SELECT product.P\_NAME,waybill\_list.WL\_VALUE,waybill\_list.WL\_TRADE\_PRICE,waybill\_list.WL\_BDATE,waybill\_list.WL\_EDATE,product.P\_PACK,product.P\_MATERIAL FROM `waybill`,`waybill\_list`,`product` WHERE `waybill\_list`.`W\_ID`=`waybill`.`W\_ID` AND `waybill\_list`.`P\_ID`=`product`.`P\_ID` AND waybill\_list.W\_ID=1;

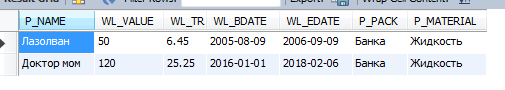
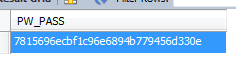


Рисунок 3.59 - Результат запроса №23

Выборка пароля директора (рисунок 3.60):

SELECT PW\_PASS FROM passwords WHERE PW\_TYPE='1';

 Рисунок 3.60 - Результат запроса №24

Утилизация товара если он просрочен:

UPDATE product\_overdue SET PP\_IS\_OVERDUE='Продано';

Создание нового пользователя:

INSERT INTO `user` (`U\_TYPE`,`U\_NAME`,`U\_PASS`) VALUES (@\_type, @\_login, md5(@\_pass));

Выборка пользователя по типу (рисунок 3.61):

SELECT U\_NAME,U\_PASS FROM `user` WHERE U\_TYPE=@\_curtype;



Рисунок 3.61 - Результат запроса №25

Изменение пользователя по типу:

UPDATE `user` SET `U\_NAME`=@\_login, `U\_PASS`=md5(@\_password) WHERE `U\_TYPE`=@\_type;

Удаление пользователя:

DELETE FROM `user` WHERE U\_NAME=@\_login AND U\_PASS=@\_password;

Изменение скидки:

UPDATE discounts SET P\_ID = @\_id,D\_PRICE = @\_price,D\_BDATE = @\_bdate,D\_EDATE = @\_edate,D\_TEXT = @\_text WHERE D\_ID = @\_curid;

Изменение работника:

UPDATE employee SET E\_NAME = @\_name,E\_TEL = @\_tel,E\_POSITION = @\_pos,E\_CONTRACT = @\_contract,E\_INN = @\_inn WHERE E\_ID = @\_curid;

Изменение производителя:

UPDATE manufacturer SET M\_NAME = @\_name,M\_COUNTRY = @\_country,M\_CITY = @\_city,M\_ADDR = @\_addr,M\_TEL = @\_tel WHERE M\_ID = @\_curid;

Проверка даты на обновление цены продукта (рисунок 3.62):

SELECT IF((SELECT DATE\_ADD(pap.PAP\_DATE, INTERVAL 7 DAY) FROM product\_actual\_price pap WHERE pap.PAP\_ID=p.PAP\_ID)>=now(),(DATE\_ADD(p.PAP\_DATE, INTERVAL 7 DAY)),0),now() FROM product\_actual\_price p WHERE p.PAP\_ID=(SELECT MAX(PAP\_ID) FROM product\_actual\_price WHERE P\_ID=p.P\_ID) AND [p.P\_ID=@\_id](mailto:p.P_ID=@_id);

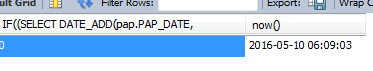


Рисунок 3.62 - Результат запроса №26

Изменение товара:

UPDATE product SET P\_NAME = @\_name,M\_ID = @\_manufacturer,P\_GROUP = @\_group,P\_PACK = @\_pack,P\_MATERIAL = @\_material,P\_FORM = @\_form,P\_INSTR = @\_instr,P\_CODE=@\_code WHERE P\_ID = @\_id;

Создание паролей при первом запуске:

INSERT INTO passwords(PW\_PASS,PW\_TYPE)VALUES(@\_dpass,1)

INSERT INTO passwords(PW\_PASS,PW\_TYPE)VALUES(@\_spass,2)

INSERT INTO `user`(U\_NAME,U\_PASS,U\_TYPE)VALUES(@\_login,@\_pass,'Директор');

Проверка пароля безопасности (рисунок 3.63):

SELECT PW\_PASS FROM passwords WHERE PW\_TYPE='2';



