Федеральное агентство по образованию

Федеральное государственное бюджетное образовательное учреждение

высшего профессионального образования

«Самарский государственный университет»

Механико-математический факультет

Кафедра «Информатики и

вычислительной математики»

Специальность «Математическое

Обеспечение и Администрирование

Информационных Систем»

Курсовая работа

«Модельная база данных: учет личных финансов»

Выполнил:

студент группы 2241

Астафьев Владислав Сергеевич.

_____

Научный руководитель

к.ф.-м.н., доцент

Рогачева Е.В.

_____

Работа защищена

«___» _____2013 г.

оценка _____

Зав. кафедрой

д.ф.-м.н., профессор

Степанов А.Н.

_____

Самара 2013

Содержание

Введение

Целая индустрия рекламного бизнеса работает каждый день с утра до вечера, не покладая рук, только для того, чтобы люди приходили и покупали то "жизненно необходимое", что предлагают по телевизору, в газетах и журналах, по радио - словом, повсеместно. Но через некоторое время чаще всего оказывается, что покупка была не такой уж и важной. Другой распространенной проблемой людей является неумение контролировать свои расходы. Неоднократно приходится наблюдать, что человек, получивший зарплату, тратит ее большую долю в первый же день, а к середине месяца денег практически не остается. Исходя из изложенного, учёт личных финансов является начальным и важнейшим звеном на протяжении всех этапов финансового планирования.

Ведение учета личных финансов позволяет решать следующие задачи:

- найти "лишние" деньги в своем кармане;
- понять причины проблем с деньгами, и найти варианты для их решения;
- комфортно жить, не боясь остаться без средств существования;
- выработать в себе привычки, которые будут вести вас к финансовой свободе;
- перестать жить в долг, полностью распоряжаться своей жизнью и своими деньгами;
- реализовать личные финансовые планы и цели;
- обеспечить своим детям финансовое благополучие.

Для того чтобы вести учет своих доходов и расходов, планировать достижение финансовых целей, контролировать кредиты и долги, создавать резервы и накопления, можно использовать следующие способы вести учет:

- на бумаге, в тетради, ежедневнике;
- в табличках MS Excel или в MS Word;
- в специализированном программном продукте, предназначенном для учета личных финансов.

Учет на бумаге, в тетради, ежедневнике не требует наличия компьютера, не нужно обучаться работе на компьютере вообще и в программах учета в частности. Но данный способ не нагляден, велика вероятность ошибок, требует большого количества времени для ведения учета и анализа. Следовательно, этот способ наиболее актуален для регионов, где еще нет повсеместного использования компьютерной техники.

Учет в таблицах MS Excel или в MS Word позволяет не приобретать и не устанавливать специализированные программные продукты. Можно быстро и гибко настроить простой учет под свои нужды. Однако сложно разносторонне подойти к учету личных финансов, выстраивать зависимости между таблицами, велика вероятность ошибок. Кроме того, сложно анализировать данные. Следовательно, этот способ наиболее актуален для ведения упрощенного учета личных финансов, а также если нет возможности приобрести и(или) разбираться в специализированных программах.

Учет в специализированном программном продукте позволяет проводить разносторонний учет и анализ личных финансов. Логика ведения учета продумана разработчиками программы, от вас требуется просто вовремя заносить необходимые данные и анализировать результаты. Этот способ требует минимального количества времени и постоянно развивается, добавляя новые возможности для ведения учета. Пользователь может накапливать данные за весь период ведения учета, что позволяет наблюдать динамику. Минусы данного способа в том, что он требует некоторых временных и денежных затрат на начальном этапе. В некоторых случаях приходится подстраиваться под логику и методы учета, реализованные в программе, что иногда бывает неудобно.

Использовать любой из вышеперечисленных способов на порядок лучше, чем не следить за своими финансами вообще. Но во много раз правильнее, быстрее и эффективнее воспользоваться специализированной программой.

Сегодня на рынке достаточно много программных продуктов, которые

способны помочь в учете и контроле личных финансов. Например, «1С:Деньги8"», «Личные финансы», «CashFly», «Семейный бюджет», «Домашние финансы», которые также позволяют вводить данные, группировать их по базам данных, считать суммы расходов и доходов, а также строить графики и диаграммы. Однако большинство из них дорогостоящие, и разобраться в них достаточно сложно. Поэтому возникает необходимость в разработке простой для понимания бесплатной информационной системы, с интуитивно понятным пользовательским интерфейсом.

Следовательно, **целью** данной курсовой работы является создание удобной в использовании ИС, выполненной по архитектуре клиент-сервер с обеспечением максимально возможной независимой клиентской части и БД «Учет личных финансов», обеспечивающей хранение в электронной форме данных о доходах и расходах семьи. При этом все модули клиентского приложения должны быть выполнены по единой схеме, что обеспечивает максимально простую модернизацию, а также поиск и исправление ошибок.

Для достижения поставленной цели в курсовой работе выполняется:

1. Изучение и описание предметной области.
2. Выбор на основе проведенного анализа инструментальных средств.
3. Создание логической модели данных.
4. Физическая реализация базы данных.
5. Разработка клиентского приложения, отвечающего требованиям, определенным во время исследования.
6. Тестирование

Глава 1. Анализ предметной области

Под доходами населения понимается сумма денежных средств и материальных благ, полученных или произведенных домашними хозяйствами за определенный промежуток времени. Личный доход - денежное или материальное вознаграждение, получаемое одним физическим лицом или семьей за выполненную работу. В личный доход входят все средства, заработанные как на основной, так и на дополнительной работе, предпринимательский доход, доход по ценным бумагам, вкладам и депозитам и т.д. В зависимости от учета динамики уровня потребительских цен доход подразделяется на:

- Номинальный – это количество денег, полученное в определённый период отдельным лицом; также он характеризует уровень денежных доходов независимо от налогообложения.
- Располагаемый доход – доход, который может быть использован на личное потребление и личные сбережения. Располагаемый доход меньше номинального дохода на сумму налогов и обязательных платежей.
- Реальный – представляет собой количество товаров и услуг, которое можно купить на располагаемый доход в течение определенного периода.

По форме единицы дохода выделяют:

- денежный (оплата труда, доходы от предпринимательской деятельности, пенсии, стипендии, пособия, социальные выплаты; поступления от собственности, в качестве процентов по вкладам, ценные бумагам, дивиденды; доходы от продажи продукции сельского хозяйства, страховые возмещения, сумма от продажи иностранной валюты и многие другие);
- натуральный (сюда относятся некоторые выплаты из социальных фондов, продукты, произведенные в личных подсобных хозяйствах, и услуги, оказываемые членами семьи в домашнем хозяйстве).

