

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана»
(национальный исследовательский университет)
Московский техникум космического приборостроения

УТВЕРЖДАЮ
Заместитель директора по УР

Н.Н. Ковзель
(подпись, дата)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
РАЗРАБОТКА СИСТЕМЫ ДЛЯ АВТОМАТИЗАЦИИ
ПОДГОТОВКИ ЗАГРУЗКИ ПЕРВИЧНЫХ ДАННЫХ В
ХРАНИЛИЩЕ

Пояснительная записка

Группа ТИП-81

Председатель предметной
(цикловой) комиссии

(подпись, дата)

Е.А. Митрошенкова

(ФИО)

Руководитель разработки
от техникума

(подпись, дата)

Л.Б. Петренко

(ФИО)

Рецензент

(подпись, дата)

Т.Н. Коваленко

(ФИО)

Руководитель разработки
от предприятия

(подпись, дата)

А.В. Вилков

(ФИО)

Консультант по
экономической части

(подпись, дата)

С.А. Ковалев

(ФИО)

Разработчик

(подпись, дата)

М. А. Никифоров

(ФИО)

Москва, 2022

СОДЕРЖАНИЕ

Введение	5
1 Теоретическое обоснование разрабатываемого программного продукта	8
1.1 Описание предметной области	8
1.2 Сравнительный анализ программ-аналогов	10
1.3 Моделирование проектируемой системы	11
1.4 Разработка функциональных требований к программной системе	13
1.5 Обоснование выбора средств реализации программной системы	15
1.6 Вывод по разделу	17
2 Разработка программного продукта	18
2.1 Разработка архитектуры программной системы.....	18
2.2 Разработка структуры данных	20
2.3 Конструирование пользовательского интерфейса.....	23
2.4 Схемы алгоритма программы и подпрограмм	27
2.4.1 Схемы алгоритма класса MainWindow	27
2.4.1.1 Схема алгоритма конструктора класса MainWindow	27
2.4.1.2 Схема алгоритма метода startPreviewBtn_Click	28
2.4.1.3 Схема алгоритма метода Wizard_Next	29
2.4.1.4 Схема алгоритма метода InitTract	30
2.4.1.5 Схема алгоритма метода Wizard_Previous	32
2.4.1.6 Схема алгоритма метода IntroPage_Enter.....	32
2.4.1.7 Схема алгоритма метода startPointCB_SelectChang	33
2.4.1.8 Схема алгоритма метода anyChangeOnPage	35
2.4.1.9 Схема алгоритма метода drawPage	36
2.4.1.10 Схема алгоритма метода Dg_CellEditEnding	36
2.4.1.11Схема алгоритма метода Row_DoubleClick	37
2.4.1.12 Схема алгоритма метода generateTrcatTable	38
2.4.1.13 Схема алгоритма метода OdsStgPage_Enter.....	39
2.4.1.14 Схема алгоритма метода OdsRawPage_Enter.....	40
2.4.1.15 Схема алгоритма метода OdsExPage_Enter.....	41

2.4.1.16	Схема алгоритма метода StagefdPage_Enter	42
2.4.1.17	Схема алгоритма метода CalcfdPage_Enter	43
2.4.1.18	Схема алгоритма метода MainWizard_Help	44
2.4.1.19	Схема алгоритма метода IntroCommentLB_AddingNewItem	44
2.4.1.20	Схема алгоритма метода RenameExtSysName	45
2.4.1.21	Схема алгоритма метода MainWizard_Finish	45
2.4.1.22	Схема алгоритма метода SaveScripts	46
2.4.2	Схемы алгоритма класса ArtifactWindow	47
2.4.2.1	Схема алгоритма конструктора класса ArtifactWindow ..	47
2.4.2.2	Схема алгоритма метода SaveBtn_Click	48
2.5	Отладка и тестирование программы	49
2.6	Руководство пользователя	54
2.7	Вывод по разделу	61
3	Охрана труда	62
3.1	Техника безопасности при работе на компьютере	62
3.2	Требования к помещению (машинного зала, ВЦ)	65
3.3	Мероприятия по противопожарной технике	66
4	Технико-экономическая часть	68
4.1	Технико-экономические расчеты экономической целесообразности разработки программы	68
4.2	Расчетная часть	71
4.2.1	Расчет трудоемкости разработанной программы	71
4.2.2	Расчет себестоимости разработанной программы	75
4.2.3	Анализ возможных путей снижения себестоимости	81
4.3	Графическая часть	82
	Заключение	84
	Список использованных источников	85
	Приложение А Листинг программы	87
	A1 Листинг класса MainWindow	87
	A2 Листинг класса ArtifactWindow	120

A3 Листинг класса TractData	122
A4 Листинг класса TractLayerArtifact.....	122
A5 Листинг класса TractTableColumn	123
A6 Листинг класса TractLayerExtProp	124
A7 Листинг класса TractSystem.....	125
A8 Листинг класса TractTable	127
A9 Листинг класса Tools.....	130
A10 Листинг хранимой процедуры dbo.create_layer_mdata	132
Приложение Б Результаты выполнения программы.....	181

ВВЕДЕНИЕ

На современном этапе развития бизнеса средние и крупные банки вышли на новый уровень, при котором одним из важнейших конкурентных преимуществ является скорость и качество обработки накопленных данных, а также возможность их разностороннего анализа. С одной стороны, финансовой организации необходимо соответствовать требованиям национального законодательства по обязательной отчётности, с другой стороны, международный характер современного банковского бизнеса все более подвигает российские банки к международным стандартам работы и технологического развития. Вместе с тем, для руководства финансовой организации важна возможность в любой момент времени составить единую информационную картину работы всех подразделений банка, которая позволит корректно отслеживать показатели доходности в различных разрезах: клиент, продукт, канал продаж, центр финансовой ответственности и т.д.

Тенденция решать все эти задачи точно, на уровне отдельных департаментов и подразделений, при помощи различных систем осталась в прошлом: сегодня банки понимают необходимость и ценность консолидированного решения для сбора, обработки и анализа данных на основе корпоративного информационного хранилища (далее КИХ).

Промсвязьбанк (ПСБ) — универсальный банк, основанный в 1995 году. Входит в десятку крупнейших банков России и в список системно значимых кредитных организаций, утвержденный Центробанком. В конце декабря 2019 года ПСБ был законодательно присвоен статус опорного банка для оборонно-промышленного комплекса (ОПК). Услугами банка пользуются 2,5 миллиона физических и свыше 200 тысяч юридических лиц, в том числе более 10 тысяч крупных корпоративных клиентов. Сеть банка насчитывает около 300 точек продаж в России. ПСБ так же, как и большинство банков мира использует КИХ.

Корпоративное хранилище данных – это специальным образом организованный массив данных предприятия (организации), обрабатываемый и хранящийся в едином аппаратно-программном комплексе, который обеспечивает

быстрый доступ к оперативной и исторической информации, многомерный анализ данных, получение прогнозов и статистики в разрезах согласованной нормативно-справочной информации.

Источниками данных для информационного хранилища служат в первую очередь данные из разрозненных транзакционных и учетных информационных систем, основанных на различных реляционных СУБД, которые обслуживают повседневную бизнес-деятельность.

Не стоит считать КХД просто большой базой данных с множеством взаимосвязанных таблиц. В отличие от традиционной SQL-СУБД, КХД имеет сложную многоуровневую (слоеную) архитектуру, которая называется LSA – Layered Scalable Architecture. По сути, LSA реализует логическое деление структур с данными на несколько функциональных уровней. Данные копируются с уровня на уровень и трансформируются при этом, чтобы в итоге предстать в виде согласованной информации, пригодной для анализа.

Для импорта данных с уровня на уровень, для каждого из них, создается определенный набор объектов, таких как таблицы, триггеры, хранимые процедуры, индексы и другие.

Использование программного продукта (далее ПП) созданного в рамках дипломного проекта позволит сократить время, затрачиваемое на создание таких объектов, а так же упростит процесс подготовки загрузки первичных данных в хранилище.

В связи с этим, целью дипломного проекта является осуществления проектирования и разработки программного продукта для упрощения загрузки первичных данных в хранилище, путем автоматического формирования скриптов создания объектов используемых при импорте данных. Для достижения поставленной цели необходимо решить следующие задачи:

- 1) изучить и описать предметную область процесса загрузки первичных данных в информационное хранилище;
- 2) определить возможности разрабатываемого ПП;
- 3) разработать интерфейс ПП;
- 4) разработать алгоритм формирования sql скриптов;

- 5) разработать и описать руководства пользователя по эксплуатации ПП;
- 6) произвести технико-экономическое обоснование проекта.

Объектом изучения является процесс загрузки первичных данных в информационное хранилище.

Предметом изучения является процесс формирования скриптов.

1 Теоретическое обоснование разрабатываемого программного продукта

1.1 Описание предметной области

KINFD – корпоративное информационное хранилище данных Промсвязьбанка. В этом хранилище собирается информация из различных первичных и вторичных систем. Оно используется для подготовки управленческой и регулярной отчетности, а также для анализа.

Использование ПП существенно сократит время, затрачиваемое на создание объектов базы данных (БД) на слое хранилища необходимых для загрузки данных.

Основные базы данных хранилища:

1) ODS – БД для интеграции данных из детального уровня из различных источников. В данную базу загружаются данные из систем-источников. Загрузка в данную БД происходит без трансформации.

2) STAGEFD – база первичных данных. В БД загружаются первичные данные из ODS. Загрузка в данную БД происходит без трансформации.

3) CALCFD – база детальных данных, расчетная база. В данной базе происходит обогащение данных и расчет витрин.

4) KINFD – БД витрин.

Порядок следования данных:

[первичная система] → [ODS] → [STAGEFD] → [CALCFD] → [KINFD].

Загрузка первичных данных возможна только в ODS. В другие слои хранилища загрузка первичных данных запрещена. Поток данных однонаправленный. Если нужно загрузить данные в базу данных CALCFD, то процедура загрузки должна находиться в БД CALCFD. Предыдущая система (из схемы выше) не может брать данные из последующей или пропускать этап, например в KINFD использовать данные STAGEFD нельзя.

Входными данными являются данные первичных систем (систем источников).

В свою очередь ODS делится на следующие слои:

1) stage – служит для временного хранения данных, полученных с источника, и не предназначен для постоянного хранения.

2) raw data – служит для хранения данных. В слое сырых данных хранятся данные максимально полные по объему и близкие по структуре к первичным системам.

3) export data – в слое создаются прикладные витрины или представления.

Разделение слоев ODS осуществляется с помощью имен объектов.

Список объектов создаваемых на слоях хранилища определяется согласно особенностям работы.

Должны соблюдаться правила наименования объектов на слое при их создании.

1.2 Сравнительный анализ программ-аналогов

На сегодняшний день существует немного программ-аналогов. Рассмотрим несколько программных продуктов наиболее близких по функциональности.

SQL Server Management Studio (SSMS) – это бесплатная графическая среда, включающая набор инструментов для разработки сценариев на T-SQL и управления инфраструктурой Microsoft SQL Server. Среда SQL Server Management Studio – это основной, стандартный и полнофункциональный инструмент для работы с Microsoft SQL Server, разработанный компанией Microsoft, который предназначен как для разработчиков, так и для администраторов SQL Server.

Данная среда предоставляет встроенный механизм формирования различных шаблонов скриптов и генерации скриптов из уже существующих объектов. Недостатком использования данного программного продукта заключается в том, что для написания необходимых sql скриптов понадобится заметно больше времени на их редактирование, так как изначально будут сформированы только общие шаблоны.

MySQL Workbench - это унифицированный визуальный инструмент для архитекторов баз данных, разработчиков и администраторов баз данных. MySQL Workbench предоставляет моделирование данных, разработку SQL и комплексные инструменты администрирования для настройки сервера, администрирования пользователей, резервного копирования и многого другого.

MySQL Workbench предоставляет визуальные инструменты для создания SQL-запросов. Использование данного программного продукта так же не подходит для решения задачи формирования необходимых скриптов, так как скрипты будут создаваться на языке MySQL, а хранилище данных использует Transact-SQL.

В программном продукте, разрабатываемом в рамках дипломного проекта, будет сведена к минимуму необходимость редактирования скриптов. Скрипты будут формироваться на языке Transact-SQL, а также будет реализована функция сохранения скриптов в удобном формате и удобный интерфейс для работы пользователя.

1.3 Моделирование проектируемой системы

Для моделирования системы в рамках дипломного проекта была выбрана методология SADT. Это методология, разработанная специально для того, чтобы облегчить описание и понимание искусственных систем, попадающих в разряд средней сложности, разработанная в 1973 г. Дугласом Россом.

На рисунке 1.1 представлен функциональный блок IDEF0 процесса формирования скриптов объектов БД, необходимых при загрузке первичных данных в хранилище.

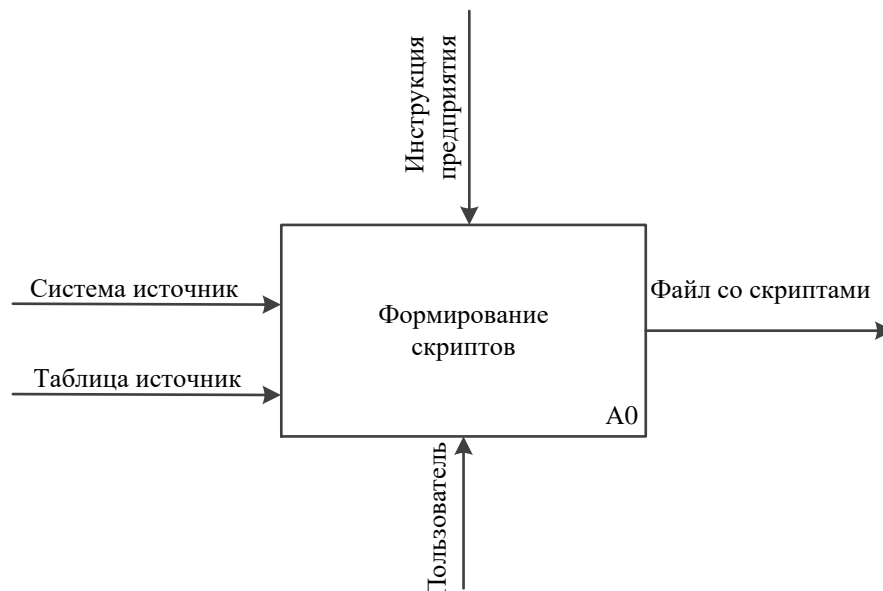


Рисунок 1.1 – Функциональный блок IDEF0

Входящие стрелки – «Система источник» и «Таблица источник». Это те вводные, которые необходимы для начала работы. Управляющая для формирования скриптов – это «Инструкция предприятия». В роли механизма выступает «Пользователь». Результатом работы системы должен быть «Файл со скриптами».