Рисунок 3.63 - Результат запроса №27

Проверка существования администратора при первом запуске (рисунок 3.64):

SELECT IFNULL(COUNT(U\_ID),0) FROM `user` WHERE U\_TYPE='Администратор';



Рисунок 3.64 - Результат запроса №28

Проверка авторизации (рисунок 3.65):

SELECT U\_TYPE FROM user WHERE U\_NAME = @\_userName AND U\_PASS=@\_userPassword



Рисунок 3.65 - Результат запроса №29

Выборка работников (рисунок 3.66):

SELECT E\_ID,E\_NAME,E\_TEL,E\_POSITION,E\_CONTRACT,E\_INN FROM `employee`;

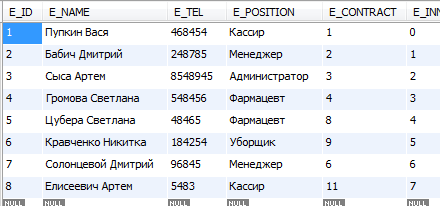


Рисунок 3.66 - Результат запроса №30

Выборка чеков (рисунок 3.67):

SELECT `check`.C\_ID,`check`.C\_DATE,`check`.C\_PAYTYPE,`employee`.E\_NAME FROM `check`,`employee` WHERE `check`.`E\_ID`=`employee`.`E\_ID`;

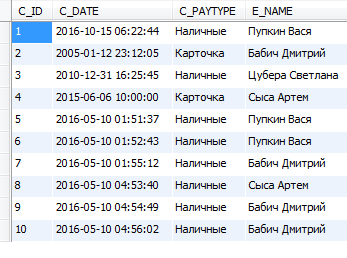


Рисунок 3.67 - Результат запроса №31

Выборка акций (рисунок 3.68):

SELECT discounts.D\_ID,product.P\_NAME,discounts.D\_PRICE,discounts.D\_BDATE,discounts.D\_EDATE,discounts.D\_TEXT FROM `discounts`,`product` WHERE `discounts`.`P\_ID`=`product`.`P\_ID`;



Рисунок 3.68 - Результат запроса №32

Выборка производителей (рисунок 3.69):

SELECT M\_ID,M\_NAME,M\_COUNTRY,M\_CITY,M\_ADDR,M\_TEL FROM manufacturer;

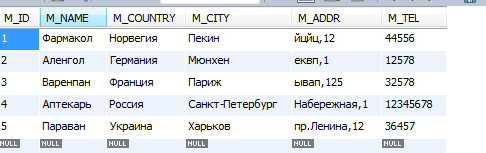


Рисунок 3.69 - Результат запроса №33

Выборка товаров (рисунок 3.70):

SELECT product. P\_ID ,product P\_NAME ,manufacturer .M\_NAME,product .P\_GROUP,product.P\_PACK,product.P\_MATERIAL,product.P\_FORM,product.P\_INSTR,product.P\_CODE FROM `product`,`manufacturer` WHERE `manufacturer` .`M\_ID `= `product`.`M\_ID`;

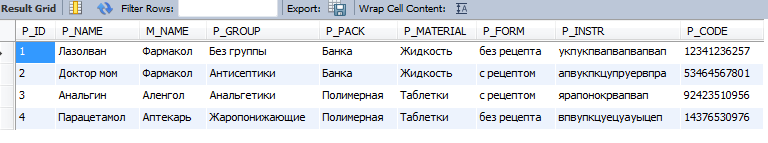


Рисунок 3.70 - Результат запроса №34

Выборка актуальной цены (рисунок 3.71):

SELECT product\_actual\_price.PAP\_PRICE,product\_actual\_price.PAP\_DATE,PAP\_ID FROM `product`,`product\_actual\_price` WHERE `product\_actual\_price`.`P\_ID`=`product`.`P\_ID` AND `product\_actual\_price`.`P\_ID`=@\_curid;



Рисунок 3.71 - Результат запроса №35

Выборка накладных (рисунок 3.72):

SELECT waybill.W\_ID,waybill.W\_DATE,employee.E\_NAME,waybill.W\_AGENT\_NAME FROM `waybill`,`employee` WHERE `waybill`.`E\_ID`=`employee`.`E\_ID`;

