Введение

Целью дипломного проектирования является создание программного продукта предназначенного для автоматизации учета оплаты договоров за обучение.

Внедрение разработанного программного позволит быстро и оперативно получать нужную информацию, значительно сократит затраты, связанные с обработкой информации, освободит от хранения большого объема информации на бумаге, сократит рутинные вычисления при получении выходных документов.

Социально-экономический раздел дипломного проекта посвящен рассмотрению роли информационных технологий в управлении. В разделе приводится обоснование ожидаемых результатов повышения эффективности работы от использования разработанного программного продукта.

В научно-исследовательском разделе описывается объект автоматизации и его математическая модель. В разделе приводится обзор программных средств разработки приложений и обосновывается выбор языка программирования. Также в этом разделе приводится классификация приложений для работы с базами данных, дается обзор существующих систем управления базами и обосновывается выбор СУБД.

В специальном разделе дается описание информационной модели объекта автоматизации, приводится структура разработанной базы данных, функциональная структура разработанного программного продукта с указанием назначения программных модулей. Кроме того, в этом разделе приводится руководство пользователя по эксплуатации созданного программного продукта и руководство программиста по установке и сопровождению программы.

1. Социально-экономический раздел

1.1 Роль информационных технологий в совершенствовании управления

Компьютер по праву считается одним из высших достижений современной цивилизации. Пройдя в течение нескольких десятилетий путь от конструкции, занимающей несколько комнат и потребляющей соответствующее количество электроэнергии, до нескольких блоков, свободно размещающихся на рабочем столе, компьютер превратился в верного помощника миллионов людей в работе и отдыхе. При этом, несмотря на столь значительный прогресс, компьютер и сопутствующие ему технологии продолжают развиваться столь же стремительными темпами.

Быстрое развитие вычислительной техники привело к прочному входу компьютеров в жизнь и деятельность человека. С появлением новых, более современных и быстродействующих компьютеров все больше увеличиваютсяобласти их применения. Сейчас уже трудно найти сферы деятельности, где бы еще не применялись компьютеры. Использование компьютера каквспомогательного средства сильно изменило множество профессий человека. Огромное увеличение объема информационных потоков в конце XX века привело к тому, что успешность любого предприятия очень сильно зависит от способности его менеджеров вовремя получить и быстро обработать нужную информацию.

Внедрение ЭВМ, вычислительных систем и средств автоматизации в различные сферы производства, в первую очередь в экономике, управлении и делопроизводстве, позволяет оперировать большими объемами информации. Применение ЭВМ становится необходимым условием успешного выполнения производственных задач. Наглядность и полнота представления большого объема информации на экране монитора способствует более легкому ее восприятию, быстрому анализу.

В 90-е годы идет огромными шагами развитие идеи объединения компьютеров в компьютерные сети для совместного хранения, обработки и использования информации, сначала в местных масштабах (локальные вычислительные сети), а затем и мировых (глобальные). Поскольку компьютеры прочно вошли во все сферы деятельности человеком, возникла проблема обмена данными между отдельными пользователями. Организация компьютерных сетей позволяет получать доступ отдельному пользователю к удаленным базам данных и использовать их для своих нужд. На сегодняшний день уже работают глобальные компьютерные сети, позволяющие миллионам пользователей, причем разных типов компьютеров, в считанные секунды получать интересующую информацию со всех концов света.

Уже сегодня в мире насчитывается несколько миллионов персональных компьютеров. Их популярность объясняется тем, что они просты в эксплуатации, потребляют мало энергии, обладают высокой надежностью быстродействием. Это способствует все большему распространению компьютеров в различные области человеческой деятельности. Развитие вычислительной техники позволяет строить системы нового типа, характеризующиеся, с одной стороны, разнообразием средств отображения информации, с другой – интеграцией этих средств, и обеспечениеммаксимальных условий работы пользователей, не имеющих специальной подготовки. Еще одним преимуществом персональных компьютеров являются их небольшие размеры, что позволяет размещать такие машины на рабочих местах. Организация автоматизированных рабочих мест (АРМ) на производстве способствует сокращению потерь времени, связанных с ежедневными операциями по учету и хранению информации, контролю протекания процесса производственного процесса. Организация связи между такими АРМ с помощью локальной сети обеспечивает:

* быстрый доступ к любой информации;
* анализ состояния производственного процесса и своевременное принятие управленческих решений в случае обнаружения сбоев в процессе производства;
* контроль исполнения управленческих решений.

Однако, сама по себе установка современных технических устройств нарабочих местах не означает повышение эффективности управления производственными процессами. Разработка и применение соответствующего программного обеспечения для таких устройств может способствовать успешному решению сложных научно-технических задач. В состав программного обеспечения входят как универсальные пакеты, так и программы, ориентированные на решение отдельных проблем. Можно выделить три основных класса программных средств:

* прикладные программы и пакеты программ;
* инструментальные системы программирования;
* операционные системы.

Прикладные программы позволяют сделать компьютер действительно удобным средством для выполнения различных работ. Именно прикладные программы и пакеты программ образуют ту информационную оболочку, которая превращает сложный электронный прибор в удобное и простое в использовании информационно-вычислительное устройство. С помощью систем программирования можно создавать программные продукты, направленные на решение: различных производственных задач. Выбор установки той или иной операционной системы зависит от класса задач, которые будут решаться с помощью данных программных средств. Программное обеспечение должно обеспечивать решение необходимых производственных задач, быть простым для обучения пользователей работе на компьютере в кратчайшие сроки, удобным для эксплуатации пользователями, а также отвечать всем требованиям по надежному хранению информации и быстрому доступу к необходимым данным.

Эффективный доступ к информационным ресурсам, контроль за распространением и обменом информацией являются ключевыми моментами управления работой любого предприятия и организации. Объединение персональных компьютеров в локальные сети, интеграция в них различного коммуникационного и офисного оборудования позволили создавать мощные информационно-вычислительные комплексы масштаба предприятий и крупных корпораций. В последние годы появилось понятие «электронный офис», под которым понимается многофункциональная гибкая информационно-вычислительная система предприятия, интегрированная в глобальные сети передачи информации посредством высокопроизводительных каналов связи и комплекса различных услуг.

Преимущества использования ПК для организации деловой жизни становятся очевидны. Это идеальный инструмент для запоминания и управления различной информацией.

Основными источниками экономической эффективности при внедрении ЭВМ служат:

* повышение производительности труда вследствие сокращения потерь рабочего времени;
* повышение научно-технического уровня системы управления и подготовки специалистов;
* увеличение масштабов внедрения программного продукта (ПП).

При широком использовании информационных технологий происходит большая экономия времени за счет сокращения малопродуктивной деятельности, уменьшения количества встреч, а также за счет ускорения процессов анализа и обработки документации. [1]

Помимо экономии времени существует много возможностей по повышению качества работы. Улучшение показателей качества непосредственно связано с 4 важнейшими факторами:

1. Прямой своевременный доступ к информационным ресурсам, осуществляемый с АРМ к внешним и внутренним базам данных.
2. Эффективная координация деятельности с помощью систем передачи речи и систем электронной почты.
3. Более эффективное взаимодействие с людьми за счет использования более информативных и наглядных документов, а также быстродействующих систем передачи сообщений.
4. Выделение необходимого времени на такие высокопродуктивные виды деятельности, как анализ, способствующие в наибольшей степени выполнению возложенных на предприятие функций.

Применение информационной техники обладает рядом существенных преимуществ: Во-первых, это *более широкое распространение* информации. Большое количество людей может быть подсоединено к цифровым сетям и может пользоваться такими услугами, как телемаркетинг, телеобучение, телеигры, телеконференции. Во-вторых, *большая простота пользования* ИТ. Кроме того, повышается удобство. По мере совершенствования программного обеспечения, которое выбирает и формализует информацию в соответствие с индивидуальными запросами, пользование терминалами становится более простой задачей. И, наконец, самое важное преимущество – *снижение цен* в связи с тем, что удельные издержки на информационную технику значительно ниже удельных издержек при использовании традиционных методов (с привлечением человека, бумаги и т.д.).

В связи с внедрением ИТ внутри компаний происходит перераспределение, исключение фаз, связанных с дополнительными затратами, изменяется структура предприятий, ликвидируются отдельные звенья. Использование интегрированных систем, безусловно, приводит к значительному сокращению управленческого аппарата, в частности его среднего звена.

Кроме того, введение ИТ влечет за собой изменение организационной структуры фирм. Более простые связи и более широкие возможности контроля обеспечивают чрезвычайно высокую эффективность использования электронных систем обмена информацией.

Используя в качестве электронных хранилищ информации базы данных пользовали при работе получают большие преимущества. Рассмотрим некоторые из этих преимуществ. В некоторой степени это зависит от того, о какой системе идет речь – однопользовательской или многопользовательской. [2]

В случае однопользовательской системы баз данных по сравнению с традиционным бумажным методом содержания записей преимущества весьма очевидны.

* *Компактность.* Нет необходимости в многотомных бумажных архивах.
* *Скорость.* Компьютер может вести поиск и изменять данные гораздо быстрее человека. В частности, на специальные вопросы, возникающие в процессе работы, можно получить ответ быстро, не затрачивая времени на визуальный поиск.
* *Низкие трудозатраты.* Нет необходимости в утомительной ручной работе над картотекой. Механическую работу машины всегда выполняют быстрее.
* *Применимость.* Точная, свежая информация всегда под рукой.

Эти преимущества приобретают еще большее значение в многопользовательской среде, где база данных, вероятно, больше и сложнее однопользовательской. Кроме того, многопользовательская среда имеет дополнительное преимущество: *система баз данных предоставляет предприятию централизованное управление его данными* (а такое управление является наиболее ценным свойством базы данных). Если предприятие не использует систему баз данных, то для каждого отдельного приложения создаются свои файлы, чаще всего размещаемые на отдельных магнитных лентах или дисках, в результате чего данные оказываются разрозненными. Систематически управлять такими данными очень сложно.

Отметим преимущества использования баз данных, связанные с централизованным управлением.

* Возможность сокращения *избыточности*.

В системах, не использующих базы данных, каждое приложение имеет свои файлы. Это приводит к избыточности хранимых данных, а следовательно, к расточительству памяти. Например, как приложение, связанное с учетом персонала, так и приложение, связанное с учетом обучения служащих, могут иметь собственный файл с ведомственной информацией о служащих. Эти два файла можно объединить с устранением избыточности (одинаковой информации) при условии, что администратор данных знает о том, какие данные нужны для каждого приложения, т.е. если на предприятии осуществляется необходимое общее управление.

* Возможность устранения (до некоторой степени) *противоречивости*.

В действительности это следствие предыдущего пункта. Действительно, если избыточность не контролируется, тогда в какой-то момент возникнет ситуация, при которой две записи в разных приложениях будут не согласованы, а именно: одна из них будет изменена, а другая – нет. Очевидно, если какой-либо факт представлен одной записью, то противоречий возникнуть не может.

* Возможность *общего доступа* к данным.

