**Волжский университет имени В.Н.Татищева**

**Факультет “Информатика и телекоммуникации”**

Кафедра “Информатика и системы управления”

**КУРСОВАЯ РАБОТА**

**по дисциплине: “Базы данных и знаний”**

**Тема: “Продажа и учет лекарственных средств ”**

Выполнил:

студент гр. ИТ-311

Захаров Д.В.

Преподаватель:

Трубачева С.И.

**Тольятти**

**2003**

**Волжский университет имени В.Н.Татищева**

**Факультет “Информатика и телекоммуникации”**

**Кафедра “Информатика и системы управления”**

**Специальность**:

ЗАДАНИЕ

на курсовой проект по дисциплине

“Базы данных и знаний”

Студенту\_ Захарову Д.В.

Группа\_\_\_ИТ-311

Тема проекта\_ Автоматизация продажи и учета лекарственных средств.

Календарный график задания\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

СОДЕРЖАНИЕ ЗАДАНИЯ

1. Изучить объект, его взаимосвязи.
2. Описать основные функции.
3. Создать таблицы.
4. Спроектировать диаграммы с помощью IDEF0, IDEF1
5. Спроектировать базу данных с помощью СУБД DELPHI.
6. Создать приложение, учитывая ввод, сохранение, удаление ненужных данных, редактирование имеющихся данных, позволяющие осуществить поиск.

Руководитель курсовой работы \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ / /

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ / /

Дата выдачи: “ \_\_\_\_ “ \_\_\_\_\_\_\_\_\_\_ 2002 г.

СОДЕРЖАНИЕ

1. Введение

2. Теоретическая часть

2.1 Постановка задачи

2.2 Описание предметной области

2.3 Построение SADT модели

### 2.4 Структурный анализ с помощью диаграмм сущность связь

3 Создание приложения в Delphi

3.1 Конфигурация системы с помощью утилиты BDE Administrtor

3.2 Создание таблиц базы данных с помощью утилиты Database Desktop

3.3 Организация пользовательского интерфейса

4. Заключение

5. Список используемой литературы

Приложения

**1.** **ВВЕДЕНИЕ**

В наше время без баз данных представить работу большинства финансовых, промышленных, торговых и прочих организаций невозможно. Базы данных позволяют информацию структурировать, хранить и извлекать оптимальным для пользователя образом. Среди большого разнообразия продуктов для разработки приложений Delphi занимает одно из ведущих мест . Delphi отдают предпочтение разработчики с разным стажем, привычками, профессиональными интересами. С помощью Delphi написано колоссальное количество приложений, десятки фирм и тысячи программистов-одиночек разрабатывают для Delphi дополнительные приложения.

Приложения с помощью Delphi разрабатываются быстро причем взаимодействие разработчика с интерактивной средой Delphi не вызывает внутреннего отторжения, а наоборот, оставляет ощущение комфорта.

Под базой данных понимают хранилище структурированных данных, при этом данные должны быть непротиворечивы, минимально избыточными и целостными.

Обычно базы данных создается для хранения и доступа к данным, содержащим сведения о некоторой предметной области, то есть некоторой области человеческой деятельности или области реального мира .Всякая база данных должна представлять систему данных о предметной области. Базы данных относящиеся к одной и той же предметной области, в различных случаях содержат более или менее детализированную информацию о ней. Степень детализации определяется рядом факторов, прежде всего целью использования информации из базы данных и сложностью производственных процессов, существующих в пределах предметной области в конкретных условиях.

Реляционные базы данных представляют связанную между собой совокупность таблиц баз данных. Связь между таблицами может находить свое отражение в структуре данных, а может только подразумеваться, то есть присутствовать на неформализованном уровне. В каждой таблице базы данных может существовать первичный ключ. Под первичным ключом понимают поле или набор полей, однозначно идентифицирующий запись. Значение первичного ключа в таблице базы данных должно быть уникальным, то есть в таблице не должно существовать двух или более записей с одинаковым значением первичного ключа. Первичный ключ должен быть минимально достаточным: в нем не должно быть полей, удаление которых из первичного ключа не отразится на его уникальности.

Так как в курсовом проекте представлена БД которая работает в сети следует это рассмотреть по подробнее. При работе с локальными базами данных сами БД расположены на том же компьютере, что и приложения, осуществляющие доступ к ним. Работа с БД происходит в однопользовательском режиме. **BDE** расположена на компьютере пользователя. Приложение ответственно за поддержание целостности БД и за выполнение запросов к БД. Общая схема однопользовательской архитектуры.

**2. ТЕОРИТИЧЕСКАЯ ЧАСТЬ**

**2.1 Постановка задачи**

Организация занимается продажей и учетом лекарственных средств.

Фирме поставляют лекарства предприятия, с которыми подписан контракт.

# Сотрудники фирмы вносят новые данные в базу, проверяют, какие лекарства продаются лучше, какие хуже и выставляют товар на продажу.

# На основе этих данных оформляют новый заказ на приобретение следующей партии лекарственных средств.

**2.2 Описание предметной области**

Фирма, занимающаяся продажей и учетом лекарственных средств. Для осуществления операций с поставками и заказами новых лекарственных средств, им необходимо заполнить накладные, в накладной указаны предприятие, которое осуществляет поставку, дата прибытия лекарств на склад. После лекарственные средства подсчитывают, устанавливают на них цену, по которой их будут продавать. Когда лекарства подсчитаны и установлена цена на него, то их распространяют по торговым точкам. В магазинах, которые занимаются продажей лекарств, при продаже его выдается чек покупателю, где указана дата продажи, количество купленного лекарства, стоимость за единицу товара. На основе полученных документов были созданы таблицы с полями.

Таблица лекарства:

* **Код лекарства;**
* **Название лекарства;**
* **Срок годности;**
* **Упаковка;**
* **Вес.**

Таблица предприятие:

* **Код предприятия;**
* **Название предприятия;**
* **Адрес предприятия.**

Таблица накладная:

* **Номер накладной;**
* **Код предприятия;**
* **Дата накладной.**

Таблица поступление:

* **Код поступления;**
* **Номер накладной;**
* **Код предприятия;**
* **Код лекарства;**
* **Цена за единицу лекарства;**
* **Количество лекарств.**

Таблица продажа:

* **Код лекарства;**
* **Номер чека;**
* **Дата продажи;**
* **Цена за единицу продукта;**

Изм.