В зависимости от государственного вмешательства:

- первичный, образованный под воздействием рыночного механизма;
- вторичный, формирование которого связано с перераспределительной политикой государства.

Понятие “личный доход” охватывает все виды доходов, начисленных в денежных и натуральных формах (независимо от источников финансирования), включая денежные суммы, начисленные физическим лицам в соответствии с законодательством за не проработанное время (ежегодный отпуск, праздничные дни и т.п.). В рыночной экономике основными источниками личных доходов являются:

- трудовая деятельность персонала, работающего по найму, и лиц свободных профессий;
- предпринимательская деятельность;
- собственность;
- средства государства и предприятий, распределенные в соответствии с принадлежностью к определенной социальной группе и категории персонала;
- личные подсобные хозяйства.

Первому из указанных источников соответствует доход в виде заработной платы и гонорара; третьему – дивиденды и проценты на капитал; четвертому – трансфертные платежи (пенсии, пособия, стипендии и т.д.), а также услуги предприятия своим работникам в виде медицинского обслуживания, повышения квалификации и т.д., пятому – продукты, возможности для отдыха, денежные средства от личных хозяйств.

Оплата труда выступает в качестве одного из основных составных элементов доходов населения. Доход от оплачиваемой работы составляет вознаграждение за труд работающих по найму, регулируемое договорной или контрактной системой отношений. Заработная плата еще подразделяется на

номинальную и реальную, начисленную и фактически выплаченную, среднюю и минимальную.

Значительное влияние на получаемые населением доходы оказывают выплаты по программам социальной помощи со стороны государства. Сюда относятся пенсионное обеспечение, выплаты на содержание нетрудоспособных, различных пособий, стипендий студентам и учащимся. Их особенность в отличие от заработной платы, состоит в характере получения, независимом от количества и качества труда. Иначе говоря, трансферты – это операции, при которых товары, услуги или денежные средства предоставляются в одностороннем порядке без получения какого-либо эквивалента взамен. Социальные трансферты в натуральной форме состоят из товаров и нерыночных услуг, предоставляемых конкретным домашним хозяйствам из федерального и местных бюджетов и общественных организаций бесплатно.

Глава 2. Проектирование и разработка БД

2.1. Use-Case диаграмма

Для лучшего понимания разрабатываемой информационной системы необходимо построить концептуальную модель данных, а именно UML-диаграмму. Имеется внешний объект — пользователь, который выступает в качестве актера. Актер взаимодействует с системой, в результате этого взаимодействия, он может выполнять одно из трех действий:

- Вводить данные.
- Получать данные.
- Создавать отчет.

Вариант использования [1] «Получить данные» включает такие компоненты, как «Учет доходов» и «Учет расходов». В результате получается следующую use-case диаграмму:

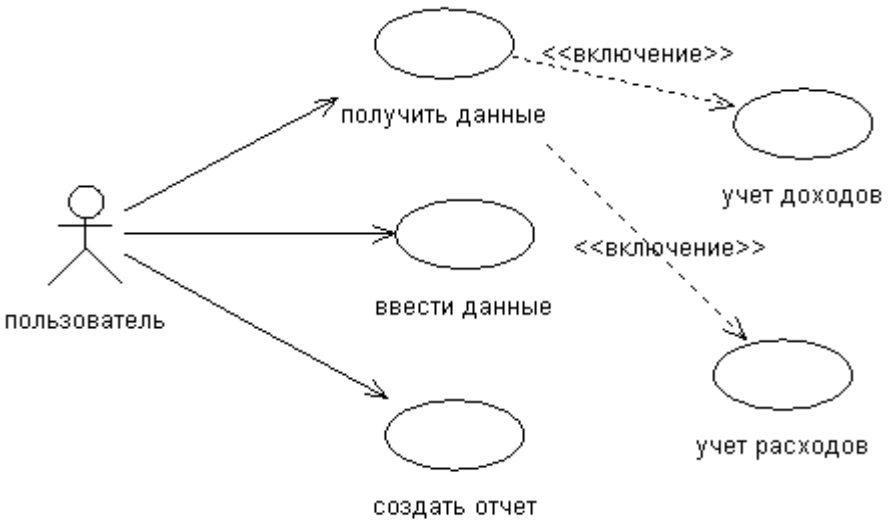

Рис1. UML-диаграмма

2.2. Выделение сущностей, атрибутов, связей

В результате исследования предметной области стало ясно, что основной сущностью является Человек.

Исходя из того, что база данных может хранить информацию о нескольких

людях (если оно используется для учета доходов семьи), то сущность человек имеет атрибуты:

- идентификатор человека;
- Ф.И.О. Человека;
- возраст человека.

Человек имеет определенные виды доходов:

- Основной доход.
- Дополнительный доход.
- Государственные пособия.
- Депозит.

Под «основным доходом» понимается заработная плата по основному месту работы, а также постоянный доход, например, доход от аренды недвижимости или доход от акций.

Следовательно, сущность имеет следующие атрибуты:

- идентификатор дохода;
- тип дохода;
- частота дохода;
- размер дохода.

Под «дополнительным доходом» понимаются какие-то другие виды заработка, а также различные неожиданные доходы, например, наследство, выигрыш, подарок и т. д.

Сущность «дополнительный доход» имеет следующие атрибуты:

- идентификатор дохода;
- тип дохода;
- размер дохода.

«Государственные пособия» включают в себя различные государственные выплаты по каким-то социальным причинам. Данная сущность имеет те же атрибуты, что и сущность «Основной доход».

«Депозит» выделяется в отдельную сущность, т. к. он имеет отличные от

других видов дохода параметры. Депозит – это деньги, которые клиент передаёт банку на ограниченный срок с определёнными условиями. Под условиями понимаются сами проценты, частота начисления. Следовательно, атрибутами этой сущности будут являться:

- идентификатор депозита;
- тип депозита;
- частота начисления;
- размер суммы;
- процент;
- срок, на который взят депозит.

Сущности «Персональный основной доход», «Персональный дополнительный доход», «Персональные гос-пособия» и «Персональный депозит», являясь альтернативой связи многие-ко-многим, хранят информацию о человеке и его видах доходов. Эти сущности идентичны и имею одинаковые атрибуты:

- идентификатор персонального дохода;
- идентификатор человека;
- идентификатор дохода.

Последняя деталь логической модели — это сущность «Расходы», которая также является справочником и хранит информацию о человеке и произведенном им виде расхода: типе, дате, размере.