Общий доступ к данным означает возможность доступа к ним нескольких приложений базы данных и разработки новых приложений для работы с этими же данными, и при этом нет необходимости в создании новых данных.

* Возможность *соблюдения стандартов*.

Благодаря централизованному управлению может обеспечиваться представление данных в определенных стандартах. Стандарты могут быть корпоративными, установочными, ведомственными, промышленными, национальными и интернациональными. Стандартизация представления данных наиболее важна для обмена и перенесения данных между системами. Кроме того, стандарты важны и для совместного использования данных, и для их понимания.

* Возможность введения *ограничений для обеспечения безопасности*.

Благодаря полному контролю над базой данных администратор может обеспечить доступ к базе данных через определенные каналы, а следовательно, может определить правила безопасности, которые будут проверяться при попытке доступа к уязвимым данным. Для разных типов доступа и разных частей базы данных можно установить различные правила. Однако необходимо отметить, что при отсутствии таких правил безопасность данных подвергается большему риску, чем в обычной (разрозненной) файловой системе; т.е. централизованная природа системы баз данных в некотором смысле *требует* наличия хорошей системы безопасности.

* Возможность обеспечения *целостности данных*.

Задача целостности заключается в обеспечении правильности и точности данных в базе данных. Для этого администратор определяет правила целостности, которые применяются при любой попытке проделать какую-либо операцию обновления.

* Возможность *сбалансировать противоречивые требования*.

Зная общие требования всего предприятия, администратор может структурировать базу данных таким образом, чтобы обслуживание в целом для предприятия было наилучшим.

1.2 Ожидаемые результаты повышения эффективности управления при использовании разработанного программного продукта

Разработанная в дипломном проекте программа обеспечивает автоматизацию работы бухгалтерии по учету оплаты договоров за обучение.

Внедрение разработанного программного продукта позволит наиболее быстро и оперативно получать нужную информацию об оплате учебы студентами, позволит повысить точность и оперативность работы с документацией, автоматизировать формирование различных отчетных документов, что значительно уменьшит временные, а соответственно и материальные затраты. Использование ЭВМ обеспечивает повышение точности и оперативности учета важной информации, освободит от выполнения дополнительных функций, таких как многократное заполнение однотипных документов и выполнения расчетов для анализа данных вручную, описания и хранение большого объема информации на бумаге. Кроме обеспечения быстрого доступа к необходимой информации, организация автоматизированных рабочих мест (АРМ) позволит своевременно обновлять и корректировать информацию. Обработка информации на ЭВМ осуществляется гораздо легче и быстрее, чем вручную, что позволяет экономить время, которое тратит работник на выполнение данной операции.

Экономическая эффективность обуславливается сокращением трудозатрат на организацию работы бухгалтерии по ведению документации и получении информации по необходимым формам.

Помимо экономии времени улучшение показателей качества работы связано со своевременным получением информации, хранящейся в БД, повышением контроля правильности ввода информации, использования более информативных и наглядных документов, сокращение рутинных вычислений при получении выходных документов.

Для выполнения поставленных целей разработанный программный продукт должен выполнять следующие функции:

* ведение договоров на обучение;
* ведение лицевых счетов студентов;
* учет оплаты за обучение;
* ведение нормативно-справочной информации;
* формирование сводной выходной информации и документации, например, получение оборотной ведомости или сведений о задолженниках;
* ведение архивов лицевых счетов и оплат.

2. Научно-исследовательский раздел

2.1 Описание объекта автоматизации

В задаче, поставленной в дипломном проектировании, в качестве объекта автоматизации выступает бухгалтерия.

Существует множество форм бухгалтерского учета. Одной из них является автоматизированная форма учета.

Автоматизированная форма учета, основанная на использовании электронно-вычислительной техники, представляет собой комплексную автоматизацию учетного процесса, начиная от сбора первичных учетных данных до получения бухгалтерской отчетности.

Обработка данных производится с помощью специальных программ, которые обеспечивают хранение, обработку информации, выдачу ее на экран или печать по запросу.

Схема автоматизированной формы учета приведена на рис. **1**.

Документы

Отчетные документы

Результирующая информация в требуемых системой учета разрезах

Машинная обработка данных

Ввод данных в ЭВМ

Рисунок 1

Одной из задач бухгалтерии ВУЗа является учет оплаты договоров за обучение.

В институте, в связи с ограниченным приемом, наряду с бесплатной формой обучения существует также и обучение на платной основе. Студенты, обучающиеся в ВУЗе на платной основе, заключают с институтом договор на обучение. По каждому студенту, с которым заключен договор, ведется лицевой счет, где отражаются все начисления и оплаты за обучение.

Сумма для оплаты начисляется за определенный период, обычно за один учебный год (с сентября по июнь). Новая сумма для оплаты начисляется через определенный период времени, например, перед началом каждого учебного года.

Для каждого студента вычисляется сальдо расчетов. Сальдо может быть дебетовым (начисления превышают оплаты) или кредитовым (оплаты превышают начисления). Сальдо рассчитывается следующим образом: при начислении новой суммы для оплаты сальдо студента увеличивается на величину этой суммы, а при оплате студентом обучения в банке сальдо уменьшается на величину оплаты.

При заключении договора студенту присваивается индивидуальный номер или код, который является уникальным для каждого студента. Этот код необходим для идентификации студента при всех расчетах.

Оплата учебы студентом осуществляется через сбербанк. Оплата за обучение может вноситься полностью (сразу) или частично. ВУЗ имеет в банке расчетный счет. Студент при оплате обучения в банке перечисляет деньги на расчетный счет ВУЗа, указывая в квитанции номер расчетного счета, а также название института и ИНН. Кроме реквизитов института в квитанции также указываются реквизиты банка, в котором находится расчетный счет института (наименование, расчетный и корреспондентский счета, БИК).

Сведения об оплатах студентов формируются ежедневно на основании выписок банка. Каждый день в институт из банка приходят выписки о движении денежных средств на расчетном счету за предыдущий день. В выписке содержится информация и о перечислениях денег студентами за обучение. В выписках указывается Ф.И.О. студента, который внес оплату, сумма оплаты, а также номер и дата выписки.

Использование вместо традиционных, бумажных методов учета электронные методы позволит проводить указанные задачи функционирования бухгалтерии более эффективно и поддерживать ее работу с меньшими затратами труда, чем при использовании ручной обработки документации. Исходя из этого, вытекает необходимость разработки системы, которая бы обеспечила бы автоматизацию учета оплаты обучения студентами при сохранении в ней всех функций, выполняемых при ведении учета бумажными методами.

Схема документооборота приведена на рис. 2.

Нормативно-справочная информация

**БУХГАЛТЕРИЯ**

ДОГОВОРА, заключенные со студентами

ВЫПИСКИ банка

Расчетный счет института

### Перечисления денег

(оплата обучения)

Выходная информация в требуемых системой учета разрезах

Рисунок 2

Нормативно-справочная информация включает данные о студентах, группах и факультетах. Сюда также входит информация, содержащая данные о самом ВУЗе (различные реквизиты, например, название и номер расчетного счета) и данные о банке, в котором находится расчетный счет ВУЗа.

Данные о студентах включают:

* код студента;
* Ф.И.О. студента;
* номер группы;
* дата окончания учебы.

Данные о группах включают следующую информацию:

* номер группы;
* факультет.

Данные о факультетах:

* краткое наименование факультета;
* полное наименование факультета.

Реквизиты института:

* краткое наименование института;
* полное наименование института:
* адрес;
* номер расчетного счета;
* ИНН института.

К нормативно-справочной информации относится кроме того Ф.И.О. ректора и Ф.И.О. главного бухгалтера.

Реквизиты банка:

* наименование банка;
* номер расчетного счета;
* номер корреспондентского счета;
* БИК.

В договорах содержится следующая информация:

* номер договора;
* дата заключения договора;
* информация о суммах, начисленных студенту для оплаты. Это дата начала и окончания периода оплаты и величина суммы.

Выписки, поступающие из банка, содержат следующую информацию:

* номер выписки;
* дата выписки;
* информация о перечислениях, сделанных на расчетный счет института. Это код и Ф.И.О. студент и сумма оплаты.

2.2 Анализ языков программирования и выбор среды разработки

Наиболее распространенными средствами разработки различных приложений в среде *Windows* в настоящее время являются *Visual Basic, Borland C++ Builder, Borland Delphi*.

Язык *Basic* вначале был разработан для создания простых программ и использовался в качестве учебного языка для изучения основ программирования с последующим переходом на более сложные и универсальные языки. Именно поэтому этот язык с ранних пор был одним из самых простых в использовании. В настоящее время последняя версия этого языка *Visual Basic 6* дает возможность решать любые современные задачи разработки приложений. При этом *Visual Basic 6* остался достаточно простым в освоении, став в то же время одним из самых распространенных языков программирования. В *Visual Basic 6* интегрирован набор инструментов, облегчающих и ускоряющих процесс разработки готового приложения. Такая реализация языка позволяет отнести его к средствам быстрой разработки приложений. Одним из направлений развития *Visual Basic* является то, что этот язык стал фактическим стандартом языка для управления всеми крупномасштабными *Windows*-приложениями. На основе *Visual Basic* созданы встроенные языки макропрограммирования для различных пакетов *Microsoft Office*. Эти языки получили общее название *Visual Basic for Application* (*VBA*). В Office единый языковый механизм *VBA* используют все основные приложения: *Word, Excel, PowerPoint, Access*. [7]

Программный продукт *Borland C++ Builder* – это средство быстрого создания корпоративных информационных систем, объединившее удобства визуальной среды разработки, объектно-ориентированный подход, разнообразные возможности повторного использования кода, открытую архитектуру и высокопроизводительный компилятор языка С++, одного из самых распространенных языков программирования. [6]

Отметим основные особенности *Borland C++ Builder* [6]:

* удобная среда разработки;
* быстрота разработки приложений;
* высокая производительность разработанного приложения за счет получения 32-разрядного исполняемого кода с помощью оптимизирующего компилятора;
* невысокие требования разработанного приложения к ресурсам компьютера;
* возможность полного доступа к функциям операционных систем *Windows*;
* возможность модернизации среды разработки за счет встраивания в нее новых компонентов, инструментов;
* возможность использования в приложениях визуальных компонентов, созданных с помощью *Borland Delphi*;
* возможность разработки новых компонентов и инструментов как собственными средствами, так и средствами *Borland Delphi*;
* архитектура доступа к данным базируется на библиотеке *BDE*, которая обеспечивает доступ к локальным БД и к *SQL*-серверам;
* возможность создавать и встраивать в приложение разнообразные отчеты.

Язык *Visual Basic* является неструктурированным языком и с его помощью достаточно сложно писать серьезные прикладные программы, он обладает ограниченными возможностями. Язык *С++* достаточно сложен в использовании и применяется в основном для решения задач системного программирования. Поэтому в дипломном проекте в качестве основного инструмента разработки клиентского приложения был выбран язык программирования *Delphi*.