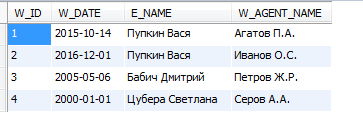


Рисунок 3.72 - Результат запроса №36

Выборка пользователей для просмотра (рисунок 3.73):

SELECT U\_TYPE,U\_NAME,U\_PASS FROM `user` WHERE U\_TYPE<>'Директор';

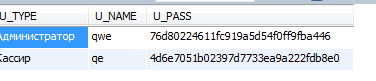


Рисунок 3.73 - Результат запроса №37

Выборка товаров для утилизации на экран (рисунок 3.74):

SELECT wl.WL\_ID,p.P\_NAME,IF((wl.WL\_VALUE-ps.PS\_COUNT)>0,wl.WL\_VALUE-ps.PS\_COUNT,0) FROM waybill\_list wl,product p,product\_sold ps,product\_overdue po WHERE wl.P\_ID=p.P\_ID AND wl.WL\_ID=ps.WL\_ID AND po.PP\_IS\_OVERDUE='Просрочено' AND po.WL\_ID=wl.WL\_ID;

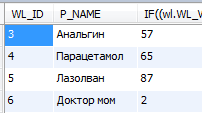


Рисунок 3.74 - Результат запроса №38

# 3.4 Описание интерфейса базы данных

При первом включении программы всплывает окно с предложением ввести пароли безопасности, а также параметры подключения. Далее необходимо создать аккаунт администратора и после корректного ввода приложение переходит в форму управления пользователями.

Затем нам открывается стандартное окно входа, где есть 2 поля ввода – для логина и для пароля, а также есть кнопка “Выход”, которая позволяет закрыть приложение. “Настройки подключения” позволяют изменять параметры подключения, а именно адрес, базу данных, логин и пароль пользователя MySQL “По умолчанию” в настройках подключения позволяет поставить стандартное значение для этих параметров.

Приложение предлагает войти как: кассир, администратор и суперпользователь.

Кассиру доступно окно кассира, где он может продавать товар покупателям, искать товар по критериям. Более подробно: кнопка “Внести аванс” позволяет внести кол-во денег, которые покупатель дал для оплаты. Кнопка “Отмена” позволяет отменить выбор товара в чек и откатить его, но для этого требуется код безопасности, который можно ввести на всплывающей форме. Кнопка “Продать” позволяет продать этот товар покупателю, эта операция будет учтена и внесется в базу данных, впрочем, как и обновленное количество товара. Для выбора товара в таблицу чека(справа) необходимо нажать на кнопку между двумя таблицами, над ней есть счетчик – он позволяет взять нужное количество товара со склада, причем нельзя взять больше чем есть на складе. В таблице слева можно осуществлять интерактивный поиск по 1 параметру, который выбирается элементом управления под нижним правым угол этой таблицы. Также там есть кнопа Х – эта кнопка позволяет сбросить фильтр и вернуть таблицу в изначальное положение. Справа снизу показывается аванс – кол-во денег, которые дает покупатель, сумма чека – общая сумма всех товаров в правой таблице, а также сдача, которую кассир обязан выдать покупателю. Можно выбирать, каким способом покупатель собирается оплачивать покупку. Также на форме, как и в большинстве мест данного приложения работают выбор по нажатию клавиш – в форме кассира это явно видно, элементы управления подписаны. Посередине вверху находится выбор кассира – если вы не выбрали себя и оставили поле пустым, вы не сможете совершать операции до тех пор, пока не выберете свое (или еще чье-то имя). Также при совершении каждой операции требуется подтверждение. После совершения операции также возможно распечатать чек, для чего появится соответствующее окно.

Администратору доступна форма администратора, на которой находятся элементы управления для просмотра данных, а также для селективного поиска по любым (или почти любым) параметрам, касающимся выбранной информации. Подробнее: в правом нижнем углу есть 5 кнопок: “Найти…”, “Изменить”, “Удалить”, “Добавить” и “Обновить”. Кнопка “Найти…” позволяет выполнять селективный поиск по выбранным критериям, элементы управления для задания поиска находятся слева формы и количество вкладок там соответствует кол-ву вкладок просмотра справа.

