Реферат

Пояснительная записка - 92 страницы, 15 таблиц, 30 рисунков, 17 источников, 12 приложений.

Ключевые слова - база данных, информационная подсистема, технический контроль, справочная система.

Объект исследования и разработки - предприятие ООО «БЮНА».

Целью данной дипломной работы является разработка автоматизированной подсистемы работы персонала отдела технического контроля.

Полученные результаты и их новизна - программа «Информационная подсистема поддержки деятельности отдела технического контроля», предназначенная для создания базы данных внутри отдела и создания текущих документов с использованием базы.

Область применения - предназначена для помощи сотрудникам отдела технического контроля при выполнении их непосредственных обязанностей.

ОГЛАВЛЕНИЕ

Введение

Определяющее значение в условиях современной экономики приобрел контроль качества выпускаемой продукции. Производители должны уделять существенное значение обеспечению высокого качества продукции, устанавливая контроль на всех этапах производственного процесса, начиная с контроля качества используемых сырья и материалов и заканчивая устанавливая соответствие выпущенной продукции техническим характеристикам и параметрам не только в ходе его испытаний, но и в эксплуатации, а для высокотехнологичных видов оборудования - с предоставлением определенного гарантийного срока. Контроль качества продукции направлен скорее не на выявление дефекта в готовой продукции, а на проверку качества изделия во время изготовления. Такой подход к контролю подразумевает проведение испытаний по мере готовности отдельных частей продукции. Контроль - это действие, включающее проведение испытаний, измерений одной или нескольких характеристик продукции или услуги и их сравненительные оценки с установленными требованиями с целью уточнения идентификации.

Контроль качества продукции является частью производственного процесса, направленой на проверку надежности в процессе ее производства или эксплуатации. Недостаточный контроль на любом из этапов изготовления серийной продукции приводит к появлению финансовых проблем и влечет за собой дополнительные финансовые затраты.

Объект исследования - предприятие ООО «БЮНА».

Целью данной дипломной работы является разработка автоматизированной подсистемы работы персонала отдела технического контроля.

Для достижения цели исследования необходимо решить следующие задачи: проанализировать структуру и направление деятельности предприятия; проанализировать возможности адаптации баз данных к специфике работы

ОТК, построение информационной подсистемы с предоставлением инструкции пользователя.

1. Общие сведения о предприятии ООО «БЮНА»

1.1 Группа компаний ООО «БЮНА»

Начало создания Группы компаний ООО «БЮНА» было положено в июле 1976 года. Тогда в городе Павловске Воронежской области начал работу Павловский ГОК. Сегодня ООО «БЮНА», это специализирующееся на добыче и производстве гранитного щебня. Данное предприятие является основой Группы компаний ООО «БЮНА». Чтобы идти в ногу со временем, предприятию потребовалось охватить весь производственный цикл, связанный с добычей, производством, реализацией, а также конечным использованием гранитного щебня. Так с 2000-го года стали появляться предприятия Группы компании ООО «БЮНА».

Группа компаний ООО «БЮНА» создана для реализации проектов в сферах промышленного, гражданского, жилищного, дорожного строительства и производства строительных материалов. Объединяет 16 предприятий в центральной части России. Общая численность работающих - 5200 чел. Партнеры предприятий ГК ООО «БЮНА» находятся в 35 областях Северо-Западного, Южного, Центрального, Приволжского федеральных округов. Базовое предприятие: ООО «БЮНА» - крупнейшее в Европе по добыче и производству нерудных материалов (основной вид продукции - гранитный щебень, проектная мощность - 11000 тыс. кубометров в год).

В Группе компаний ООО «БЮНА» выделены предприятия производства строительных материалов, предприятия дорожно-строительного комплекса и предприятия жилищного, гражданского и промышленного строительства.

ООО «БЮНА» стал одним из основных поставщиков гранита при строительстве федеральной трассы М-4 «Дон». Магистраль реконструируют в соответствии с международными стандартами к олимпиаде в Сочи в 2014 году. В 2010 году на проходящий через Воронежскую область участок трассы выделили более 5 миллиардов рублей.

Компания ООО «БЮНА» расположена в городе Воронеж. Предприятие основано в 1976 году на базе крупнейшего в Европе Шкурлатовского месторождения гранитов. Запасы месторождения в объеме 613,5 миллионов кубических метров утверждены ГКЗ СССР протоколом № 10137 от 4.03.1987 г. Добыто за 35 лет около 97 миллионов кубических метров. Размеры карьерного поля на сегодняшний день составляют: длина - 2,1 км, ширина - 1,5 км, глубина — 140 м. Утвержденные запасы гранита, при нынешнем уровне добычи, позволят вести разработку как минимум до окончания XXI века. Структура предприятия представлена на рисунке 1.1.

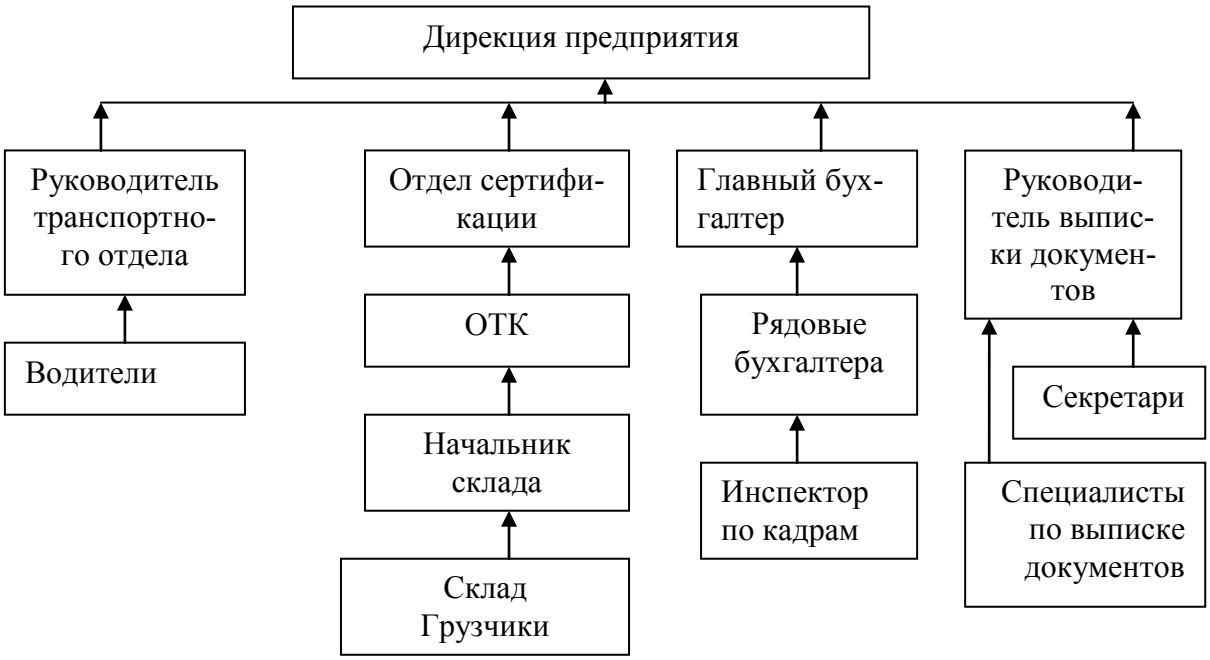

Рисунок 1.1- Общая организационная структура ООО «БЮНА»

Основное производство ООО «БЮНА» представлено на рисунке 1.2.

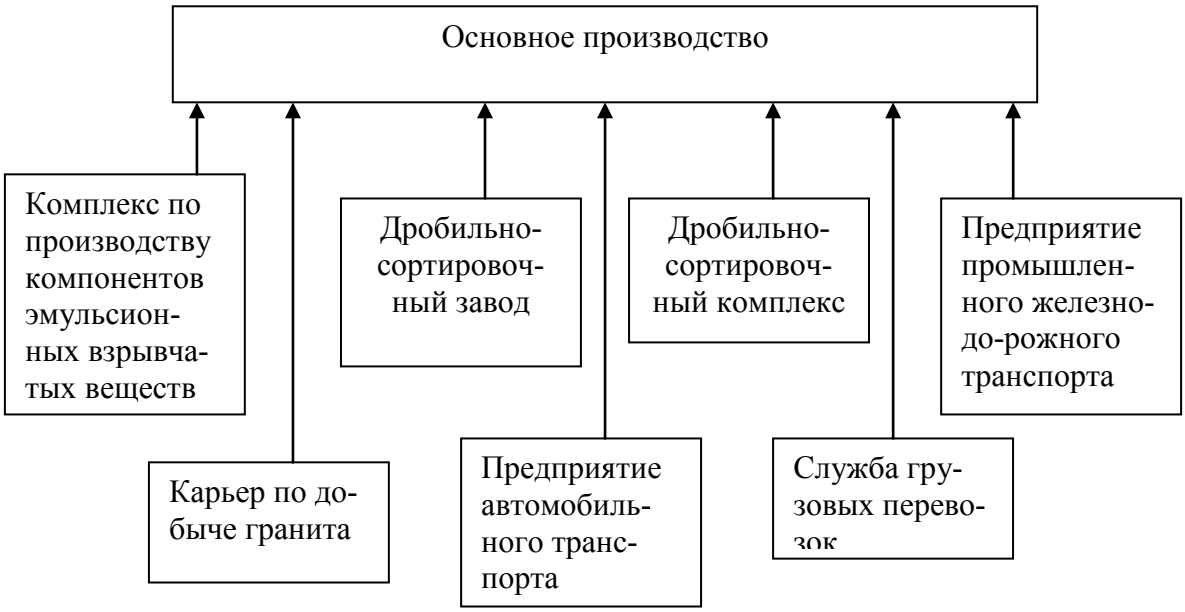

Рисунок 1.2 - Основное производство ООО «БЮНА»

Проектная мощность предприятия по выпуску нерудных материалов была определена в размере 10 800 тыс. тонн нерудных материалов в год. После проведения комплекса мероприятий, направленных на модернизацию и реконструкцию существующей технологии выпуска нерудных материалов. Количество основного технологического оборудования - 1 250 ед. Общая численность работающих - 3 003 человека Основные потребители продукции представлены на рисунке 1.3.

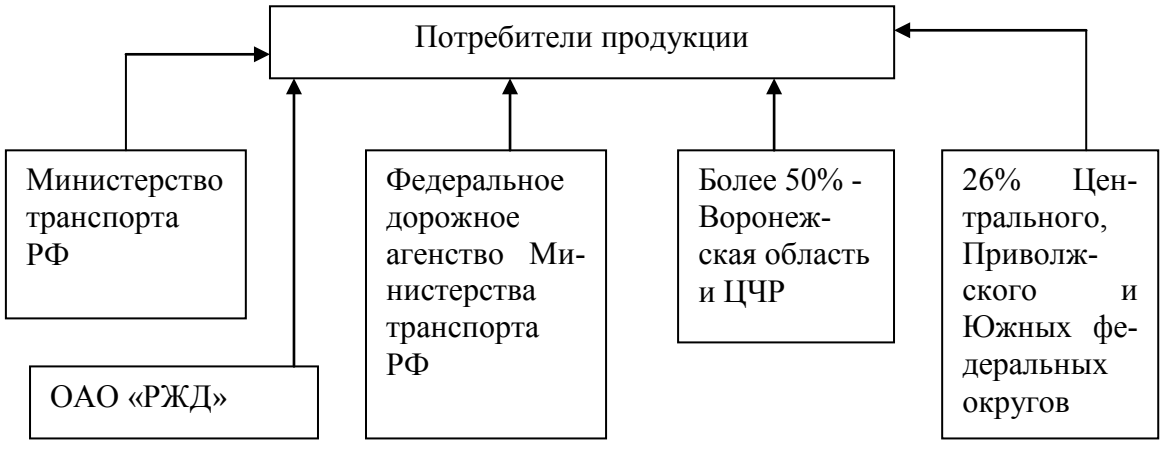

Рисунок 1.3 - Основные потребители продукции

Для осуществления своей деятельности ООО «БЮНА» оформило 21 лицензию, в том числе лицензии на:

— открытая разработка Шкурлатовского месторождения гранитов;

— производство взрывчатых материалов промышленного назначения;

— применение взрывчатых материалов промышленного назначения;

— эксплуатация взрывоопасных производственных объектов.

Добыча руды в карьере ООО «БЮНА» представлены на рисунке 1.4.

Рисунок 1.4 - Схема добычных работ на предприятии

Технология переработки гранита представлена на рисунках 1.5, 1.5.1, 1.5.2 и 1.5.3.

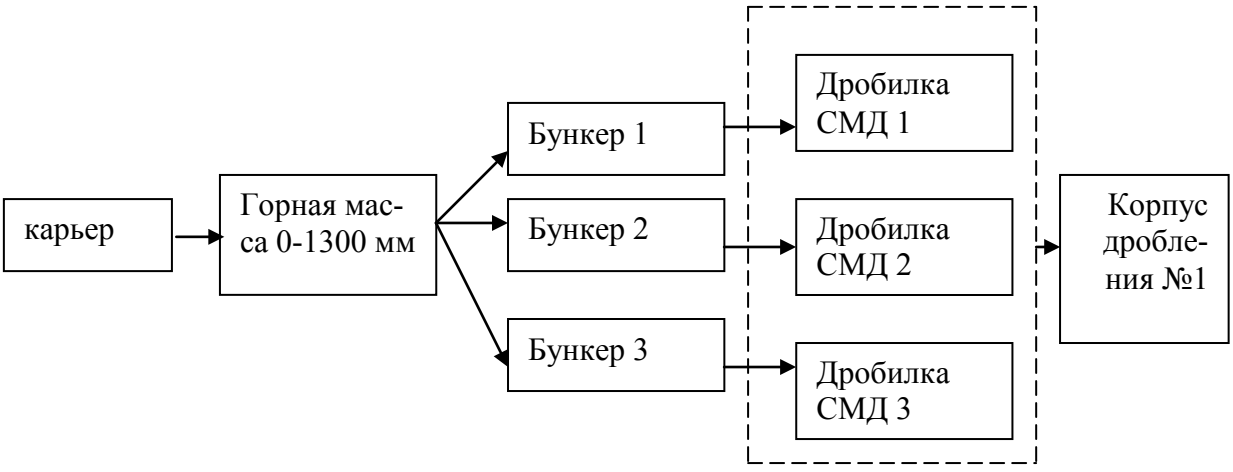

Рисунок 1.5 - Первый этап переработки гранита

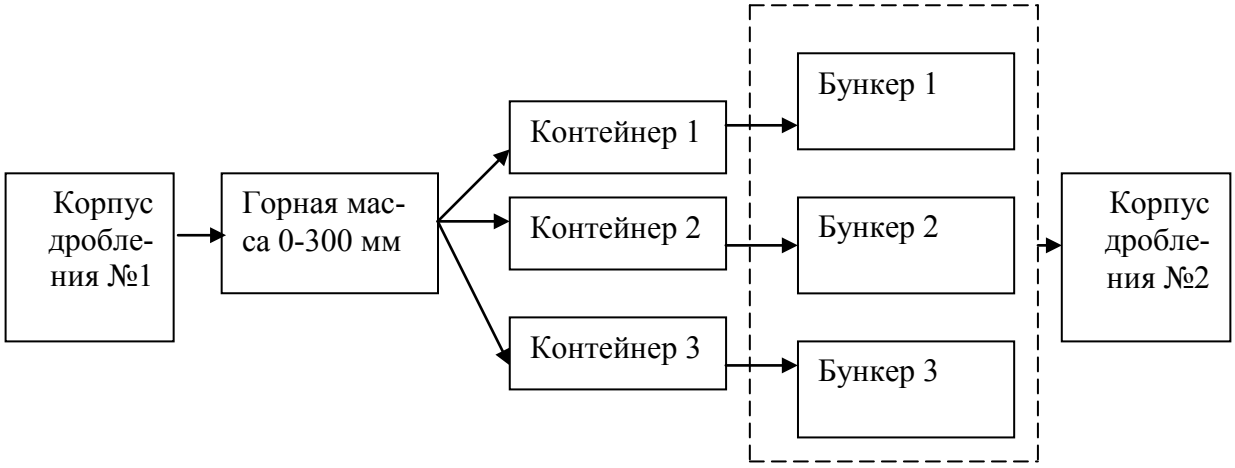

Рисунок 1.5.1 - Второй этап переработки гранита

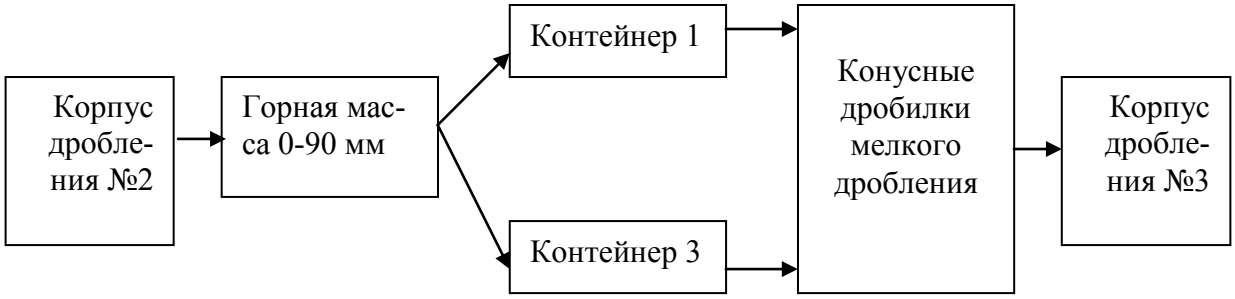

Рисунок 1.5.2 - Третий этап переработки гранита

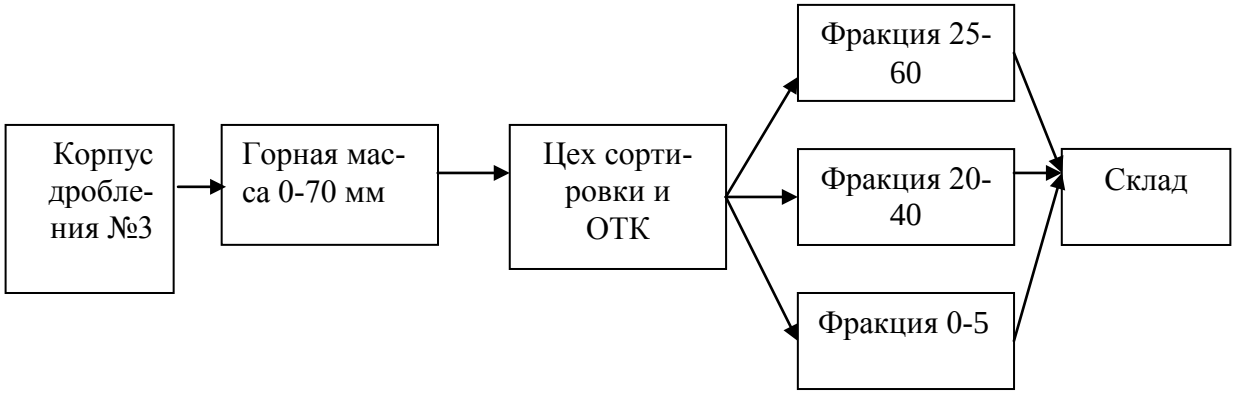

Рисунок 1.5.3 - Финальный этап переработки гранита

Гранит - кислая магматическая интрузивная горная порода. Состоит из кварца, плагиоклаза, калиевого полевого шпата и слюд - биотита и/или мусковита. Эффузивные аналоги гранитов - риолиты. Плотность гранита - 2600 кг/м^3, прочность на сжатие до 300 МПа.