Среда *Delphi* обладает с одной стороны, высокой производительностью приложений благодаря созданию полностью скомпилированного кода, удобной настраиваемой средой разработки, компонентной архитектурой, позволяющей строить приложение путем сборки его из отдельных компонентов, множество которых имеет широкое распространение, а с другой стороны – возможностью доступа к разнообразным данным, начиная от плоских таблиц типа *dBase* и *Paradox* и заканчивая разнообразными серверными СУБД. С помощью *Delphi* возможен достаточно безболезненный переход к архитектуре клиент-сервер по сравнению с другими средствами, модернизация с помощью которых была бы сопряжена с большими трудовыми и материальными затратами.

На сегодняшний день *Delphi* является одним из самых распространенных средств создания приложений баз данных для корпоративных применений. Простота и естественность языка, ориентация системы на разработку именно такого рода приложений, наконец, эффективность (большая производительность и небольшие размеры) создаваемых с ее помощью программ сделали *Delphi* незаменимым средством разработки различного рода клиентских мест, т.е. программ для доступа к базам данных. [3]

*Delphi* представляет собой 32-разрядную рабочую среду для создания 32-разрядных приложений, которые могут исполняться под управлением *Windows 95, 98* или *Windows NT*. В основу этого программного продукта легли концепции объектно-ориентированного программирования на базе языка Object Pascal и визуального подхода к построению приложений.

*Delphi* позиционируется фирмой *Inprise* как средство создания полноценных распределенных корпоративных систем доступа к данным [4].

Но в то же время среда разработки сохранила простоту и наглядность процесса создания приложений, основанного на использовании технологий визуального программирования. Компонентный подход позволяет легко и быстро создавать не только интерфейс программ, но и достаточно сложные механизмы доступа к данным, а также повторять и тиражировать удачные программные решения. *Delphi* позволяет также создавать прикладные программы за рекордно короткое время, поскольку можно оперативно поочередно выполнять и отлаживать свою программу в рамках среды разработки. Оптимизирующий компилятор *Delphi* прозрачно и быстро преобразует интерфейс и программу на исходном языке в компактную исполняемую программу на машинном языке или в библиотеку динамических связей *DLL*.

В состав поставки *Delphi* входит множество средств для разработки и эксплуатации приложений, использующих базы данных:

* машина баз данных *Borland Database Engine* (*BDE*), представляющая собой набор *dll*-библиотек, обеспечивающих низкоуровневый доступ к локальным и клиент / серверным СУБД;
* драйверы для работы с удаленными серверами данных *SQL Links*;
* утилита для установки псевдонимов баз данных, параметров и драйверов БД на конкретном компьютере;
* средство для создания, изменения и просмотра БД *Database Desktop*, ориентированное на работу с локальными СУБД;
* универсальная утилита *SQL Explorer*, позволяющая вести визуальное программирование БД с использованием данных от сервера;
* средство для трассировки выполнения *SQL*-запросов *SQL Monitor* и другие средства разработки приложений для БД.

Все это делает *Delphi* одним из самых удобных средств программирования.

2.3 Анализ СУБД и выбор БД для создания информационной модели

*Базами данных* (БД) называют электронные хранилища информации, доступ к которым осуществляется с помощью одного или нескольких компьютеров. Обычно БД создается для хранения и доступа к данным, содержащим сведения о некоторой предметной области человеческой деятельности или области реального мира. Почти все продукты, созданные с конца 70-х годов, основаны на подходе, который называют *реляционным*. Реляционный подход представляет собой основную тенденцию сегодняшнего рынка, и реляционная модель – единственная наиболее существенная разработка в истории развития БД. *Реляционная БД* – это база данных, в которой данные воспринимаются пользователем в виде таблиц. [2]

*Системы управления базами данных* (СУБД) – это программные средства, предназначенные для создания, наполнения, обновления и удаления баз данных. [3]

В зависимости от расположения различают *локальные* и *распределенные* (удаленные) СУБД.

Все части локальной СУБД размещаются на компьютере пользователя базы данных. Локальный вариант реально может обеспечить лишь однопользовательский режим доступа к данным. Этот вариант на предприятиях практически не встречается, так как в нем чрезвычайно трудно синхронизировать содержимое нескольких копий БД, количество которых (копий) должно равняться количеству одновременно работающих с данными пользователей.

До недавнего времени для организации коллективных БД использовалась, как правило, архитектура файл-сервер. В файл-серверных БД данные располагаются на сетевом файл-сервере, который может быть доступен одновременно нескольким пользователям, поэтому к таким БД возможен многопользовательский режим доступа. Данные в БД хранятся в единственном экземпляре, а каждый клиент в каждый момент времени работает с некоторой локальной копией этих данных, причем управление данными целиком возлагается на клиентские программы. Именно они должны заботиться о синхронизации локальных копий данных на каждом клиентском месте с содержимым основной (и единственной) базы данных. В ходе эксплуатации таких систем были выявлены общие недостатки файл-серверной архитектуры, которые состоят в следующем.

Вся тяжесть вычислительной работы ложится на компьютер клиента; на рабочую станцию присылаются не только данные, необходимые конечному пользователю, но и данные, которые будут использоваться для выполнения запроса и затем будут отброшены. Объем лишней информации может быть значительно больше, чем объем информации нужной. В результате возрастает загрузка сети, и увеличиваются требования к аппаратным мощностям пользовательского компьютера.

Поскольку БД представляет собой набор файлов на сетевом сервере, доступ к таблицам регулируется только сетевой операционной системой, что делает такую БД по сути беззащитной от случайного или намеренного искажения хранящейся в ней информации, уничтожения или хищения.

Бизнес-правила в системах файл-сервер реализуются в программе клиента, что в принципе не исключает проектирование противоречащих бизнес-правил в различных программах. Смысловая целостность информации при этом может нарушаться.

Недостаточно развитый аппарат транзакций локальных СУБД служит потенциальным источником ошибок как при одновременном внесении изменений в одну и ту же запись, так и при реализации отката результатов серии объединенных по смыслу в единое целое операций над БД, когда некоторые из них завершились успешно, а некоторые – нет. Это может нарушать ссылочную и смысловую целостность БД.

Перечисленных недостатков во многом лишены СУБД, построенные по архитектуре клиент-сервер.

В архитектуре клиент-сервер между клиентом и БД появляется важное промежуточное звено – сервер БД (специальная программа, управляющая базой данных)

Клиент формирует запрос к серверу на языке запросов *SQL* (*Structured Query Language* – структурированный язык запросов), являющийся промышленным стандартом для реляционных БД. *SQL*-сервер обеспечивает интерпретацию запроса, его выполнение, формирование результата и выдачу этого результата клиенту. При этом ресурсы клиентского компьютера не участвуют в физическом выполнении запроса: клиентский компьютер лишь отсылает запрос к серверной БД и получает результат, после чего интерпретирует его необходимым образом и предоставляет пользователю. Так как клиентскому компьютеру посылается результат выполнения запроса, по сети передаются только те данные, которые в действительности нужны клиенту. В итоге снижается нагрузка на сеть. Кроме того, *SQL*-сервер, если это возможно, оптимизирует полученный запрос таким образом, чтобы он был выполнен за минимально возможное время. Все это повышает быстродействие системы и снижает время ожидания результата запроса.

При выполнении запросов сервером существенно повышается степень безопасности данных, поскольку правила целостности данных определяются на сервере и являются едиными для всех приложений, использующих эту БД. В результате исключается возможность определения противоречивых правил поддержания целостности. Мощный аппарат транзакций, поддерживаемый *SQL*-серверами, блокирует одновременное изменение одних и тех е данных различными пользователями и предоставляет возможность откатов к первоначальным значениям при внесении в БД изменений, закончившихся аварийно.

Данные в серверной БД обычно физически хранятся на диске в виде одного большого файла, что в сочетании с назначаемыми каждому пользователю паролями и привилегиями существенно повышает защиту данных от намеренной порчи и хищений.

По вышеназванным причинам при разработке базы данных в дипломном проекте была выбрана архитектура клиент-сервер.

В архитектуре клиент-сервер используются промышленные серверы данных типа *Oracle, MS SQL Server, Informix, Sybase, InterBase* и ряд других [5]. Серверы *Oracle, Informix, Sybase* управляют базами данных объемом десятки и сотни гигабайт. Для управления меньшими корпоративными базами данных *MS SQL Server* и *InterBase*. Каждый тип сервера работает на собственном диалекте языка, который более или менее соответствует стандарту *ANSI 92*. Серверы *Oracle* и *InterBase* имеют некоторые дополнения. А диалекты серверов *Sybase* и *MS SQL* отличаются от стандарта довольно сильно. [4]

*Oracle* была первой коммерческой реляционной СУБД, поддерживающей ставший ныне индустриальным стандартом язык *SQL*; ее первая версия появилась в 1970 году. Ранние версии этой СУБД были предназначены для мэйнфреймов, а в качестве рабочих мест использовались терминалы. Однако со временем появились версии *Oracle*, предназначенные для использования в архитектуре клиент-сервер (первой такой версией была *Oracle 5*, выпущенная в 1985 году). Первоначально версии *Oracle* были предназначены для различных серверных платформ. Первые версии этого сервера для персональных компьютеров появились в середине 90-х годов. *Oracle* была первой компанией, создавшей СУБД, использовавшую предоставляемыми некоторыми серверными платформами средства параллельных вычислений. На сегодняшний день последней версией этого сервера является версия *Oracle 8i*, отличительными свойствами которой являются:

* наличие объектных расширений и соответствующих типов данных;
* наличие функций аналитической обработки данных;
* возможность создания таблиц, содержащих агрегатные данные;
* поддержка хранения мультимедиа-данных;
* дополнительные возможности обеспечения безопасности данных, например, шифрование данных;
* возможность создания систем, устойчивых к сбоям и ряд других.

*Oracle* производит собственные средства разработки клиентских приложений, например, *Developer/2000* или *Oracle Power Objects*, а также предоставляет пользователям возможность создавать клиентские приложения с помощью других средств. [5]

Первая версия *Microsoft SQL Server* предназначалась для платформы *OS/2*, последующие версии этого сервера баз данных предназначались для платформы *Windows NT* и со временем были тесно интегрированы с этой операционной системой. На сегодняшний день наиболее широко используемой является выпущенная в 1998 году версия *Microsoft SQL Server 7.0*. Эта версия предназначена исключительно для платформы *Windows NT*, в ее состав входят удобные утилиты администрирования, сервисы преобразования данных, поддержка распределенных запросов и транзакций, утилиты для создания хранилищ данных. Клиентские приложения для *Microsoft SQL Server* можно создавать как с помощью средств разработки *Microsoft – Visual Basic, Visual C++* и других, так и с помощью средств разработки других производителей. Для этого поставляются специальные драйверы и библиотеки. [5]

Ведущий продукт фирмы *Informix* – *Informix Dynamic Server*, последняя версия которого называется *Informix Dynamic Server.2000*. Данный продукт поддерживает платформы *UNIX* и *Microsoft SQL Server* и обеспечивает эффективную работу как на одно-, так и на многопроцессорных системах. В числе основных характеристик сервера следует отметить следующие:

* использование для управления дисковым пространством как средств ОС, так и собственных функций, что позволяет добиться более высокой производительности;
* управление разделением памяти;
* поддержка фрагментации таблиц и индексов на нескольких дисках;
* распараллеливание запросов;
* зеркалирование данных;
* двухфазное завершение транзакций и др.

