Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий
Системы автоматики, автоматизированное управление и проектирование

УТВЕРЖДАЮ
Заведующий кафедрой
_____С. В. Ченцов
«____» _____2017г.

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

Разработка интегрированного решения информационной поддержки процессов производственного планирования и управления

Направление 27.04.04 Управление в технических системах
Магистерская программа 27.04.04.01Интегрированные системы управления производством

			доцент, канд. техн. наук
Научный руководитель	_____	_____.2017 г.	Д. В. Капулин
Выпускник	_____	_____.2017 г.	Р.И. Чорномаз
Нормоконтролер	_____	_____.2017 г.	Т. А. Грудинова

Красноярск 2017

РЕФЕРАТ

Магистерская диссертация на тему «Разработка интегрированного решения информационной поддержки процессов производственного планирования и управления» содержит основной текст 70 страниц, 25 иллюстраций, 12 таблиц, список использованных источников из 34 наименований, 3 приложения.

ПРОЕКТИРОВАНИЕ ИНФОРМАЦИОННЫХ СИСТЕМ, ORM, ШИНА ИНТЕГРАЦИИ.

Магистерская диссертация посвящена исследованию проблем интегрирования системы в ЕИП. В частности рассмотрено создание подсистемы, позволяющей интегрировать модуль в ЕИП, реализованный на доступе через интерфейс. Результатом данной работы является создание модуля интеграции системы в ЕИП предприятия АО «НПП «Радиосвязь».

В работе проанализированы существующие подходы к реализации методов интегрирования подсистемы в единое информационное пространство. Сформированы требования к системе; поставлены задачи на разработку модуля интеграции. Разработана программная структура и архитектура. В заключительной главе представлена реализация подсистемы.

Оглавление

ВВЕДЕНИЕ

Для любого предприятия огромное значение имеет ритмичная работа, в процессе которой на каждом рабочем месте и участке, в каждом производственном подразделении будет производиться в данную единицу времени строго определенное количество продукции. Такая работа, как правило, весьма эффективна, рациональна и обладает признаком высокой культуры производства.

Однако, как свидетельствует производственный опыт, добиться строго определенного и заранее рассчитанного ритма очень сложно. Для этого нужно обеспечить полную согласованность действий для всех структурных подразделений, обеспечить их производственную пропорциональность, постоянно отслеживать возможные сбои согласованного ритма производства и вводить поправки в его ход, если где-то на каком-то участке установленный ритм будет нарушен. Отклонения ритма от запланированного могут приводить к огромным экономическим потерям на предприятии: к простоям цехов и участков, к дополнительным затратам на восстановление нормального хода производства. Чтобы этого не происходило, каждая служба должна согласовывать свои действия со всеми подразделениями предприятия. Достигается это посредством реализации процедур оперативно-производственного планирования.

Большинство разработанных к настоящему времени методик оперативно-календарного планирования основано на упрощенных моделях задачи, что снижает их практическую значимость, либо эти методики применимы лишь для определенных специфичных условий. Поэтому существует актуальная задача разработать программный модуль интеграции построения сетевого графика в единое пространство предприятия.

Целью диссертационного исследования является обеспечение межмодульного взаимодействие с системами производственного

планирования за счет использования интерфейсов, состоящих из набора сигнатур.

Задачи диссертационного исследования:

– анализ информационных подсистем единого информационного пространства предприятия для организации последующего взаимодействия с ними;

– анализ паттернов проектирования и выбор оптимального для решения задачи обеспечения межмодульного взаимодействие с системами производственного планирования; создание

– интерфейса по получению данных из смежных информационных систем;

– развертывание прикладного программного решения на базе единого информационного пространства предприятия.

Научная новизна диссертационного исследования заключается в реализации паттерна «Фасад» для различных СУБД с применением системы Object Relation Mapping, что позволяет объединить использование различных баз данных без использования иных средств интеграции.

Методы исследования. В диссертации использованы методы системного анализа и объектно-ориентированного подхода. Проектирование алгоритмического и программного обеспечения выполнено с использованием средств языка UML. При реализации использовалась среда web-разработки ASP.NET MVC, язык программирования C#, T-SQL, сервер баз данных SQL Server.

Магистерская диссертация состоит из введения, трёх разделов, списка использованных источников и приложения.

Во введении обосновывается актуальность темы, определяется цель научно – исследовательской работы и перечень решаемых задач, излагается основная идея диссертации, перечисляются основные методы проведенных исследований.

В первом разделе рассматриваются основные принципы построения единого информационного пространства и проблемы интеграции разработанного решения в информационную структуру предприятия.

Во втором разделе представлены требования к интеграции. Далее рассматриваются шаблоны проектирования, выбор наиболее подходящего и обоснование.

В третьем разделе разработана и реализована система ORM, реализован алгоритм разузлования состава изделия. Спроектирована архитектура «Facade» для ORM на основе диаграммы UML. Представлено описание методов WEB API.

В ходе исследования были реализованы требования к разрабатываемому модулю интеграции. Описаны основные механизмы для реализации системы. На UML диаграмме представлена архитектура «Facade» и подмодуля ORM, разработана спецификация WEB API.

1 Единое информационное пространство

Единое информационное пространство (ЕИП) представляет собой совокупность баз и банков данных, технологий их ведения и использования, информационно-телекоммуникационных систем и сетей, функционирующих на основе единых принципов и по общим правилам, обеспечивающим информационное взаимодействие [1]. Создание единого информационного пространства организации – весьма актуальная задача, решение которой позволяет не только упорядочить деятельность подразделений и сотрудников, но и повысить скорость принятия решений. Технически решить эту задачу можно разными способами. Одним из них может быть интеграция используемых в организации систем на основе корпоративного портала или единого аппаратно-программного решения [2].

Единое информационное пространство складывается из следующих главных компонентов:

– информационные ресурсы, содержащие данные, сведения и знания, зафиксированные на соответствующих носителях информации;

– организационные структуры, обеспечивающие функционирование и развитие единого информационного пространства, в частности, сбор, обработку, хранение, распространение, поиск и передачу информации;

– средства информационного взаимодействия, обеспечивающие доступ к информационным ресурсам на основе соответствующих информационных технологий, включающие программно-технические средства.

Характерным свойством ЕИП является его структурированность, т. е. выделены его элементы, установлены связи между ними, введены обозначения, элементы и связи упорядочены. Свойство структурированности в разных видах информационных пространств может быть выражено в разной степени. Высокий уровень обеспечивает возможность представления информации в виде документов и манипулирования данными с помощью

программно-технических средств информационных систем. Различают пять степеней структурированности информационного пространства:

- неструктурированное информационное пространство;
- слабо структурированное;
- структурированное;
- формализовано-структурированное;
- машинно-структурированное.

В роли информационных ресурсов ЕИП могут выступать не только данные, но и различные прикладные программы. Тогда в каждой из информационных систем ЕИП часть методов обработки данных реализуется в виде приложений, доступных из других информационных систем (ИС). Например, при взаимодействии двух ИС первая пользуется сервисами, предоставляемыми второй, и как результат получает уже обработанные данные, которые могут быть подвергнуты дальнейшей обработке компонентами первой ИС. Данный подход соответствует распределенной, одноранговой архитектуре взаимодействия. Согласно этой архитектуре, любые приложения из различных ИС могут выступать как в роли клиента, так и в роли сервера по отношению друг к другу, совместно решая те или иные задачи. Такой подход минимизирует дублирование приложений. Распределение приложений по различным информационным системам позволяет добиться оптимального баланса загрузки приложений и аппаратных средств, и, следовательно, приводит к эффективному использованию информационных ресурсов систем в целом.

Знание схемы базы данных необходимо только тому приложению, которое обрабатывает данные из этой базы данных [3]. Использование клиентом сервисов, предоставляемых информационной системой-сервером и реализующих методы обработки данных, позволяет решить проблему изменения схемы удаленной базы данных. И, так как в рамках конкретных информационных систем локализованы не только данные, но и методы их

обработки, происходит существенное уменьшение затрат на администрирование, сопровождение и модификацию информационных систем, составляющих единое информационное пространство.

Таким образом, концепция единого информационного пространства характеризуется следующими особенностями:

– не зависит от аппаратных и системных программных средств;

– опирается на международные и промышленные стандарты;

– обеспечивает расширяемость системы, т.е. простоту и легкость добавления новых компонентов в существующие ИС;

– позволяет интегрировать старые функционирующие приложения в новые ИС;

– обеспечивает безопасность, надежность и отказоустойчивость;

– позволяет накапливать, тиражировать и развивать формализованные знания специалистов;

– существенно снижает суммарные затраты на создание ИС.

1.1 Основные принципы построения ЕИП

Первоочередное требование при построении единого информационного пространства – единая (электронная) форма представления данных, пригодная для хранения этих данных на машинных носителей. Формирование и развитие ЕИП предусматривает, в первую очередь, обеспечение оперативного доступа к имеющимся информационным ресурсам и проведение работ по их включению в ЕИП. Это подразумевает как перевод имеющихся данных на традиционных носителях в электронную форму, так и предоставление общего доступа к уже имеющимся в электронной форме данным [4].

Реализация ЕИП возможна при создании и последующем соблюдении стандарта на взаимодействие между собой как информационных систем, так

и их отдельных приложений. Особо необходима комплексность проведения работ по стандартизации и сертификации средств и систем информатизации на современном этапе для формирования и развития единого информационного пространства. Сертификация средств информационных технологий и систем информатизации, являясь выходным этапом оценки их качества и конкурентоспособности, требует полной нормативно-методической и инструментальной поддержки. Также необходимо внедрение международных стандартов, регламентирующих формы представления информации, протоколов связи и коммуникаций для обеспечения вхождения пользователей со своих оконечных устройств в корпоративную систему.

Таким образом, представляется следующая стратегия построения автоматизированных систем поддержки единого информационного пространства:

– создание динамических средств описания информационных объектов с возможностью расширения описания, по мере увеличения знаний об этих объектах, с адекватным отображением в структуре баз данных;

– создание системы поддержки ЕИП (целостность, репликация, управление) и эффективных средств его расширения;

– создание инструментального набора средств обработки информации по информационным объектам с возможностью его конфигурирования для обработки нескольких взаимосвязанных объектов;

– создание эффективной технологии миграции информации по ИО при расширении ЕИП на новые узлы из ИС предыдущего поколения (под узлом понимается субъект, осуществляющий обработку информации – СОИ);

– разработка технологии создания приложений, обрабатывающих связанные данные по нескольким объектам.

Основными свойствами программных средств поддержки ЕИП должны являться простота и эффективность информационного и пространственного расширения ЕИП, в т. ч. создание новых рабочих мест (принцип

масштабирования). Процесс построения такой системы начинается с формирования ядра ЕИП, что включает в себя разработку информационных объектов, их начального описания и разработку инструментальных средств. После формирования ядра ЕИП производится его развертывание по технологиям, среди которых наиболее часто используются «Сверху-вниз» и «Снизу-вверх».

Технология «Сверху-вниз» подразумевает начало построения ЕИП с формирования узлов интеграции [8]. В этом случае распространение ЕИП происходит согласно некоторой древовидной структуре. Формируется первый уровень узлов интеграции информации, который связывается «ветвями» с ядром ЕИП. Второй уровень интеграции связывается «ветвями» с предыдущим узлом – и так до формирования автоматизированного рабочего места, которое также может порождать свои узлы интеграции. Такой подход не подразумевает немедленный отказ от действующих программных средств. Однако, в случае их использования, требуется передача накопленной информации в хранилище (репозиторий) ЕИП. Кроме того, возможна разработка собственных приложений для работы с несколькими взаимосвязанными объектами в соответствии с предложенной технологией.

Технология «Снизу-вверх» имеет существенное преимущество – для старта работы ЕИП нужны средства одного или нескольких вычислительных узлов (серверов), которые позволят сформировать программно-аппаратное ядро ЕИП. На основе таких узлов обычно разворачивается система хранения данных, центр обработки, сервер системных приложений, репозиторий и т. п. Создание ядра ЕИП требует выполнения тех же мероприятий, что и при использовании технологии «Сверху-вниз». Однако, в силу ограниченного числа узлов, включаемых в ЕИП при старте наблюдается некоторое снижение общих затрат и времени реализации [5].

1.2 Информационное пространство на предприятии АО «НПП «Радиосвязь»

Информационный ресурс охватывает все подразделения и службы организации. В этом смысле можно говорить об информационном пространстве предприятия, понимая под этим термином не только информацию и средства ее обработки, но и географию информационных отношений.

Рисунок 1.1 – Единое информационное пространство предприятия

Единое информационное пространство (ЕИП) АО «НПП «Радиосвязь» представляет собой совокупность распределенных баз данных, в которых действуют единые правила хранения, обновления, поиска и передачи информации. Такая информационная среда позволяет осуществить

безбумажное взаимодействие между всеми участниками процесса подготовки производства изделия. При этом однажды созданная информация хранится в ЕИП, не дублируется и не требует каких-либо перекодировок в процессе обмена, сохраняет актуальность и целостность. На сегодняшний день в рамках ЕИП на предприятии внедрено и успешно развивается комплексное решение АСКОН на базе систем Лоцман: *PLM*, Вертикаль, Компас 3*D*. Таким образом, информационная инфраструктура предприятия имеет вид, представленный на рисунке 1.2

Рисунок 1.2 – Информационная инфраструктура АО «НПП «Радиосвязь»

Информационное пространство предприятия формируется с помощью технических средств обработки информации, компьютерной и телекоммуникационной технологии. В зависимости от формы их взаимодействия и использования можно выделить четыре основных уровня реализации ЕИП предприятия [6].

Анализ результатов внедрения *ERP*-систем на приборостроительных предприятиях, основных тенденций развития концепции *PLM*,

взаимодействия процессов конструкторско-технологической подготовки и оперативного управления производством, выявил необходимость интеграции в ЕИП предприятия функций следующих информационных систем: *CAD* (системы автоматизированного проектирования), *CAM* (системы технологической подготовки производства), *PDM* (системы хранения данных), *FRP* (системы финансового планирования), *MRP* II (системы управления производственными ресурсами), *MES* (системы оперативного производственного планирования). Учитывая рассмотренный опыт автоматизации различных предприятий, созданная информационная база комплексной автоматизированной системы управления предприятием была разработана с учетом взаимодействия *PLM*- и *ERP*-концепций и приведена на рисунке 1.3.

Рисунок 1.3 – Укрупненная структура системы управления предприятием АО «НПП «Радиосвязь»

Использованный при создании интегрированной системы автоматизации управления предприятием подход позволил подойти именно к комплексной автоматизации проектных, технологических, производственных

и финансово-учетных процессов. Такой метод создания комплексной автоматизированной системы как совокупности взаимодействующих информационных систем позволяет предусмотреть дальнейшие механизмы расширения, «бесшовной» интеграции новых систем автоматизации в единое информационное пространство предприятия. Стоит отметить, что процесс развития комплексной автоматизированной системы является постоянным, итеративным, что также поддерживается примененной методикой разработки и внедрения. Спроектированная и внедренная таким образом автоматизированная система закладывает необходимую информационную поддержку дальнейшего развития деятельности предприятия, не препятствуя, а способствуя данному процессу.

1.3 Смежные системы интеграции

Подсистема интеграции с внешними системами, предназначена для информационного взаимодействия со следующими автоматизированными системами:

- система учета конструкторской документация;
- система учета технологической документация;
- система диспетчеризации производства.

Основными задачами подсистемы является:

- прием и передача информации в электронной форме между Сетевым графиком и системой учета конструкторской документации;
- прием и передача информации в электронной форме между Сетевым графиком и системой технологической документации;
- прием и передача информации в электронной форме между сетевым графиком и системой диспетчеризации производства.

1.4 Методы интегрирования системы в единое информационное пространство

Рассмотренный опыт разработки и внедрения комплексной интегрированной информационной системы управления предприятием АО «НПП «Радиосвязь» основан на интеграции общенаучных подходов, системных принципов и общих закономерностей построения, планирования, функционирования и развития сложных многоуровневых систем. Это позволяет рассматривать процесс проектирования интегрированных информационных систем управления как процесс создания единых динамических систем взаимодействия процессов конструкторско-технологической подготовки и оперативного управления производством изделий радиоэлектронной аппаратуры.

Структура и состав интегрированной информационной системы управления должны выбираться таким образом, чтобы в перспективе было возможно реализовать комплексирование систем класса *PLM* и *ERP* в рамках единого информационного пространства предприятия. Методы и принципы организации «бесшовной» интеграции различных систем автоматизации в рамках комплексной интегрированной системы позволят перейти к автоматизации управления процессами взаимодействия с потребителем продукции, что, несомненно, выведет на новый уровень эффективности бизнес-процессы предприятия и будет способствовать дальнейшему снижению издержек и повышению конкурентоспособности выпускаемой продукции и предприятия в целом [8].