Но это – только основные рамки процесса. Так как описана общая схема процесса, его необходимо детализировать. Для этого была построена диаграмма модели IDEF0 (рисунок 1.2).

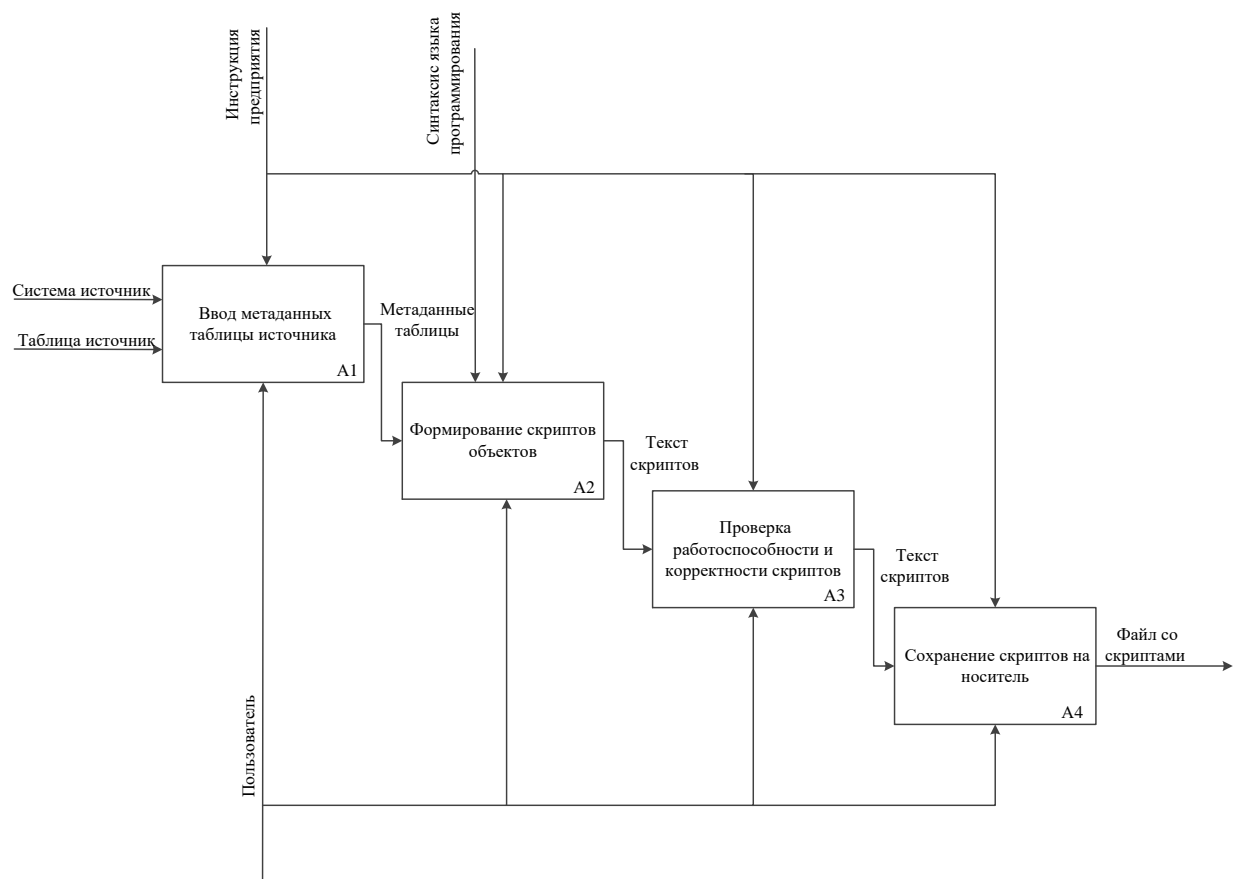


Рисунок 1.2 – Диаграмма модели IDEF0

Процесс «Формирование скриптов» делится на 4 этапа:

- 1) «ввод метаданных таблицы источника»;
- 2) «формирование скриптов объекта»;
- 3) «проверка работоспособности и корректности скриптов»;
- 4) «сохранение скриптов на носитель».

На диаграмме видно, на каком этапе какие управляющие элементы и какие механизмы задействованы.

Построенные диаграммы должны обеспечить правильность выполнения логики данного дипломного проекта и упростить создание фундамента программы.

1.4 Разработка функциональных требований к программной системе

Разрабатываемая система предназначена для автоматизации подготовки загрузки первичных данных в хранилище. Использование разрабатываемой системы позволит сократить время на написание скриптов для создания объектов на слоях хранилища, необходимых при загрузке данных. Система будет эксплуатироваться в отделе банка.

Система должна обеспечивать возможность выполнения следующих функций:

- 1) формирование скриптов объектов необходимых при загрузке данных;
- 2) при необходимости давать возможность редактирования метаданных таблиц используемых во время формирования скриптов;
- 3) редактирование сформированных скриптов;
- 4) сохранение скриптов в выбранное пользователем место.

Надежность технических средств обеспечивается использованием сертифицированных средств вычислительной техники и их комплектующих.

Надежность ПО обеспечивается использованием сертифицированных операционных систем, программных средств, используемых при разработке программного продукта.

Надежность прикладного ПО обеспечивается комплексом мероприятий, осуществляющих управление качеством создания ПО на всех этапах жизненного цикла.

Прикладные программы должны иметь защиту от некорректных действий пользователей и ошибочных исходных данных.

Состав и параметры технических средств должны отвечать следующим минимальным системным требованиям:

- процессор с тактовой частотой 1,8 ГГц и выше;
- объем оперативного запоминающего устройства – 2 Гб и более;
- объем постоянного запоминающего устройства – 64 Гб и более;
- устройство вывода (монитор);

– устройства ввода (клавиатура, компьютерная мышь).

Программа должна быть интуитивно понятной и расширяемой при необходимости. Так же она должны быть самодокументированы, т.е. листинг программы должен содержать все необходимые комментарии.

В состав сопровождающей документации должна входить пояснительная записка, содержащая описание разработки.

1.5 Обоснование выбора средств реализации программной системы

Для разработки данной системы была выбрана среда разработки Microsoft Visual Studio 2019 - продукт компании Microsoft, включающий интегрированную среду разработки программного обеспечения и ряд других инструментальных средств. Эта среда разработки обеспечивает высокое качество кода на протяжении всего цикла жизни программного обеспечения, от проектирования до внедрения, а так же позволяет быстро создавать подключаемые к базам данных приложения, способные обеспечить широчайшие возможности для работы пользователей.

Для разработки системы был выбран язык программирования C# – компилируемый статически типизированный язык программирования общего назначения.

C# поддерживает такие парадигмы программирования как процедурное программирование, объектно-ориентированное программирование, обобщённое программирование. А также обеспечивает модульность, отдельную компиляцию, обработку исключений, абстракцию данных, объявление типов (классов) объектов, виртуальные функции.

Одно из преимуществ языка C# является наличие конструированных форм, и задача событий для элементов форм, что упрощает процесс программирования и делает его интуитивно понятным даже для начинающих. Так же он объединяет достоинства таких языков как Java и C++.

Для реализации была выбрана технология Windows Presentation Foundation (WPF). WPF – представляет собой обширный API-интерфейс для создания настольных графических программ имеющих насыщенный дизайн и интерактивность. В отличие от устаревшей технологии Windows Forms, WPF включает новую модель построения пользовательских приложений (в основе WPF лежит мощная инфраструктура, основанная на DirectX). Это означает возможность применения развитых графических эффектов, не платя за это производительностью, как это было в Windows Forms. Фактически даже становятся доступными такие расширенные средства, как поддержка видеофайлов и

трехмерное содержимое. Используя эти средства (при наличии хорошего инструмента графического дизайна), можно создавать бросающиеся в глаза пользовательские интерфейсы и визуальные эффекты, которые были просто невозможны в Windows Forms.

В качестве инструмента для работы с СУБД был выбран SQL Server Management Studio (SSMS). SQL Server Management Studio – это бесплатная графическая среда для управления инфраструктурой SQL Server, разработанная компанией Microsoft. С помощью SSMS можно разрабатывать базы данных, выполнять инструкции T-SQL, а также администрировать Microsoft SQL Server.

Среда SQL Server Management Studio является полнофункциональным инструментом для работы с Microsoft SQL Server, который предназначен как для разработчиков, так и для администраторов SQL Server.

1.6 Вывод по разделу

В данном разделе был проведен анализ предметной области дипломного проекта и ее описание. Так же было проведено функциональное моделирование создаваемой системы. После чего были сформированы функциональные требования к разрабатываемой системе и описаны средства, которые будут использоваться для дальнейшей разработки.

На основе собранной информации можно начать проектирование и разработку программного продукта.

2 Разработка программного продукта

2.1 Разработка архитектуры программной системы

Концепция слоев является одной из наиболее распространенных моделей, используемых разработчиками программных систем для разбиения сложного приложения на составные части.

Основной ее идеей является выделение частей системы, выполняющих логически взаимосвязанные функции, в отдельные слои.

Слой более высокого уровня использует службы, которые ему предоставляет слой более низкого уровня.

Преимущества использования концепции слоев:

1) каждый слой является единым самодостаточным целым, которое можно отдельно разрабатывать и модифицировать, не беспокоясь о слоях, которые его окружают;

2) зависимости между слоями могут быть сведены к минимуму. Вся зависимость определяется только интерфейсом, который предоставляет слой;

3) очень много слоев (особенно низких уровней) можно использовать повторно, т.к. слой является обособленной единицей.

Несмотря на множество преимуществ, данная концепция имеет ряд недостатков:

1) сохраняется опасность каскадных модификаций;

2) наличие избыточных слоев снижает производительность системы.

В ходе проектирования система функционально разделилась на 2 слоя – клиент и сервер.

Клиент содержит слои презентации, бизнес-логики и передачи данных. Сервер включает хранилища и базы данных.

Выделив 2 наиболее значимых слоя, была выбрана классическая архитектура «клиент – сервер». Под клиент-серверным приложением в этом случае понимается система, основанная на использовании серверов баз данных.

Клиент-сервер (Client-server) – вычислительная или сетевая архитектура, в которой задания или сетевая нагрузка распределены между поставщиками услуг (сервисов), называемых серверами, и заказчиками услуг, называемых клиентами. Нередко клиенты и серверы взаимодействуют через компьютерную сеть и могут быть как различными физическими устройствами, так и программным обеспечением. На рисунке 2.1 предоставленная архитектура «клиент- сервер».



Рисунок 2.1 – Классическое представление архитектуры «клиент–сервер»

На стороне клиента выполняется код приложения, в который обязательно входят компоненты, поддерживающие интерфейс с конечным пользователем, выполняющие специфичные для приложения функции.

Клиентская часть приложения взаимодействует с клиентской частью программного обеспечения управления базами данных, которая, фактически, является индивидуальным представителем СУБД для приложения.

Заметим, что интерфейс между клиентской частью приложения и клиентской частью сервера баз данных, как правило, основан на использовании языка SQL. Поэтому такие функции, как, например, предварительная обработка форм, предназначенных для запросов к базе данных, или формирование результирующих отчетов выполняются в коде приложения.

Наконец, клиентская часть сервера баз данных, используя средства сетевого доступа, обращается к серверу баз данных, передавая ему текст оператора языка SQL.

2.2 Разработка структуры данных

В ходе проектирования системы возникла необходимость создания базы данных (БД) хранящей некоторую информацию, схема данных которой представлена на рисунке 2.2.

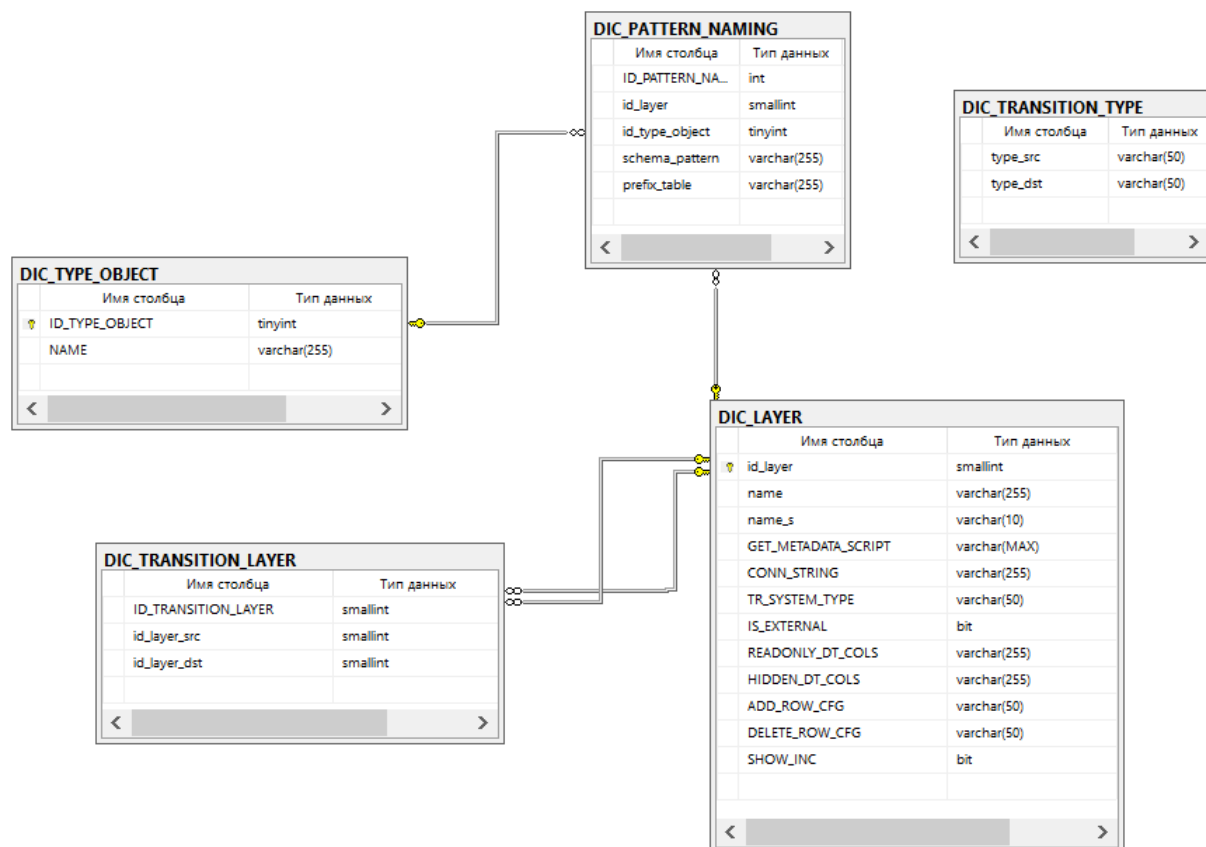


Рисунок 2.2 – Схема данных БД

Структура таблиц БД представлена в таблицах 2.1 – 2.5.

