Содержание

Введение

Эффективность функционирования предприятия или организации любой отрасли и сферы деятельности напрямую зависит от скорости, точности и своевременности обмена данными как внутри этого предприятия между его составляющими частями (отделами, подсистемами и т.д.), так и вне его, то есть взаимодействие и обмен данными этой организации с другими (конкурирующими, предприятиями-партнерами и т.д.). И чем больше, масштабнее предприятие, тем серьезнее перед его управляющими встает проблема организации и контроля потоков огромного количества информации предприятия.

Чтобы обеспечить взаимодействия человека с персональным компьютером в интерактивном режиме, стали разрабатываться автоматизированные информационные системы, которые является совокупностью аппаратных и программных средств, которые обеспечивают взаимодействия человека и компьютера, а обеспечивают следующими функциями: возможность ввода в ПК и возможность вывода информации, как на экран, так и на устройства вывода.

Целью данной курсовой работы является разработка приложения учета валютных операций.

1.Логическое проектирование структуры базы данных

Для логического проектирования структуры базы данных возможно использование одной из нескольких стратегий проектирования:

1) Нормализация схем отношений. Нормализация предусматривает идентификацию требуемых атрибутов с последующим созданием из них нормализованных таблиц, основанных на функциональных зависимостях между этими атрибутами. Процесс проектирования представляет собой процесс нормализации.

2) Концепция модели «сущность-связь». Начинается этот подход с разработки моделей данных, которые содержат несколько высокоуровневых сущностей и связей, затем работа продолжается в виде серии нисходящих уточнений низкоуровневых сущностей, связей и относящихся к ним атрибутов.

3) Подход «от общего к частному» напоминает нормализацию, но отличается от него тем, что вначале выявляется набор основных сущностей с последующим расширением круга рассматриваемых сущностей, связей и атрибутов, которые взаимодействуют с первоначально определенными сущностями.

4) «Смешанная» стратегия проектирования предполагает использование первых двух стратегий для создания разных частей модели, после чего все подготовленные фрагменты собираются в единое целое.

Воспользуемся «смешанной» стратегией проектирования.

На первом этапе выделим сущности и опишем их реквизитный состав, на втором – сформируем отношения, уточним количество этих отношений и их атрибутный состав. Затем проверим отношения на соответствие 3 НФ.

а. Выделение сущностей и описание их реквизитного состава.

В задаче учета командировок сотрудников можно выделить следующие сущности:

Сущность «Валюта», которая характеризуется реквизитами: идентификатор сущности, наименование валюты.

Сущность «Тип операции», которая характеризуется реквизитами: идентификатор сущности, название операции.

Сущность «Курс», которая характеризуется реквизитами: идентификатор сущности, идентификатор валюты, отношение единицы валюты к единицы доллара.

Сущность «Учет операция», которая характеризуется реквизитами: идентификатор сущности, дата операции, валюта, количество платежных средств, тип операции, общая сумма.

b. Подготовка к применению ER-метода логического проектирования.

Использование данного метода возможно тогда, когда в результате выполнения концептуального проектирования БД уже выполнены следующие действия:

выделены все сущности, информация о которых должна содержаться в искомой БД;

определены основные атрибуты для каждой сущности;

назначен ключевой атрибут для каждой сущности;

сформулированы связи между выделенными сущностями.

Итак, для применения ER-метода назначим ключевые атрибуты для каждой сущности и сформулируем связи между выделенными сущностями:

Валюта (Ид, Название)

ТипОперации(Ид, Название)

Курс(Ид, ИдВалюты, ОтношениеКДоллару)

УчетОпераций(Ид, Дата, ИдВалюты, КоличествоВалюты, ИдТипОперации, ОбщаяСумма)

Сущности *«Курс»* и *«Валюта»* соотносятся с помощью *связи «Включает»*.

Сущности *«УчетОпераций»* и *«Курс»* соотносятся с помощью *связи «Содержит»*.

Сущности *«УчетОпераций»* и *«ТипОперации»* соотносятся с помощью *связи «Содержит»*. Использование ER-метода логического проектирования.

С помощью указанного метода логического проектирования уточним количество таблиц-отношений и их атрибуты.

Построим ER-диаграмму для проектируемой задачи (рис. 2.1).

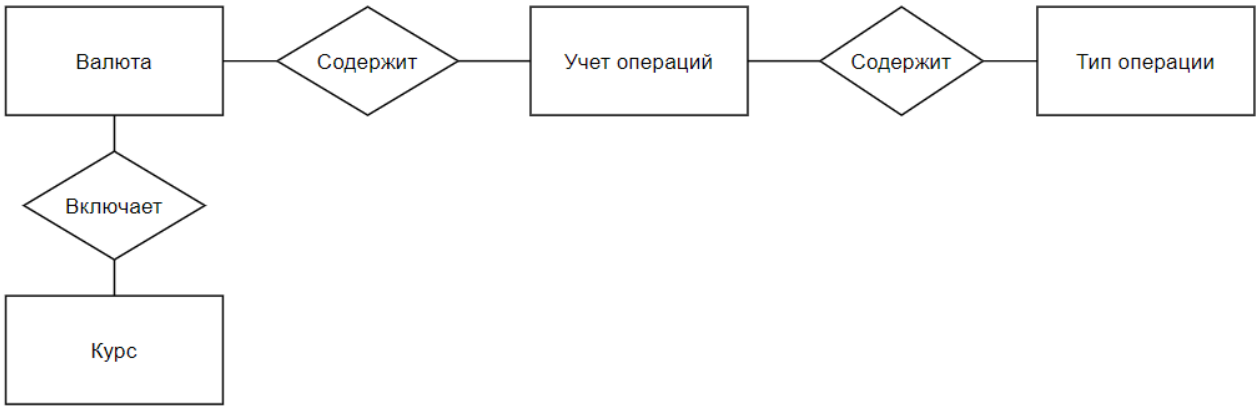


Рисунок 2.1. –Диаграмма ER-типа проектируемой базы

Таким образом, искомая БД состоит из четырех отношений:

*Валюта (**Ид**, Название)*

*ТипОперации(**Ид**, Название)*

*Курс(**Ид**, **ИдВалюты**, ОтношениеКДоллару)*

*УчетОпераций(**Ид**, Дата, **ИдВалюты**, КоличествоВалюты, **ИдТипОперации**, ОбщаяСумма)*

У всех таблиц первичным ключом является одно поле индивидуальный идентификатор (ид).