Клиентские приложения могут создаваться с использованием *C, C++, Java, Visual Basic, Delphi*. Помимо этого существует собственное средство разработки *Informix 4GL* и *Informix Client Software Developer's Kit*. [5]

Одним из серверных продуктов фирмы *Sybase* является продукт под названием *Adaptive Server Enterprise*. Этот продукт существует для *Windows NT* и некоторых версий *UNIX* и предназначен для обслуживания крупных предприятий. Другим продуктом этой фирмы является *Adaptive Server Anywhere*. Этот сервер предназначен для обслуживания небольших рабочих групп, для применения в портативных компьютерах в качестве персонального сервера с периодическим копированием информации в общую БД. [5]

Для реализации архитектуры клиент / сервер в дипломном проекте был использован сервер *InterBase* компании *Inprise* (прежнее название – *Borland*). Выбор в пользу именно этого сервера был сделан по следующим причинам.

*InterBase* – «родной» сервер для Delphi и для доступа к нему не нужно устанавливать дополнительных драйверов. Продукция фирмы *Inprise* (*Borland*) давно зарекомендовала себя с положительной стороны.

*InterBase* весьма прост в установке и не требует дополнительных усилий по настройке. *InterBase* относительно прост в администрировании по сравнению с другими *SQL*-серверами. Установка программного обеспечения клиента также чрезвычайно проста.

Сервер *InterBase* использует ряд уникальных технологий. Например, активное ядро сервера реализует блокировку данных на уровне отдельных записей. Кроме того, сервер не прихотлив и практически не требует администрирования. Этот сервер обладает прекрасными функциональными возможностями [3]. Также *InterBase* работает практически со стандартным *SQL* (дополнения и изменения невелики).

За работой сервера внимательно следит утилита *InterBase Guardian*, ее пиктограмма видна на нижней панели *Windows*. Эта утилита осуществляет начальный запуск сервера, а также его перезапуск, если по каким-либо причинам сервер «рухнул». Фактически восстановление работоспособности сервера происходит за считанные доли секунды, так что многие пользователи даже, возможно, не заметят сбоя в работе. [3]

Кроме того, в вариант поставки *Delphi Enterprise* входит *InterBase 4.2* для *Windows 32*, что делает его доступным для большинства пользователей.

В *Delphi 5* для работы с данными, когда в качестве сервера баз данных задействуется *InterBase*, поддерживается технология *InterBase Express* (*IBX*), с помощью которой компоненты *IBX* связываются с сервером минуя *BDE* через программный интерфейс *IB DataBase API*, который предоставляется клиентской частью *IB DataBase*. Созданные прикладные системы будут обладать максимально возможной производительностью, способностью работать при минимально доступных ресурсах и смогут использовать специфические возможности этого сервера баз данных: сигнализаторы событий, поля BLOB, средства двухфазного завершения транзакций и т.п.

2.4 Построение математической модели

Разработанный программный продукт предназначен для использования его в архитектуре клиент / сервер. Предполагается, что сеть будет включать сервер и 3 клиентских компьютера. Рассмотрим следующую схему обработки сервером запросов от клиентских компьютеров.

РС-1

РС-3

### РС-2

С

Е

Р

В

Е

Р

Рисунок 3

От рабочих станций к серверу поступают запросы. Сервер должен обслуживать запросы, приходящие от клиентов. При чем в каждый момент времени сервер может выполнять запрос только от одной рабочей станции. Если рабочая станция при обращении к серверу застает его в этот момент свободным, то ее заявка обслуживается немедленно. В противном случае, она становится в очередь и ждет пока сервер не освободится и не выполнит ее запрос.

Для рабочих станций возможны следующие состояния:

* работа в локальном режиме;
* обмен информацией с сервером, который включает:
* обращение к серверу;
* получение информации от сервера;
* ожидание;
* ожидание в очереди.

Для сервера возможны следующие состояния:

* ожидание запросов;
* обмен информацией с одной из рабочих станций, включающий:
* прием информации от клиента;
* обработка информации для клиента;
* передача информации клиенту.

Как нетрудно убедиться, эта система обслуживания сервером клиентских компьютеров представляет собой замкнутую систему массового обслуживания (рис. **4**). Заявки (запросы от рабочих станций), обслуженные системой, опять возвращаются в источник заявок и дополняют его, вновь становясь потенциальными источниками появления заявок.

Система состоит из одного канала обслуживания (сервер) и трех источников заявок (рабочие станции). Сервер может одновременно обслуживать только одну заявку (запрос рабочей станции). На вход системы из ограниченного источника поступает поток заявок, поэтому в системе может находиться не более 3 заявок. Заявки, которые поступили в систему и застали канал обслуживания свободным, сразу же идут на обслуживание. Если же при поступлении заявки канал обслуживания занят, то заявка становится в очередь и ждет до тех пор, пока канал не освободится.

Канал обслуживания

поток заявок

поток обслуженных заявок

Время появления заявок в системе, т.е. время обращения рабочих станций к серверу, τ является случайной величиной. Примем, что поток поступающих заявок является пуассоновским. Интенсивность потока требований каждой рабочей станции равна λ0, а интервал времени между обращениями к серверу имеет показательное распределение с плотностью:

.

Интенсивность потока заявок, поступающих к серверу, зависит от того, сколько рабочих станций в данный момент работают в локальном режиме. Если в системе N потенциальных источников заявок и n из них обслуживаются в среднем в единицу времени или ожидает очереди для обслуживания (т.е. находится в системе), то плотность потока заявок в системе будет равна:

.

Тогда для нашей системы имеем:

. (1)

Время обслуживания заявки, т.е. продолжительность обработки сервером данных для рабочих станций, Θ также является случайной величиной и в общем случае ее закон распределения может быть отличен от показательного. Если считать, что Θ имеет показательное распределение:

,

то рассматриваемая замкнутая система массового обслуживания относится к классу марковских систем и ее параметры могут быть оценены аналитически.

0

3

2

1

3⋅λ0

2⋅λ0

λ0

μ

μ

μ

Рисунок 5

Изобразим процесс перехода компьютерной сети в процессе работы из одного состояния в другое в виде графа состояний (рис. **5**).

Возможные состояния системы:

1. все станции работают в локальном режиме, сервер «простаивает»;
2. одна станция обслуживается сервером, остальные работают локально;
3. одна станция обслуживается сервером, одна станция ждет обслуживания в очереди, одна станция работает локально;
4. одна станция обслуживается сервером, две другие станции ждут обслуживания в очереди.

Напишем линейные алгебраические уравнения для финальных вероятностей состояний данной системы (их существование вытекает из того, что из каждого состояния можно перейти в каждое другое, и число состояний конечно) [8]. Для состояния 0 имеем:

. (2)

Для состояния 1:

.

В силу формулы 2 это равенство приводится к виду:

.

Аналогично можно вывести формулы для остальных состояний. Тогда получим следующую систему уравнений:



Вводя в систему нормировочное условие

P0 + P1 + P2 + P3 = 1,

решим эту систему уравнений и получим выражения для нахождения финальных вероятностей.

Величина  называется интенсивностью обслуживания. Тогда получим следующие формулы:

, (3)

, (4)

, (5)

. (6)

Приведем расчетные формулы для некоторых других характеристик системы:

1. среднее число заявок в системе в единицу времени:

; (7)

1. коэффициент «простоя» сервера:

; (8)

1. среднее время между обращениями рабочих станций к серверу:

; (9)

1. среднее время пребывания заявки в системе (время ожидания рабочей станцией данных от сервера):

. (10)

Зададим численные значения интенсивностей λ0 и μ и найдем характеристики системы.

Пусть λ0 = 0.2  (τ = 5 сек.),

μ = 0.5 (Θ = 2 сек.).

Тогда .

Вычислим финальные вероятности системы (по формулам 3-6):

,

;

;



Найдем значения других характеристик системы:

* интенсивность обращения рабочих станций к серверу (по формуле 1):

.

* среднее число заявок в системе в единицу времени (по формуле 7):

;

* коэффициент «простоя» сервера (по формуле 8):



* среднее время между обращениями рабочих станций к серверу (по формуле 9):



* время ожидания рабочей станцией данных от сервера (по формуле 10):

.

Таким образом, по полученным значениям характеристик системы можно сделать вывод, что при заданных значениях λ0 и μ наиболее вероятным будет состояние, при котором сервер обслуживает одну станцию, а две другие работают локально. Среднее время ожидания рабочими станциями данных 3,33 секунды, т.е. рабочие станции ждут относительно недолго. Все это говорит о том, что сервер справляется с нагрузкой.

3. Специальный раздел

3.1 Описание информационной модели

Логическая модель базы данных:

Информационная модель объекта автоматизации включает следующие объекты:

* факультеты;
* группы;
* студенты;
* лицевые счета студентов;
* выписки банка;
* архивы.

Данные объекты можно представить в виде таблиц, связанных между собой и образующих реляционную базу данных.

В спроектированной БД используются следующие таблицы:

* справочник факультетов;
* справочник групп;
* справочник студентов;
* книга лицевых счетов;
* книга выписок банка;
* книга оплат;
* файл параметров.

С учетом требований эффективности программной реализации необходимо учитывать, что, несмотря на схожесть структуры текущих таблиц и архивов, размещать все данные о студентах, их лицевых счетах, о выписках и оплатах и архивы в одних таблицах нецелесообразно, так как со временем разросшиеся данные существенно замедлят скорость работы с таблицами. Поэтому кроме текущих таблиц, которые хранят часто используемые данные, необходимо использовать архивные таблицы, где будет содержаться устаревшая и редко используемая информация. Такими таблицами в БД являются:

* архив справочника студентов;
* архив книги лицевых счетов;
* архив книги выписок банка;
* архив книги оплат.

При проведении архивации (закрытие периода) в архив из текущих таблиц удаляются:

* студенты, для которых дата окончания учебы относится к архивируемому периоду и сальдо которых имеет нулевое значение;
* лицевые счета, относящиеся к указанным студентам;
* выписки банка, дата которых меньше даты архивации;
* все оплаты, относящиеся к удаляемым выпискам.

Так как в удаляемых выписках банка могут быть оплаты студентов, которые еще находятся в текущих таблицах, то для того, чтобы не обращаться постоянно к архивам, необходимо сохранять для каждого студента в дополнительном поле сумму оплат, которые содержатся в архиве.

Физическая модель базы данных:

Для реализации физической модели БД использовано СУБД была выбрана СУБД *InterBase*.