Таблица 2.1 – Таблица «DIC_TRANSACTION_TYPE»

Идентификатор	Описание	Тип данного	Размер
type_src	Тип данных в источнике	varchar	50
type_dst	Тип данных в системе назначения	varchar	50

Таблица «DIC_TRANSACTION_TYPE» хранит маппинг типов первичной системы на типы ms sql.

Таблица 2.2 – Таблица «DIC_LAYER»

Идентификатор	Описание	Тип данного	Размер
id_layer	Id системы	smalint	–
name	Название слоя для отображения	varchar	255
name_s	Используется для формирования схем в слоях	varchar	50
GET_METADATA_SCRIPT	Скрипт получения метаданных из первичного слоя	varchar	max
CONN_STRING	Строка подключения к слою. Используется для получения данных первичного слоя и верификации скриптов	varchar	255
TR_SYSTEM_TYPE	Тип системы	varchar	50
IS_EXTERNAL	Признак внешней системы	bit	–
READONLY_DT_COLS	Признак только чтения для полей объекта	varchar	255
HIDDEN_DT_COLS	Скрытые колонки для полей объекта	varchar	255
ADD_ROW_CFG	Возможность добавлять строки в областях интерфейса соответственно 0,1,2 (для 1,2,3 областей главного окна)	varchar	50
DELETE_ROW_CFG	Возможность добавлять строки в областях интерфейса соответственно 0,1,2 (для 1,2,3 областей главного окна)	varchar	50
SHOW_INC	Возможность выбора инкрементальной/фулл загрузки	bit	–

Таблица «DIC_LAYER» хранит все слои участвующие в формировании скриптов.

Таблица 2.3 – Таблица «DIC_TRANSACTION_LAYER »

Идентификатор	Описание	Тип данного	Размер
ID_TRANSACTION_LAYER	ID транзакции	smalint	—
id_layer_src	Id системы источника	smalint	—
id_layer_dst	Id системы приемника	smalint	—

Таблица «DIC_TRANSACTION_LAYER» хранит последовательность перехода из слоя в слой. Используется линейная последовательность.

Таблица 2.4 – Таблица «DIC_TYPE_OBJECT»

Идентификатор	Описание	Тип данного	Размер
ID_TYPE_OBJECT	ID типа объекта	tinyint	—
NAME	Наименование	varchar	255

Таблица «DIC_TYPE_OBJECT » хранит список артефактов создаваемых на слое.

Таблица 2.5 – Таблица «DIC_PATTERN_NAMING»

Идентификатор	Описание	Тип данного	Размер
ID_PATTERN_NAMING	Id правила наименования	int	—
id_layer	Id системы	smalint	—
id_type_object	Id типа объекта	tinyint	—
shema_pattern	Правило наименования схемы	varchar	255
prefix_table	Правило наименования таблицы	varchar	255

Таблица «DIC_PATTERN_NAMING» хранит правила именования объектов в слое.

2.3 Конструирование пользовательского интерфейса

После запуска приложения открывается главная форма, на которой размещен компонент управления Wizard. Он позволяет реализовать панель навигации по страницам. Страницы являются контейнером для компонентов. С помощью страниц реализован пользовательский интерфейс на каждом из шагов формирования скриптов. Интерфейс разрабатываемого приложения предоставлен на рисунках 2.3 – 2.9.

На рисунке 2.3 показано главное окно приложения с начальной страницей «IntroPage».

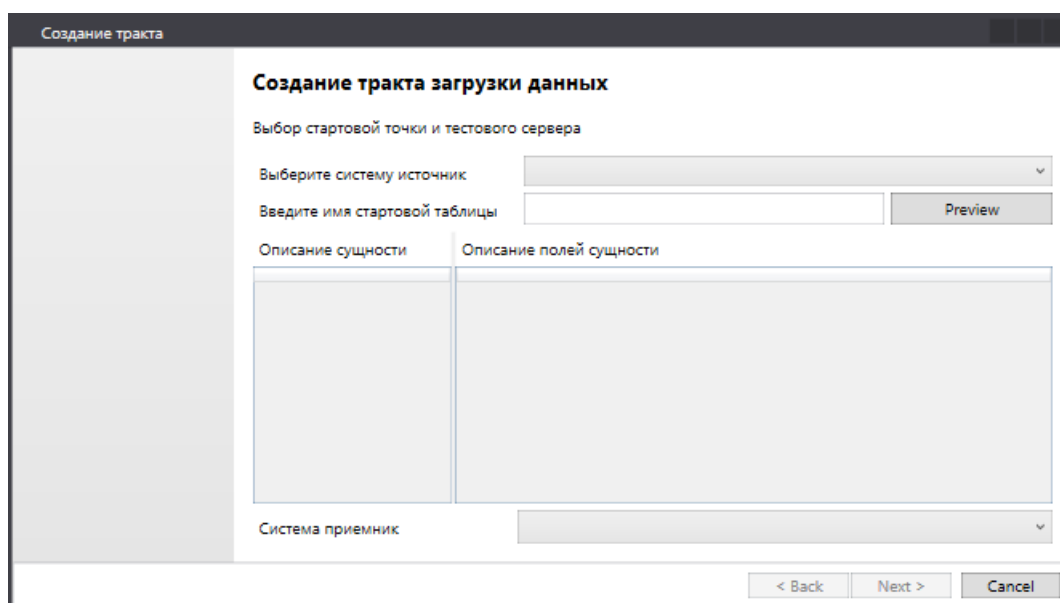


Рисунок 2.3 – Страница «IntroPage» главного окна

На странице «IntroPage» размещен элемент ComboBox, который позволяет выбрать систему источник (рисунок 2.4). Также можно увидеть поле для ввода стартовой таблицы (рисунок 2.5).

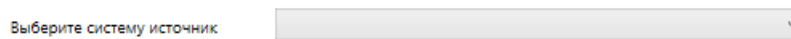


Рисунок 2.4 – Элемент ComboBox для выбора системы источник

Рисунок 2.5 – Поле для ввода стартовой таблицы

Ниже расположены три компонента (рисунок 2.6):

- 1) DataGrid описание сущности;
- 2) DataGrid описание полей сущности;
- 3) ComboBox для отображения системы приемника.

Рисунок 2.6 – Компоненты DataGrid и ComboBox

Навигация для данной страницы состоит из трех элементов (рисунок 2.7).

Рисунок 2.7 – Элементы навигации

При переходе от слоя к слою открываются страницы, которые спроектированы используя следующий принцип.

Страница делится на три области состоящих из элементов DataGrid:

- 1) поля объекта (таблиц/вью);
- 2) описание таблицы;
- 3) артефакты слоя (процедуры, триггеры и прочее).

На рисунке 2.8 изображен макет страницы.

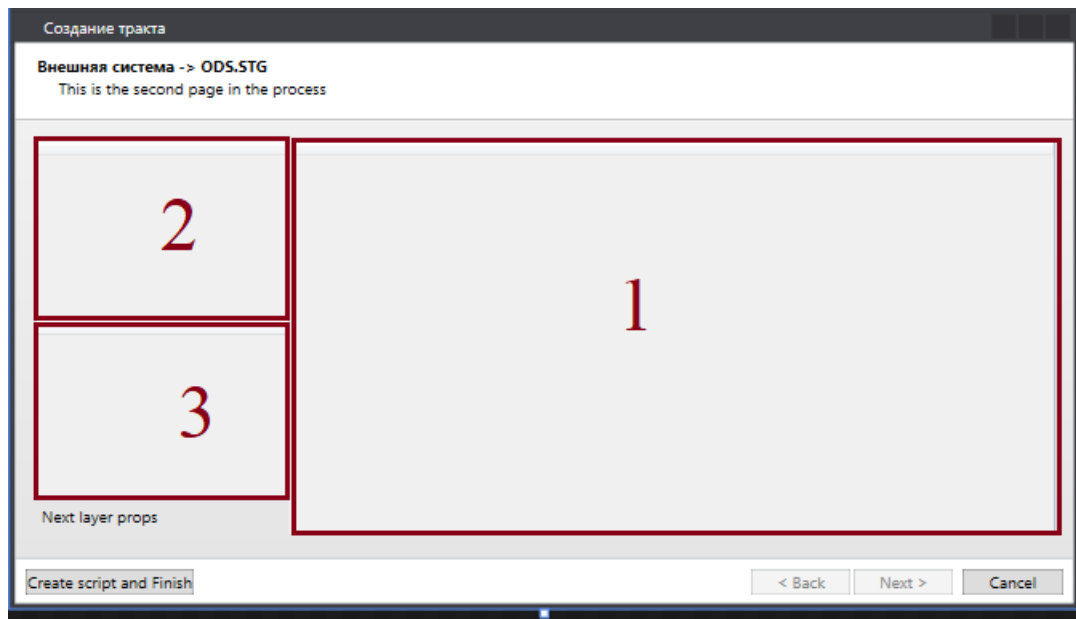


Рисунок 2.8 – Макет страницы

При двойном клике по полю DataGrid области 3 открывается окно редактирования сформированного скрипта (рисунок 2.9).

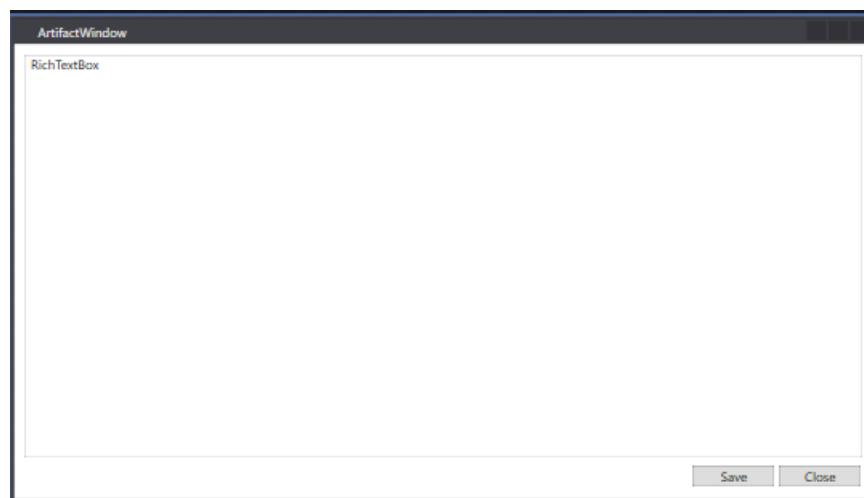


Рисунок 2.9 – Окно редактирования скрипта

Структура страниц приложения предоставлена на рисунке 2.10.