2.3. Логическая модель данных

ER-диаграмма логической модели включает в себя: [1]

- Список сущностей предметной области
- Полный список атрибутов каждой сущности
- Списки ключевых атрибутов
- Описание всех взаимосвязей между сущностями

Подробное описание выделения сущностей, их атрибутов и связей было

выполнено в предыдущем пункте.

С учетом имеющихся данных была получена и построена в Visual Paradigm for UML 7.1 следующая модель:

Рис2.Логическая модель данных

2.4. Нормализация

Для проектирования БД недостаточно просто построить модель данных. Необходимо проследить, чтобы таблицы, в которые превратятся сущности после преобразования модели данных, не обладали различного рода аномалиями. [2]

Итак, рассмотрим наши отношения на предмет наличия в них аномалий. К счастью, за счет свойств самой предметной области все отношения содержат информацию лишь об одной сущности, и мы можем надеяться на отсутствие аномалий. Кроме того, для всех данных, которые могут быть в отношении больше чем в одном экземпляре, введены отдельные отношения. Это еще сильнее снижает риск возникновения аномалий. Для таких данных, например, нет аномалии вставки.

Все возникающие зависимости имеют в качестве детерминанта первичный ключ. При этом ни в одной из таблиц нет взаимно зависимых не ключевых параметров. Все атрибуты, входящие в состав первичного ключа любой таблицы также являются независимыми. У многих таблиц есть альтернативные ключи. Как правило, это числовые идентификаторы, не несущие смысловой нагрузки и используемые для оптимизации операций поиска, выборки, а также в качестве ключей для внешних связей.

Это значит, что отношения находятся в ЗНФ (по определению отношение находится в ЗНФ если в нем нет транзитивных зависимостей и любой не ключевой атрибут неприводимо зависит от первичного ключа).

2.5. Выбор СУБД. Oracle Database 10g Express Edition

До недавнего времени, из-за лицензионных ограничений, использование коммерческих баз данных, таких как Oracle, в средах разработки было затруднено. В прошлом, любой мог свободно скачать и использовать программное обеспечение баз данных Oracle® для целей разработки, но любое промышленное использование требовало приобретение полной лицензии.[11]

С появлением версии Database 10g Express Edition (Oracle Database XE), и для разработчиков и для небольших производственных сред появилась возможность использовать программное обеспечение Oracle Database XE абсолютно бесплатно. Оно бесплатно для скачивания, разработки с ее помощью, бесплатно для внедрения и распространения. Исполняемый код Oracle Database XE может работать с объемом до 4GB пользовательских данных (в дополнение к системным данным Oracle) и на любой системе, используя до 1GB оперативной памяти и один процессор.

Oracle Database XE построена на основе кода Oracle Database 10g Release 2 и предоставляет полный набор интерфейсов программирования приложений (API) для разработчиков приложений. На платформе Red Hat Enterprise Linux она включает в себя Oracle Call Interface для C и C++, ODBC, PHP, JDBC и др.

Как и другие редакции Oracle, Oracle Database XE поставляется с предопределенной и полностью настроенной базой данных, включающей в себя схему HR в качестве примера. Это позволяет пользователю сразу попробовать примеры приложений, включенные в поставку или доступные для загрузки.

Oracle Database XE имеет следующие преимущества:

- абсолютно свободная СУБД;
- можно тестировать, разрабатывать и распространять с нулевыми инвестициями в ПО и без риска;
- легко мигрировать на промышленные редакции;
- не требуется переписывание приложений при переходе на промышленные редакции.

2.6. Физическая модель БД

Анализ сущностей, их атрибутов и связей позволяет сделать вывод: каждая сущность может быть представлена отдельной таблицей, причем, как было определено при нормализации, все они уже находятся в третьей нормальной форме. [1]

Переход от логического к физическому описанию модели состоит из следующих шагов:

1. Все простые сущности превращаются в отношения, имя сущности становится именем отношением.
2. Каждый атрибут становится возможным столбцом с тем же именем. Столбцы, соответствующие необязательным атрибутам, могут содержать NULL - значения.
3. Компоненты уникального идентификатора сущности превращаются в первичный ключ отношения.
4. Связи "многие к одному" становятся внешними ключами.
5. Указание ограничений целостности проектируемой базы данных и краткое описание полученных таблиц и их полей.
6. Каждый атрибут должен быть определен на конкретном домене, а каждый домен должен быть определен на допустимом простом типе данных.

Таким образом, чтобы получить физическую модель, необходимо в логической модели учесть такие особенности СУБД, как допустимые простые типы данных, наименование полей и таблиц, ограничение целостности данных и т.д.

С учетом описанного выше была построена ER-диаграмма физической модели, которая выглядит следующим образом:

Рис3.Физическая модель данных

Глава 3. Разработка базы данных «Учет личных финансов»

3.1. Описание таблиц базы данных

Т.к. каждое отношение может быть представлено отдельной таблицей, то с учетом всех известных данных структура БД «Учет личных финансов» состоит из десяти таблиц:

1) Таблица «Человек»;
2) Таблица «Основной доход»;
3) Таблица «Дополнительный доход»;
4) Таблица «Государственные пособия»;
5) Таблица «Депозит»;
6) Таблица «Персональный основной доход»;
7) Таблица «Персональный дополнительный доход»;
8) Таблица «Персональные государственные пособия»;
9) Таблица «Персональный депозит»;
10) Таблица «Расходы».

Рассмотрим более подробно каждую таблицу:

1) Таблица «Человек»

Man

Атрибуты	Тип данных	Description	PK	FK	UNIQUE	NOT NULL
id_man	integer	идентификатор человека	+	-	+	+
name	varchar(150)	Ф.И.О. человека	-	-	-	+
age	integer	возраст человека	-	-	-	-

2) Таблица «Основной доход»

Basic_income

Атрибуты	Тип данных	Description	PK	FK	UNIQUE	NOT NULL
id_bas	integer	идентификатор основного дохода	+	-	+	+
type_bas	varchar (100)	тип основного дохода	-	-	-	+
kind_bas	varchar (100)	вид основного дохода	-	-	-	+
freq_bas_in_month	real	частота основного дохода в месяц	-	-	-	-
size_bas	integer	размер основного дохода	-	-	-	+

3) Таблица «Дополнительный доход»

Additional_income

Атрибуты	Тип данных	Description	PK	FK	UNIQUE	NOT NULL
id_ad	integer	идентификатор дополнительного дохода	+	-	+	+
type_ad	varchar (100)	тип дополнительного дохода	-	-	-	+
size_ad	integer	размер дополнительного дохода	-	-	-	+