В настоящее время, информационные системы все больше проникают в систему управления предприятием. При этом специфичные программные продукты используются в соответствующих областях. В виду этого, задача интеграции существующих систем в единый ландшафт является важной, решение которой позволяет выполнять управление и учет в глобальном масштабе, избежать дублирования данных, обеспечить целостность и

непротиворечивость данных. В целом эффективность IT-инфраструктура зависит от уровня интеграции существующих систем.

Так как отдельные системы являются звеньями единой цепи, и работа последующих элементов сильно зависит от входящих данных из предыдущих систем, то мы имеем дело со сквозными процессами, которые являются основополагающими для полной автоматизации работы. В виду этого, мы должны иметь методологию для выбора способа интеграции систем с учетом собственной специфики.

В целом, процесс интеграции может быть построен с использованием:

1) Односторонней интеграции (рисунок 1.4).

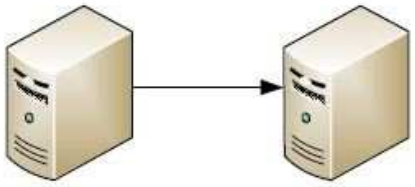


Рисунок 1.4 – Односторонняя интеграция

Применение односторонней интеграции не рекомендуется, исходя из сведений, приведенных в таблице 1.1.

2) Двухсторонней интеграции (рисунок 1.5).

Сравнение двусторонней интеграции приведено в таблице 1.2.

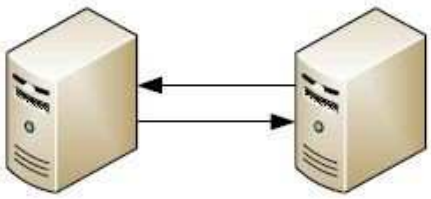


Рисунок 1.5 – Двухсторонняя интеграция

Таблица 1.1 – Достоинства и недостатки односторонней интеграции

Плюсы	Минусы
Низкие трудозатраты на реализацию.	Отсутствуют механизмы контроля. Невозможно определить статус результата процесса в системе выгрузки. В случае сбоя, получить информацию можно будет только с определенной временной задержкой, равной интервалам между сеансами выгрузки, или инициализировать вручную.

Таблица 1.2 – Достоинства и недостатки двухсторонней интеграции

Плюсы	Минусы
Реализация механизмов мониторинга. Прозрачность процесса. Стресс-устойчивость – реализуется за счет механизмов обратной связи	Высокие трудозатраты по сравнению с односторонней интеграцией. Необходимость совместимости протоколов в 2х направлениях.

В настоящее время более распространен механизм двухсторонней интеграции, что обусловлено наличием протоколов высокого уровня поддерживающих обмен информацией в обоих направлениях и так же требованиями к целостности данных. Это также иллюстрируется достоинствами, приведёнными в таблице 1.2.

Многообразие применяемых технологий и систем, разнообразие форматов данных, циркулирующих в информационных потоках, обилие аналитических и отчётных форм сделали чрезвычайно актуальной задачу интеграции технологических и информационных объектов и сущностей, а также физические и виртуальные пространства их взаимодействия в единую информационно-управленческую среду [10].

Методы интеграции системы в единое информационное пространство:

– микросервисы;

– плоский файл;

– интеграция на уровне базы данных;

– интеграция на уровне физических, программных и пользовательских интерфейсов;

– интеграция при помощи Web-сервисов.

В связи со сложностью и различностью дискретных производств, создать универсальную MES–систему – задача невыполнимая, будет либо переизбыток функций – для одного пользователя, либо недостаточная скорость обработки информации для другого. В любом случае, универсальная система – это всегда дорого для конечно пользователя, т. к. покупатель вынужден приобретать подсистему, имеющую лишние для него функции, либо должен доплачивать разработчикам за подстройку системы под свое производство. В этом ключе актуальна разработка модульной системы оперативного управления, которая может быть интегрирована в ЕИП предприятия.

Внедрение системы в единое информационно пространство предприятия является трудоемкой работой. Основная проблема внедрения системы заключается в специфичности производственного планирования.

2 Анализ требований и методов к интеграции

2.1 Компонентная архитектура системы

В состав разрабатываемой системы будут включены технологические компоненты, приведенные на рисунке 2.1.1.

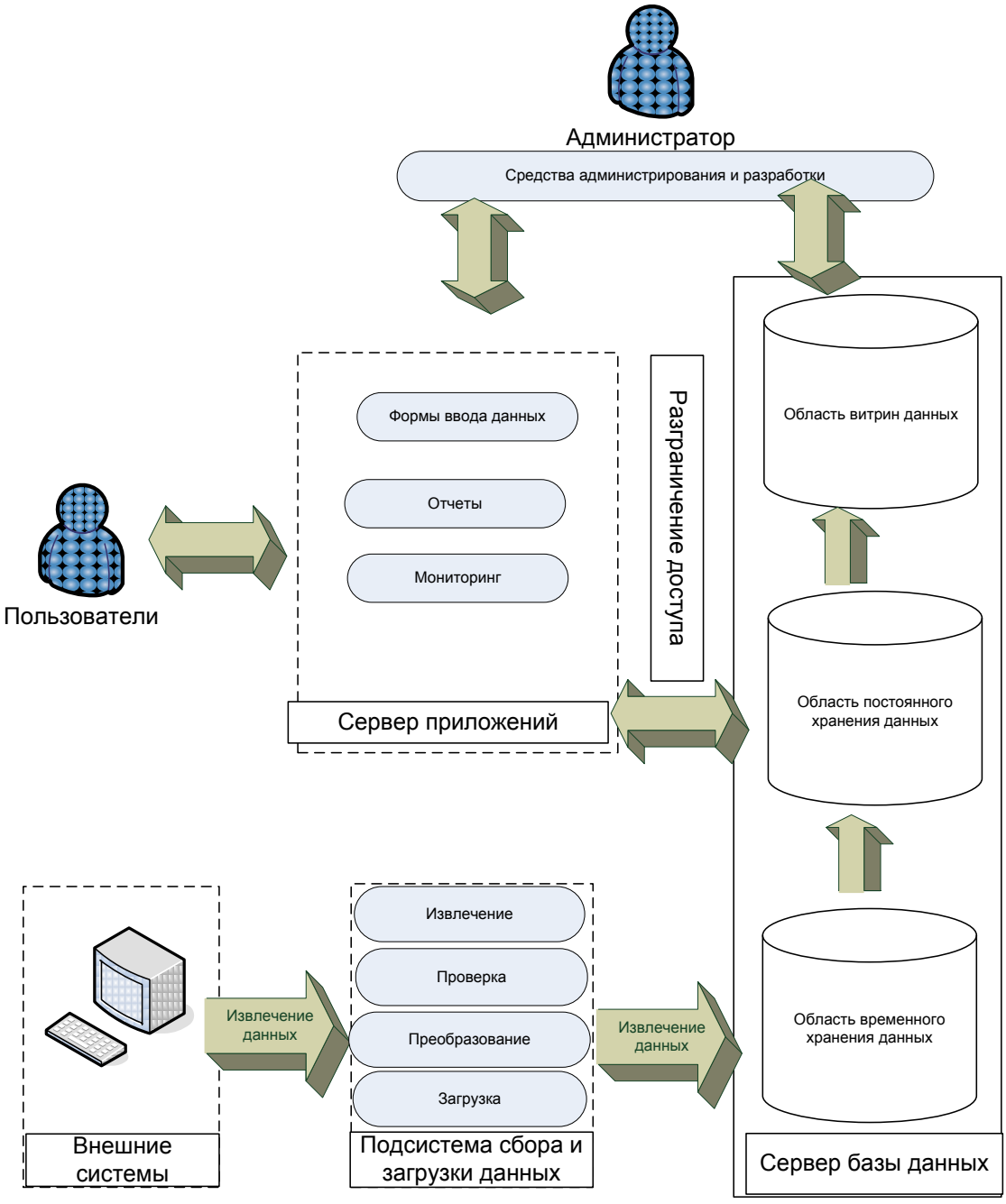


Рисунок 2.1.1 – Компонентная архитектура системы

В качестве основных технологических компонент следует выделить:

– приложение сбора и загрузки данных–решение, с помощью которого реализуются процессы извлечения, проверки, преобразования и загрузки данных из источников;

– сервер БД– представляет собой промышленную систему управления базами данных (СУБД), в которой хранятся НСИ, область временного и постоянного хранения данных, агрегаты данных. Будет реализована система разграничений прав доступа на уровне объектов и записей в таблицах. В качестве сервера БД будет использоваться Oracle DB EE 10g rel.2;

– сервер приложений – продукт, обеспечивающий поддержку промышленной инфраструктуры бизнес-приложений [11].

2.2 Основные требования к разрабатываемой подсистеме

Подсистема должна иметь возможность функционировать в штатном и аварийном режимах. Штатный режим должен являться основным режимом функционирования, обеспечивающим выполнение задач подсистемы. Аварийный режим должен являться технологическим режимом, при котором подсистема должна функционировать при необработанной ошибке в приложении либо аварии на одном из центра обработки данных (ЦОД). Переход к функционированию подсистемы в аварийном режиме должен осуществляться автоматически. В аварийном режиме система переходит в режим ожидания, до тех пор, пока проблема не будет устранена.

Комплекс механизмов защиты информации должен обеспечивать выполнение следующих функций:

– доступ к подсистемам должен предоставляться только предварительно зарегистрированным администратором системы пользователям;

– для каждого пользователя должна иметься возможность разграничения доступа к подсистемам;

– для каждого пользователя должна иметься возможность установить уровень доступа, обеспечивающий только просмотр или модификацию информации;

– разграничения и/или уровни доступа пользователей должны управляться через группы доступа;

– аутентификация и авторизация пользователей;

– регистрация входа (выхода) субъектов доступа в систему (из системы) в журнале.

В параметрах регистрации указываются:

1. Идентификатор пользователя, предъявленный при запросе доступа.

2. Регистрация в журнале безопасности попыток доступа к операциям в системе.

Уровень хранения данных в системе должен быть построен на основе современных реляционных или объектно-реляционных СУБД. Для обеспечения целостности данных должны использоваться встроенные механизмы СУБД. Доступ к данным должен быть предоставлен только авторизованным пользователям с учетом их служебных полномочий, а также с учетом категории запрашиваемой информации.

В разрабатываемом модуле должна быть обеспечена совместимость с информационным обеспечением систем, взаимодействующих с разрабатываемой системой.

Информационная база должна быть расположена на MS SQL Server R2

Технические средства, обеспечивающие хранение информации, должны использовать современные технологии, позволяющие обеспечить повышенную надежность хранения данных и оперативную замену оборудования (распределенная избыточная запись/считывание данных; зеркалирование; независимые дисковые массивы; кластеризация).

При проектировании и развертывании системы необходимо рассмотреть возможность использования накопленной информации из уже функционирующих информационных систем.

В системе должны быть предусмотрены меры по контролю и обновлению данных в информационных массивах и восстановлению массивов при сбоях технических устройств.

2.3 ORM (Object Relational Mapping)

Объектно-реляционное отображение (ORM) – технология программирования, которая связывает базы данных с концепциями объектно-ориентированных языков программирования, создавая «виртуальную объектную базу данных» (рисунок 2.2). Существуют как проприетарные, так и свободные реализации этой технологии.

Проприетарные ORM:

- Entity Framework;
- Linq To SQL.

Свободные к распространению ORM с открытым исходным кодом:

- NHibernate;
- Dapper.

Для хранения информации используются реляционные системы управления базами данных. Использование реляционной базы данных для хранения объектно-ориентированных данных приводит к семантическому разрыву, необходимость разработчикам писать программное обеспечение, которое должно уметь как обрабатывать данные в объектно-ориентированном виде, так и уметь сохранить эти данные в реляционной форме [16]. Эта постоянная необходимость в преобразовании между двумя разными формами данных не только сильно снижает производительность, но

и создает трудности для программистов, так как обе формы данных накладывают ограничения друг на друга.

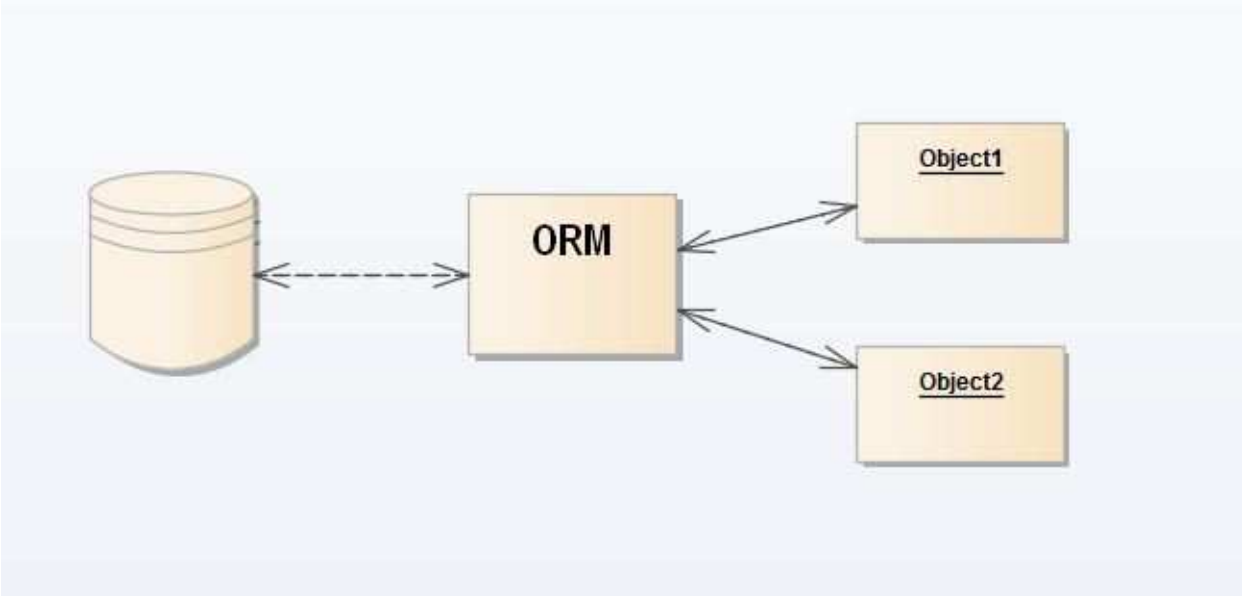


Рисунок 2.2 – Работа с ORM

Реляционные базы данных используют набор таблиц, представляющих простые данные [13]. Дополнительная или связанная информация хранится в других таблицах. Часто для хранения одного объекта в реляционной базе данных используется несколько таблиц; это, в свою очередь, требует применения операции JOIN для получения всей информации, относящейся к объекту, для её обработки.

Важной особенностью ORM является отображение, которое используется для связи объекта с его данными в БД. ORM создает «виртуальную» схему базы данных в памяти и позволяет манипулировать данными уже на уровне объектов. Отображение показывает, как объект и его свойства связанны с одной или несколькими таблицами и их полями в базе данных. ORM использует информацию этого отображения для управления процессом преобразования данных между базой и формами объектов, а также

для создания SQL-запросов для вставки, обновления и удаления данных в ответ на изменения, которые приложение вносит в эти объекты [20].

Некоторые реализации ORM автоматически синхронизируют загруженные в память объекты с базой данных. Для того чтобы это было возможным, после создания коллекции объектов из SQL-запроса (класса, реализующего связь с БД) полученные данные копируются в поля объекта, как во всех других реализациях ORM. После этого объект должен следить за изменениями этих значений и записывать их в базу данных.

Системы управления реляционными базами данных показывают хорошую производительность на глобальных запросах, которые затрагивают большой участок базы данных, но объектно-ориентированный доступ более эффективен при работе с малыми объёмами данных, так как это позволяет сократить семантический разрыв между объектной и реляционной формами хранения данных.

Если необходимо загружать или вставлять данные в SQL, требуется отобразить («замапить») .NET типы данных в SQL. Для .NET это означает использование ADO.NET для отправки SQL запросов к SQL-серверу. Затем необходимо отобразить SQL типы в .NET типы.

ADO.NET помогает использовать CRUD операции над базой данных, но не выполняет работу по обработке сырых наборов данных и созданию объектов .NET. В конечном итоге получается, что взаимодействуем с объектами и типами .NET, а ORM транслирует это в SQL запросы и обратно.

При прямом отображении сущностей таблицы базы данных соотносятся 1:1 с сущностями в системе. Когда добавляется свойство к объекту – добавляется и колонка к таблице. Использование такого способа строится вокруг загрузки сущности (или агрегата) по его идентификатору, управлению этим объектом и, возможно, связанными объектами, а затем сохранения этого объекта в базу данных посредством ORM.

ORM в этом случае предоставляет множество функционала, например:

– отслеживание изменений;

– ленивая загрузка (lazy-loading);
– предзагрузка (eager fetching);
– каскадность;
– обеспечение уникальности объектов (Identity map);
– работа с единицами работы (Unit of work).

