ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ»
(Н И У « Б е л Г У »)

ИНСТИТУТ ИНЖЕНЕРНЫХ ТЕХНОЛОГИЙ И ЕСТЕСТВЕННОХ НАУК
КАФЕДРА МАТЕМАТИЧЕСКОГО И ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ИНФОРМАЦИОННЫХ СИСТЕМ

РАЗРАБОТКА ПРОГРАММЫ АНАЛИЗА ПРОДАЖ ДЛЯ ИНТЕРНЕТ МАГАЗИНА AUTODOC.RU

Выпускная квалификационная работа
обучающейся по направлению подготовки
02.03.03 Математическое обеспечение и администрирование информационных систем
очной формы обучения,
группы 07001302
Ковыляевой Марии Олеговны

Научный руководитель:
доцент, к.т.н. Михелев В.М.

БЕЛГОРОД 2017

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Любая компания в наше время нуждается в информационной продукции. В зависимости от потребностей в рамках рабочей области, требуются системы, решающие конкретные задачи. Не исключение и Белгородское представительство интернет-магазина Autodoc.ru, которое заинтересовано в собственном развитии. Сама компания находится в Москве и предоставляет доступ к сайту и клиент-серверному приложению представителям других городов. Однако, не все задачи выполняет данная информационная система. Бухгалтерия и руководители нуждаются в приложении, которое посчитает прибыль с продаж за несколько секунд и наглядным образом отобразит динамику развития компании. Возможность получать результирующий показатель функционирования торговой организации позволяет оценить, насколько эффективна общая деятельность предприятия, сделать определенные выводы и принять новые решения. Эта тема важна, так как в России предприятий малого бизнеса умирает значительно больше, чем появляется новых. А одна из главных их целей – получение максимального размера прибыли при минимальных расходах. Следовательно, такие цифры должны быть в лёгкой доступности для полного контролирования.

Актуальность ВКР заключается в её востребованности в компании Autodoc. Там нет возможности использовать какие-либо другие технологии, поскольку нужно изъять данные из файла, сформированного клиент-серверным приложением по особому принципу. В ВКР разработано узконаправленное приложение, использование которого, нигде, кроме компании Autodoc, будет невозможным. Однако, именно там, оно значительно облегчит работу некоторых отделов.

Основной целью дипломной работы является оптимизация рабочего процесса в магазине Autodoc.

Достижение указанной цели осуществлялось посредством решения следующих основных задач:

1. ознакомиться с данными, которые требуют вычислений;
2. исследовать формулу вычисления прибыли;
3. спроектировать информационное и программное обеспечение;
4. исследовать OLE технологию;
5. разработать приложение;
6. протестировать приложение;
7. внедрить приложение в работу компании Autodoc.

Объектом исследования является программное обеспечение, позволившее перешагнуть предпринимателям на новый уровень ведения бизнеса. А предметом является оптимизация рабочего процесса, внедрением информационных технологий.

В первой главе выпускной квалификационной работы рассматривается состояние информационной системы в организации на данный момент времени, анализ прибыли как наука и постановка задачи. Во второй главе представлены этапы проектирования и разработки программы. В третьей главе описаны результаты тестирования и внедрения программы в работу интернет магазина Autodoc.ru. В приложении можно увидеть листинг полный листинг.

В результате исследования ВКР требуется получить такие результаты:

1. в приложении интерфейс должен быть интуитивно понятным и не создавать сложности в использовании сотрудниками Autodoc;
2. приложение должно решать поставленные задачи;
3. приложение должно работать без ошибок;
4. время на расчёт прибыли должно быть незначительным.

ВКР содержит: 51 лист, 2 таблицы, 22 рисунка и 6 листингов.

ГЛАВА 1. ИНФОРМАЦИОННАЯ ПОДДЕРЖКА ДЕЯТЕЛЬНОСТИ ИНТЕРНЕТ – МАГАЗИНА AUTODOC.RU И ПУТИ ЕЁ СОВЕРШЕНСТВОВАНИЯ

С развитием интернета весь бизнес постепенно начал перемещаться в виртуальное пространство. Теперь каждая уважающая себя компания обязана иметь свой веб-сайт, а количество интернет магазинов давно достигло огромных цифр. С каждым годом количество различных гаджетов, разработок, систем и прочего растет в геометрической прогрессии. Человечество перешло в новый период, в период, где ценятся информация и информационные технологии. И теперь традиционные подходы к бизнесу, производству и образованию уже не дают столь впечатляющих результатов. Для обеспечения наивысших показателей необходимы некоторые нововведения, в частности внедрение информационных технологий. В этой главе рассмотрена информационная поддержка интернет – магазина Autodoc.ru и пути совершенствования.

Чтобы руководитель имел возможность более эффективно использовать информацию, он должен получать ее в меньшем объеме, более концентрированной и соответствующей тем задачам, которые решаются на данном уровне управления. Обычно только незначительная часть факторов имеет существенное значение при принятии решений. Поэтому основную массу данных, которые возникают при функционировании объектов, следует тщательно «фильтровать» и только необходимые данные передавать в подсистемы управления для принятия соответствующих управляющих воздействий. Прошедшую «фильтр» информацию следует агрегировать, т.е. исключить второстепенное, обобщить и укрупнить.

Современные информационные процессы требуют, чтобы предприниматели по-новому взглянули на информационные технологии управления. В современных компаниях информационный менеджмент

опирается на использование компьютеров. Направления деятельности, которые были улучшены благодаря информационным технологиям, — это прогнозирование, планирование и загрузка оборудования, управление запасами, планирование потребности в материалах, деталях и узлах, программное управление оборудованием, компьютерное проектирование и экспертные системы и протокол стандартизации в области автоматизации производства.

В этой главе рассмотрена информационная поддержка интернет – магазина Autodoc.ru и пути совершенствования.

1.1 Современное состояние информационного обеспечения в организации

Интернет – магазин Autodoc.ru входит в тройку лучших по интернет – продажам автозапчастей в России. С 1998 года открылось более 190 магазинов в 115 городах России, количество сотрудников превышает 2 000 человек, около 200 000 000 автозапчастей и более 1 000 000 клиентов. Благодаря тесному сотрудничеству и налаженным оптимальным и надежным схемам взаимодействия, компания может осуществлять быструю поставку нужной клиенту продукции. Эти цифры говорят о том, что компании требуется многофункциональная, быстрая и надёжная информационная система.

На сегодняшний день компания имеет сайт для взаимодействия с клиентами, приложение для ios и Android и клиентскую программу для обеспечения слаженной работы всех отделов во всех магазинах. Зайдя на сайт у клиента есть возможность выбрать город, зарегистрироваться, создав свою учётную запись, посмотреть представленный товар, выбрать, поместить в корзину, оформить заказ и оплатить различными удалёнными способами. Сайт удобен и понятен пользователю. Учитывая огромное количество представленного товара, клиент без труда может найти нужную деталь

конкретно для своего автомобиля, так как поиск производится по марке и win – коду автомобиля. Структура клиент-серверного взаимодействия, по принципу которого работает сайт и клиентская программа, представлена на рисунке 1.1.

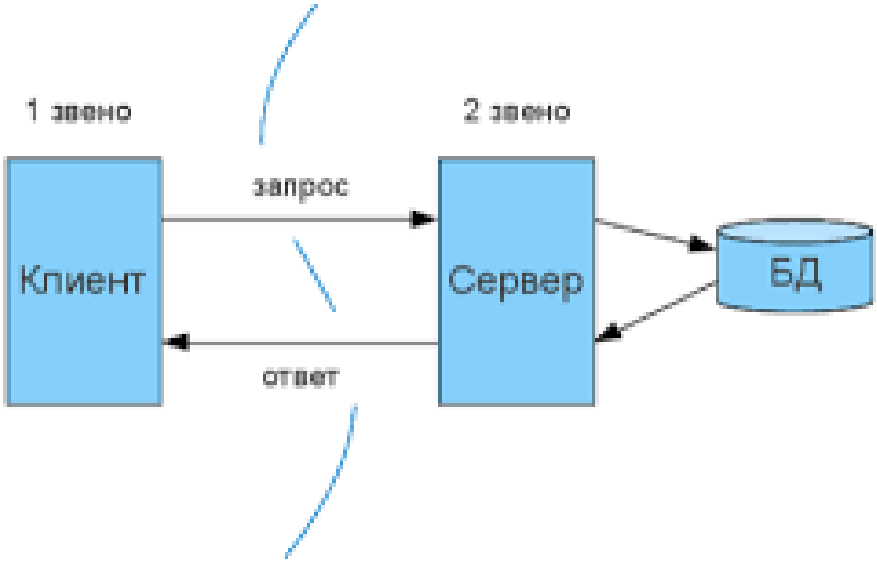

Рис.1.1. Структура клиент-серверного приложения

Когда клиент создаёт учётную запись на сайте, у менеджеров магазина в программе появляется уведомление о новом зарегистрированном пользователе, после чего производится звонок клиенту с целью удостовериться, что тот уверенно может использовать сайт для поиска автозапчастей. В зависимости от выбранного города на сайте отображаются разные цены на товар, управление которыми производится из клиентской программы.

Потребность в информации разных субъектов и звеньев управления неодинакова и определяется прежде всего теми задачами, которые решает в процессе управления тот или иной субъект, тот или иной руководитель, работник управленческого аппарата. Она зависит также от масштаба и важности принимаемых решений (чем масштабнее и важнее решение, тем больше по объему и разнообразней по содержанию информация необходима для его подготовки и принятия), от количества и характера управляемых,

регулируемых параметров, от количества вариантов возможного состояния и поведения управляемого объекта, от величины и разнообразия возмущающих управляемую систему внутренних и внешних воздействий, от количества и качества показателей, характеризующих результаты функционирования данной системы.

Клиентская программа позволяет регулировать всё, что отображается у клиента на сайте и, что остаётся невидимым. Из сотрудников компании программу используют все, а именно отдел менеджеров, закупок, маркетинга, сотрудники склада, кассиры, бухгалтерия и руководство компании. Далее описано всё то, что выполняют сотрудники, используя клиентскую программу.

Менеджеры фиксируют каждого зарегистрированного клиента, составляют карточку покупателя, которая впоследствии хранится в базе данных, управляют заказами интернет – магазина и контролируют их движения. Менеджеры могут отслеживать действие клиента на сайте, что позволяет анализировать предпочтения клиента и сразу предоставлять полезную информацию. Также они оформляют возвраты, формируя заявления.

Отдел закупок договаривается со всеми поставщиками о своевременной доставке товара на склад и фиксирует это в программе. Основываясь на данных, сотрудники распределяют товар по магазинам, а также ведут учёт о наличии товара. Они вносят в программу информацию о времени доставке заказа, которая отображается на сайте для клиента, получают и обрабатывают претензии на поставщиков, и формируют ответ.

Отдел маркетинга разрабатывает и контролирует акции, которые впоследствии вводят на сайт посредством клиентской программы. После чего формирует отчёт о проделанной работе, результатах и изменениях, и предоставляет руководству.

Сотрудники склада фиксируют выдачу товара и формируют накладные. При поступлении товара, считывая штрих – код, вносят в базу информацию о

получении, после чего автоматически клиент получает сообщение о том, что он может прийти и получить свой заказ.

Кассиры проводят клиентам деньги, если они решили оплатить заказ наличными. Эти сотрудники управляют балансом клиента.

Бухгалтерия проводит безналичные платежи, а также производят возврат денежных средств на банковскую карту.

Руководство в программе видит работу каждого менеджера и, исходя из этих данных, анализирует их работу. Это позволяет рассчитать коэффициент эффективности сотрудника. Руководство формирует претензии для отдела закупок, принимает ответы, контролирует балансы клиентов, ведет анализ полученных результатов работы компании и причин возвратов. Программа позволяет устанавливать уровень наценки на каждый сегмент рынка, устанавливать уровень доступа к клиентской программе для каждой должности, проводить инвентаризацию и также анализировать статистику поставок по каждому поставщику.

Всё выше перечисленное является основными функциями, которые выполняют сотрудники компании Autodoc посредством клиентской программы. Это далеко не всё, но основываясь на этих данных можно сказать, что компания имеет стабильную, слаженную информационную систему, но в которой имеются свои минусы. Сотрудники часто жалуются на то, что программа зависает и медленно работает. Вероятнее всего это случается, потому что, как уже выше говорилось, сотрудников во всей сети Autodoc более 2 000, а значит примерно в одно и то же время происходит большое количество запросов к серверу, что естественно затормаживает работу программы. Чтобы усовершенствовать уже существующую программу, требуется разрешение президента компании, который находится в Москве, а это на данном этапе невозможно. Поэтому были проанализированы возможные варианты усовершенствовать информационную систему в Белгородском представительстве Autodoc.

Одной проблемой, стоящей перед руководителями, является уголок на складе, с товаром, который по причине ненадобности вернули в магазин. Эти детали нигде не учитываются, их нельзя вернуть поставщику, так как не все принимают возвраты и их накопилось уже достаточное количество, чтобы задуматься об их реализации. В результате хочется получить информационную систему, где бы вёлся учёт этих товаров, для удобства их продажи, или реализацию подобной функции в существующей клиентской программе.

Усовершенствовать информационное состояние компании можно и внедрением программы, которая автоматически рассчитывала бы зарплату сотрудникам и составляла расчётный лист. Но на сегодняшний день это невозможно, поскольку руководители компании еще определились точно по какому принципу им рассчитывать заработную плату. Сегодня они пробуют различные варианты, и эта идея остаётся нереализованной до тех пор, пока не будет утверждена одна расчётная схема заработной платы.