Все получившиеся таблицы удовлетворяют требованиям 3НФ (см. Приложение Б). Эта степень нормализации вполне достаточна для разрабатываемой базы данных, поэтому будем считать, что процесс проектирования заданной БД завершен.

2.Физическое проектирование базы данных

2.1. Описание структуры таблиц

Опишем структуру таблиц, полученных в ходе логического проектирования, в применении к конкретной СУБД (СУБД MySQL).

Таблица 2.1 – Структура таблиц базы данных

Имя таблицы	**Столбец**	**Описание столбца**	**Комментарий**
currencyYM	Id	integer, unsigned, AUTO_INCREMENT, PRIMARY KEY	код сущности, счетчик, первичный ключ
	Name	varchar(255)	название валюты
typeOperationYM	Id	integer, unsigned, AUTO_INCREMENT, PRIMARY KEY	код сущности, счетчик, первичный ключ
	Name	varchar(255)	Название типа операции
CourseYM	Id	integer, unsigned, AUTO_INCREMENT, PRIMARY KEY	код сущности, счетчик, первичный ключ
	Currency	integer, FOREIGN KEY	Код валюты
	ratioToTheDollarUnit	float	Отношение единицы валюты к единице доллара
AccountingFor TransactionsYM	Id	integer, unsigned, AUTO_INCREMENT, PRIMARY KEY	код сущности, счетчик, первичный ключ
	Dates	Date	Дата проведения операции
	Currency	integer, FOREIGN KEY	Код валюты
	Count	Float	Количество валюты
expense	typeOperation	integer, FOREIGN KEY	код типа операции
	summ	float	Общая сумма в долларах

Для создания базы данных AccountingCurrencyTransactions в MySql Workbench возможно использование одного из двух подходов: прямое проектирование (FORVARD INGINEER) и обратное (REVERSE INGINEER).

В первом случае процесс получения структуры базы данных для выбранной СУБД осуществляется на основе построенной ER-модели.

Другими словами, вначале создается графическая модель базы данных и на ее основе строится физическая модель базы данных: структура таблиц и связи между ними.

Во втором случае – выполняется создание ER-модели на основе уже созданной базы данных. Т.е. вначале описывается структура таблиц и связи, а затем создается графическая модель базы данных.

Для создания базы в Microsoft Access достаточно выбрать пункт Создание-Таблицы и следовать инструкции создании таблицы.

Графическая модель базы данных AccountingCurrencyTransactions представлена на рисунке 2.1.

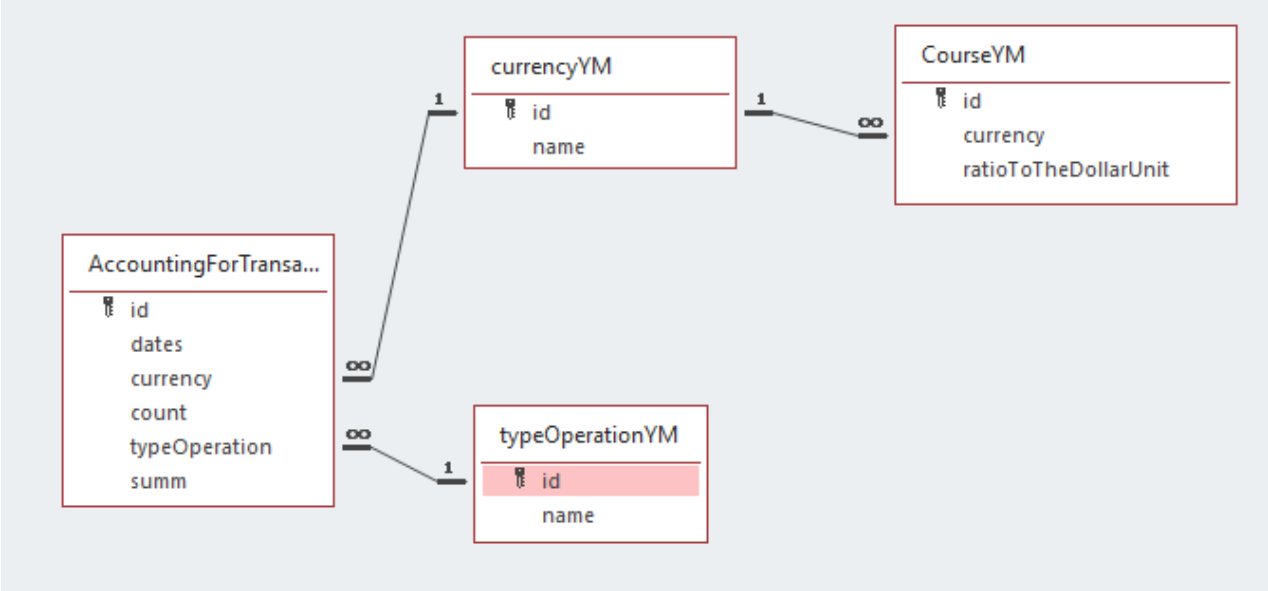


Рисунок 2.1 – Графическое представление базы данных

2.2. Описание создания таблиц базы данных AccountingCurrencyTransactions

Для создания базы данных и всех ее таблиц, используется следующий код:

```
CREATE TABLE currencyYM (
  [id] AUTOINCREMENT,
  [name] TEXT NOT NULL,
  PRIMARY KEY (id)
```

```
);

CREATE TABLE typeOperationYM (
  [id] AUTOINCREMENT,
  [name] TEXT NOT NULL,
  PRIMARY KEY (id)
);
CREATE TABLE CourseYM (
  [id] AUTOINCREMENT,
  [currency] INTEGER NOT NULL,
  [ratioToTheDollarUnit] FLOAT NOT NULL,
  PRIMARY KEY (id),
  FOREIGN KEY([currency]) REFERENCES currencyYM(id)
);

CREATE TABLE AccountingForTransactionsYM (
  [id] AUTOINCREMENT,
  [dates] DATE NOT NULL,
  [currency] INTEGER NOT NULL,
  [count] INTEGER NOT NULL,
  [typeOperation] INTEGER NOT NULL,
  [summ] FLOAT,
  PRIMARY KEY (id),
  FOREIGN KEY([currency]) REFERENCES currencyYM(id),
  FOREIGN KEY([typeOperation]) REFERENCES typeOperationYM(id)
);
```