Если работать с только одной сущностью или агрегатом одновременно, то такие ORM как NHibernate очень подходят. Они используют указанную конфигурацию для слежения за загруженными сущностями и автоматическим сохранением изменений во время коммита транзакции. И это является достоинством, потому что нет необходимости следить за слоем работы с данными. NHibernate выполняет эти функции [21].

Загрузка объекта по идентификатору с целью внесения изменений позволяет избежать большого количества кода, который бы потребовался чтобы следить за добавлением объектов, их сохранением или изменением.

Обратная сторона такого подхода в том, что ORM не знает, какую операцию необходимо будет провести: только читать объекты или загружать сущность, чтобы изменить её.

Если загрузить сущность чтобы её изменить и сохранить изменения (или создать новую сущность), этот подход обеспечивает большую гибкость от включения уровня доступа к данным в инфраструктурный слой и позволяет типам сущностей быть относительно независимыми от их метода сохранения. Эта независимость не означает, что модель C# и схема данных могут расходиться. Напротив, это означает, что слой доступа к данным не проникнет в объектную модель, которую можно нагрузить наборами условий.

Основной недостаток в использовании ORM – потеря производительности. Это происходит потому, что создается дополнительный слой абстракции для взаимодействия между приложением и базой данных.

2.4 Варианты реализации интегрированных программных решений

2.4.1 Сервис-ориентированная интеграция

В настоящее время при формировании информационной инфраструктуры предприятия, при проектировании и реализации всё чаще применяется сервис-ориентированная архитектура (Service-Oriented Architecture – SOA). Это такая архитектура информационной системы, в которой система строится из набора гетерогенных слабосвязанных компонентов (сервисов). SOA понимается как парадигма организации и использования распределенного множества функций, которые могут контролироваться различными владельцами. Базовыми понятиями в такой архитектуре являются «информационная услуга» и «композитное приложение».

Информационная услуга (сервис) – это атомарная прикладная функция автоматизированной системы, пригодная для использования при разработке приложений, реализующих прикладную логику автоматизируемых процессов как в самой системе, так и для использования в приложениях других автоматизированных систем.

Сервис обычно характеризуется следующими свойствами:

– возможность многократного применения;

– сервис может быть определен одним или несколькими технологически независимыми интерфейсами;

– выделенные услуги слабо связаны между собой и каждая из них может быть вызвана независимо от других посредством коммуникационных протоколов, обеспечивающих возможность взаимодействия сервисов между собой.

2.4.2 Интеграция на уровне данных

Интеграция данных распределенных информационных систем обеспечивает работу всех бизнес-приложений придприятия с единым массивом информации и, тем самым, позволяет формировать сводную аналитическую отчетность в масштабах всего предприятия.

Одной из главных проблем интеграции данных является многообразие форматов и типов (неструктурированные, частично-структурированные, жёстко-структурированные) данных, а также лавинообразное нарастание их объёмов. Циркулирование разнородных массивов данных и информации в сетях различных служб предприятия создает множество проблем с их сбором, структурированием, обработкой, анализом, хранением, архивированием и передачей пользователю для принятия решения.

Если база данных не экранирована хранимыми процедурами или триггерами и не имеет необходимых ограничений целостности, то информация из разных приложений могут приводить данные в противоречивые состояния.

Если база данных экранирована и целостность обеспечивается, то в этом случае, в параллельно работающих с одной БД приложениях, будут дублирующиеся части кода, выполняющие одинаковые или похожие операции. Кроме того, при изменениях структуры базы необходимо будет отдельно переписывать код всех приложений, с ней работающих.

Преимущества данного подхода заключаются в быстрой разработке такого решения интеграции.

2.4.3 Интеграция на уровне физических, программных и пользовательских интерфейсов

Этот вид интеграции начинался как один из видов «лоскутной интеграции», когда требовалось объединить разрозненные программные приложения, написанные в разное время разными разработчиками, в подобие единого целого. Приложения объединялись по принципу «каждый с каждым», что, в конечном счёте, усложняло их взаимодействие и создавало массу проблем. Кроме того, всё сложнее становилось использовать унаследованные (Legacy Software) и встроенные (Embedded System) системы.

Такой подход хорош для небольшого количества приложений. При большом их числе он практически не работает и не позволяет строить качественно новые запросы к агрегированным данным, т.е. существенного выигрыша от объединения данных нет. В настоящее время проблема интеграции на уровне интерфейсов решается на базе использования информационных подсистем, реализованных стандартными программными приложениями с открытыми интерфейсами (Open Application Programming Interface).

2.4.4 Плоский файл

Основной принцип – системы экспортируют данные, пригодные для импорта в другие системы. В качестве формата данных, как правило, выбирают плоские файлы или XML файлы.

Преимуществом плоских файлов:

- универсальность;
- отсутствие обработки при импорте;
- преимущества XML;
- наличие стандартов, описывающих формат документа;

– поддержка Unicode;
– наличие средств разбора;
– наличие средств преобразования документа;
– платформно-независимы.

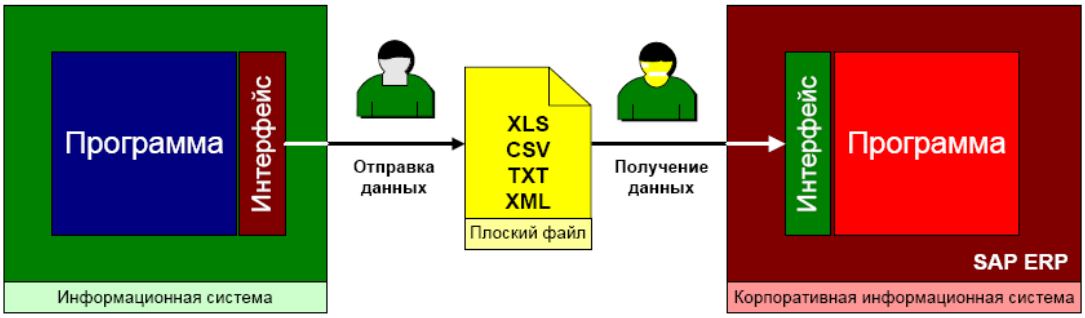


Рисунок 2.3 – Концептуальная модель интеграции на основе плоских файлов

В настоящее время использование XML файлов становится новым стандартом, в виду его официальной поддержки такими производителей ПО как Microsoft, Oracle [16], SUN и т. д. Данные производители включают в свои продукты средства работы с XML файлами и обеспечивают поддержку версий.

Недостатками данного типа обмена данными является преобразование форматов файлов, которые должны быть адаптированы для совместимости. Далее написание инструментов мониторинга процесса для обеспечения контроля, если данные инструменты отсутствуют. В целом, главным плюсом данного типа обмена является простота, так как все современные системы могут принимать файлы. Главным минусом является ненадежность и асинхронность, так как когда файл выгружен, целевая система узнает об этом только после опроса директории, а в случае потери файла, придется принудительно инициализировать повторный экспорт.

Так же стоит учесть, что время на обращение к файловой системе при частом опросе директории будет значительно уменьшать скорость работы.

2.4.5 Web- сервисы

Web(API)-сервисы – это реализация абсолютно четких интерфейсов обмена данными между различными приложениями, которые написаны не только на разных языках, но и распределены на разных узлах сети, идентифицируемая веб-адресом. API является важной абстракцией, описывающей функциональность «в чистом виде». [15] Сервис определяет функциональность, которую предоставляет программа (модуль, библиотека), при этом API позволяет абстрагироваться от того, как именно эта функциональность реализована.

На сегодняшний день наибольшее распространение получили следующие протоколы реализации веб-сервисов:

– SOAP (Simple Object Access Protocol) – объединение SOAP/WSDL/UDDI;

– REST (Representational State Transfer);

– XML-RPC (XML Remote Procedure Call).

К преимуществам веб-сервисов можно отнести следующее:

– Веб-сервисы позволяют интегрировать свои бизнес-процессы с бизнес-процессами других систем;

– Веб-сервисы обеспечивают преемственность в отношении уже имеющихся в компании ИС, то есть веб-сервисы надстраиваются над существующими ИС, но не вместо них;

– Построение новых корпоративных решений с применением веб-сервисов реализуется быстрее, поскольку основное внимание сосредотачивается на создании бизнес-логики решения, программирование самих веб-сервисов лишь по необходимости «обрамляет» этот процесс, не

требуя больших трудозатрат за счет эффективного применения повторно используемого кода и адаптированных средств разработки (IDE и SDK).

К недостаткам можно отнести снижение производительности из-за расположения сервисов и увеличение сетевого трафика по сравнению с другими технологиями.

Веб-сервисы выполняются на серверах приложений. Сегодня практически все веб-серверы приложений поддерживают эту технологию:

– Axis и Tomcat (оба являются проектами Apache);

– Microsoft .NET серверы от Microsoft;

– Java Web Services Development Pack (JWSDP) от Sun Microsystems (основан на Jakarta Tomcat);

– Oracle Application Server от Oracle Corporation.

Обычно при разработке сервис-ориентированной архитектуры обмен данными между сервером и клиентом осуществляется с использованием формата данных JSON.

JSON (*JavaScript Object Notation*) – текстовый формат обмена данными, основанный на JavaScript. Как и многие другие текстовые форматы, JSON легко читается людьми.

Несмотря на происхождение от JavaScript (точнее, от подмножества языка стандарта ECMA-262 1999 г.), формат считается независимым от языка и может использоваться практически с любым языком программирования. Для многих языков существует готовый код для создания и обработки данных в формате JSON.

За счёт своей лаконичности по сравнению с XML, формат JSON может быть более подходящим для сериализации сложных структур. Если говорить о веб-приложениях, в таком ключе он уместен в задачах обмена данными как между браузером и сервером (AJAX), так и между самими серверами (программные HTTP-сопряжения). Поскольку формат JSON является

подмножеством синтаксиса языка JavaScript, то он может быть быстро сериализован/десериализован встроенными функциями.

2.5 Шаблоны проектирования

Шаблон проектирования или паттерн – повторимая архитектурная конструкция, представляющая собой решение проблемы проектирования в рамках некоторого часто возникающего контекста.

Паттерны проектирования не являются готовыми решениями, которые можно трансформировать непосредственно в код, а представляют общее описание решения проблемы, которое можно использовать в различных ситуациях. Большинство паттернов проектирования предназначены для получения расширяемости системы в определенной плоскости. Причем эта плоскость может быть полезной для одного приложения и вредной – для другого. Наличие иерархии наследования может добавлять сложности простому приложению, но в случае библиотеки нередко делает решение чересчур сложным.

Существуют несколько типов паттернов проектирования, каждый из которых предназначен для решения своего круга задач:

– Порождающие паттерны, предназначенные для создания новых объектов в системе;

– Структурные паттерны, решающие задачи компоновки системы на основе классов и объектов;

– Паттерны поведения, предназначенные для распределения обязанностей между объектами в системе.

Паттерн описывают схемы детализации программных подсистем и отношений между ними, при этом они не влияют на структуру программной системы в целом и сохраняют независимость от реализации языка программирования. Под паттернами проектирования объектно-

ориентированных систем понимается описание взаимодействия объектов и классов, адаптированных для решения общей задачи проектирования в конкретном контексте[23].

2.5.1 Фасад

Фасад – структурный шаблон проектирования, позволяющий скрыть сложность системы путем сведения всех возможных внешних вызовов к одному объекту, делегирующему их соответствующим объектам системы.

При проектировании сложных систем, зачастую применяется т.н. принцип декомпозиции, при котором сложная система разбивается на более мелкие и простые подсистемы. Причем, уровень декомпозиции (ее глубину) определяется исключительно при проектировании. Благодаря такому подходу, отдельные компоненты системы могу быть разработаны изолированно, затем интегрированы вместе. Однако возникает, проблема – высокая связность модулей системы. Это проявляется, в первую очередь, в большом объеме информации, которой модули обмениваются друг с другом.

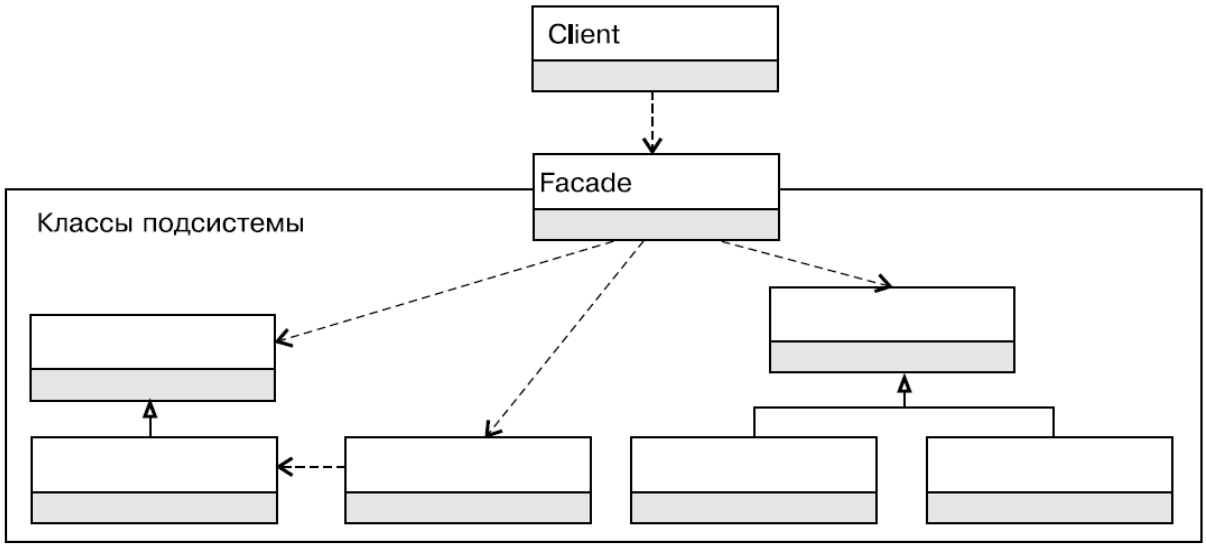


Рисунок 2.4 – Классическая диаграмма классов паттерна «Фасад»

К тому же, для подобной коммуникации одни модули должны обладать достаточной информацией о природе других модулей.

Назначение: предоставляет унифицированный интерфейс вместо набора интерфейсов некоторой подсистемы. Фасад определяет интерфейс более высокого уровня, который упрощает использование подсистемы.

– facade (SqlServerFacade) – фасадный класс, который прячет детали реализации подсистемы от клиентов;

– client – клиент фасада, который работает с фасадом, а не с классами подсистемы.

Использование фасадов не только упрощает использование библиотек или сторонних компонентов, но и решает ряд насущных проблем:

– повторное использование кода и лучших практик. Многие библиотеки довольно сложные, поэтому их корректное использование требует определенных навыков. Инкапсуляция работы с ними в одном месте позволяет корректно использовать их всеми разработчиками;

– переход на новую версию библиотеки. При выходе новой версии библиотеки достаточно будет протестировать лишь фасад;

– переход с одной библиотеки на другую. Благодаря фасаду приложение не так сильно завязано на библиотеку, так что переход на другую библиотеку потребует лишь создание еще одного фасада.

2.5.2 Заместитель (Proxy)

Заместитель – структурный шаблон проектирования, который предоставляет объект, который контролирует доступ к другому объекту, перехватывая все вызовы (выполняет функцию контейнера).

При проектировании сложных систем, достаточно часто возникает необходимость обеспечить контролируемый доступ к определенным объектам системы. Мотивацией для этого служит ряд приобретаемых

преимуществ. Таких как, ленивая инициализация по требованию для «больших» объектов, подсчет количества ссылок на объект и т.д. и т.п. Однако, не всегда потребность в контролируемом доступе к объекту базируется только на преимуществах.

Идея паттерна «Заместитель» заключается в предоставлении клиенту другого объекта (заместителя), взамен объекту с контролируемым доступом. При этом, объект-заместитель, реализует тот-же интерфейс, что и оригинальный объект, в результате чего, поведение клиента не требует изменений. Иными словами, клиент взаимодействует с заместителем ровно как с оригинальным объектом посредством единого интерфейса. Клиент, так же, не делает предположений о том работает ли он с реальным объектом или его заместителем. Контролирование доступа к объекту, при этом, достигается за счет использования ссылки на него в заместителе, благодаря которой заместитель переадресовывает внешние вызовы контролируемому объекту, возможно сопровождая их дополнительными операциями.