Также одним из вариантов ускорить работу сотрудникам, можно считать разработку программы, которая могла бы автоматически заполнять карточку клиента и отправлять заказ в работу, если все необходимые требования соблюдены, а именно корректно составленный заказ клиентом. Программа сокращала бы время менеджерам на обработку заказов, что значительно повысило бы эффективность компании, учитывая, что они не успевают из-за большого количества клиентов. А нанимать на работу ещё сотрудников не целесообразно в эпоху Информационных технологий.

Ну и главной проблемой, решение которой и представлено в этой дипломной работе, это расчёт прибыли компании от продаж. Клиентское программа формирует excel – файл со всеми заказами за определенный пользователем период, где указан номер заказа, уровень наценки и сумма (рис. 1.2). Все записи расположены хаотично, а количество их измеряется сотнями. Следовательно, посчитать прибыль компании от продаж и по каждому сегменту занимает большое количество времени.

	A	B	C
79	BETA-13404	Опт9	2172
80	BETA-13416	Опт1	417
81	BETA-13429	Опт2	834
82	BETA-13453	Опт0	1627
83	BETA-13463	Опт0	428
84	BETA-13472	Опт10	2780
85	BETA-13487	Опт10	4420
86	BETA-13502	Опт0	690
87	BETA-13503	Опт9	4537
88	BETA-13516	Опт8	346
89	BETA-13526	Опт10	358
90	BETA-13545	Опт6	2657
91	BETA-13571	Опт6	1415
92	BETA-13572	Опт9	4786,34
93	BETA-13657	Опт0	1216
94	BETA-13682	Опт10	6354
95	BETA-13689	Опт5	1162
96	BETA-13700	Опт0	2942
97	BETA-13708	Опт0	3135
98	BETA-13729	Опт4	8765
99	BETA-13734	Опт9	3071
100	BETA-13740	Опт7	652
101	BETA-13745	Опт9	408
102	BETA-13746	Опт9	417

Рис.1.2. Исходный Excel - файл

На рисунке 1.2 мы видим во втором столбце названия: «Опт 0», «Опт 1», «Опт 2» и т.д. Это всё отдельные группы покупателей, каждая из которых имеют свою наценку на товар. В этом и заключается основная проблема расчёта прибыли.

Руководители основываются на том, что эти цифры должны быть в лёгком доступе, чтобы их всегда можно было оценить и сделать выводы. Реализация и внедрение такой программы рассмотрена в ВКР.

1.2. Анализ прибыли предприятия

Прибыль — это важнейший качественный показатель эффективности деятельности организации, характеризующий рациональность использования средств производства, материальных, трудовых и финансовых ресурсов.

Прибыль представляет собой финансовый результаты хозяйственной деятельности предприятия.

Основными задачами анализа прибыли являются:

- проверка обоснованности плановой величины прибыли (план по прибыли должен быть увязан с объемом реализуемой продукции и ее себестоимостью);
- оценка выполнения бизнес-плана по прибыли;
- исчисление влияния отдельных факторов на отклонение фактической суммы прибыли от плановой;
- выявление резервов дальнейшего роста прибыли и путей мобилизации (использования) этих резервов.

Важнейшими источниками информации для проведения анализа прибыли являются:

- бухгалтерский баланс;
- отчет о прибылях и убытках;
- учетный регистр;
- финансовый план организации.

Прибыль организации складывается из трех основных элементов:

- прибыль от реализации продукции, работ и услуг;
- прибыль от прочей реализации;
- операционные, внереализационные и чрезвычайные доходы и расходы.

Основную часть получаемой прибыли составляет прибыль от реализации продукции, работ, услуг.

Целенаправленная и эффективная работа по экономии всех видов ресурсов возможна только при умело налаженном учете и анализе, что подтверждается всей практикой их ведения. Бесхозяйственность и расточительность во многих случаях являются следствием слабого учета, контроля и анализа. Недостатки в учете порождают потери еще потому, что при их наличии зачастую снимается ответственность за рациональное

использование ресурсов и исключается возможность эффективного контроля за их использование. Получаемая на основе достоверного учета объективная информация позволяет хозяйственным руководителям вовремя установить участки и причины потерь ресурсов, определить наиболее результативные пути их экономии.

Прибыль от реализации продукции — это финансовый результат, полученный от основной деятельности предприятия, которая может осуществляться в любых видах, зафиксированных в его уставе и не запрещенных законом. Финансовый результат определяется раздельно по каждому виду деятельности предприятия, относящемуся к реализации продукции, выполнению работ, оказанию услуг. Он равен разнице между выручкой от реализации продукции в действующих ценах и затратами на ее производство и реализацию (1.1).

$$Пр = Вр — С/с \text{ , где} \quad (1.1)$$

- Вр — выручка от реализации;
- С/с — себестоимость (затраты на производство и реализацию).

Выручка принимается в расчет без налога на добавленную стоимость и акцизов, которые, являясь косвенными налогами, поступают в бюджет. Из выручки также исключается сумма наценок (скидок), поступающая торговым и снабженческо-сбытовым предприятиям, участвующим в сбыте продукции.

Данными для расчётов, которыми мы владеем, исходя из рисунка 1.2 являются значение выручки от реализации и процент наценки. Нам следовало преобразовать формулу, для расчёта исходя из исходных данных. В ВКР расчёт прибыли осуществлялся по формуле 1.2.

$$Пр = Вр - \frac{Вр}{1+\frac{Нац}{100}} \text{ , где} \quad (1.2)$$

- Нац – кол-во процентов наценки.

Эта формула была использована в написании алгоритма программы для расчёта прибыли.

1.3 Постановка задачи

Настоящее техническое задание распространяется на разработку приложения для автоматизации процесса подсчёта прибыли компании от продаж. Также система должна использоваться для хранения и анализа данных.

На сегодняшний день существует довольно большой круг систем учета и управления, которые обладают достаточно широким функциональными возможностями. Однако они являются коммерческими и стоят достаточно дорого, а также они не учитывают специфику конкретной предметной области, в пределах которой многие из функций остаются невостребованными. А в следствии этого могут использовать лишние вычислительные ресурсы.

Создание приложения, в рамках которого были бы реализованы наиболее востребованные функции, позволит автоматизировать управление состава вычислительной техники.

Создание автоматизированной системы позволит обеспечить:

- повышение оперативности в работе за счет внедрения прогрессивных технологий, современного программного и технического обеспечения;
- повышение оперативности получения отчетной информации для анализа;
- повышение информативности системы для принятия конструктивных управленческих решений руководством компании.

Опишем техническое задание.

1. Основание для разработки. Система разрабатывается на основании приказа директора института на выпускную квалификационную работу по направлению бакалавра.

2. Назначение. Программное средство предназначено для облегчения работы пользователей системы, связанной с расчётами над прибылью магазина, а также получения наглядного отображения о динамике развития компании. Пользователями приложения являются руководители компании и отдел бухгалтерии.

3. Требования к программе и её функциональным характеристикам. Система должна представлять совокупность методических и программных средств решения следующих задач:

- произведение расчётов над выручкой компании от продаж для получения прибыли;
- ввод новой информации в БД;
- просмотр информации в БД;
- удаление информации из БД.

ГЛАВА 2. РАЗРАБОТКА ПРОГРАММЫ

2.1 Проектирование информационного и программного обеспечения

На начальном шаге проектирования информационной системы нужно выполнить анализ предметной области, т.е. выделить объекты предметной области и их связи между собой. Во время выбора структуры и состава предметной области возможны несколько подходов: предметный и функциональный [7].

Функциональный подход реализует принцип движения «от условий задачи» и используется, когда определен комплекс задач, для обслуживания которых создается информационная система. В этом случае можно определить минимальный набор объектов предметной области, которые необходимо описать.

В предметном подходе объекты предметной области определяются с таким расчетом, чтобы их можно было использовать при реализации множества разнообразных задач.

Чаще всего используют комбинацию этих двух подходов, которая, с одной стороны, ориентирована на функциональные потребности пользователей или конкретные задачи, а с другой стороны, учитывает возможность наращивания новых приложений.

Требуется разработать программу для расчёта прибыли компании от продаж. Система, во-первых, должна предусматривать ведение журнала. Запись в журнале должна содержать в себе:

1. номер записи в журнале;
2. вся выручка;
3. вся прибыль;
4. комментарий;

5. дата записи в журнал;
6. выручка по каждому сегменту;
7. прибыль по каждому сегменту;
8. группа наценки.

Выручка и прибыль по каждому сегменту должна содержать в себе соответствующие значения для каждой оптовой группы и одной розничной:

1. Opt0;
2. Opt1;
3. Opt2;
4. Opt3;
5. Opt4;
6. Opt5;
7. Opt6;
8. Opt7;
9. Opt8;
10. Opt9;
11. Opt10;
12. розница.

Приложение должно подразумевать в себе, что наценка на товар у каждого сегмента может меняться с течением времени, следовательно, необходима возможность выбора группы наценок, удаления и добавления. Помимо всех субъектов информация о наценки должна включать в себя дату добавления новой группы [4].

Приведенный выше анализ объектов и атрибутов позволяет выделить сущности проектируемой базы дынных и, приняв решение о создании реляционной базы данных, построить ее логическую модель на языке «сущность – связь».

Опишем сущности.

1. Журнал анализа прибыли (№ записи в журнале, вся выручка, вся прибыль, комментарий, дата добавления, № наценки). Эта сущность

предназначена для хранения результатов расчётов и ведения отчётного журнала.

2. Выручка (№ записи «Выручка», выручка Opt0, выручка Opt1, выручка Opt2, выручка Opt3, выручка Opt4, выручка Opt5, выручка Opt6, выручка Opt7, выручка Opt8, выручка Opt9, выручка Opt10, выручка розницы, № записи в журнале). Здесь ведётся учёт выручки по каждому сегменту.

3. Прибыль (№ записи «Прибыль», прибыль Opt0, прибыль Opt1, прибыль Opt2, прибыль Opt3, прибыль Opt4, прибыль Opt5, прибыль Opt6, прибыль Opt7, прибыль Opt8, прибыль Opt9, прибыль Opt10, прибыль розницы, № записи в журнале). Производится учёт прибыли по каждому сегменту.

4. Наценка (№ наценки, кол-во процентов Opt0, кол-во процентов Opt1, кол-во процентов Opt2, кол-во процентов Opt3, кол-во процентов Opt4, кол-во процентов Opt5, кол-во процентов Opt6, кол-во процентов Opt7, кол-во процентов Opt8, кол-во процентов Opt9, кол-во процентов Opt10, кол-во процентов розницы, дата добавления).

Для проектирования и документирования базы данных было выбрано средство AllFusion ERwin Data Modeler (ранее ERwin). Оно помогает визуализировать структуру данных, обеспечивая эффективный процесс разработки программы. Были построены две модели базы данных: логическая и физическая, они представлены на рисунках 2.1 и 2.2.

Были определены не идентифицирующие связи для таблиц базы данных, потому что экземпляр любой дочерней сущности не идентифицируется через связь с родительской и атрибуты, составляющие первичный ключ родительской сущности, входят в состав не ключевых атрибутов дочерней сущности. Таблица «Журнал анализа прибыли» является дочерней для таблицы «Наценка» и имеет связь один-ко-многим, что определяется внешним ключом. А для таблиц «Выручка» и «Прибыль»

сущность «Журнал» является родительской. Существование записей в этих таблицах невозможно без связи с номером записи в журнале.

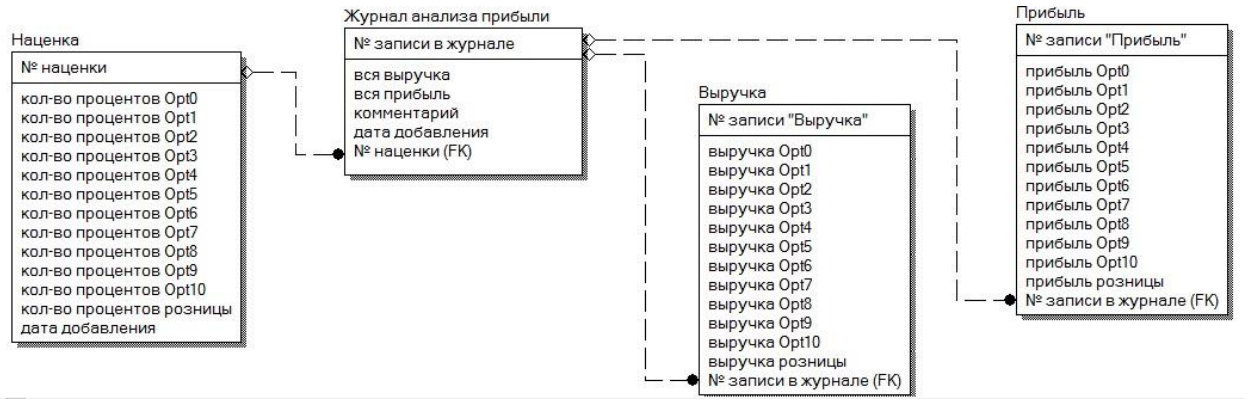

Рис.2.1. Логическая модель базы данных

Физическая структура базы данных, а также сама заполняемая данными база данных являются отображением реальной предметной области (рис.2.2). Спроектировать физическую структуру базы данных означает определить все информационные единицы базы данных и связи между ними, задать их имена, типы и другие требуемые характеристики [6].

Все атрибуты, в которых будет храниться денежное значение, имеют тип numeric(9.2). Количество процентов наценки на каждой группе имеет тип integer. Такие атрибуты как «дата добавления наценки» и «дата добавления записи в журнал» имеют тип date. Комментарий в журнале можно будет оставить любыми символами до 50 знаков.