4) Таблица «Государственные пособия»

State_grants

Атрибуты	Тип данных	Description	PK	FK	UNIQUE	NOT NULL
id_st	integer	идентификатор гос-пособий	+	-	+	+
type_st	varchar (100)	тип гос-пособий	-	-	+	+

freq_st_in_month	integer	частота гос-пособий в месяц	-	-	-	-
size_st	integer	размер гос-пособий	-	-	-	+

5) Таблица «Депозит»

Deposit

Атрибуты	Тип данных	Description	PK	FK	UNIQUE	NOT NULL
id_dep	integer	идентификатор депозита	+	-	+	+
type_dep	varchar (100)	тип депозита	-	-	-	+
freq_of_charge_in_year	integer	частота начисления депозита в год	-	-	-	+
size_of_sum	integer	размер суммы депозита	-	-	-	+
percents	integer	проценты от депозита	-	-	-	+
period_in_years	real	срок хранения депозита в год	-	-	-	+

6) Таблица «Персональный основной доход»

Personal_basic_income

Атрибуты	Тип данных	Description	PK	FK	UNIQUE	NOT NULL
id_pers_bas	integer	идентификатор персонального основного дохода	+	-	+	+
id_man	integer	идентификатор человека	-	+	-	+
id_bas	integer	идентификатор основного дохода	-	+	-	+

7) Таблица «Персональный дополнительный доход»

Personal_additional_income

Атрибуты	Тип данных	Description	PK	FK	UNIQUE	NOT NULL
id_pers_ad	integer	идентификатор персонального дополнительного дохода	+	-	+	+
id_man	integer	идентификатор человека	-	+	-	+
id_ad	integer	идентификатор доп. дохода	-	+	-	+

8) Таблица «Персональные государственные пособия»

Personal_state_grants

Атрибуты	Тип данных	Description	PK	FK	UNIQUE	NOT NULL
id_pers_st	integer	идентификатор персональных гос-пособий	+	-	+	+
id_man	integer	идентификатор человека	-	+	-	+
id_st	integer	идентификатор гос-пособий	-	+	-	+

9) Таблица «Персональный депозит»

Personal_deposit

Атрибуты	Тип данных	Description	PK	FK	UNIQUE	NOT NULL
id_pers_dep	integer	идентификатор персонального депозита	+	-	+	+
id_man	integer	идентификатор человека	-	+	-	+
id_dep	integer	идентификатор депозита	-	+	-	+

10) Таблица «Расходы»

Expenses

Атрибут	Тип данных	Description	PK	FK	UNIQUE	NOT

ы						NULL
id_exp	integer	идентификатор расходов	+	-	+	+
id_man	integer	идентификатор человека	-	+	-	+
type_exp	varchar(100)	тип расходов	-	-	-	+
size_exp	integer	размер расходов	-	-	-	+
data	date	дата расходов	-	-	-	+

3.2. Выбор инструментария для разработки клиентской части приложения

Клиентская часть разработана в среде Delphi7. Среда разработки Delphi была выбрана, исходя из соображений простоты и эффективности проектирования. Кроме того, навыки работы с этой средой уже были приобретены ранее.

Действительно, процесс разработки в Delphi предельно упрощен, В первую очередь это относится к созданию интерфейса, на который уходит 80% времени разработки программы. Нужные компоненты просто помещаются на поверхность Windows-окна (в Delphi оно называется формой) и их свойства настраиваются с помощью специального инструмента (Object Inspector). С его помощью можно связать события этих компонентов с кодом его обработки - и вот простое приложение готово. Причем разработчик получает в свое распоряжение мощные средства отладки (вплоть до пошагового выполнения команд процессора), удобную контекстную справочную систему (в том числе и по Microsoft API); средства коллективной работы над проектом, всего просто не перечислить. [6]

В Delphi 7 реализовано достаточно большое число разнообразных технологий доступа к данным. Но последовательность операций при конструировании приложений баз данных остается почти одинаковой. И в работе используются по сути одни и те же компоненты, доработанные для применения с той или иной технологией доступа к данным.

Модуль данных

Для размещения компонентов доступа к данным в приложении баз данных желательно использовать специальную "форму" — модуль данных (класс TDataModule). Модуль данных не имеет ничего общего с обычной формой приложения, ведь его непосредственным предком является класс TComponent. В модуле данных можно размещать только невизуальные компоненты. Модуль данных доступен разработчику, как и любой другой модуль проекта, на этапе разработки. Пользователь приложения не может увидеть модуль данных во время выполнения.

Для создания структуры (модели, диаграммы) данных, с которой работает приложение, можно воспользоваться возможностями, предоставляемыми страницей Diagram Редактора кода. Любой элемент из иерархического дерева компонентов модуля данных можно перенести на страницу диаграммы и задать связи между ними.

При помощи управляющих кнопок можно задавать между элементами диаграммы отношения синхронного просмотра и главный/подчиненный. При этом производится автоматическая настройка свойств соответствующих компонентов. Для обращения компонентов доступа к данным, расположенным в модуле данных, из других модулей проекта необходимо включить имя модуля в секцию uses.

Преимуществом размещения компонентов доступа к данным в модуле данных является то, что изменение значения любого свойства проявится сразу же во всех обычных модулях, к которым подключен этот модуль данных. Кроме этого, все обработчики событий этих компонентов, т.е. вся логика работы с данными приложения, собраны в одном месте, что тоже весьма удобно.

Подключение набора данных

Компонент доступа к данным является основой приложения баз данных. На основе выбранной таблицы БД он создает набор данных и позволяет эффективно управлять им. [7]

При подключении к таблице используются компоненты для работы с Microsoft® ActiveX® Data Objects. ADO - это технология стандартного обращения к реляционным данным от Microsoft. Эта технология аналогична BDE по назначению и довольно близка по возможностям.