3.Архитектура приложения

Каркас приложения выполнен в архитектуре MVC.

Модель данных представлена в виде набора классов (рисунки 4.1-4.4).

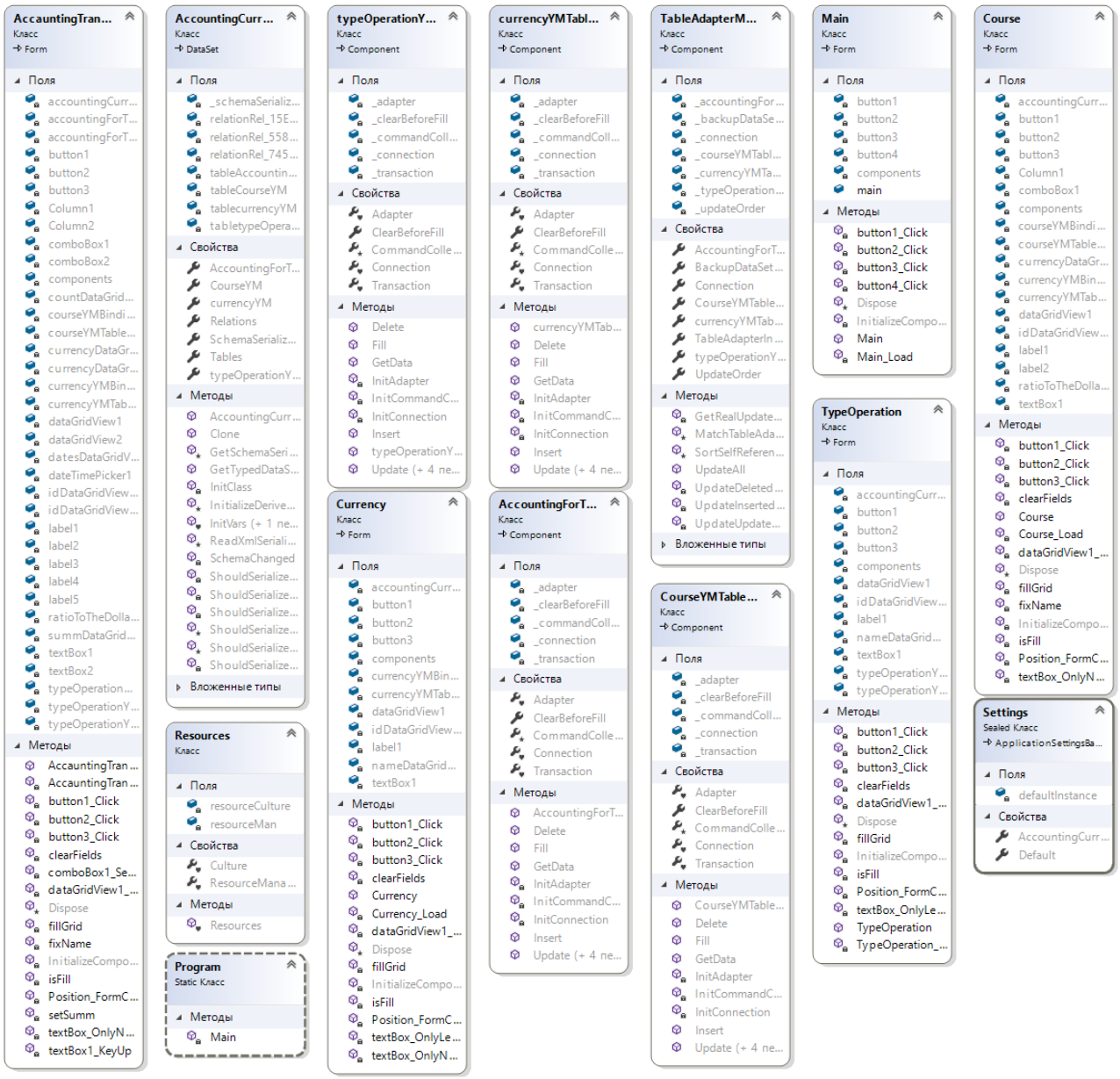


Рисунок 3.1 – Список классов приложения

Эти классы выступают в роли контроллеров форм. Таким образом, в них описывается взаимодействие клиента с интерфейсом и ответы на них.

Для отображения (View) каждой формы, используются внутренние данные каждого класса, оформленным отдельным классом с именем *Название-Класса.Designer.cs*.

В паттерне «модель – представление – контроллер» модель представляет данные приложения и связанную с ними бизнес-логику.

Данные передаются в БД и из нее в объектах передачи данных, и к ним обращаются с помощью объектов доступа к данным.

Представление – это наглядное отображение содержащихся в модели данных. Подмножество модели содержится в отдельном представлении, таким образом, представление действует в качества фильтра для данных модели. Пользователь взаимодействует с данными модели с помощью предлагаемого представлением наглядного отображения и обращается к бизнес логике, которая, в свою очередь, воздействует на данные модели.