Для дальнейшей работы следует определить СУБД. Выбор системы управления баз данных (СУБД) представляет собой сложную многопараметрическую задачу и является одним из важных этапов при разработке приложений баз данных. Выбранный программный продукт должен удовлетворять как текущим, так и будущим потребностям предприятия.

Мы использовали СУБД Firebird. Это полнофункциональная и мощная СУБД, она может обслуживать базы данных размером от нескольких

килобайт до многих гигабайт, показывая хорошую производительность и практически не нуждаясь в обслуживании.

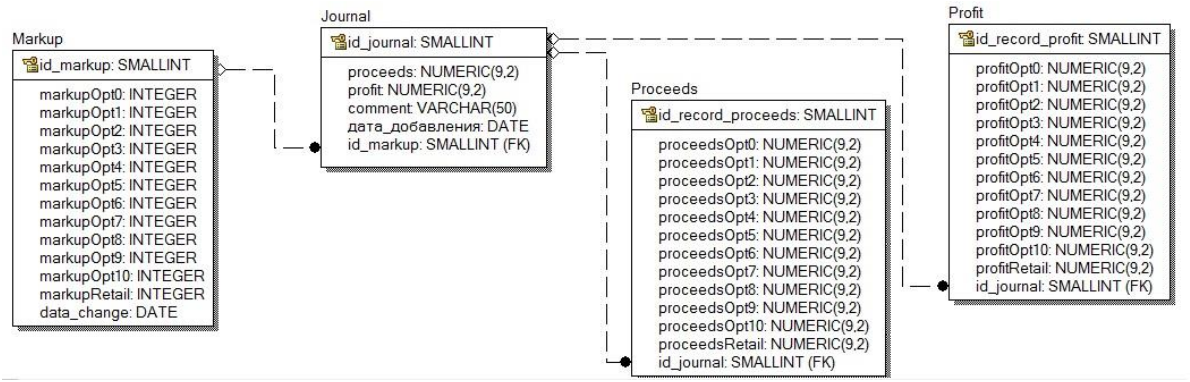

Рис.2.2. Физическая модель базы данных

Основные положительные характеристики Firebird:

- полная поддержка хранимых процедур и триггеров;
- транзакции, полностью совместимые с концепцией ACID;
- ссылочная целостность;
- очень небольшой размер;
- Firebird практически не требует работы системного администратора или позволяет свести её к минимуму;
- почти не требует настройки – использовать СУБД можно сразу же после её установки;
- огромное интернет-сообщество пользователей и разработчиков, где можно получить быструю и бесплатную помощь;
- безопасная запись данных – быстрое восстановление после сбоев;
- большое количество средств доступа к БД;
- поддержка большинства распространенных операционных систем;
- и др.

Firebird включает в себя набор консольных программ, позволяющих создавать базы данных, исследовать их характеристики, выполнить

операторы SQL и скрипты, производить резервное копирование данных, их восстановление из резервной копии и так далее.

Из инструментов, первым будет служить IBExpert. IBExpert — GUI-оболочка, предназначенная для разработки и администрирования баз данных InterBase и Firebird, а также для выбора и изменения данных, хранящихся в базах. Он обладает множеством облегчающих работу компонентов: визуальный редактор для всех объектов базы данных, редактор SQL и исполнитель скриптов, отладчик для хранимых процедур и триггеров, построитель области, инструмент для импорта данных из различных источников, собственный скриптовый язык, а также дизайнер баз данных и т. д.

Для того, чтобы программа удовлетворяла требованиям, она должна выполнять ряд задач. Пользователь должен иметь возможность выбрать в программе файл, в котором хранятся значения для вычислений, выбрать группу наценки, получить результаты расчётов. Если он посчитает нужным сохранить эти значения, должна быть кнопка внесения расчётов в базу данных, где он указывает период выручки магазина от продаж и может оставить комментарий. В приложении должна быть возможность просмотра всего журнала, выручки и прибыли по каждому сегменту, а также удаление любых записей.

Разработать программу расчёта прибыли нам поможет C++ Builder. Это программный продукт, инструмент быстрой разработки приложения, интегрированная среда программирования, система, используемая программистами для разработки программного обеспечения на языке программирования C++. C++ Builder может быть использован везде, где требуется дополнить существующие приложения расширенным стандартом языка C++, повысить быстродействие и придать пользовательскому интерфейсу качества профессионального уровня.

Создание таблиц в IBExpert было произведено посредством SQL – запросов (Листинг 2.1). Для того, чтобы программа выполняла все поставленные задачи требуется создать таблицы:

1. Journal;
2. Marcup;
3. Proceeds;
4. Profit.

Формируя запрос серверу, мы обязательно должны указать название таблицы, название всех полей и их типы данных. Названия не должны повторяться и вводить в заблуждение при разработке программы. Должно быть примерно понятно, какую информацию будет содержать поле. В противном случает можно запутаться, а также, в будущем, никто не разберется в таблицах и полях этой БД.

Листинг 2.1. Запрос на создание таблицы «Journal»

```
CREATE TABLE JOURNAL (
   ID_JOURNAL   SMALLINT NOT NULL,
   PROCEEDS     NUMERIC(9,2) NOT NULL,
   PROFIT       NUMERIC(9,2) NOT NULL,
   COMENT       VARCHAR(50),
   DATE_INSERT  DATE NOT NULL,
   ID_MARKUP    SMALLINT NOT NULL
);
```

Конец листинга.

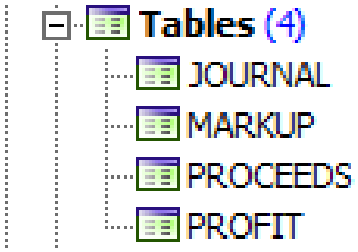

Рис.2.3. Созданные таблицы

Для обеспечения целостности БД при создании внешних ключей была использована функция «ON DELETE CASCADE», которая при удалении записи в родительской сущности автоматически удаляет запись в дочерней. SQL – запрос представлен в листинге 2.2.

Листинг 2.2. Создание внешнего ключа для таблицы «Journal»

```
ALTER TABLE JOURNAL ADD CONSTRAINT FK_MARKUP
FOREIGN KEY (ID_MARKUP) REFERENCES MARKUP
(ID_MARKUP) ON DELETE CASCADE;
```

Конец листинга.

Для того, чтобы приложение удовлетворяло требованиям, оно должно выполнять ряд задач. Пользователь должен иметь возможность выбрать группу наценки для расчётов, которая хранится в базе данных. Т.е. данные из таблицы «Marcup» должны отображаться в программе. Также он должен иметь возможность записать расчёты в журнал, а именно в таблицу «Journal», после чего посмотреть все записи и по желанию удалить. По одному клику пользователь должен получать результаты расчётов по каждому субъекту, это значит отображаться должны данные из таблиц «Proceeds» и «Profit». Добавление и удаление групп наценок в БД тоже остаётся за пользователем. Для выполнения всех эти функций со стороны сервера, были разработаны представления и хранимые процедуры.

#	PK	FK	UNQ	Field Name	Field Type	Domain	Size	Scale	Subtype	Array	Not Null
1	PK1			ID_JOURNAL	SMALLINT						☒
2				PROCEEDS	NUMERIC		9	2			☒
3				PROFIT	NUMERIC		9	2			☒
4				COMENT	VARCHAR		50				☐
5				DATE_INSERT	DATE						☒
6		FK		ID_MARKUP	SMALLINT						☒

Рис. 2.4. Таблица «Journal»

Представление – это виртуальная таблица, представляющая собой поименованный запрос, который будет подставлен как подзапрос при использовании представления. Использование представлений не даёт каких-то совершенно новых возможностей в работе с БД, но может быть очень удобно. Опишем основные плюсы использования представлений.

- Представления скрывают от прикладной программы сложность запросов и саму структуру таблиц БД. Когда прикладной программе требуется таблица с определённым набором данных, она делает простейший запрос из подготовленного представления. При этом даже если для получения этих данных требуется чрезвычайно сложный запрос, сама программа этого запроса не содержит.
- Использование представлений позволяет отделить прикладную схему представления данных от схемы хранения. С точки зрения прикладной программы структура данных соответствует тем представлениям, из которых программа эти данные извлекает. В действительности данные могут храниться совершенно иным образом, достаточно лишь создать представления, отвечающие потребностям программы. Разделение позволяет независимо модифицировать прикладную программу и схему хранения данных: как при изменении структуры физических таблиц, так и при изменении программы достаточно изменить представления соответствующим образом. Изменение программы не затрагивает физические таблицы, а изменение физической структуры таблиц не требует корректировки программы.
- С помощью представлений обеспечивается ещё один уровень защиты данных. Пользователю могут предоставляться права только на представление, благодаря чему он не будет иметь доступа к данным, находящимся в тех же таблицах, но не предназначенных для него.
- Поскольку SQL-запрос, выбирающий данные представления, зафиксирован на момент его создания, СУБД получает возможность применить к этому запросу оптимизацию или предварительную компиляцию,

что положительно сказывается на скорости обращения к представлению, по сравнению с прямым выполнением того же запроса из прикладной программы.

Были созданы следующие представления:

1. VIEW_JOURNAL отображает содержимое таблицы, за исключением поля ID_MARCUP, которое является полем для сохранения целостности, а именно поле с внешним ключом;
2. VIEW_MARKUP отображает содержимое таблицы, за исключением ключевого поля ID_MARKUP, значение которого не требуется пользователю для работы в программе.

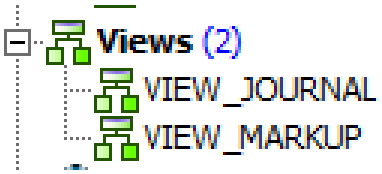

Рис. 2.5. Представления

В SQL – запросе на создание представления было прописано название полей на русском языке, отображение которых значительно улучшит и сделает понятнее пользователю интерфейс программы. SQL – запрос на создание одного представления отображён в листинге 2.3.

Листинг 2.3. Создание представления «VIEW_MARKUP»

```
CREATE VIEW VIEW_MARKUP( "№", "Опт0", "Опт1", "Опт2", "Опт3", "Опт4", "Опт5", "Опт6", "Опт7", "Опт8", "Опт9", "Опт10", "розница", "дата")
AS select id_markup, markupopt0, markupopt1, markupopt2, markupopt3, markupopt4, markupopt5, markupopt6, markupopt7, markupopt8, markupopt9, markupopt10, markupretail, data_change
from markup;
```

Конец листинга.

Хранимая процедура — объект базы данных, представляющий собой набор SQL-инструкций, который компилируется один раз и хранится на сервере. Хранимые процедуры очень похожи на обыкновенные процедуры языков высокого уровня, у них могут быть входные и выходные параметры и локальные переменные, в них могут производиться числовые вычисления и операции над символьными данными, результаты которых могут присваиваться переменным и параметрам. В хранимых процедурах могут выполняться стандартные операции с базами данных. Кроме того, в хранимых процедурах возможны циклы и ветвления, то есть в них могут использоваться инструкции управления процессом исполнения.

Были созданы следующие хранимые процедуры:

1. PROCEEDS_VIEW представляет значение выручки от каждого сегмента по номеру записи в журнале;
2. PROFIT_VIEW представляет значение прибыли от каждого сегмента по номеру записи в журнале;
3. INSERT_JOURNAL позволяет добавить в журнал новую запись с расчётами, комментарием и датой;
4. INSERT_PROCEEDS позволяет добавить в таблицу «Proceeds» расчёты выручки по каждому сегменту;
5. INSERT_PROFIT позволяет добавить в таблицу «Profit» расчёты прибыли по каждому сегменту;
6. INSERT_MARKUP позволяет пользователю добавить новую группу наценок для расчёта прибыли;
7. DEL_MARCUP совершает удаление из таблицы наценок по полю «id_markup»;
8. DEL_JOURNAL совершает удаление из журнала по полю «id_journal»;
9. SAVE_ID_JOURNAL хранимая процедура, которая играет немаловажную роль в целостности базы данных. Она подтягивает значение поля «id_journal» последней записи в таблице.

10. NEW_PROCEDURE предоставляет все значения наценки по идентификационному номеру

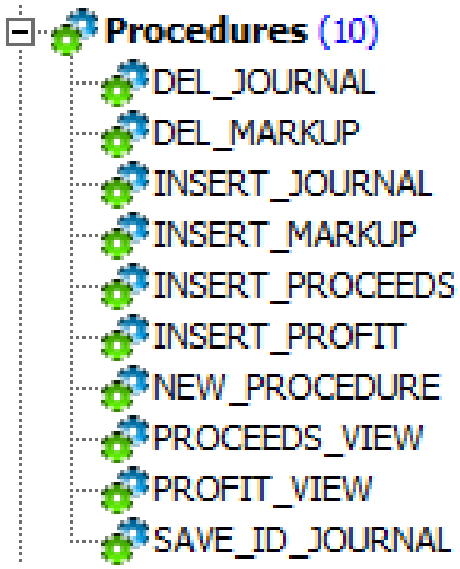

Рис.2.6. Хранимые процедуры

В разработке программы анализа продаж хранимые процедуры будут играть одну из самых важных ролей. Благодаря им пользователь будет добавлять и удалять записи из базы данных. А также у него будет представлена возможность детально рассмотреть все записи в БД, выбрать из имеющихся и использовать для другой хранимой процедурой. В итоге большую часть операций с БД будет происходить благодаря хранимым процедурам.