Месторождение оценено комплексно. Отходы дробления пригодны для производства песка в соответствии с ТУ 571100 - 05169287 - 99, «Песок из отсевов дробления гранитов Шкурлатовского месторождения для дорожного строительства».

Для разработки, внедрения и поддержания в рабочем состоянии системы менеджмента качества (далее - СМК), постоянного улучшения ее результативности предприятие:

- выделило процессы, необходимые для СМК, и их применение во всей организации;

- определила последовательность и взаимодействие этих процессов;

определило критерии и методы, необходимые для обеспечения результативности, как при осуществлении, так и при управлении этими процессами;

- обеспечивает наличие ресурсов и информации, необходимых для поддержки этих процессов и их мониторинга;

- осуществляет мониторинг, измерение и анализ этих процессов;

- принимает меры, необходимые для достижения запланированных результатов и постоянного улучшения этих процессов.

Документы СМК позволяют распределить ответственность, права, обязанности, установить порядок взаимодействия подразделений при выполнении своих функций по обеспечению качества выполняемых работ.

Общая стратегия ООО «БЮНА» в вопросах качества определена Политикой в области качества.

В рамках требований по постоянному совершенствованию СМК, Политика в области качества анализируется исполнительным директором ООО «БЮНА» на постоянную пригодность на основании данных

мониторинга рынка и требований заказчика и актуализируется по мере накопления информации и данных один раз в год.

Согласуясь с принятой в ООО «БЮНА» политикой в области качества, исходя из условия непрерывного улучшения и совершенствования, устанавливаются Цели в области качества, к которым стремится ООО «БЮНА» в ближайшем году.

Отдел технического контроля располагает штатом и оборудованием, необходимым для контроля и производства анализов в соответствии с ГОСТами 8 267-93 «Щебень и гравий из плотных горных пород для строительных работ» - технические условия, 8 269-87 «Щебень из природного камня для строительных работ» - методы испытаний, 25 607-94 «Смеси щебёночно-гравийно-песчаные для покрытий и оснований автомобильных дорог и аэродромов» - технические условия.

1.2 Отдел технического контроля

Для определения качества исходного сырья и готовой продукции, контроль технологического процесса в ООО «БЮНА» предусмотрен одел технического контроля входящей в структуру ДСЗ, лаборатория располагается возле корпуса перегрузок ДСЗ. Отдел технического контроля располагает штатом и оборудованием, необходимым для контроля и производства анализов в соответствии с ГОСТами 8 267-93 «Щебень и гравий из плотных горных пород для строительных работ» - технические условия, 8 269-87 «Щебень из природного камня для строительных работ» - методы испытаний, 25 607-94 «Смеси щебёночно-гравийно-песчаные для покрытий и оснований автомобильных дорог и аэродромов» - технические условия. Отдел технического контроля контролирует соблюдение всего технологического регламента дробления, переработки и обогащения горной массы на всех трёх стадиях дробления и сортировки ДСЗ.

Отдел технического контроля является самостоятельным структурным подразделением в составе дробильно-сортировочного завода (ДСЗ) непосредственно подчиняется главному технологу - зам. директора ДСЗ, а функционально директору по производству и главному инженеру ООО «БЮНА», руководствуется законами, указами, постановлениями и другими нормативными документами высших органов власти РФ, определяющими и регулирующими деятельность ООО «БЮНА»; уставом компании и положением об ОТК. Структура ОТК представлена на рисунке 1.6.

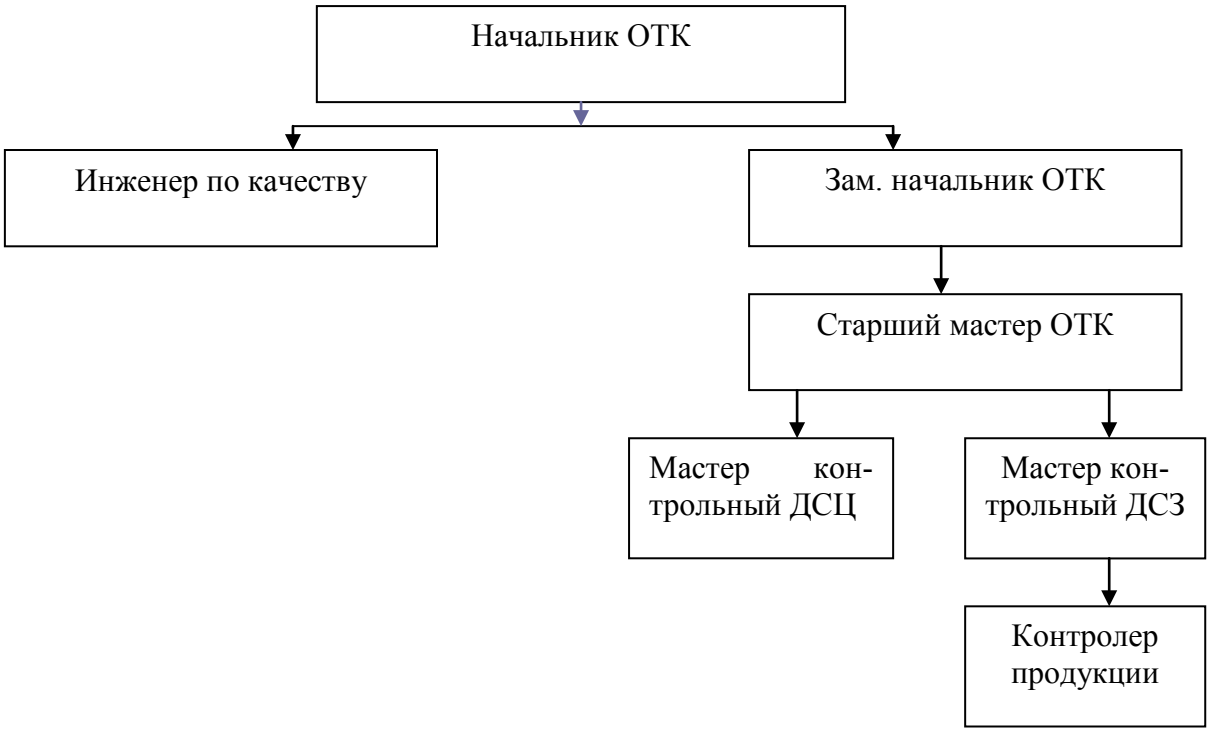

Рисунок 1.6 - Структура ОТК

Задачи и функции ОТК:

1. Предотвращение выпуска продукции, не соответствующей требованиям нормативно-технической документации (НТД), условиям поставки и договоров.

2. Укрепление производственной и трудовой дисциплины и повышение ответственности всех звеньев производства за качество

выпускаемой продукции, за сохранность имущества и материальных ценностей.

3. Контроль за качеством продукции, за соответствием требованиям ее НТД.

4. Оформление документации на принятую и отгруженную продукцию.

5. Анализ и учёт продукции не соответствующей требованиям НТД.

6. Участие в разработке и контроль за осуществлением мероприятий, направленных на предупреждение выпуска некачественной продукции.

Ответственность ОТК:

1. Контроль качества продукции, сырья;
2. Соблюдение технических режимов;
3. Качество и достоверность лабораторных испытаний;
4. Правильность оценки качества продукции;
5. Ведение технической документации по проведению испытаний;
6. Оформление паспортов, удостоверяющих качество продукции;
7. Выпуск продукции, несоответствующей требованиям ГОСТ и ТУ.

Взаимоотношения ОТК с другими цехами:

1. Цех дробления
2. Цех сортировки
3. Цех погрузки
4. Карьер

Все виды технического контроля, проводимые отделом технического контроля можно представить в виде древовидной схемы на рисунке 1.7.

Аккредитацию лаборатории отдела технического контроля проводит федеральное государственное учреждение «Воронежский центр стандартизации, метрологии и сертификации» один раз в три года. По результатам

проверки составляется акт оценки состояния измерений в отделе технического контроля компании.

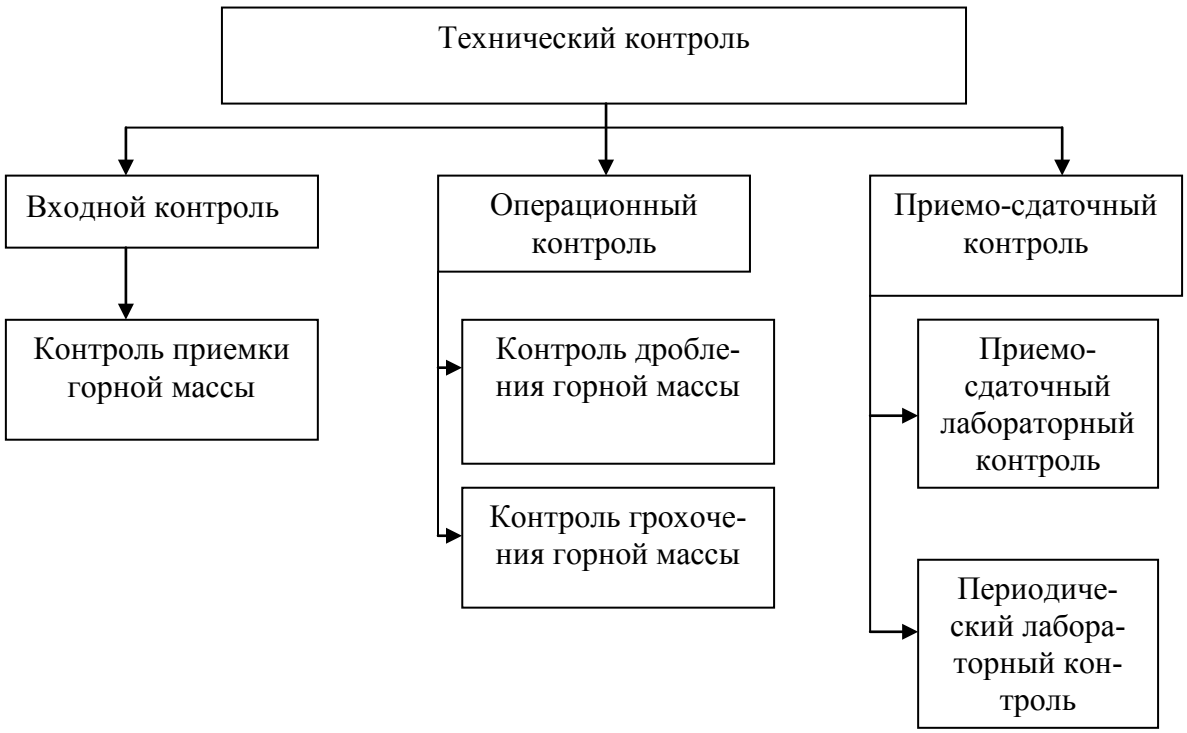

Рисунок 1.7 - Виды контроля

1.3 Цель и задачи дипломного проектирования

Целью выпускной работы является разработка подсистемы поддержки деятельности отдела технического контроля ООО «БЮНА».

Для достижения поставленной цели необходимо решить следующие задачи:

- Провести анализ методов и средств построения информационных систем ориентированных на использование в ОТК;

- Разработать информационное обеспечение на основе инфологического и даталогического проектирования баз данных;

- Осуществить проектирование компонентов информационной системы с использованием функциональных диаграмм (IDEF0);

- Разработать программное обеспечение информационной системы на основе объектно-ориентированного программирования.

Провести апробацию результатов дипломного проектирования в ОТК ООО «БЮНА».

2 Разработка информационной подсистемы поддержки деятельности отдела технического контроля.

2.1 Формирование информационных моделей на основе концепции баз данных

В современном производственно-экономическом пространстве без баз данных представить функционирование большинства финансовых, промышленных, торговых и прочих фирм весьма нелегко. Базы данных позволяют представить информацию структурировано, а также хранить и извлекать оптимальным образом для пользователей.

Обычно база данных строятся для хранения и доступа к данным, имеющим сведения о некоторой предметной области, то есть области деятельности человека. Любая база данных должна представлять систему данных об анализируемой области. Базы данных относящиеся к одной и той же области, в различных ситуациях содержат более или менее детализированную информацию. Степень детализации устанавливается несколькими факторами, прежде всего целью применения информации из базы данных и сложностью производственных процессов, существующих в пределах анализируемой области в конкретной ситуации.

Банк данных - это система специальным образом сгруппированных данных, а именно базы данных, программные, технические, языковые, организационно-методические средства, предназначенные для обеспечения централизованного аккумулирования и коллективного многоцелевого использования информации [6].

База данных (БД) - именованная совокупность данных, соответствующая состоянию объектов и их отношений в анализируемой предметной области.

Система управления базами данных (СУБД) - совокупность языковых и программных средств, предназначенных для создания, ведения и совместного использования БД многими пользователями [5].

Программы, позволяющие пользователям работать с базой данных, называют приложениями. В общем случае с одной базой данных может работать несколько различных приложений. Например, если база данных моделирует некоторую фирму, то для работы с ней можно создать приложение, позволяющее обслуживать подсистему кадрового учета, другое приложение позволяет оптимально сформировать работу подсистемы расчета заработной платы сотрудников, третье приложение разработано как подсистема складского учета, четвертое приложение занимается планированием производственного процесса. При анализе приложений, работающих с одной базой данных, предполагаем, что они работают параллельно и независимо друг от друга, и именно СУБД обеспечивает работу нескольких приложений с единой базой данных таким образом, чтобы каждое из них функционировало корректно, с учетом изменений базы данных, вносимых другими приложениями.

Непосредственно термины «база данных» и «банк данных», а также и терминология в СУБД, заимствованы из финансовой деятельности. Это заимствование носит закономерный характер и объясняется тем, что работа с информацией и работа с денежными массами во многом аналогичны, так как и в первом и во втором случаях отсутствует персонификация объекта обработки: две купюры номиналом в пятьдесят рублей так же неотличимы и взаимозаменяемы, как два одинаковых байта (конечно, за исключением серийных номеров). Деньги можно положить на некоторый счет и предоставить возможность вторым лица их для других целей. Можно поручить банку оплатить ваши траты с вашего счета или получить их

наличными в другом банке, и это будут другие денежные купюры, но их ценность будет эквивалентна исходной, которую вы имели, когда клали наличность на свой счет.

В ходе научных исследований, посвященных вопросу анализа СУБД, предлагались различные способы реализации. Самым жизнеспособным из них оказалась предложенная американским комитетом по стандартизации ANSI (American National Standards Institute) трехуровневая система организации БД, изображенная на рисунке 2.1.

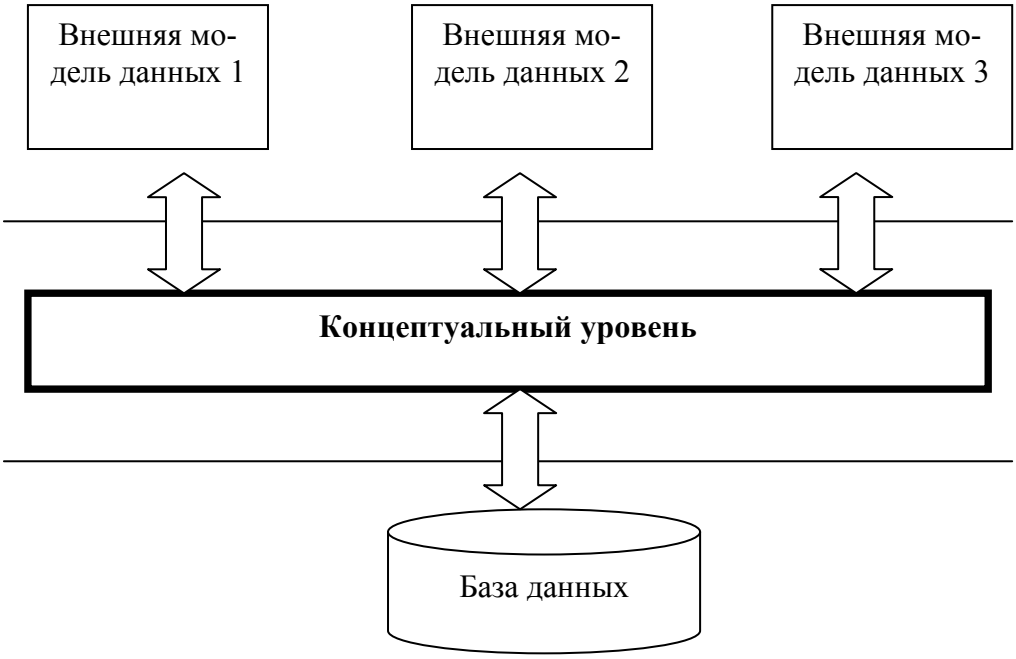

Рисунок 2.1 - Трехуровневая модель системы управления базой данных

В общем случае под СУБД понимается любой программный продукт, поддерживающий процессы создания, ведения и использования БД. Рассмотрим, какие из существующих на рынке программ имеют отношение к БД и на сколько они связаны с базами данных.

К СУБД относятся следующие виды программ:

- полнофункциональные системы управления базами данных;
- серверы баз данных;
- клиенты баз данных;

- средства разработки программ работы с базами данных.

Полнофункциональные СУБД представляют собой традиционные СУБД, которые изначально появились для больших машин, потом для мини-машин и для ПЭВМ. Из числа всех СУБД современные полнофункциональные СУБД являются наиболее многочисленными и мощными по своим возможностям.

Обычно полнофункциональные СУБД имеют развитый интерфейс, позволяющий с помощью команд меню выполнять основные действия с БД: создавать и модифицировать структуры таблиц, вводить данные, формировать запросы, разрабатывать отчеты, выводить их на печать и так далее. Для формирования запросов не обязательно программирование, а удобно пользоваться языком QBE - формулировки запросов по образцу. Многие полнофункциональные СУБД включают средства программирования для профессиональных разработчиков [6].

Серверы БД используются для организации центров обработки данных в сетях ЭВМ. Эта группа БД в настоящее время менее многочисленна, но их количество постепенно возрастает. Серверы БД реализуют функции управления базами данных, запрашиваемые другими (клиентскими) программами обычно с помощью операторов SQL.

В роли клиентских программ для серверов БД в общем случае могут использоваться различные программы: полнофункциональные СУБД, электронные таблицы, текстовые процессоры, программы электронной почты и т.п. При этом элементы пары «клиент-сервер» могут принадлежать одному или разным производителям программного обеспечения.

В случае, когда клиентская и серверные части выполнены одной фирмой, естественно ожидать, что функции распределены между ними вполне рационально. В других случаях зачастую преследуется цель обеспечения доступа к информации «любой ценой». В качестве примера подобного соединения может являться ситуация, когда полнофункциональная СУБД выступает в качестве сервера, а другая (другого производителя) - клиента.

Представим разновидности программ, полученные в результате взаимодействия ПО с БД:

- клиентские программы;
- серверы баз данных и их отдельных компонентов;
- пользовательские приложения.

Программы обоих видов достаточно малочисленны, так как используются, в основном, для системными программистами. Пакетов третьего типа представлены в большем объеме, но в существенно меньшем, чем полнофункциональные СУБД.

К средствам разработки пользовательских приложений можно отнести системы программирования, разнообразные библиотеки программ для различных языков программирования, а также пакеты автоматизации разработок.

По характеру использования СУБД делят на персональные и многопользовательские [11].

Первые из них, обычно обеспечивают возможность создания персональных БД и недорогих приложений, работающих с ними. Персональные СУБД или разработанные с их помощью приложения как правило выступают в роли клиентской части многопользовательской СУБД.