Условно процесс подключения базы данных к клиентскому приложению можно отобразить на рис4:

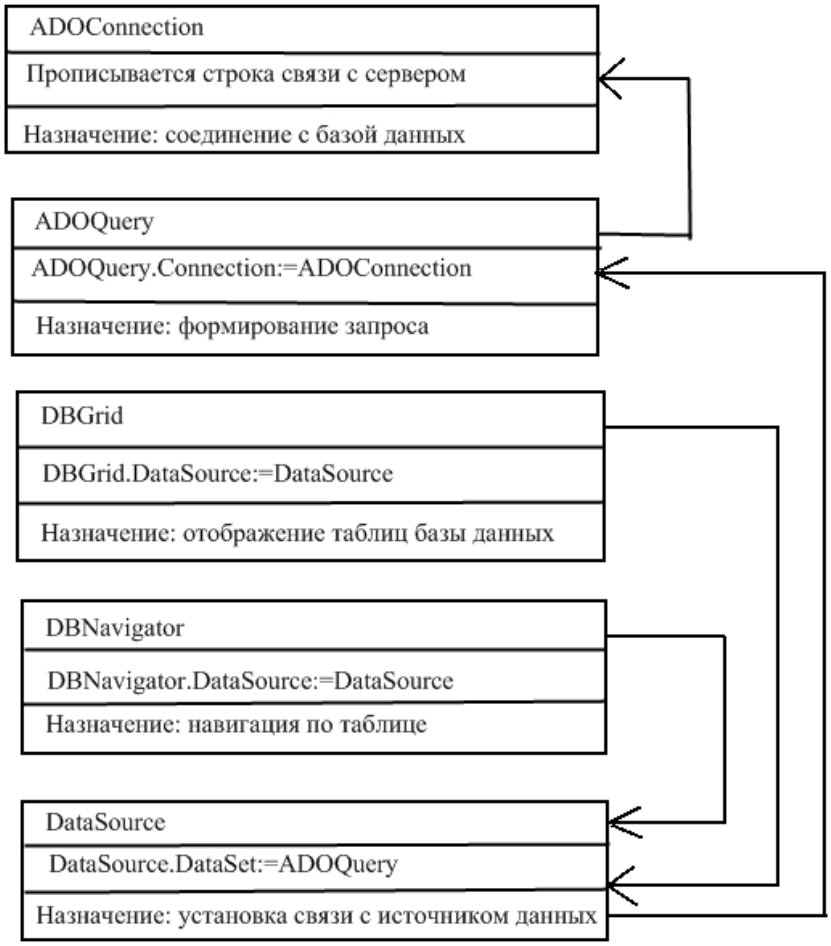

Рис4. Подключение базы данных к серверу СУБД

Обзор компонент

Для работы с ADO на вкладке компонентов ADO есть шесть компонентов: TADOConnection, TADOCommand, TADODataSet, TADOTable, TADOQuery, TADOStoredProc. [6]

TADOConnection аналогичен компоненту BDE TDatabase и используется для указания базы данных и работы транзакциями.

TADOTable – таблица доступная через ADO.

TADOQuery – запрос к базе данных. Это может быть как запрос, в результате которого возвращаются данные и базы (например, SELECT), так и запрос, не возвращающий данных (например, INSERT).

TADOStoredProc – вызов хранимой процедуры. В отличие от BDE и InterBase хранимые процедуры в ADO могут возвращать набор данных, поэтому компонент данного типа является потомком от TDataSet, и может выступать источником данных в компонентах типа TDataSource.

TADOCommand и TADODataSet являются наиболее общими компонентами для работы с ADO, но и наиболее сложными в работе. Оба компонента позволяют выполнять команды на языке провайдера данных (так в ADO называется драйвер базы данных).

Разница между ними в том, что команда, исполняемая через TADODataSet, должна возвращать набор данных и этот компонент позволяет работать с ними средствами Delphi (например, привязать компонент типа TDataSource). А компонент TADOCommand позволяет исполнять команды не возвращающие набор данных, но не имеет штатных средств Delphi для последующего использования возвращенного набора данных.

Очевидно, что все компоненты должны связываться с базой данных. Делается это двумя способами либо через компонент TADOConnection, либо прямым указанием базы данных в остальных компонентах. К TADOConnection остальные компоненты привязываются с помощью свойства Connection, к базе данных напрямую через свойство ConnectionString.

База данных может быть указана двумя способами через файл линка к данным (файл в формате Microsoft Data Link, расширение UDL), либо прямым заданием параметров соединения. Использование файлов Microsoft Data Link упрощает поддержку приложений, так как возможно использовать средства Windows для настройки приложения.

Глава 4. Реализация ИС

4.1. Серверная часть

Обеспечение ссылочной целостности

Ссылочная целостность — необходимое качество реляционной базы данных, заключающееся в отсутствии в любом её отношении внешних ключей, ссылающихся на несуществующие кортежи. [2]

Данная задача решается с помощью триггеров обновления и удаления. Для любой потенциально опасной операции над таблицей создаётся триггер, который производит необходимые проверки или даже изменяет данные в связанных таблицах, чтобы исключить потерю ссылок.

Так, для обеспечения каскадных изменений триггер может быть установлен на операцию изменения записи в таблице. Если окажется, что при редактировании изменилось значение ключевого поля, триггер должен произвести согласованные изменения во всех таблицах, связанных с данной, поменяв старое значение внешних ключей на новое.

Для исключения потери ссылок от некорректного редактирования внешнего ключа триггер должен при каждом изменении соответствующего поля проверять, имеется ли в связанной таблице запись с таким первичным ключом.

Для защиты от удаления записи, на которую имеются ссылки, триггер на связанной таблице должен при удалении проверять наличие ссылок и, в зависимости от необходимости, либо запрещать удаление, либо обнулять внешние ключи тем или иным образом.

Примеры триггеров, созданных при разработке Базы данных.

1) При изменении значений первичного ключа в таблице Человек должны поменяться значения внешних ключей в таблицах Персональный основной доход, Персональный дополнительный доход, Персональные гос.пособия, Персональный депозит, Расходы, если они есть.

```
create or replace trigger up_man
after update on man
for each row
begin
if (:old.id_man<>:new.id_man) then
update personal_basic_income set personal_basic_income.id_man=:new.id_man
where personal_basic_income.id_man=:old.id_man;
update personal_additional_income set personal_additional_income.id_man=:new.id_man
where personal_additional_income.id_man=:old.id_man;
update personal_state_grants set personal_state_grants.id_man=:new.id_man
where personal_state_grants.id_man=:old.id_man;
update personal_deposit set personal_deposit.id_man=:new.id_man
where personal_deposit.id_man=:old.id_man;
update expenses set expenses.id_man=:new.id_man
where expenses.id_man=:old.id_man;
end if;
end;
```

2) При удалении строки из таблицы Основной доход, удаляются и соответствующие записи в таблице Персональный основной доход.

```
create or replace trigger del_basic_income
before delete on basic_income
for each row
begin
delete personal_basic_income where personal_basic_income.id_bas=:old.id_bas;
end;
```

Кроме обеспечения ссылочной целостности в данной БД триггеры используются для формирования некоторых первичных ключей с помощью последовательности. Таким образом формируются все идентификаторы.

```
create sequence bas_id
increment by 1
start with 7
```

```
maxvalue 100
nocycle
nocache;

create or replace trigger key_basic_income
before insert on basic_income
for each row
begin
select bas_id.nextval into :new.id_bas from dual;
end;
```

4.2. Клиентская часть

Написанное приложение является MDI-приложением (multiple document interface). Каждое MDI-приложение имеет три основные составляющие: одну (и только одну) родительскую форму MDI, несколько дочерних форм MDI и основное меню MDI.