Контроллер связывает представление с моделью и управляет потоками данных приложения. Он выбирает, какое представление визуализировать для пользователя в ответ на вводимые им данные и в соответствии с выполняемой бизнес-логикой. Контроллер получает сообщение от представления и пересылает его модели. Модель, в свою очередь, подготавливает ответ и отправляет его обратно контроллеру, где происходит выбор представления и отправка его пользователю.

Компонент BindingSource выступает в качестве источника данных для некоторых или всех элементов управления формы. В Visual Studio объект BindingSource можно привязать к элементу управления с помощью DataBindings свойства, доступного в окне Свойства .

Компонент BindingSource можно привязать как к источникам простых данных, например одиночному свойству объекта или базовой коллекции, такому как ArrayList, так и к источникам сложных данных, таким как таблица базы данных. Компонент BindingSource является посредником, обеспечивающим привязку и управление валютой. Во время разработки или во время выполнения компонент BindingSource можно привязать к источнику сложных данных, указав в качестве значений его свойств DataSource и DataMember базу данных и таблицу. На следующем рисунке показано, как компонент BindingSource встраивается в существующую архитектуру привязки данных.

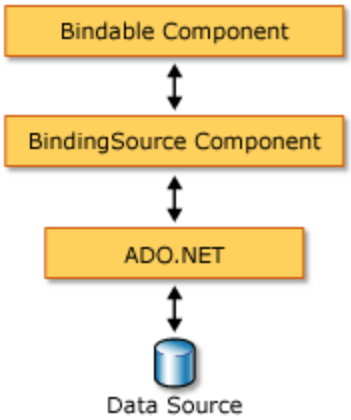


Рисунок 3.2 – Встраивание компонента в архитектуру

Доступ к данным в СУБД осуществляться при помощи настраиваемого файла конфигурации.