На этом проектирование и разработка базы данных закончена. Следующими этапами являются изучение и разработка алгоритмов OLE – технологии и разработка программы, подразумевающая под собой интерфейс, подключение БД к программе, реализацию вывода таблиц и представления на формы, выполнение хранимых процедур, написание алгоритмов расчёта прибыли.

2.2. OLE – технология

Технология OLE (Object Linking and Embedding с англ. - связывание и внедрение объектов) — технология управления и обмена информацией между программным интерфейсом других приложений.

OLE позволяет создавать объекты (рисунки, чертежи и текст) в одном приложении, а затем отображать эти объекты в других приложениях. Объекты, помещенные в приложение, использующее OLE, называются OLE-объектами. Для того, чтобы технология OLE действовала, приложение, используемое для создания OLE-объекта, и приложение, в которое помещается OLE-объект, должны поддерживать режим OLE.

При использовании OLE в обмене информацией участвуют два приложения - приложение-сервер и приложение-клиент (рис.2.7). Приложение-сервер используется для создания и редактирования OLE-объектов (рисунков, чертежей, текстов). После того как объект создан, он помещается в приложение-клиент. Некоторые приложения могут действовать и как серверные, и как клиентские, другие такой способностью не обладают.

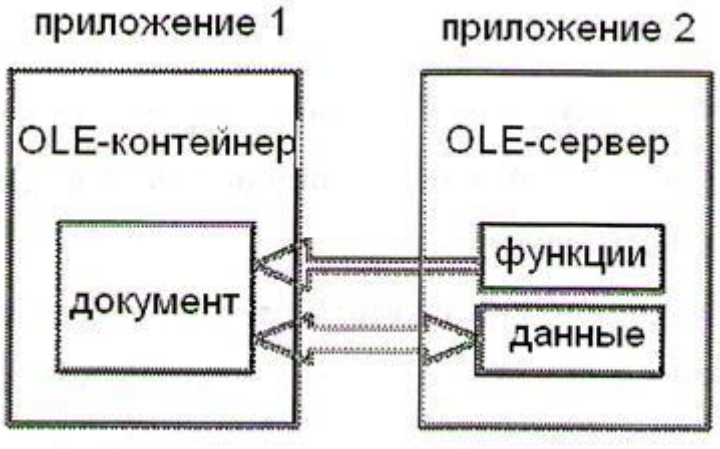

Рис.2.7. Схема взаимодействия приложений через OLE – технологию

OLE-объекты могут связываться с приложениями клиента или внедряться в них. OLE-связанный объект подключается к отдельному файлу. Управление появлением OLE-объекта в приложении-клиенте осуществляется

на основе информации, хранящейся во внешнем файле. Когда этот внешний файл изменяется в серверном приложении, OLE-объект соответствующим образом обновляется. Внедренный OLE-объект полностью содержится в файле приложения-клиента, поэтому он не связан с внешним файлом.

Буфер обмена представляет собой временную область памяти, используемую для хранения информации. Реализована возможность копирования в буфер обмена элемент или его часть из приложения-сервера, а затем размещения его в приложение-клиент. Этот элемент становится OLE-объектом. При простом копировании и вставке информации этот элемент становится OLE-внедренным объектом. При создании OLE-связанного объекта с помощью буфера обмена используется команда "Специальная вставка", но при использовании буфера обмена вставляемый элемент не всегда становится OLE-объектом.

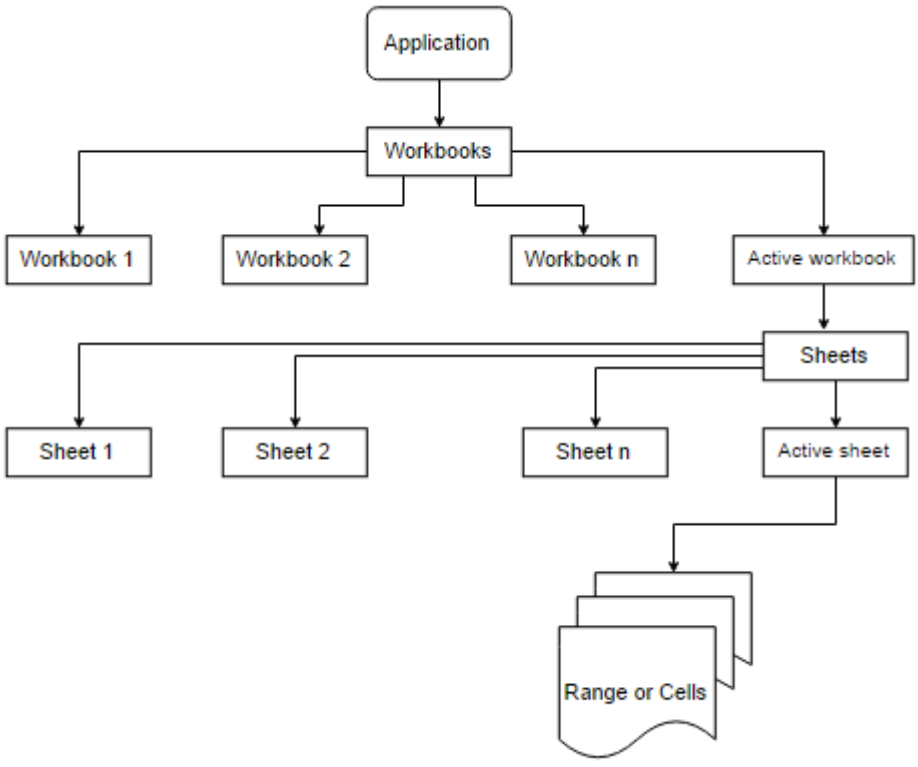

Рис. 2.8. Объектная модель MS Excel

Одной из задач ВКР является разработка программы анализа продаж для компании Autodoc. Как уже говорилось в первой главе, данные для

расчётов формирует существующее клиентское приложение в виде Excel файла. Для того, чтобы программа могла считать эти данные и записать в переменные была изучена и использована OLE - технология.

Объектная модель MS Excel представляет собой иерархию объектов, подчинённых одному объекту Application, который соответствует самому приложению MS Excel. На рисунке 2.8 представлены основные компоненты объектной модели MS Excel.

Для разработки решений, использующих Microsoft Office Excel, необходимо взаимодействие с объектами, предоставляемыми объектной моделью Excel. Для выполнения большинства операций в MS Excel применяются объекты:

- Excel.Application (приложение) - объект, представляющий приложение Microsoft Excel;
- Workbook (рабочая книга) - представляет рабочую книгу - аналог документа Microsoft Word. Однако, в Word мы работаем с данными, расположенными в документе, а в Excel на пути к данным есть еще один объект - рабочий лист;
- Worksheet (рабочий лист) - книга в MS Excel разбита на рабочие листы. Именно на листе расположены ячейки, которые могут хранить информацию и формулы;
- Cells (ячейка) – представляет собой одну ячейку. Используя её работа с листом становится очень четкой и удобной - чтобы работать с какой-либо ячейкой, надо знать лишь ее имя (в формате A1) или адрес (R1C1).

Для использования OLE-технологии, к проекту была подключена библиотека «utilcls.h».

Требуемые нам для расчётов ячейки в Excel файле находятся в столбиках «B» и «C». По первому столбику мы определяем к какому субъекту относится число во втором столбике (рис. 2.9.).

B	C
Отч9	2172
Отч1	417
Отч2	834
Отч0	1627
Отч0	428

Рис. 2.9. Столбцы Excel-файла

С помощью этой технологии было реализовано считывание данных из excel файла:

1. запуск Excel из программы следующей функцией: CreateOleObject("EXCEL.Application");
2. открытие книги: OlePropertyGet("Workbooks").OlePropertyGet("Open",a.c_str());
3. реализация цикла записи значения из ячеек в вектор (Листинг2.3);
4. после завершения цикла, мы закрываем Excel: OleProcedure("Quit");

Листинг 2.4. Цикл считывания данных из Excel-файла

```
for(int j=0; ;j++){
int l=j+2;
h=cnn.OlePropertyGet("Cells",l,3);
A.push_back(j);
float value=0;
if(TryStrToFloat(h, value)) A[j] = value;
else {
break;
}
```

Конец листинга.

2.3. Разработка программы

В C++ Builder для организации доступа к данным используется компонент DataModule – невидимая форма, на которой располагаются компоненты для взаимодействия с базой данных [1]. Основными компонентами являются IBDatabase, IBTransaction и DataSource. IBDatabase обеспечивает соединение с базой данных. Один компонент может быть связан только с одной базой одновременно. IBTransaction – компонент для явной работы с транзакциями. Без него невозможно выполнить какие-либо действия с БД в Firebird. И DataSource обеспечивает синхронность работы таблиц с визуальными компонентами.

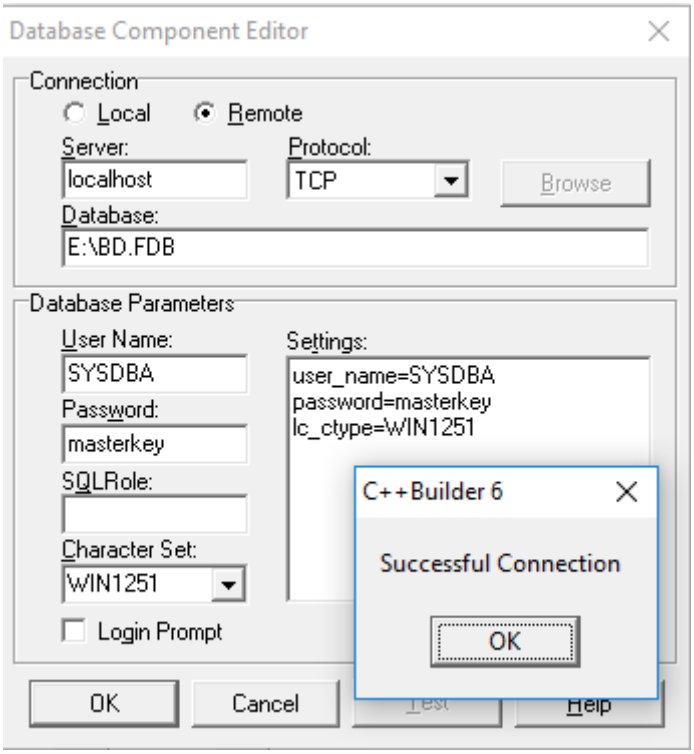

Рис.2.10. Подключение к БД

Поместив на DataModule компонент IBDatabase, осуществили подключение к базе данных. Указали имя сервера, путь к базе данных и параметры соединения: имя пользователя, пароль и кодировку (рис.2.10).

Поместив на DataModule компонент IBTransaction, указали в его свойстве DefaultTransaction подключенную базу данных – IBDatabase. На данном этапе установлено соединение и можно производить транзакции.

Для вычисления прибыли компании от продаж нам нужно было осуществить возможность пользователю выбрать Excel файл для расчётов, вывести на форму результат представления VIEW_MARKUP, для выбора группы наценки и реализовать алгоритм расчёта. Мы поместили на форму компонент для открытия файлов OpenDialog и кнопку, при нажатии на которую пользователю открывается возможность выбрать файл, после чего адрес файла сохраняется в переменную строкового типа.

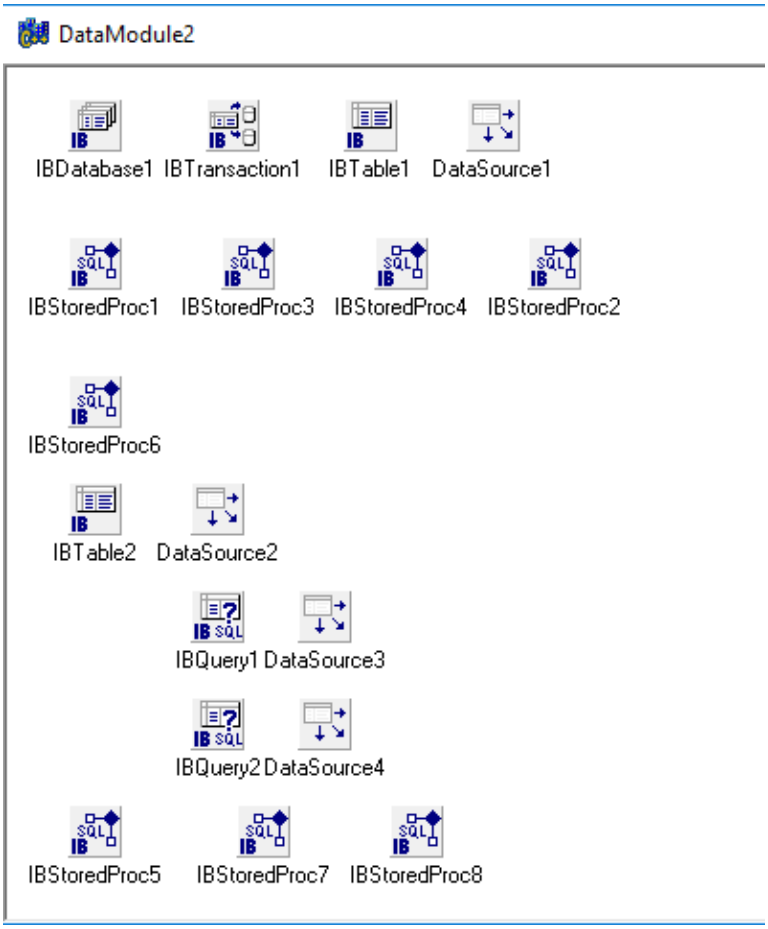

Рис.2.11. Все компоненты на DataModule

Следующий компонент, который мы поместили на DataModule был IBTable. IBTable используется для доступа к таблицам БД. Компоненту мы указали базу данных и в свойстве TableName выбрали представление

VIEW_MARKUP. Для отображения групп наценок мы поместили на форму компонент DBGrid, который обеспечивает вывод данных из компонентов IBTable или IBQuery табличным способом. Следом на DataModule поместили DataSource, которому указали в свойстве DataSet компонент IBTable. В свойсве DataSource прописали: «DataModule2->DataSource1». На этом этапе на форме отобразились результаты представления VIEW_MARKUP, а именно группы наценок.