Кнопки “Сбросить фильтры” а также “Общий сброс” позволяют возвращать фильтры в исходное положение, т.е. в выключенное состояние. Кнопка “Изменить” позволяет вызвать форму изменения выбранной информации, если таковая имеется, иначе ничего не произойдет. Кнопка “Добавить” позволяет вызвать формы добавления для выбранной вкладки. Кнопка “Обновить” позволяет вручную обновить отображаемую информацию, если что-то пошло не так (а это возможно, так как изготовитель данного приложения работал без лицензии и, что самое важное – без зарплаты).

Вкладки. Подробнее: Вкладка “Чеки” позволяет просмотреть информацию о чеках, а именно: кассира, перечень товаров и скидки к этим товарам, которые видны при нажатии на конкретный товар, а также подсчитывается общая сумма и выводится аванс данного чека, т.е. сколько покупатель дал денег.

Вкладка “Акции” позволяет просмотреть информацию об акциях: на какой товар, какой процент и сроки акции. Также возможно и описание.

Вкладка “Работники” помимо своей стандартной информации вводит также количество работников выделенного типа.

Вкладка “Производители” содержит в себе данные о местоположении производителя и его телефон.

Вкладка “Товары” содержит в себе подробную информацию о характеристиках товара, а также его количество на складе, сколько из него просрочено. Кнопка “Утилизация” позволяет утилизировать просроченный товар, который выводится вместе с сопутствующей информацией в таблицу. На форме утилизации кнопка “Назад” позволяет вернуться назад на форму администратора, а кнопка “Утилизировать” “утилизирует” товар (не воспринимайте буквально, базы данных ее не научились самостоятельно взаимодействовать с материальными предметами). Под “утилизацией” понимается изменение данных, сама утилизация товара выполняется, естественно, людьми. Но вернемся к вкладке “Товары”. Еще она содержит информацию о цене и дате этой цены. Забегая вперед, скажу, что цену можно менять раз в неделю, по причинам целесообразности.

Вкладка “Накладные” содержит информацию о дате ее подписания, кто принял, имя контрагента, а также перечень товаров, который был принят (если вы не знаете, как составляются накладные и зачем они нужны, обратитесь к Интернету, там наверняка найдется интересующая вас информация). Также при нажатии на определенный товар под ним появится 3 цифры: сколько продано, сколько просрочено и сколько еще осталось для данной партии. И еще выводится общая цена накладной.

Теперь перейдем к настройкам: в левом верхнем углу есть меню, в котором есть пункт “Настройки”. В них можно настроить путь архивации удаляемых данных, а также видимость предупреждающих окон.

И последнее, “Печать” – позволяет напечатать чек или накладную, в зависимости от выделенного элемента.

Пункт меню “Выход” позволяет выйти в окно входа.

# **ВЫВОД**

Результатом выполнения курсовой работы стала программа, позволяющая автоматизировать процессы, происходящие в аптеке и касающиеся предметной области данного проекта.

Разработанное приложение отвечает всем требованиям предметной области, таблицы созданной базы данных отвечают требованиям нормализации, что позволяет обеспечить целостность и непротиворечивость информации.

Как было сказано выше, в приложении присутствует разграничение доступа на 3 категории: Директор, Администратор и Кассир. На этом строится логика приложения.

Соответственно есть 3 окна: окно суперпользователя, окно администратора и окно кассира.

В окне кассира мы можем продавать товары, а также осуществлять интерактивный поиск по различным категориям без вложения.

Окно администратора позволяет просмотреть большую часть информации, хранящуюся в базе данных, а также осуществлять с ней определенные манипуляции, обладающие определенными ограничениями. Также в этом окне можно распечатать чеки и накладные, управлять данными пользователей БД, осуществлять селективный поиск максимальной вложенности по критериям и настраивать отображение предупреждающих окон.

Окно суперпользователя позволяет открыть окно администратора или кассира без ввода, просмотреть логи действий пользователей и настройки приложения, где можно задать пароли безопасности, а также поменять пароль записи директора. Ну и напоследок – можно задать процент розничной стоимости товара.

Средствами языка программирования и среды разработки Microsoft Visual Studio был создан пользовательский интерфейс. Приложение позволяет решать все задачи, относящиеся к предметной области.