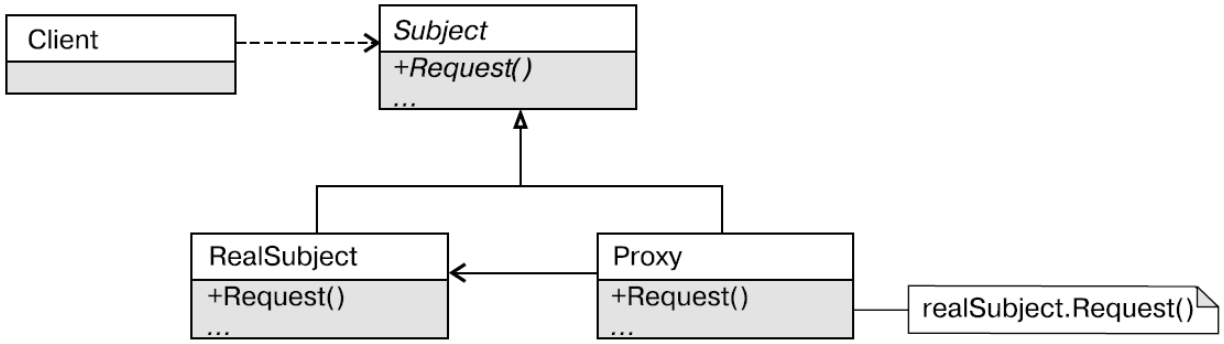


Рисунок 2.5 – Классическая диаграмма классов паттерна «Заместитель»

– client – работает с абстрактным компонентом, не зная, является он настоящим или нет;

– subject (ILogSaver) – определяет интерфейс компонента;

– proxy (LogSaverClient) – объект-заместитель, который реализует интерфейс компонента, но делегирует всю работу настоящему объекту;

– realSubject – реальный компонент, доступ к которому осуществляется через заместитель.

В классическом виде паттерн состоит из трех видов сценария использования:

– удаленный заместитель отвечает за кодирование запроса и его аргументов для работы с компонентом в другом адресном пространстве;

– виртуальный заместитель может кэшировать дополнительную информацию о реальном компоненте, чтобы отложить его создание;

– защищающий заместитель проверяет, имеет ли вызывающий объект необходимые для выполнения запроса права.

Второй и третий варианты паттерна «Заместитель» применяются на практике, но своей известностью этот паттерн обязан первому варианту. Классы-заместители являются стандартным паттерном в подавляющем большинстве современных технологий построения распределенных приложений.

2.5.3 Адаптер

Адаптер преобразует интерфейс одного класса в интерфейс другого, который ожидают клиенты. Такой шаблон делает возможной совместную работу классов с несовместимыми интерфейсами.

Далеко не все системы обладают прекрасным дизайном. Даже если модуль или приложение были хорошо продуманы изначально, внесение изменений разными разработчиками в течение длительного времени может привести к неприятным последствиям. Одно из таких последствий – рассогласованность реализации однотипных задач.

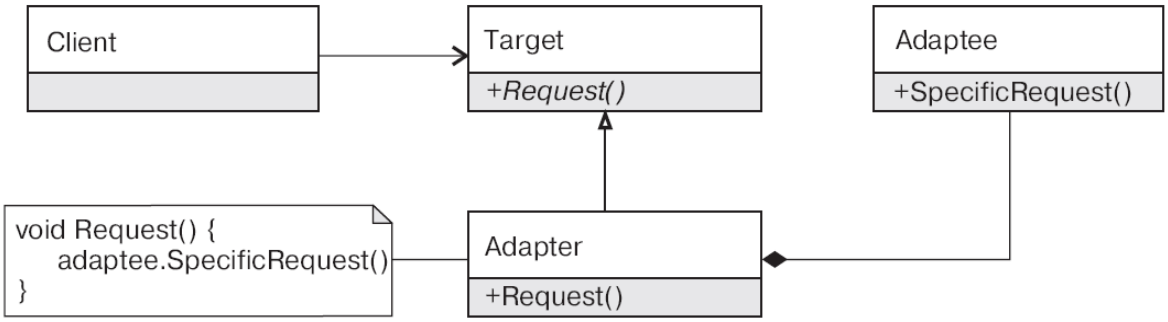


Рисунок 2.6 – Классическая диаграмма классов паттерна «Адаптер»

– Target (ILogSaver) – целевой интерфейс, к которому нужно преобразовать интерфейс существующих классов;

– Adaptee (SqlServerLogSaver) – существующий класс, чей интерфейс нужно преобразовать;

– Adapter (SqlServerLogSaverAdapter) – класс-адаптер, который преобразует интерфейс адаптируемого класса к целевому;

– .Client (клиенты ILogSaver) – клиенты нового интерфейса, которые работают с адаптированными классами полиморфным образом.

Классический адаптер классов использует множественное наследование. Адаптер реализует новый интерфейс, но также использует адаптируемый класс в качестве базового класса. Обычно в этом случае используется закрытое наследование, что предотвращает возможность конвертации адаптера к адаптируемому объекту.

Наследование от адаптируемого объекта позволяет получить доступ к защищенному представлению, а также реализовать лишь несколько методов, если новый интерфейс не слишком отличается от интерфейса адаптируемого класса.

Адаптер позволяет повторное использование чужого кода в случаях, когда уже есть код, который решает нужную задачу, но его интерфейс не подходит для текущего приложения. Вместо изменения кода библиотеки можно создать слой адаптеров. Так же шаблон позволяет плавно изменять

существующую функциональность путем выделения нового «правильного» интерфейса, но с использованием старой проверенной функциональности.

В настоящие время существует большой выбор средств и методов интегрирования разнородных систем в ЕИП АО «НПП «Радиосвязь». Сравнительный анализ методов интеграции приведен в таблице 2.1.

Таблица 2.1 Сравнительная характеристика методов интеграции

Признак	Плоский файл	БД	Web - интерфейс
Безопасность передачи данных	-	-	+
Простота интегрирование с другими системами	+	+	+
Скорость передачи данных	-	+	+

Стоит отметить, что на предприятии существуют системы собственной разработки по планирования и диспетчеризации производства [4]. Используемые системы взаимодействуют между собой на уровне базы данных. Это позволяет эффективно работать с одними и теми же данными из разных систем. Недостатки такого подхода заключаются в одновременном изменении одних и тех же данных.

3 Реализация программного обеспечения модуля интеграции

В основе разработки программного обеспечения, как и любой отрасли промышленного производства, лежит технологический процесс. Уровень качества программного продукта, его рыночная стоимость, сроки создания определяются тем, насколько удобна используемая технология разработки программы и насколько точно она соблюдается. В связи увеличением размеров программ потребовалось применение различных подходов программирования.

Процедурный подход потребовал структурирования будущей программы, разделения её на отдельно выполняемые процедуры. При реализации процедуры о других процедурах необходимо знать только их назначение и способ вызова. Появилась возможность удалять, изменять и добавлять отдельные процедуры, не затрагивая остальной части программы, сокращая при этом затраты труда на разработку и модернизацию программ.

Следующим шагом в детальном структурировании программ стало такое понятие как структурное программирование, при котором программа в целом и отдельные процедуры рассматривались как цепочка канонических конструкций: линейных участков, условий, циклов и разветвлений. Появилась возможность читать и проверять программу как последовательный текст, что повысило эффективность при проектировании, разработке, профилировании и отладке программ. С целью повышения структурности программы были выдвинуты требования к большей независимости подпрограмм, подпрограммы должны связываться с вызывающими их программами только путем передачи им аргументов в конструктор, использование в подпрограммах переменных, принадлежащих другим процедурам или главной программе, стало считаться нежелательным.

Процедурное и структурное программирование затронули прежде всего процесс описания алгоритма как последовательности операций, ведущих от варьируемых исходных данных к искомому результату. Для решения

специальных задач стали разрабатываться языки программирования, ориентированные на конкретный класс задач: на системы управления базами данных(СУБД), имитационное моделирование и т.д [26].

Далее совершенствование аппаратных средств и многократное усложнение программного обеспечения обусловило появление так называемого объектно-ориентированного программирования, ставшего большим шагом вперед в развитии программирования. Однако, опыт программирования показывает, что любой методологический подход в технологии программирования не должен применяться слепо с игнорированием других подходов. Это относится и к объектно-ориентированному подходу, называемому объектной моделью. Основными его принципами являются: абстрагирование, инкапсуляция, полиморфизм, наследование, модульность, типизация. Каждый из этих принципов сам по себе не нов, но в объектной модели они впервые применены в совокупности.

Абстракция выделяет основные характеристики некоторого объекта, отличающие его от всех других объектов и, таким образом, четко определяет его концептуальные границы с точки зрения наблюдателя. Абстрагирование концентрирует внимание на внешних свойствах объекта и позволяет отделить самые существенные особенности поведения от несущественных.

Инкапсуляция – это процесс отделения друг от друга свойств и методов объекта, определяющих его характеристики и поведение; инкапсуляция служит для того, чтобы изолировать контрактные обязательства абстракции от их реализации.

Модульность – это свойство системы, которая была разложена на внутренне связные, но слабо связанные между собой модули. Таким образом, принципы абстрагирования, инкапсуляции и модульности являются взаимодополняющими. Объект логически определяет границы определенной абстракции, а инкапсуляция и модульность делают их физически неизменяемыми.

Наследование – способность объекта расширять свои характеристики за счет свойств и методы родителя. Такой подход позволяет избегать дублирование одинаковых свойств и выстраивать иерархию классов.

Типизация является способом защиты от использования объектов одного класса вместо другого, или по крайней мере, управлять таким использованием. Типизация заставляет выражать абстракции так, чтобы язык программирования, используемый в реализации, поддерживал соблюдение принятых проектных решений.

Объект автоматизации спроектирован и представлен в виде модуля, который предназначен для интеграции с внешними системами по приему и передачи информации в электронной форме по телекоммуникационным каналам связи между системами ЕИП АО «НПП «Радиосвязь» и системой построения сетевого графика.

3.1 Компонент для работы с базой данных

Для доступа к базе данных предусмотрен интерфейс, который скрывает все тонкости работы с различными базами данных. Интерфейс разработан согласно паттерну «Фасад» (рис 3.1).

Для всех типов запросов определен один публичный родительский класс Dispatcher. Класс Dispatcher содержит свойства: ConnectionString – строка соединения с базой данных, TypeDataBase – тип используемой базы данных. Свойства соединений к базе данных доступно только для чтения, значение типа базы данных присваивается в момент создания экземпляра класса.

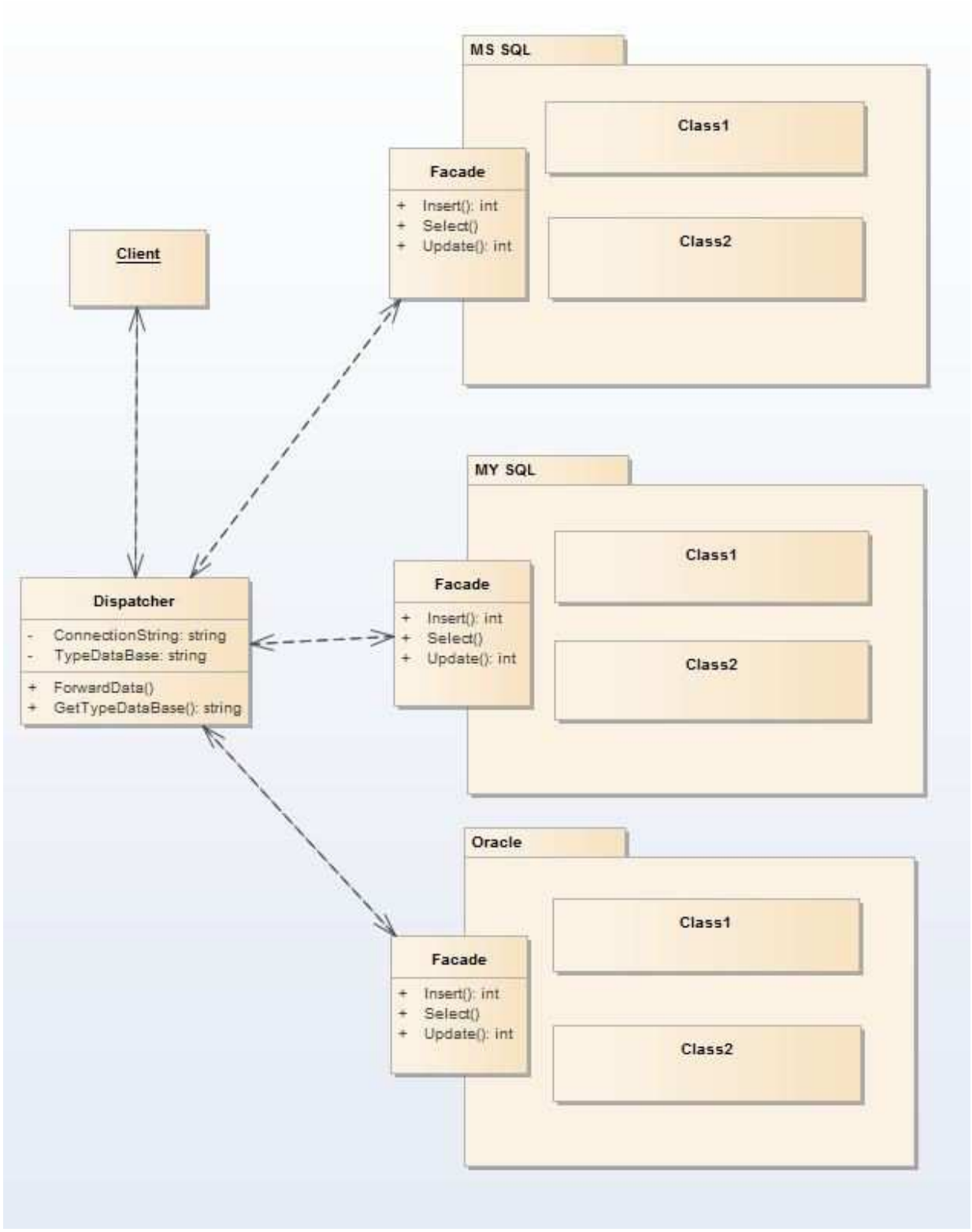


Рисунок 3.1 – Реализация паттерна «Фасад».

3.2 Формирование массивов входных данных

Для формирования входных данных, необходимо использовать табличные представления, на основе которых формируются запросы.

Для формирования запроса были использованы тестовые таблицы базы данных предприятия:

Soprpoz. Сопроводительная карта (рисунок 3.2).

	id_soprpoz	Obozn	naim	material	kol_det	norm_ras	firstC
1	9323495		ВХОДНОЙ КОНТРОЛЬ ФЛАНЦЕВЫХ КОРПУСОВ И ФЛАНЦЕВ		0	0	15
2	9323496	АВДЮ2030082	УСИЛИТЕЛЬ ПРД5-300		0	0	15
3	9323497	АВДЮ2030094	УСИЛИТЕЛЬ ПРД6-300		0	0	15
4	9323498	АВДЮ2030115	УСИЛИТЕЛЬ ПРД8-300		0	0	15
5	9323499	АИСТ757532006-02	ПРОКЛАДКА	ГЕТИНАКС V 0.5 ГОСТ2718-74	1	0,00109999999403954	15
6	9323500	СПИК741121013	ПЛАСТИНА	СТЕКЛОТЕКСТОЛИТ СТЕКЛОТЕКСТОЛИТ СТЭФ-I IC-3....	1	0,0930000022053719	15
7	9323501	СПИК755471001	СТЕКЛО	СТЕКЛО СТЕКЛО ТОСП 2 ICOPT БЕСЦВЕТНОЕ ПРОЗРА...	1	0,00630000000819564	15
8	9323502	Н4400001	ПЕТЛЯ НЕРАЗЪЕМНАЯ		0	0	15
9	9323503	Н4406005	ЗАПОР		0	0	15
10	9323504	ЖЭ4804029	СЕРДЕЧНИК		0	0	15
11	9323505	ИХ7841857-01	ПРОКЛАДКА	ГЕТИНАКС ГЕТИНАКС Х0.5 ГОСТ2718-74	65	0,000130000000353903	15
12	9323506	УЭ1110-3686	КАБЕЛЬ		0	0	15
13	9323507	УЭ1110-3686-01	КАБЕЛЬ		0	0	15
14	9323508	УЭ2017044-01	ПРИБОР У200-Л1		0	0	15
15	9323509	УЭ2032149	МОДУЛЬ УДМ-Н		0	0	15
16	9323510	УЭ2032149-01	МОДУЛЬ УДМ-Н		0	0	15
17	9323511	УЭ2032149-02	МОДУЛЬ УДМ-Н		0	0	15

Рисунок 3.2 – Таблица БД SoprPoz

Таблица 3.1 – Описание таблицы БД SoprPoz

Название	Тип	Обозначение
Id_soprpoz	int	Первичный ключ
Obozn	nvachar(4)	Обозначение
Naim	nvachar (100)	Имя детали
Material	nvachar (220)	Материал
Kol_det	int	Количество деталей
Norm_ras	float	Норма расхода
FirstC	nvachar(2)	Первичный цех

SoprOper. Таблица, в которой указан список операций, цехов и профессий, необходимых для изготовления блока (рисунок 3.3).

	id_soproper	id_soprpoz	oper	c	prof	naim_oper
1	85296100	9323495	0010	15	686	ПОДГОТОВИТЕЛЬНАЯ
2	85296101	9323495	0020	15	686	СБОРКА
3	85296102	9323495	0030	15	206	ВХОДНОЙ КОНТРОЛЬ
4	85296103	9323495	0040	15	686	СБОРКА
5	85296104	9323495	0050	15	206	КОНТРОЛЬНАЯ ОТК
6	85296105	9323495	0060	15	686	СБОРКА
7	85296106	9323495	0070	15	266	МАРКИРОВАНИЕ
8	85296107	9323495	0080	15	206	КОНТРОЛЬНАЯ
9	85296108	9323496	0010	15	686	ПОДГОТОВИТЕЛЬНАЯ
10	85296109	9323496	0020	15	582	ВСПОМОГАТЕЛЬНАЯ (ДЛЯ РЕГУЛИРОВЩИКА)
11	85296110	9323496	0030	15	686	СБОРКА
12	85296111	9323496	0040	15	206	КОНТРОЛЬНАЯ (ВХОДНОЙ КОНТРОЛЬ)
13	85296112	9323496	0050	15	582	РЕГУЛИРОВОЧНАЯ
14	85296113	9323496	0060	15	582	РЕГУЛИРОВОЧНАЯ
15	85296114	9323496	0070	15	201	КОНТРОЛЬНАЯ
16	85296115	9323496	0080	15	206	КОНТРОЛЬНАЯ ОТК, ПЗ
17	85296116	9323496	0090	15	582	РЕГУЛИРОВОЧНАЯ ОТК, ПЗ

Рисунок 3.3 – Таблица БД SoprOper

Таблица 3.2 – Описание таблицы БД SoprOper

Название	Тип	Обозначение
Id_soproper	int	Первичный ключ
Id_soprpoz	int	Внешний ключ
Oper	nvachar(4)	Операция
C	nvachar(2)	Цех
Prof	nvachar(3)	Шифр профессии
Naim_oper	nvachar(100)	Наименование данной операции

NAPROF. Таблица, в которой содержаться шифр и наименование профессии (рисунок 3.4).