При запуске программы появляется меню, состоящее из пунктов:

- Таблицы
- Создать отчет
- Посмотреть информацию о редактировании

Первый пункт меню содержит имена таблиц базы данных. Пользователь может выбирать одну из таблиц для вставки, обновления или удаления данных таблицы.

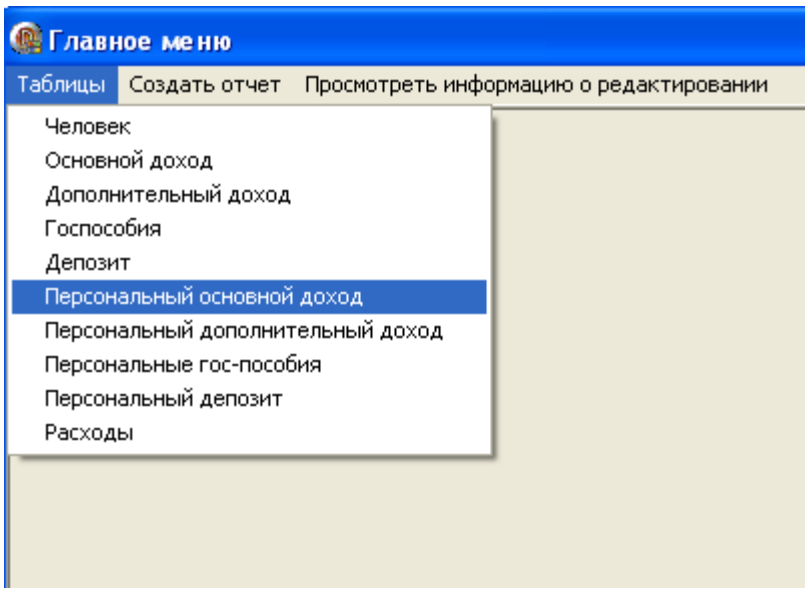

Рис5. Меню программы

При выборе пункта меню, например Основной доход, откроется форма, отображающая соответствующую таблицу, в которой можно отредактировать данные.

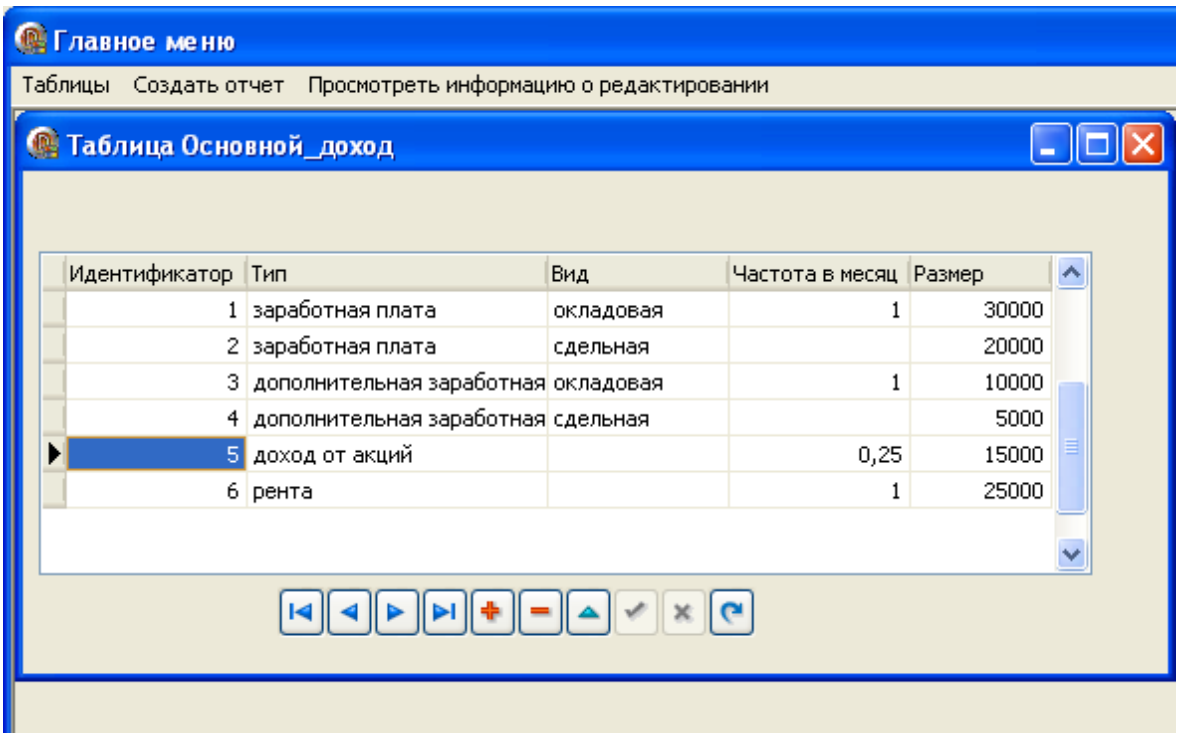

Рис6. Форма редактирования таблицы Основной доход

Чтобы добавить какому-либо человеку новый доход, необходимо выбрать один из пунктов меню — Персональный основной доход, Персональный дополнительный доход, Персональные гос-пособия, Персональный депозит. Откроется окно, которое отображает данные, у какого пользователя какой доход имеется, и 2 выпадающих списка, в которых можно выбрать кому и какой доход пользователь хочет добавить.