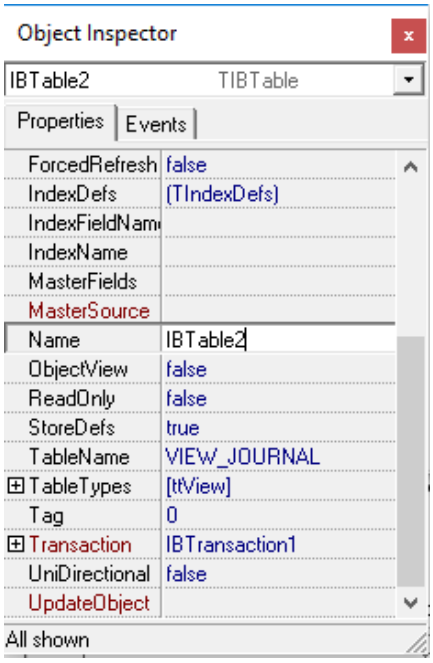

Рис.2.12. Настройка компонента IBTable

После того как пользователь выбрал Excel – файл, выбрал группу наценки, кликнув по ней мышью, он должен иметь возможность увидеть расчёты. Мы поместили на форму кнопку «Расчитать» и прописали функцию, для считывания значений из таблицы MARKUP, которая отображена в листинге 2.5.

После чего в компонентах Label отображаются расчёт выручки и прибыли по каждому субъекту, а именно переменные prof0, prof1, prof2, prof3 и т.д и proc0, proc1, proc2, proc3 и т.д.

Листинг 2.5. Функция расчёта прибыли для одного субъекта

```
proc0=0;
```

```
a=Edit1->Text;
m= StrToInt(Form1->DBGrid1->SelectedField->Text);
DataModule2->IBStoredProc1->ParamByName("ID_MARKUP_NEW")-
>AsInteger=m;
DataModule2->IBStoredProc1->Prepare();
DataModule2->IBStoredProc1->ExecProc();
m0 = DataModule2->IBStoredProc1->ParamByName("markupopt0_new")-
>AsInteger;
prof0=proc0-(proc0/(1+(m0/100)));
```

Конец листинга.

Для внесения расчётов в БД были подключены четыре хранимые процедуры, для трёх разных таблиц. Это INSERT_ JOURNAL, INSERT_PROCEEDS, INSERT_PROFIT и SAVE_ID_JOURNAL. В первую очередь заносится запись в таблицу Journal, после чего через хранимую процедуру мы получаем идентификационный номер этой записи и во время занесения значений в две другие таблицы передаём значение идентификатора, тем самым связывая записи внешним ключом и сегодняшнюю дату.

Рядом с кнопкой «Внести расчёты в журнал» мы поместили кнопку «Сбросить», которая очищает все поля на форме (Листинг 2.6).

Листинг 2.6. Обработка кнопки «Сбросить»

```
Edit1->Text="";
Edit2->Text="";
Label1->Caption="";
Label2->Caption="";
Button1->Enabled=false;
Button3->Enabled=false;
```

Конец листинга.

На следующей форме было использовано три DBGrida для отображения всех записей в журнале, выручки и прибыли по каждому сегменту. DBGrid с помощью IBTable и DataSource подключили к представлению VIEW_JOURNAL и получили отображение на форме записи из таблицы JOURNAL. Поставили две кнопки, первая вызывает функцию для отображения выручки и прибыли по каждому сегменту из выбранной таблицы с помощью хранимой процедуры VIEW_PROCEEDS и VIEW_PROFIT, а вторая вызывает функцию удаления через хранимую процедуру DEL_JOURNAL.

На третьей форме разместили DDGrid для отображения всех групп наценки (рис.2.13). Его подключили к представлению через те же компоненты, что использовались на первой форме. Для добавления новой группы наценки кнопке «Добавить группу» прописали функцию подключения к хранимой процедуре DEL_MARKUP. Ей передаются параметры из двенадцати Edit-ов, а также текущая дата.

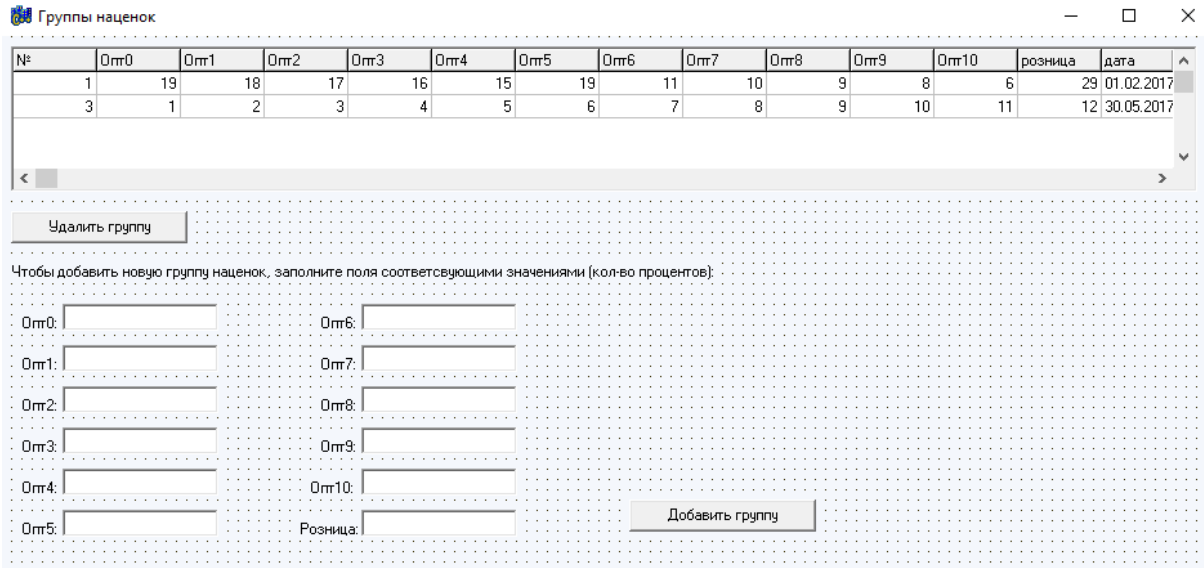

Рис.2.13. Третья форма «Группы наценок»

На четвёртой форме мы размести информацию о программе, для кого она предназначена и кем разработана.

2.4. Интерфейс программы

Интерфейс программы, должен использовать стандартные, привычные пользователям элементы, и обеспечивать максимальное удобство. Всё это в конечном счёте определяется таким критерием как эффективность интерфейса - максимальный результат с минимальными усилиями.

Принципы создания удобного интерфейса известны. В качестве самых общих принципов при создании пользовательских интерфейсов можно рассматривать три основных положения:

- программа должна помогать выполнить задачу, а не становиться этой задачей;
- при работе с программой пользователь не должен ощущать себя глупым;
- программа должна работать так, чтобы пользователь не считал компьютер глупым.

Первый принцип — это так называемая "прозрачность" интерфейса. Интерфейс пользователя должен быть интуитивно понятным, простым для освоения, и не создавать для пользователя проблем, которые он вынужден будет преодолевать в процессе работы. Следует использовать стандартные, без излишнего украшательства компоненты, применять привычные, используемые аналогичными программами приёмы управления, и мы достигнем критериев выполнения первого принципа.

Второй принцип заключается в пренебрежении интеллектуальными способностями пользователей. Нам известно, что часто пользователи не только не умеют работать за компьютером, но и просто боятся предпринять что-либо самостоятельно. Поэтому интерфейс пользователя должен быть максимально дружественным. Тем более, что опасения пользователей зачастую оправданны, ведь стоимость программы, да и самого компьютера не идёт ни в какое сравнение со стоимостью, например, созданной многолетними усилиями базы данных.

Третий принцип заключается в том, чтобы создавать программу с максимально возможными "умственными" способностями. Несмотря на быстрое развитие компьютерной техники, даже широко распространённые программы лишь весьма условно можно назвать имеющими искусственный интеллект. Они мешают работе пользователя, выводя на экран диалоговые окна с вопросами, вызывающими недоумение даже в простейших ситуациях.

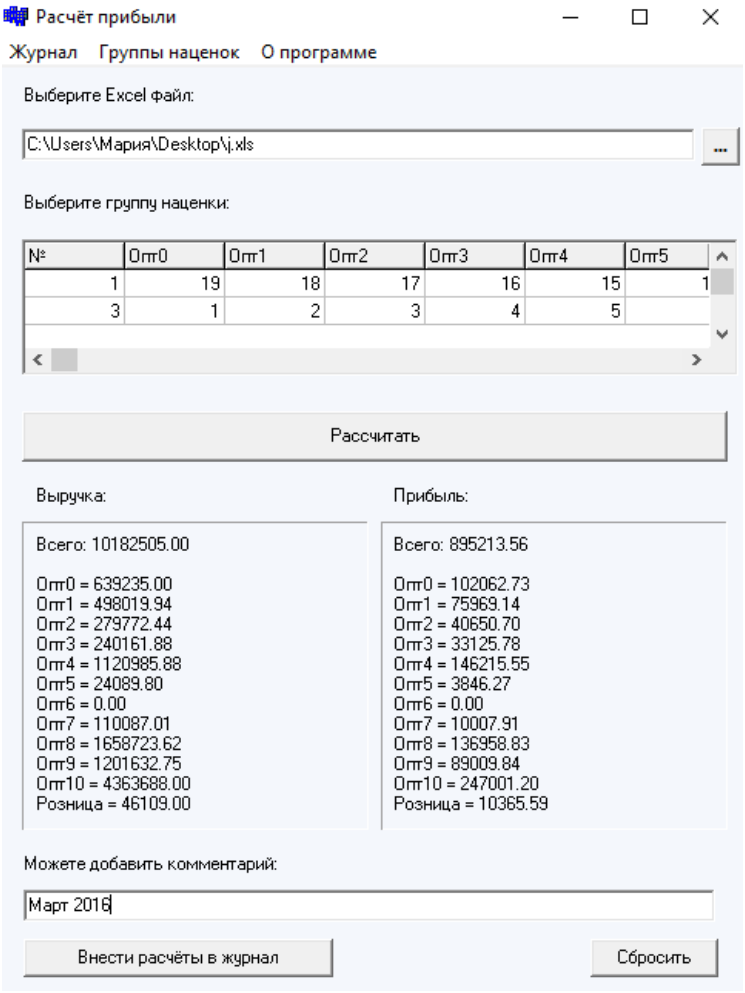

Рис. 2.14. Главная форма программы

Запуская программу пользователю открывается первая форма (см. рис.2.14), на которой он может выбрать файл для расчётов, нажав на кнопку в правом верхнем углу.

На ней представлена таблица, на которой отображены группы наценок. Кликнув по строчке с требуемой ему группой, пользователь выбирает с какой наценкой ему рассчитать прибыль из файла. Дальше пользователь жмёт

кнопку «Рассчитать» и ниже отображаются все полученные значения: вся выручка, выручка по каждому сегменту, вся прибыль и прибыль по каждому сегменту.

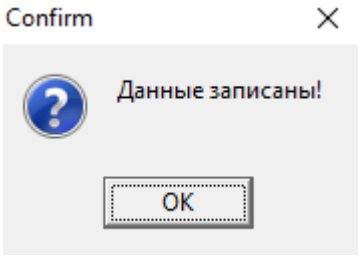

Рис.2.15. «Подтверждение добавления записи в БД»

Произведя расчёты, пользователь может записать их в журнал и по желанию добавить комментарий, кликнув по кнопке «Внести расчёты в журнал». Если запись прошла успешно высветится окошко подтверждения (Рис 2.6).

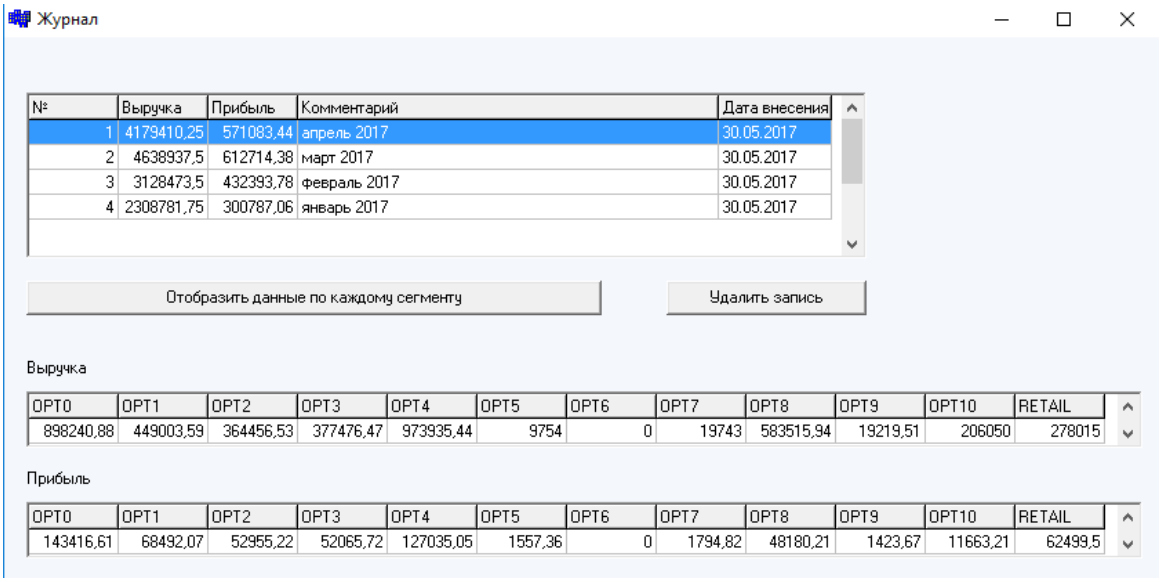

Рис. 2.16. Журнал в программе

Кликнув мышью по вкладке «Журнал» перед пользователем откроется новое окно (см. рис.2.16). На нём мы можем видеть все записи в журнале. Если потребуется посмотреть более точную информацию о расчётах конкретной записи, нужно выбрать запись и нажать на кнопку «Отобразить по каждому сегменту».