Таблица 1. Структура таблиц БД

|  |  |  |  |
| --- | --- | --- | --- |
| Поле | Тип | Описание | Ограничения |
| *1* | *2* | *3* | *4* |
| *Справочник факультетов S\_Facul* | | | |
| KodFacul | *VarChar(5)* | Краткое наименование факультета *(Primary Key)* | *Not Null* |
| NameFacul | *VarChar(60)* | Полное наименование факультета | *Not Null* |
| *Справочник групп S\_Group* | | | |
| NoGroup | *VarChar(5)* | Номер группы *(Primary Key)* | *Not Null* |
| KodFacul | *VarChar(5)* | Краткое наименование факультета | *Not Null* |
| *Список студентов S\_Student* | | | |
| KodStudent | *Integer* | Код студента *(Primary Key)* | *Not Null* |
| NameStudent | *VarChar(80)* | Ф.И.О. студента | *Not Null* |
| NoGroup | *VarChar(5)* | Номер группы студента |  |
| NoDogovor | *VarChar(10)* | Номер заключенного договора |  |
| DateDogovor | *Date* | Дата заключения договора |  |
| DateEnd | *Date* | Дата окончания учебы | *>DateDogovor* |
| Saldo | *Float* | Сальдо  Значение по умолчанию = 0 | *Not Null* |
| ArhOplat | *Float* | Сумма оплат, находящихся в архиве  Значение по умолчанию = 0 | *Not Null* |
| *Книга лицевых счетов Book\_Schet* | | | |
| IdSchet | *Integer* | Идентификатор счета *(Primary Key)* | *Not Null* |
| KodStudent | *Integer* | Код студента | *Not Null* |
| DateOpen | *Date* | Начальная дата периода оплаты | *Not Null*  *≥DateDogovor* |
| DateClose | *Date* | Конечная дата периода оплаты | *Not Null*  *>DateOpen*  *≤ DateEnd* |
| Summa | *Float* | Сумма для оплаты | *Not Null*  *>0* |
| Таблица 1(продолжение) | | | |
| *1* | *2* | *3* | *4* |
| *Книга выписок банка Book\_Vypis* | | | |
| IdVypis | *Integre* | Идентификатор выписки *(Primary Key)* | *Not Null* |
| NoVypis | *Integre* | Номер выписки банка | *Not Null*  *>0* |
| DateVypis | *Date* | Дата выписки | *Not Null*  *>DateArh* |
| *Книга оплат Book\_Oplat* | | | |
| IdOplat | *Integer* | Идентификатор оплаты *(Primary Key)* | *Not Null* |
| IdVypis | *Integer* | Идентификатор выписки | *Not Null* |
| KodStudent | *Integer* | Код студента | *Not Null* |
| SummaOplat | *Float* | Сумма оплаты | *Not Null*  *>0* |
| Primech | *VarChar(100)* | Примечание  Значение по умолчанию = Null |  |
| *Файл параметров Params* | | | |
| Id | *Integer* | Идентификатор *(Primary Key)* | *Not Null* |
| ShortName | *VarChar(30)* | Краткое наименование организации | *Not Null* |
| FullName | *VarChar(100)* | Полное наименование организации | *Not Null* |
| Adres | *VarChar(100)* | Адрес организации | *Not Null* |
| Rukovod | *VarChar(80)* | Руководитель | *Not Null* |
| Buhgalter | *VarChar(80)* | Главный бухгалтер | *Not Null* |
| INN | *Char(10)* | ИНН организации | *Not Null* |
| Schet | *Char(20)* | Расчетный счет организации | *Not Null* |
| BankName | *VarChar(80)* | Наименование банка | *Not Null* |
| BankKorSchet | *Char(10)* | Корреспондентский счет банка | *Not Null* |
| BankRasSchet | *Char(20)* | Расчетный счет банка | *Not Null* |
| BankBIK | *Char(20)* | БИК банка | *Not Null* |
| DateArh | *Date* | Дата последней архивации | *Not Null* |
| *Архив списка студентов Arh\_Student* | | | |
| KodStudent | *Integer* | Код студента *(Primary Key)* | *Not Null* |
| Таблица 1(продолжение) | | | |
| *1* | *2* | *3* | *4* |
| NameStudent | *VarChar(80)* | Ф.И.О. студента | *Not Null* |
| NoGroup | *VarChar(5)* | Номер группы |  |
| NoDogovor | *VarChar(10)* | Номер заключенного договора |  |
| DateDogovor | *Date* | Дата заключение договора |  |
| DateEnd | *Date* | Дата окончания учебы | *>DateDogovor* |
| *Архив книги лицевых счетов Arh\_Schet*  *Архив книги выписок банка Arh\_Vypis*  *Архив книги оплат Arh\_Oplat* | | | |
| Имеют структуру аналогичную структуре текущих таблиц. | | | |

Ограничения для столбцов, которые невозможно реализовать при создании таблиц были реализованы с помощью триггеров.

Формирование первичных ключей. Генераторы:

В InterBase отсутствует аппарат автоинкрементных полей. Вместо этого для установки уникальных значений столбцов используется аппарат генераторов.

Генератором называется хранимый на сервере БД механизм, возвращающий уникальные значения, никогда не совпадающие со значениями, выданными тем же самым генератором в прошлом.

В БД созданы генераторы для ключевых полей следующих таблиц:

* книга лицевых счетов: *Gen\_Schet*;
* книга выписок банка: *Gen\_Vypis*;
* книга оплат: *Gen\_Oplat*.

При добавлении записи с помощью оператора INSERT для присваивания значения ключевому полю необходимо непосредственно обращаться к генератору с помощью функции GEN\_ID [3]. Присваивание ключевому полю уникального значения в приложении клиента реализовано с помощью хранимых процедур *Proc\_Gen\_Book\_Schet, Proc\_Gen\_Book\_Vypis, Proc\_Gen\_Book\_Oplat* соответственно для таблиц счетов, выписок, оплат.

Поддержка ссылочной целостности:

Информация в БД должна быть однородной. Для этого используется механизм ссылочной целостности, представляющий совокупность связей между отдельными таблицами БД. Существует два вида связей между таблицами: один-к-одному и один-ко-многим. Отношение один-к-одному используется довольно редко. В разработанной базе данных таблицы объединены между собой связями типа один-ко-многим. Таблицы, участвующие в отношение один-ко-многим, называются главными и подчиненными.

В базе данных должны быть определены правила ссылочной целостности, т.е. необходимо определить, что делать при изменении или удалении записей в родительской таблице, которые имеют соответствующие записи в подчиненных таблицах. Существует два варианта определения правил: ограничить изменения в родительской таблице (RESTRICT) или каскадировать изменения в дочерней таблице (CASCADE).

В InterBase соединение дочерних и родительских полей является «жестким», т.е. нельзя реализовать каскадные воздействия при изменении записей в родительской таблице. Поэтому для реализации каскадных воздействий используются триггеры.

Таблица 2. Описание связей

|  |  |  |
| --- | --- | --- |
| **Наименование связи** | **Правило удаления** | **Правило обновления** |
| Справочник факультетов→Справочник групп | RESTRICT | CASCADE |
| Справочник групп→Список студентов | RESTRICT | CASCADE |
| Список студентов→Книга лицевых счетов | RESTRICT | CASCADE |
| Список студентов→Книга оплат | RESTRICT | CASCADE |
| Книга выписок банка→Книга оплат | CASCADE | CASCADE |
| Справочник групп→Архив списка студентов | RESTRICT | CASCADE |
| Список студентов→Архив книги оплат | RESTRICT | CASCADE |
| Архив списка студентов→Архив лицевых счетов | RESTRICT | RESTRICT |
| Архив списка студентов→Архив книги оплат | RESTRICT | RESTRICT |
| Архив книги выписок→Архив книги оплат | RESTRICT | RESTRICT |

Ведение индексов:

Важнейшей проблемой для любой БД является достижение максимальной производительности и ее сохранение при дальнейшем увеличении объемов хранимых данных. Использование индексов облегчает решение этой задачи. Индекс представляет собой часть БД, в которой содержится информация об организации данных в таблицах БД. В отличие от ключей, индексы занимают дополнительные объемы памяти (довольно значительные). Индексы создаются вместе со своей таблицей и обновляются при модификации данных. При этом обновление индекса для большой таблицы может отнимать много ресурсов, поэтому имеет смысл ограничить число индексов и создавать только те индексы, которые будут использоваться.

Индексы содержат уникальные идентификаторы записей и дополнительную информацию об организации данных. Поэтому получение данных при выполнении запроса с помощью индексов занимает значительно меньше времени [4].

По определениям первичных и вторичных ключей СУБД автоматически строит индексы.

В БД существуют следующие индексы:

*Справочник факультетов:*

* по коду факультета (PK);

*Справочник групп:*

* по номеру группы (PK);
* по коду факультета;

*Справочник студентов:*

* по коду студента (PK);
* по Ф.И.О. студента;
* по номеру группы и Ф.И.О. студента;

*Книга лицевых счетов:*

* по идентификатору счета (PK);
* по коду студента;

*Книга выписок банка:*

* по идентификатору выписки (PK);
* по номеру выписки;
* по дате выписки;

*Книга оплат:*

* по идентификатору оплаты (PK);
* по идентификатору выписки;
* по коду студента;

*Архив списка студентов:*

* по коду студента (PK);
* по Ф.И.О. студента;
* по номеру группы и Ф.И.О. студента;

*Архив книги лицевых счетов:*

* по идентификатору счета (PK);
* по коду студента;

*Архив книги выписок банка:*

* по идентификатору выписки (PK);
* по номеру выписки;
* по дате выписки;

*Архив книги оплат:*

* по идентификатору оплаты (PK);
* по идентификатору выписки;
* по коду студента.

Хранимые процедуры:

Хранимая процедура – это модуль, написанный на процедурном языке InterBase и хранящийся в БД.

Преимущества хранимых процедур:

* одна процедура может использоваться многими приложениями;
* разгрузка приложений клиента путем переноса части кода на сервер и вследствие этого – упрощение клиентских приложений;
* при изменении хранимой процедуры все изменения немедленно становятся доступны для всех клиентских приложений;
* улучшенные характеристики выполнения, связанные с тем, что хранимые процедуры выполняются сервером, в частности – уменьшенный сетевой трафик.

В БД были созданы процедуры для архивации и получения оборотной ведомости. Тексты процедур приведены в приложении 2.

Создание БД:

Вся база данных на сервере InterBase хранится в одном файле с расширением GDB. Для создания, изменения и удаления объектов базы данных используется язык SQL.

В качестве инструмента для выполнения запросов SQL используется утилита Windows ISQL. В данной утилите используются сценарии SQL – текстовые файлы с расширением SQL, содержащие группу запросов.

3.2 Описание функциональной структуры программного комплекса

В разработанном программном продукте по учету оплаты договоров за обучение реализованы следующие функции:

* ведение договоров на обучение;
* ведение лицевых счетов студентов;
* учет оплаты за обучение;
* ведение нормативно-справочной информации;
* формирование сводной выходной информации и документации, например, получение оборотной ведомости или сведений о задолженниках;
* ведение архивов.

В составе программного комплекса ведутся следующие справочники:

* справочник студентов;
* справочник групп;
* справочник факультетов.

По каждому студенту, с которым заключен договор на обучение, ведется лицевой счет, где отражаются все начисления для оплаты.

Информация об оплатах вводится ежедневно на основе выписок банка.