Лист

№ докум.

Подпись

Дата

Лист

2

* **Количество проданных лекарств.**

**2.3 Структурный анализ с помощью диаграмм SADT**

Метод SADT разработан Дугласом Россом в 1973 г. и успешно используется в военных, промышленных и коммерческих организациях для решения широкого круга задач, таких как, долгосрочное и стратегическое планирование, автоматизированное производство и проектирование, разработка ПО для оборонных систем, управление финансами и материально – техническим снабжением и т.д. Метод SADT поддерживается министерством обороны США, которое было инициатором разработки стандарта IDEF0- подмножества SADT. IDEF0 был утвержден в качестве федерального стандарта США.

Метод SADT представляет собой совокупность правил и процедур, предназначенных для построения функциональной модели объекта какой – либо предметной области. *Функциональная модель* SADT отображает функциональную структуру объекта, т.е. производимые им действия и связи между этими действиями.

Результатом применения метода SADT является модель, которая состоит из диаграмм, фрагментов текстов и глоссария, имеющих ссылки друг на друга. Диаграммы – основные компоненты модели, все функции организации и интерфейсы на них представлены как блоки и дуги соответственно. Место соединения дуги с блоком определяет тип интерфейса. *Управляющая информация* входит в блок сверху, в то время как *входная информация*, которая подвергается обработке, показана с левой стороны блока, а *результат (выход)* показаны с правой стороны. *Механизм* (человек или автоматизированная система), который осуществляет операцию, представляется дугой, входящей в блок снизу (рис.2.1.).

Функция

А0

Управление

Вход

Выход

Механизм

##### Рис.2.1. Функциональный блок и интерфейсные дуги

Одной из наиболее важных особенностей метода SADT является постепенной введение все больших уровней детализации по мере создания диаграмм, отображающих модель. Каждый компонент модели может быть декомпозирован на другой диаграмме. В данном курсовом проекте разработана SADT диаграмма до 3-х уровней. Каждая диаграмма иллюстрирует «внутреннее строение» блока на родительской диаграмме (Приложение)

Диаграмма SADT состоит из 3-х уровней вложенности

**1 уровень -** самый основной, в него входят:

1) Входящие параметры, служат финансовые средства на приобретение лекарственных препаратов.

2) Управляющие параметры служат сотрудники выполняющие различные действия, например транспортировка, разгрузка лекарственных средств, учет лекарственных с средств.

3) Объектом в данной диаграмме является сама организация по учету и продаже лекарственных средств.

4) Механизмами служат законы, рынок, устав предприятия.

5) Выходные параметры прибыль, проданные лекарства, заказы на новую партию.

**2 уровень.** Состоит из 4-х составляющих а) Покупка и учет лекарственных препаратов, б) Транспортировка лекарственных препаратов, в) складирование лекарственных препаратов г) Продажа и учет лекарственных препаратов. У этих составляющих также есть входящие параметры, управляющие параметры, механизмы, выходные параметры.

**3 уровень.** В нем мы разбиваем на подуровни пункт г) из 2 уровня продажа и учет лекарственных препаратов. И получается 3 составляющих а) Установка цен на лекарственные препараты, б) Реализация покупателям, в) Учет лекарственных препаратов.

Структурная диаграмма представлена в **Приложении 1**.

**2.4 Структурный анализ с помощью диаграмм сущность-связь**

Широкое распространение реляционных СУБД и их использование в самых разнообразных приложениях показывает, что реляционная модель данных достаточна для моделирования предметных областей. Однако проектирование реляционной базы данных в терминах отношений на основе кратко рассмотренного нами механизма нормализации часто представляет собой очень сложный и неудобный для проектировщика процесс. При этом проявляется ограниченность реляционной модели данных в следующих аспектах:

1). Модель не предоставляет достаточных средств для представления смысла данных. Семантика реальной предметной области должна независимым от модели способом представляться в голове проектировщика. В частности, это относится к упоминавшейся нами проблеме представления ограничений целостности.

2). Для многих приложений трудно моделировать предметную область на основе плоских таблиц. В ряде случаев на самой начальной стадии проектирования проектировщику приходится производить насилие над собой, чтобы описать предметную область в виде одной (возможно, даже ненормализованной) таблицы.

3). Хотя весь процесс проектирования происходит на основе учета зависимостей, реляционная модель не предоставляет каких-либо средств для представления этих зависимостей.

4). Несмотря на то, что процесс проектирования начинается с выделения некоторых существенных для приложения объектов предметной области ("сущностей") и выявления связей между этими сущностями, реляционная модель данных не предлагает какого-либо аппарата для разделения сущностей и связей.

Далее мы кратко рассмотрим некоторые черты одной из наиболее популярных семантических моделей данных - модель "Сущность-Связи" (часто ее называют кратко ER-моделью).

На использовании разновидностей ER-модели основано большинство современных подходов к проектированию баз данных (главным образом, реляционных). Модель была предложена Ченом (Chen) в 1976 г. Моделирование предметной области базируется на использовании графических диаграмм, включающих небольшое число разнородных компонентов. Основными понятиями ER-модели являются сущность, связь и атрибут. Сущность - это реальный или представляемый объект, информация о котором должна сохраняться и быть доступна. В диаграммах ER-модели сущность представляется в виде прямоугольника, содержащего имя сущности. При этом имя сущности - это имя типа, а не некоторого конкретного экземпляра этого типа. Для большей выразительности и лучшего понимания имя сущности может сопровождаться примерами конкретных объектов этого типа.

##### Нормальные формы ER-схем

Как и в реляционных схемах баз данных, в ER-схемах вводится понятие нормальных форм, причем их смысл очень близко соответствует смыслу реляционных нормальных форм. Заметим, что формулировки нормальных форм ER-схем делают более понятным смысл нормализации реляционных схем. Мы приведем только очень краткие и неформальные определения трех первых нормальных форм.