Ниже отобразится значение выручки и прибыли по каждому сегменту, для более детального рассмотрения.

Чтобы избавиться от записи, пользователь выбирает нужную запись, после чего нажимает кнопку «Удалить запись» и запись мгновенно исчезает.

При нажатии на кнопку «Группы наценок» на главной форме всплывает новое окно, на котором мы видим отображение всех групп наценок (рис. 2.17). Пользователю предоставляется возможность удалить любую группу, для этого ему нужно кликнуть на строку в таблице и следом на кнопку «Удалить группу».

Ниже представлены поля для добавления новой группы наценок. Заполнив их и нажав на кнопку «Добавить группу», произведется запись в базу данных и отобразится в таблице со всеми группами.

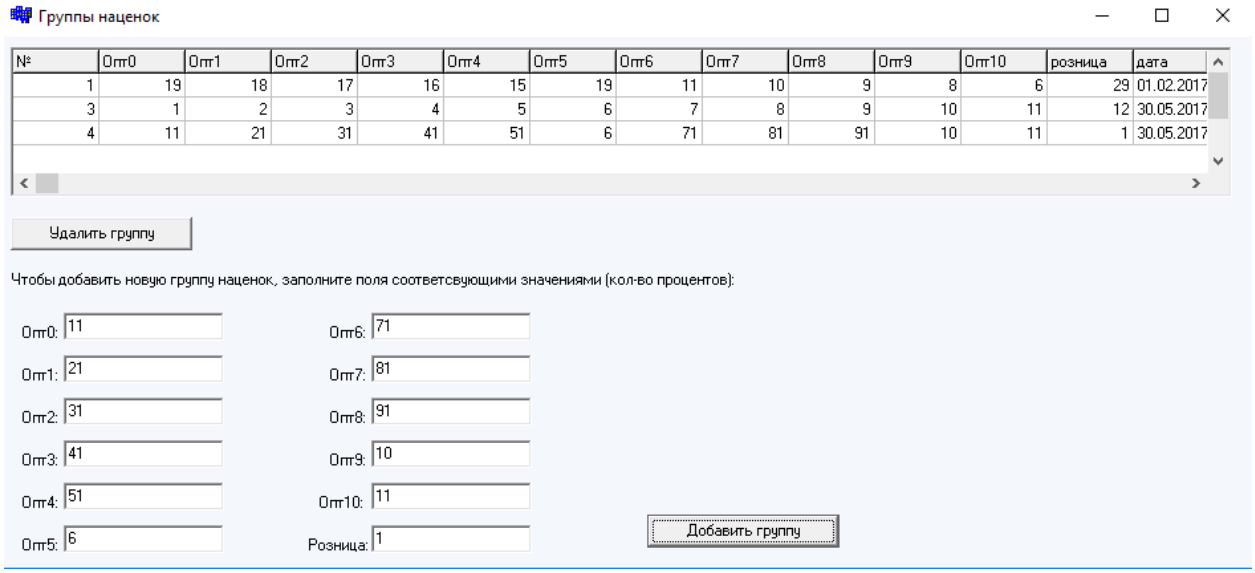

Рис. 2.17. Группы наценок в программе

Последней, не обязательной формой является «О программе» (рис.2.18). Она не выполняет никаких функций, а просто предоставляет информацию о том, для кого предназначена программа, её версия и автор. Единственное, что пользователь может с ней сделать это закрыть, нажав на кнопку «Закрыть».

В итоге мы получили программу из 4 форм, одна из них главная и имеет кнопки для перехода на следующие. На всех формах пользователь взаимодействует с базой данных.

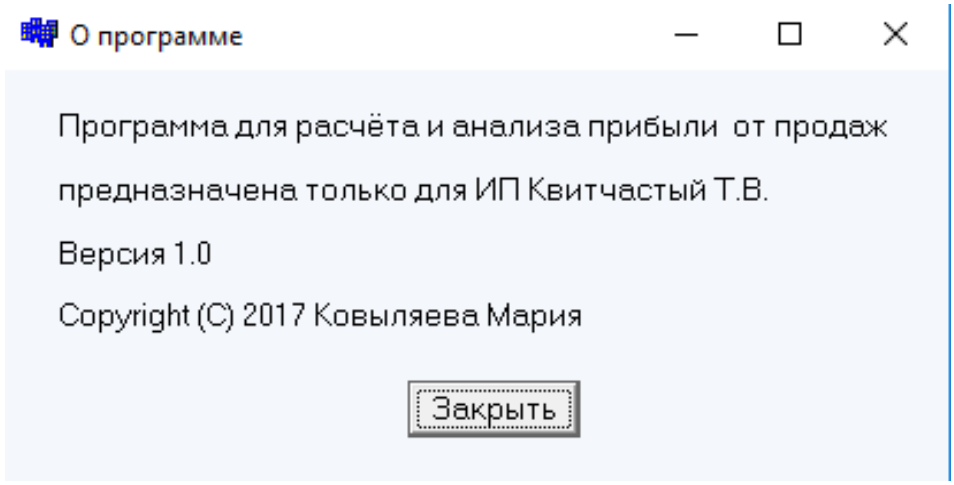

Рис. 2.18. Четвёртая форма «О программе»

Программа не сложная в управлении и понятии, она не выполняет много функций, её задача рассчитать и сохранить значения, и по требованию пользователя отобразить.

Глава 3. Тестирование и внедрение программы

3.1 Тестирование программы

Тестирование программы – это процесс проверки программного продукта с целью продемонстрировать разработчику и заказчику то, что программа выполняет заявленные требования и определить некорректное выполнение задач для дальнейшего исправления.

Функциональное тестирование является одним из ключевых видов тестирования, задача которого – установить соответствие разработанного программного обеспечения исходным функциональным требованиям заказчика. То есть проведение функционального тестирования позволяет проверить способность информационной системы в определенных условиях решать задачи, нужные пользователям.

В зависимости от степени доступа к коду системы можно выделить два типа функциональных испытаний:

- тестирование black box (черный ящик) – проведение функционального тестирования без доступа к коду системы,
- тестирование white box (белый ящик) – функциональное тестирование с доступом к коду системы.

Тестирование black box проводится без знания внутренних механизмов работы системы и опирается на внешние проявления ее работы. При этом тестировании проверяется поведение ПО при различных входных данных и внутреннем состоянии систем. В случае тестирования white box создаются тест-кейсы, основанные преимущественно на коде системы ПО. Также существует расширенный тип black-box тестирования, включающего в себя изучение кода, – так называемый grey box (серый ящик).

Мы произвели тестирование по принципу чёрного ящика. Хотя у нас есть доступ к коду, мы всё таки будем тестировать от лица пользователя. Это

нам позволит в первую очередь решить проблемы пользователя. В этой ВКР было проведено тестирование программы анализа продаж для интернет – магазина Autodoc.ru. Результаты, которые требовалось получить:

1. в приложении интерфейс должен быть интуитивно понятным и не создавать сложности в использовании сотрудниками Autodoc;
2. приложение должно решать поставленные задачи;
3. приложение должно работать без ошибок;
4. время на расчёт прибыли должно быть незначительным.

Определить насколько понятен интерфейс пользователю будет возможным на этапе внедрения. Поэтому первый пункт мы пока пропустили. Пункт второй говорит о том, что программа должна решать поставленные задачи, а именно: считывать данные из Excel-файла, рассчитывать прибыль и записывать расчёты в базу данных.

Считывание данных из файла производится без ошибок, в случае если выбран не Excel файл, программа просто выведет значение всех расчётов равными нулями. Размер файла и количество в нём записей не имеет значения. Однако если будет нарушен принцип записей, то программа рассчитает не верно. Для получения результата, о точности расчёта прибыли программой, были использованы тестовые данные.

Чтобы убедиться в том, что программа верно рассчитывает прибыль по каждому сегменту был проведен сравнительный анализ. Для этого нам требуется рассчитать всё, что требуем от программы, любым проверенным способом и после этого сравнить результаты с теми, что предоставила нам программа. Был выбран ручной способ расчёта, ведь именно таким способом и рассчитывали прибыль в магазине Autodoc, на что тратили много времени. Анализ предоставлен в таблице 3.1.

В результате мы видим, что расчёты программы и ручные ничем не отличаются друг от друга. Исходя из чего можно убедиться в корректности считывания из Excel - файла и подсчёта прибыли разработанной программы.

Таблица 3.1.

Сравнительный анализ расчётов

Название сегмента	Результат расчёта прибыли предоставленный программой	Результат ручного расчёта прибыли
Вся прибыль	895213,56	895213,56
Опт 0	102062,73	102062,73
Опт 1	75969,14	75969,14
Опт 2	40650,70	40650,70
Опт 3	33125,78	33125,78
Опт 4	146215,55	146215,55
Опт 5	3846,27	3846,27
Опт 6	0,00	0,00
Опт 7	10007,91	10007,91
Опт 8	136958,83	136958,83
Опт 9	89099,84	89099,84
Опт 10	247001,20	247001,20
Розница	10365,59	10365,59

В ходе тестирования ошибок с записью в БД выявлено не было. Внесение в базу производятся мгновенно, и пользователь сразу видит результат. Запросы также не заставляют пользователя ждать. Например, на форме «Журнал» при выборе записи и нажатии на кнопку «Отобразить данные по каждому сегменту» мгновенно отображаются расчёты по каждому сегменту (рис. 3.1).

Для того чтобы проанализировать потраченное время на расчёт прибыли в зависимости от размера Excel – файла были проведены тестовые расчёты, представленные в таблице 3.2.

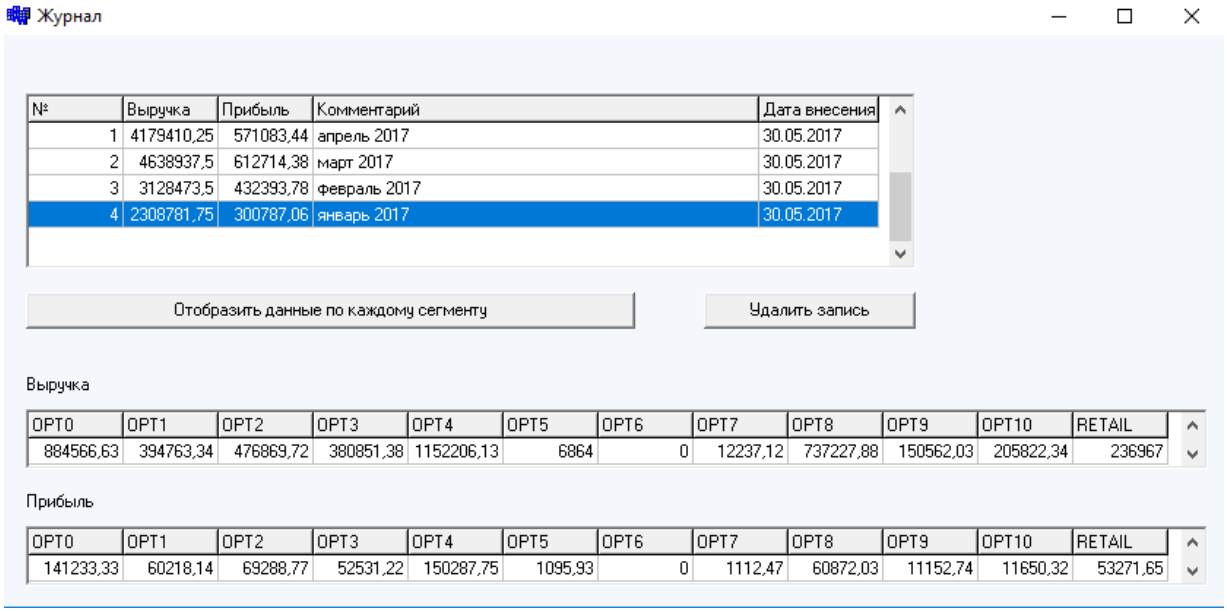

Рис.3.1. Отображение запроса в Журнале

Из тестовых данных (таблица 3.2) следует, что увеличение времени, потраченного на расчёты, прямо пропорционально увеличению записей в Excel – файле. Однако рост незначителен и при очень большом количестве записей, что мало вероятно со слов руководителя магазина Autodoc, расчёт не займет много времени. График роста времени от объёма начальных данных представлен на рисунке 3.2.