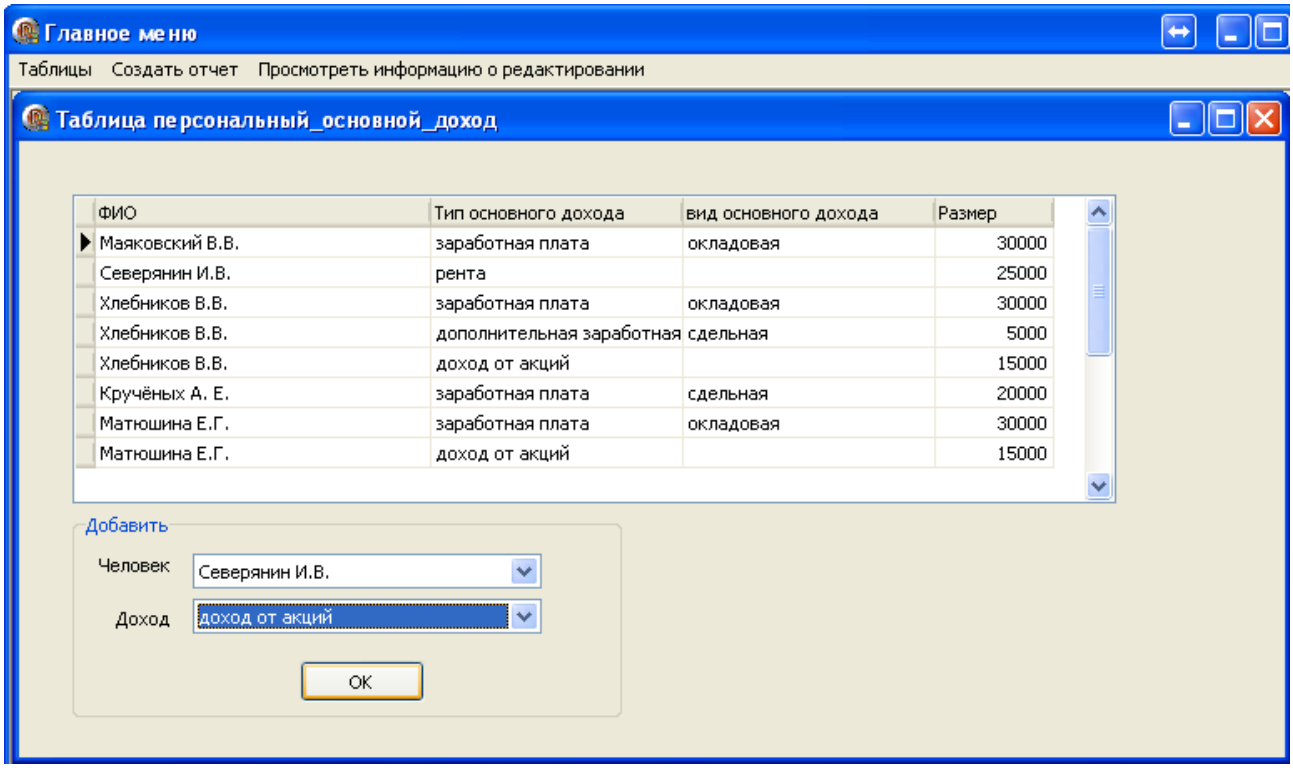

Рис7. Форма редактирования таблицы Персональный основной доход

Также пользователь может получить отчеты, содержащие сведения о размере суммы всех личных доходов человека, о размере суммы расходов человека, осуществленных в определенный диапазон времени, о том, что конкретный человек приобрел в определенный диапазон времени.