	L	NAZVO
1	006	АВТ.ЛИТЬЕ
2	007	АВТОМАТ
3	008	АВ.ХОЛ.ВЫС
4	010	АККУМУЛЯТ.
5	013	АПП.ПЛАЗМ.
6	015	АПП.ОЧ.ВОД
7	018	АПП-СУШИЛ.
8	020	АП. ХИМВОД
9	030	БЕТОНЩИК
10	033	БУНКЕРОВ.
11	037	ВОДИТЕЛЬ
12	043	ВОД. ПОГР.
13	048	ВУЛКАНИЗ.
14	049	ВЫБИВ.ЛИТ.
15	050	КАБ.ШНУР

Рисунок 3.4 – Таблица БД NAPROF

Таблица 3.3 – Описание таблицы БД NAPROF

Название	Тип	Обозначение
L	nvachar(3)	Шифр профессии
NAZVO	nvachar(10)	Наименование профессии

TEXNORM. Таблица, в которой содержится информация о технологических маршрутах и нормах расходов материалов по операциям, цехам и блокам (рисунок 3.4).

	TIP	IND	PICH	DATE	C	Y	F	SH	SO	KC	KR	VO	TS	KP	TZ	W	L	R	E1	E2	NV	T	obozn
1	Б	БПРА	6275001	2011-01-01 00:00:00.000	16	NULL	10	NULL	NULL	NULL	NULL	NULL	21	NULL	NULL	3	686	3	6	6	0,055333	0,055333	6275001
2	Б	БПРА	6275001	2011-01-01 00:00:00.000	16	NULL	20	NULL	NULL	NULL	NULL	NULL	21	NULL	NULL	3	686	2	5	5	0,15333	0,15333	6275001
3	Б	БПРА	6275001	2011-01-01 00:00:00.000	16	NULL	30	NULL	NULL	NULL	NULL	NULL	21	NULL	NULL	3	206	2	6	6	0,015333	0,015333	6275001
4	Б	БПРА	6275001	2011-01-01 00:00:00.000	44	6	40	NULL	NULL	NULL	NULL	NULL	21	NULL	NULL	3	265	3	6	6	0,055833	0,055833	6275001
5	Б	БПРА	6275001	2011-01-01 00:00:00.000	44	6	50	NULL	NULL	NULL	NULL	NULL	21	NULL	NULL	3	191	3	7	7	0,0026316	0,0026316	6275001
6	Б	УЭ	2000159-03	2013-05-15 00:00:00.000	16	NULL	1150	NULL	NULL	NULL	NULL	NULL	21	NULL	NULL	3	686	4	0	0	10	10	2000159-03
7	Б	УЭ	4450011-01	2011-09-30 00:00:00.000	5	23	30	NULL	NULL	NULL	NULL	NULL	21	NULL	NULL	3	194	3	7	7	0,0013333	0,0013333	4450011-01
8	Б	УЭ	5032166	2011-01-01 00:00:00.000	45	NULL	360	NULL	NULL	NULL	NULL	NULL	21	NULL	NULL	3	200	3	6	6	0,025	0,025	5032166
9	Б	АШХ	6679417	2014-04-08 00:00:00.000	44	NULL	10	NULL	NULL	NULL	NULL	NULL	21	NULL	NULL	3	528	3	5	5	0,0173	0,0173	6679417
10	Б	АШХ	6679417	2014-04-08 00:00:00.000	44	NULL	20	NULL	NULL	NULL	NULL	NULL	21	NULL	NULL	1	199	3	5	5	0,001	0,001	6679417
11	Б	АШХ	6679417	2014-04-08 00:00:00.000	44	NULL	30	NULL	NULL	NULL	NULL	NULL	21	NULL	NULL	3	528	2	5	5	0,017	0,017	6679417
12	Б	АШХ	6679417	2014-04-08 00:00:00.000	44	NULL	40	NULL	NULL	NULL	NULL	NULL	21	NULL	NULL	3	528	3	5	5	0,316	0,316	6679417
13	Б	АШХ	6679417	2014-04-08 00:00:00.000	44	NULL	50	NULL	NULL	NULL	NULL	NULL	25	NULL	NULL	3	528	3	5	5	0,035	0,035	6679417
14	Б	АШХ	6679417	2014-04-08 00:00:00.000	44	NULL	60	NULL	NULL	NULL	NULL	NULL	21	NULL	NULL	3	528	2	5	5	0,02	0,02	6679417
15	Б	АШХ	6679417	2014-04-08 00:00:00.000	44	NULL	70	NULL	NULL	NULL	NULL	NULL	21	NULL	NULL	3	457	3	5	5	0,09	0,09	6679417
16	Б	АШХ	6679417	2014-04-08 00:00:00.000	44	NULL	80	NULL	NULL	NULL	NULL	NULL	21	NULL	NULL	1	457	4	5	5	0,216	0,216	6679417
17	Б	АШХ	6679417	2014-04-08 00:00:00.000	44	NULL	90	NULL	NULL	NULL	NULL	NULL	21	NULL	NULL	1	199	3	5	5	0,03	0,03	6679417

Рисунок 3.5 – Таблица БД TEXNORM

Таблица 3.4 – Описание таблицы БД TEXNORM

Название	Тип	Обозначение
TIP	nvachar(1)	Тип
IND	nvachar(4)	Индекс
PICH	nvachar(39)	Обозначение
DATE	datetime	Дата
C	float	Цех
Y	float	Не используется в расчетах
F	float	Порядковый номер операции
SH	float	Не используется в расчетах
SO	float	Не используется в расчетах
KC	float	Не используется в расчетах
KR	float	Не используется в расчетах
VO	float	Не используется в расчетах
TS	float	Не используется в расчетах
KP	float	Не используется в расчетах
TZ	float	Не используется в расчетах
W	float	Не используется в расчетах
L	float	Шифр профессии
R	float	Не используется в расчетах
E1	float	Не используется в расчетах
E2	float	Не используется в расчетах
Название	Тип	Обозначение
NV	float	Трудоемкость
T	float	Трудоемкость без ОТК
obozn	nvachar(39)	Обозначение (нормализованное)

tempPOSPRIMB. Таблица результатов разузлования (рисунок 3.5).

	id_record	TIP	IND1	PICH	IND2	P2NI	CS	RP	DP	Z	NS	KSP	KSZ	OP	Depth	id	Parent
1	39739	Б	УЭ	3032081	0000	00000000000	NULL	NULL	NULL	207620	43	1	1	NULL	0	Branch0	NULL
2	39740	Б	УЭ	2746004	УЭ	3032081	NULL	NULL	NULL	207620	32	1	1	NULL	1	Branch0	39739
3	39741	Б	УЭ	4835013-01	УЭ	3032081	NULL	NULL	NULL	207620	39	1	1	NULL	1	Branch0	39739
4	39742	Б	УЭ	5032113	УЭ	3032081	NULL	NULL	NULL	207620	0	1	1	NULL	1	Branch0	39739
5	39743	Б	УЭ	5032114	УЭ	3032081	NULL	NULL	NULL	207620	0	1	1	NULL	1	Branch0	39739
6	39744	Б	УЭ	6115604	УЭ	3032081	NULL	NULL	NULL	207620	0	1	1	NULL	1	Branch0	39739
7	39745	Б	УЭ	6185346	УЭ	3032081	NULL	NULL	NULL	207620	32	1	1	NULL	1	Branch0	39739
8	39746	Д	УЭ	7834594-38	УЭ	3032081	NULL	NULL	NULL	207620	31	0	0	NULL	1	Branch0	39739
9	39747	Д	УЭ	7841400-04	УЭ	3032081	NULL	NULL	NULL	207620	31	0	0	NULL	1	Branch0	39739
10	39748	Д	УЭ	8130195-02	УЭ	3032081	NULL	NULL	NULL	207620	32	0	0	NULL	1	Branch0	39739
11	39749	Д	УЭ	8130213-01	УЭ	3032081	NULL	NULL	NULL	207620	31	0	0	NULL	1	Branch0	39739
12	39750	Д	УЭ	8186386	УЭ	3032081	NULL	NULL	NULL	207620	32	0	0	NULL	1	Branch0	39739
13	39751	Д	УЭ	8605045-01	УЭ	3032081	NULL	NULL	NULL	207620	31	0	0	NULL	1	Branch0	39739
14	39752	Д	УЭ	8640164-07	УЭ	3032081	NULL	NULL	NULL	207620	0	1	1	NULL	1	Branch0	39739
15	39753	Д	УЭ	8685401-12	УЭ	3032081	NULL	NULL	NULL	207620	0	1	1	NULL	1	Branch0	39739

Рисунок 3.6 – tempPOSPRIMB

Таблица 3.5 – Описание таблицы БД tempPOSPPRIMB

Название	Тип	Обозначение
Id_record	int	Первичный ключ
TIP	nvachar(1)	Тип
IND1	nvachar(4)	Индекс1
PICH	nvachar(47)	Блок
IND2	nvachar(4)	Индекс2
P2NI	nvachar(11)	Список головных блоков
CS	float	Не используется в расчетах
RP	float	Не используется в расчетах
DP	float	Не используется в расчетах
Z	nvachar(47)	Номер заказа
NS	int	Номер серии
KSP	float	Количество на блок
KSZ	float	Количество на заказ
OP	float	Не используется в расчетах
Depth	int	Уровень вложенности
Id	nvachar(128)	Идентификатор приложения
Parent	int	Ссылка на головной элемент

После выбора необходимых таблиц, формируется запрос, который свяжет их согласно схеме приведенной на рисунке 3.7.