Периодически формируются различные отчеты: оборотная ведомость, ведомость задолженников. Кроме того, выходной информацией являются квитанции для оплаты, которые выдаются студентам.

Через определенное время проводится закрытие периода (архивация). При этом вся информация о студентах, лицевых счетах, выписках банка и оплатах переносится в архивы.

Функция по ведению нормативно-справочной информации реализована с помощью модулей *Facul.pas, Group.pas, Student.pas*. Модуль *Facul.pas* реализует просмотр и редактирование справочника факультетов. С помощью модуля *Group.pas* осуществляется просмотр и редактирование справочника групп. Модуль *Student.pas* выполняет функцию просмотра данных из списка студентов. Редактирование списка студентов осуществляется с помощью модуля *StudentDlg.pas*. К нормативно-справочной информации кроме данных о студентах, группах и факультетах относится информация как реквизиты ВУЗа и реквизиты банка. Просмотреть и отредактировать эту информацию можно с помощью модулей *Bank.pas* и *Params.pas*. В модулях *Student.pas* и *StudentDlg.pas* также реализуется функция ведения договоров на обучение.

Функция ведения лицевых счетов реализована в модуле *Schet.pas,* с помощью которого можно просмотреть и отредактировать суммы, начисленные для оплаты одному из студентов, и в модуле *Oplat.pas*, с помощью которого можно просмотреть все оплаты, относящиеся к данному студенту.

Учет оплат за обучение реализуется в модуле *Vypis.pas*, предназначенном для просмотра и регистрации выписок банка.

Функция ведения архивов осуществляется с помощью модулей *Arh.pas* (реализуется проведение архивации), *ArhStudent.pas*, *ArhSchet.pas*, *ArhOplatStud.pas*, *ArhVypis.pas* (просмотр архивов).

В программном продукте предусмотрена возможность получения различных отчетов. Формирование отчетов реализуется с помощью модулей *KvitDlg.pas*, *KvitRep.pas* (получение квитанции), *Oborot.pas*, *OborotRep.pas* (получение оборотной ведомости), *Dolg.pas*, *DolgRep.pas* (получение ведомости задолженников).

Помимо перечисленных модулей программа включает также следующие модули:

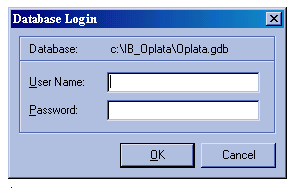
* *About.pas* – модуль, предназначенный для вывода окна «О программе»;
* *DataModule.pas* – модуль данных, который представляет собой контейнер, где расположены все невизуальные компоненты доступа к данным, такие как *TIBDataBase, TIBTransaction, TIBQuery, TDataSource*;
* *Main.pas* – модуль, который соединен с главным меню программы.

Описание всех модулей, входящих в состав проекта, содержится в файле проекта *Oplata.dpr*, который представляет собой программу, предназначенную для обработки ее компилятором Delphi.

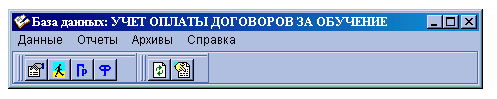
3.3 Руководство пользователя

Главное меню программы:

При запуске программы на экране появляется диалоговое окно для ввода имени пользователя и его пароля.



Если имя и пароль введены правильно, то на экране появляется основная форма, содержащая главное меню программы (рис.**10**). В противном случае на экран выводится предупреждение о том, что данные введены неправильно.



С помощью главного меню можно вызвать другие формы программы, предназначенные для просмотра и редактирования различных сведений из хранящейся на сервере БД, получения отчетов и т.п.

Вызывать различные справочники, таблицы, отчеты можно как с помощью пунктов меню, так и с помощью кнопок, расположенных в нижней части окна.

При закрытии главного окна программы осуществляется выход из программы.

Главное меню включает следующие пункты:

*Данные:*

* Выписки банка (кнопка) – для вызова книги выписок банка;



* Студенты (кнопка) – для вызова списка студентов;



* Группы (кнопка) – вызов справочника групп;



* Факультеты (кнопка) – вызов справочника факультетов;



* Выход;

*Отчеты:*

* Оборотная ведомость (кнопка) – вызов формы для получения оборотной ведомости;



* Должники (кнопка) – вызов формы для получения ведомости должников;

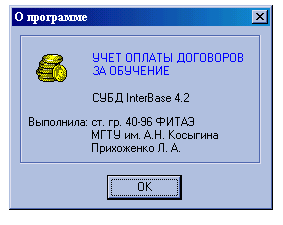


*Архивы:*

* Закрытие периода (архивация) – вызов диалогового окна для проведения процедуры архивации;
* Архив книги студентов – просмотр архива списка студентов;
* Архив книги выписок банка – просмотр архива книги выписок банка;

*Справка:*

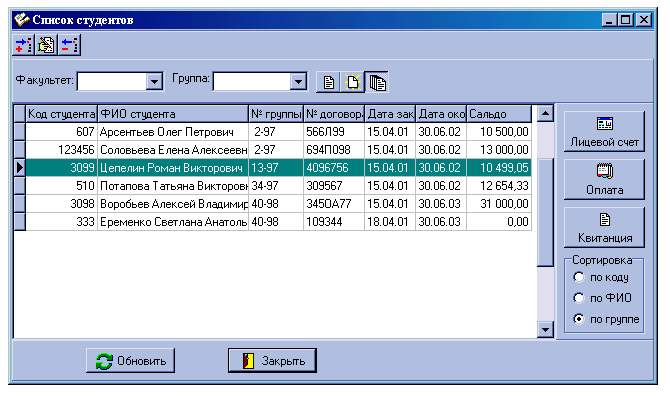
* Реквизиты предприятия – вызов диалогового окна для просмотра и редактирования информации об организации;
* Банковские реквизиты – вызов диалогового окна для просмотра и редактирования информации о банке, в котором находится расчетный счет организации;
* О программе – вызов диалогового окна «О программе».



Форма «Список студентов»:

Эта форма предназначена для просмотра данных из справочника студентов. По каждому студенту в справочнике хранится следующая информация:

* код студента;
* фамилия, имя, отчество студента;
* группа, в которую зачислен студент;
* номер договора и дата его заключения;
* дата окончания учебы;
* текущее сальдо студента (сумма денег, которую должен заплатить за учебу студент; если студент заплатил за учебу больше, чем ему было начислено, сальдо будет отрицательным).



При открытии формы на экран выводится весь список студентов, упорядоченный по группам и внутри группы по Ф.И.О. студентов. Для изменения порядка сортировки списка студентов необходимо использовать группу переключателей, расположенную в правом нижнем углу формы. Имеется возможность сортировки:

* по коду студентов в порядке увеличения кода студентов;
* по Ф.И.О. в алфавитном порядке без разбивки по группам;
* по группам и внутри группы по Ф.И.О..

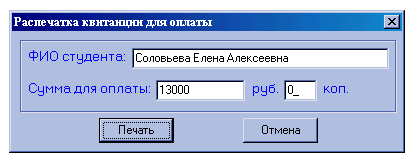
Пользователю предоставляется возможность просматривать не весь список студентов, а только студентов определенной группы. Для задания группы ее необходимо выбрать из поля со списком «Группа» в верхней части окна. Список может включать все группы или группы только заданного факультета, который задается с помощью списка «Факультет». После выбора группы необходимо нажать кнопку для отбора записей. Кроме студентов определенной группы можно отобрать тех студентов, которые еще не были зачислены ни в одну группу. Для этого необходимо нажать кнопку . То же самое произойдет, если нажать кнопку отбора записей по группе, не выбрав при этом номер группы. Для отображения всего списка студентов нажмите кнопку.



Кнопка «Обновить» предназначена для обновления данных списка студентов. При изменении данных несколькими клиентами пользователь видит только свои изменения до тех пор, пока не нажмет эту кнопку. Кроме того, осуществляется контроль одновременного изменения одних и тех же данных различными пользователями. При изменении какой-либо записи в БД каким-либо пользователем эта запись блокируется для изменений другими пользователями.

Кнопки «Лицевой счет» и «Оплата» вызывают на экран соответственно модальные окна» Лицевой счет «и» Оплаты студента». Назначение этих окон будет объяснено в соответствующих разделах.

Оплата обучения студентами происходит через банк. В квитанции для оплаты указываются реквизиты банка и ВУЗа, например, номера счетов, которые имеют большое количество знаков. А значит, при заполнении квитанции легко ошибиться. Поэтому для удобства в программе предусмотрена возможность получения квитанции. Для вызова диалогового окна, с помощью которого распечатывается квитанция, необходимо щелкнуть по кнопке «Квитанция». При этом на экране появляется окно, в котором отражаются Ф.И.О. текущего студента и сумма для оплаты. При необходимости эти данные можно отредактировать.



Кнопки «Печать» и «Отмена» этого окна нужны соответственно для распечатки квитанции или отмены.

Кнопка «Закрыть» в форме «Список студентов» приводит к закрытию окна.

Для редактирования списка студентов предназначены инструментальные кнопки, расположенные в верхней области окна. Назначение этих кнопок следующее:

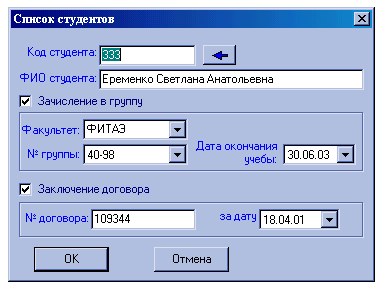
- добавление нового студента в БД;

– изменение данных текущего студента;

– удаление текущего студента из БД.

При удалении текущего студента на экран выводится диалоговое окно, в котором необходимо подтвердить удаление записи (или отказаться от него). Пользователь может удалить только тех студентов, для которых не начислены суммы для оплаты в книге лицевых счетов и не присутствует ни одной оплаты в книге оплат. При попытке удаления студента, для которого имеются указанные записи, на экран выдается предупреждение об этом и процедура удаления не происходит.

Добавление и изменение данных происходит в отдельном окне.



При добавлении нового студента все поля появляющейся на экране формы пустые. Поля необходимо заполнить необходимой информацией. Обязательно заполняемыми полями являются поля для ввода кода и Ф.И.О. студента. Код студента можно заполнят не вручную, а формировать автоматически с помощью кнопки. Полученный код будет на единицу больше максимального кода в списке студентов. Если студент зачисляется в группу, то нужно выбрать флажок «Зачисление в группу». При этом становятся доступными поля «Факультет», «№ группы», «Дата окончания учебы». Далее необходимо ввести нужную информацию. Обязательно указываются номер группы и дата окончания учебы. При выборе факультета в списке «№ группы» отображаются только те группы, которые принадлежат указанному факультету. При заключении со студентом договора об оплате обучения необходимо выбрать флажок «Заключение договора» и далее заполнить соответствующие поля.