В первой нормальной форме ER-схемы устраняются повторяющиеся атрибуты или группы атрибутов, т.е. производится выявление неявных сущностей, "замаскированных" под атрибуты.

Во второй нормальной форме устраняются атрибуты, зависящие только от части уникального идентификатора. Эта часть уникального идентификатора определяет отдельную сущность.

В третьей нормальной форме устраняются атрибуты, зависящие от атрибутов, не входящих в уникальный идентификатор. Эти атрибуты являются основой отдельной сущности.

Диаграмма представлена в Приложении 2.

**3. СОЗДАНИЕ ПРИЛОЖЕНИЯ В DELPHI**

**3.1 Работа с утилитой BDE Administrator**

**Назначение BDE Administrator**

В Delphi для доступа к базам данных из приложения, как известно использует-ся цепочка *"Приложение* -> *BDE* -> *База данных".* Это означает что при любом обра-щении к БД из приложения реально адресуется BDE (напомним, это аббревиатура Borland Database Engine, машины баз данных фирмы Borland). BDE, используя соб-ственные функции, связывав непосредственно с базой данных. Действия, осуществля-емые при этом ВDE мы здесь обсуждать не будем, поскольку эта тема отдельного рассмотрения.

Для работы с конкретной базой данных BDE, во-первых, должна знать:

• где БД физически расположена;

• параметры этой БД;

• общие параметры драйвера БД того типа, к которому принадлежит. обрабатываемая БД;

• общие системные установки.

Параметры драйвера БД определяют параметры конкретной БД, знача которых не указаны.

Системные установки являются общими для всех драйверов.

Замечание. Те, кто работал с Delphi версий 1 и 2, будут поначалу неприятно удивлены, не обнаружив в 3-ей версии BDE Configuration Utility. Именно её функции и выполняет в Delphi 3 утилита BDE Administrator. Несомненно, что вы найдете в ней много общего с BDE Configuration Utility.

###### Создание псевдонима БД

Параметры БД и ее местоположение определяются псевдонимом БД. *Псевдоним* - это некоторое имя (псевдоним БД). Именно этот псевдоним и используют при логическом обращении к БД компоненты типа “набор данных” приложения Delphi, например ТТаble и TQuery. BDE считывает параметры поставленные в соответствие данному псевдониму, что во многом определяет ее дальнейшие действия по физической работе с БД.

Псевдонимы баз данных определяются в утилите BDE Administrator. Для того, чтобы определить псевдоним, необходимо:

1. Выбрать элемент меню *Object \ New;*

2. Выбрать в появившемся окне имя драйвера базы данных (STANDARD для Paradox и dBase, MSACCESS для Microsoft Access, ORAC INTRBASE, SYBASE, MSSQL, INFORMIX, DB2 соответственно для данных Oracle, InterBase, Sybase, MS SQL Server, Informix, DB2 и, если установлен, драйвер ODBC);

3. Ввести имя псевдонима в левом окне;

4. Определить необходимые параметры псевдонима в правом окне

5. Щелкнуть по псевдониму правой кнопкой мыши и выбрать всплывающем меню элемент *Apply* для подтверждения или *Cancel* отказа.

**3.2 Создание таблиц базы данных**

**Объявление полей**

Для создания таблиц базы данных необходимо запустить утилиту Database Desktop(DBD). После запуска утилиты установим рабочий псевдоним утилиты. Это псевдоним, с которым утилита работает по умолчанию. Если рабочий псевдоним не установлен, придется при работе с DBD всякий раз указывать псевдоним, что отнимает время.

Для установки рабочего псевдонима нужно выбрать элемент главного меню File| Working Directory и в выпадающем списке Aliases выбрать имя псевдоним РRОВА, после чего нажать кнопку ОK.

Для создания таблицы БД нужно выбрать элемент главного меню File| New| Table. В появившемся окне Create Table оставляем без изменения тип создаваемой таблицы (Paradox 7) и нажимаем кнопку Ok . После этого появится окно определения структуры таблицы БД.

Каждая строка таблицы соответствует полю. Назначения столбцов:

Fields Name -имя поля;

Туре - тип поля;

Size - размер поля (для строковых полей, поскольку иные поля подразумевают размер, определяемый типом поля);

Кеу - содержит звездочку '\*', если поле входит в состав первичного ключа. Если в первичный ключ входит несколько полей, они должны определяться в той последовательности, в которой они присутствуют в первичном ключе. Кроме того, все поля, входящие в состав первичного индекса, должны определяться перед иными полями, то есть быть в списке полей наверху.

Определим поля, входящие в таблицу "Товары". Введем Тоvаrу в столбец field Name. Для того, чтобы определить тип поля, щелкните по столбцу

Туре и нажмите клавишу пробела. В ответ на это будет выдан список типов полей, из которых необходимо произвести выбор нужного типа.

Таблица 1 **Лекарства**

|  |  |  |  |
| --- | --- | --- | --- |
| **Номер** | **Название полей** | **Тип поля** | **Ключ** |
| 1 | Код лекарства | N | \* |
| 2 | Название лекарства | A |  |
| 3 | Срок годности | D |  |
| 4 | Упаковка | A |  |
| 5 | Вес | N |  |

В таблице 1 поля имеют тип:

Код лекарства (N) имеет числовой тип, Название лекарства (А) символьные значения, Срок годности (D) дата, Упаковка (А) символьные значения, Вес (N) числовой тип .

Поле Код лекарства является ключевым(\*).

Таблица 2 **Предприятие:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Номер** | **Название полей** | **Тип поля** | **Ключ** |
| 1 | Код предприятия | N | \* |
| 2 | Название предприятия | A |  |
| 3 | Адрес предприятия | А |  |

В таблице 2 поля имеют тип:

Код предприятия (N) имеет числовой тип, Название предприятия (А) символьные значения, Адрес предприятия (А)

Поле Код предприятия является ключевым(\*).