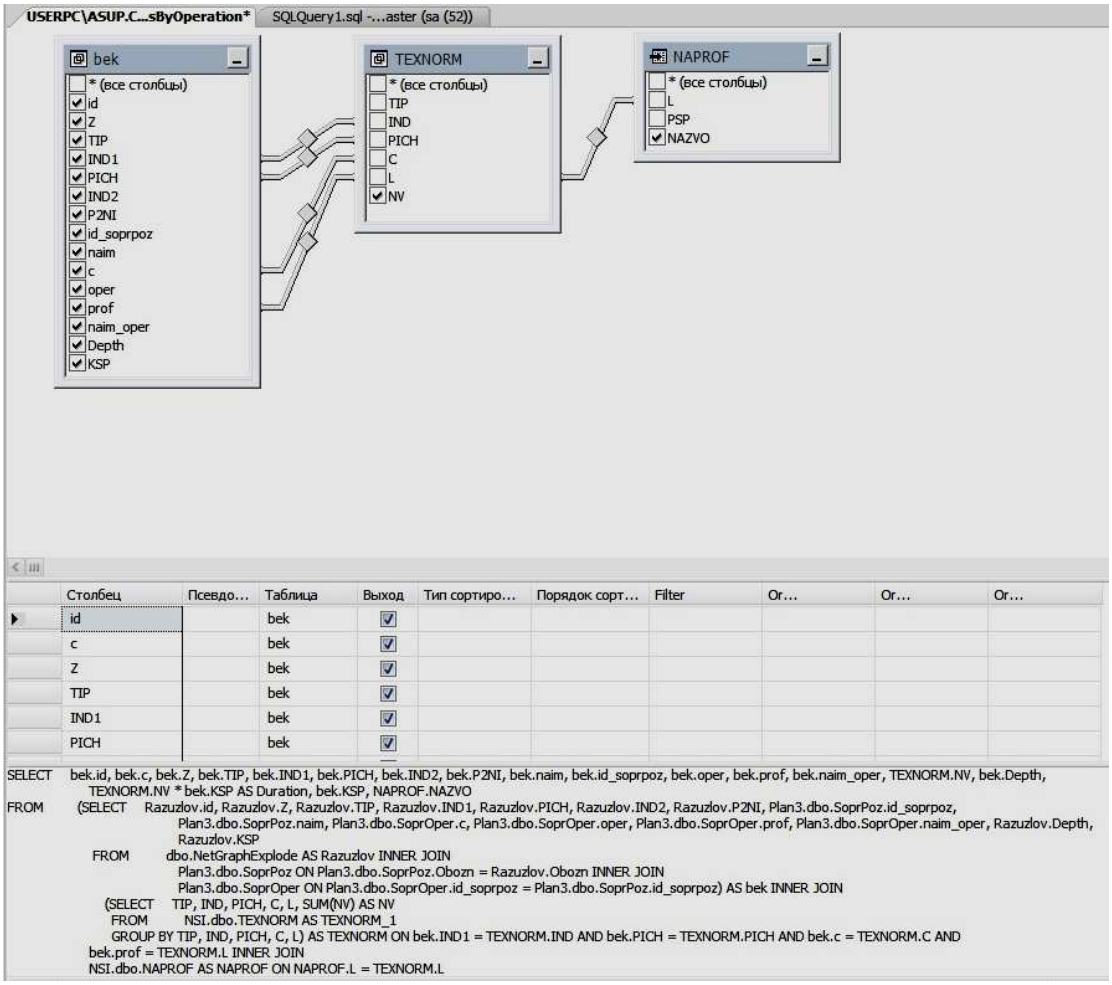


Рисунок 3.7 – Формирование запроса BlocksByOpeartion

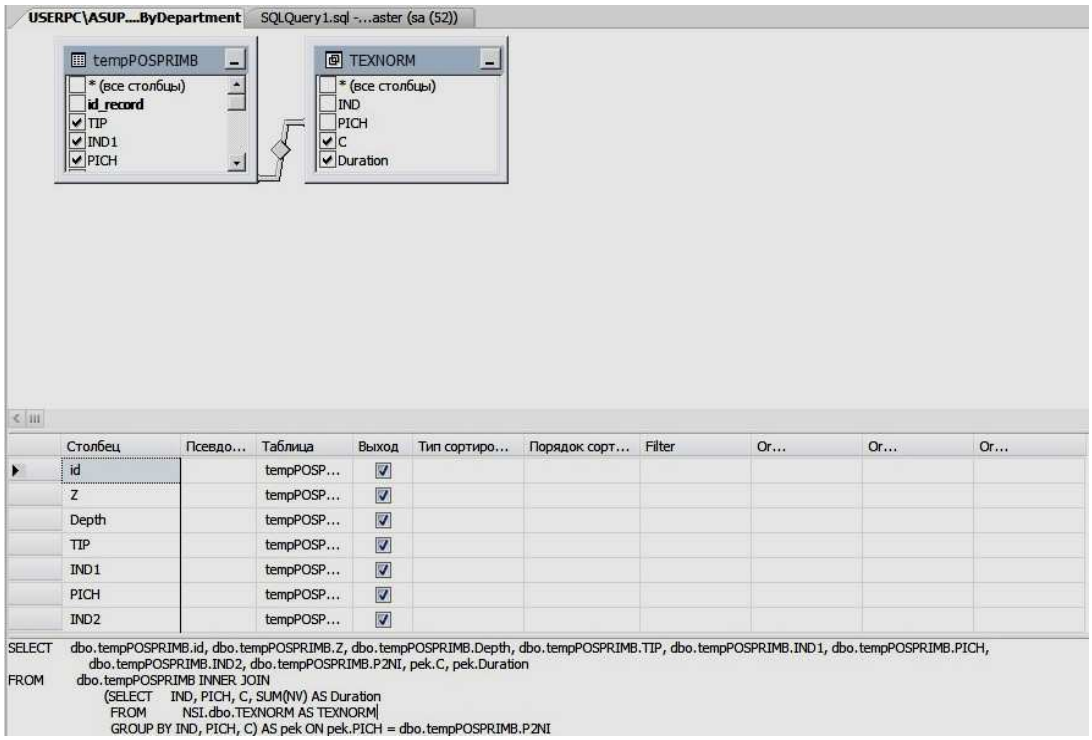
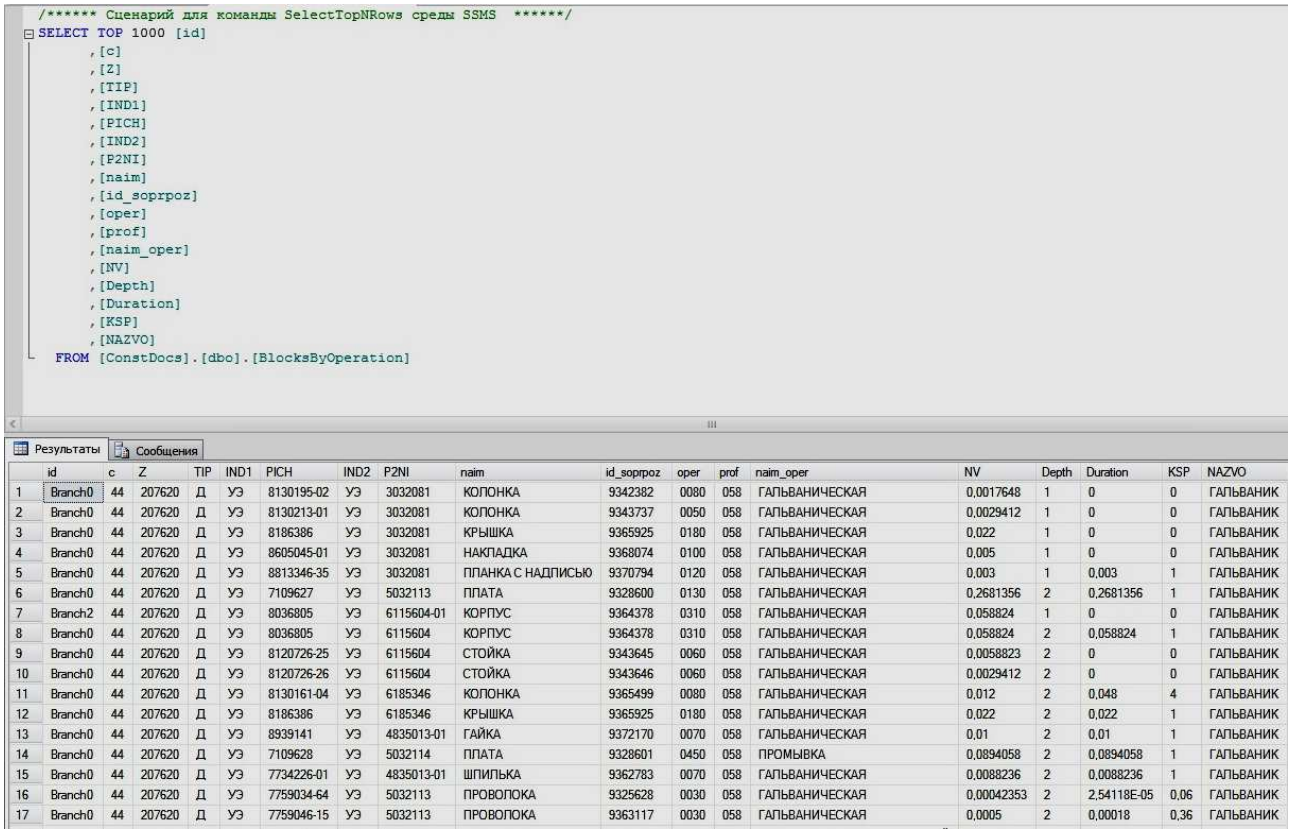


Рисунок 3.8 – Формирование запроса BlocksByDepartment



	id	c	Z	TIP	IND1	PICH	IND2	P2NI	naim	id_soprpoz	oper	prof	naim_oper	NV	Depth	Duration	KSP	NAZVO
1	Branch0	44	207620	Д	УЭ	8130195-02	УЭ	3032081	КОЛОНКА	9342382	0080	058	ГАЛЬВАНИЧЕСКАЯ	0,0017648	1	0	0	ГАЛЬВАНИК
2	Branch0	44	207620	Д	УЭ	8130213-01	УЭ	3032081	КОЛОНКА	9343737	0050	058	ГАЛЬВАНИЧЕСКАЯ	0,0029412	1	0	0	ГАЛЬВАНИК
3	Branch0	44	207620	Д	УЭ	8186386	УЭ	3032081	КРЫШКА	9365925	0180	058	ГАЛЬВАНИЧЕСКАЯ	0,022	1	0	0	ГАЛЬВАНИК
4	Branch0	44	207620	Д	УЭ	8605045-01	УЭ	3032081	НАКЛАДКА	9368074	0100	058	ГАЛЬВАНИЧЕСКАЯ	0,005	1	0	0	ГАЛЬВАНИК
5	Branch0	44	207620	Д	УЭ	8813346-35	УЭ	3032081	ПЛАНКА С НАДПИСЬЮ	9370794	0120	058	ГАЛЬВАНИЧЕСКАЯ	0,003	1	0,003	1	ГАЛЬВАНИК
6	Branch0	44	207620	Д	УЭ	7109627	УЭ	5032113	ПЛАТА	9328600	0130	058	ГАЛЬВАНИЧЕСКАЯ	0,2681356	2	0,2681356	1	ГАЛЬВАНИК
7	Branch2	44	207620	Д	УЭ	8036805	УЭ	6115604-01	КОРПУС	9364378	0310	058	ГАЛЬВАНИЧЕСКАЯ	0,058824	1	0	0	ГАЛЬВАНИК
8	Branch0	44	207620	Д	УЭ	8036805	УЭ	6115604	КОРПУС	9364378	0310	058	ГАЛЬВАНИЧЕСКАЯ	0,058824	2	0,058824	1	ГАЛЬВАНИК
9	Branch0	44	207620	Д	УЭ	8120726-25	УЭ	6115604	СТОЙКА	9343645	0060	058	ГАЛЬВАНИЧЕСКАЯ	0,0058823	2	0	0	ГАЛЬВАНИК
10	Branch0	44	207620	Д	УЭ	8120726-26	УЭ	6115604	СТОЙКА	9343646	0060	058	ГАЛЬВАНИЧЕСКАЯ	0,0029412	2	0	0	ГАЛЬВАНИК
11	Branch0	44	207620	Д	УЭ	8130161-04	УЭ	6185346	КОЛОНКА	9365499	0080	058	ГАЛЬВАНИЧЕСКАЯ	0,012	2	0,048	4	ГАЛЬВАНИК
12	Branch0	44	207620	Д	УЭ	8186386	УЭ	6185346	КРЫШКА	9365925	0180	058	ГАЛЬВАНИЧЕСКАЯ	0,022	2	0,022	1	ГАЛЬВАНИК
13	Branch0	44	207620	Д	УЭ	8939141	УЭ	4835013-01	ГАЙКА	9372170	0070	058	ГАЛЬВАНИЧЕСКАЯ	0,01	2	0,01	1	ГАЛЬВАНИК
14	Branch0	44	207620	Д	УЭ	7109628	УЭ	5032114	ПЛАТА	9328601	0450	058	ПРОМЫВКА	0,0894058	2	0,0894058	1	ГАЛЬВАНИК
15	Branch0	44	207620	Д	УЭ	7734226-01	УЭ	4835013-01	ШПИЛЬКА	9362783	0070	058	ГАЛЬВАНИЧЕСКАЯ	0,0088236	2	0,0088236	1	ГАЛЬВАНИК
16	Branch0	44	207620	Д	УЭ	7759034-64	УЭ	5032113	ПРОВОЛОКА	9325628	0030	058	ГАЛЬВАНИЧЕСКАЯ	0,00042353	2	2,54118E-05	0,06	ГАЛЬВАНИК
17	Branch0	44	207620	Д	УЭ	7759046-15	УЭ	5032113	ПРОВОЛОКА	9363117	0030	058	ГАЛЬВАНИЧЕСКАЯ	0,0005	2	0,00018	0,36	ГАЛЬВАНИК

Рисунок 3.9 – Выполнение запроса BlocksByOperation

Сформированные табличные представления являются массивом входных данных для разрабатываемого приложения.

	Z	D	PTP	T_KU	IND_KU	OBOZN_KU	T_CH	IND_CH	OBOZN_CH	KSP
1	203401	2013-04-04 00:00:00.000	00	Б	0000	00000000000	Б	УЭ	2097082	1
2	203420	2014-05-28 00:00:00.000	00	Б	0000	00000000000	Б	УЭ	2099055-01	1
3	203439	2013-04-04 00:00:00.000	00	Б	0000	00000000000	Б	УЭ	5032151	1
4	203466	2014-02-25 00:00:00.000	00	Б	0000	00000000000	Б	УЭ	2092264	1
5	203466	2014-02-25 00:00:00.000	02	Б	УЭ	2092264	Б	УЭ	2248212	0
6	203467	2014-04-04 00:00:00.000	00	Б	0000	00000000000	Б	УЭ	2390265	1
7	203467	2014-04-04 00:00:00.000	00	Б	0000	00000000000	Б	УЭ	4039072	1
8	205042	2015-01-27 00:00:00.000	00	Б	0000	00000000000	Б	УЭ	3620441	2
9	205045	2014-02-05 00:00:00.000	00	Б	0000	00000000000	Б	УЭ	2236184-01	2
10	205048	2013-11-29 00:00:00.000	00	Б	0000	00000000000	Б	УЭ	2248205-01	1
11	205050	2013-12-25 00:00:00.000	00	Б	0000	00000000000	Б	УЭ	6452640	1
12	205053	2015-02-16 00:00:00.000	00	Б	0000	00000000000	Б	УЭ	3032118	1
13	205058	2014-04-17 00:00:00.000	00	Б	0000	00000000000	Б	УЭ	2068375	1
14	205059	2014-04-21 00:00:00.000	00	Б	0000	00000000000	Б	УЭ	3035175	1
15	206501	2013-04-05 00:00:00.000	00	Б	0000	00000000000	Б	УЭ	2097075-02	3

Рисунок 3.10 – Таблица заказов

Таблица 3.6 – Описание таблицы БД Заказов

Название	Тип	Обозначение
Z	nvachar(8)	Номер заказа
D	datetime	Дата
PTP	nvachar(5)	
Название	Тип	Обозначение
T_KU	nvachar(4)	Тип
IND_KU	nvachar(8)	Индекс
OBOZN_KU	nvachar(14)	Обозначение головного блока
T_CH	nvachar(3)	Тип
IND_CH	nvachar(6)	Индекс
OBOZN_CH	nvachar(13)	Обозначение
KSP	float	Количество на блок

Кроме этого задаемся номером заказа для построения календарного графика.

3.3 Разузлование состава изделия

Для построения календарного графика и получения списка работ по цехам необходимо получить список блоков, используемых при сборке изделия. Для этого необходимо прибегнуть к рекурсивной функции разузлования, которая построит для заказа иерархическое дерево всех входящих позиций.

Разузлование – иерархическое разбиение изделия на входящие в него узлы, сборки, детали и покупные элементы (рисунок 3.11).

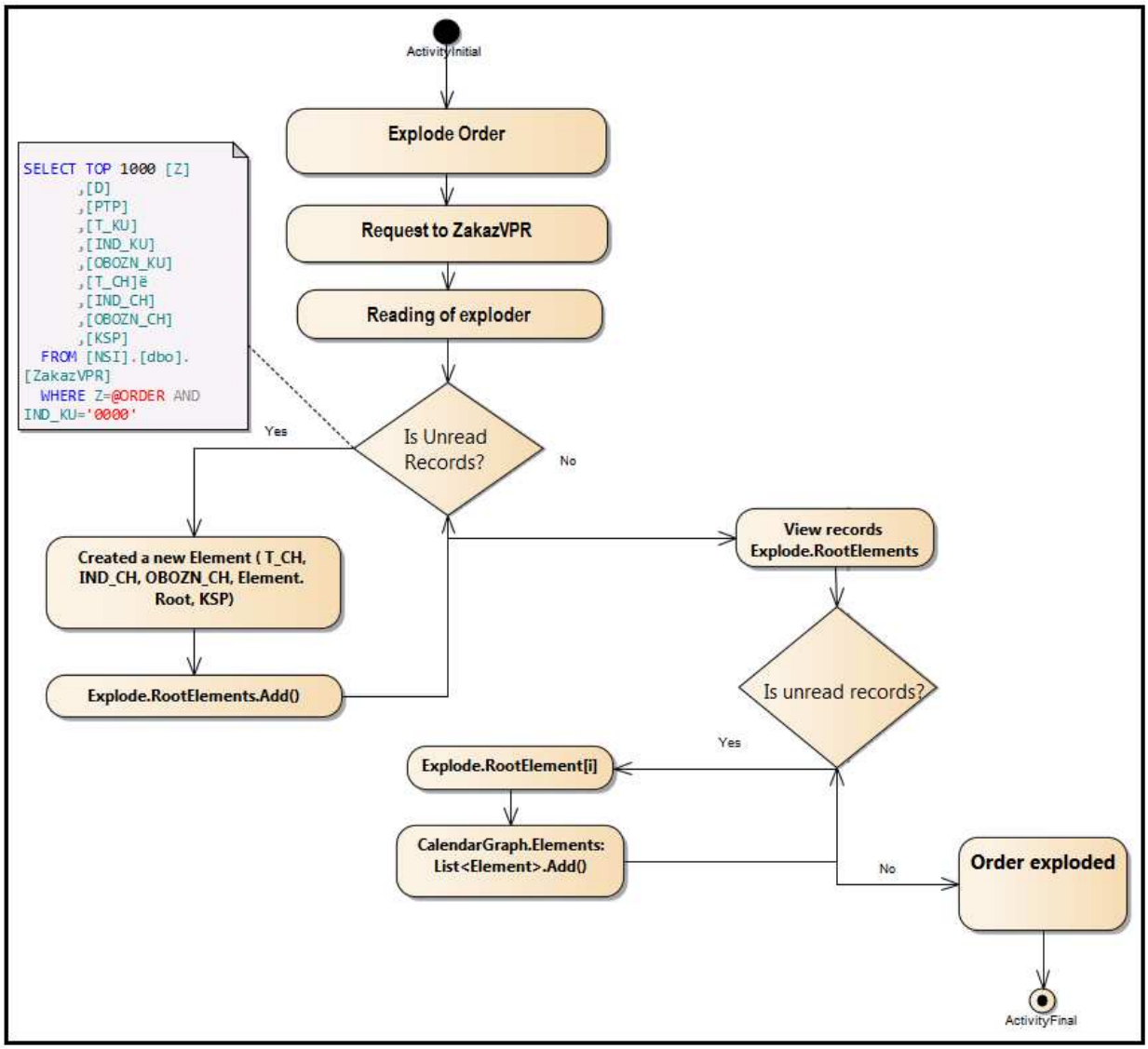


Рисунок 3.11 – Процедура разузлование (диаграмма деятельности)

Чтение разузлования заказа начинается с головных блоков и их заполнением и разузлованием головных блоков на позиции, которые также заполняются.

Результатом работы является полный список разузлования. На рисунке 3.11 изображена диаграмма последовательности, выполненная методом разузлования(OrderExplode).