Многопользовательские СУБД, включающие в себя сервер БД и клиентскую часть и, работают в неоднородной вычислительной среде (с разными типами ЭВМ и операционными системами).

Уровень внешних моделей - самый верхний уровень, где каждая модель имеет свое представление данных, это определяющий с точки зрения взгляда на базу данных конкретных приложений. Каждое приложение анализирует и обрабатывает только те данные, которые необходимы данному приложению. Например, подсистема формирования видов работ использует информацию о квалификации сотрудников и ее не интересуют информация об окладе, домашнем адресе и телефоне сотрудника, и наоборот, эта информация используется в подсистеме отдела кадров.

Концептуальный уровень - определяющее звено, здесь база данных представлена в наиболее общем виде, объединяющем данные, которые используют все приложениями, работающие с конкретной базой данных. Фактически концептуальный уровень соответствует обобщенной модели предметной области, для которой формировалась БД. Как любая модель, концептуальная модель выражает только существенные, с точки зрения обработки, особенности объектов реального мира.

Физический уровень - непосредственно данные, находящиеся в файлах или в страничных структурах, размещенных на внешних носителях информации

Эта архитектура обеспечивает логическую (между уровнями 1 и 2) и физическую (между уровнями 2 и 3) независимости при работе с данными. Логическая независимость подразумевает возможность корректировки одного приложения без изменения других приложений, взаимодействующих с этой же БД. Физическая независимость подразумевает возможность перемещения хранимой информации с одного носителя на другой при сохранении работоспособности всех приложений, взаимодействующих с конкретной БД. Это именно то, чего не хватало при использовании файловых систем.

Построение концептуального уровня позволило разработать аппарат централизованного управления базой данных [7].

Модель данных - это некоторая абстракция, привязанная к конкретным данным, позволяющая разработчикам и пользователям определять их уже как информацию, содержащую не только данные, но и взаимосвязи между ними. Главное назначение модели данных состоит в систематизации разнообразной информации и отражении ее свойств, отражающих содержание, структуру, объем, связяи, динамику с учетом удовлетворения информационных потребностей всех категорий пользователей.

Иерархическая модель данных соответствует информационным отображениям объектов реального мира, их сущности, связям в виде ориентированного графа или дерева.

В иерархической модели отношения между данными бывают типа «родитель - потомки», т.е. у каждого объекта только один родитель, но в принципе может быть несколько потомков. Такие отношения, как правило, изображают в виде дерева, в котором ребро между объектами соответствует наличию некоторого отношения, причем название отношения пишется на ребре.

В случае когда граф отношений между объектами представляется не только древовидными структурами, тогда имеем дело с сетевой моделью данных, являющейся расширением иерархической модели. В иерархических структурах запись-потомок имеет только одного предка - в сетевой структуре потомок может иметь любое число предков.

Сетевая модель является более общей, предоставляющей большие возможности по сравнению с иерархической, но она сложнее в построении и использовании.

В настоящее время наиболее распространена при построении БД реляционная модель данных, характеризующаяся простотой структуры данных, удобным для пользователя табличным представлением и возможностью использования формального аппарата реляционной алгебры и реляционного исчисления для обработки данных.

Реляционная модель ориентируется на формирование данных в виде двумерных таблиц. Реляционная таблица представляет собой двумерный массив, обладающий рядом свойств:

- каждый элемент таблицы является одним элементом данных;
- все столбцы в таблице однородные, т.е. все элементы в столбце имеют одинаковый тип (числовой, символьный или другой);
- каждому столбцу соответствует уникальное имя;
- одинаковые строки в таблице отсутствуют;
- порядок следования строк и столбцов носит произвольный характер.

В каждой таблице базы данных может существовать первичный ключ, под которым понимают поле или набор полей, однозначно

идентифицирующий запись. Значение первичного ключа в таблице базы данных носит уникальный характер, то есть в таблице не может быть двух или более записей с одинаковым значением первичного ключа. Первичный ключ должен быть минимально достаточным: в нем не должно быть полей, удаление которых из первичного ключа не отразится на его уникальности.

Для простоты и лучшего понимания структуры инфологической модели при описании сущностей используем русскоязычные обозначения. На рисунке 2.2 представлена диаграмма информационной системы логического уровня. Функциональная схема 0-уровня представлена на рисунке 2.3

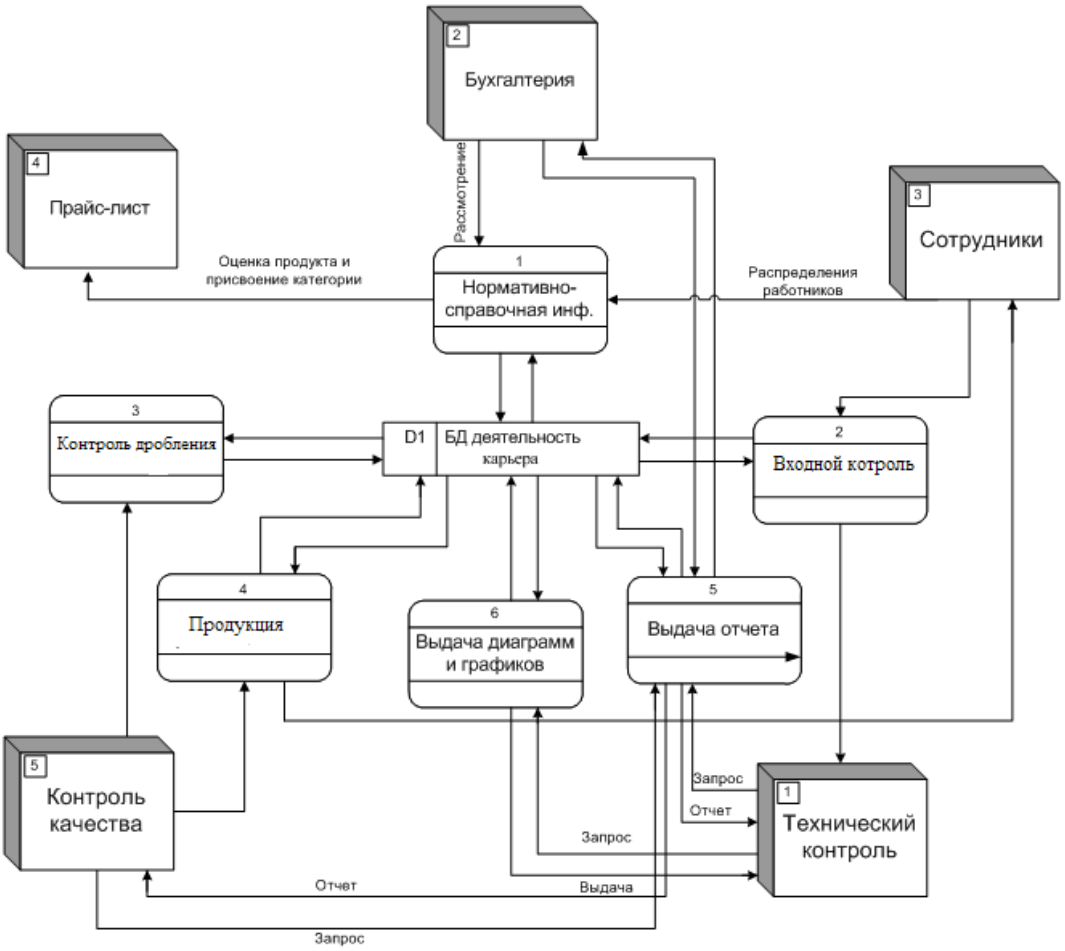

Рисунок 2.2 - DFD- диаграмма нулевого уровня

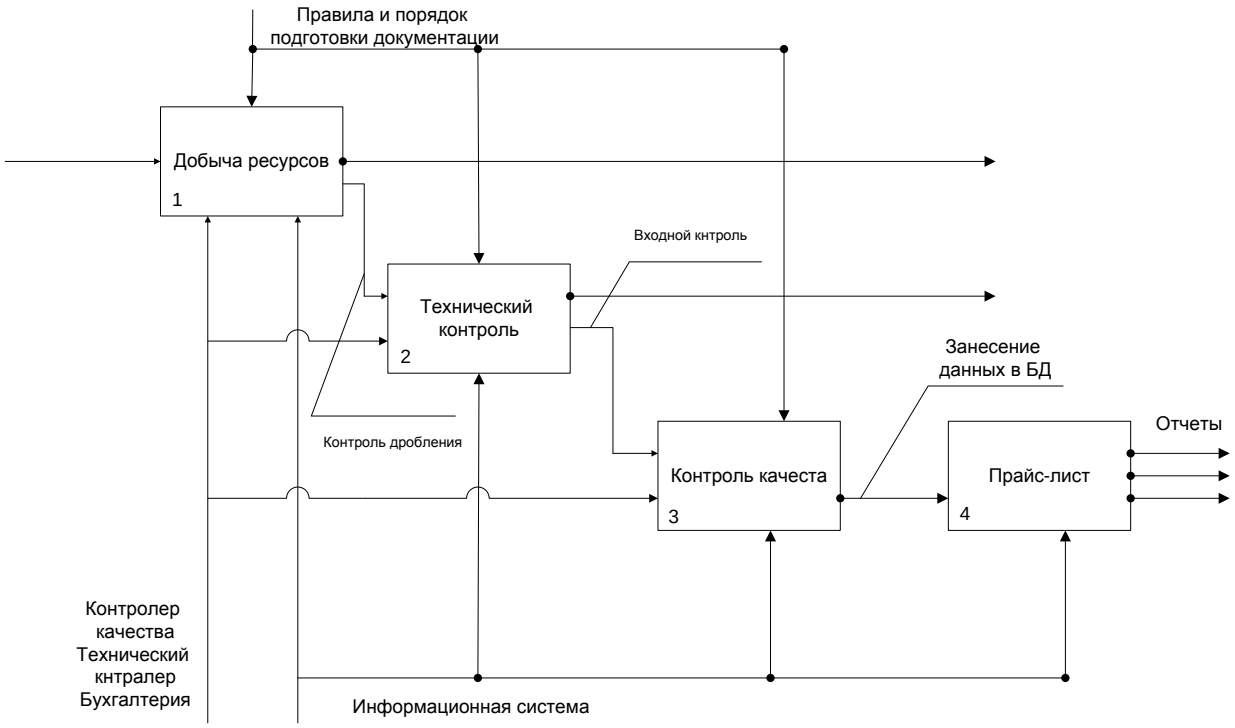

Рисунок 2.4 - Функциональная схема 0-го уровня

2.2 Проектирование компонентов информационной системы с использованием нотации IDEF0

На начальном этапе создания информационной системы необходимо понять принципы работы организация, которую собираемся автоматизировать. Топ-менеджер хорошо знает работу в целом, но он не может вникать в детали работы каждого сотрудника. Рядовой сотрудник хорошо знает что творится на его конкретном месте, но может и не знать, как работают другие. Поэтому для описания работы организации необходимо построение модели, адекватной предметной области и содержащей в себе знания всех участников бизнес-процессов предприятия.

Наиболее удобным языком моделирования бизнес-процессов является IDEFO, в которой системные представления выглядят в виде совокупности взаимодействующих работ или функций. Процедура моделирования системы в IDEFO происходит, начиная с построения контекстной диаграммы, соответствующей наиболее абстрактному уровню описания системы в целом,

содержащему определение субъекта моделирования, цели и точки зрения на модель.

В основу методологических основ IDEF0 положен графический язык интерпретации бизнес - процессов [9]. Совокупность систематизированных и взаимосвязанных диаграмм лежит в основе построения модели в нотации IDEF0. Всякая диаграмма представляет собой единицу описания системы и ее располагают на отдельном листе.

В модель входят четыре вида диаграмм:

- контекстную диаграмму;
- диаграммы декомпозиции;
- диаграммы дерева узлов;
- диаграммы только для экспозиции (FEO).

Контекстная диаграмма является вершиной древовидной структуры диаграмм и представляет собой самое общее описание системы и ее взаимодействия с внешней средой. После описания системы в целом проводится разбиение ее на крупные фрагменты. Такой подход называется функциональной декомпозицией, а диаграммы, описывающие отдельный фрагмент и их взаимодействие, называются диаграммами декомпозиции. После декомпозиции контекстной диаграммы проводится декомпозиция каждого большого фрагмента системы на более мелкие и так далее, до достижения нужного уровня подробности описания. Каждый этап декомпозиции заканчивается экспертным этапом предметной области и указывает на соответствие реальных бизнес-процессов построенным диаграммам. Выявленные несоответствия исправляются, и только после прохождения экспертизы без замечаний можно приступить к следующему этапу декомпозиции. Так достигаем соответствие модели реальным бизнес-процессам на каждом уровне модели, синтаксис описания системы в целом и каждого ее элемента одинаков во всей модели.

Диаграмма дерева узлов показывает иерархическую зависимость работ, но не взаимосвязи между работами. Диаграмм деревьев узлов может быть

сколь угодно много, поскольку дерево может быть построено на произвольную глубину и не обязательно с корня.

Диаграммы для экспозиции (FEO) строятся для иллюстрации отдельных элементов модели, для иллюстрации альтернативной точки зрения или для специальных целей.

В результате анализа предметной области была разработана функциональная модель системы учета закупок и реализации товара. Проектирование проводилось на основе методологии IDEFO. При помощи такого подхода построена диаграмма декомпозиции работы бухгалтерии, представленная на рисунке 2.5, декомпозиция работы контролера продукции представлена на рисунке 2.6.

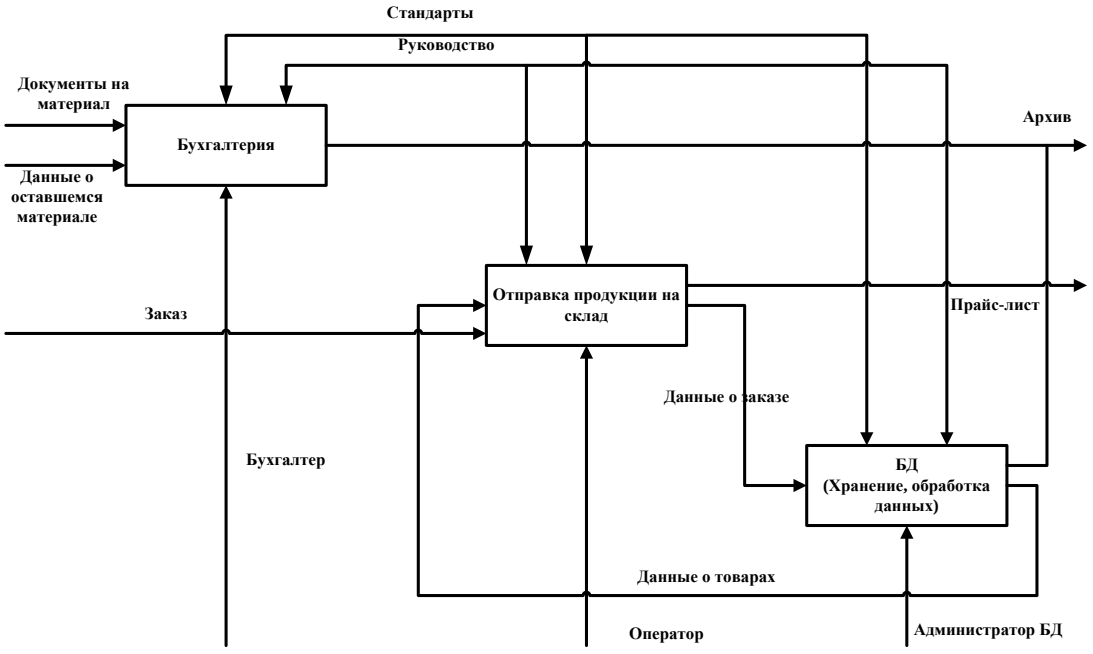

Рисунок 2.5 - Диаграмма IDEFO декомпозиции работы бухгалтерии

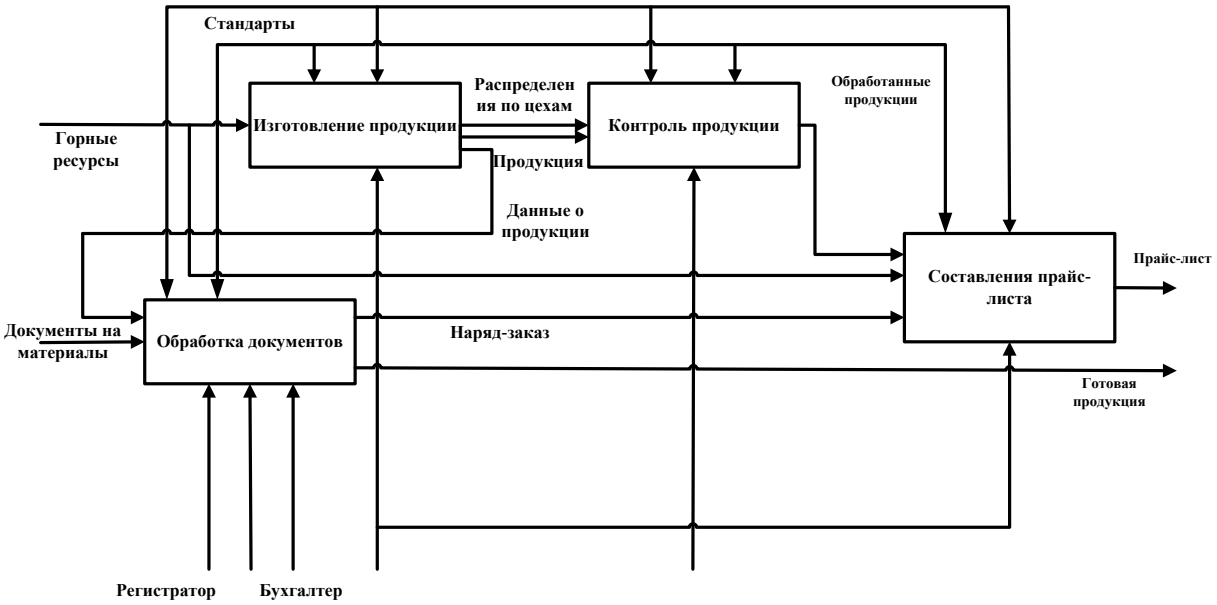

Рисунок 2.6 - Диаграмма IDEFO декомпозиции работы контролера продукции

Периодичность выполнения данных процессов:

- Выработка продукции
- Подведение контроля продукции
- Логистическое направление.

Ведение единой БД по предоставляемой продукции ER – диаграмма представлена на рисунке 2.7.

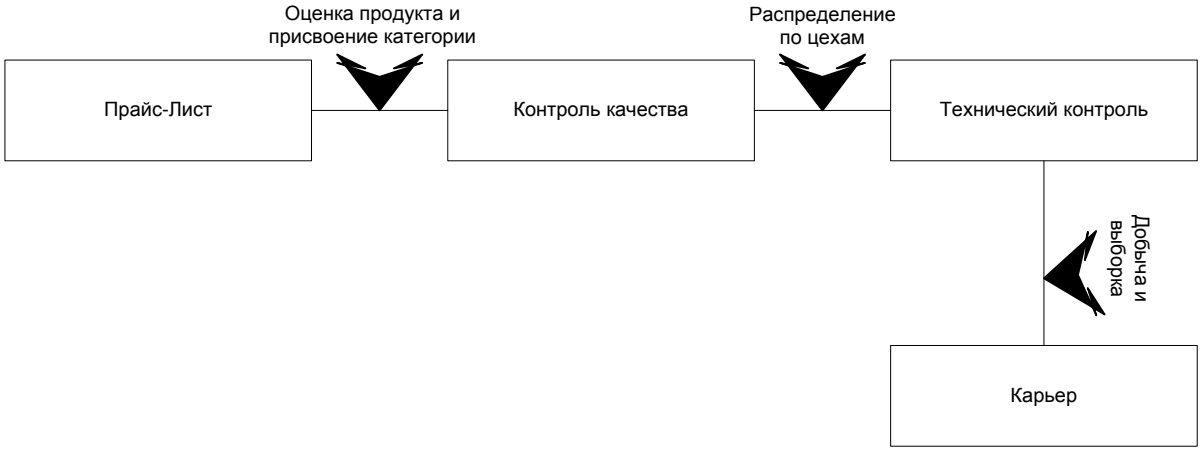

Рисунок 2.7 - ER-диаграмма информационной системы на логическом уровне

Нормализация предусматривает определение требуемых атрибутов с последующим созданием из них нормализованных таблиц, основанных на функциональных зависимостях между этими атрибутами. Отношение, в котором на пересечении каждой строки и каждого столбца содержится атомарное значение, находится в первой нормальной форме, при этом необходимо, чтобы отношение имело первичный ключ.