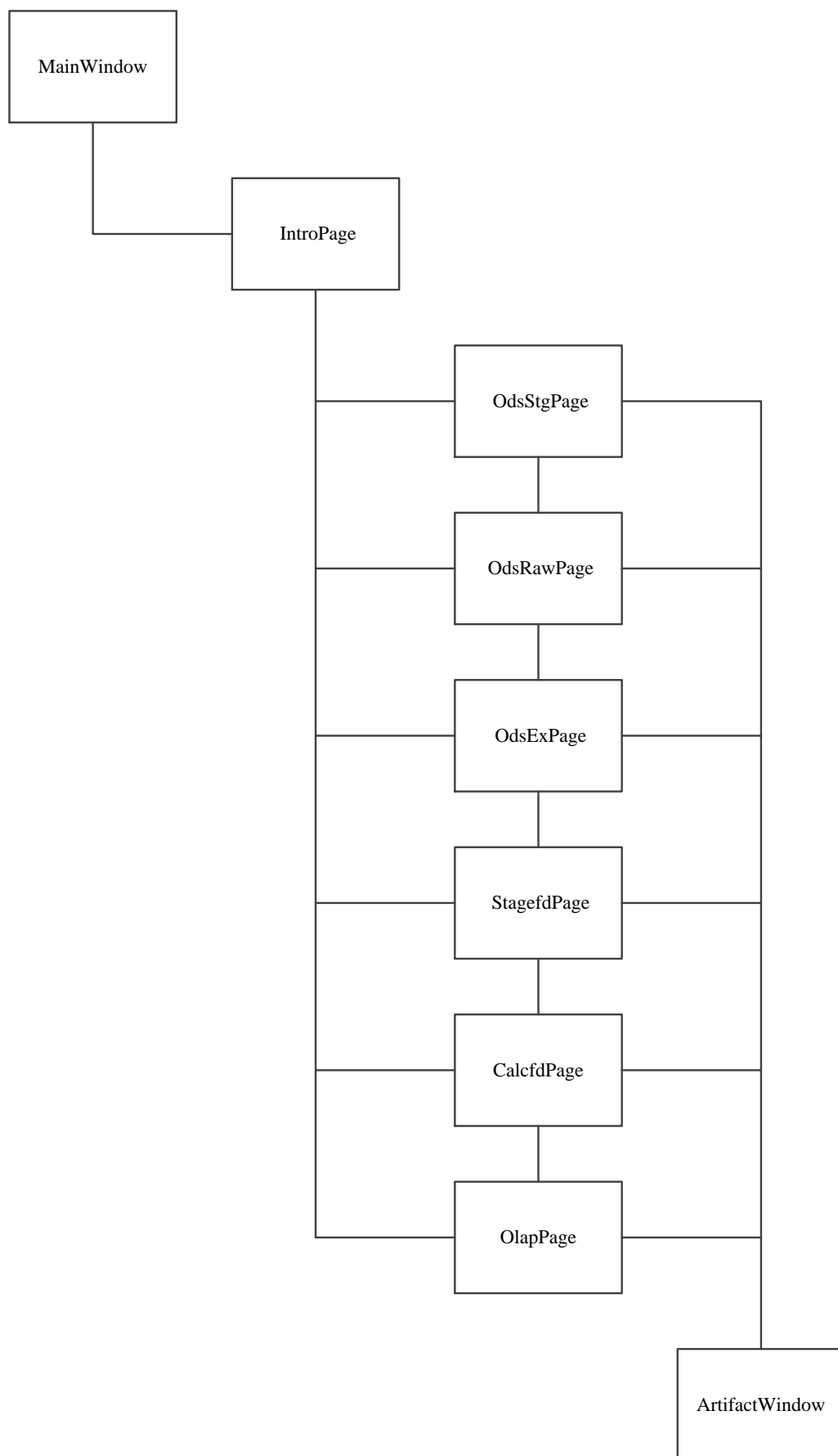


Рисунок 2.10 – Структура страниц приложения

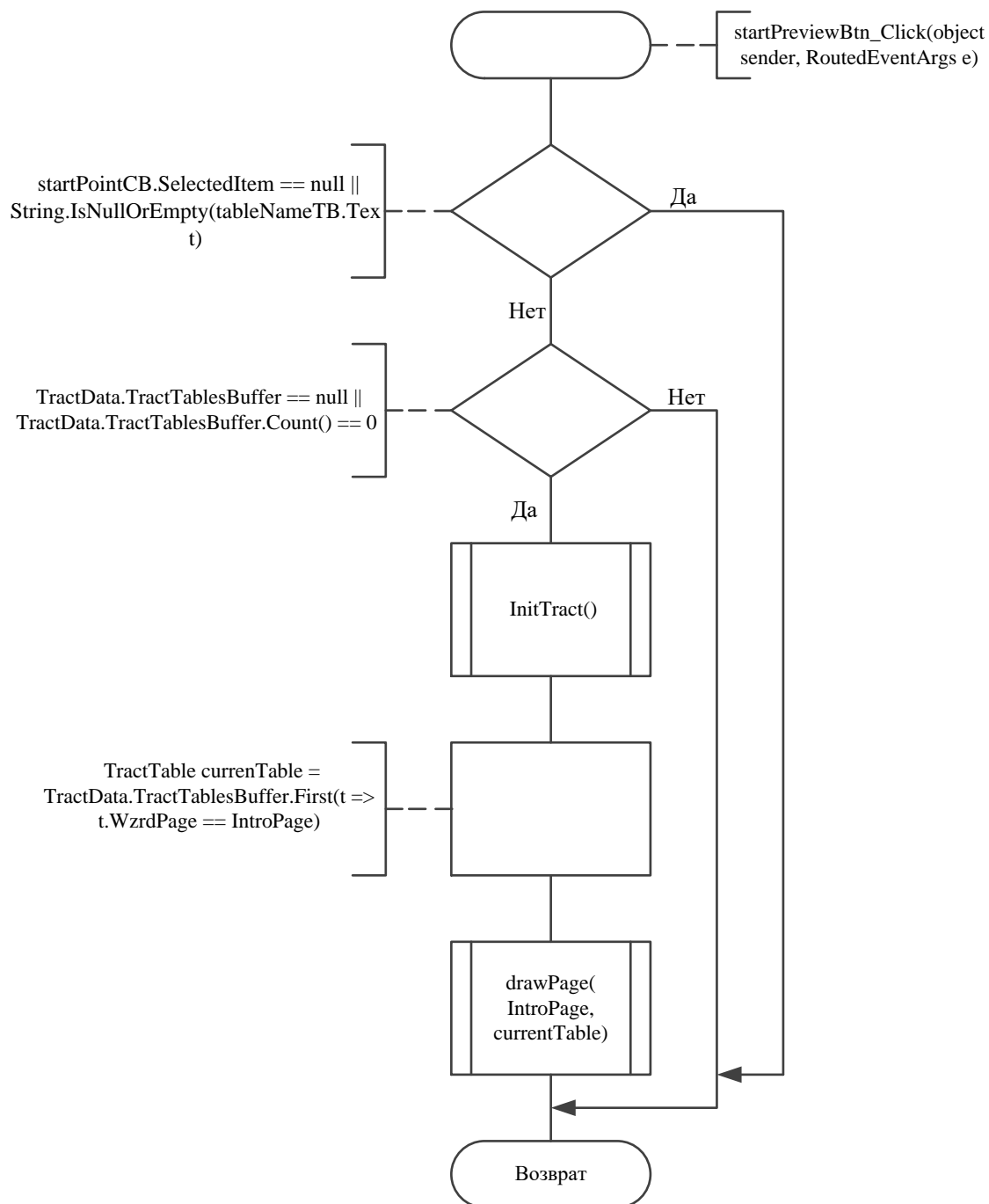
2.4 Схемы алгоритма программы и подпрограмм

2.4.1 Схемы алгоритма класса MainWindow

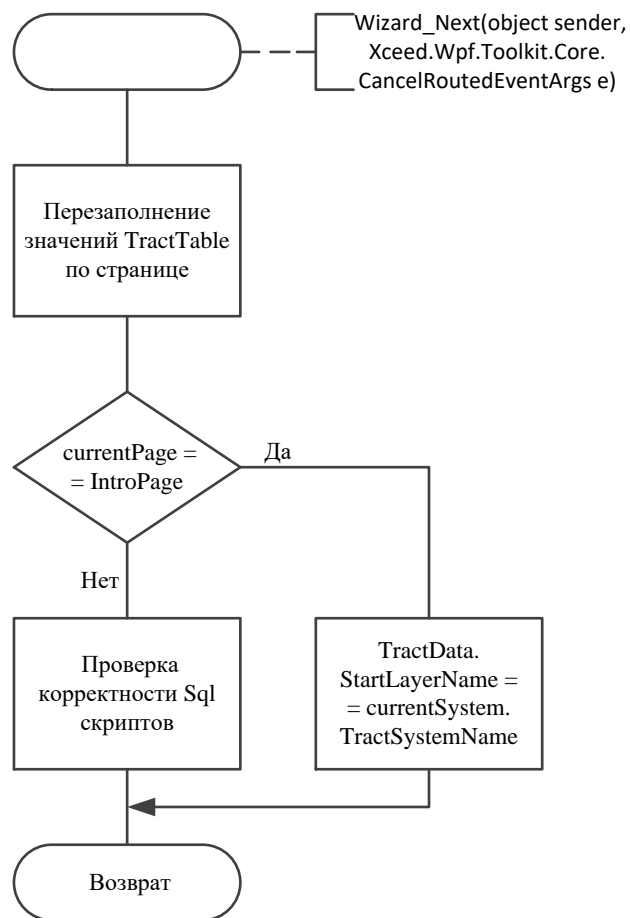
2.4.1.1 Схема алгоритма конструктора класса MainWindow



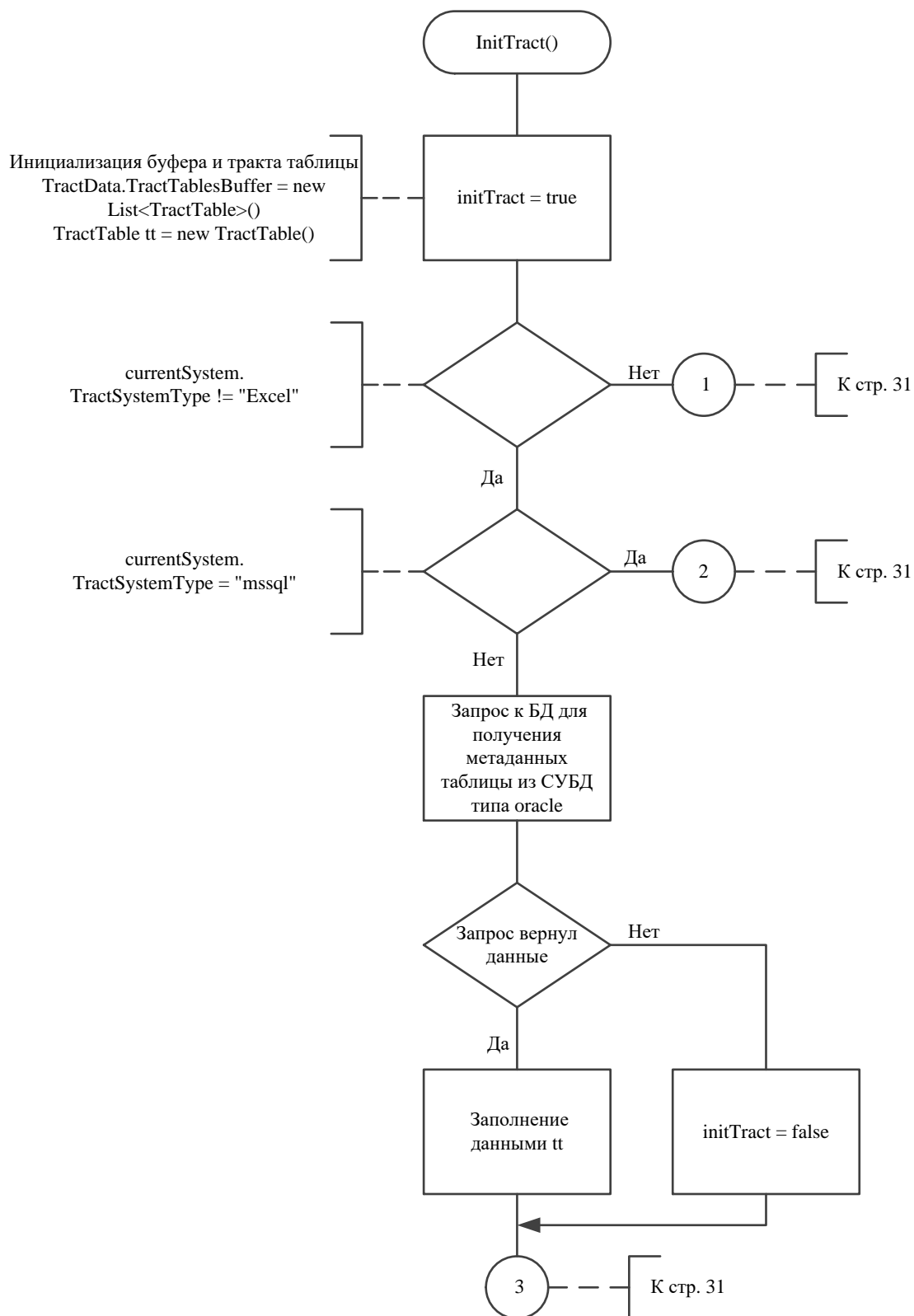
2.4.1.2 Схема алгоритма метода startPreviewBtn_Click