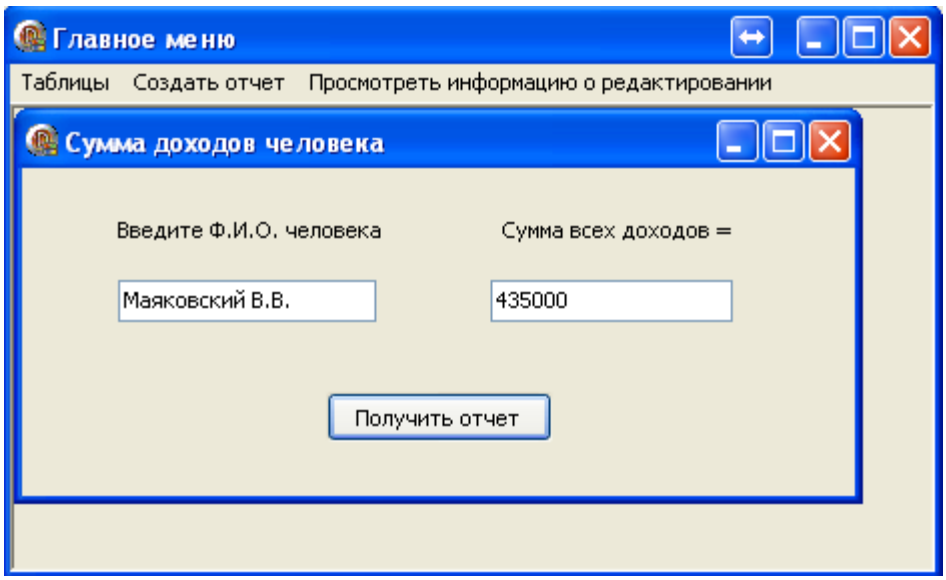

Рис8. Отчет по сумме всех доходов

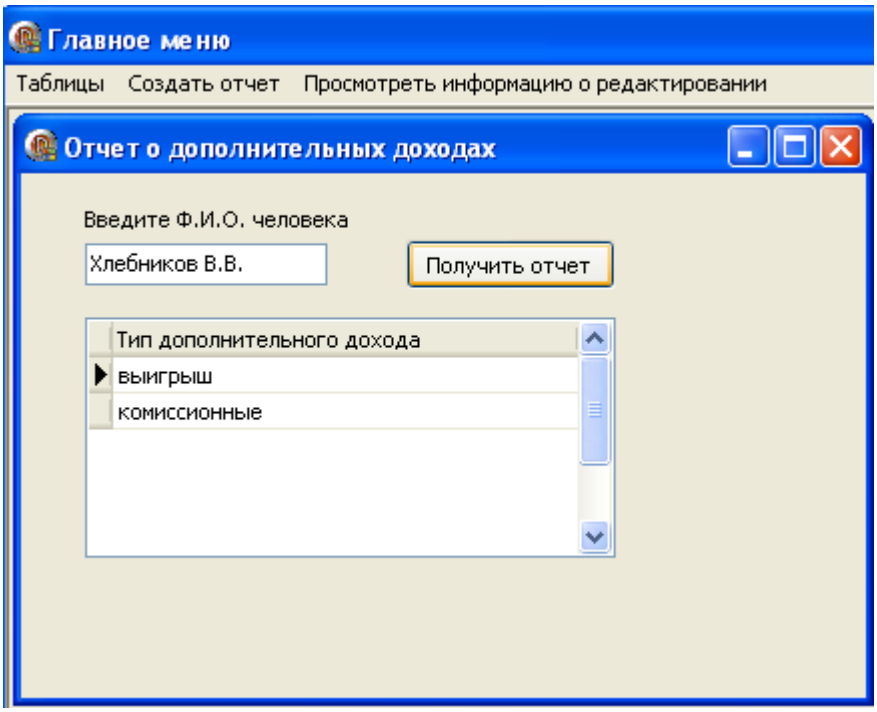

Рис9. Отчет о дополнительных доходах

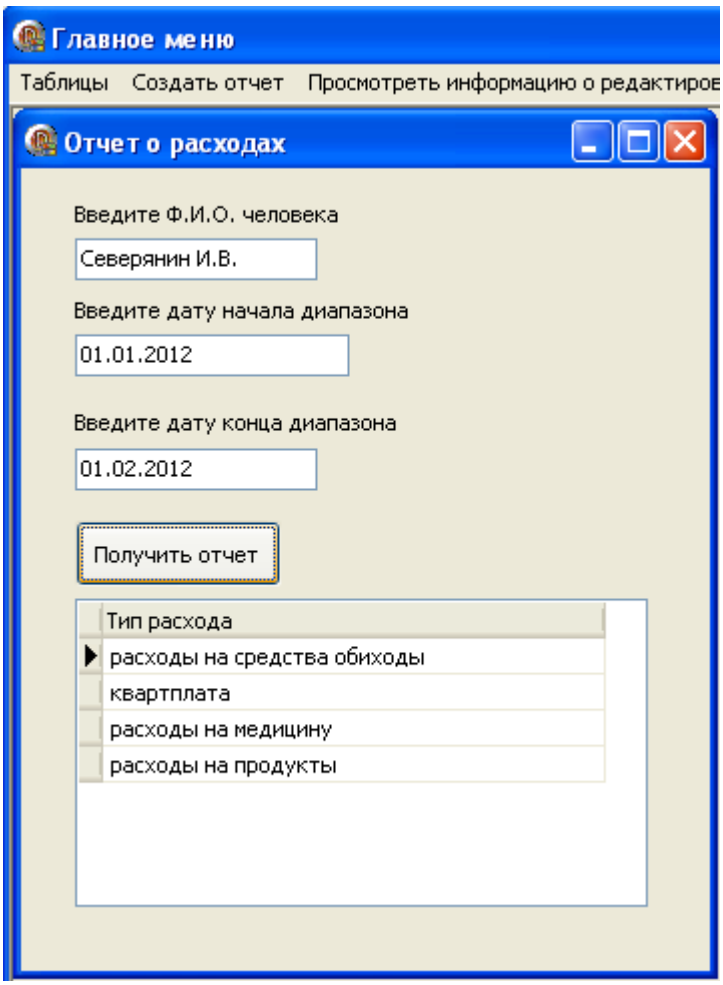

Рис10. Отчет о расходах

Глава 5. Тестирование

Тестирование является одним из наиболее устоявшихся способов обеспечения качества разработки программного обеспечения и входит в набор эффективных средств современной системы обеспечения качества программного продукта. [10]

Тестирование обеспечивает:

- Обнаружение ошибок.
- Демонстрацию соответствия функций программы ее назначению.
- Демонстрацию реализации требований к характеристикам программы.
- Отображение надежности как индикатора качества программы.

Тестирование не может показать отсутствие дефектов (оно может показывать только присутствие дефектов). Важно помнить это утверждение при проведении тестирования.

Методов тестирования много. В данной курсовой работе использовалось тестирование по принципам «белого» и «черного» ящиков.

5.1. Тестирование по принципу «белого ящика»

Известна: внутренняя структура программы.

Исследуются: внутренние элементы программы и связи между ними.

Объектом тестирования является не внешнее, а внутреннее поведение программы. В ходе тестирования была проанализирована управляющей структуры программы, проверена корректность построения всех элементов программы и правильность их взаимодействия друг с другом. [10]

5.2. Тестирование по принципу «черного ящика»

Известны: функции программы.

Исследуется: работа каждой функции на всей области определения.

Основное место приложения тестов «черного ящика» - интерфейс ПО.

Тестирование «черного ящика» обеспечивает поиск следующих категорий

ошибок:

- Некорректных или отсутствующих функций;
- Ошибок интерфейса;
- Ошибок во внешних структурах данных или в доступе к внешней БД;
- Ошибок характеристик (необходимая емкость памяти и т.д.);
- Ошибок инициализации и завершения.

Подобные категории ошибок способами «белого ящика» не выявляются.

В ходе тестирования методом «черного ящика» были выявлены некоторые ошибки, связанные с неверным доступом к данным, которые в последствии были устранены. [10]

Заключение

В данной курсовой работе была разработана ИС «Учет личных финансов». База данных имеет архитектуру клиент/сервер. В процессе проектирования были построены логическая и физическая модели базы данных в среде Visual Paradigm for UML 7.1, use-case модель проектирования системы.

Серверная часть поставленной задачи была разработана в Oracle 10g XE. Клиентское приложение разработано в Delphi7. Были также использованы дополнительные компоненты ADODB, позволяющие подключаться к серверу Oracle и обеспечивающих доступ к базе данных. По завершению написания программного кода данное ПО было протестировано методом «белого и черного ящиков».

С помощью данного приложения можно отказаться от ведения личного или семейного бюджета посредством бумаги и калькулятора, т. к. абсолютно любой человек может использовать разработанное программное обеспечение для своих нужд в домашних условиях.

Список литературы

1. А. В. Леоненков. «Самоучитель UML», «БХВ-Петербург», 2004 г.
2. К. Дж. Дейт «Введение в системы баз данных», 8-е издание, «Вильямс», 2005г.
3. Скотт Урман «Oracle 9i. Программирование на языке PL/SQL», «Лори», 2002г
4. Кузин А.В., Левонисова С.В. «Базы данных», 2-е издание, «Академия», 2008г.
5. С. Д. Кузнецов «Основы баз данных», 2-е издание, «Бином. Лаборатория знаний», 2007г.
6. Елманова Н., Федоров А. «ADO в Delphi», «БХВ-Петербург», 2002 г.
7. Фаронов В.В. «Программирование баз данных в Delphi 7»
8. Галисеев Г.В., «Компоненты в Delphi 7», Вильямс 2004
9. Вендров А.М. Case-технологии. Современные методы и средства проектирования ИС, 1998 год
10. http://www.intuit.ru/
11. http://www.oracle.com
12. http://www.delphi-manual.ru/