Вторая нормальная форма применяется к отношениям с составными ключами, т.е. к таким отношениям, первичный ключ которых состоит из двух или больше атрибутов. Отношение с первичным ключом на основе единственного атрибута всегда находится в 2НФ. Отношение, которое находится в 1НФ и каждый атрибут которого, не входящий в состав первичного ключа, зависит только от полного значения ключа и не зависит ни от какого отдельного атрибута, входящего в состав первичного ключа, имеет вторую нормальную форму.

Отношение находится в 3НФ, если оно представлено в 2НФ и не имеет не входящих в первичный ключ атрибутов, которые находились бы в транзитивной функциональной зависимости от этого первичного ключа.

Разработанная на первом этапе модель не удовлетворяет условиям нормальных форм.

Разработанная модель находится в третьей нормальной форме т.к.:

- атрибуты сущностей являются атомарными;

- каждый неключевой атрибут функционально полно зависит от первичного ключа;

- в модели отсутствуют транзитивные зависимости неключевых атрибутов от ключа.

2.3 Создание реляционной базы данных отдела технического контроля

Предварительная структура базы данных включает следующие таблицы:

- «Сотрудники»;
- «Лист ознакомления»;
- «Лист рассылки»;
- «Лист регистрации изменений»;
- «Лист согласования»;
- «Лист контроля»;
- «Прайс»;
- «Продукция».

Построение базы данных начинается с проектирования таблиц. В Microsoft Access существует четыре способа создания пустой таблицы [6].

Использование мастера баз данных для построения всей базы, содержащей требуемые отчеты, таблицы и формы, за одну операцию. Мастера баз данных строят новую базу данных, ее нельзя использовать для добавления новых таблиц, форм, отчетов в уже существующую базу данных.

Мастер таблиц делает возможным выбрать поля для данной таблицы из множества определенных ранее таблиц, таких как деловые контакты.

- Ввод данных непосредственно в пустую таблицу в режиме таблицы. При сохранении новой таблицы в Microsoft Access данные анализируются и каждому полю ставится в соответствие необходимый тип данных и формат.
- Уточнение всех параметров макета таблицы в режиме конструктора.
- Построение новой таблицы в режиме Конструктор состоит из нескольких шагов:

- В столбце Имя поля вводят имя поля таблицы и нажимают клавишу TAB (Enter), оставляют текстовый тип в столбце, тип данных или щелкают по стрелке раскрывающегося списка и выбирают нужный тип поля. Можно выбрать следующие типы данных:

- текстовый - текст или комбинация букв и цифр, а также числа, не участвующие в вычислениях; тип данных - по умолчанию; число символов в поле не должно превышать 255; максимальное число символов, которое можно ввести в поле, задается в свойстве Размер поля;

- MEMO - длинный текст или сочетание текста и числовых данных; максимальная длина 64000 символов;

- числовой - данные, используемые в вычислениях; конкретные варианты числового типа и их длина задаются в свойстве Размер поля;

- денежный - денежные значения или данные для вычислений, проводимых с точностью 15 знаков до и 4 знака после запятой; длина поля 8 байт; при обработке числовых значений из денежных полей выполняются вычисления с фиксированной точкой более быстрые, чем вычисления для полей с плавающей точкой, кроме того, при вычислениях предотвращается округление;

- дата / время - даты и время, относящиеся к годам от 100 до 9999 включительно; длина поля 8 байт;

- счетчик - уникальные последовательно возрастающие на единицу или случайные числа, автоматически вводимые при добавлении каждой новой записи в таблицу. Значение полей этого типа изменить или удалить нельзя; длина поля 4 байта для длинного целого, для кода репликации - 128 байт; в таблице не может быть более одного поля этого типа; используется для определения уникального ключа таблицы;

- логический - логические данные, которые могут иметь одно из двух возможных значений Да/Нет; Истина/Ложь; Вкл./Выкл.; длина поля 1 бит;

- поле объекта OLE - объект, связанный или внедренный в таблицу Access; длина поля - до 1 Гигабайта; для полей типа OLE и MEMO не допускается сортировка и индексирование;

- гиперссылка - путь к файлу на жестком диске, путь UNC или адрес URL. Если щелкнуть мышью на поле гиперссылки, Access выполнит переход на соответствующий объект, документ, страницу Web или другое место назначения. Максимальная длина 64000 символов;

- мастер подстановок. Выбор этого типа данных запускает мастера подстановок. Мастер строит для поля список значений на основе полей из другой таблицы. Значения в такое поле будут вводиться из одного из полей списка. Соответственно, фактически тип данных поля определяется типом данных поля списка. Возможно также определение поля со списком постоянных значений.

Нажимают клавишу TAB и вводят описание поля (необязательный реквизит).

Устанавливают необходимые свойства поля во вкладках Общие и Подстановка.

Повторяют пп. 1 - 4 для каждого создаваемого поля.

Определяют первичный ключ.

Сохраняют таблицу. Щелкают по кнопке Сохранить, вводят имя таблицы и щелкают по кнопке ОК.

Опишем создание таблици представим описание в виде таблицы 1. Определяем в таблице каждое поле - т.е. задаем его имя, тип и размер в режиме конструктора.

Таблица 1 –Технический контроль

Поле	**Тип данных**	**Размер**	**Подстановка**
Код	Счетчик	Длинное целое	Ключевое Поле
Код входного контроля	Числовой	Длинное целое	Поле

Продолжение таблицы 1

Входной контроль	Логический	Да/Нет	Поле со списком (таб. Специальность)
Контроль приемки горной массы	Логический	Да/Нет	Флажок
Операционный контроль	Логический	Да/Нет	Флажок
Код контроля дробления	Числовой	Длинное целое	Поле со списком (таб. Контроль дробления)
Контроль дробления горной массы	Логический	Да/Нет	Флажок
Контроль грохочения горной массы	Логический	Да/Нет	Флажок
Код продукции	Числовой	Длинное целое	Поле со списком (таб. Продукция)
Приемо-сдаточный контроль	Логический	Да/Нет	Флажок
Приемо-сдаточный лабораторный контроль	Логический	Да/Нет	Флажок
Периодический лабораторный контроль	Логический	Да/Нет	Флажок
Код сотрудника	Числовой	Длинное целое	Поле со списком (таб. Сотрудник)

Таблица 2 – Контроль качества

Поле	**Тип данных**	**Размер**	**Подстановка**
Код контроля	Счетчик	Длинное целое	Ключевое Поле
Дефективность	Текстовый	30	Поле
Брак	Логический	Да/Нет	Флажок
Образец	Текстовый	100	Поле
Категория пароды (частота)	Текстовый	50	Поле
Раздавленные	Логический	Да/Нет	Флажок
Полнота добычи	Текстовый	100	Поле
Код карьера	Числовой	Длинное целое	Поле со списком (таб. Карьер)
Код сотрудника	Числовой	Длинное целое	Поле со списком (таб. Сотрудник)

Таблица 3 – Лист согласования

Поле	Тип данных	Размер	Подстановка
Код	Счетчик	Длинное целое	Ключевое Поле
Код согласующего	Числовой	Длинное целое	Поле со списком (таб. Сотрудники)
Замечания предложения	Текстовый	100	Поле

Таблица 4 – Прайс-лист

Поле	Тип данных	Размер	Подстановка
Код листа	Счетчик	Длинное целое	Ключевое Поле
Код продукта	Числовой	Длинное целое	Поле со списком (таб. Продукция)
Наименование продукции	Текстовый	50	Поле
Количество	Числовой	Длинное целое	Поле
Единицы измерения	Текстовый	15	Поле
Цена	Денежный	Денежный	Поле

Таблица 5 – Продукция

Поле	Тип данных	Размер	Подстановка
Код продукции	Счетчик	Длинное целое	Ключевое Поле
Наименование	Текстовый	50	Поле
Категория	Дата/время	Авто	Поле
Код дробления	Числовой	Длинное целое	Поле со списком (таб. Цех дробления)
Наименования дробления	Текстовый	50	Поле со списком (таб. Цех дробления)
Код склада	Длинное целое	Длинное целое	Поле со списком (таб. Склад)
Наименование склада	Текстовый	50	Поле со списком (таб. Склад)

Таблица 6 – Контроль дробления

Поле	Тип данных	Размер	Подстановка
Код	Счетчик	Длинное целое	Ключевое Поле
Контроль дробления горной массы	Логический	Да/Нет	Флажок
Контроль грохочения горной массы	Логический	Да/Нет	Флажок
Приемо-сдаточный контроль	Логический	Да/Нет	Флажок
Код сотрудника	Числовой	Длинное целое	Поле со списком (таб. Сотрудники)
Код дробления	Числовой	Длинное целое	Поле со списком (таб. Цех дробления)

Таблица 7 – Входной контроль

Поле	Тип данных	Размер	Подстановка
Код	Счетчик	Длинное целое	Ключевое Поле
Входной контроль	Логический	Да/Нет	Флажок
Контроль приемки горной массы	Логический	Да/Нет	Флажок
Операционный контроль	Логический	Да/Нет	Флажок
Код сотрудника	Числовой	Длинное целое	Поле со списком (таб. Сотрудники)
Код погрузки	Числовой	Длинное целое	Поле со списком (таб. Цех погрузки)

Таблица 8 – Лист рассылки

Поле	Тип данных	Размер	Подстановка
Код	Счетчик	Длинное целое	Ключевое Поле
Дата	Дата/время	Краткий формат даты	
№ листа	Числовой	Длинное целое	Поле
Наименование подразделения	Текстовый	255	Поле
Фамилия	Текстовый	15	Поле
Имя	Текстовый	15	Поле
Отчество	Текстовый	15	Поле

Таблица 9 – Карьер

Поле	Тип данных	Размер	Подстановка
№ карьера	Счетчик	Длинное целое	Ключевое Поле
Наименование	Текстовый	255	Поле
Запасы карьера	Числовой	Длинное целое	Поле
Период использования	Числовой	Длинное целое	Поле
Объем горной массы	Числовой	Длинное целое	Поле
Размеры карьера	Текстовый	20	Поле

Таблица 10 – Склад

Поле	Тип данных	Размер	Подстановка
Код	Счетчик	Длинное целое	Ключевое Поле
Наименование	Текстовый	50	Поле
Код дробления	Числовой	Длинное целое	Поле со списком (таб. Цех дробления)
Наименование цеха	Текстовый	50	Поле со списком (таб. Цех дробления)

Таблица 11 – Сотрудник

Поле	Тип данных	Размер	Подстановка
№ сотрудника	Счетчик	Длинное целое	Ключевое Поле
Фамилия	Текстовый	15	Поле
Имя	Текстовый	15	Поле
Отчество	Текстовый	15	Поле
Должность	Текстовый	50	Поле
Телефон	Текстовый	20	Поле
Адрес	Текстовый	50	Поле
Подпись	Текстовый	20	Поле

Таблица 12 – Цех сортировки

Поле	Тип данных	Размер	Подстановка
Код	Счетчик	Длинное целое	Ключевое Поле
Наименование	Текстовый	50	Поле
Норма в час	Числовой	Длинное целое	Поле
Код погрузки	Числовой	Длинное целое	Поле со списком (таб. Цех погрузки)
Наименование погрузки	Текстовый	20	Поле со списком (таб. Цех погрузки)

Таблица 13 – Цех дробления

Поле	Тип данных	Размер	Подстановка
Код	Счетчик	Длинное целое	Ключевое Поле
Наименование	Текстовый	50	Поле
Норма в час	Числовой	Длинное целое	Поле
Код погрузки	Числовой	Длинное целое	Поле со списком (таб. Цех сортировки)
Наименование сортировки	Текстовый	50	Поле со списком (таб. Цех сортировки)

Таблица 14 – Цех дробления

Поле	Тип данных	Размер	Подстановка
Код	Счетчик	Длинное целое	Ключевое Поле
Наименование	Текстовый	50	Поле
Норма в час	Числовой	Длинное целое	Поле
Код карьера	Числовой	Длинное целое	Поле со списком (таб. Карьер)
Наименование карьера	Текстовый	255	Поле со списком (таб. Карьер)

Связь между таблицами устанавливает отношения между совпадающими значениями в ключевых полях, обычно между полями

разных таблиц, имеющими одинаковые имена. В большинстве случаев с ключевым полем одной таблицы, являющимся уникальным идентификатором каждой записи, связывается внешний ключ другой таблицы. Для того чтобы определить связь между таблицами, следует добавить таблицы в окно Схема данных и перенести с помощью мыши ключевое поле одной таблицы в другую таблицу Тип создаваемой связи зависит от полей, для которых определяется связь. Отношение «один-ко-многим» создается в том случае, когда только одно из полей является ключевым или имеет уникальный индекс. Отношение «один-к-одному» создается в том случае, когда оба связываемых поля являются ключевыми или имеют уникальные индексы.

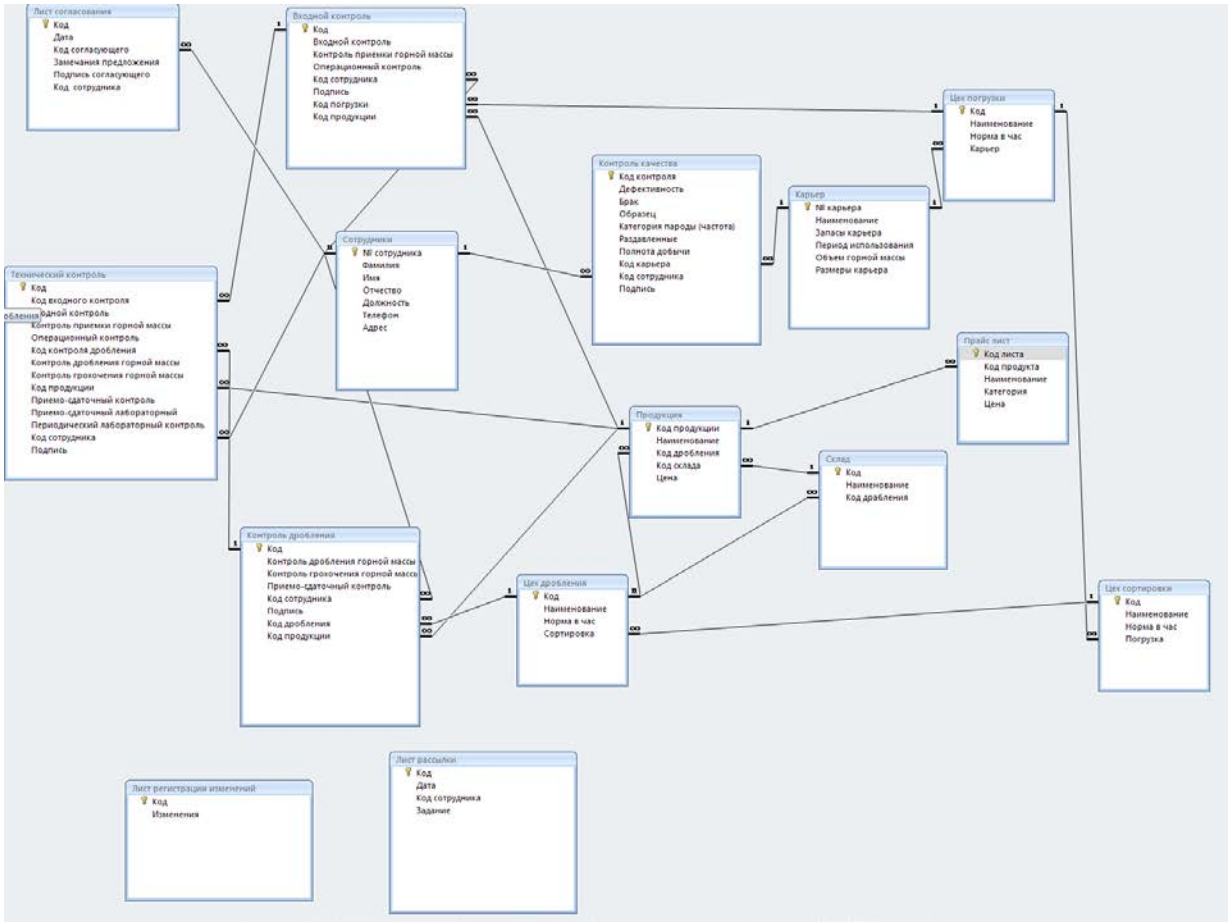

Рисунок 2.8 - Схема данных

3. Описание информационной подсистемы поддержки деятельности отдела технического контроля

3.1 Функциональные требования к программе

Процецедура проектирования программного обеспечения (ПО) берет начало с уточнения требований к разрабатываемому ПО и его исходной информации. Результатом анализа требований служит набор спецификаций программного обеспечения представленный текстовыми описаниями, структурными схемами и диаграммами. Во время уточнения спецификаций строится общая модель предметной области и конкретизируются основные функции программного продукта, а также его поведения при взаимодействии с окружающей средой.

Этап постановки задачи является одним из наиболее существенных этапов построения ПО, тогда и принимается решение касательно функций создаваемого ПО, эксплуатационных ограничений, которые на него накладываются, происходит уточнение архитектуры, среды разработки ПО, интерфейса пользователя, этот выбор определяет качество и стоимость конечного ПО.

При формировании функциональных требований учитывается, что чем они более детализированы, тем оценка работ будет наиболее точной как по времени, так и по стоимости. Оценка будет точной, если на последующих стадиях разработки ПО не появятся дополнения к первоначально установленным функциональным требованиям, также описывая требования, не стоит слишком углубляться в детализацию. Описания требуют только функции программы, детализация должна быть уже в ходе разработки технического задания.

Функциональная спецификация составляется из трех частей:

1. Описании внешней информационной среды, которая взаимодействует с разрабатываемым программным обеспечением. Должны быть уточнены каналы ввода и вывода и все информационные объекты, которые будут ис-

пользованы и к которым будет применено разрабатываемое программное средство, а также существенны связи между этими информационными структурами.