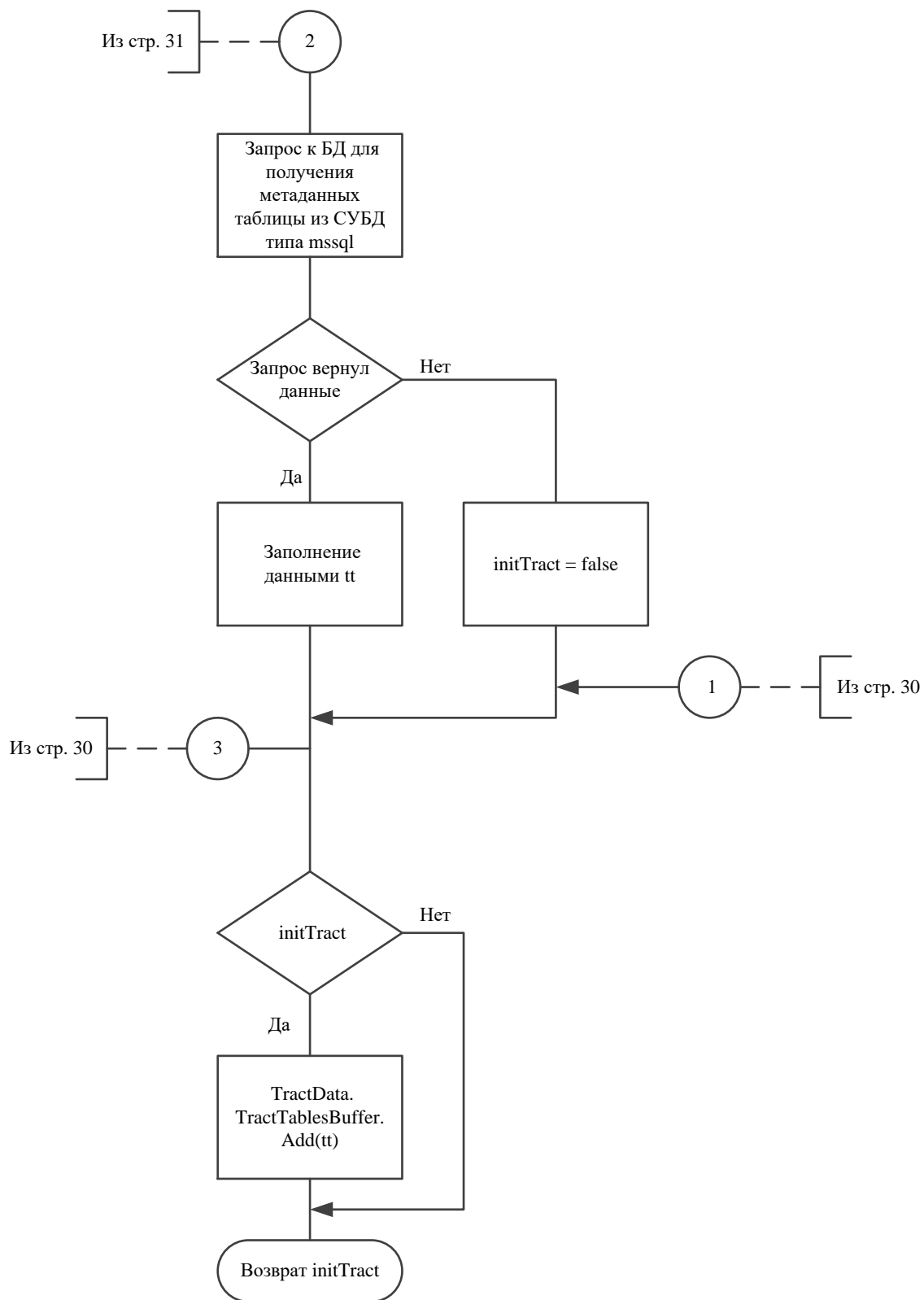


2.4.1.3 Схема алгоритма метода Wizard_Next

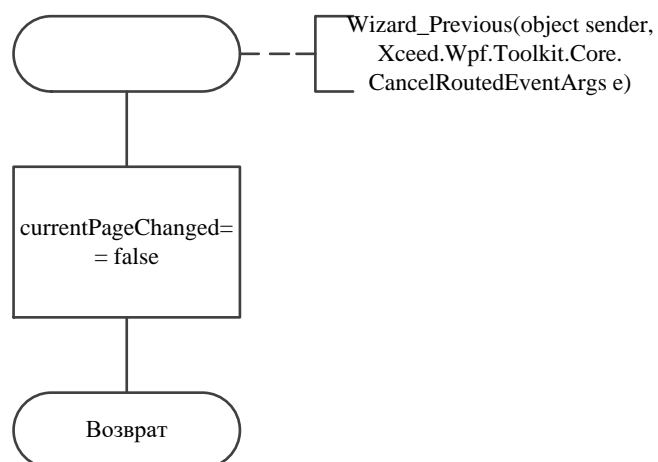


2.4.1.4 Схема алгоритма метода InitTract

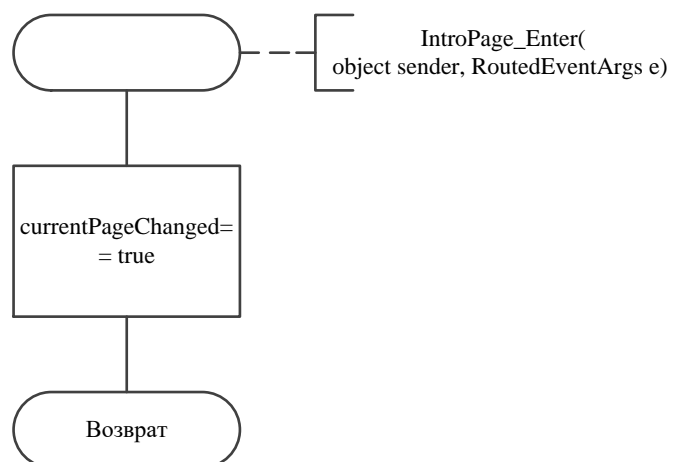




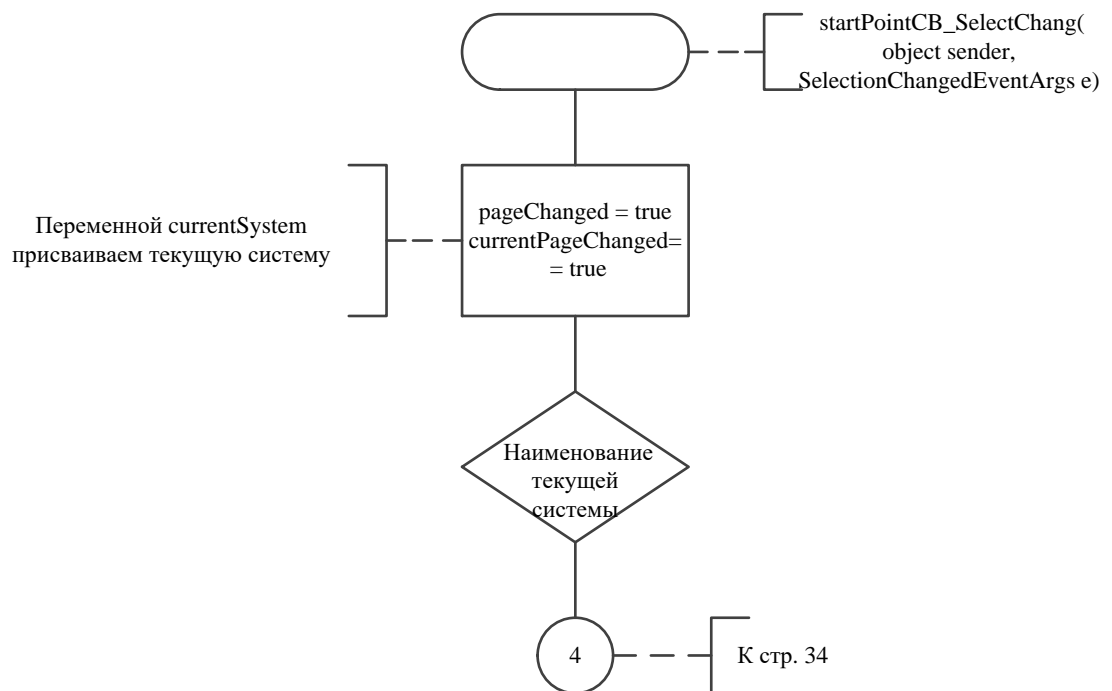
2.4.1.5 Схема алгоритма метода Wizard_Previous

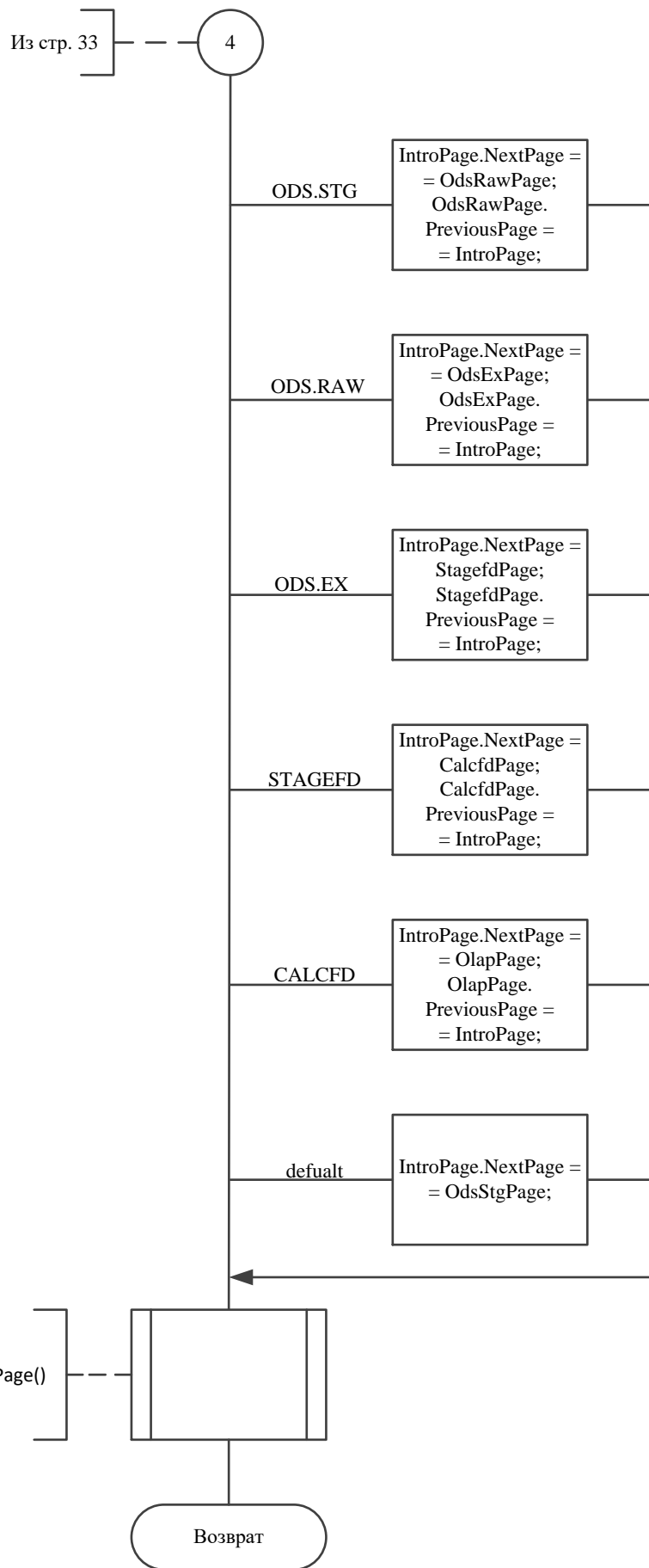


2.4.1.6 Схема алгоритма метода IntroPage_Enter

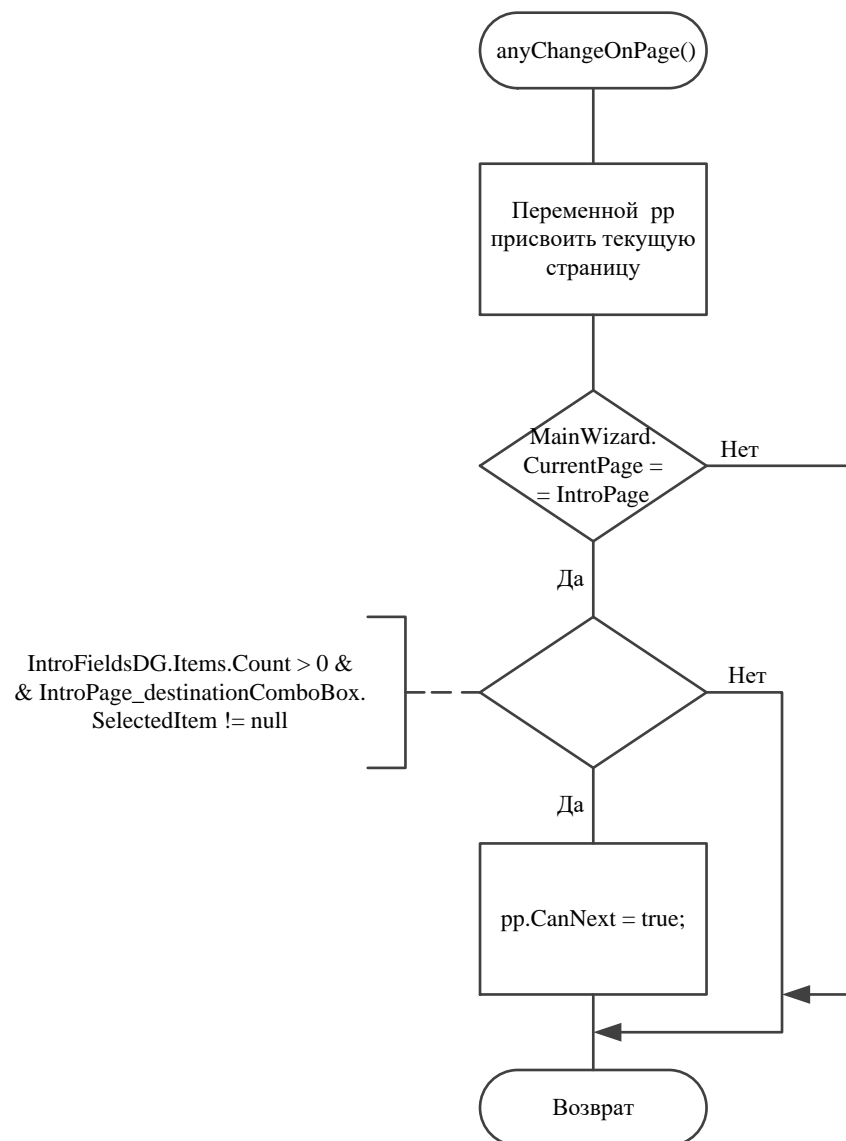


2.4.1.7 Схема алгоритма метода startPointCB_SelectChang

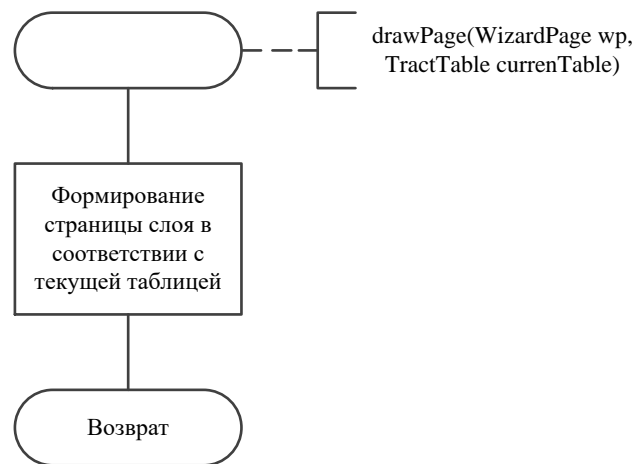




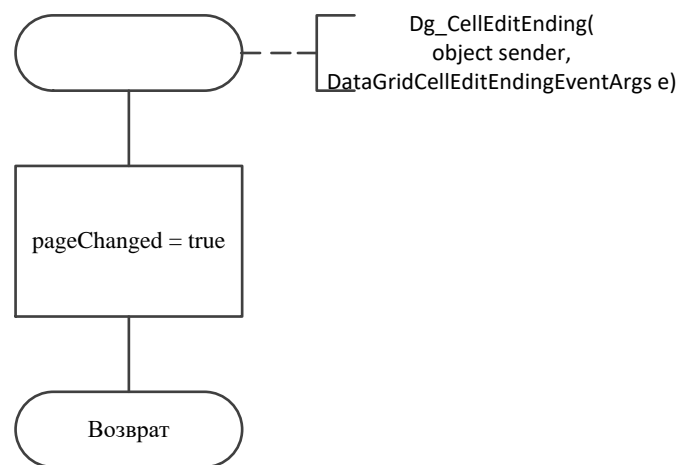
2.4.1.8 Схема алгоритма метода anyChangeOnPage