СПИСОК ЛИТЕРАТУРЫ

* + - 1. Карпова Т.С. Базы данных: модели, разработка, реализация / Т.С. Карпова. – СПб.: Питер, 2001. – 304 с. – ISBN 5-272-00278-4.
      2. Хомоненко А.Д. Базы данных: учебник для высших учебных заведений / А.Д. Хомоненко, В.М. Цыганков, М.Г. Мальцев. – 4-е изд., доп. и перераб. – СПб.: КОРОНА принт, 2004. – 736 с. – ISBN 5-7931-0284-1.
      3. Герберт Шилдт Полное руководство по C#.

ПРИЛОЖЕНИЯ

Скрипт создания БД:

create database if not exists `pharmacy\_db`;

use `pharmacy\_db`;

create table if not exists `pharmacy\_db`.`manufacturer`(

M\_ID int(9)unsigned auto\_increment not null comment 'Код производителя',

M\_NAME char(50)not null comment 'Название производителя',

M\_COUNTRY char(15)not null comment 'Страна производителя',

M\_CITY char(15) comment 'Грод производителя',

M\_ADDR char(20) comment 'Адрес производителя',

M\_TEL char(15) comment 'Телефон производителя',

primary key (`M\_ID`)

)ENGINE=InnoDB AUTO\_INCREMENT=1 DEFAULT CHARSET=utf8;

create table if not exists `pharmacy\_db`.`product`(

P\_ID int(9)unsigned auto\_increment not null comment 'Код продукта',

P\_NAME char(50)not null comment 'Имя продукта',

M\_ID int(9)unsigned not null comment 'Код производителя',

P\_GROUP enum('Без группы','Антисептики','Слабительные','Противовирусные','Антибакериальные','Отхаркивающие и рвотные','Антидепрессанты','Снотворные','Анальгетики',

'Противогрибковые','Противопаразитные','Противовоспалительные','Противозачаточные','Противоалергические','Гастроэнтероголические','Противорвотные',

'Противодиарейные','Ингибиторы','Бронхолитические','Муколитические','Противокашлевые','Иммуномодуляторы','Иммуноглобулины','Витаминные','Минеральные',

'Поливитаминные','Питательные смеси','Жаропонижающие','Антианемические','Фитопрепараты','Средства для контрацепции','Сосудосуживающие',

'Антиглаукомные','Офтальмологические','Проктологические','Урологические','Нефрологические','Дерматологические','Диагностические','Перевязочные',

'Токсикологические','Антидоты','Гомеопатические','Эфирные масла','Средства личной гигиены','Косметические') not null comment 'Группа товара',

P\_PACK enum('Без упаковки','Ампула','Картонная коробка','Банка','Пакет','Полимерная','Баллон','Туба','Флакон','Бумага') not null comment 'Форма упаковки', P\_MATERIAL enum ('Неизвестно' ,'Жидкость' ,'Сироп','Таблетки','Капсулы','Шприц-тюбики', 'Порошок','Гранулы','Растительные волокна','Аэрозоль','Суспензия',

'Мазь','Паста')not null comment 'Материал продукта',

P\_FORM enum('с рецептом','без рецепта', 'строгого учета')not null comment 'Форма отпуска',

P\_INSTR text default null comment 'Инструкция',

`P\_CODE` BIGINT(13) UNSIGNED NOT NULL comment 'штрих-код',

primary key(`P\_ID`),

foreign key (`M\_ID`) references `pharmacy\_db`.`manufacturer`(`M\_ID`) on update cascade on delete cascade

)ENGINE=InnoDB AUTO\_INCREMENT=1 DEFAULT CHARSET=utf8;

create table if not exists `pharmacy\_db`.`employee`(

E\_ID int(9)unsigned not null auto\_increment comment 'Код сотрудника',

E\_NAME char(50)not null comment 'ФИО',

E\_TEL char(15) comment 'Телефон', E\_POSITION enum('Кассир','Уборщик','Менеджер','Администратор','Фармацевт')not null comment 'Должность', E\_CONTRACT bigint(11)unsigned not null unique comment 'Номер контракта',

E\_INN int(9) unsigned not null unique,

primary key (`E\_ID`)

)ENGINE=InnoDB AUTO\_INCREMENT=1 DEFAULT CHARSET=utf8;

create table if not exists `pharmacy\_db`.`waybill`(

W\_ID int(9)unsigned auto\_increment not null comment 'Код накладной',

W\_DATE date comment 'Дата накладной',

E\_ID int(9)unsigned not null comment 'Код сотрудника',

W\_AGENT\_NAME char(50) not null comment 'Имя контрагента',

primary key (`W\_ID`),

foreign key (`E\_ID`) references `pharmacy\_db`.`employee`(`E\_ID`) on update cascade on delete cascade