2. Уточнение функционального назначения программного обеспечения, определенного множеством состояний информационной среды. Введем обозначение для всех определяемых функций, специфицкацию их входных данных и результатов выполнения, с указанием вида данных и заданий всех ограничений, которым должны соответствовать эти данные и результаты, конкретизируется содержание каждой из этих функций

3. Описании форс-мажорных ситуаций, если они возникают при реализации программы, и отклика на подобные ситуации, которые должны разрешать соответствующие программы. Должны быть приведены все конкретные ситуации, когда программное обеспечение не сможет качественно выполнять свои определенные функции. На каждый конкретный случай определяется реакция программы. Эксплуатационными требованиями определяются характеристики разрабатываемого программного обеспечения, которые могут появиться в процессе его эксплуатации. К таким характеристикам относем:

правильность - функционирование полностью отвечающее техническому заданию, это необходимое требование для любого программного обеспечения, но из-за того, что любое тестирование не дает 100%-ных гарантий правильности, речь можно вести об определенной вероятности наличия ошибок;

универсальность - гарантия правильного функционирования при любых данных и защиты от неверных данных. Как в предыдущем случае, подтвердить универсальность программы весьма непросто, поэтому имеет не смысла говорить о степени ее универсальности;

надежность - обеспечение абсолютной повторяемости результатов, т. е. обеспечение достоверности результатов при наличии различного вида сбоев. Источниками помех могут быть технические и программные средства, а

также люди, эксплуатирующие технические средства. В настоящее время существует множество способов, позволяющих избежать потери информации при сбоях [17]. Этого можно добиться, воспользовавшись «созданием контрольных точек», при которых сохраняются промежуточные результаты, позволяющие после сбоя программы продолжать работу с информацией, записанной в последней контрольной точке. Возможно также сократить число ошибок, пользуясь дублированием систем или ввода избыточной информации;

программная совместимость - возможность совместной работы с другим программным продуктом. Зачастую речь идет о работе программы под управлением заданной операционной системы, но может возникнуть необходимость обмена данными с некоторой другой программой, тогда необходимо точно оговарить формат передаваемой информации;

эффективность - использование минимально возможного числа ресурсов технических средств.

адаптируемость - возможность быстрой модификации с целью корректировки под изменяющиеся условиям работы. Дать количественную оценку этой характеристики практически невозможно. Можно только свидетельствовать о том, что при разработке данного ПО использовались приемы, упрощающие его модернизацию;

Четкая формулировка спецификации требований к разрабатываемому ПО, с последующим ее внесением в техническое задание, представляет собой сложную и ответственную задачу, требующую проведения предпроектных исследований.

3.2 Структура программного обеспечения

Пакет прикладных программ (ППП) используется для контроля и реализации математической части. Состав ППП на ПК называют программной конфигурацией.

Используемый в работе пакет состоит из двух взаимосвязанных модулей (рисунок 3.1) «CodeGear RAD Studio 2009» – универсальный пакет разработки анализа программных продуктов, «Microsoft Office Access 2007» – хранение первичной информации, а так же пополнение базы данных системы.

Рисунок 3.1 – Структурная схема ППП деятельности ООО «БЮНА»

Программное обеспечение, может быть условно разделено на три категории:

– Системное ПО, реализующее разные вспомогательные функции, например, формирование копий используемых данных, выдачу справочной информации о компьютере, анализ работоспособности устройств компьютера и т.д.;

– Прикладное ПО, поддерживающее выполнение необходимых работ на ПК: редактирование текстовых файлов, разработку рисунков или картинок, обработку информационных массивов;

– Инструментальное ПО, служащее для разработки нового программного обеспечения для компьютера на языке программирования.

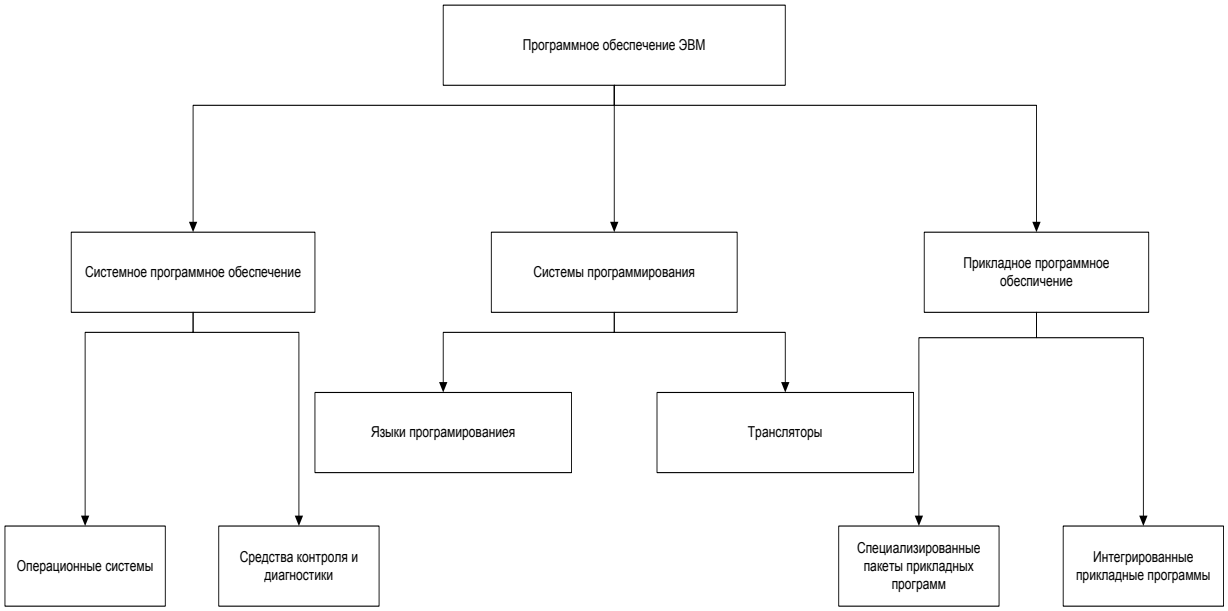

Рисунок 3.2 – «Программное обеспечение»

Системное ПО, предназначено для общего пользования, не связано с конкретным использованием ПК и выполняет функции планирования и управления задачами, управления вводом-выводом и т.д. К системному ПО отнесем: операционные системы (ПО загружается в ОЗУ при включении компьютера); программы – оболочки (обеспечивающие более удобный и наглядный способ взаимодействия с компьютером, чем используя командную строку DOS); операционные оболочки – интерфейсные системы, используемые для построения графических интерфейсов, мультипрограммирования и.т.д.; драйверы - программы, которые предназначены для управления портами периферийных устройств, как правило они загружаются в оперативную память при включении компьютера; утилиты - вспомогательные или специальные программы, предоставляющие пользователю дополнительные услуги.

К утилитам можно отнести:

– Диспетчеров файлов или файловых менеджеров;
– Средства динамического сжатия данных;
– Средства просмотра и воспроизведения;

– Средства диагностики; средства контроля позволяющие проверить конфигурацию компьютера и функциональность его устройств, прежде всего жестких дисков;

– Средства коммуникаций, способствующие организации обмена информацией между компьютерами;

– Средства, направленные на поддержание компьютерной безопасности.

3.3 Основные этапы разработки информационной системы

Использование полного цикла тестирования и настройка на всех основных этапах построения и внедрения программ, гарантирует полноценное управление качеством программного, алгоритм разработки представлен на рисунке 3.3.

1) Сбор и анализ. Подходы к анализу работоспособности приложений похожи с оценкой конечного пользователя, поэтому к разработке модуля нужно привлекать специалистов на самом раннем этапе, то есть при сборе и анализе требований.

Начальный этап проекта подразумевает первичную оценку сроков и построения плана работ с привязкой к этапам разработки и выпуску версий приложений, подбор типов тестирования и их очередности, а также оценки трудозатратности. План тестирования состоит из:

- выбора методологий и инструментов тестирования;

Рисунок 3.3 – Алгоритм разработки

- определения цели и продолжительности каждого этапа;
- состава работ: охват и используемые виды тестирования;
- планирования трудозатрат в контексте необходимого квалификационного уровня и структуры проектной группы.

Работа с требованиями. Если требования к программному обеспечению плохо систематизированы или плохо составлена документация к этим требованиям, то самый качественный код не будет иметь практического значения. Требования должны быть сформулированы и введены еще до начала этапа разработки, что способствует достижению наилучших результатов процесса обеспечения качества.

2) Разработка тестовой документации. Варьирование степени детализации, формата и охвата тестовой документации зависит от степени детализации проекта, которая формируется использованием специфических параметров. Кроме того, документация должна иметь актуальный вид, в котором учтены новые дополнения и изменения, производимые по ходу реализации проекта.

3) Тестирование прототипа служит для снижения рисков посредством раннего выявления несоответствий требованиям, «слабых мест» в структуре

приложений, затруднений, связанных с неудобствами эксплуатации и недостатками логики функционала приложений еще на стадии разработки. Своевременные изменения, выполненные на начальном этапе во время макетирования, способствует предотвращению весьма затратных переделок системы на этапах разработки.

3) Основное тестирование. В план процедур по обеспечению качества могут быть включены различные виды тестирования, которые можно представить в виде таблицы (Таблица 15).

Таблица 15 - Виды тестирования

Компоненты системы	Глубина/тип тестирования	Область тестирования
Модульное тестирование Интеграционное тестирование Системное интеграционное тестирование	Приемочное тестирование «Позитивное» тестирование «Негативное» тестирование Исследовательское тестирование	Функциональное тестирование Тестирование производительности Тестирование графического интерфейса пользователя Тестирование удобства пользования Тестирование безопасности Тестирование базы данных Тестирование совместимости

Результаты тестирования обрабатываются в виде отчета о проделанной работе, с детализацией обнаруженных дефектов, что повышает удобство и обеспечение должного уровня вовлеченности в процесс, подобные отчеты как в шаблонах отчетности компании, так и в шаблонах заказчиков.

4) Стабилизация. По окончании формирования всех функций, когда продукт или система практически готова к вводу в эксплуатацию, проходит тестирование на стабилизацию. Такой этап тестирования проходит в условиях, максимально близким к реальности. Некоторые функциональные возможности могут быть проверены только на этом этапе, например, on-laine оплата и т.п.

5) Эксплуатация. Несмотря на то, что система уже введена в эксплуатацию, роль тестирования на стадии поддержки все еще велика.

Все вносимые в ПО корректировки (а также изменения рабочей среды) подвергаются тщательному тестированию, подтверждающему, что программное обеспечение продолжает выполнение заложенных в него бизнес-функции, не прерывая при этом работоспособности остальных и вообще системы приложений, с которыми оно интегрировано.

3.4 Основные операции использования программы

Разработка алгоритма работы информационной системы была сделана для более детального описания функционирования информационной подсистемы. Приведем описание схемы алгоритма работы программы.

Блок 1. «Вводить справочную информацию?» В данном блоке ставиться вопрос о ввод нормативно справочной информации: продукции, лист согласования. При вводе справочной информации переходим к блоку 2. Если такая операция не целесообразна, то переход происходит к блоку 3.

Блок 2. «Ввод справочной информации» В данном блоке происходит ввод необходимой нормативно-справочной информации: информация о контроле дробления, контроле качества. После завершения заполнения переход осуществляется к блоку 1.

Блок 3. «Обрабатывать справочную информацию» Ставиться условие о необходимости обработки введенной или имеющейся информации. В соответствии с выбранным решение происходит переход к блоку 4, если данное решение не целесообразно переходим к блоку 5.

Блок 4. «Обработка справочной информации» на данном этапе происходит анализ введенной информации. После завершения выполнения переход осуществляется к блоку 3.

Блок 5. «Вывод статистической информации». В данном блоке происходит вывод статистической информации.

По окончании вывода информации переход происходит к блоку 5.

Блок 6. «Формирование прайс-листа» Данный блок позволяет вывести график. При нецелесообразности данного блока. Завершить работу программы или вернуться к блоку 1.

Согласно данному алгоритму, а также особенностям проектирования, как информационных модулей, так и поддержки принятия управленческих решений в работе ООО «БЮНА», модуль данной системы использует интерфейс программы, который представлен на рисунке 3.4.

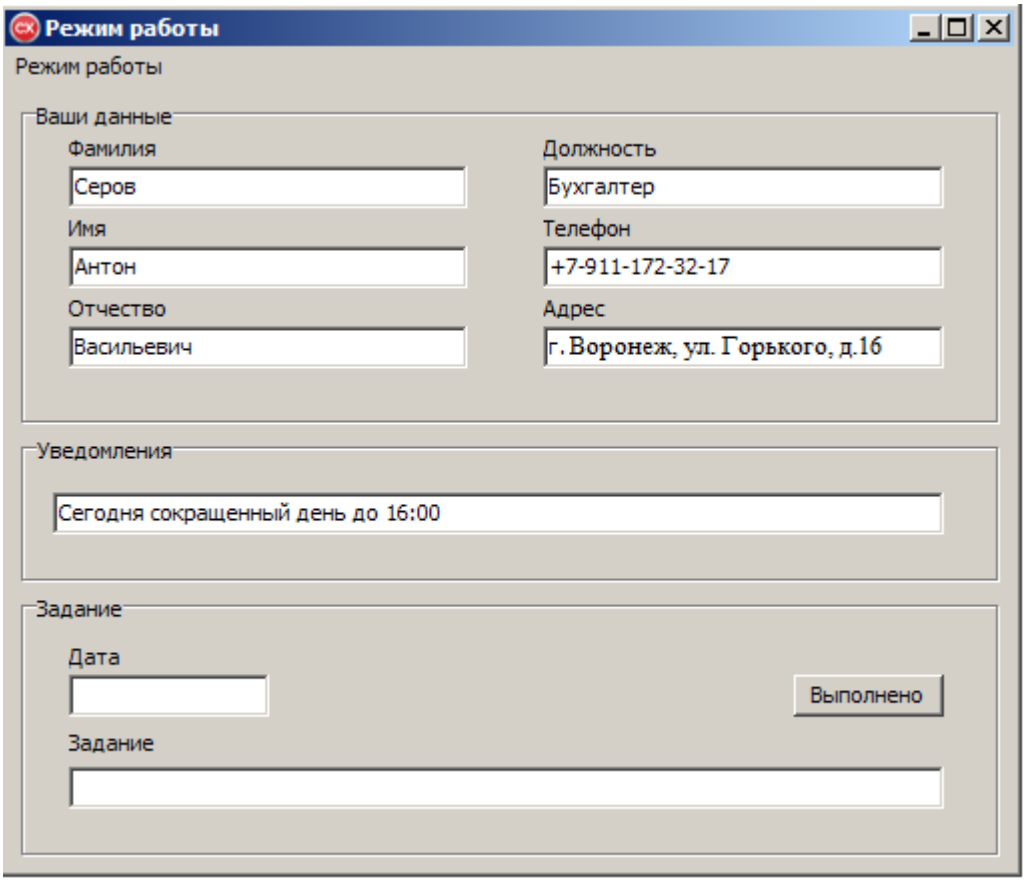

Рисунок 3.4 - Интерфейс программы

Далее необходимо осуществить переход по формам программы по нажатию кнопок. Для перехода к содержанию и последующему открытию форм представлен следующий код:

Описание библиотек:

#include <vcl.h>

```
#pragma hdrstop
#include "Unit1.h"
#include "Unit2.h"
#include "Unit3.h"
#include "Unit4.h"
#include "Unit5.h"
//---------------------------------------------------------------------------
```

Инициализация пакетов, взаимосвязь установки взаимными ссылками использующих их модулей:

```
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------------
```

Создавайте обработчики событий от IDE:

```
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{}
//---------------------------------------------------------------------------
```

Переход на другу форму:

```
void __fastcall TForm1::N2Click(TObject *Sender)
{
Form2->Show();
}
//---------------------------------------------------------------------------
```

Аналогично выглядит команды для других кнопок.

Далее создадим взаимосвязь, базы данных созданной заранее и элементов формы: ADOConnection, ADOTable, DataSource, DBNavigator, DBGrid.

1) Щелкаем по DataSource1 и в DataSet выбираем ADOTable1. Щелкаем по ADOTable1 и в свойстве Connection выбираем ADOConnection1.

Дважды щелкаем по ADOConnection1 и нажимаем кнопку Build. Выбираем пункт Microsoft jet 4.0 OLE DBProvider.

2) В Object Inspector ставим Connected в true, в User Name напишем "Admin", поле пароля оставим пустым. Во избежании повторения этой процедуры, установите LoginPrompt в false. Щелкаем по DBGrid и в DataSource выбираем DataSource1. Тоже самое повторяем и для DBNavigator. Щелкаем по ADOTable1 в TableName выбираем таблицу. Устанавливаем Active в true.

3) При двойном нажатие по DBGrid, в появившемся окошечке нажимаем на кнопку "Add All Fields". Щелкая по полям, можно задать им другие имена для отображения, свойство FieldName и изменить ширину - Width. Подключение остальных таблиц происходит аналогичным образом.

Также на форму необходимо добавить элементы Label которые дают возможность подписывать наименование производимых нами действий.

На форме есть необходимость и в расположении элементов Edit для вывода статичной информации и информация, которая будет вычислена как приведено в форме «Прайс-лист» (рисунок 3.5).

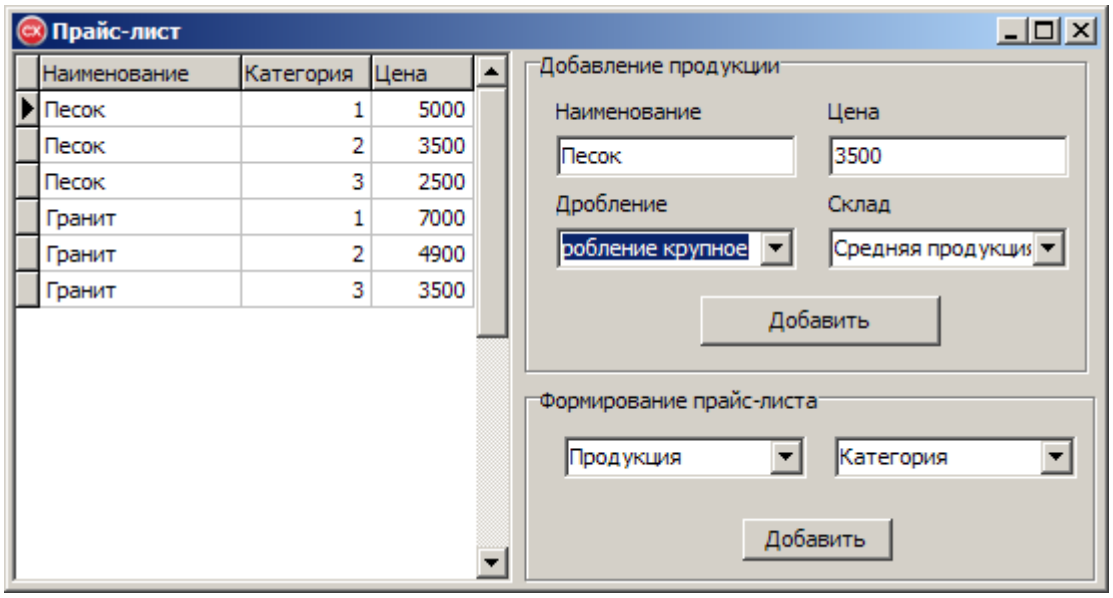

Рисунок 3.5 - форма «Прайс-лист»

3.5 Инструкция пользователя

Запуск программы осуществляется двойным нажатием на системный ярлык с наименованием «Project1.exe». При запуске систем появляется окно с предложением ввести логин и пароль (рисунок 3.6). После ввода будет осуществлен вход в систему с теми правами, которые были заданы при создании учетной записи.

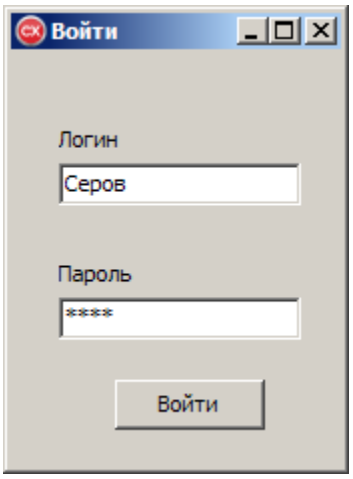

Рисунок 3.6 – форма «Вход»

Далее, исходя из прав доступа, пользователю будет дана возможность редактирования и просмотра записей в базе данных. Так, например бухгалтер имеет возможность доступа к складу и прайс-листу.