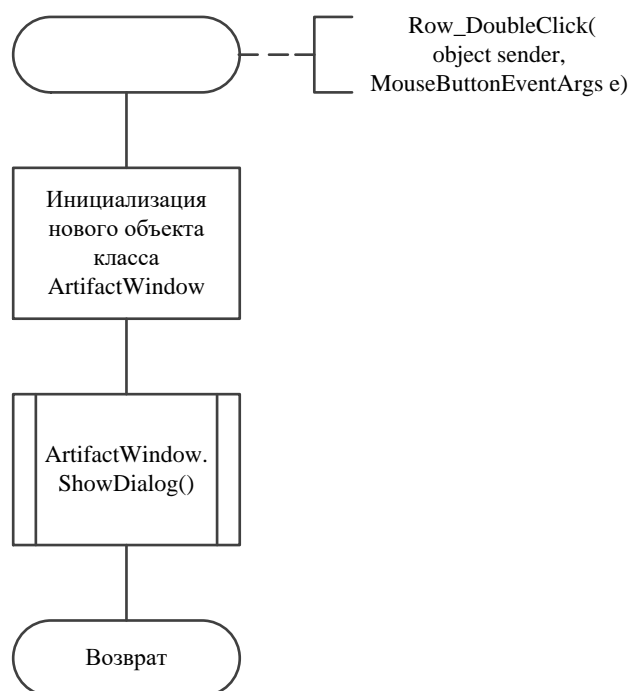
2.4.1.9 Схема алгоритма метода drawPage



2.4.1.10 Схема алгоритма метода Dg_CellEditEnding



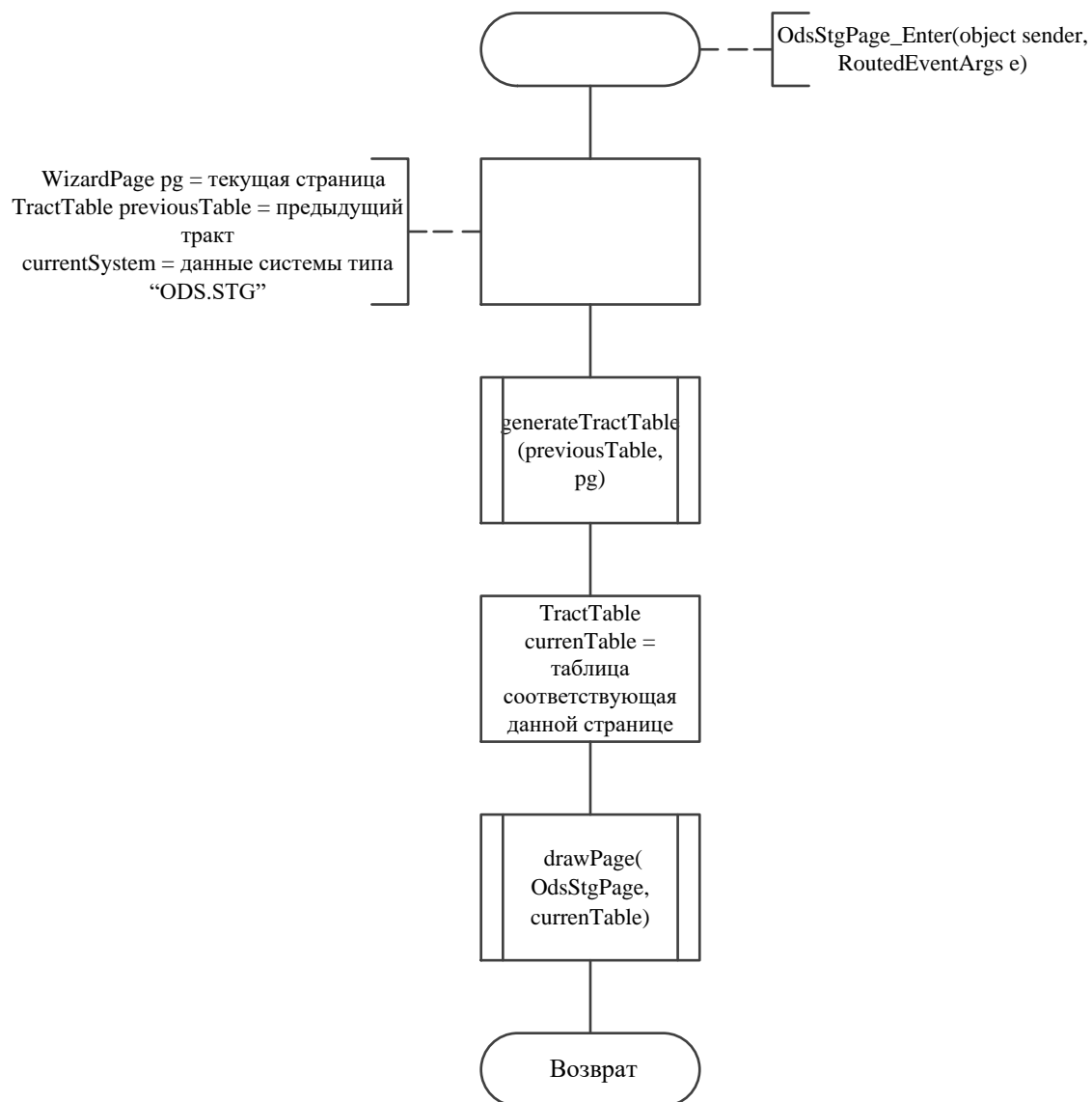
2.4.1.11 Схема алгоритма метода Row_DoubleClick



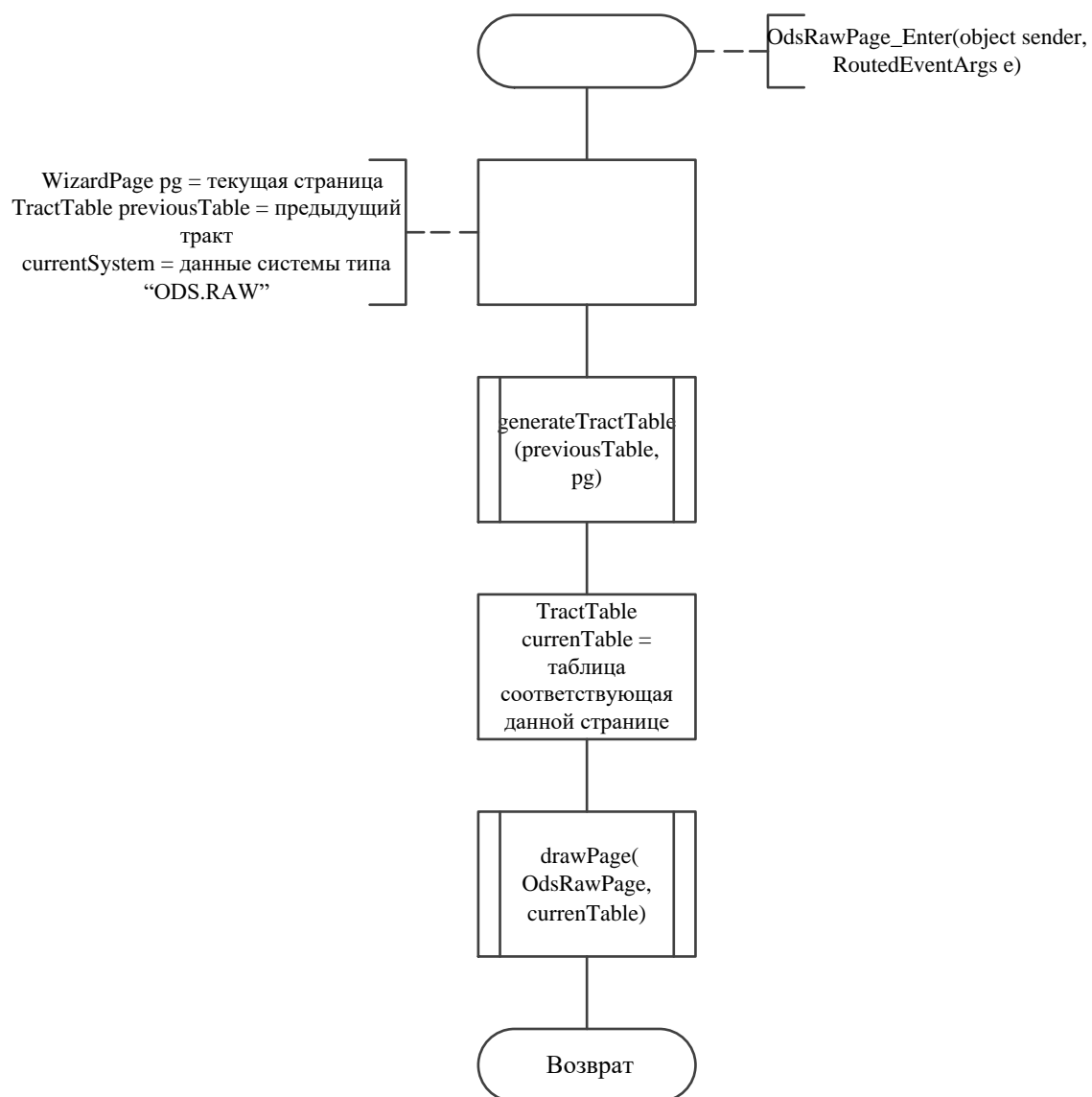
2.4.1.12 Схема алгоритма метода generateTrcatTable



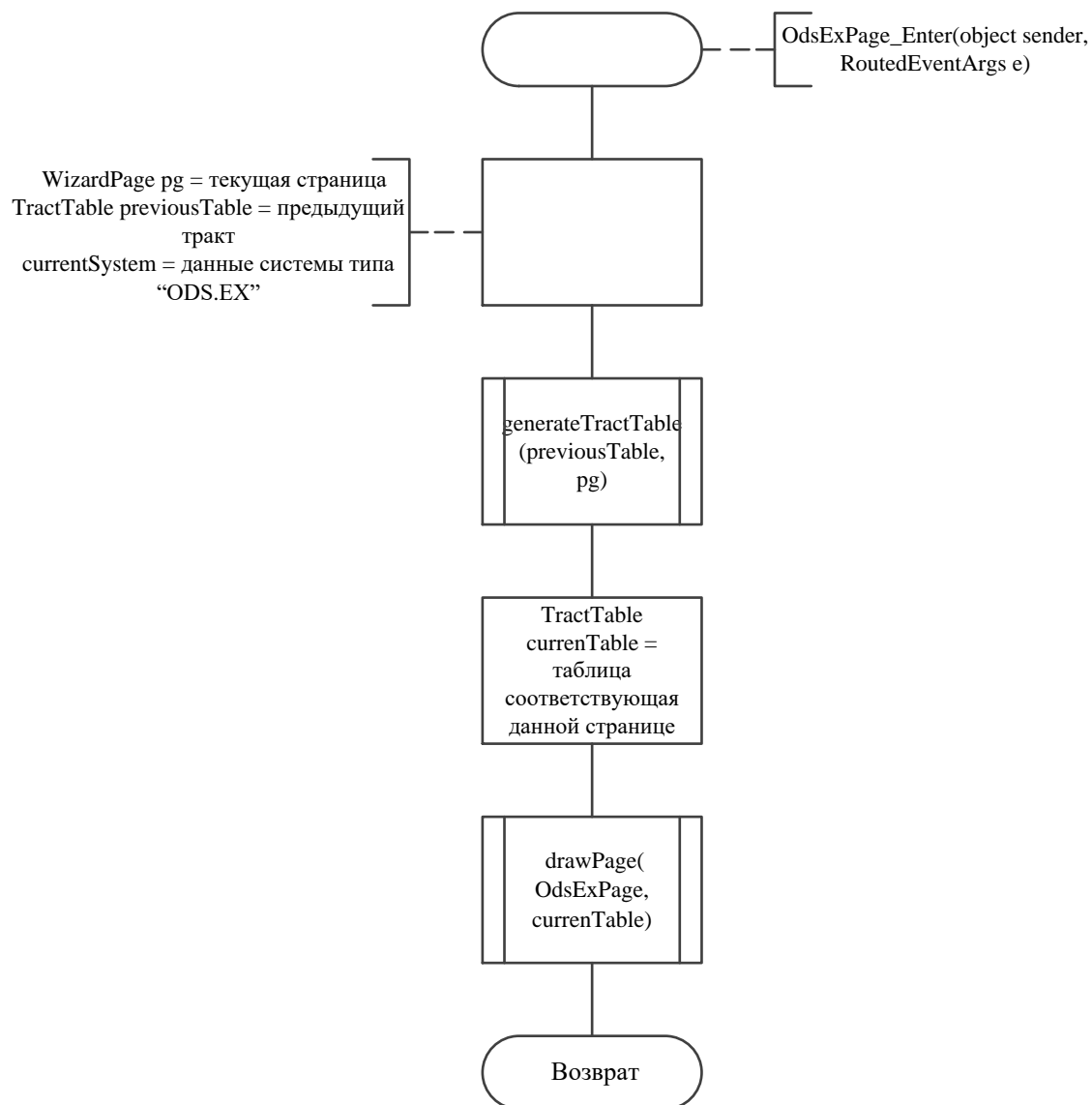
2.4.1.13 Схема алгоритма метода OdsStgPage_Enter



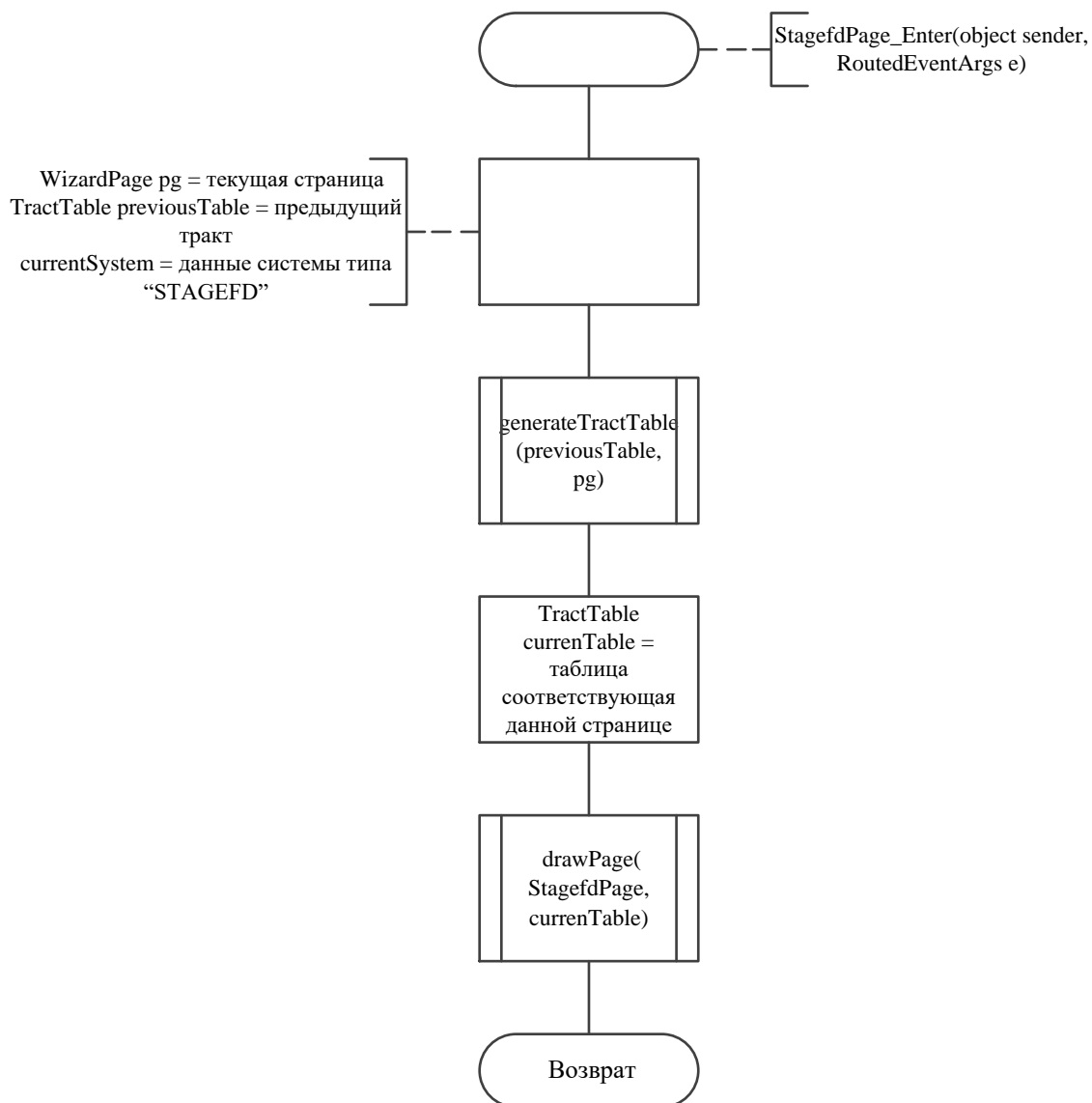
2.4.1.14 Схема алгоритма метода OdsRawPage_Enter