)ENGINE=InnoDB AUTO\_INCREMENT=1 DEFAULT CHARSET=utf8;

create table if not exists `pharmacy\_db`.`waybill\_list`(

WL\_ID int(9)unsigned auto\_increment not null comment 'Код списка накладной',

W\_ID int(9)unsigned not null comment 'Код накладной',

P\_ID int(9)unsigned not null comment 'Код продукта',

WL\_VALUE mediumint(5)unsigned not null comment 'Количество',

WL\_TRADE\_PRICE float(7,2)unsigned not null comment 'Оптовая цена',

WL\_BDATE date not null comment 'Дата поступления товара',

WL\_EDATE date not null comment 'Срок реализации товара',

primary key (`WL\_ID`),

foreign key (`P\_ID`) references `pharmacy\_db`.`product`(`P\_ID`) on update cascade on delete cascade,

foreign key (`W\_ID`) references `pharmacy\_db`.`waybill`(`W\_ID`) on update cascade on delete cascade )ENGINE=InnoDB AUTO\_INCREMENT=1 DEFAULT CHARSET=utf8;

create table if not exists `pharmacy\_db`.`product\_actual\_price`(

PAP\_ID int(9)unsigned auto\_increment not null comment 'Код актуальной цены',

P\_ID int(9)unsigned not null comment 'Код продукта',

PAP\_PRICE float(7,2)unsigned not null comment 'Цена',

PAP\_DATE datetime not null default current\_timestamp comment 'Дата актуальности',

primary key (`PAP\_ID`),

foreign key (`P\_ID`) references `pharmacy\_db`.`product`(`P\_ID`) on update cascade on delete cascade)ENGINE=InnoDB AUTO\_INCREMENT=1 DEFAULT CHARSET=utf8;

create table if not exists `pharmacy\_db`.`discounts`(

D\_ID mediumint(6)unsigned auto\_increment not null comment 'Код акции',

P\_ID int(9)unsigned comment 'Код продукта',

D\_PRICE int(4)unsigned not null comment 'Скидка',

D\_BDATE date comment 'Начало акции',

D\_EDATE date comment 'Конец акции',

D\_TEXT text comment 'Описание',

primary key (`D\_ID`),

foreign key (`P\_ID`) references `pharmacy\_db`.`product`(`P\_ID`) on update cascade on delete cascade

)ENGINE=InnoDB AUTO\_INCREMENT=1 DEFAULT CHARSET=utf8;

create table if not exists `pharmacy\_db`.`check`(

C\_ID int(9)unsigned auto\_increment not null comment 'Код операции',

E\_ID int(9)unsigned comment 'Код сотрудника',

C\_DATE datetime not null comment 'Дата операции',

C\_PAYTYPE enum('Наличные','Карточка') not null comment 'Тип оплаты',

C\_SUM FLOAT(7,2) UNSIGNED DEFAULT 0 comment 'Сумма чека',

C\_PREPAYMENT FLOAT(7,2) UNSIGNED default 0 comment 'Аванс',

primary key (`C\_ID`),

foreign key (`E\_ID`) references `pharmacy\_db`.`employee`(`E\_ID`) on update cascade on delete cascade

)ENGINE=InnoDB AUTO\_INCREMENT=1 DEFAULT CHARSET=utf8;

create table if not exists `pharmacy\_db`.`check\_list`(

CL\_ID int(9)unsigned auto\_increment not null comment 'Код списка',

C\_ID int(9)unsigned not null comment 'Код операции',

P\_ID int(9)unsigned not null comment 'Код продукта',

CL\_VALUE mediumint(5)unsigned not null comment 'Количество товара',

primary key(`CL\_ID`),

foreign key (`C\_ID`) references `pharmacy\_db`.`check`(`C\_ID`) on update cascade on delete cascade,

foreign key (`P\_ID`) references `pharmacy\_db`.`product`(`P\_ID`) on update cascade on delete cascade