Таблица 3 **Накладная:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Номер** | **Название полей** | **Тип поля** | **Ключ** |
| 1 | Номер накладной | N | \* |
| 2 | Код предприятия | N |  |
| 3 | Дата накладной | D |  |

В таблице 3 поля имеют тип:

Номер накладной (N) имеет числовой тип, Код предприятия (N)

числовой тип,

Дата накладной (D) дата.

Поле Номер Накладной является ключевым(\*).

Таблица 4 **Поступление:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Номер** | **Название полей** | **Тип поля** | **Ключ** |
| 1 | Номер накладной | N |  |
| 2 | Код предприятия | N |  |
| 3 | Код лекарства | N |  |
| 4 | Код поступления | N | \* |
| 5 | Цена за ед. лекарства | N |  |
| 6 | Кол-во лекарства | N |  |

В таблице 4 поля имеют тип:

Номер накладной (N) числовой тип, Код предприятия (N) имеет числовой тип, Код лекарства (N) имеет числовой тип, Код поступления (N) имеет числовой тип, Цена за ед. лекарства (N) имеет числовой тип, Кол-во лекарства (N) имеет числовой тип.

Поле Код поступления является ключевым(\*).

Таблица 5 **Продажа:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Номер** | **Название полей** | **Тип поля** | **Ключ** |
| 1 | Код лекарства | N |  |
| 2 | Номер чека | N | \* |
| 3 | Дата продажи | D |  |
| 4 | Цена за ед. продукта | N |  |
| 5 | Кол-во проданных лекарств | N |  |

В таблице 5 поля имеют тип:

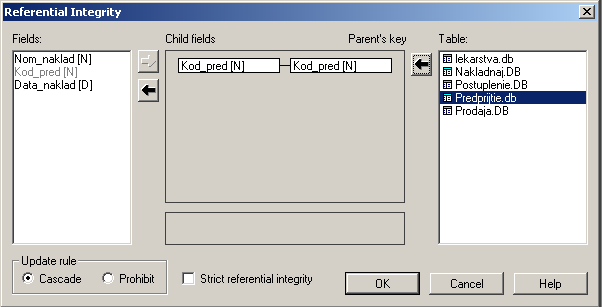
Код лекарства (N) имеет числовой тип, Номер чека (N) имеет числовой тип, Дата продажи (D) дата, Цена за ед. продукта (N) имеет числовой тип, Кол-во проданных лекарств (N) числовой тип.

Поле Номер чека является ключевым(\*).

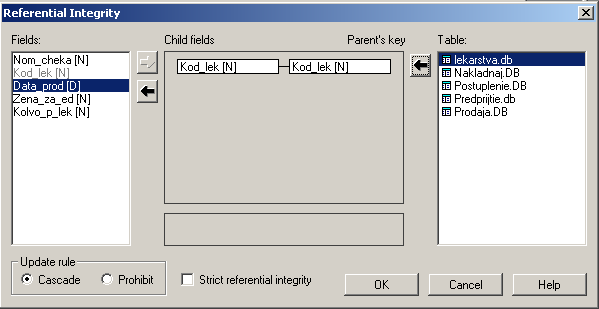
Связи в базе организованны по соотношению **один ко многим.**

Для установления связей между таблицами открывали дочернюю таблицу

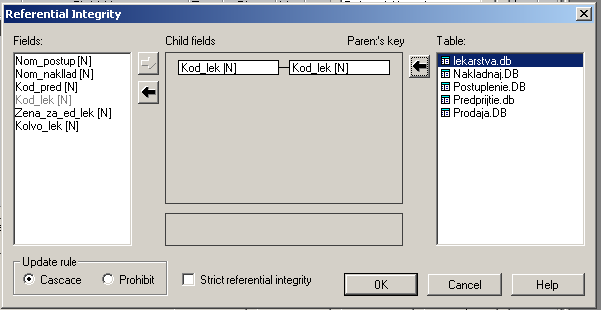
И выбирали в спадающем списке **Table properties** пункт **Referential Integrity.**



Связь таблицы Предприятие с таблицей Накладная.



Связь таблицы Лекарства с таблицей Продажа.

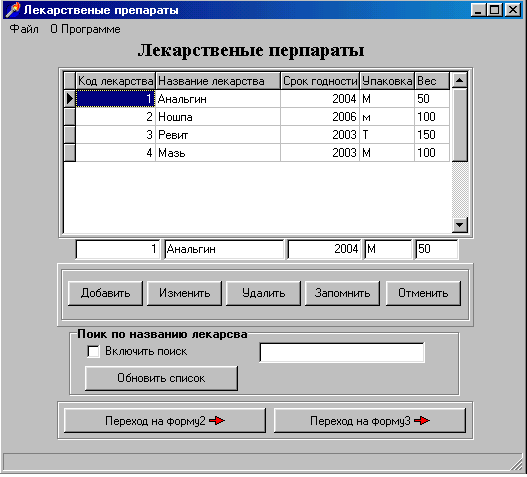


Связь таблицы Лекарства с таблицей Поступление.

По такому же принципу сделана связи с таблицей Накладная Поступление.

**3.3 Организация пользовательского интерфейса**

Интерфейс программы “Лекарственные средства” представлен на первой форме. Для удобства работы с базой при наведении курсора на объект появляются подсказки в статусной строке. Также присутствуют кнопки, для управления базой данных. В первой форме предусмотрен поиск позволяющий искать нужную запись.