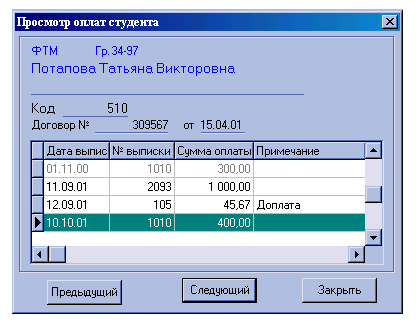
После ввода всей необходимой информации необходимо нажать кнопку «Ok», что позволяет запомнить введенную информацию, а также при этом закрывается диалоговое окно. При нажатии кнопки «Отмена» введенная информация не будет запомнена. При нажатии кнопки «Следующий» происходит добавление студента в БД, но окно не закрывается, а лишь очищаются все поля для ввода следующего студента.

При изменении текущего студента в форме (рис.**14**) отражается вся текущая информация для выбранного студента. После изменения данных студента для подтверждения изменений необходимо нажать кнопку «Ok». Нажатие кнопки «Отмена» приводит к отмене всех изменений. Кнопка «Следующий» в данном случае в форме отсутствует.

При заполнении данных на студента необходимо помнить, что для каждого студента должен заводится уникальный код. Причем код должен быть уникален не только для текущего списка студентов, но также и для архива студентов. Кроме того, дата заключения договора должна быть меньше даты окончания учебы. Если хотя бы одно из этих требований не выполнено, то на экран выводится соответствующее сообщение и пользователь не может покинуть окно, пока не изменит неправильные данные или не откажется от изменений.

Форма «Просмотр оплат студента»:

Эта форма (рис.**15**) отражает все оплаты одного выбранного студента. Данные по студенту отображаются в верхней части окна. Список оплат выводится в таблице и включает дату и номер выписки банка, сумма оплаты и примечание. При этом некоторые оплаты студента могут быть уже отправлены в архив, т.к. они принадлежат закрытому периоду. Такие оплаты выводятся не черным, а серым шрифтом.

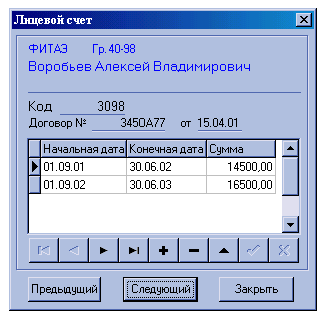


Эта форма может быть вызвана только из формы «Список студентов». Первоначально при появлении окна в нем отражаются оплаты текущего в форме «Список студентов» оплаты. Для передвижения по записям списка студентов, не закрывая окна «Оплаты», предназначены кнопки «Предыдущий» и «Следующий».

Изменять, добавлять или удалять информацию об оплатах студента в данном окне нельзя. Эти действия производятся с помощью формы «Выписки банка».

Форма «Лицевой счет»:

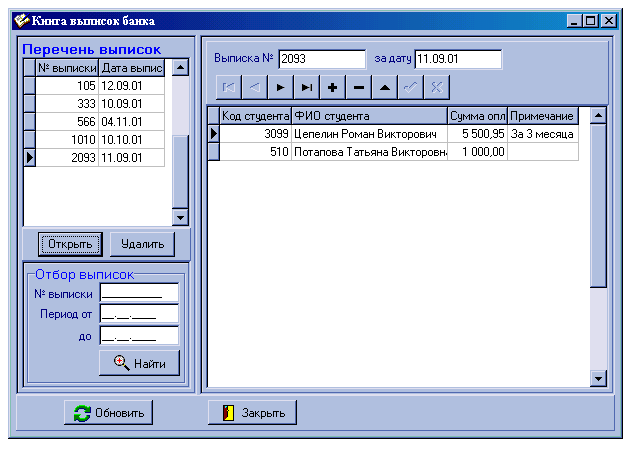
Эта форма предназначена для просмотра и редактирования данных книги лицевых счетов. Для каждого студента в книге лицевых счетов хранится информация о начисленных ему суммах для оплаты. Кроме суммы оплаты в БД хранится информация о периоде, на который начислена суммы (начальная и конечная дата периода). При вводе и редактировании информации контролируется правильность ввода дат периода: дата начала периода не может быть меньше даты окончания периода и даты не могут выходить за рамки действия договора (от даты заключения до даты окончания учебы). Если условия нарушены, то об этом выдается соответствующее сообщение. Кнопки «Следующий», «Предыдущий» и «закрыть» имеют такое же значение как и в окне «Оплаты студента».



Форма «Выписки банка»:

Форма «Выписки банка» отражает все данные, относящиеся к выпискам банка. Это номер, дата выписки и вся информация об оплатах за эту дату (код и Ф.И.О. студента, сделавшего оплату, сумма оплаты, примечание).

В левом верхнем углу формы отражается перечень выписок (№ и дата). Информацию об оплатах из этой выписки можно просмотреть, нажав на кнопку «Открыть». При этом в правой части формы отразится вся информация об оплатах по текущей выписке. Номер и дата выписки, для которой отражается информация, показаны в полях в верхней части формы. Навигатор предназначен для манипулирования с данными об оплатах.



Для удаления какой-либо выписки необходимо сделать эту выписку текущей, а затем нажать кнопку «Удалить». Вместе в выпиской будет удалена и вся информация об оплатах для этой выписки.

Для регистрации новой выписки необходимо сначала записать номер и дату этой выписки в перечень выписок, а затем нажать кнопку «Открыть» для регистрации всех оплат новой выписки.

Аналогично происходит изменение данных для какой-либо выписки. Изменение номера и даты выписки необходимо осуществлять в перечне выписок (поля в верхней части формы, где отражается номер и дата выписки, доступны только для чтения), а для изменения информации об оплатах надо открыть выписку и затем произвести необходимые изменения. При заполнении кода студента Ф.И.О. студента будет выбрано автоматически. И наоборот, если выбрать Ф.И.О. студента из списка, появляющегося в колонке «Ф.И.О. студента», то сразу же заполняется и код выбранного студента.

Кнопки «Обновить» и «Закрыть» имеют такое же значение, как и в форме «Список студентов».

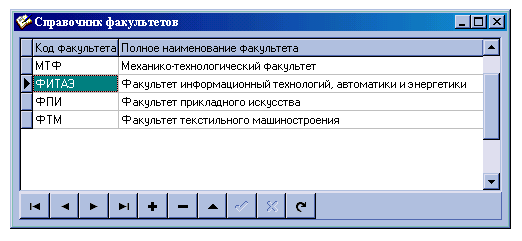
Перечень выписок банка первоначально при появлении формы отсортирован в порядке увеличения номера выписки. Для изменения порядка сортировка необходимо нажать мышкой на заголовке одной из колонок таблицы «Перечень выписок». При щелчке по заголовку «№ выписки» перечень упорядочивается по увеличению номера выписки, а при щелчке по заголовку «Дата выписки» – в порядке увеличения даты выписки.

Также в перечне выписок можно отражать не все имеющиеся выписки, а только нужные пользователю. Для этого предназначена группа «Отбор выписок». В поля необходимо внести информацию для отбора, а затем нажать кнопку «Найти». При этом будут отобраны те выписки, для которых выполняются все условия поиска. Если одно из условий не задано, то этот параметр не будет учитываться при поиске. Таким образом, если не будет задано ни одно из условий, будут отобраны все выписки. Если для периода времени задана только нижняя граница, то будут отобраны выписки, дата которых больше введенной даты. И наоборот, при вводе только верхней границы периода будут отобраны выписки дата которых меньше введенной.

Форма «Факультеты»:

Форма «Факультеты» предназначена для просмотра и редактирования справочника факультетов. В справочнике факультетов хранятся следующие данные:

* краткое название факультета (код факультета);
* полное наименование факультета.

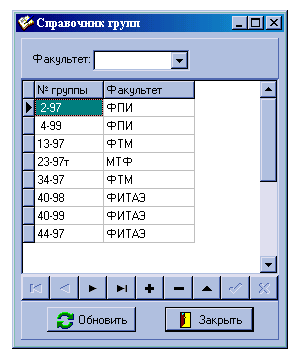


При добавлении и редактировании кода факультета необходимо следить, чтобы код был уникален. Если будет введен уже существующий код, то программа выдаст предупреждение об этом. При удалении факультета надо проверить, что ни одна из групп не принадлежит этому факультету. Попытка удаления факультета, в котором существует хотя бы одна группа, приведет к появлению сообщения о невозможности удаления факультета и факультет не будет удален.

Форма «Группы»:

Эта форма предназначена для просмотра и редактирования информации из справочника групп. Информация включают в себя следующие данные:

* номер группы;
* факультет (краткое название).



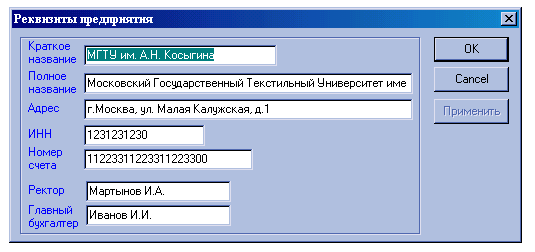
Номер группы должен быть уникальным. Программа выдаст сообщение об ошибке, если при редактировании или добавлении группы будет введена неверная информация. При добавлении и изменении группы информация о факультете выбирается из списка, который появляется в столбце «Факультет» при переходе таблицы в режим редактирования.

Удалять можно только те группы, в которые не зачислен ни один студент, иначе на экран будет выдано предупреждение о невозможности удаления группы.

При желании можно отображать не все группы, о только группы определенного факультета. Для этого необходимо выбрать факультет из списка в верхней части формы. Чтобы снова отобразить все записи, нужно выбрать строку «<все>» из того же списка.

Форма «Реквизиты предприятия»:

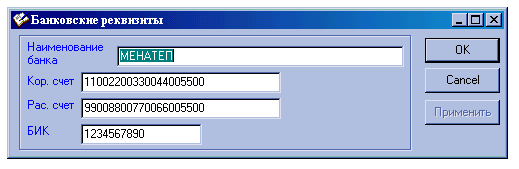
Эта форма используется для работы с постоянной и условно-постоянной информацией о предприятии. Это наиболее общая информация об организации: наименование предприятия, адрес, ИНН, номер счета, Ф.И.О. главного бухгалтера и т.п. Эта информация необходима для использования ее в различных отчетах.



При изменении информации о предприятии для подтверждения всех изменений необходимо нажать кнопку «Ok» или «Применить». Кнопка «Ok» также закрывает окно. Кнопка «Cancel» закрывает окно и отменяет все изменения.

Форма «Банковские реквизиты»:

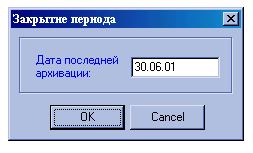
Данная форма предназначена для работы с информацией о банке, в котором находится лицевой счет организации.



Кнопки имеют такое же назначение, что и в форме «Банковские реквизиты».

Архивация:

Разросшиеся данные по студентам, лицевым счетам выпискам и оплатам будут препятствовать эффективному функционированию программы и работе пользователя. Поэтому необходимо регулярно проводить процедуру архивации (или закрытия периода). Эта процедура позволяет все записи, которые больше не используются, отправить в архив. В дальнейшем в архиве эти записи нельзя изменять, но можно будет просмотреть.