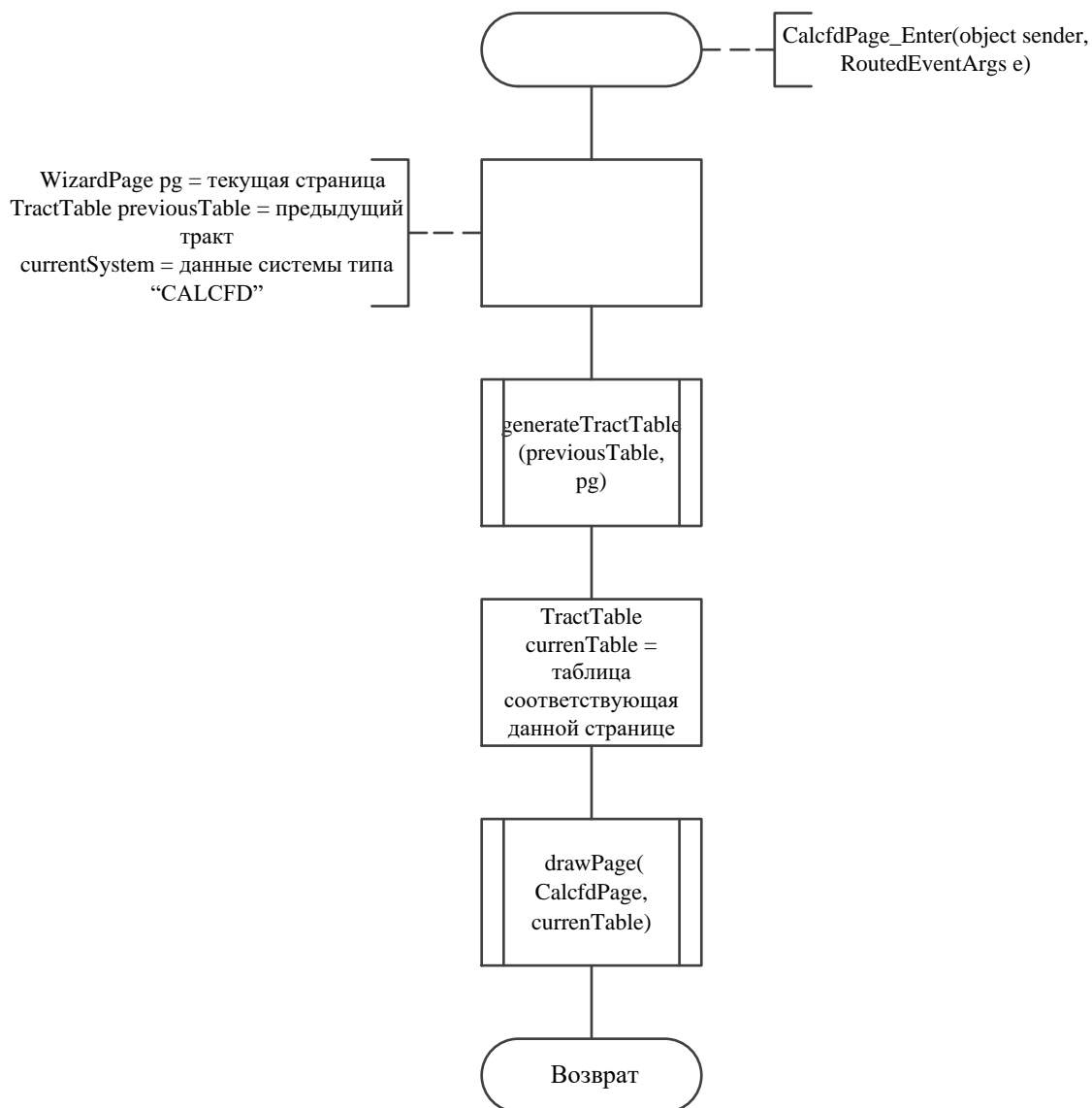
2.4.1.15 Схема алгоритма метода OdsExPage_Enter



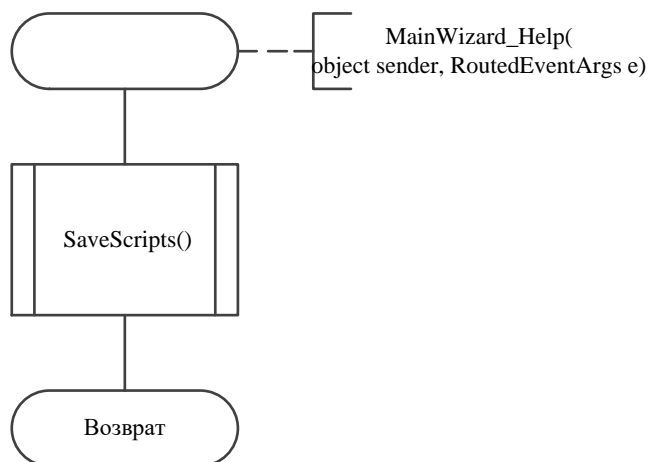
2.4.1.16 Схема алгоритма метода StagefdPage_Enter



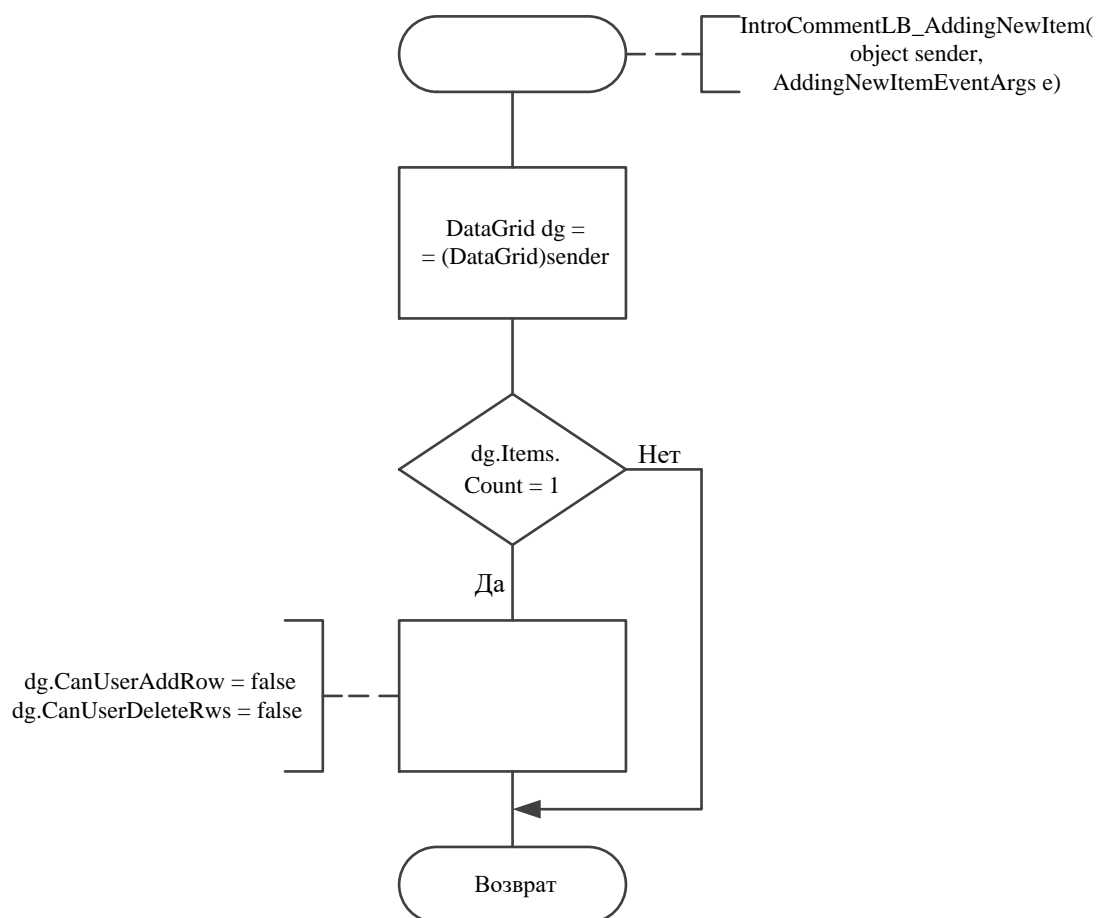
2.4.1.17 Схема алгоритма метода CalcfdPage_Enter



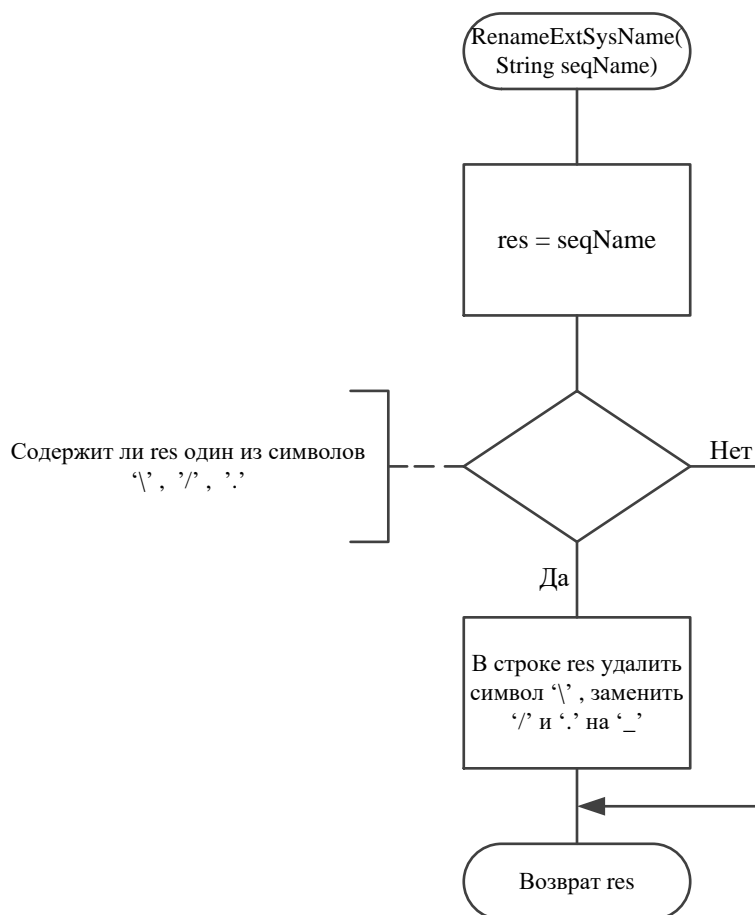
2.4.1.18 Схема алгоритма метода MainWizard_Help



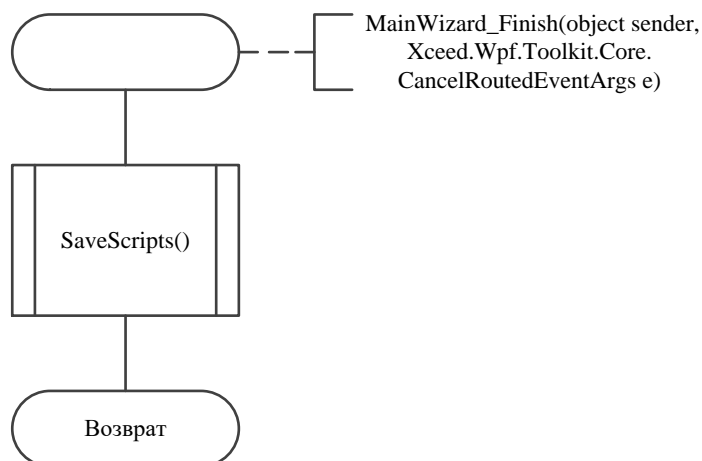
2.4.1.19 Схема алгоритма метода IntroCommentLB_AddingNewItem



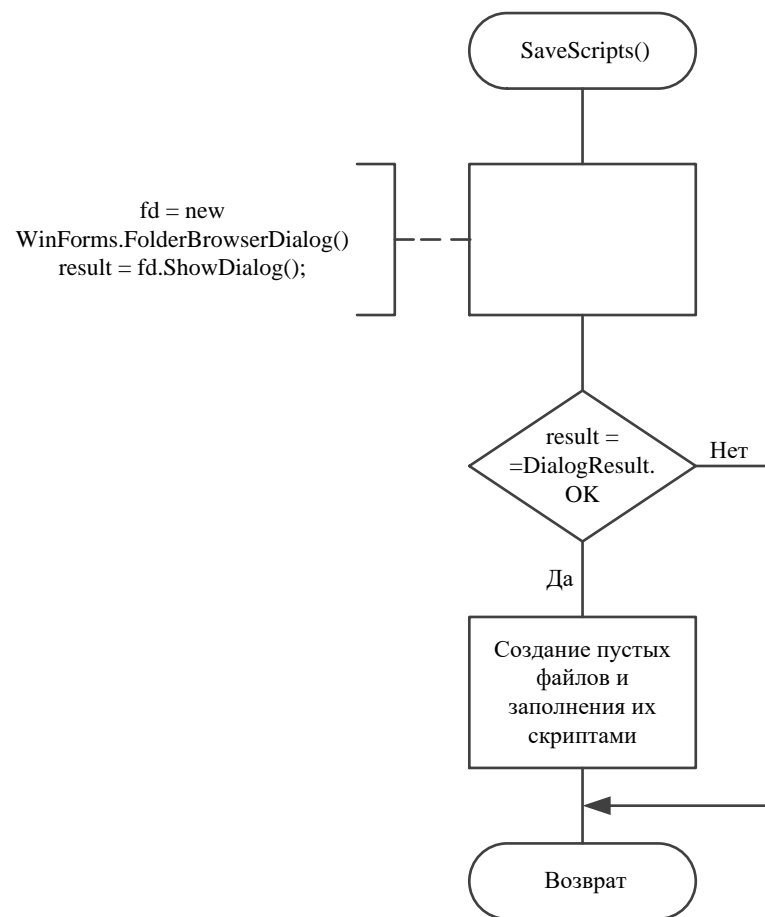
2.4.1.20 Схема алгоритма метода RenameExtSysName