В окне «Склад» (рисунок 3.7) показана информация о продукции и на каком складе она хранится.

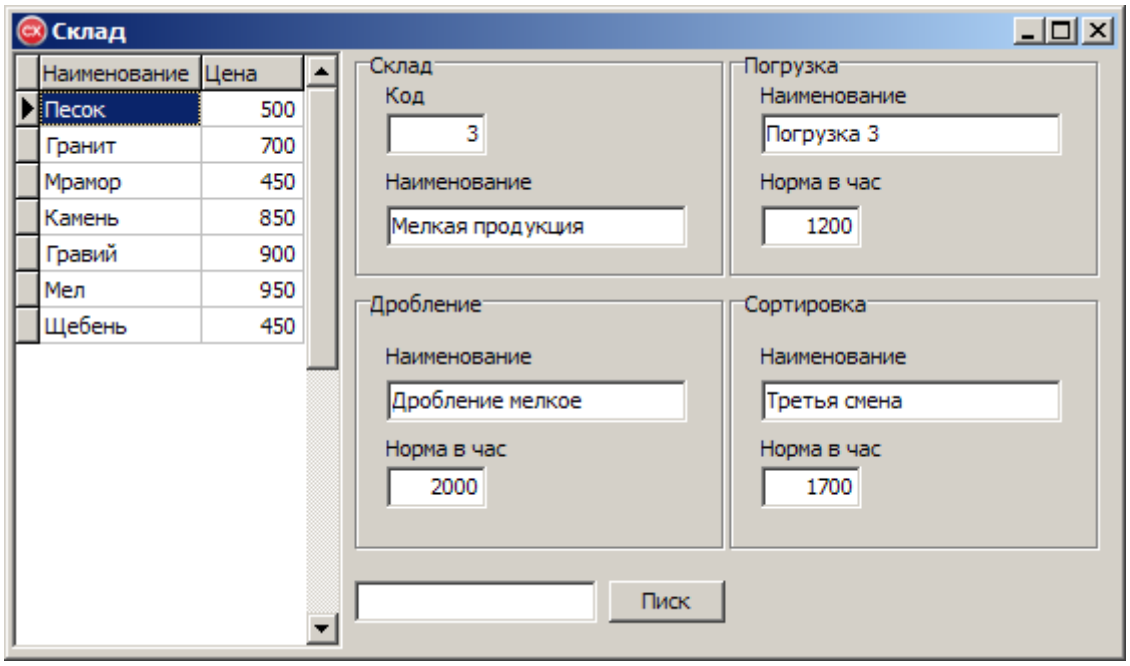

Рисунок 3.7 – форма «Склад»

В окне «Прайс-лист» есть возможность добавления новой продукции и добавления ее в прайс-лист.

Для должности согласующего доступны окна «Лист регистрации изменений» (рисунок 3.8) в которые вводятся, какие либо уведомления для сотрудников, а также доступно окно «Прием сотрудников» в которое вносится принятые сотрудники.

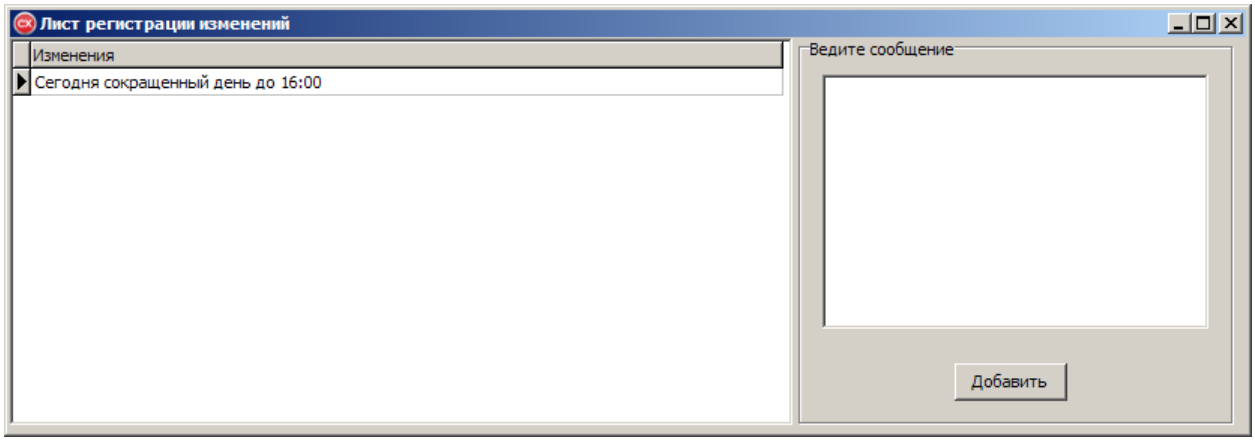

Рисунок 3.8 – форма «Лист регистрации изменений»

Для должности контролер качества доступно окно «Контроль качества» (рисунок 3.9), в данном окне сотрудник может выделить замечания по продукту который находится на осмотре и выделить полноту выработки.

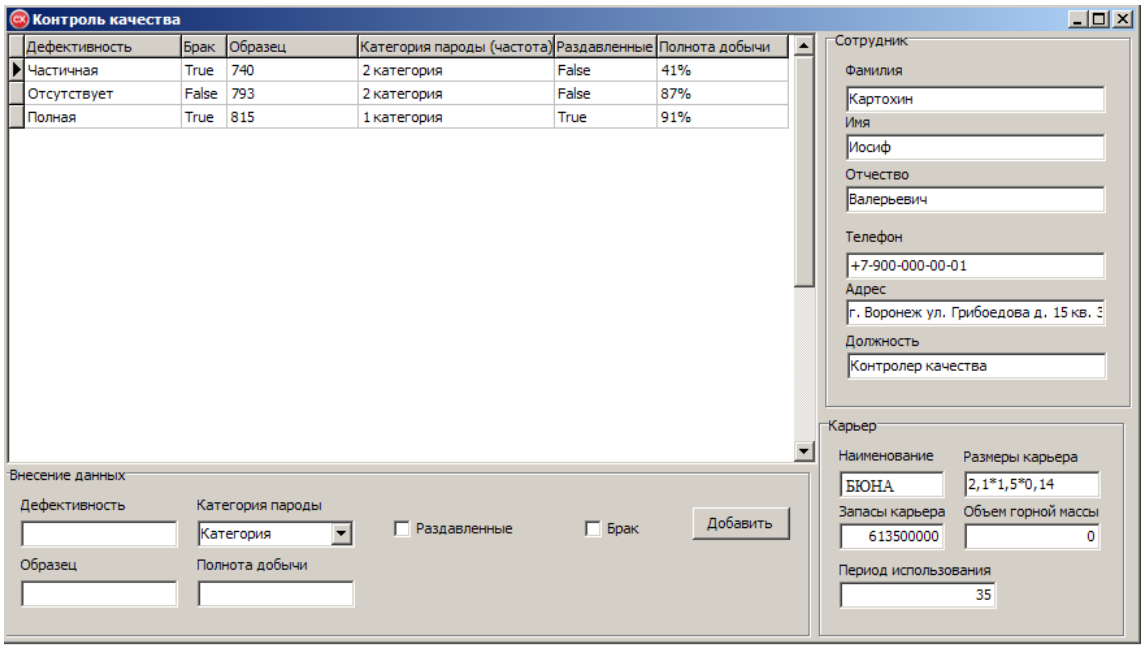

Рисунок 3.9 – форма «Контроль качества»

Для должности технический контролер доступны следующие окна: «Технический контроль», «Входной контроль», «Контроль дробления», «Лист рассылки». Форма технический контроль (рисунок 3.10) формируется из двух форм «Входной контроль», «Контроль дробления» при наведении и выборе продукта пользователю будет видно, какой контроль прошел тот или иной продукт.

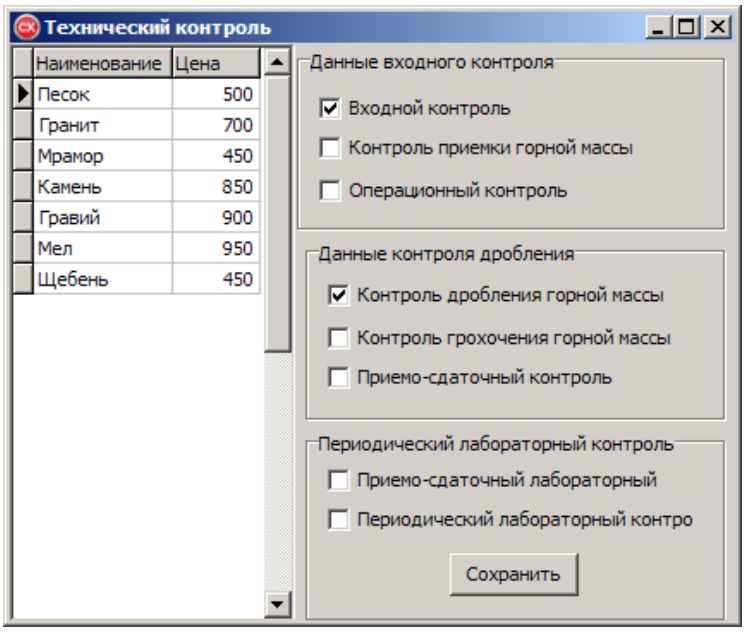

Рисунок 3.10 – форма «Технический контроль»

В форме «Входной контроль» (рисунок 3.11) вносятся проверки по отгруженной продукции.

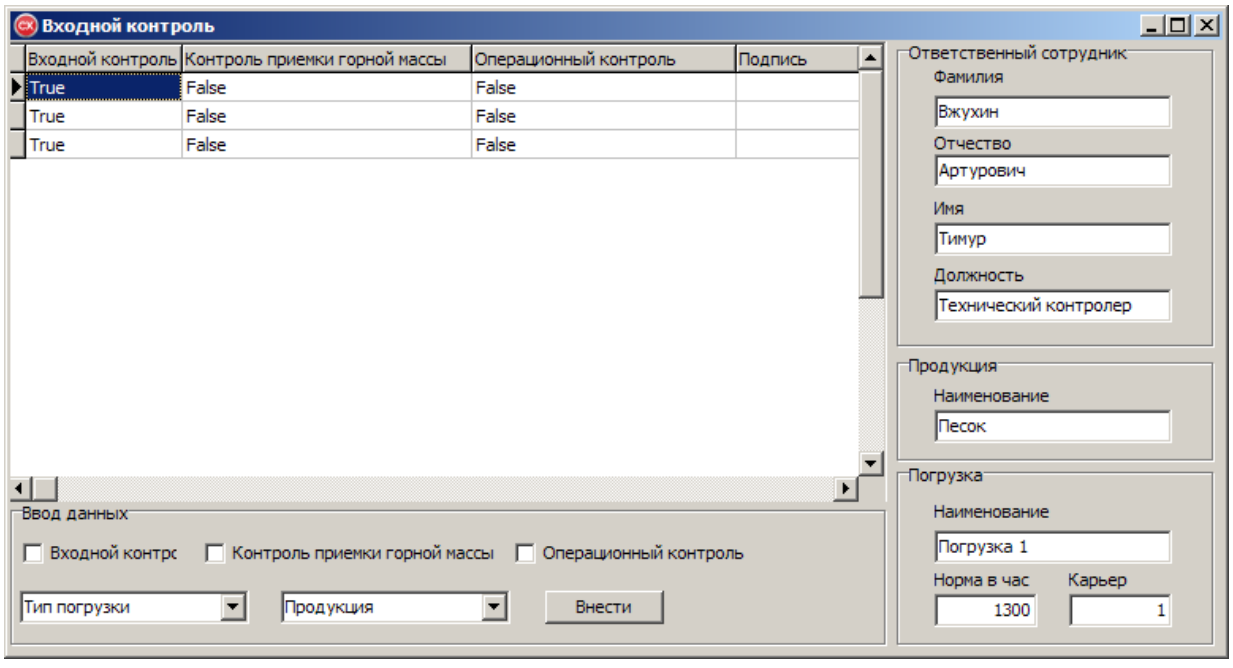

Рисунок 3.11 – форма «Входной контроль»

В форме «Контроль дробления» (рисунок 3.12) вносятся проверки по дроблению горной массы и ее степень.

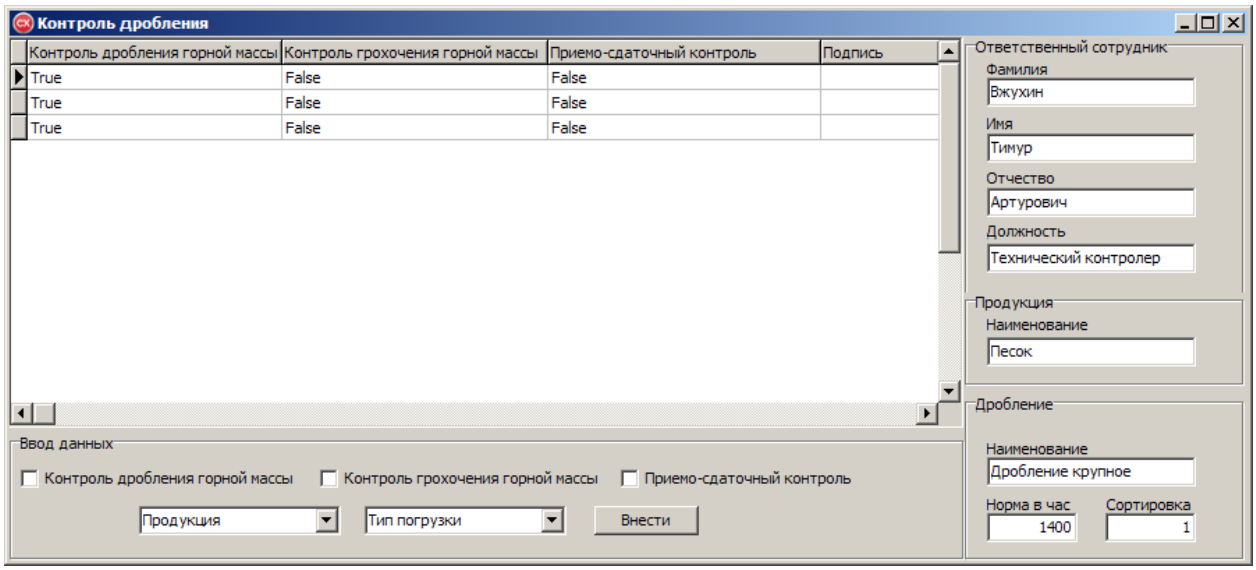

Рисунок 3.12 – форма «Контроль дробления»

В форме «Лист рассылки» (рисунок 3.13) приходят индивидуальные сообщения. При написании текста и выборе сотрудника, ему будет доступно отправлено сообщение с датой создания уведомления.

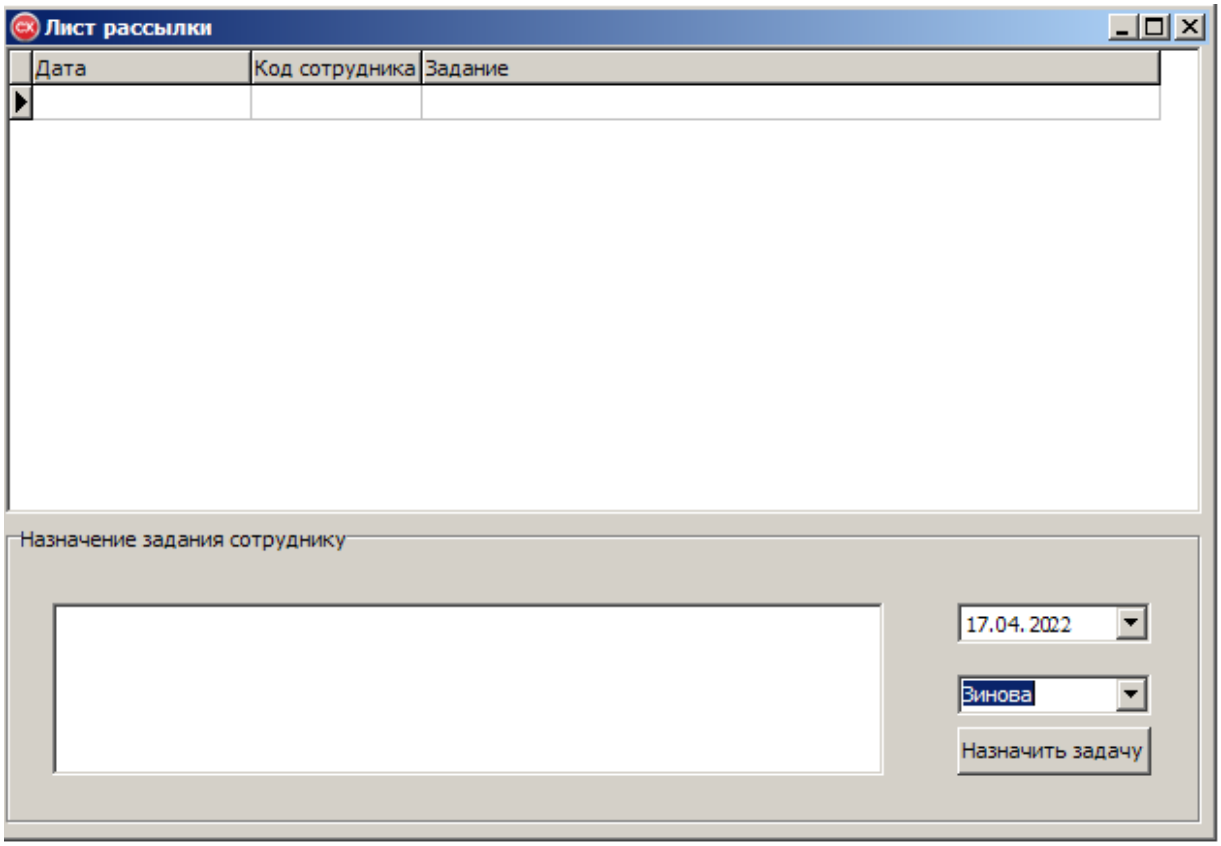

Рисунок 3.13 – форма «Лист рассылки»

Заключение

В данной работе была спроектирована и реализована информационная подсистема поддержки деятельности отдела технического контроля. Данная подсистема обеспечивает:

а) ведение базы данных контроля продукции;

б) регистрация качества;

в) определение рабочих отдела;

г) формирование прайс-листа организации.

д) формирование следующих отчетных и первичных документов:

- лист контроля;
- лист ознакомления;
- лист рассылки;
- лист регистрации изменений;
- лист согласования.

При написании программы было уделено внимание удобству работы пользователя и построению дружественного интерфейса.

Список литературы

1 Титореико Г.А. Информационные системы в экономике: учеб. для вузов / Г.А. Титоренко. - 2-е изд., перераб. и доп. - М.: Юнити-Дана, 2019. - 463с.

2 Информационные системы и технологии в экономике: учебник / Т.П. Барановская, В.И. Лойко, М.И. Семенов, А.И. Трубилин; под ред. В.И. Лойко. - 2-е изд., перераб. и доп. - М.: Финансы и статистика, 2017. - 416с.