Процедуры кнопок формы1:

Кнопка добавление:

**procedure TForm1.Button1Click(Sender: TObject);**

**begin**

**If table1.State = dsBrowse then**

**table1.insert;**

**end;**

Кнопка Изменения:

**procedure TForm1.Button2Click(Sender: TObject);**

**begin**

**If table1.State = dsBrowse then**

**table1.edit;**

**end;**

Кнопка Удаления:

**procedure TForm1.Button3Click(Sender: TObject);**

**begin**

**If table1.State = dsBrowse then**

**If Messagedlg('',mtconfirmation,[mbYes,mbNo],0)=mrYes then**

**table1.delete;**

**end;**

Процедура на кнопку поиск:

**procedure TForm1.Table1FilterRecord(DataSet: TDataSet;**

**var Accept: Boolean);**

**begin**

**Accept:=Dataset['Name\_lek'] = Edit1.text;**

**end;**

**procedure TForm1.CheckBox1Click(Sender: TObject);**

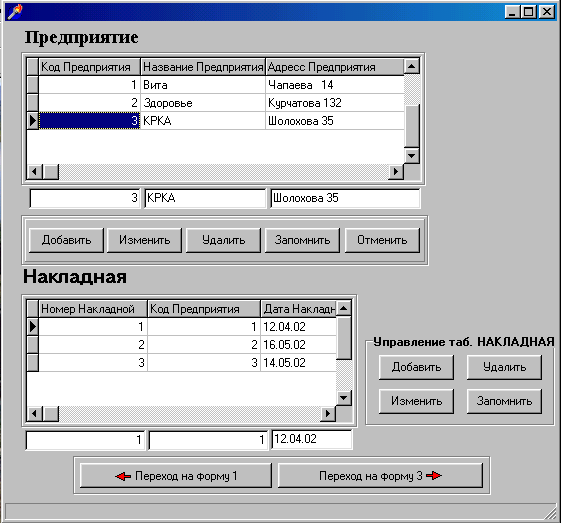
**begin**

**Table1.Filtered:=CheckBox1.Checked;**

**Table1.Filtered:=true;**

**end;**

Форма 2 на которой находятся таблицы Предприятие и Накладная.



Процедуры кнопок Формы 2:

Процедура кнопки перехода на Форму1:

**procedure TForm2.BitBtn1Click(Sender: TObject);**

**begin**

**Form1.show;**

**Form2.hide;**

**end;**

Процедура кнопки перехода на Форму3:

**procedure TForm2.BitBtn2Click(Sender: TObject);**

**begin**

**Form3.show;**

**Form2.hide;**

**end;**

Процедура соединения таблиц двух таблиц, дочерняя таблица Накладная с родительской таблицей Предприятие :

**procedure TForm2.OnActivate(Sender: TObject);**

**begin**

**DBGrid2.Columns.Items[1].PickList.Clear;**

**with Table1 do begin**

**first;**

**While not eof do Begin**

**DBGrid2.columns.Items[1].Picklist.add(FieldByName('Kod\_pred').value);**

**next;**

**end;**

**end;**

А также на форме присутствуют кнопки управления базой данных, как для таблицы Предприятие так и для таблицы Накладная:

кнопки добавления ,изменения, удаления, сохранения, отмены.

Процедура кнопки Запомнить:

**procedure TForm2.Button4Click(Sender: TObject);**

**begin**

**If table1.State in [dsInsert,dsEdit]then**

**table1.post;**

**end;**

Процедура кнопки Отменить:

**procedure TForm2.Button9Click(Sender: TObject);**

**begin**

**If table1.State in [dsInsert,dsEdit]then**

**table1.Cancel;**

**end;**

Процедура кнопки Добавить:

**procedure TForm2.Button1Click(Sender: TObject);**

**begin**

**If table1.State = dsBrowse then**

**table1.insert;**

**end;**

Процедура кнопки Изменить:

**procedure TForm2.Button2Click(Sender: TObject);**

**begin**

**If table1.State = dsBrowse then**

**table1.edit;**

**end;**

Процедура кнопки Удалить:

**procedure TForm2.Button3Click(Sender: TObject);**

**begin**

**If table1.State = dsBrowse then**

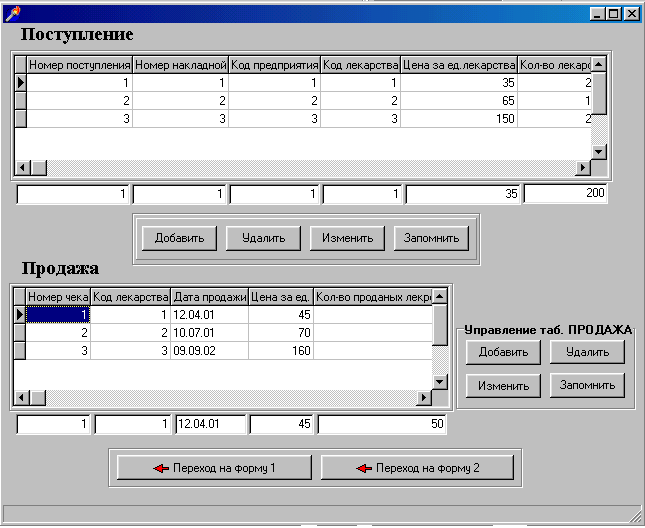
**If Messagedlg('',mtconfirmation,[mbYes,mbNo],0)=mrYes then**

**table1.delete;**

**end;**

Форма 3 с таблицами Поступление и Продажа.

На форме 3 находятся таблицы Продажа и Поступление. В этих таблицах фиксируется сколько лекарственных средств поступило на склад и по какой цене. Также фиксируется Код предприятия с которым установлены торговое соглашение, номер накладной где указан товар и в каком количестве. В таблице Продажа также фиксируются цена на лекарственные средства но при этом учитывается транспортировка и по этим параметрам устанавливается продажная цена.



Процедуры управляющих кнопок:

**procedure TForm3.Button1Click(Sender: TObject);**

**begin**

**If table2.State = dsBrowse then**

**table2.insert;**

**end;**

**procedure TForm3.Button4Click(Sender: TObject);**

**begin**

**If table2.State in [dsInsert,dsEdit]then**

**table2.post;**

**end;**

**procedure TForm3.Button3Click(Sender: TObject);**

**begin**

**If table2.State = dsBrowse then**

**table2.edit;**

**end;**

**procedure TForm3.Button2Click(Sender: TObject);**

**begin**

**If table2.State = dsBrowse then**

**If Messagedlg('',mtconfirmation,[mbYes,mbNo],0)=mrYes then**

**table2.delete;**

**end;**

**procedure TForm3.Button5Click(Sender: TObject);**

**begin**

**If table1.State = dsBrowse then**

**table1.insert;**

**end;**

**procedure TForm3.Button6Click(Sender: TObject);**

**begin**

**If table1.State = dsBrowse then**

**If Messagedlg('',mtconfirmation,[mbYes,mbNo],0)=mrYes then**

**table1.delete;**

**end;**

**procedure TForm3.Button7Click(Sender: TObject);**

**begin**

**If table1.State = dsBrowse then**

**table1.edit;**

**end;**

**procedure TForm3.Button8Click(Sender: TObject);**

**begin**

**If table1.State in [dsInsert,dsEdit]then**

**table1.post;**

**end;**

**end;**

**end;**

Процедуры соединения таблиц:

**DBGrid2.Columns.Items[1].PickList.Clear;**

**with Form2.Table2 do begin**

**first;**

**While not eof do Begin**

**DBGrid2.columns.Items[1].Picklist.add(FieldbyName('Nom\_naklad').value);**

**next;**

**end;**

**end;**

**DBGrid2.Columns.Items[2].PickList.Clear;**

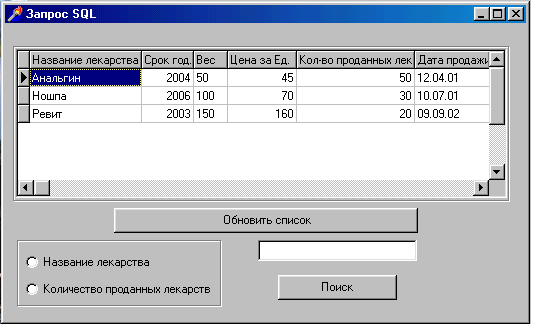
**with Form2.Table1 do begin**

**first;**

**While not eof do Begin**

**DBGrid2.columns.Items[2].Picklist.add(FieldbyName('Kod\_pred').value);**

Форма 4 запрос SQL. На данной форме выводится параметрический запрос по полям “Название лекарства” и по ”Количеству проданных лекарств”.



Процедура Запроса SQL:

Процедура кнопки обновить список:

**procedure TForm4.Button1Click(Sender: TObject);**

**begin**

**Query1.Filtered:=false;**

**Query1.Active:=true;**

**end;**

Процедура кнопки поиска :

**procedure TForm4.Button3Click(Sender: TObject);**

**begin**

**Query1.Filtered:=false;**

**Case Radiogroup1.itemindex of**

**0:pole:='Name\_lek';**

**1:pole:='Kolvo\_p\_lek'**

**end;**

**Query1.Filtered:=true;**

**end;**

**procedure TForm4.Query1FilterRecord(DataSet: TDataSet;**

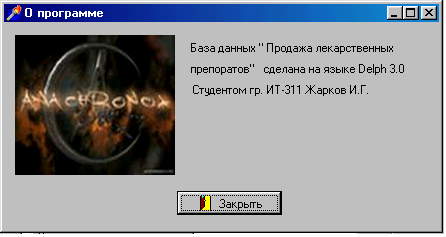
**var Accept: Boolean);**

**begin**

**Accept:=Dataset[pole]=edit1.text;**

**end;**

Форма 5 на ней показана информация о Программе.



**ЗАКЛЮЧЕНИЕ**

Данная программа пригодится небольшим фирмам занимающихся продажей и учетом лекарственных средств. Моя программа облегчит им работу, уменьшит работу с бумагами накладными и т.д.

После разработки своей программы я узнал более подробно о языке программирования Delphi и приложений к нему в частности BDE Administrator и Database Desktop . BDE Administratorнужен для установления псевдонима. При переносе базы данных на другой компьютер надо обязательно установить псевдоним т.е. каталог где она находится иначе при запуске базы данных выйдет сообщение об ошибке. Database Desktopнужна для создания и редактирования таблиц баз данных. Язык программирования Delphi является более простым для написания баз данных в нем есть все возможные компоненты упрощающие работу. На Delphi написано довольно большое число баз данных.

**ЛИТЕРАТУРА**

1. Delphi 3 и создание приложений баз данных Шумаков П.В. 704 с. иллюстрации есть.

2. Delphi 4 для профессионалов издание Питер М.Кенту 1088 с. иллюстрации есть.

**Листинг программы**

unit Unit1;

interface

uses

Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,

ExtCtrls, StdCtrls, Grids, DBGrids, Db, DBTables, ComCtrls, Menus,

Buttons, Mask, DBCtrls;

type

TForm1 = class(TForm)

DataSource1: TDataSource;

Table1: TTable;

GroupBox1: TGroupBox;

DBGrid1: TDBGrid;

Label1: TLabel;

Panel1: TPanel;

GroupBox2: TGroupBox;

Button1: TButton;

Button2: TButton;

Button3: TButton;

Button4: TButton;

Button5: TButton;

MainMenu1: TMainMenu;

N1: TMenuItem;

N2: TMenuItem;

GroupBox3: TGroupBox;

BitBtn3: TBitBtn;

BitBtn4: TBitBtn;

N3: TMenuItem;

DBEdit1: TDBEdit;

DBEdit2: TDBEdit;

DBEdit3: TDBEdit;

DBEdit4: TDBEdit;

DBEdit5: TDBEdit;

SQL1: TMenuItem;

RadioGroup1: TRadioGroup;

CheckBox1: TCheckBox;

Edit1: TEdit;

Button6: TButton;

StatusBar1: TStatusBar;

N4: TMenuItem;

procedure Button1Click(Sender: TObject);

procedure Button2Click(Sender: TObject);

procedure Button3Click(Sender: TObject);

procedure Button4Click(Sender: TObject);

procedure Button5Click(Sender: TObject);

procedure BitBtn3Click(Sender: TObject);

procedure BitBtn4Click(Sender: TObject);

procedure N3Click(Sender: TObject);

procedure SQL1Click(Sender: TObject);

procedure Table1FilterRecord(DataSet: TDataSet; var Accept: Boolean);

procedure CheckBox1Click(Sender: TObject);

procedure FormCreate(Sender: TObject);

procedure ShowHint(Sender: TObject);

procedure N4Click(Sender: TObject);

procedure Button6Click(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

var

Form1: TForm1;

implementation

uses Unit2, Unit3, Unit4, Unit5;