App.config

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
    <configSections>
    </configSections>
    <connectionStrings>
        <add name="AccountingCurrencyTransactions.Properties.Settings.AccountingCurrencyTransactionsConnectionString"
            connectionString="Provider=Microsoft.ACE.OLEDB.12.0;Data Source=|DataDirectory|\AccountingCurrencyTransactions.accdb"
            providerName="System.Data.OleDb" />
    </connectionStrings>
    <startup>
        <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.6" />
    </startup>
</configuration>
```

Рисунок 3.3 – Файл app.config

В разрабатываемом приложении обеспечена возможность добавления, редактирования и удаления записей из базы данных, сохранение табличных результатов в excel-файле.

4.Документация по используемым функциям приложения

Описание класса AccauntingTransaction

Опишем методы класса AccauntingTransaction:

- BusinessTrip_Load – метод, при загрузке формы обновляет значения таблицы и выпадающих списокв
- BusinessTrip_FormClosing – метод, при закрытии формы, возвращает предыдущую на экран
- fixName – метод, исправляющий отображение наименований данных других таблиц, поскольку из базы данных получаем только ид
- textBox_OnlyNumbers – метод, позволяющий вводить в поле только цифры
- fillGrid – метод, обновляющий данные в таблице и выпадающих списках
- isFill – метод, проводящий проверку на заполнение полей формы
- clearFields – метод, очищающий все поля формы
- button1_Click – метод добавления строки в бд
- button2_Click – метод редактирование строки в бд
- button3_Click – метод, удаление строки из бд
- dataGridView1_Click – метод, заполняющий поля формы при выборе определенной записи в таблице
- button4_Click – метод сохранения таблицы в Excel-файл.

Кассы Course, Currency, TypeOperaion содержат тот же набор методов.

5.Руководство пользователя

5.1Назначение приложения

Данное приложение предназначено для автоматизации ведения учета валютных операций. Пользователям доступны операции по добавлению, изменению, удалению данных о валютах, курсах, типах операций с валютой, учету операций.

5.2Условия выполнения приложения

Исполняемый файл AccountingCurrencyTransactions занимает 10 КБ памяти и может выполняться на любом персональном компьютере, на котором установлено Windows 7 и выше, MicrosoftOffice, .NET, MiccrosoftAccess 2010.

5.3Выполнение приложения

Для получения доступа к приложению требуется запустить файл AccountingCurrencyTransactions.exe и дождаться появления главного окна приложения.

6.Результаты тестирования разработанного приложения

При входе в приложения мы видим следующую форму

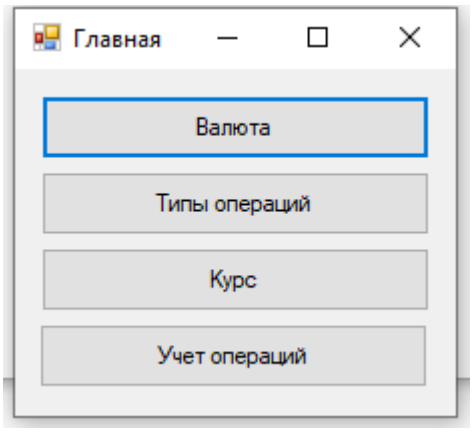


Рисунок 6.1 – Главная форма

Первым делом стоит заполнить список валют. Для этого перейдем с помощью первой кнопки в справочник валют.

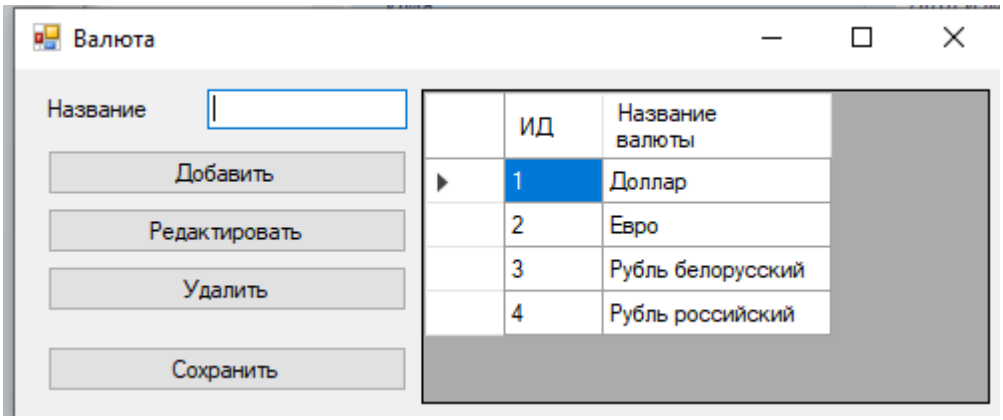


Рисунок 6.2 – Валюта

На этой форме можно выполнить основные операции с данными таблицы. При отсутствии заполнения поля для добавления, приложение выдаст соответствующую ошибку.

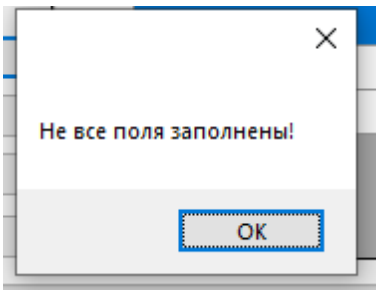


Рисунок 6.3 – Ошибка при незаполненных полях

Можно также перейти на страницу списка операций, где работа уже пойдет непосредственно с ними.

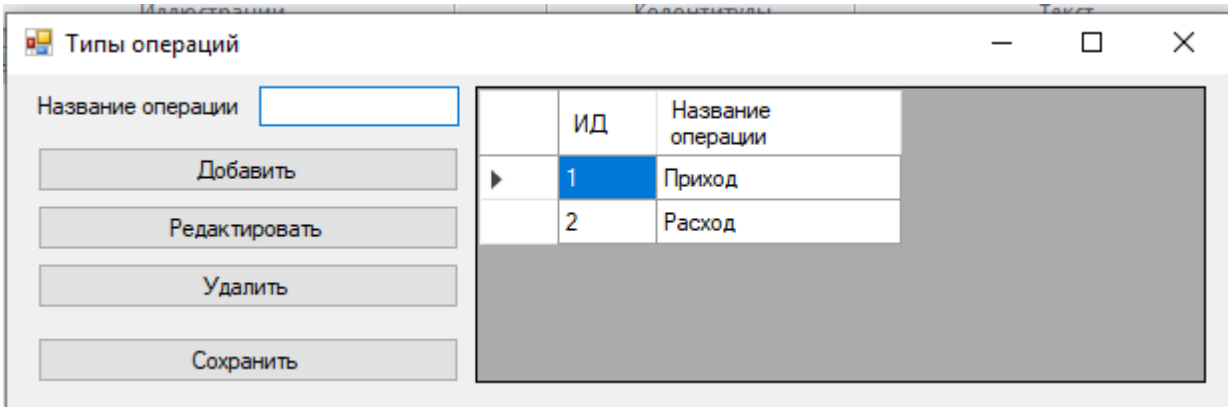
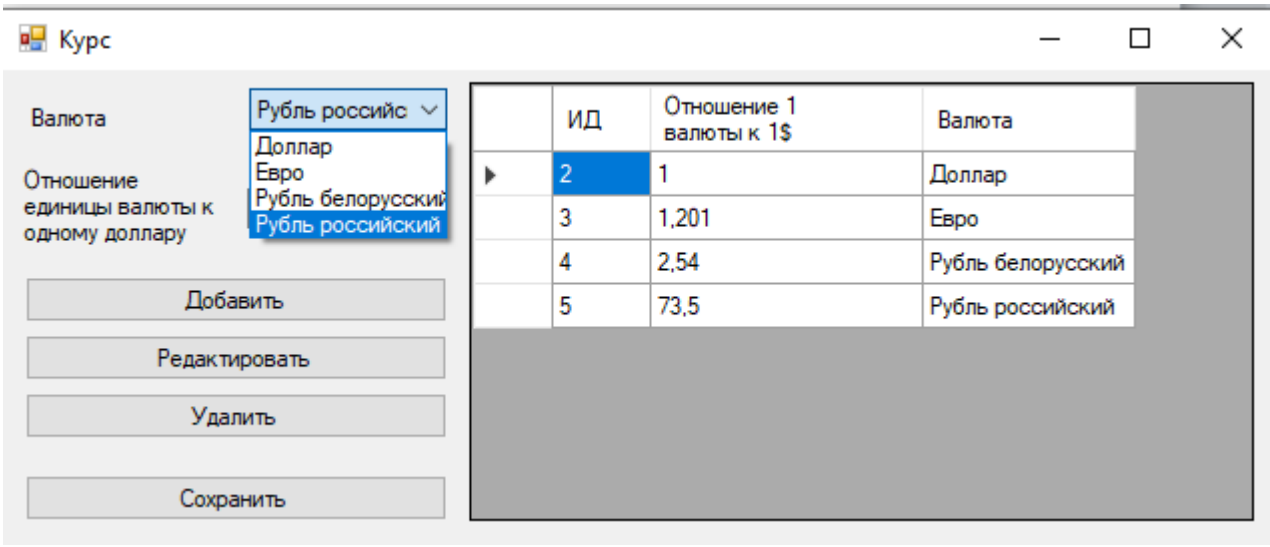


Рисунок 6.4 – Типы операций

При работе с таблицей курса уже необходимо выбирать валютуиз списка, который мы заполняли ранее.



Рисунокк 6.5 – Курс

Последняя работа с формой учета операций на которой работаем с таблицей учета непосредственно и таблицами валют и типоа расхода относительно.

Из выпадающих списков выбирается валюта для операции и тип операции, который происходит с этой валютой. Затем, в поле количества валюты вносятся данные о количестве предоставляемой валюты, после которого в поле итоговой суммы высчитывается значение в долларах (для чего используется значения из таблицы курса).

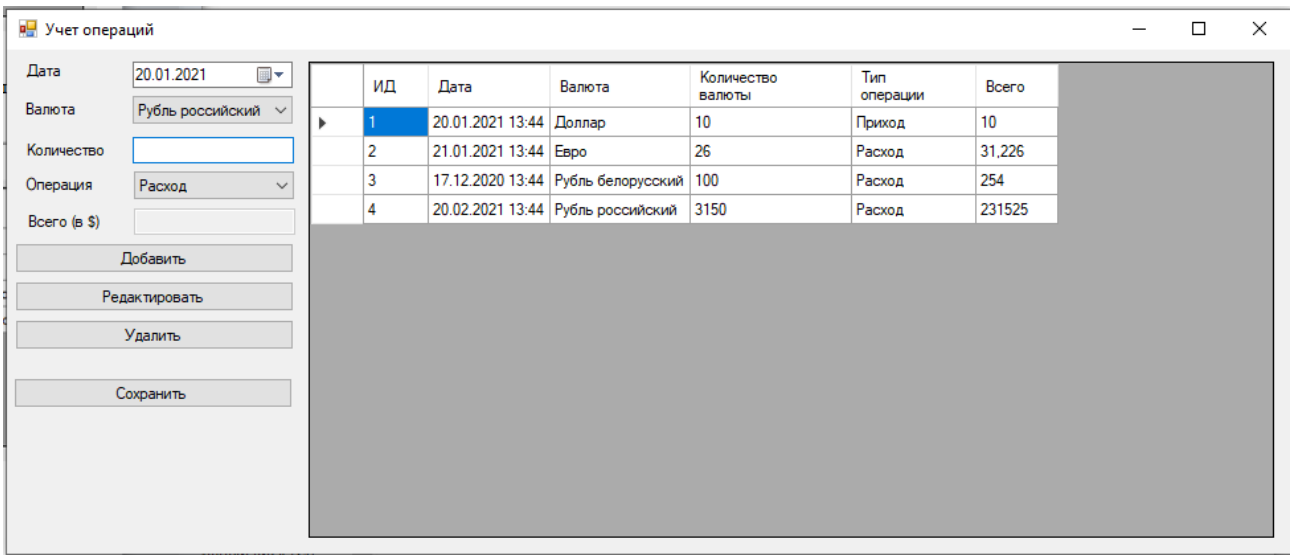


Рисунок 6.6 – Учет операций

Заключение

В ходе выполнения данной курсовой работы было разработано приложение для учета валютных операций, закреплены теоретические знания и отработаны практические навыки объектно-ориентированного проектирования и программирования на языке C#, изучена литература по объектно-ориентированному программированию, проведено обучение работе в интегрированной среде программирования Visual Studio 2019..

В рамках разработки приложения было проведено логическое проектирование структуры базы данных, физическое проектирование базы данных. Была рассмотрена архитектура MVC.

В практической части работы было разработано приложение для учета валютных операций при помощи языка программирования высокого уровня C#.

Список использованных источников

1. Коваленко, Е.А. Базы и банки данных: практическое пособие для студентов специальности I – 53 01 02 «Автоматизированные системы обработки информации» / Е.А.Коваленко, В.Н.Леванцов. – Гомель: учреждение образования «Гомельский государственный университет имени Франциска Скорины», 2007. – 95 с.

2. Лекция №4 – Проектирование баз данных Подходы к проектированию базы данных // StudFiles [Электронный ресурс]. – 2019. – Режим доступа: https://studfile.net/preview/1705211/. – Дата доступа: 31.12.2020.