3 Проектирование экономических информационных систем: учеб. для вузов /Г.Н. Смирнова, А.А. Сорокин, Ю.Ф. Тельнов; под ред. Ю.Ф. Тельнова. - М.: Финансы и статистика, 2020. - 512с.

4 Базы данных: учебник / А.Д. Хомоненко, В.М. Цыганков, М.Г. Мальцев; под ред. А.Д. Хомоненко. - 4-е изд., перераб. и доп. - СПб.: Корона принт, 2018. - 736с.

5 Карпова Т.С. Базы данных: модели, разработка, реализация: учеб. для вузов / Т.С. Карпова. - СПб.: Питер, 2019. - 304с.

6 Вендров А.М. Проектирование программного обеспечения экономических информационных систем / А.М. Вендров. - М.: Финансы и статистика, 2019. - 350с.

7 Грекул В.И. Проектирование информационных систем: учеб. пособие / В.И. Грекул, Г.Н. Денишенко, Н.Л. Коровкина. - М.: Интернет-Ун-т Информ технологий, 2021. - 304с.

8 Фленов М.Е. Библия Delphi: учебник/ М.Е. Фленов. - СПб.: Санкт- Петербург, 2016. - 1205с.

9 МишенинА.И. Теория экономических информационных систем: учеб. для вузов/А.И. Мишенин. - 4-е изд., доп. и перераб. - М.: Финансы и статистика, 2018. - 240с.

10 Архангельский А.Я. Программирование в Delphi / А.Я. Архангельский.- М.: Бином, 1999.-1152с.

11 Delphi 2007 http://www.delphimaster.ru/.

12 Хоменко А. Д. Delphi 7 / А.Д. Хоменко. - СПб.: БХВ-Петербург, 2018. - 1216 с.

13 База данных Borland Interbase http://www.ibprovider.com/rus/documentation/interbase.html.

14 Базы данных http://www.lessons-tva.info/edu/e-inf2/m2t4 2.html.

15 Microsoft Access http://allprogramm.com/microsoft-access-2010.html

16 Кузнецов С. Д. Основы баз данных / С.Д. Кузнецов. - 2-е изд. - М.: БИНОМ. Лаборатория знаний, 2019. - 484 с

17 ГОСТ 19.003-80. ЕСПД. Схемы алгоритмов и программ. Обозначения условные графические

Приложение А1 – Авторизация форма «Вход»

```
//---------------------------------------------------------------------------

#ifndef Unit19H
#define Unit19H
//---------------------------------------------------------------------------
#include <System.Classes.hpp>
#include <Vcl.Controls.hpp>
#include <Vcl.StdCtrls.hpp>
#include <Vcl.Forms.hpp>
#include <Data.DB.hpp>
#include <Data.Win.ADODB.hpp>
#include <Vcl.DBGrids.hpp>
#include <Vcl.Grids.hpp>
#include <Vcl.DBCtrls.hpp>
#include <Vcl.Mask.hpp>
//---------------------------------------------------------------------------
class TAutoriz : public TForm
{
__published:   // IDE-managed Components
        TLabel *Label1;
        TLabel *Label2;
        TEdit *Edit1;
        TEdit *Edit2;
        TButton *Button1;
        TADOConnection *ADOConnection1;
        TDataSource *DataSource1;
        TADOTable *ADOTable1;
        TADOQuery *ADOQuery1;
        TIntegerField *ADOTable1Код;
        TWideStringField *ADOTable1Логин;
        TWideStringField *ADOTable1Пароль;
        TIntegerField *ADOTable1Коддоступа;
        TWideStringField *ADOTable1Кодсотрудника;
```

```
        TLabel *Label3;
        TDBEdit *DBEdit1;
        void __fastcall Button1Click(TObject *Sender);
        void __fastcall FormCreate(TObject *Sender);
private:        // User declarations
public:         // User declarations
        __fastcall TAutoriz(TComponent* Owner);
};
//---------------------------------------------------------------------------
extern PACKAGE TAutoriz *Autoriz;
//---------------------------------------------------------------------------
#endif
```

Приложение А2 – форма «Режим работы»

```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "Unit1.h"
#include "Unit2.h"

#include "Unit4.h"

#include "Unit6.h"
#include "Unit7.h"
#include "Unit8.h"
#include "Unit9.h"
#include "Unit10.h"
#include "Unit11.h"

#include "Unit13.h"
#include "Unit14.h"
#include "Unit15.h"
#include "Unit17.h"
#include "Unit18.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
  int i=0;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
```

```
void __fastcall TForm1::N11Click(TObject *Sender)
{
            Form9->Show();
}
//---------------------------------------------------------------------------

void __fastcall TForm1::N12Click(TObject *Sender)
{
           Form10->Show();
}
//---------------------------------------------------------------------------

void __fastcall TForm1::N13Click(TObject *Sender)
{
          Form11->Show();
}
//---------------------------------------------------------------------------

void __fastcall TForm1::N5Click(TObject *Sender)
{
Form4->Show();
}
//---------------------------------------------------------------------------

void __fastcall TForm1::N6Click(TObject *Sender)
{
 InputControl->Show();
}
//---------------------------------------------------------------------------

void __fastcall TForm1::N7Click(TObject *Sender)
{
ControlDrob->Show();
}
//---------------------------------------------------------------------------
```

```
void __fastcall TForm1::N2Click(TObject *Sender)
{
  Form2->Show();
}
//---------------------------------------------------------------------------

void __fastcall TForm1::N4Click(TObject *Sender)
{
        Form7->Show();
}
//---------------------------------------------------------------------------

void __fastcall TForm1::N8Click(TObject *Sender)
{
        Form8->Show();
}
//---------------------------------------------------------------------------


void __fastcall TForm1::FormCreate(TObject *Sender)
{
   //Открытие БД
String filename = "db1.mdb";
if(!FileExists(ExtractFilePath(Application->ExeName)+filename))
          {
   AnsiString ds = "Файл базы данных не обнаружен \n\n\t" +filename;
          }
   else
           {
//Открытие таблицы 1
String WayToBase=ExtractFilePath(Application->ExeName)+filename;
ADOConnection1->ConnectionString = "Provider=Microsoft.Jet.OLEDB.4.0;User
ID=Admin;Data Source="+WayToBase+";Mode=Share Deny None;Jet OLEDB:System data-
base="";Jet OLEDB:Registry Path="";Jet OLEDB:Database Password="";Jet OLEDB:Engine
```

```
Type=5;Jet OLEDB:Database Locking Mode=1;Jet OLEDB:Global Partial Bulk Ops=2;Jet
OLEDB:Global Bulk Transactions=1;Jet OLEDB:New Database Password="";Jet
OLEDB:Create System Database=False;Jet OLEDB:Encrypt Database=False;Jet OLEDB:Don't
Copy Locale on Compact=False;Jet OLEDB:Compact Without Replica Repair=False;Jet
OLEDB:SFP=False;";
ADOConnection1->Connected = true;
ADOTable1->Active = true;
ADOTable1->Open();

ADOTable3->Active = true;
ADOTable3->Open();
ADOTable4->Active = true;
ADOTable4->Open();
ADOTable2->Active = true;
ADOTable2->Open();
  }
TLocateOptions SearchOptions;
SearchOptions.Clear();
SearchOptions << loPartialKey;
ADOTable3->Locate("№ сотрудника", Edit1->Text, SearchOptions);
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button1Click(TObject *Sender)
{
ADOTable4->Delete();
ADOTable4->Active = false;
ADOTable4->Active = true;
}
//---------------------------------------------------------------------------
```

Приложение А3 – форма «Прайс-лист»

```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "Unit2.h"
#include "Unit16.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm2 *Form2;
int ok, cat;
//---------------------------------------------------------------------------
__fastcall TForm2::TForm2(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TForm2::FormCreate(TObject *Sender)
 {
        //Открытие БД
String filename = "db1.mdb";
if(!FileExists(ExtractFilePath(Application->ExeName)+filename))
         {
  AnsiString ds = "Файл базы данных не обнаружен \n\n\t" +filename;
         }
  else
         {
//Открытие таблицы 1
String WayToBase=ExtractFilePath(Application->ExeName)+filename;
ADOConnection1->ConnectionString = "Provider=Microsoft.Jet.OLEDB.4.0;User
ID=Admin;Data Source="+WayToBase+";Mode=Share Deny None;Jet OLEDB:System data-
base="";Jet OLEDB:Registry Path="";Jet OLEDB:Database Password="";Jet OLEDB:Engine
Type=5;Jet OLEDB:Database Locking Mode=1;Jet OLEDB:Global Partial Bulk Ops=2;Jet
```

```
OLEDB:Global Bulk Transactions=1;Jet OLEDB:New Database Password="";Jet
OLEDB:Create System Database=False;Jet OLEDB:Encrypt Database=False;Jet OLEDB:Don't
Copy Locale on Compact=False;Jet OLEDB:Compact Without Replica Repair=False;Jet
OLEDB:SFP=False;";
ADOConnection1->Connected = true;
ADOTable1->Active = true;
ADOTable1->Open();
ADOTable2->Active = true;
ADOTable2->Open();
ADOTable3->Active = true;
ADOTable3->Open();
ADOTable4->Active = true;
ADOTable4->Open();

  }
        ComboBox1->Items->Add("1");
        ComboBox1->Items->Add("2");
        ComboBox1->Items->Add("3");

  Q1->Close();
  Q1->SQL->Clear();
  Q1->SQL->Add("SELECT Наименование FROM Продукция");
  Q1->Open();
    Q1->First();
  for(int i = 0; i < Q1->RecordCount; i++)
        {
          ComboBox4->Items->Add(Q1->FieldValues["Наименование"]);
          Q1->Next();
        }
        ComboBox2->Items->Add("Дробление крупное");
        ComboBox2->Items->Add("Дробление среднее");
        ComboBox2->Items->Add("Дробление мелкое");

        ComboBox3->Items->Add("Крупная продукция");
        ComboBox3->Items->Add("Средняя продукция");
```

```
        ComboBox3->Items->Add("Мелкая продукция");
}
//---------------------------------------------------------------------------

void __fastcall TForm2::Button2Click(TObject *Sender)
{
Form2->ADOTable2->Active = false;
Form2->ADOTable2->Active = true;
ADOTable2->Insert();

ADOTable2->FieldByName("Наименование")->Value = Edit2->Text;
//ADOTable2->FieldByName("Категория")->Value = ComboBox1->Text;
ADOTable2->FieldByName("Код дробления")->Value = ComboBox2->ItemIndex+1;
ADOTable2->FieldByName("Код склада")->Value = ComboBox3->ItemIndex+1;
ADOTable2->FieldByName("Цена")->Value = Edit3->Text;

ADOTable2->Post();

Form2->ADOTable2->Active = false;
Form2->ADOTable2->Active = true;
}
//---------------------------------------------------------------------------

void __fastcall TForm2::Button3Click(TObject *Sender)
{
ADOTable2->Delete();
}
//---------------------------------------------------------------------------

void __fastcall TForm2::Button4Click(TObject *Sender)
{

ok=cat*Q1->FieldValues["Цена"];
ADOTable1->Insert();
```

```
ADOTable1->FieldByName("Код продукта")->AsString=Q1->FieldValues["Код
продукции"];
ADOTable1->FieldByName("Наименование")->AsString=Q1-
>FieldValues["Наименование"];
ADOTable1->FieldByName("Категория")->AsString=ComboBox1->Text;
ADOTable1->FieldByName("Цена")->AsString=ok;


ADOTable1->Post();
 ADOTable1->Active = false;
 ADOTable1->Active = true;
}
//---------------------------------------------------------------------------

void __fastcall TForm2::ComboBox1Change(TObject *Sender)
{
if (ComboBox1->ItemIndex == 0 ) {
cat = 10;
}
else if (ComboBox1->ItemIndex == 1) {
          cat = 7;
           }
           else if (ComboBox1->ItemIndex == 2) {
           cat = 5;
           }

}
//---------------------------------------------------------------------------

void __fastcall TForm2::ComboBox4Change(TObject *Sender)
{
Q1->Close();
Q1->SQL->Clear();
Q1->SQL->Add("SELECT * FROM Продукция WHERE [Код продукции] = "
+IntToStr(ComboBox4->ItemIndex+1));
```

```
Q1->Open();
ok=cat*Q1->FieldValues["Цена"];
TLocateOptions SearchOptions;
SearchOptions.Clear();
SearchOptions << loPartialKey;
Form2->ADOTable2->Locate("Наименование", Q1->FieldValues["Наименование"], Sear-
chOptions);

}
//---------------------------------------------------------------------------
```

Приложение А4 – форма «Технический контроль»

```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "Unit4.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm4 *Form4;
//---------------------------------------------------------------------------
__fastcall TForm4::TForm4(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------

void __fastcall TForm4::Button1Click(TObject *Sender)
{

ADOTable3->Insert();
ADOTable3->FieldByName("Код входного контроля")->AsString=ADOTable4-
>FieldByName("Код")->Text;
ADOTable3->FieldByName("Входной контроль")->AsString=ADOTable4-
>FieldByName("Входной контроль")->Text;
ADOTable3->FieldByName("Контроль приемки горной массы")->AsString=ADOTable4-
>FieldByName("Контроль приемки горной массы")->Text;
ADOTable3->FieldByName("Операционный контроль")->AsString=ADOTable4-
>FieldByName("Операционный контроль")->Text;

ADOTable3->FieldByName("Код контроля дробления")->AsString=ADOTable5-
>FieldByName("Код")->Text;
ADOTable3->FieldByName("Контроль дробления горной массы")->AsString=ADOTable5-
>FieldByName("Контроль дробления горной массы")->Text;
```

```
ADOTable3->FieldByName("Контроль грохочения горной массы")->AsString=ADOTable5-
>FieldByName("Контроль грохочения горной массы")->Text;
ADOTable3->FieldByName("Приемо-сдаточный контроль")->AsString=ADOTable5-
>FieldByName("Приемо-сдаточный контроль")->Text;
ADOTable3->FieldByName("Код продукции")->Value =ADOTable2->FieldByName("Код
продукции")->Text;
ADOTable3->FieldByName("Код сотрудника")->Value = Edit1->Text;
if (CheckBox1->Checked) {
ADOTable3->FieldByName("Приемо-сдаточный лабораторный")->Value = true;
}
else
{
ADOTable3->FieldByName("Приемо-сдаточный лабораторный")->Value = false;
}
if (CheckBox2->Checked) {
 ADOTable3->FieldByName("Периодический лабораторный контроль")->Value = true ;
}
else
{
 ADOTable3->FieldByName("Периодический лабораторный контроль")->Value = false ;
}

ADOTable3->Post();
 ADOTable3->Active = false;
 ADOTable3->Active = true;
}
//---------------------------------------------------------------------------

void __fastcall TForm4::FormCreate(TObject *Sender)
{
        //Открытие БД
String filename = "db1.mdb";
if(!FileExists(ExtractFilePath(Application->ExeName)+filename))
          {
  AnsiString ds = "Файл базы данных не обнаружен \n\n\t" +filename;
```

```
        }
  else
        {
//Открытие таблицы 1
String WayToBase=ExtractFilePath(Application->ExeName)+filename;
ADOConnection1->ConnectionString = "Provider=Microsoft.Jet.OLEDB.4.0;User
ID=Admin;Data Source="+WayToBase+";Mode=Share Deny None;Jet OLEDB:System data-
base="";Jet OLEDB:Registry Path="";Jet OLEDB:Database Password="";Jet OLEDB:Engine
Type=5;Jet OLEDB:Database Locking Mode=1;Jet OLEDB:Global Partial Bulk Ops=2;Jet
OLEDB:Global Bulk Transactions=1;Jet OLEDB:New Database Password="";Jet
OLEDB:Create System Database=False;Jet OLEDB:Encrypt Database=False;Jet OLEDB:Don't
Copy Locale on Compact=False;Jet OLEDB:Compact Without Replica Repair=False;Jet
OLEDB:SFP=False;";
ADOConnection1->Connected = true;

ADOTable2->Active = true;
ADOTable2->Open();
ADOTable3->Active = true;
ADOTable3->Open();
ADOTable4->Active = true;
ADOTable4->Open();
ADOTable5->Active = true;
ADOTable5->Open();

  }
}
//---------------------------------------------------------------------------
```

Приложение А5 – форма «Контроль качества»

```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "Unit7.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm7 *Form7;
//---------------------------------------------------------------------------
__fastcall TForm7::TForm7(TComponent* Owner)
        : TForm(Owner)
{
        ComboBox1->Items->Add("1 категория");
        ComboBox1->Items->Add("2 категория");
        ComboBox1->Items->Add("3 категория");
}
//---------------------------------------------------------------------------

void __fastcall TForm7::Button1Click(TObject *Sender)
{
ADOTable1->Insert();

ADOTable1->FieldByName("Дефективность")->Value = Edit2->Text;
ADOTable1->FieldByName("Брак")->Value = CheckBox1;
ADOTable1->FieldByName("Образец")->Value = Edit3->Text;
ADOTable1->FieldByName("Категория пароды (частота)")->Value =ComboBox1->Text;
ADOTable1->FieldByName("Раздавленные")->Value = CheckBox2;
ADOTable1->FieldByName("Полнота добычи")->Value = Edit5->Text;
ADOTable1->FieldByName("Код карьера")->Value = "1";
ADOTable1->FieldByName("Код сотрудника")->Value = Edit1->Text;

ADOTable1->Post();
```

```
        ComboBox1->Clear();

         CheckBox1->Checked = false;
         CheckBox2->Checked = false;;
Edit2->Clear();
Edit3->Clear();
Edit5->Clear();
ADOTable1->Active = false;
ADOTable1->Active = true;
}
//---------------------------------------------------------------------------

void __fastcall TForm7::FormCreate(TObject *Sender)
{
        //Открытие БД
String filename = "db1.mdb";
if(!FileExists(ExtractFilePath(Application->ExeName)+filename))
         {
  AnsiString ds = "Файл базы данных не обнаружен \n\n\t" +filename;
         }
  else
          {
//Открытие таблицы 1
String WayToBase=ExtractFilePath(Application->ExeName)+filename;
ADOConnection1->ConnectionString = "Provider=Microsoft.Jet.OLEDB.4.0;User
ID=Admin;Data Source="+WayToBase+";Mode=Share Deny None;Jet OLEDB:System data-
base="";Jet OLEDB:Registry Path="";Jet OLEDB:Database Password="";Jet OLEDB:Engine
Type=5;Jet OLEDB:Database Locking Mode=1;Jet OLEDB:Global Partial Bulk Ops=2;Jet
OLEDB:Global Bulk Transactions=1;Jet OLEDB:New Database Password="";Jet
OLEDB:Create System Database=False;Jet OLEDB:Encrypt Database=False;Jet OLEDB:Don't
Copy Locale on Compact=False;Jet OLEDB:Compact Without Replica Repair=False;Jet
OLEDB:SFP=False;";
ADOConnection1->Connected = true;

ADOTable2->Active = true;
```

```
ADOTable2->Open();
ADOTable3->Active = true;
ADOTable3->Open();
ADOTable1->Active = true;
ADOTable1->Open();

  }
}
//---------------------------------------------------------------------------
```