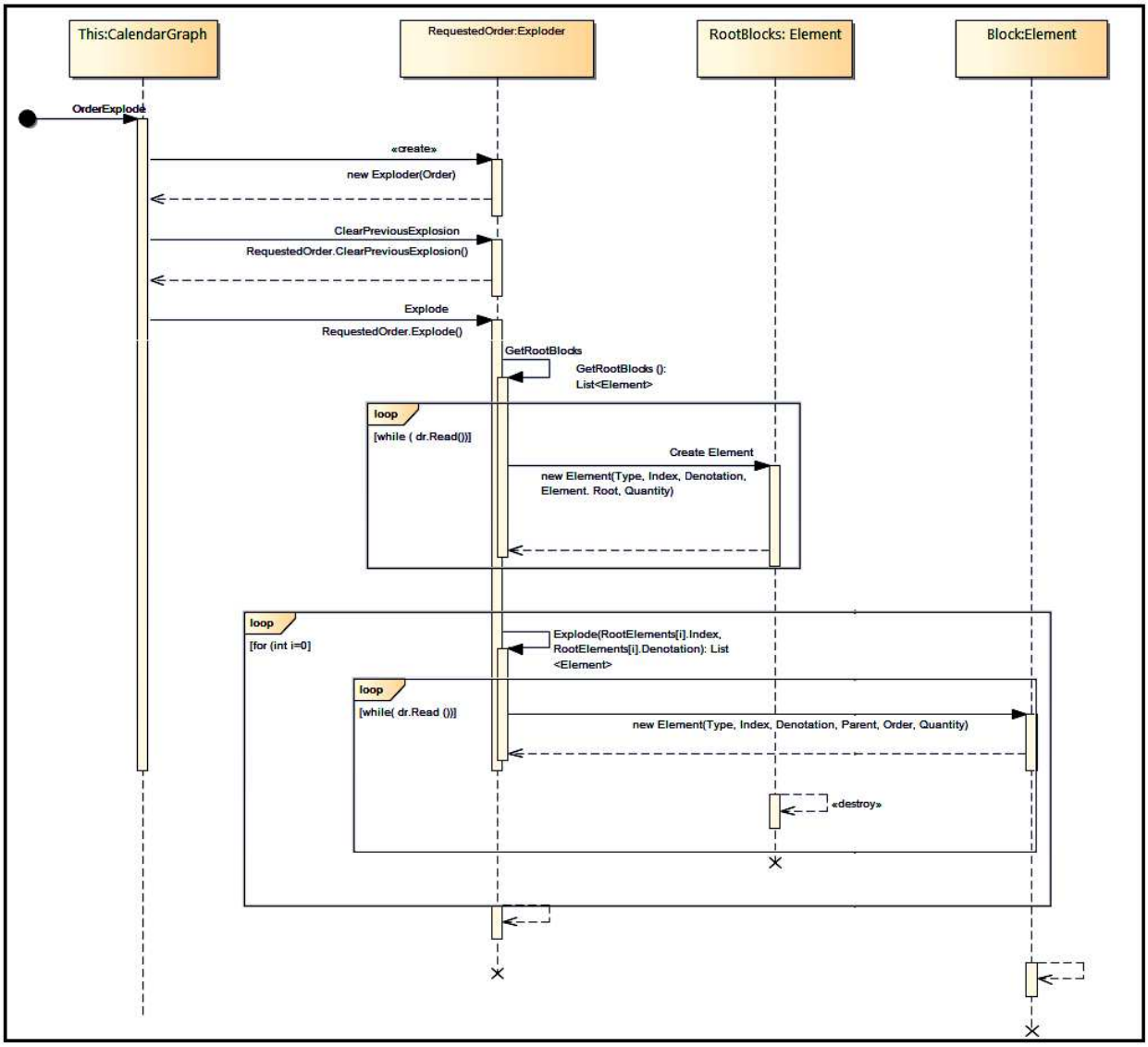


Рисунок 3.12 – Диаграмма последовательности, описывающая метод разузлования заказа

Алгоритм получения агрегированных работ приведен на рисунке 3.13.

Диаграмма на рисунке 3.12 показывает порядок действий, по которому получается список агрегированных работ по цехам [33]. Общий порядок говорит о том, что заготовительные работы начинаются после заключения контракта, торгов, первого транша оплаты и закупки материалов (на которые при расчетах выделяется фиксированное время), а работа цехов основного производства начинается после того, как завершена закупка ПКИ.

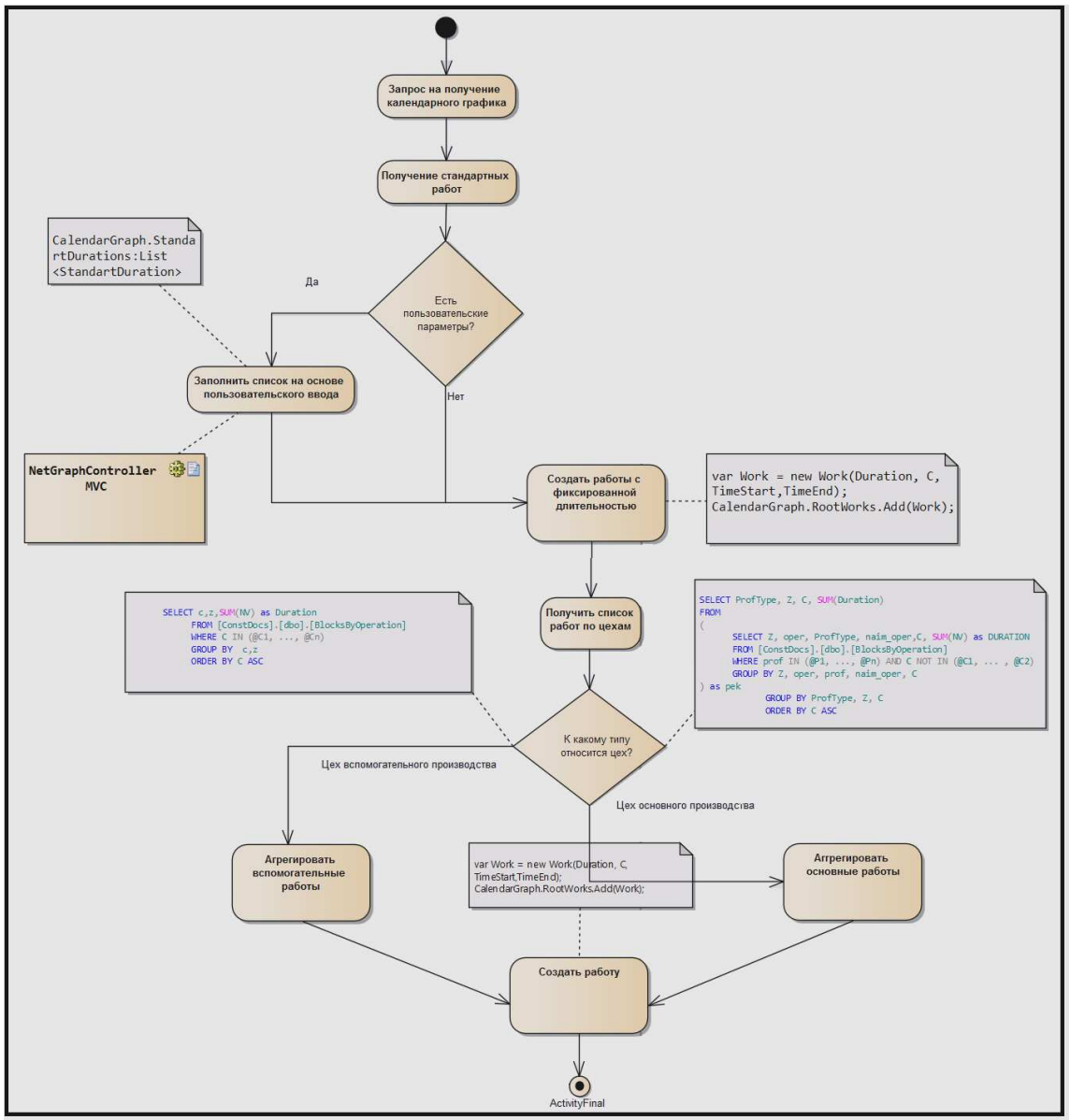


Рисунок 3.13 – Диаграмма деятельности, описывающая алгоритм получение агрегированных работ по цехам

3.4 Спецификация используемого API

Спецификация WEB API реализована следующими методами:

1) Метод получение состава изделия с длительностями работ имеет входные параметры, представленные в таблице 3.7 [34].

API доступен с модификатором доступа GET, по адресу URL: {IpAddress}/api/order/{value}/build

Таблица 3.7 – Общие параметры запроса построения диаграммы Ганта

Значение	Требуется	Тип	Описание
Value	Да	String	Номер заказа.

Ответ предоставляется в формате JSON. Пример результата ответа от сервера представлен в приложении А.

2) Метод получения состава изделия по уровням работ имеет входные параметры, представленные в таблице 3.8. API доступен с модификатором доступа GET, по адресу URL: {IpAddress}/api/order/{value}/Exploder

Таблица 3.8 – Общие параметры запроса получения номенклатуры изделия

Значение	Требуется	Тип	Описание
Value	Да	String	Обозначение детали или сборки.

Ответ предоставляется в формате JSON. Пример ответа представлен в приложении Б.

3) Метод получения списка технологического маршрута с наименованием выполняемых работ и длительностью работ имеет входные параметры, представленные в таблице 3.9. API доступен с модификатором доступа GET, по адресу URL: {IpAddress}/api/order/{value}/WorkDetail.

Ответ предоставляется в формате JSON. Пример ответа представлен в приложении В.

Таблица 3.9 – Общие параметры запроса получения технологии

Значение	Требуется	Тип	Описание
Value	Да	String	Обозначение детали или сборки.

3.5 Алгоритм работы ORM

Алгоритм работы ORM представлен на рисунке 3.14 в виде диаграммы последовательности.

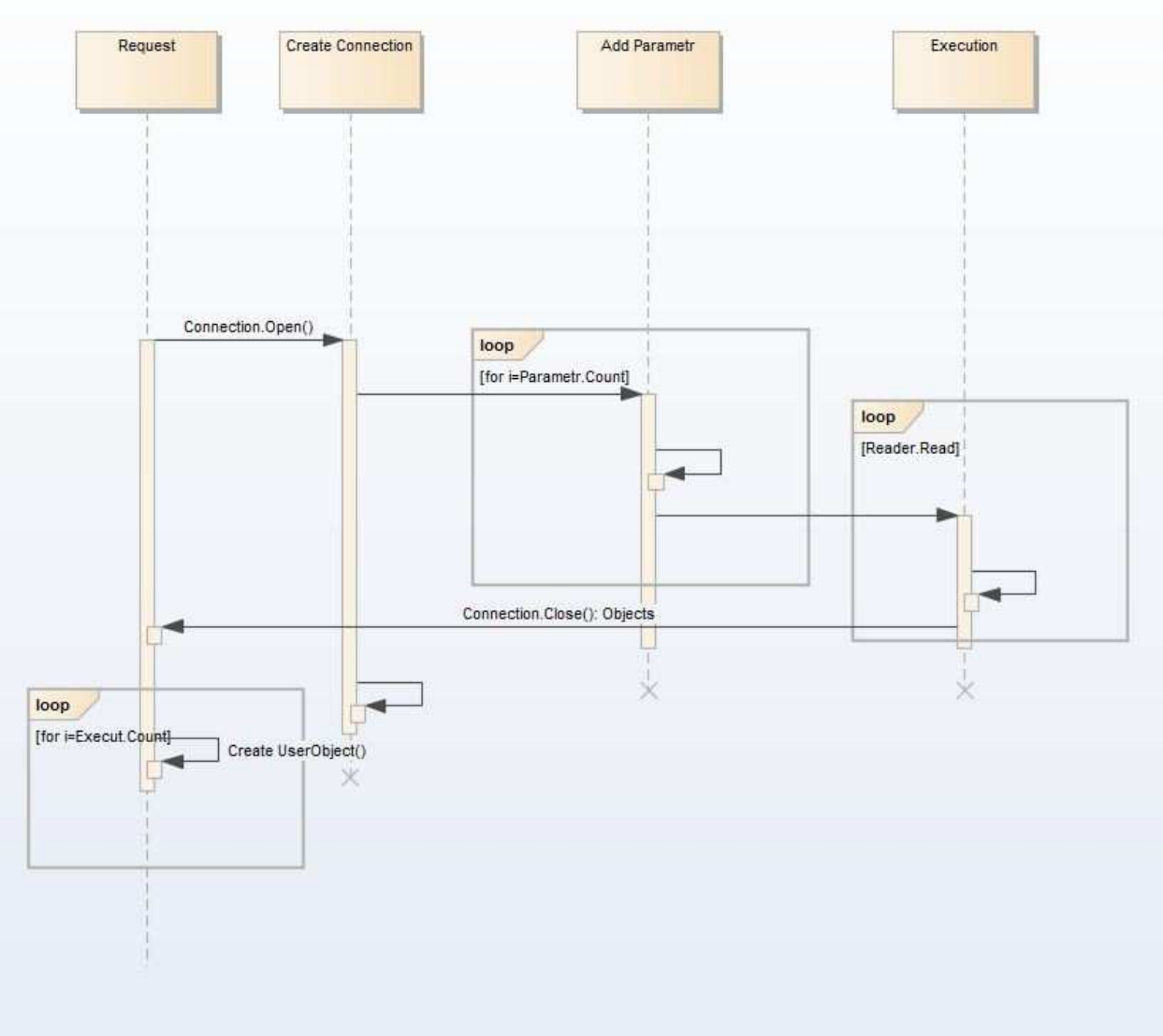


Рисунок 3.13 – Диаграмма последовательности работы ORM

Для отображения массива данных в виде коллекции объектной модели используется класс DatabaseManager.cs. Методы данного класса включают:

1. Метод инициализации подключения к базе данных:

```
private SqlConnection PrimaryConnection
{
	get
	{
		if(_primaryConnection == null)
		{
			_primaryConnection = new
SqlConnection(PrimaryConnectionString);
		}
		if(_primaryConnection.State != ConnectionState.Open)
		{
			_primaryConnection.Open();
			if(_primaryConnection.State != ConnectionState.Open)
			{
				Console.WriteLine(«Primary SQL Connection
established!»);
			}
		}
		return _primaryConnection;
	}
}
```

2. Метод добавления параметров:

```
public void SendCommand(String Command, Dictionary<String, Object>
Parameters)
{
	using(SqlCommand cmd = new SqlCommand(««,PrimaryConnection))
	{
```

```
                foreach(var Parameter in Parameters)
                {
                        cmd.Parameters.AddWithValue(Parameter.Key,
Parameter.Value);
                }
                cmd.CommandText = Command;
                cmd.ExecuteNonQuery();
                }
        }
}
```

3. Метод выполнения команды:

```
public IList<Row> SendRequest(String Command, Dictionary<String, Object>
Parameters = null)
{
                // TODO : Конкретизировать результат
                var Result = new List<Row>();
         if(Parameters == null)
         {
            Parameters = new Dictionary<String, Object>();
         }
using(SqlCommand cmd = new SqlCommand(Command,PrimaryConnection))
        {
            foreach(var Parameter in Parameters)
            {
               cmd.Parameters.AddWithValue(Parameter.Key, Parameter.Value);
            }
               using(SqlDataReader dr = cmd.ExecuteReader())
               {
                        var Request = cmd.CommandText;
                        var RequestParams = cmd.Parameters.ToString();
```

```
        Console.WriteLine(«Send request for database:» + Request);
        Console.WriteLine(«Parameters:» + RequestParams);
            var Dataset = new Dataset(dr);
            Result = Dataset.Rows;
            }
        }
        return Result;
}
```

Таким образом, разработана система ORM, реализован алгоритм разузлования состава изделия. Спроектирована архитектура «Facade» для ORM на основе диаграммы UML. Представлено описание методов WEB API.

ЗАКЛЮЧЕНИЕ

Проектирование системы оперативного управления производством – инструмент, помогающий в принятии управленческих решений. Задача этого процесса – обеспечить нововведения и изменения на предприятии, в достаточной степени минимизировать затраты на производство всех видов продукции.

В квалификационной работе проанализированы существующие принципы построения систем интеграции и методы формирования единого информационного пространства предприятия, их структура и основные функции, проанализированы методы реализации программных продуктов данного профиля. Сформированы основные функциональные требования к информационной системе, поставлена задача на ее разработку.

В ходе выполнения диссертационного исследования создан программный модуль, обеспечивающий автоматизированное построение календарного графика, ориентированный на использование в едином информационном пространстве АО «НПП «РАДИОСВЯЗЬ». Для эффективной интеграции разработанного прикладного решения предложены механизмы, представляющие собой реализацию интеграционной шины на основе паттерна «Фасад» и технологии ORM с реализацией поддержки реляционных баз данных различного типа.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Туровец, О.Г. Организация производства и управления предприятием: учебник / О. Г. Туровец. – Москва: ИНФРА-М, 2009. – 544с.

2. Новицкий, Н. И. Организация и планирование производства: Практикум / Н.И. Новицкий. – Минск: Новое знание, 2004. – 331 с.

3. Самочкин, В. Н. Гибкое развитие предприятия. Анализ и планирование. – 2-е изд. испр., доп. – М.: Дело, 2000. – 312 с.

4. Фатхутдинов, Р.А. Организация производства: Учебник. – М.: ИНФРА-М, 2002. – 672 с.

5. Мордвинцева, А.Д. Современные методы планирования и управления на предприятиях. Конспект лекций / А.Д. Мордвинцева. – Владивосток: Изд-во ВГУЭС, 2002.

6. Галеев Р. Г информационная поддержка организации производства изделий радиоэлектронной аппаратуры на предприятии ОАО «НПП «Радиосвязь» / Р. Г. Галеев, В. Г. Коннов, М. А. Казанцев, С. В. Ченцов // Журнал Сибирского Федерального Университета. Техника и технологии – 2014. – Т.7, №6. – С. 758–766

7. Капулин, Д. В. Автоматизация планирования мелкосерийного производства сетевыми методами / Д. В. Капулин, М. В. Винниченко, Д. И. Винниченко // Прикладная информатика: научно-практический журнал / Московский финансово–промышленный университет «Синергия» – 2016. – Т. 11, № 6 (66). – С. 6–18.