{$R \*.DFM}

procedure TForm1.Button1Click(Sender: TObject);

begin

If table1.State = dsBrowse then

table1.insert;

end;

procedure TForm1.Button2Click(Sender: TObject);

begin

If table1.State = dsBrowse then

table1.edit;

end;

procedure TForm1.Button3Click(Sender: TObject);

begin

If table1.State = dsBrowse then

If Messagedlg('',mtconfirmation,[mbYes,mbNo],0)=mrYes then

table1.delete;

end;

procedure TForm1.Button4Click(Sender: TObject);

begin

If table1.State in [dsInsert,dsEdit]then

table1.post;

end;

procedure TForm1.Button5Click(Sender: TObject);

begin

If table1.State in [dsInsert,dsEdit]then

table1.Cancel;

end;

procedure TForm1.BitBtn3Click(Sender: TObject);

begin

Form2.show;

Form1.hide;

end;

procedure TForm1.BitBtn4Click(Sender: TObject);

begin

Form3.show;

Form1.hide;

end;

procedure TForm1.N3Click(Sender: TObject);

begin

close;

end;

procedure TForm1.SQL1Click(Sender: TObject);

begin

Form4.show;

end;

procedure TForm1.Table1FilterRecord(DataSet: TDataSet;

var Accept: Boolean);

begin

Accept:=Dataset['Name\_lek'] = Edit1.text;

end;

procedure TForm1.CheckBox1Click(Sender: TObject);

begin

Table1.Filtered:=CheckBox1.Checked;

Table1.Filtered:=true;

end;

procedure TForm1.FormCreate(Sender: TObject);

begin

Application.OnHint := ShowHint;

end;

procedure TForm1.ShowHint(Sender: TObject);

begin

StatusBar1.Panels[0].text := Application.Hint;

end;

procedure TForm1.N4Click(Sender: TObject);

begin

Form5.show;

end;

procedure TForm1.Button6Click(Sender: TObject);

begin

Table1.Filtered:=False;

end;

end.

unit Unit2;

interface

uses

Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,

Grids, DBGrids, Db, DBTables, StdCtrls, ExtCtrls, ComCtrls, Buttons,

Mask, DBCtrls;

type

TForm2 = class(TForm)

GroupBox1: TGroupBox;

DataSource1: TDataSource;

Table1: TTable;

DBGrid1: TDBGrid;

GroupBox2: TGroupBox;

DBGrid2: TDBGrid;

DataSource2: TDataSource;

Table2: TTable;

GroupBox3: TGroupBox;

Button5: TButton;

Button6: TButton;

Button7: TButton;

Button8: TButton;

Label1: TLabel;

Label2: TLabel;

GroupBox4: TGroupBox;

Panel1: TPanel;

Button1: TButton;

Button2: TButton;

Button3: TButton;

Button4: TButton;

Button9: TButton;

StatusBar1: TStatusBar;

GroupBox5: TGroupBox;

BitBtn1: TBitBtn;

BitBtn2: TBitBtn;

DBEdit1: TDBEdit;

DBEdit2: TDBEdit;

DBEdit3: TDBEdit;

DBEdit4: TDBEdit;

DBEdit5: TDBEdit;

DBEdit6: TDBEdit;

procedure Button1Click(Sender: TObject);

procedure Button2Click(Sender: TObject);

procedure Button3Click(Sender: TObject);

procedure Button4Click(Sender: TObject);

procedure Button9Click(Sender: TObject);

procedure Button5Click(Sender: TObject);

procedure Button6Click(Sender: TObject);

procedure Button8Click(Sender: TObject);

procedure Button7Click(Sender: TObject);

procedure BitBtn1Click(Sender: TObject);

procedure BitBtn2Click(Sender: TObject);

procedure OnActivate(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

var

Form2: TForm2;

implementation

uses Unit1, Unit3;

{$R \*.DFM}

procedure TForm2.Button1Click(Sender: TObject);

begin

If table1.State = dsBrowse then

table1.insert;

end;

procedure TForm2.Button2Click(Sender: TObject);

begin

If table1.State = dsBrowse then

table1.edit;

end;

procedure TForm2.Button3Click(Sender: TObject);

begin

If table1.State = dsBrowse then

If Messagedlg('',mtconfirmation,[mbYes,mbNo],0)=mrYes then

table1.delete;

end;

procedure TForm2.Button4Click(Sender: TObject);

begin

If table1.State in [dsInsert,dsEdit]then

table1.post;

end;

procedure TForm2.Button9Click(Sender: TObject);

begin

If table1.State in [dsInsert,dsEdit]then

table1.Cancel;

end;

procedure TForm2.Button5Click(Sender: TObject);

begin

If table2.State = dsBrowse then

table2.insert;

end;

procedure TForm2.Button6Click(Sender: TObject);

begin

If table2.State = dsBrowse then

If Messagedlg('',mtconfirmation,[mbYes,mbNo],0)=mrYes then

table2.delete;

end;

procedure TForm2.Button8Click(Sender: TObject);

begin

If table2.State = dsBrowse then

table2.edit;

end;

procedure TForm2.Button7Click(Sender: TObject);

begin

If table2.State in [dsInsert,dsEdit]then

table2.post;

end;

procedure TForm2.BitBtn1Click(Sender: TObject);

begin

Form1.show;

Form2.hide;

end;

procedure TForm2.BitBtn2Click(Sender: TObject);

begin

Form3.show;

Form2.hide;

end;

procedure TForm2.OnActivate(Sender: TObject);

begin

DBGrid2.Columns.Items[1].PickList.Clear;

with Table1 do begin

first;

While not eof do Begin

DBGrid2.columns.Items[1].Picklist.add(FieldByName('Kod\_pred').value);

next;

end;

end;

end;

end.

unit Unit3;

interface

uses

Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,

StdCtrls, ExtCtrls, Db, DBTables, Grids, DBGrids, ComCtrls, Buttons,

Mask, DBCtrls;

type

TForm3 = class(TForm)

GroupBox1: TGroupBox;

DBGrid1: TDBGrid;

DataSource1: TDataSource;

Table1: TTable;

Label1: TLabel;

Label2: TLabel;

GroupBox2: TGroupBox;

Panel1: TPanel;

Button1: TButton;