Приложение А6 – форма «Лист рассылки»

```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "Unit8.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm8 *Form8;
//---------------------------------------------------------------------------
__fastcall TForm8::TForm8(TComponent* Owner)
        : TForm(Owner)
{

}
//---------------------------------------------------------------------------
void __fastcall TForm8::Button1Click(TObject *Sender)
{
ADOTable1->Insert();
ADOTable1->FieldByName("Задание")->AsString= Memo1->Text;
ADOTable1->FieldByName("Дата")->Value = DateTimePicker1-
>DateTime.FormatString("yyyy.MM.dd");
ADOTable1->FieldByName("Код сотрудника")->AsString= Q1->FieldValues["№
сотрудника"];

ADOTable1->Post();
 ADOTable1->Active = false;
 ADOTable1->Active = true;
}
//---------------------------------------------------------------------------

void __fastcall TForm8::FormCreate(TObject *Sender)
{
```

```
        //Открытие БД
String filename = "db1.mdb";
if(!FileExists(ExtractFilePath(Application->ExeName)+filename))
        {
  AnsiString ds = "Файл базы данных не обнаружен \n\n\t" +filename;
        }
  else
        {
//Открытие таблицы 1
String WayToBase=ExtractFilePath(Application->ExeName)+filename;
ADOConnection1->ConnectionString = "Provider=Microsoft.Jet.OLEDB.4.0;User
ID=Admin;Data Source="+WayToBase+";Mode=Share Deny None;Jet OLEDB:System data-
base="";Jet OLEDB:Registry Path="";Jet OLEDB:Database Password="";Jet OLEDB:Engine
Type=5;Jet OLEDB:Database Locking Mode=1;Jet OLEDB:Global Partial Bulk Ops=2;Jet
OLEDB:Global Bulk Transactions=1;Jet OLEDB:New Database Password="";Jet
OLEDB:Create System Database=False;Jet OLEDB:Encrypt Database=False;Jet OLEDB:Don't
Copy Locale on Compact=False;Jet OLEDB:Compact Without Replica Repair=False;Jet
OLEDB:SFP=False;";
ADOConnection1->Connected = true;

ADOTable2->Active = true;
ADOTable2->Open();

ADOTable1->Active = true;
ADOTable1->Open();

  }
  Memo1->Clear();
DateTimePicker1->Date = Date();
  Q1->Close();
  Q1->SQL->Clear();
  Q1->SQL->Add("SELECT Фамилия FROM Сотрудники");
  Q1->Open();
    Q1->First();
  for(int i = 0; i < Q1->RecordCount; i++)
```

```
        {
          ComboBox1->Items->Add(Q1->FieldValues["Фамилия"]);
          Q1->Next();
        }
Memo1->Clear();
ADOTable1->Last();
}
//---------------------------------------------------------------------------

void __fastcall TForm8::ComboBox1Change(TObject *Sender)
{
Q1->Close();
Q1->SQL->Clear();
Q1->SQL->Add("SELECT * FROM Сотрудники WHERE [№ сотрудника] = "
+IntToStr(ComboBox1->ItemIndex+1));
Q1->Open();
        int q = Q1->FieldValues["№ сотрудника"];
}
//---------------------------------------------------------------------------
```

Приложение А7 – форма «Регистрация изменений»

```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "Unit9.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm9 *Form9;
//---------------------------------------------------------------------------
__fastcall TForm9::TForm9(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------

void __fastcall TForm9::FormCreate(TObject *Sender)
{
        //Открытие БД
String filename = "db1.mdb";
if(!FileExists(ExtractFilePath(Application->ExeName)+filename))
         {
  AnsiString ds = "Файл базы данных не обнаружен \n\n\t" +filename;
         }
  else
          {
//Открытие таблицы 1
String WayToBase=ExtractFilePath(Application->ExeName)+filename;
ADOConnection1->ConnectionString = "Provider=Microsoft.Jet.OLEDB.4.0;User
ID=Admin;Data Source="+WayToBase+";Mode=Share Deny None;Jet OLEDB:System data-
base="";Jet OLEDB:Registry Path="";Jet OLEDB:Database Password="";Jet OLEDB:Engine
Type=5;Jet OLEDB:Database Locking Mode=1;Jet OLEDB:Global Partial Bulk Ops=2;Jet
OLEDB:Global Bulk Transactions=1;Jet OLEDB:New Database Password="";Jet
```

```
OLEDB:Create System Database=False;Jet OLEDB:Encrypt Database=False;Jet OLEDB:Don't
Copy Locale on Compact=False;Jet OLEDB:Compact Without Replica Repair=False;Jet
OLEDB:SFP=False;";
ADOConnection1->Connected = true;



ADOTable1->Active = true;
ADOTable1->Open();

  }
Memo1->Clear();
ADOTable1->Last();
}
//---------------------------------------------------------------------------

void __fastcall TForm9::ДобавитьClick(TObject *Sender)
{
ADOTable1->Insert();
ADOTable1->FieldByName("Изменения")->AsString= Memo1->Text;
ADOTable1->Post();
 ADOTable1->Active = false;
 ADOTable1->Active = true;
}
//---------------------------------------------------------------------------
```

Приложение А8 – форма «Прием сотрудника»

```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop
   #include "jpeg.hpp"
#include "Unit10.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm10 *Form10;
int q;
//---------------------------------------------------------------------------
__fastcall TForm10::TForm10(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------}
//---------------------------------------------------------------------------

void __fastcall TForm10::Button1Click(TObject *Sender)
{

if (ComboBox1->Text == "Бухгалтер") {
  q=1;
}
if (ComboBox1->Text == "Согласующий") {
  q=2;
}
if (ComboBox1->Text == "Контролер качества") {
  q=3;
}
if (ComboBox1->Text == "Технический контролер") {
  q=4;
}
```

```
    Edit1->Text = q;
ADOTable2->Insert();
ADOTable2->FieldByName("Фамилия")->AsString = Edit2->Text;
ADOTable2->FieldByName("Имя")->AsString = Edit3->Text;
ADOTable2->FieldByName("Отчество")->AsString = Edit4->Text;
ADOTable2->FieldByName("Должность")->AsString = ComboBox1->Text;
ADOTable2->FieldByName("Телефон")->AsString = Edit6->Text;
ADOTable2->FieldByName("Адрес")->AsString = Edit7->Text;
ADOTable2->Post();

ADOTable3->Insert();
ADOTable3->FieldByName("Логин")->AsString = Edit8->Text;
ADOTable3->FieldByName("Пароль")->AsString = Edit9->Text;
ADOTable3->FieldByName("Код доступа")->AsString = q;
ADOTable3->FieldByName("Код сотрудника")->AsString = ADOTable2->FieldByName("№
сотрудника")->AsString;
ADOTable2->Active = false;
ADOTable2->Active = true;
ADOTable1->Active = false;
ADOTable1->Active = true;
}
//---------------------------------------------------------------------------

void __fastcall TForm10::Button2Click(TObject *Sender)
{

ADOTable3->Post();
}
//---------------------------------------------------------------------------

void __fastcall TForm10::FormCreate(TObject *Sender)
{
        //Открытие БД
String filename = "db1.mdb";
if(!FileExists(ExtractFilePath(Application->ExeName)+filename))
```

```
        {
  AnsiString ds = "Файл базы данных не обнаружен \n\n\t" +filename;
        }
  else
        {
//Открытие таблицы 1
String WayToBase=ExtractFilePath(Application->ExeName)+filename;
ADOConnection1->ConnectionString = "Provider=Microsoft.Jet.OLEDB.4.0;User
ID=Admin;Data Source="+WayToBase+";Mode=Share Deny None;Jet OLEDB:System data-
base="";Jet OLEDB:Registry Path="";Jet OLEDB:Database Password="";Jet OLEDB:Engine
Type=5;Jet OLEDB:Database Locking Mode=1;Jet OLEDB:Global Partial Bulk Ops=2;Jet
OLEDB:Global Bulk Transactions=1;Jet OLEDB:New Database Password="";Jet
OLEDB:Create System Database=False;Jet OLEDB:Encrypt Database=False;Jet OLEDB:Don't
Copy Locale on Compact=False;Jet OLEDB:Compact Without Replica Repair=False;Jet
OLEDB:SFP=False;";
ADOConnection1->Connected = true;
ADOTable1->Active = true;
ADOTable1->Open();

ADOTable3->Active = true;
ADOTable3->Open();

ADOTable2->Active = true;
ADOTable2->Open();

  }
Edit9->PasswordChar = '*';
        ComboBox1->Items->Add("Бухгалтер");
        ComboBox1->Items->Add("Согласующий");
        ComboBox1->Items->Add("Контролер качества");
        ComboBox1->Items->Add("Технический контролер");
        }
//---------------------------------------------------------------------------
```

Приложение А9 – форма «Склад»

```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "Unit11.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm11 *Form11;
//---------------------------------------------------------------------------
__fastcall TForm11::TForm11(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------

void __fastcall TForm11::Button1Click(TObject *Sender)
{
  TLocateOptions SearchOptions;
SearchOptions.Clear();
SearchOptions << loPartialKey;
if(Form11->ADOTable5->Locate("Наименование", Edit1->Text, SearchOptions))
{
                Application->Title="Поиск";
                ShowMessage("Товар найден");
}
        else
        {
                Application->Title="Поиск";
                ShowMessage("Товар не найден");
        }
  Edit1->Clear();
}
```

```
//---------------------------------------------------------------------------
void __fastcall TForm11::FormCreate(TObject *Sender)
{
  //Открытие БД
String filename = "db1.mdb";
if(!FileExists(ExtractFilePath(Application->ExeName)+filename))
        {
  AnsiString ds = "Файл базы данных не обнаружен \n\n\t" +filename;
        }
  else
         {
//Открытие таблицы 1
String WayToBase=ExtractFilePath(Application->ExeName)+filename;
ADOConnection1->ConnectionString = "Provider=Microsoft.Jet.OLEDB.4.0;User
ID=Admin;Data Source="+WayToBase+";Mode=Share Deny None;Jet OLEDB:System data-
base="";Jet OLEDB:Registry Path="";Jet OLEDB:Database Password="";Jet OLEDB:Engine
Type=5;Jet OLEDB:Database Locking Mode=1;Jet OLEDB:Global Partial Bulk Ops=2;Jet
OLEDB:Global Bulk Transactions=1;Jet OLEDB:New Database Password="";Jet
OLEDB:Create System Database=False;Jet OLEDB:Encrypt Database=False;Jet OLEDB:Don't
Copy Locale on Compact=False;Jet OLEDB:Compact Without Replica Repair=False;Jet
OLEDB:SFP=False;";
ADOConnection1->Connected = true;
ADOTable1->Active = true;
ADOTable1->Open();
ADOTable3->Active = true;
ADOTable3->Open();
ADOTable4->Active = true;
ADOTable4->Open();
ADOTable2->Active = true;
ADOTable2->Open();
ADOTable5->Active = true;
ADOTable5->Open();
  }
}
//---------------------------------------------------------------------------
```

Приложение А10 – форма «Контроль дробления»

```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "Unit18.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TControlDrob *ControlDrob;
//---------------------------------------------------------------------------
__fastcall TControlDrob::TControlDrob(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TControlDrob::Button1Click(TObject *Sender)
{
ADOTable5->Insert();
  if (CheckBox1->Checked) {
ADOTable5->FieldByName("Контроль дробления горной массы")->Value = true ;
}
else
{
ADOTable5->FieldByName("Контроль дробления горной массы")->Value= false ;
}
        if (CheckBox2->Checked) {
ADOTable5->FieldByName("Контроль грохочения горной массы")->Value = true ;
}
else
{
ADOTable5->FieldByName("Контроль грохочения горной массы")->Value= false ;
}   if (CheckBox3->Checked) {
ADOTable5->FieldByName("Приемо-сдаточный контроль")->Value = true ;
```

```
}
else
{
ADOTable5->FieldByName("Приемо-сдаточный контроль")->Value= false ;
}

ADOTable5->FieldByName("Код сотрудника")->Value = Edit1->Text;
ADOTable5->FieldByName("Подпись")->Value = "";
ADOTable5->FieldByName("Код дробления")->Value = ComboBox1->ItemIndex+1;
 ADOTable5->FieldByName("Код продукции")->Value = Q1->FieldValues["Код
продукции"];

ADOTable5->Post();
        CheckBox1->Checked = false;
        CheckBox2->Checked = false;;
        CheckBox3->Checked = false;
ADOTable5->Active = false;
ADOTable5->Active = true;
}
//---------------------------------------------------------------------------
void __fastcall TControlDrob::FormCreate(TObject *Sender)
{
  //Открытие БД
String filename = "db1.mdb";
if(!FileExists(ExtractFilePath(Application->ExeName)+filename))
        {
  AnsiString ds = "Файл базы данных не обнаружен \n\n\t" +filename;
        }
  else
         {
//Открытие таблицы 1
String WayToBase=ExtractFilePath(Application->ExeName)+filename;
ADOConnection1->ConnectionString = "Provider=Microsoft.Jet.OLEDB.4.0;User
ID=Admin;Data Source="+WayToBase+";Mode=Share Deny None;Jet OLEDB:System data-
base="";Jet OLEDB:Registry Path="";Jet OLEDB:Database Password="";Jet OLEDB:Engine
```

```
Type=5;Jet OLEDB:Database Locking Mode=1;Jet OLEDB:Global Partial Bulk Ops=2;Jet
OLEDB:Global Bulk Transactions=1;Jet OLEDB:New Database Password="";Jet
OLEDB:Create System Database=False;Jet OLEDB:Encrypt Database=False;Jet OLEDB:Don't
Copy Locale on Compact=False;Jet OLEDB:Compact Without Replica Repair=False;Jet
OLEDB:SFP=False;";
ADOConnection1->Connected = true;
ADOTable1->Active = true;
ADOTable1->Open();

ADOTable3->Active = true;
ADOTable3->Open();
ADOTable4->Active = true;
ADOTable4->Open();
ADOTable5->Active = true;
ADOTable5->Open();

  }
        ComboBox1->Items->Add("Дробление крупное");
        ComboBox1->Items->Add("Дробление среднее");
        ComboBox1->Items->Add("Дробление мелкое");

          Q1->Close();
  Q1->SQL->Clear();
  Q1->SQL->Add("SELECT Наименование FROM Продукция");
  Q1->Open();
          Q1->First();
  for(int i = 0; i < Q1->RecordCount; i++)
        {
          ComboBox2->Items->Add(Q1->FieldValues["Наименование"]);
          Q1->Next();
        }
}
//---------------------------------------------------------------------------
void __fastcall TControlDrob::ComboBox2Change(TObject *Sender)
{
```

```
Q1->Close();
Q1->SQL->Clear();
Q1->SQL->Add("SELECT * FROM Продукция WHERE [Код продукции] = "
+IntToStr(ComboBox2->ItemIndex+1));
Q1->Open();
TLocateOptions SearchOptions;
SearchOptions.Clear();
SearchOptions << loPartialKey;
ControlDrob->ADOTable3->Locate("Наименование", Q1->FieldValues["Наименование"],
SearchOptions);
}
//---------------------------------------------------------------------------
```

Приложение А11 – форма «Входной контроль»

```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "Unit17.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TInputControl *InputControl;
//---------------------------------------------------------------------------
__fastcall TInputControl::TInputControl(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TInputControl::FormCreate(TObject *Sender)
{
        //Открытие БД
String filename = "db1.mdb";
if(!FileExists(ExtractFilePath(Application->ExeName)+filename))
         {
  AnsiString ds = "Файл базы данных не обнаружен \n\n\t" +filename;
         }
  else
         {
//Открытие таблицы 1
String WayToBase=ExtractFilePath(Application->ExeName)+filename;
InputControl->ADOConnection1->ConnectionString = "Provid-
er=Microsoft.Jet.OLEDB.4.0;User ID=Admin;Data Source="+WayToBase+";Mode=Share De-
ny None;Jet OLEDB:System database="";Jet OLEDB:Registry Path="";Jet OLEDB:Database
Password="";Jet OLEDB:Engine Type=5;Jet OLEDB:Database Locking Mode=1;Jet
OLEDB:Global Partial Bulk Ops=2;Jet OLEDB:Global Bulk Transactions=1;Jet OLEDB:New
Database Password="";Jet OLEDB:Create System Database=False;Jet OLEDB:Encrypt Data-
```

```
base=False;Jet OLEDB:Don't Copy Locale on Compact=False;Jet OLEDB:Compact Without
Replica Repair=False;Jet OLEDB:SFP=False;";
InputControl->ADOConnection1->Connected = true;

InputControl->ADOTable1->Active = true;
InputControl->ADOTable1->Open();

InputControl->ADOTable2->Active = true;
InputControl->ADOTable2->Open();



  InputControl->ADOTable3->Active = true;
InputControl->ADOTable3->Open();

InputControl->ADOTable4->Active = true;
InputControl->ADOTable4->Open();

   }
   //Provider=Microsoft.Jet.OLEDB.4.0;User ID=Admin;Data
Source=C:\Users\Пк\Documents\Embarcadero\Studio\Projects\IS TC\db1.mdb;Mode=Share
Deny None;Jet OLEDB:System database="";Jet OLEDB:Registry Path="";Jet OLEDB:Database
Password="";Jet OLEDB:Engine Type=5;Jet OLEDB:Database Locking Mode=1;Jet
OLEDB:Global Partial Bulk Ops=2;Jet OLEDB:Global Bulk Transactions=1;Jet OLEDB:New
Database Password="";Jet OLEDB:Create System Database=False;Jet OLEDB:Encrypt Data-
base=False;Jet OLEDB:Don't Copy Locale on Compact=False;Jet OLEDB:Compact Without
Replica Repair=False;Jet OLEDB:SFP=False;
          ComboBox1->Items->Add("Погрузка 1");
          ComboBox1->Items->Add("Погрузка 2");
          ComboBox1->Items->Add("Погрузка 3");

   Q1->Close();
   Q1->SQL->Clear();
   Q1->SQL->Add("SELECT Наименование FROM Продукция");
   Q1->Open();
   Q1->First();
```

```
  for(int i = 0; i < Q1->RecordCount; i++)
          {
           ComboBox2->Items->Add(Q1->FieldValues["Наименование"]);
           Q1->Next();
          }
}
//---------------------------------------------------------------------------
void __fastcall TInputControl::Button1Click(TObject *Sender)
{
ADOTable4->Insert();


 if (CheckBox1->Checked) {
ADOTable4->FieldByName("Входной контроль")->Value = true ;
}
else
{
ADOTable4->FieldByName("Входной контроль")->Value= false ;
}
 if (CheckBox2->Checked) {
ADOTable4->FieldByName("Контроль приемки горной массы")->Value = true ;
}
else
{
ADOTable4->FieldByName("Контроль приемки горной массы")->Value= false ;
}
 if (CheckBox3->Checked) {
ADOTable4->FieldByName("Операционный контроль")->Value = true ;
}
else
{
ADOTable4->FieldByName("Операционный контроль")->Value= false ;
}
ADOTable4->FieldByName("Код сотрудника")->AsString = Edit1->Text;
ADOTable4->FieldByName("Код погрузки")->AsString = ComboBox1->ItemIndex+1;
```

```
ADOTable4->FieldByName("Код продукции")->AsString = Q1->FieldValues["Код продукции"];

ADOTable4->Post();

//CheckBox1->Checked = false;
//CheckBox2->Checked = false;;
//CheckBox3->Checked = false;
ADOTable4->Active = false;
ADOTable4->Active = true;
}
//---------------------------------------------------------------------------
void __fastcall TInputControl::ComboBox2Change(TObject *Sender)
{
Q1->Close();
Q1->SQL->Clear();
Q1->SQL->Add("SELECT * FROM Продукция WHERE [Код продукции] = "
+IntToStr(ComboBox2->ItemIndex+1));
Q1->Open();
TLocateOptions SearchOptions;
SearchOptions.Clear();
SearchOptions << loPartialKey;
InputControl->ADOTable3->Locate("Наименование", Q1->FieldValues["Наименование"],
SearchOptions);

}
//---------------------------------------------------------------------------
```