3. MySQL/Руководство для начинающих [Электронный ресурс]. – Режим доступа: https://wiki.gentoo.org/wiki/MySQL/Startup_Guide/ru. – Дата доступа: 31.12.2020.

4. Руководство по MVC [Электронный ресурс]. – Режим доступа: https://metanit.com/sharp/mvc5/. – Дата доступа: 31.12.2020.

ПРИЛОЖЕНИЕ А

Правила генерации отношений по диаграммам ER-типа

ПРАВИЛО 1. Если показатель кардинальности бинарной связи равен 1:1 и класс принадлежности обеих сущностей является обязательным, то требуется только **одно** отношение. Первичным ключом этого отношения может быть ключ любой из двух сущностей.

ПРАВИЛО 2. Если показатель кардинальности бинарной связи равен 1:1 и класс принадлежности одной сущности является обязательным, а другой – необязательным, то необходимо построение двух отношений. Под каждую сущность необходимо выделение одного отношения, при этом ключ сущности должен служить первичным ключом для соответствующего отношения. Кроме того, ключ сущности, для которой класс принадлежности является необязательным, добавляется в качестве атрибута в отношение, выделенное для сущности с обязательным классом принадлежности.

ПРАВИЛО 3. Если показатель кардинальности бинарной связи равен 1:1 и класс принадлежности ни одной сущности не является обязательным, то необходимо использовать три отношения: по одному для каждой сущности, ключи которых служат в качестве первичных в соответствующих отношениях, и одного для связи. Среди своих атрибутов отношение, выделяемое связи, будет иметь по одному ключу сущности от каждой сущности.

ПРАВИЛО 4. Если показатель кардинальности бинарной связи равен 1:n и класс принадлежности n-связной сущности является обязательным, то достаточным является использование двух отношений, по одному на каждую сущность, при условии, что ключ сущности каждой сущности служит в качестве первичного ключа для соответствующего отношения. Дополнительно ключ односвязной сущности должен быть добавлен как атрибут в отношение, отводимое n-связной сущности.

ПРАВИЛО 5. Если показатель кардинальности бинарной связи равен 1:n и класс принадлежности n-связной сущности является необязательным, то

необходимо формирование трех отношений: по одному для каждой сущности, причем ключ каждой сущности служит первичным ключом соответствующего отношения, и одного отношения для связи. Связь должна иметь среди своих атрибутов ключ сущности от каждой сущности.

ПРАВИЛО 6. Если показатель кардинальности бинарной связи равен m:n, то для хранения данных необходимо три отношения: по одному для каждой сущности, причем ключ каждой сущности используется в качестве первичного ключа соответствующего отношения, и одного отношения для связи. Последнее отношение должно иметь в числе своих атрибутов ключ сущности каждой сущности.

ПРИЛОЖЕНИЕ Б

Описание нормальных форм

В теории реляционных баз данных обычно выделяется следующая последовательность нормальных форм:

- первая нормальная форма (1NF);
- вторая нормальная форма (2NF);
- третья нормальная форма (3NF);
- нормальная форма Бойса-Кодда (BCNF);
- четвертая нормальная форма (4NF);
- пятая нормальная форма, или нормальная форма проекции-соединения (5NF или PJ/NF).

1НФ – таблица находится в первой нормальной форме (1НФ) тогда и только тогда, когда ни одна из ее строк не содержит в любом своем поле более одного значения и ни одно из ее ключевых полей не пусто. (Любое поле таблицы содержит неделимую информацию и в таблице определен первичный ключ).

2НФ – Таблица находится во второй нормальной форме (2NF) в том и только в том случае, когда находится в 1NF, и каждый ее не ключевой атрибут полностью зависит от первичного ключа.

Если таблица в 1 НФ имеет простой первичный ключ, то она автоматически находится и во 2 НФ.

3НФ – Таблица находится в третьей нормальной форме (3NF) в том и только в том случае, если находится в 2NF и каждый не ключевой атрибут не транзитивно зависит от первичного ключа. (Иными словами, таблица должна находиться во второй нормальной форме и ни одно из ее не ключевых полей не должно однозначно идентифицироваться значением другого не ключевого поля (полей)).

Теоретики реляционных систем Кодд и Бойс обосновали и предложили более строгое определение для 3НФ, которое учитывает, что в таблице может быть несколько возможных ключей. Таблица, соответствующая этому опреде-

лению называется таблицей в улучшенной третьей форме или таблицей в нормальной форме Бойса-Кодда.

Таблица находится в нормальной форме Бойса-Кодда (НФБК), если и только если любая функциональная зависимость между его полями сводится к полной функциональной зависимости от возможного ключа.

В следующих нормальных формах (4НФ и 5НФ) учитываются не только функциональные, но и многозначные зависимости между полями таблицы. Для их описания познакомимся с понятием полной декомпозиции таблицы.

Полной декомпозицией таблицы называют такую совокупность произвольного числа ее проекций, соединение которых полностью совпадает с содержимым таблицы.

Теперь можно дать определения высших нормальных форм. И сначала будет дано определение для последней из предложенных - 5НФ.

5НФ – Таблица находится в пятой нормальной форме тогда и только тогда, когда в каждой ее полной декомпозиции все проекции содержат возможный ключ. Таблица, не имеющая ни одной полной декомпозиции, также находится в 5НФ.