Таблица 3.2

Анализ времени на расчёт прибыли и выручки

№	Количество записей в Excel – файле(шт.)	Время потраченное программой на расчёт (с)
1.	797	2,8
2.	583	2,2
3.	484	2,2

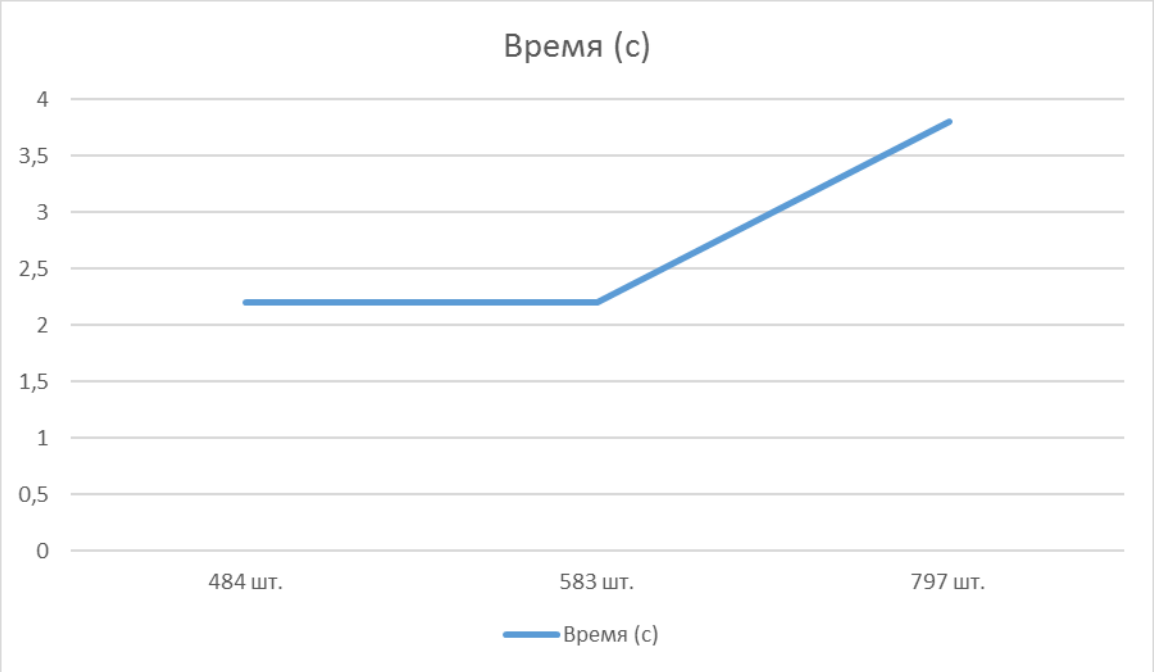

Рис. 3.2. График зависимости времени от кол-ва расчётных данных

Протестировав программу, сделаны следующие выводы: программа работает без ошибок, расчёты проводит верно, время на расчёты остаётся незначительным. Успешно пройденное тестирование позволяет перейти к следующему этапу оптимизации рабочего процесса в магазине Autodoc, а именно к внедрению программы.

3.2. Внедрение программы

Внедрение программного обеспечения — процесс настройки программного обеспечения под определённые условия использования, а также обучения пользователей работе с программным продуктом.

При внедрении программного обеспечения требуется действие в трёх следующих плоскостях работ:

- это выделение критических, с точки зрения общего результата, процедур в деятельности организации. Когда набор таких процедур определен, необходимо в первую очередь использовать ИТ-решение для

автоматизации операций внутри именно этих процедур. Таким образом, разработанное ИТ-решение автоматически становится жизненно важным и востребованным для организации, а также будет обеспечена публичность процесса внедрения;

- это по своей сути расширение нормативной базы организации путём включения в неё регламентов, описывающих порядок выполнения процедур автоматизируемых процессов. В противном случае есть опасность возникновения рассогласования между автоматизированными процедурами и остальными процессами организации;
- это выполнение работ по общей стандартизации существующей деятельности организации, когда выделяются лучшие практики выполнения процедур и включаются в ИТ-решение по принципу наибольшей полезности для большинства участников. Процент таких процедур относительно общего объема автоматизации может быть невелик, но это придает процессу построения решения вес в организации за счет увеличения его «полезности».

Этап внедрения является одним из самых важных для магазина Autodoc. Ведь именно тогда пользователь встречается с программным продуктом, с которым в перспективе ему предстоит работать. Программа должна облегчать ему работу, а не в коем случае не усложнять. На этом этапе пользователь может попробовать и оценить насколько мы выполнили свою главную цель, а именно автоматизировали расчёт прибыли в магазине Autodoc. Для программы, в случае необходимости, может потребоваться создание сопровождения системы, закупка и завоз специального оборудования, подключение к каналам связи или монтирование оборудования.

Обговорив все нюансы с руководителем магазина Autodoc было принято решение: когда будет открываться третий магазин в г. Белгороде, там будет поставлен свой сервер, к серверу подключены все магазины и только тогда на сервер и будет загружена база данных. Поэтому на данном этапе программу устанавливали только вместе с БД.

Так как разработанной программой будут пользоваться руководители и отдел бухгалтерии, в первую очередь было произведено их обучение. Был рассказан принцип работы программы, указано где выводятся результаты расчётов, принцип записей в базе данных, где смотреть содержимое, как удалять и добавлять новые записи и представлен пользовательский интерфейс. После обучения программа была предоставлена для рассмотрения и первого использования руководителям и отделу бухгалтерии.

Интерфейс пользователям был удобен и интуитивно - понятен. Сложностей с ним не возникло.

Следующим шагом было испытание на работоспособность и соответствие техническому заданию. Были использованы реальные данные и произведены расчёты. Дефектов и ошибок выявлено не было, устранение неисправностей не потребовалось.

Далее на этапе тестирование производился ввод и удаление из БД. Записи были корректными, сразу отображались в таблицах.

В течении недели программа постоянно эксплуатировалась. За это время ошибок выявлено не было. Программа работала исправно.

3.3. Возможности развития программы

Использование программы в других компаниях невозможно, поскольку она разработана на решение конкретной задачи, только для магазина Autodoc. Однако улучшить программу можно следующими способами:

1. сделать подсчёт количества клиентов по каждому субъекту, для более детального рассмотрения продаж относительно наценок;
2. реализовать составление отчётов в Word, также посредством OLE-технологии, по выбранным параметрам. Представить на документе графики зависимости и таблицы результатов расчётов за месяц, год или два;

3. реализовать дополнительный журнал для различных расходов компании, которые по желанию, могли автоматически вычитаться из прибыли;

4. поместить базу данных на сервер, что уже является задачей на будущее;

5. реализовать учётные записи, чтобы сохранять конфиденциальность важной информации, а точнее прибыли магазина.

Мы разобрали некоторые из возможных вариантов развития программы. В случае, если потребность в её существовании в магазине не прекратится, можно будет задуматься о её усовершенствовании.

На сегодняшний день это первое внедрение программного обеспечения в работу Белгородского представительства интернет магазина Autodoc.ru.

ЗАКЛЮЧЕНИЕ

В настоящей выпускной квалификационной работе была рассмотрена разработка программы анализа продаж для интернет магазина Autodoc.ru.

В ходе выполнения ВКР была достигнута цель, а именно оптимизирован процесс расчёта прибыли в интернет магазине Autodoc. Достижение указанной цели осуществилось посредством решения ряда задач:

1. изучены данные для расчёта;
2. исследована формула вычисления прибыли;
3. спроектировано информационное и программное обеспечение;
4. исследована OLE технология;
5. разработана программа анализа прибыли;
6. программа внедрена в работу магазина Autodoc.

В итоге мы получили программу с удобным, интуитивно понятным интерфейсом, которая выполняет поставленные задачи и сокращает время сотрудникам Autodoc на подсчёт прибыли. Результаты, которые мы получили, это повышение оперативности в работе за счет внедрения прогрессивных технологий, современного программного и технического обеспечения, повышение оперативности получения отчетной информации для анализа и повышение информативности системы для принятия конструктивных управленческих решений руководством компании.

В ходе выполнения ВКР были получены теоретические знания и практические навыки по разработке программ.

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

1. Дэвид Гриффитс, Дон Гриффитс - Изучаем программирование на С 2013г. Изд. Эксмо.

2. Брюс Эккель, Чак Эллисон - Философия С++. Часть 1. Введение в стандартный С++ 2004г. Изд. Питер.

3. Архангельский А.Я. - Программирование в С++ Builder 2010г. Изд. Бином, 7-е издание.

4. Построение инфологической модели [Электронный ресурс] - http://www.sbras.ru/rus/docs/db/rdbms/5-2.html.

5. Инфологическое моделирование [Электронный ресурс] - http://orloff.am.tpu.ru/labs_itkd/kr1/infomodel.htm.

6. Физическая модель бд [Электронный ресурс] - http://bourabai.kz/dbt/dbms/03.htm.

7. Определение проектирования программного обеспечения [Электронный ресурс] - http://dic.academic.ru/dic.nsf/fin_enc/27885.