8. Октаров А.Н. Проектирование подсистем производства. – Москва: Эксмо, 2000г.– 128с.

9. Тютюкин, В.К. Математические методы календарного планирования / В.К.Тютюкин. – Ленинград: Изд-во Ленинград.ун-та, 1984. – 196с.

10. Новицкий, Н. И. Организация производства на предприятиях: Учеб.-метод. Пособие. М.: Финансы и статистика, 2002.– с.178-215.

11. Информационный портал [Электронный ресурс] https://habrahabr.ru/post/181988/ - разработка Web API

12. Дубровин, И. А. Организация и планирование производства на предприятиях / И.А. Дубровин. - М.: КолосС, 2008. - 359 с

13. Александров, В.В.; Вишняков, Ю.С.; Горская, Л.М. и др. Информационное обеспечение интегрированных производственных комплексов; Л.: Машиностроение, 2009**. -** 511 с.

14. Форум [Электронный ресурс] http://www.uppro.ru/library/ - information_systems/production/promyshennost-is.html - виды информационных систем в промышленности.

15. Словарь [Электронный ресурс] https://ru.wikipedia.org/wiki/ - Автоматизорованная_система_управления - автоматизированная система управления.

16. Агуров, П.В. Практика программирования интерфейсов баз данных. /П.В. Агуров. – Спб.: БХВ-Петербург., 2006. – 624с.

17. Информационный портал [Электронный ресурс] https://habrahabr.ru/post/278571/ - Подготовка ASP.NET 5 (Core) проекта и DNX окружения.

18. Информационный портал [Электронный ресурс] https://habrahabr.ru/post/164945/- RESTFul Api контроллеры в .NET MVC 4.

19. Словарь [Электронный ресурс] https://ru.wikipedia.org/wiki/- Диаграмма_Ганта - Диаграмма Ганта.

20. Дэвидсон, Луис проектирование баз данных на SQL Server 2000; Бином, 2009. **-** 631 с.

21. Кайт, Т., Кун, Д. Oracle для профессионалов: архитектура и методики программирования. 3-е изд. Apress, 2016. - 960 с.

22. Гольцман, В. - MySQL 5.0. Питер, 2013. – с.783.

23. Тепляков, С. Паттерны проектирования на платформе .NET. – СПб.: Питер, 2015. – 320 с.

24. Орлик С., Булуй Ю. Введение в программную инженерию и управление жизненным циклом ПО Программная инженерия. Программные требования. Copyright © Сергей Орлик, 2004– 2005. http://www.sorlik.ru/swebok/3– 1– software_engineering_requirements.pdf

25. Информационный порта [Электронный ресурс] https://habrahabr.ru/post/117468/ - интеграция информационных систем.

26. Форум [Электронный ресурс] http://corpsys.ru/Integration/Methods.aspx - системная интеграция

27. Форум [Электронный ресурс] http://citforum.ru/database/kbd96/41.shtml - Проектирование баз данных: новые требования, новые подходы.

28. http://www.tadviser.ru - обзор MES интеграторов Портал.

29. Прохоров, А. Ю. Определение оптимальной структуры базы данных //Informix magazine. Русское издание. – 1998. – Апрель.

30. Хаббард, Д. Автоматизированное проектирование баз данных. - М.: Мир, 1984. – 294 с.

31. Ладыженский, Г.М. Системы управления базами данных - коротко о главном //СУБД. - 1995. - №1,2,3,4.

32. ГОСТ Р ИСО/МЭК 12207/99. Государственный стандарт РФ. Информационная технология. Процессы жизненного цикла информационных систем. Издание официальное. – Москва, 1999.

33. Свид. 2016616446 Российская федерация. Государственная регистрационная программа для ЭВМ. Программный модуль разузлования материалокомплектов / Чорномаз Р. И., Капулин Д. В., Винниченко Д. И., Джиоева Н. Н., Котова М. В.; заявитель и правообладатель Федеральное государственное автономное образовательное учреждение высшего образования «Сибирский федеральный университет» (СФУ) (RU). – № 2016614024; заявл. 22.04.2016; опубл. 20.07.2016, Реестр программ для ЭВМ. – 1с.

34. Свид. 2017611504 Российская федерация. Государственная регистрационная программа для ЭВМ. Программный модуль построения объектов для диаграммы Ганта / Чорномаз Р. И., Капулин Д. В., Винниченко М. В., Винниченко Д. И.; заявитель и правообладатель Федеральное государственное автономное образовательное учреждение высшего образования «Сибирский федеральный университет» (СФУ) (RU). – № 2016663431; заявл. 08.12.2016; опубл. 06.02.2017, Реестр программ для ЭВМ. – 1с.

ПРИЛОЖЕНИЕ А

data: [
{ "id": 1, "text": "ФИГУРА М5", "start_date": "24.04.2013 0:00:00", "duration": 88, "progress": 0, "open": true, "parent": 0 },
{ "id": 3027627, "text": "(15)2097082", "start_date": "24.04.2013 0:00:00", "duration": 7.892999999999999, "progress": 0.5, "open": true, "parent": 0 },
{ "id": 3027631, "text": "(15)5097221", "start_date": "25.04.2013 0:00:00", "duration": 0.28200000000000003, "progress": 0.5, "open": true, "parent": 3027627 },
{ "id": 3027632, "text": "(15)5097222", "start_date": "26.04.2013 0:00:00", "duration": 0.28200000000000003, "progress": 0.5, "open": true, "parent": 3027627 },
{ "id": 3027635, "text": "(15)7760028", "start_date": "27.04.2013 0:00:00", "duration": 0.403, "progress": 0.5, "open": true, "parent": 3027627 },
{ "id": 3027636, "text": "(15)7760029", "start_date": "28.04.2013 0:00:00", "duration": 0.43800000000000006, "progress": 0.5, "open": true, "parent": 3027627 },
{ "id": 3027637, "text": "(15)7850145", "start_date": "29.04.2013 0:00:00", "duration": 0.1487, "progress": 0.5, "open": true, "parent": 3027627 },
{ "id": 3027634, "text": "(38)7746164", "start_date": "30.04.2013 0:00:00", "duration": 0.0025, "progress": 0.5, "open": true, "parent": 3027627 },
{ "id": 3027650, "text": "(38)8170135", "start_date": "01.05.2013 0:00:00", "duration": 0.022, "progress": 0.5, "open": true, "parent": 3027627 },
{ "id": 3027651, "text": "(38)8210633", "start_date": "02.05.2013 0:00:00", "duration": 0.008, "progress": 0.5, "open": true, "parent": 3027627 },
{ "id": 3027652, "text": "(38)8224652", "start_date": "03.05.2013 0:00:00", "duration": 0.0083, "progress": 0.5, "open": true, "parent": 3027627 },

{ "id": 3027653, "text": "(38)8224653", "start_date": "04.05.2013 0:00:00", "duration": 0.0083, "progress": 0.5, "open": true, "parent": 3027627 }, {"id":32,"text":"(45)7842102","start_date":"27.07.2010 0:00:00","duration":0.02,"progress":0.5,"open":false,"parent":3028171},{"id":33,"text":"(45)7842102-01","start_date":"27.07.2010 0:00:00","duration":0.02,"progress":0.5,"open":false,"parent":3028171},{"id":34,"text":"(45)7842102-02","start_date":"27.07.2010 0:00:00","duration":0.02,"progress":0.5,"open":false,"parent":3028171},{"id":35,"text":"(2)7842103","start_date":"27.07.2010 0:00:00","duration":0.0276,"progress":0.5,"open":false,"parent":3028171},{"id":36,"text":"(3)7842103","start_date":"26.07.2010 0:00:00","duration":8,"progress":0.5,"open":false,"parent":3028171},{"id":37,"text":"(44)7842103","start_date":"25.07.2010 0:00:00","duration":8,"progress":0.5,"open":false,"parent":3028171},{"id":38,"text":"(5)7852065","start_date":"03.08.2010 0:00:00","duration":0.0436,"progress":0.5,"open":false,"parent":3028174},{"id":39,"text":"(5)7852081","start_date":"03.08.2010 0:00:00","duration":0.1082,"progress":0.5,"open":false,"parent":3028174},{"id":40,"text":"(5)8030484","start_date":"27.07.2010 0:00:00","duration":0.35,"progress":0.5,"open":false,"parent":3028171},{"id":41,"text":"(44)8030484","start_date":"26.07.2010 0:00:00","duration":8,"progress":0.5,"open":false,"parent":3028171},{"id":42,"text":"(44)8080340","start_date":"27.07.2010 0:00:00","duration":0.021,"progress":0.5,"open":false,"parent":3028171},{"id":43,"text":"(5)8120970-01","start_date":"27.07.2010 0:00:00","duration":0.1562,"progress":0.5,"open":false,"parent":3028171},{"id":44,"text":"(38)8120970-01","start_date":"26.07.2010 0:00:00","duration":8,"progress":0.5,"open":false,"parent":3028171}

].

ПРИЛОЖЕНИЕ Б

data: [
{"id_record":"3027628","Type":"B","Ind":"","Denotation":"ВИНТ B2.M3-6GX16.36.016 ГОСТ 17475-80","Amount":"2","Depth":"1"},{"id_record":"3027629","Type":"B","Ind":"","Denotation":"ГАЙКА M3-6H.5.016 (S5,5) ГОСТ 5927-70","Amount":"2","Depth":"1"},{"id_record":"3027630","Type":"B","Ind":"","Denotation":"ШТИФТ 1X8.T ГОСТ 3128-70","Amount":"1","Depth":"1"},{"id_record":"3027631","Type":"K","Ind":"УК","Denotation":"5097221","Amount":"1","Depth":"1"},{"id_record":"3027632","Type":"K","Ind":"УК","Denotation":"5097222","Amount":"1","Depth":"1"},{"id_record":"3027633","Type":"C","Ind":"УК","Denotation":"7733205","Amount":"1","Depth":"1"},{"id_record":"3027634","Type":"C","Ind":"УК","Denotation":"7746164","Amount":"1","Depth":"1"},{"id_record":"3027635","Type":"C","Ind":"УК","Denotation":"7760028","Amount":"1","Depth":"1"},{"id_record":"3027636","Type":"C","Ind":"УК","Denotation":"7760029","Amount":"1","Depth":"1"},{"id_record":"3027637","Type":"C","Ind":"УК","Denotation":"7850145","Amount":"1","Depth":"1"},{"id_record":"3027638","Type":"C","Ind":"УК","Denotation":"7854338","Amount":"1","Depth":"1"},{"id_record":"3027639","Type":"C","Ind":"УК","Denotation":"7854338-01","Amount":"1","Depth":"1"},{"id_record":"3027641","Type":"C","Ind":"УК","Denotation":"7854338-03","Amount":"1","Depth":"1"},{"id_record":"3027642","Type":"C","Ind":"УК","Denotation":"7860887-01","Amount":"2","Depth":"1"},{"id_record":"3027643","Type":"C","Ind":"УК","Denotation":"7860896-05","Amount":"1","Depth":"1"},{"id_record":"3027644","Type":"C","Ind":"УК","Denotation":"7860896-

06","Amount":"1","Depth":"1"},{"id_record":"3027645","Type":"С","Ind":"УК","Denotation":"7860933","Amount":"1","Depth":"1"},{"id_record":"3027646","Type":"С","Ind":"УК","Denotation":"7860934","Amount":"1","Depth":"1"},{"id_record":"3027647","Type":"С","Ind":"УК","Denotation":"8170128","Amount":"1","Depth":"1"},{"id_record":"3027648","Type":"С","Ind":"УК","Denotation":"8170129","Amount":"1","Depth":"1"},
{"id_record":"3028208","Type":"П","Ind":"","Denotation":"ШАЙБА А 4.21 ГОСТ 10450-78","Amount":"4","Depth":"2"},{"id_record":"3028209","Type":"Н","Ind":"","Denotation":"ЮПИЯ.715331.002-07","Amount":"2","Depth":"2"},{"id_record":"3028211","Type":"Б","Ind":"УЭ","Denotation":"2029039-04","Amount":"1","Depth":"2"},{"id_record":"3028219","Type":"Б","Ind":"УЭ","Denotation":"4835013-01","Amount":"1","Depth":"2"},{"id_record":"3028221","Type":"Б","Ind":"УЭ","Denotation":"5103391","Amount":"1","Depth":"2"},{"id_record":"3028222","Type":"Б","Ind":"УЭ","Denotation":"5103402","Amount":"1","Depth":"2"},{"id_record":"3028223","Type":"Б","Ind":"УЭ","Denotation":"5165110","Amount":"1","Depth":"2"},{"id_record":"3028224","Type":"Б","Ind":"УЭ","Denotation":"6186028","Amount":"1","Depth":"2"},{"id_record":"3028226","Type":"Д","Ind":"УЭ","Denotation":"7750475","Amount":"2","Depth":"2"},{"id_record":"3028228","Type":"Д","Ind":"УЭ","Denotation":"7842102","Amount":"2","Depth":"2"},
{"id_record":"3028229","Type":"Д","Ind":"УЭ","Denotation":"7842102-01","Amount":"1","Depth":"2"},{"id_record":"3028230","Type":"Д","Ind":"УЭ","Denotation":"7842102-02","Amount":"1","Depth":"2"},{"id_record":"3028231","Type":"Д","Ind":"УЭ","Denotation":"7842103","Amount":"2","Depth":"2"},{"id_record":"3028234","Type":"Д","Ind":"УЭ","Denotation":"8030484","Amount":"1","Depth":"2"},{"id_record":"3028235","Type":"Д","Ind":"УЭ","Denotation":"8080340","Amount":"4","Depth":"2"}].

ПРИЛОЖЕНИЕ В

data:[

{"Ind":"УК","Denotation":"2097082","Chain":"100","Department":"15","Duration":"0,01","Operation":"ТК.МАМЕРН"},{"Ind":"УК","Denotation":"2097082","Chain":"160","Department":"15","Duration":"0,25","Operation":"ТК.РАДИОАП"},{"Ind":"УК","Denotation":"2097082","Chain":"80","Department":"15","Duration":"0,46","Operation":"ТК.СЛ.СБОР"},{"Ind":"УК","Denotation":"2097082","Chain":"130","Department":"15","Duration":"0,01","Operation":"ТК.СЛ.СБОР"},{"Ind":"УК","Denotation":"2097082","Chain":"190","Department":"15","Duration":"0,1","Operation":"ТК.СЛ.СБОР"},{"Ind":"УК","Denotation":"2097082","Chain":"210","Department":"15","Duration":"0,222","Operation":"ТК.СЛ.СБОР"},{"Ind":"УК","Denotation":"2097082","Chain":"230","Department":"15","Duration":"0,02","Operation":"ТК.СЛ.СБОР"},{"Ind":"УК","Denotation":"2097082","Chain":"90","Department":"15","Duration":"0,167","Operation":"МАРКИРОВ."},{"Ind":"УК","Denotation":"2097082","Chain":"20","Department":"15","Duration":"0,67","Operation":"ПРИГОТ.РАС"},{"Ind":"УК","Denotation":"2097082","Chain":"140","Department":"15","Duration":"0,075","Operation":"РЕГУЛ.Р/АП"},{"Ind":"УК","Denotation":"2097082","Chain":"150","Department":"15","Duration":"0,25","Operation":"РЕГУЛ.Р/АП"},{"Ind":"УК","Denotation":"2097082","Chain":"70","Department":"15","Duration":"0,133","Operation":"СЛ-СБОРЩ."},
{"Ind":"УЭ","Denotation":"2517010-01","Chain":"230","Department":"45","Duration":"0,156","Operation":"РЕГУЛ.Р/АП"},{"Ind":"УЭ","Denotation":"2517010-01","Chain":"240","Department":"45","Duration":"0,563","Operation":"РЕГУЛ.Р/АП"},{"Ind":"УЭ","Denotation":"2517010-01","Chain":"10","Department":"45","Duration":"0,5","Operation":"СЛ-СБОРЩ."}

].

Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий
Системы автоматики, автоматизированное управление и проектирование

УТВЕРЖДАЮ
Заведующий кафедрой
_____ С. В. Ченцов
«22» 06 2017г.

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

РАЗРАБОТКА ИНТЕГРИРОВАННОГО РЕШЕНИЯ ИНФОРМАЦИОННОЙ ПОДДЕРЖКИ ПРОЦЕССОВ ПРОИЗВОДСТВЕННОГО ПЛАНИРОВАНИЯ И УПРАВЛЕНИЯ

Направление 27.04.04 Управление в технических системах
Магистерская программа 27.04.04.01Интегрированные системы управления производством

Научный руководитель	_____	22.06. .2017 г.	доцент, канд. техн. наук Д. В. Капулин
Выпускник	_____	22.06 .2017 г.	Р. И. Чорномаз
Рецензент	_____	22.06 .2017 г.	зав. каф., канд. техн. наук А. С. Кузнецов
Нормоконтролер	_____	22.06 .2017 г.	Т. А. Грудинова

Красноярск 2017