4НФ – Четвертая нормальная форма является частным случаем 5НФ, когда полная декомпозиция должна быть соединением ровно двух проекций. Весьма не просто подобрать реальную таблицу, которая находилась бы в 4НФ, но не была бы в 5НФ.

Третья нормальная форма считается оптимальной для небольших баз данных, при проведении дальнейшей нормализации следует учитывать, что при увеличении количества связанных таблиц увеличивается время обработки информации, хранящейся в них. Поэтому разработчик должен сам принимать решение о том, какая ступень нормализации будет оптимальной для проектируемой базы данных.

ПРИЛОЖЕНИЕ В

Содержание файла AccauntingTransaction.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace AccountingCurrencyTransactions
{
   public partial class AccauntingTransaction : Form
   {
      /// <summary>
      /// Конструктор по умолчанию
      /// </summary>
      public AccauntingTransaction()
      {
         InitializeComponent();
      }
      /// <summary>
      /// Метод, срабадывающий при загрузке формы на экран. Обновляет все таблицы и выпадающие списки
      /// </summary>
      /// <param name="sender">Входящий объект</param>
      /// <param name="e">Аргументы события</param>
      private void AccauntingTransaction_Load(object sender, EventArgs e)
```

```
        {
            fillGrid();
        }
        /// <summary>
        /// Метод, выводящий предыдущую форму при закрытии текущей
        /// </summary>
        /// <param name="sender">Входящий объект</param>
        /// <param name="e">Аргументы события</param>
        private void Position_FormClosing(object sender, FormClosingEventArgs
e)
        {
            Main.main.Show();
        }
        /// <summary>
        /// Метод, разрешающий вставлять только определенный набор сим-
волы. Только цифры
        /// </summary>
        /// <param name="sender">Входящий объект</param>
        /// <param name="e">Входящее событие</param>
        private void textBox_OnlyNumbers(object sender, KeyPressEventArgs e)
        {
            char number = e.KeyChar;
            if (!Char.IsDigit(number) && number != 8) // цифры и клавиша
BackSpace
            {
                e.Handled = true;
            }
        }
```

```
        /// <summary>
        /// Метод заполнение таблицы данными из бд
        /// </summary>
        private void fillGrid()
        {

this.courseYMTableAdapter.Fill(this.accountingCurrencyTransactionsDataSet.CourseYM);

this.typeOperationYMTableAdapter.Fill(this.accountingCurrencyTransactionsDataSet.typeOperationYM);

this.currencyYMTableAdapter.Fill(this.accountingCurrencyTransactionsDataSet.currencyYM);

this.accountingForTransactionsYMTableAdapter.Fill(this.accountingCurrencyTransactionsDataSet.AccountingForTransactionsYM);
            fixName();
        }
        /// <summary>
        /// Метод проверки полей на заполнение
        /// </summary>
        /// <returns></returns>
        private bool isFill()
        {
            if (textBox1.Text.Length < 1)
            {
                MessageBox.Show("Не все поля заполнены!");
                return false;
            }
```

```
    return true;
}
/// <summary>
/// Метод очистки полей
/// </summary>
private void clearFields()
{
    textBox1.Text = "";
    textBox2.Text = "";
}
//add
/// <summary>
/// Метод добавления значения а базу данных. Он получает все значения из введенных полей и выбранных выпадающий списков и вносит в новую строку таблицы
/// После добавление данных в таблицу идет перезагрузка таблиц и списков формы
/// </summary>
/// <param name="sender">Входящий объект</param>
/// <param name="e">Аргументы события</param>
private void button1_Click(object sender, EventArgs e)
{
    if (isFill())
        try
        {
            DataRowView row = (DataRowView)accountingForTransactionsYMBindingSource.AddNew();

            row[1] = dateTimePicker1.Value;
            row[2] = comboBox1.SelectedValue;
```

```
                row[3] = textBox1.Text;
                row[4] = comboBox2.SelectedValue;
                row[5] = textBox2.Text;

                accountingForTransactionsYMBindingSource.EndEdit();

this.accountingForTransactionsYMTableAdapter.Update(accountingCurrencyTransa
ctionsDataSet);
                clearFields();
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
        fillGrid();
    }
    //edit
    /// <summary>
    /// Метод редактирования данных в базе данных. К выбранной записи
обновляются соответствующие поля, после чего ижет сохранение изменений в
базе данных
    /// После изменений обновляются все таблицы и выпадающие списки
формы
    /// </summary>
    /// <param name="sender"></param>
    /// <param name="e"></param>
    private void button2_Click(object sender, EventArgs e)
    {
        if (isFill())
            try
```

```
                {
                    dataGridView1.CurrentRow.Cells[1].Value = dateTimePicker1.Value;
                    dataGridView1.CurrentRow.Cells[2].Value = comboBox1.SelectedValue;
                    dataGridView1.CurrentRow.Cells[4].Value = textBox1.Text;
                    dataGridView1.CurrentRow.Cells[5].Value = comboBox2.SelectedValue;