Для проведения процедуры закрытия периода необходимо в главном меню программы выбрать пункт Архивы → Закрытие периода (архивация). При этом на экране появится окно, где необходимо будет ввести дату закрытия периода, и далее нажать кнопку «Ok».

После удачного проведения архивации на экран выводится сообщение об этом. При этом в архив будут отправлены все выписки, дата которых меньше введенной даты архивации, вместе с информацией об оплатах, а также все студенты, у которых дата окончания учебы меньше введенной и сальдо нулевое.

Отчеты:

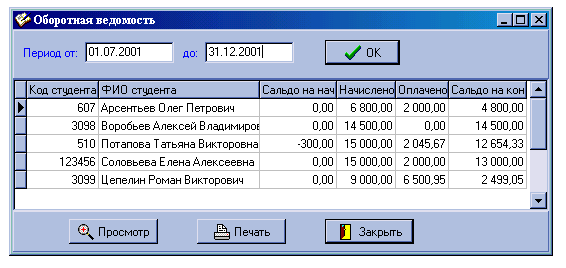
В программе имеется возможность получения следующих отчетов:

* оборотная ведомость;
* ведомость задолженников.

Для получения отчета надо в пункте «Отчеты» главного меню выбрать необходимую ведомость.

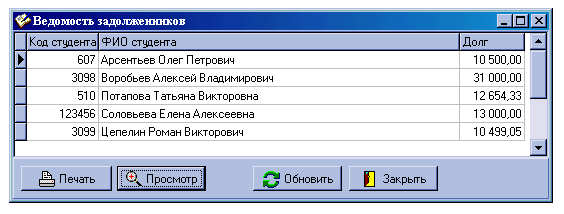
Формы отчетов представлены в приложении 1.

Для формирования оборотной ведомости необходимо в появившемся на экране окне ввести границы периода, для которого будет создана ведомость, и нажать кнопку «Ok». Кнопки «Просмотр» и «Печать» служат соответственно для предварительного просмотра и печати отчета.



Необходимо помнить, что составить оборотную ведомость можно только для периода, который не относится к закрытому периоду, т. к. данные для этого периода находятся в архиве.

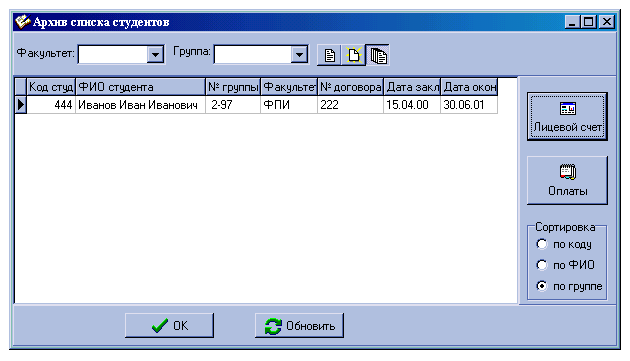
При выборе ведомости задолженников на экране появляется окно, где отражаются данные, которые будут выводиться в отчете.



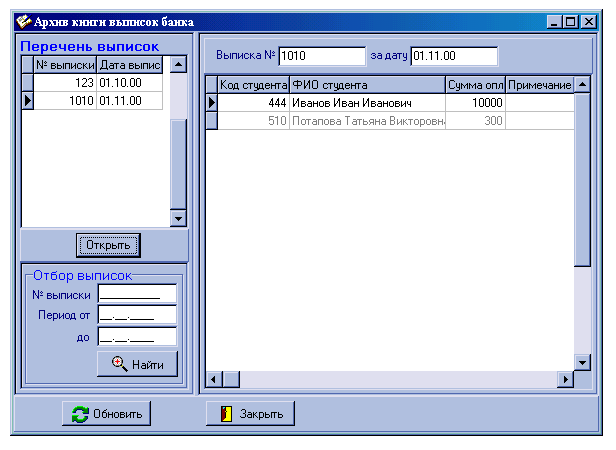
Кнопки имеют такое же назначение, как и в форме «Оборотная ведомость»

Форма «Архив списка студентов»:

Форма «Архив списка студентов» (рис.**25**) служит для просмотра всех данных о студентах, их лицевых счетах и оплатах из архива. Назначение кнопок, полей со список такое же, как и в форме «Список студентов». При нажатии на кнопки «Лицевой счет» и «Оплаты» на экране появляются соответственно формы «Архив лицевых счетов» и «Архив книги оплат», которые выглядят также как и формы для соответствующих текущих таблиц.



Форма «Архив книги выписок банка»:



Форма «Архив книги выписок банка» (рис.**26**) отражает всю информацию о выписках и оплатах, которые находятся в архиве. Назначение кнопок аналогично кнопкам формы «Книга выписок банка».

В архиве могут храниться оплаты не только тех студентов, данные на которых уже отправлены в архив, но и студентов, которые хранятся в текущем списке студентов. Поэтому в «Архиве книги выписок банка» оплаты для студентов, которые содержатся в текущих таблицах, выводятся не черным, а серым шрифтом.

Работа с навигатором:

Навигатор представляет собой совокупность управляющих кнопок и выполняет операции перемещения по набору данных и модификации записей целиком.



Назначение кнопок в порядке их следования следующее:

* перемещение на первую запись;
* перемещение на предыдущую запись;
* перемещение на следующую запись;
* перемещение на последнюю запись;
* добавление новой записи в текущей позиции;
* удаление текущей записи;
* переход в режим редактирования;
* сохранение всех изменений;
* отмена изменений, которые не были сохранены;
* обновление набора данных (присутствует только в справочнике факультетов).

3.3 Руководство программиста

Инструкция по установке.

Разработанная в рамках дипломного проекта программа по учету оплаты договоров за обучение для клиентских мест может исполняться только под управлением операционных систем Windows 95, 98 или Windows NT.

На компьютере, который будет использоваться в качестве сервера, т.е. где будет располагаться файл базы данных, должно быть установлено программное обеспечение для сервера InterBase. К моменту обращения клиентской программы к базе данных сервер InterBase должен быть запущен. Если на компьютере установлен InterBase, его старт происходит автоматически при загрузке операционной системы. Об этом сигнализирует пиктограмма справа на нижней панели Windows. Щелкнув на этом значке правой кнопкой мыши, можно вызвать вспомогательное меню. Опция Startup Configuration этого меню позволяет просмотреть и переопределить стартовые установки InterBase. Опция Shutdown завершает работу SQL-сервера. Опция Properties позволяет просматривать свойства InterBase и текущей сессии, например, число активных подключений и число используемых БД.

Связь программы для клиентских мест с базой данных происходит напрямую через программный интерфейс IB DataBase API, минуя BDE. Поэтому на компьютере клиента установка этих библиотек не требуется.

Для обслуживания сервера Interbase предназначена утилита InterBase Windows Interactive SQL (WISQL). В отличие от других утилит, WISQL использует собственный механизм доступа к базе данных, поэтому может работать на машинах без BDE и не использует псевдонимы BDE.

При доступе к локальному серверу с помощью утилиты WISQL указывается только путь к БД.

При работе с удаленным (сетевым) сервером необходимо «прописать» его на компьютере, с которого происходит обращение к серверу. При использовании протокола TCP/IP:

1. IP-адрес сервера и имя должны быть описаны в файле HOSTS, например: 10.12.0.41 spv.
2. протокол доступа к InterBase должен быть описан в файле SERVICE: gds\_db 3050/tcp.

Оба указанных файла находятся в каталоге WINDOWS.

Кроме того, для правильной работы с программой необходимо указать путь к файлу базы данных InterBase. Путь прописывается в файле path.txt, который должен находится в том же каталоге, что и сама программа, т.е. exe-файл.

Системные требования:

* Pentium 133;
* 16 Mb RAM;
* Windows 95.

Разграничение доступа к данным.

Для архитектуры клиент / сервер безопасность данных обеспечивается на двух уровнях – сервера и приложения, так как пользователь может не только посылать на сервер команды модификации данных, но и получать для дальнейшей работы наборы данных на сторону клиента.

В системе защиты сервера можно выделить следующие основные элементы:

* хранение списков пользователей, их паролей и прав доступа;
* организация регистрации пользователей с проверкой правильности паролей и установкой прав доступа к различным категориям данных;
* определение групп доступных пользователю операций:
* только просмотр данных;
* редактирование данных;
* добавление данных;
* создание специальных инструментов управления данными (просмотры, хранимые процедуры, триггеры).

При установлении соединения с БД появляется диалоговое окно для ввода имени пользователя и пароля. Результаты регистрации отправляются на сервер. Сервер отслеживает уровни доступа для каждого пользователя и блокирует попытки неразрешенных для пользователя действий.

На уровне приложения разграничение доступа обеспечивается контролем за выполняемыми пользователем операциями. В зависимости от приоритета пользователя ему могут быть доступны только операции просмотра данных, редактирования данных или ввода новых данных. В созданной программе права доступа отслеживаются с помощью управления свойствами элементов форм.

Владельцем созданной БД *oplata.gdb* является пользователь *SYSDBA* (пароль «masterkey»), т.е. системный администратор. Системный администратор обладает полным набором прав доступа к БД, т.е. может изменять, добавлять, удалять данные, просматривать данные, а также модифицировать структуру БД. Кроме системного администратора к БД имеет доступ также пользователь с именем *USROPL* (пароль «1234»). Права пользователя *USROPL* приведены в таблице.

Таблица 3. Права доступа к БД пользователя USROPL

|  |  |
| --- | --- |
| Таблицы, процедуры | Права доступа |
| Справочник факультетов S\_Facul | Select, Insert, Update, Delete |
| Справочник групп S\_Group | Select, Insert, Update, Delete |
| Справочник студентов S\_Student | Select, Insert, Update, Delete |
| Книга лицевых счетов Book\_Schet | Select, Insert, Update, Delete |
| Книга выписок банка Book\_Vypis | Select, Insert, Update, Delete |
| Книга оплат Book\_Oplat | Select, Insert, Update, Delete |
| Файл параметров Params | Select, Update |
| Архив списка студентов Arh\_Student | Select, Insert, Update |
| Архив книги лицевых счетов Arh\_Schet | Select, Insert |
| Архив книги выписок банка Arh\_Vypis | Select, Insert |
| Архив книги оплат Arh\_Oplat | Select, Insert, Update |
| Процедура архивации Arhivation | Execute |
| Процедура получения оборотной ведомости OborotVed | Execute |
| Процедура получения значения генератора Gen\_Schet Proc\_Gen\_Book\_Schet | Execute |
| Процедура получения значения генератора Gen\_Vypis Proc\_Gen\_Book\_Vypis | Execute |
| Процедура получения значения генератора Gen\_Oplat Proc\_Gen\_Book\_Oplat | Execute |

Пользователь *SYSDBA* создается СУБД автоматически. Пользователя *USROPL* необходимо создать с помощью утилиты InterBase Server Manager. Эта утилита присутствует в стандартной поставке InterBase. Кроме управления пользователями и их паролями, она предназначена для определения параметров сервера, сохранения и восстановления БД, сборки «мусора» и т.д.