ПРИЛОЖЕНИЕ

Form1:

```
#include <vcl.h>
#pragma hdrstop
#include "Unit1.h"
#include <utilcls.h>    //подключили библиотеку для OLE
#include <cmath>
#include <cstdlib>
#include <vector>
#include <string>
#include "Unit2.h"
#include "Unit3.h"
#include "Unit4.h"
#include "Unit5.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;

float proc, proc0, proc1, proc2, proc3, proc4, proc5, proc6, proc7, proc8, proc9, proc10, procR;
float prof, prof0, prof1, prof2, prof3, prof4, prof5, prof6, prof7, prof8, prof9, prof10, profR;
float m, m0, m1, m2, m3, m4, m5, m6, m7, m8, m9, m10, mr;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
      : TForm(Owner)
{
Button1->Enabled=false;
Button3->Enabled=false;
Button4->Enabled=false;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button1Click(TObject *Sender)
{
String a;
proc=0;
proc0=0; proc1=0; proc2=0; proc3=0; proc4=0; proc5=0; proc6=0; proc7=0; proc8=0; proc9=0;
proc10=0; procR=0;
prof=0;
prof0=0; prof1=0; prof2=0; prof3=0; prof4=0; prof5=0; prof6=0; prof7=0; prof8=0; prof9=0;
prof10=0; profR=0;
std::vector<float>A;
a=Edit1->Text;
m= StrToInt(Form1->DBGrid1->SelectedField->Text);
DataModule2->IBStoredProc1->ParamByName("ID_MARKUP_NEW")->AsInteger=m;
DataModule2->IBStoredProc1->Prepare();
DataModule2->IBStoredProc1->ExecProc();
```

```
m0 = DataModule2->IBStoredProc1->ParamByName("markupopt0_new")->AsInteger;
m1 = DataModule2->IBStoredProc1->ParamByName("markupopt1_new")->AsInteger;
m2 = DataModule2->IBStoredProc1->ParamByName("markupopt2_new")->AsInteger;
m3 = DataModule2->IBStoredProc1->ParamByName("markupopt3_new")->AsInteger;
m4 = DataModule2->IBStoredProc1->ParamByName("markupopt4_new")->AsInteger;
m5 = DataModule2->IBStoredProc1->ParamByName("markupopt5_new")->AsInteger;
m6 = DataModule2->IBStoredProc1->ParamByName("markupopt6_new")->AsInteger;
m7 = DataModule2->IBStoredProc1->ParamByName("markupopt7_new")->AsInteger;
m8 = DataModule2->IBStoredProc1->ParamByName("markupopt8_new")->AsInteger;
m9 = DataModule2->IBStoredProc1->ParamByName("markupopt9_new")->AsInteger;
m10 = DataModule2->IBStoredProc1->ParamByName("markupopt10_new")->AsInteger;
mr = DataModule2->IBStoredProc1->ParamByName("markupretail_new")->AsInteger;

Variant cnn ;   //Обработка исключения
try
{
cnn = CreateOleObject("EXCEL.Application");    //запускаем Excel
}
catch (...)
{
MessageBox(0, "Ошибка при открытии сервера excel", "Ошибка", MB_OK);
return;
}
cnn.OlePropertyGet("Workbooks").OlePropertyGet("Open",a.c_str());//Открываем книгу

int N=0;          //Переменная будет сохранять кол-во элементов в векторе
String h;        //Переменная будет хранить содержимое из ячей xls файла

for(int j=0; ;j++){

int l=j+2;                    //Переменная содержит номер строки в xls файле
h=cnn.OlePropertyGet("Cells",l,3);   //Вытаскиваем содержимое
A.push_back(j);                    //Создаём новый элемент
float value=0;
if(TryStrToFloat(h, value)) A[j] = value;   //Присваиваем элементу вектора содержимое
else {
N=j;                                //Присваиваем общее кол-во ячеек
break;
}
}

for(int i=0; i<N-1 ;i++){
int l=i+2;                        //Переменная содержит номер строки в xls файле
h=cnn.OlePropertyGet("Cells",l,2);   //Вытаскиваем содержимое следующего столбца

if (h=="Опт0") proc0+=A[i];
else if (h=="Опт1") proc1+=A[i];
else if (h=="Опт2") proc2+=A[i];
else if (h=="Опт3") proc3+=A[i];
else if (h=="Опт4") proc4+=A[i];
else if (h=="Опт5") proc5+=A[i];
else if (h=="Опт6") proc6+=A[i];
```

```
else if (h=="Опт7") proc7+=A[i];
else if (h=="Опт8") proc8+=A[i];
else if (h=="Опт9") proc9+=A[i];
else if (h=="Опт10") proc10+=A[i];
else procR+=A[i];
}

cnn.OleProcedure("Quit");  //Закрываем Excel
A.clear();

prof0=proc0-(proc0/(1+(m0/100)));
prof1=proc1-(proc1/(1+(m1/100)));
prof2=proc2-(proc2/(1+(m2/100)));
prof3=proc3-(proc3/(1+(m3/100)));
prof4=proc4-(proc4/(1+(m4/100)));
prof5=proc5-(proc5/(1+(m5/100)));
prof6=proc6-(proc6/(1+(m6/100)));
prof7=proc7-(proc7/(1+(m7/100)));
prof8=proc8-(proc8/(1+(m8/100)));
prof9=proc9-(proc9/(1+(m9/100)));
prof10=proc10-(proc10/(1+(m10/100)));
profR=procR-(procR/(1+(mr/100)));

proc=proc0+proc1+proc2+proc3+proc4+proc5+proc6+proc7+proc8+proc9+proc10+procR;
prof=prof0+prof1+prof2+prof3+prof4+prof5+prof6+prof7+prof8+prof9+prof10+profR;

String s;
Label1->Caption= "Всего: " + s.sprintf("%0.2f",proc) + "\n\n"
+ "Опт0 = " + s.sprintf("%0.2f",proc0)+ "\n"
+ "Опт1 = " + s.sprintf("%0.2f",proc1) + "\n"
+ "Опт2 = " + s.sprintf("%0.2f",proc2) + "\n"
+ "Опт3 = " + s.sprintf("%0.2f",proc3) + "\n"
+ "Опт4 = " + s.sprintf("%0.2f",proc4) + "\n"
+ "Опт5 = " + s.sprintf("%0.2f",proc5) + "\n"
+ "Опт6 = " + s.sprintf("%0.2f",proc6) + "\n"
+ "Опт7 = " + s.sprintf("%0.2f",proc7) + "\n"
+ "Опт8 = " + s.sprintf("%0.2f",proc8) + "\n"
+ "Опт9 = " + s.sprintf("%0.2f",proc9) + "\n"
+ "Опт10 = "+ s.sprintf("%0.2f",proc10) + "\n"
+ "Розница = " + s.sprintf("%0.2f",procR);

Label2->Caption= "Всего: " + s.sprintf("%0.2f",prof) + "\n\n"
+ "Опт0 = " + s.sprintf("%0.2f",prof0)+ "\n"
+ "Опт1 = " + s.sprintf("%0.2f",prof1) + "\n"
+ "Опт2 = " + s.sprintf("%0.2f",prof2) + "\n"
+ "Опт3 = " + s.sprintf("%0.2f",prof3) + "\n"
+ "Опт4 = " + s.sprintf("%0.2f",prof4) + "\n"
+ "Опт5 = " + s.sprintf("%0.2f",prof5) + "\n"
+ "Опт6 = " + s.sprintf("%0.2f",prof6) + "\n"
+ "Опт7 = " + s.sprintf("%0.2f",prof7) + "\n"
+ "Опт8 = " + s.sprintf("%0.2f",prof8) + "\n"
+ "Опт9 = " + s.sprintf("%0.2f",prof9) + "\n"
```

```
+ "Опт10 = "+ s.sprintf("%0.2f",prof10) + "\n"
+ "Розница = " + s.sprintf("%0.2f",profR);

Button3->Enabled=true;
Button4->Enabled=true;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button2Click(TObject *Sender)
{
String A;
OpenDialog1->Execute();
A=OpenDialog1->FileName;
Edit1->Text=A;
if (Edit1->Text!="")
Button1->Enabled=true;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button4Click(TObject *Sender)
{
Edit1->Text="";
Edit2->Text="";
Label1->Caption="";
Label2->Caption="";

Button1->Enabled=false;
Button3->Enabled=false;
Button4->Enabled=false;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button3Click(TObject *Sender)
{

int id_jour=0;
String comment;
TDateTime CurrentDate = Date();

comment = Edit2->Text;

DataModule2->IBStoredProc6->ParamByName("ID_JOURNAL_NEW")->AsString="";
DataModule2->IBStoredProc6->ParamByName("PROCEEDS_NEW")->AsFloat=proc;
DataModule2->IBStoredProc6->ParamByName("PROFIT_NEW")->AsFloat=prof;
DataModule2->IBStoredProc6->ParamByName("COMMENT_NEW")->AsString=comment;
DataModule2->IBStoredProc6->ParamByName("DATE_INSERT_NEW")-
>AsString=CurrentDate;
DataModule2->IBStoredProc6->ParamByName("ID_MARKUP_NEW")->AsInteger=m;
DataModule2->IBStoredProc6->Prepare();
DataModule2->IBStoredProc6->ExecProc();

DataModule2->IBStoredProc2->ExecProc();
id_jour = DataModule2->IBStoredProc2->ParamByName("ID_journal_NEW")->AsInteger;
```

```
DataModule2->IBStoredProc3->ParamByName("ID_PROCEEDS_NEW")->AsString="";
DataModule2->IBStoredProc3->ParamByName("PROCEEDS_OPT0_NEW")->AsFloat=proc0;
DataModule2->IBStoredProc3->ParamByName("PROCEEDS_OPT1_NEW")->AsFloat=proc1;
DataModule2->IBStoredProc3->ParamByName("PROCEEDS_OPT2_NEW")->AsFloat=proc2;
DataModule2->IBStoredProc3->ParamByName("PROCEEDS_OPT3_NEW")->AsFloat=proc3;
DataModule2->IBStoredProc3->ParamByName("PROCEEDS_OPT4_NEW")->AsFloat=proc4;
DataModule2->IBStoredProc3->ParamByName("PROCEEDS_OPT5_NEW")->AsFloat=proc5;
DataModule2->IBStoredProc3->ParamByName("PROCEEDS_OPT6_NEW")->AsFloat=proc6;
DataModule2->IBStoredProc3->ParamByName("PROCEEDS_OPT7_NEW")->AsFloat=proc7;
DataModule2->IBStoredProc3->ParamByName("PROCEEDS_OPT8_NEW")->AsFloat=proc8;
DataModule2->IBStoredProc3->ParamByName("PROCEEDS_OPT9_NEW")->AsFloat=proc9;
DataModule2->IBStoredProc3->ParamByName("PROCEEDS_OPT10_NEW")-
>AsFloat=proc10;
DataModule2->IBStoredProc3->ParamByName("PROCEEDS_OPTR_NEW")-
>AsFloat=procR;
DataModule2->IBStoredProc3->ParamByName("ID_JOURNAL_NEW")->AsInteger=id_jour;
DataModule2->IBStoredProc3->Prepare();
DataModule2->IBStoredProc3->ExecProc();

DataModule2->IBStoredProc4->ParamByName("ID_PROFIT_NEW")->AsString="";
DataModule2->IBStoredProc4->ParamByName("PROFIT_OPT0_NEW")->AsFloat=prof0;
DataModule2->IBStoredProc4->ParamByName("PROFIT_OPT1_NEW")->AsFloat=prof1;
DataModule2->IBStoredProc4->ParamByName("PROFIT_OPT2_NEW")->AsFloat=prof2;
DataModule2->IBStoredProc4->ParamByName("PROFIT_OPT3_NEW")->AsFloat=prof3;
DataModule2->IBStoredProc4->ParamByName("PROFIT_OPT4_NEW")->AsFloat=prof4;
DataModule2->IBStoredProc4->ParamByName("PROFIT_OPT5_NEW")->AsFloat=prof5;
DataModule2->IBStoredProc4->ParamByName("PROFIT_OPT6_NEW")->AsFloat=prof6;
DataModule2->IBStoredProc4->ParamByName("PROFIT_OPT7_NEW")->AsFloat=prof7;
DataModule2->IBStoredProc4->ParamByName("PROFIT_OPT8_NEW")->AsFloat=prof8;
DataModule2->IBStoredProc4->ParamByName("PROFIT_OPT9_NEW")->AsFloat=prof9;
DataModule2->IBStoredProc4->ParamByName("PROFIT_OPT10_NEW")->AsFloat=prof10;
DataModule2->IBStoredProc4->ParamByName("PROFIT_OPTR_NEW")->AsFloat=profR;
DataModule2->IBStoredProc4->ParamByName("ID_JOURNAL_NEW")->AsInteger=id_jour;
DataModule2->IBStoredProc4->Prepare();
DataModule2->IBStoredProc4->ExecProc();

MessageDlg("Данные записаны!", mtConfirmation, TMsgDlgButtons()<< mbOK , 0);
DataModule2->IBTable2->Close();
DataModule2->IBTable2->Open();
}
//---------------------------------------------------------------------------
void __fastcall TForm1::N1Click(TObject *Sender)
{
Form3->Show();
}
//---------------------------------------------------------------------------
void __fastcall TForm1::N2Click(TObject *Sender)
{
Form4->Show();
}
//---------------------------------------------------------------------------
```

```
void __fastcall TForm1::N4Click(TObject *Sender)
{
Form5->Show();
}
//---------------------------------------------------------------------------
```

Form3:

```
#include <vcl.h>
#pragma hdrstop
#include "Unit1.h"
#include "Unit2.h"
#include "Unit3.h"
#include "Unit4.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm3 *Form3;

void __fastcall TForm3::Button1Click(TObject *Sender)
{
int id;
id= StrToInt(Form3->DBGrid1->SelectedField->Text);

DataModule2->IBQuery1->Close();
DataModule2->IBQuery1->ParamByName("id_journal_new")->AsInteger=id;
DataModule2->IBQuery1->Open();

DataModule2->IBQuery2->Close();
DataModule2->IBQuery2->ParamByName("id_journal_new")->AsInteger=id;
DataModule2->IBQuery2->Open();
}
//---------------------------------------------------------------------------
void __fastcall TForm3::Button2Click(TObject *Sender)
{
int id;
id= StrToInt(Form3->DBGrid1->SelectedField->Text);

DataModule2->IBStoredProc5->ParamByName("ID_JOURNAL_NEW")->AsInteger=id;
DataModule2->IBStoredProc5->Prepare();
DataModule2->IBStoredProc5->ExecProc();

DataModule2->IBTable2->Close();
DataModule2->IBTable2->Open();
}
```

Form4:

```
#include <vcl.h>
#pragma hdrstop
#include "Unit1.h"
```

```
#include "Unit2.h"
#include "Unit3.h"
#include "Unit4.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm4 *Form4;
//---------------------------------------------------------------------------
void __fastcall TForm4::Button1Click(TObject *Sender)
{
int id;
id= StrToInt(Form4->DBGrid1->SelectedField->Text);

DataModule2->IBStoredProc7->ParamByName("ID_MARKUP_NEW")->AsInteger=id;
DataModule2->IBStoredProc7->Prepare();
DataModule2->IBStoredProc7->ExecProc();

DataModule2->IBTable1->Close();
DataModule2->IBTable1->Open();
}
//---------------------------------------------------------------------------
void __fastcall TForm4::Button2Click(TObject *Sender)
{
int m0=0, m1=0, m2=0, m3=0, m4=0, m5=0, m6=0, m7=0, m8=0, m9=0, m10=0, mr=0;
m0=StrToInt(Edit1->Text);
m1=StrToInt(Edit2->Text);
m2=StrToInt(Edit3->Text);
m3=StrToInt(Edit4->Text);
m4=StrToInt(Edit5->Text);
m5=StrToInt(Edit6->Text);
m6=StrToInt(Edit7->Text);
m7=StrToInt(Edit8->Text);
m8=StrToInt(Edit9->Text);
m9=StrToInt(Edit10->Text);
m10=StrToInt(Edit11->Text);
mr=StrToInt(Edit12->Text);

TDateTime CurrentDate = Date();

DataModule2->IBStoredProc8->ParamByName("ID_markup_NEW")->AsString="";
DataModule2->IBStoredProc8->ParamByName("markupopt0_NEW")->AsInteger=m0;
DataModule2->IBStoredProc8->ParamByName("markupopt1_NEW")->AsInteger=m1;
DataModule2->IBStoredProc8->ParamByName("markupopt2_NEW")->AsInteger=m2;
DataModule2->IBStoredProc8->ParamByName("markupopt3_NEW")->AsInteger=m3;
DataModule2->IBStoredProc8->ParamByName("markupopt4_NEW")->AsInteger=m4;
DataModule2->IBStoredProc8->ParamByName("markupopt5_NEW")->AsInteger=m5;
DataModule2->IBStoredProc8->ParamByName("markupopt6_NEW")->AsInteger=m6;
DataModule2->IBStoredProc8->ParamByName("markupopt7_NEW")->AsInteger=m7;
DataModule2->IBStoredProc8->ParamByName("markupopt8_NEW")->AsInteger=m8;
DataModule2->IBStoredProc8->ParamByName("markupopt9_NEW")->AsInteger=m9;
DataModule2->IBStoredProc8->ParamByName("markupopt10_NEW")->AsInteger=m10;
DataModule2->IBStoredProc8->ParamByName("markupoptR_NEW")->AsInteger=mr;
```

```
DataModule2->IBStoredProc8->ParamByName("data_change_NEW")->AsString=CurrentDate;
DataModule2->IBStoredProc8->Prepare();
DataModule2->IBStoredProc8->ExecProc();

DataModule2->IBTable1->Close();
DataModule2->IBTable1->Open();
DataModule2->IBTable2->Close();
DataModule2->IBTable2->Open();
}
```

Form5:

```
#include <vcl.h>
#pragma hdrstop

#include "Unit5.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm5 *Form5;
//---------------------------------------------------------------------------

void __fastcall TForm5::Button1Click(TObject *Sender)
{
Form5->Close();
}
```

Выпускная квалификационная работа выполнена мной совершенно самостоятельно. Все использованные в работе материалы и концепции из опубликованной научной литературы и других источников имеют ссылки на них.

«___» _____ _____ г.

_____ _____

(подпись) *(Ф.И.О.)*