                    dataGridView1.CurrentRow.Cells[7].Value = textBox2.Text;

                    accountingForTransactionsYMBindingSource.EndEdit();

this.accountingForTransactionsYMTableAdapter.Update(((DataRowView)dataGridView1.CurrentRow.DataBoundItem).Row);

                    clearFields();
                }
                catch (Exception ex)
                {
                    MessageBox.Show(ex.Message);
                }
            fillGrid();
        }

        //delete
        /// <summary>
        /// Метод удаления данных  из базы данных
        /// Берется ид выбранной записи и удаляется из бд
        /// </summary>
        /// <param name="sender"></param>
```

```
        /// <param name="e"></param>
        private void button3_Click(object sender, EventArgs e)
        {
            if (dataGridView1.Rows.Count > 0 && dataGridView1.CurrentRow !=
null)
            {
                try
                {
                    accountingCurrencyTransactionsDataSet.AcceptChanges();
                    accountingForTransaction-
sYMBindingSource.RemoveAt(dataGridView1.CurrentRow.Index);
                    accountingForTransactionsYMBindingSource.EndEdit();
                    accountingForTransactionsYMTableAdapt-
er.Update(accountingCurrencyTransactionsDataSet.AccountingForTransactionsYM);
                }
                catch (Exception ex)
                {
                    MessageBox.Show(ex.Message);
                }
                fillGrid();
            }
        }

        /// <summary>
        /// Метод вывода данных выбранной записи из таблицы в формы
приложения. Так, при каждом нажатии на запись в таблице, метод получает
данные из нее и в каждое поле заполняется значением из соответствующего
столбца.
        /// </summary>
        /// <param name="sender">Входящий объект</param>
```

```
        /// <param name="e">Аргументы события</param>
        private void dataGridView1_Click(object sender, EventArgs e)
        {
            if (dataGridView1.Rows.Count > 0 && dataGridView1.CurrentRow != null)
            {
                comboBox1.SelectedValue = dataGridView1.CurrentRow.Cells[2].Value.ToString();
                comboBox2.SelectedValue = dataGridView1.CurrentRow.Cells[5].Value.ToString();
                textBox1.Text = dataGridView1.CurrentRow.Cells[4].Value.ToString();
                textBox2.Text = dataGridView1.CurrentRow.Cells[7].Value.ToString();
            }
        }
        /// <summary>
        /// Метод, исправляющий отображение наименований из других таблиц.
        /// Походу в базе данных идет работа по Ид, приходится эти ИД выводить в таблицу.
        /// Чтобы пользвоатель понимал о чем речь, этот метод по Ид получает наименование записи в таблице и выводти его имя в таблицу
        /// </summary>
        private void fixName()
        {
            for (int i = 0; i < dataGridView1.RowCount; i++)
            {
                comboBox1.SelectedItem = comboBox1.Items[
                                                currencyYMBindingSource.Find(
```

```
                                                    "id",
                                                    int.Parse(dataGridView1[2,
i].Value.ToString())
                                                    )
                                                ];
                    dataGridView1[3, i].Value = comboBox1.Text;


                    comboBox2.SelectedItem = comboBox2.Items[
                                                typeOperationYMBindingSource.Find(
                                                    "id",
                                                    int.Parse(dataGridView1[5,
i].Value.ToString())
                                                    )
                                                ];
                    dataGridView1[6, i].Value = comboBox2.Text;
                }
            }
            /// <summary>
            /// Расчет значения Общей суммы
            /// </summary>
            private void setSumm() {
                textBox2.Text = "";
                try
                {
                    for (int i = 0; i < dataGridView2.RowCount; i++)
                    {
                        if                                          (dataGridView2[1,
i].Value.ToString().Equals(comboBox1.SelectedValue.ToString()))
                            {
```

```
                    double summ = double.Parse(dataGridView2[2, i].Value.ToString()) * int.Parse(textBox1.Text);
                    textBox2.Text = summ.ToString();
                }
            }
        } catch (Exception e) {
            MessageBox.Show(e.ToString());
            MessageBox.Show("Не удается преобразовать количество валюты в число \n" +
                    "или не внесен курс отношения валюты к доллару");
        }
    }

    private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
    {
        if (textBox1.Text.Length > 0) setSumm();
    }

    private void textBox1_KeyUp(object sender, KeyEventArgs e)
    {

        if (textBox1.Text.Length > 0) setSumm();
    }

    private void button4_Click(object sender, EventArgs e)
    {
        Saver.Save(dataGridView1);
    }  }}
```