)ENGINE=InnoDB AUTO\_INCREMENT=1 DEFAULT CHARSET=utf8;

create table if not exists `pharmacy\_db`.`user`(

U\_ID smallint(3)unsigned auto\_increment not null comment 'Код пользователя',

U\_TYPE enum('Кассир','Директор','Администратор')not null unique comment 'Тип',

U\_NAME char(20) not null unique comment 'Логин',

U\_PASS char(32) not null unique comment 'Пароль',

-- U\_ONLINE enum('offline','online') not null comment 'Онлайн',

primary key (`U\_ID`))ENGINE=InnoDB AUTO\_INCREMENT=1 DEFAULT CHARSET=utf8;

create table if not exists `pharmacy\_db`.`product\_quantity`(

PQ\_ID int(9)unsigned auto\_increment not null comment 'Код количества товара',

P\_ID int(9)unsigned comment 'Код продукта',

PQ\_IN mediumint(7)unsigned default 0 comment 'Пришло продукта',

PQ\_OUT mediumint(7)unsigned default 0 comment 'Продано продукта',

primary key (PQ\_ID),

foreign key (`P\_ID`) references `pharmacy\_db`.`product`(`P\_ID`) on update cascade on delete cascade)ENGINE=InnoDB AUTO\_INCREMENT=1 DEFAULT CHARSET=utf8;

create table if not exists `pharmacy\_db`.`product\_overdue`(

PP\_ID int(9)unsigned auto\_increment not null comment 'Код просроченности',

WL\_ID int(9)unsigned comment 'Код списка накладной',

PP\_IS\_OVERDUE enum('Не просрочено','Просрочено','Скоро истекает срок годности','N/A','Продано') default 'N/A' comment 'Просроченность',

primary key (PP\_ID),

foreign key (`WL\_ID`) references `pharmacy\_db`.`waybill\_list`(`WL\_ID`) on update cascade on delete cascade

)ENGINE=InnoDB AUTO\_INCREMENT=1 DEFAULT CHARSET=utf8;

create table `pharmacy\_db`.`product\_sold`(

PS\_ID int(9)unsigned auto\_increment not null comment 'Код количества проданных продуктов',

WL\_ID int(9)unsigned comment 'Код списка накладной',

PS\_COUNT int(5)unsigned default 0 comment'Количество проданных продуктов',

primary key (PS\_ID),

foreign key (`WL\_ID`) references `pharmacy\_db`.`waybill\_list`(`WL\_ID`) on update cascade on delete cascade

)ENGINE=InnoDB AUTO\_INCREMENT=1 DEFAULT CHARSET=utf8;

create table if not exists `pharmacy\_db`.`user\_action`(

UA\_ID int(9)unsigned not null auto\_increment comment 'Код действия',

U\_ID smallint(3)unsigned not null comment 'Код пользователя',

UA\_TYPE enum('Insert','Update','Delete','Select') not null comment 'Тип действия',

UA\_DATETIME datetime not null default current\_timestamp comment 'Дата операции',

UA\_DESCRIPTION char(200)not null comment 'Описание',

primary key (UA\_ID),

foreign key (U\_ID) references `pharmacy\_db`.`user`(U\_ID) on update cascade on delete cascade

)ENGINE=InnoDB AUTO\_INCREMENT=1 DEFAULT CHARSET=utf8;

create table if not exists `pharmacy\_db`.`passwords`(

PW\_ID smallint(3)unsigned auto\_increment not null comment 'Код',

PW\_PASS char(32) not null comment 'Пароль',

PW\_TYPE enum('1','2') not null unique comment 'Тип пароля',

primary key (`PW\_ID`)

)ENGINE=InnoDB AUTO\_INCREMENT=1 DEFAULT CHARSET=utf8;

DROP procedure IF EXISTS `insert\_price`;

DELIMITER $$

USE `pharmacy\_db`$$

CREATE PROCEDURE `insert\_price`(IN id INT(11),IN price FLOAT(5,2),IN dat DATE)

BEGIN

IF (SELECT COUNT(P\_ID) from product\_actual\_price WHERE DATE\_ADD(PAP\_DATE, INTERVAL 7 DAY)<=DATE(NOW()) AND P\_ID=id)>0

THEN INSERT INTO product\_actual\_price(P\_ID,PAP\_PRICE,PAP\_DATE)VALUES(id,price,dat);

END IF;

END$$

DELIMITER ;