Button2: TButton;

Button3: TButton;

Button4: TButton;

GroupBox3: TGroupBox;

DBGrid2: TDBGrid;

DataSource2: TDataSource;

Table2: TTable;

GroupBox4: TGroupBox;

Button5: TButton;

Button6: TButton;

Button7: TButton;

Button8: TButton;

StatusBar1: TStatusBar;

GroupBox5: TGroupBox;

BitBtn1: TBitBtn;

BitBtn2: TBitBtn;

DBEdit1: TDBEdit;

DBEdit2: TDBEdit;

DBEdit3: TDBEdit;

DBEdit4: TDBEdit;

DBEdit5: TDBEdit;

DBEdit6: TDBEdit;

DBEdit7: TDBEdit;

DBEdit8: TDBEdit;

DBEdit9: TDBEdit;

DBEdit10: TDBEdit;

DBEdit11: TDBEdit;

procedure Button1Click(Sender: TObject);

procedure Button4Click(Sender: TObject);

procedure Button3Click(Sender: TObject);

procedure Button2Click(Sender: TObject);

procedure Button5Click(Sender: TObject);

procedure Button6Click(Sender: TObject);

procedure Button7Click(Sender: TObject);

procedure Button8Click(Sender: TObject);

procedure BitBtn1Click(Sender: TObject);

procedure BitBtn2Click(Sender: TObject);

procedure OnActivate(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

var

Form3: TForm3;

implementation

uses Unit1, Unit2;

{$R \*.DFM}

procedure TForm3.Button1Click(Sender: TObject);

begin

If table2.State = dsBrowse then

table2.insert;

end;

procedure TForm3.Button4Click(Sender: TObject);

begin

If table2.State in [dsInsert,dsEdit]then

table2.post;

end;

procedure TForm3.Button3Click(Sender: TObject);

begin

If table2.State = dsBrowse then

table2.edit;

end;

procedure TForm3.Button2Click(Sender: TObject);

begin

If table2.State = dsBrowse then

If Messagedlg('',mtconfirmation,[mbYes,mbNo],0)=mrYes then

table2.delete;

end;

procedure TForm3.Button5Click(Sender: TObject);

begin

If table1.State = dsBrowse then

table1.insert;

end;

procedure TForm3.Button6Click(Sender: TObject);

begin

If table1.State = dsBrowse then

If Messagedlg('',mtconfirmation,[mbYes,mbNo],0)=mrYes then

table1.delete;

end;

procedure TForm3.Button7Click(Sender: TObject);

begin

If table1.State = dsBrowse then

table1.edit;

end;

procedure TForm3.Button8Click(Sender: TObject);

begin

If table1.State in [dsInsert,dsEdit]then

table1.post;

end;

procedure TForm3.BitBtn1Click(Sender: TObject);

begin

Form1.show;

Form3.hide;

end;

procedure TForm3.BitBtn2Click(Sender: TObject);

begin

Form2.show;

Form3.hide;

end;

procedure TForm3.OnActivate(Sender: TObject);

begin

DBGrid2.Columns.Items[3].PickList.Clear;

with Form1.Table1 do begin

first;

While not eof do Begin

DBGrid2.columns.Items[3].Picklist.add(FieldbyName('Kod\_lek').value);

next;

end;

end;

DBGrid2.Columns.Items[1].PickList.Clear;

with Form2.Table2 do begin

first;

While not eof do Begin

DBGrid2.columns.Items[1].Picklist.add(FieldbyName('Nom\_naklad').value);

next;

end;

end;

DBGrid2.Columns.Items[2].PickList.Clear;

with Form2.Table1 do begin

first;

While not eof do Begin

DBGrid2.columns.Items[2].Picklist.add(FieldbyName('Kod\_pred').value);

next;

end;

end;

DBGrid1.Columns.Items[1].PickList.Clear;

with Form1.Table1 do begin

first;

While not eof do Begin

DBGrid1.columns.Items[1].Picklist.add(FieldbyName('Kod\_lek').value);

next;

end;

end;

end;

end.

unit Unit4;

interface

uses

Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,

StdCtrls, Db, DBTables, Grids, DBGrids, ExtCtrls;

type

TForm4 = class(TForm)

GroupBox1: TGroupBox;

DBGrid1: TDBGrid;

DataSource1: TDataSource;

Button1: TButton;

Query1: TQuery;

Edit1: TEdit;

Button3: TButton;

RadioGroup1: TRadioGroup;

procedure Button1Click(Sender: TObject);

procedure Button2Click(Sender: TObject);

procedure Button3Click(Sender: TObject);

procedure Query1FilterRecord(DataSet: TDataSet; var Accept: Boolean);

private

{ Private declarations }

public

{ Public declarations }

end;

var

Form4: TForm4;

pole :shortstring;

implementation

{$R \*.DFM}

procedure TForm4.Button1Click(Sender: TObject);

begin

Query1.Filtered:=false;

Query1.Active:=true;

end;

procedure TForm4.Button2Click(Sender: TObject);

begin

Query1.Active:=false;

end;

procedure TForm4.Button3Click(Sender: TObject);

begin

Query1.Filtered:=false;

Case Radiogroup1.itemindex of

0:pole:='Name\_lek';

1:pole:='Kolvo\_p\_lek'

end;

Query1.Filtered:=true;

end;

procedure TForm4.Query1FilterRecord(DataSet: TDataSet;

var Accept: Boolean);

begin

Accept:=Dataset[pole]=edit1.text;

end;

end.

unit Unit5;

interface

uses

Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,

StdCtrls, ExtCtrls, Buttons;

type

TForm5 = class(TForm)

Image1: TImage;

Label1: TLabel;

Label2: TLabel;

Label3: TLabel;

BitBtn1: TBitBtn;

Label6: TLabel;

procedure Button1Click(Sender: TObject);

procedure BitBtn1Click(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

var

Form5: TForm5;

implementation

{$R \*.DFM}

procedure TForm5.Button1Click(Sender: TObject);

begin

Form5.close;

end;

procedure TForm5.BitBtn1Click(Sender: TObject);

begin

Form5.Close;

end;

end.