2.4.1.21 Схема алгоритма метода MainWizard_Finish



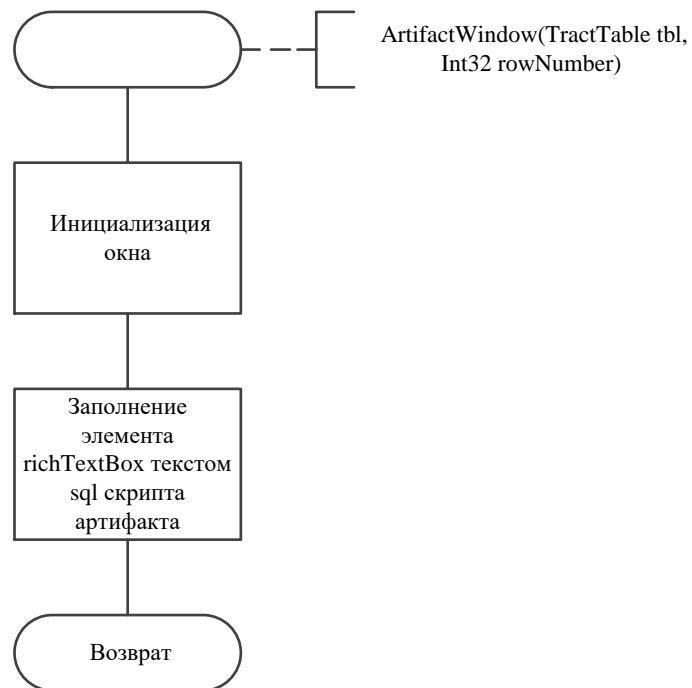
2.4.1.22 Схема алгоритма метода SaveScripts



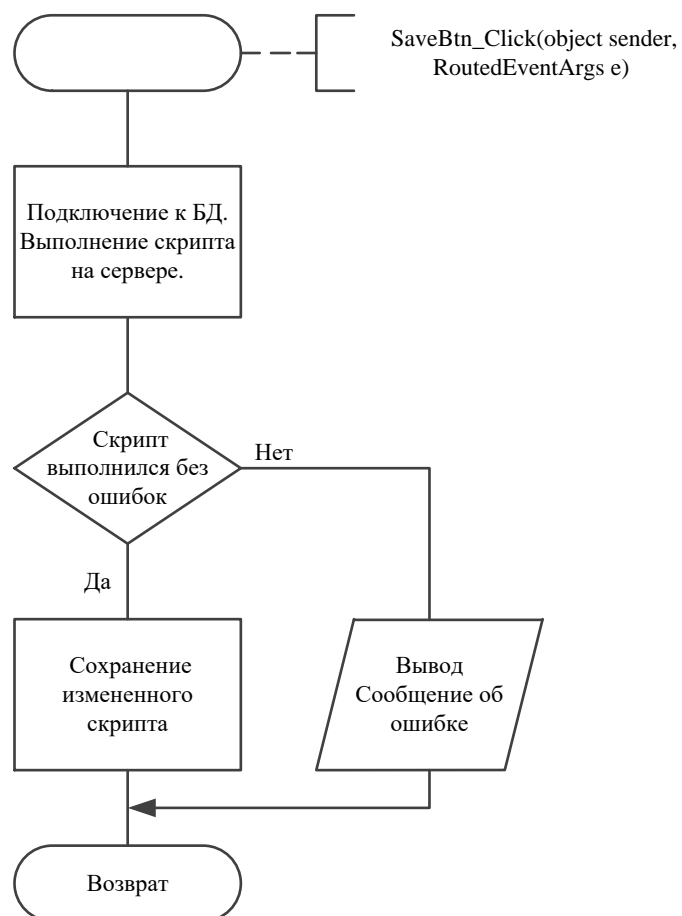
2.4.2 Схемы алгоритма класса ArtifactWindow

2.4.2.1 Схема алгоритма конструктора класса

ArtifactWindow



2.4.2.2 Схема алгоритма метода SaveBtn_Click



2.5 Отладка и тестирование программы

Отладка программы один из самых сложных этапов разработки программного обеспечения.

Отладка — это процесс локализации и исправления ошибок, обнаруженных при тестировании программного обеспечения. Локализацией называют процесс определения оператора программы, выполнение которого вызвало нарушение нормального вычислительного процесса. Для исправления ошибки необходимо определить ее причину, т. е. определить оператор или фрагмент, содержащие ошибку. Причины ошибок могут быть как очевидны, так и очень глубоко скрыты.

В соответствии с этапом обработки, на котором проявляются ошибки, различают следующие виды ошибок:

1) синтаксические ошибки - ошибки, фиксируемые компилятором (транслятором, интерпретатором) при выполнении синтаксического и частично семантического анализа программы;

2) ошибки компоновки — ошибки, обнаруженные компоновщиком (редактором связей) при объединении модулей программы;

3) ошибки выполнения - ошибки, обнаруженные операционной системой, аппаратными средствами или пользователем при выполнении программы.

Во время отладки данной программы были допущены следующие ошибки:

1) синтаксические ошибки:

- ошибки в названиях переменных;
- неправильное имя оператора.

Пример ошибки:

```
pageChang = false;
```

Исправленный вариант:

```
pageChanged = false;
```

2) ошибки компоновки:

- неверный список перечисляемых параметров.

Пример ошибки:

```
drawPage(IntroPage, currenTable);
```

Исправленный вариант:

```
drawPage(IntroPage, currenTable);
```

3) ошибки выполнения:

– несовпадение полученных результатов с ожидаемыми;

Пример ошибки:

```
stext = "CREATE TABLE " + TName + "." + TSchema + "(" +  
Environment.NewLine;
```

Исправленный вариант:

```
stext = "CREATE TABLE " + TSchema + "." + TName + "(" +  
Environment.NewLine;
```

Все ошибки были исправлены. Для того чтобы убедиться в правильности работы программы, было проведено тестирование.

Тестирование – это набор процедур и действий, предназначенных для демонстрации правильности работы программы в заданных режимах и внешних условиях. Цель тестирования – выявить наличие ошибок или убедительно продемонстрировать их отсутствие. Процесс тестирования проходит в 3 этапа:

- проверка поведения программы в нормальных условиях;
- проверка поведения программы в экстремальных условиях;
- проверка поведения программы в исключительных ситуациях.

Каждый из этапов предполагает задание определенного, характерного для данного этапа набора данных.

Для тестирования программы в нормальных условиях достаточно сформировать входные данные, при которых подразумевается безотказное выполнение программы. На рисунках 2.11 – 2.14 представлена работа программы в нормальных условиях.

Создание тракта

Создание тракта загрузки данных

Выбор стартовой точки и тестового сервера

Опишите сущность, как она есть в источнике

Выберите систему источник

RADIUS (Excel)

Введите имя стартовой таблицы

GRAPHS

Create

Описание сущности

Name	Value
MS_Description	ОЛК Графики

Описание полей сущности

Primary Key Number	Column Name	Column Type	Ext Properties	Comment
	ОрганизацийИИН	varchar(50)		Наименование организации, заключившей договор
	График	varchar(150)		Название графика
	Сумма	varchar(20)		Сумма платежа
	Номер	varchar(10)		Порядковый номер платежа
	Валюта	varchar(10)		Валюта платежа
	ДоговорGUID	varchar(36)		Идентификатор договора
	Дата	varchar(20)		Дата
	Статья	varchar(100)		Статья
	СпецификацияGUID	varchar(36)		Идентификатор спецификации
	Спецификация	varchar(100)		Тип спецификации
	Вид графика	varchar(50)		Тип графика
	Активность	varchar(10)		Признак активирования графика, без него - черновик.
1	Период	varchar(20)		Дата создания графика

Система приемник

ODS.STG (localhost)

< Back

Next >

Cancel

Рисунок 2.11 – Успешный выбор стартовой точки и тестового сервера

Создание тракта

Внешняя система - ODS.STG

Объекты для создания в слое ODS.STG (localhost)

Основной объект stg_radius.dbo_GRAPHS

Name	Value
MS_Description	ОЛК Графики
Source_table_name	dbo.GRAPH5
Источник данных	RADIUS
Пакет загрузки	
Владелец	

Primary Key Number	Column Name	Column Type	Ext Properties	Comment
	ОрганизацийИИН	VARCHAR(50)		Наименование организации, заключившей договор
	График	VARCHAR(150)		Название графика
	Сумма	VARCHAR(20)		Сумма платежа
	Номер	VARCHAR(10)		Порядковый номер платежа
	Валюта	VARCHAR(10)		Валюта платежа
	ДоговорGUID	VARCHAR(36)		Идентификатор договора
	Дата	VARCHAR(20)		Дата
	Статья	VARCHAR(100)		Статья
	СпецификацияGUID	VARCHAR(36)		Идентификатор спецификации
	Спецификация	VARCHAR(100)		Тип спецификации
	Вид графика	VARCHAR(50)		Тип графика
	Активность	VARCHAR(10)		Признак активирования графика, без него - черновик.
1	Период	VARCHAR(20)		Дата создания графика
	IS_DEL	BIT		Признак удаления

Name	Type	SqlText
INSERT INTO dbo.TBL_LOAD_DATE	INSERT SCRIPT	IF NOT EXISTS (S INSERT INTO dbo VALUES('radius.d

Next layer props

Increment

Create scripts and Finish

< Back

Next >

Cancel

Рисунок 2.12 – Успешное формирование тракта загрузки на слое ODS.STG

51

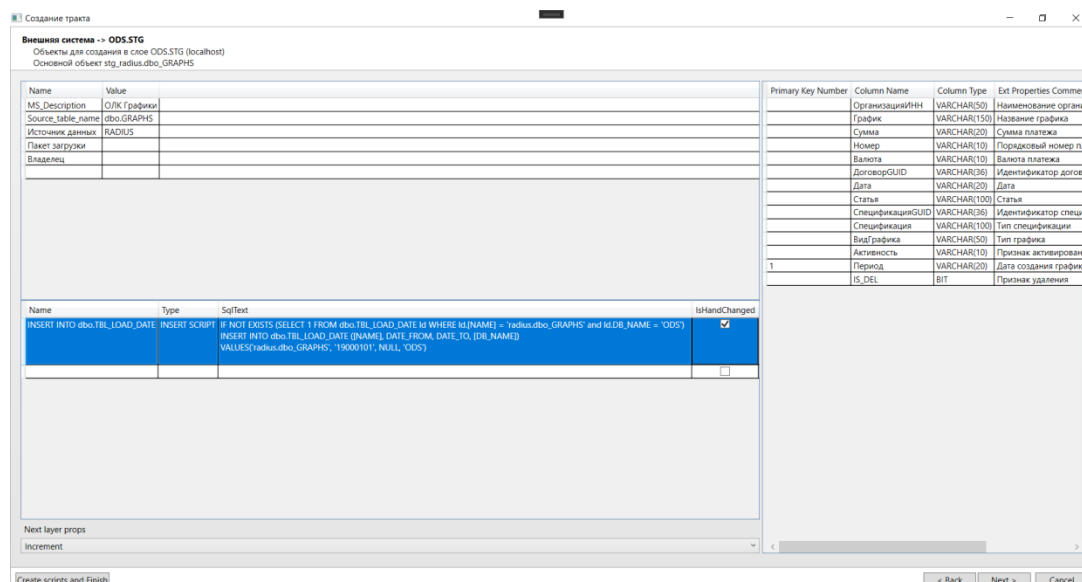


Рисунок 2.13 – Успешное изменение сформированного скрипта

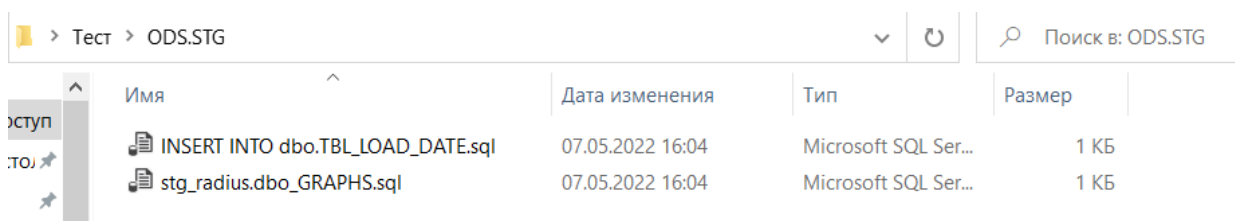


Рисунок 2.14 – Успешное сохранение скриптов

Тестирование было подтверждено визуально. Исходя из результата тестирования, можно сделать вывод, что программа в нормальных условиях работает корректно.

Тестирование в экстремальных условиях подразумевает входные данные, лежащие на концах интервала допустимых значений. В экстремальных условиях программа не была протестирована, так как в ней производится контроль всех вводимых параметров.

Произвести тестирование в исключительных ситуациях в данной программе невозможно, так как в ней используется защита от ошибок, исключающая возможность ввода неправильных данных.

Так как тестирование, в нормальных, экстремальных условиях и исключительных ситуациях не выявило наличие ошибок, можно сказать, что программа работает правильно.

2.6 Руководство пользователя

Данная система предназначена для автоматического формирования скриптов, создаваемых объектов на слое хранилища, при загрузке данных в него. Система будет использоваться в отделе банка.

Минимальные системные требования:

- Windows 7 и выше;
- процессор с тактовой частотой 1,8 ГГц и выше;
- ОЗУ от 2 ГБ;
- свободное место на жестком диске 100 МБ;
- пакет .NET Framework 4.5 и выше.

Для работы с программой необходимо следовать пошаговому руководству, представленному ниже.

Для запуска приложения, нужно запустить файл «OdsWizard.exe» (рисунок 2.15).

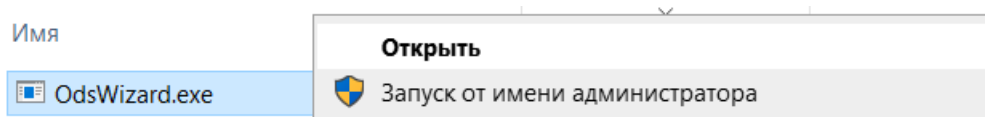


Рисунок 2.15 – Запуск программы

После запуска открывается главное окно приложения (рисунок 2.16).

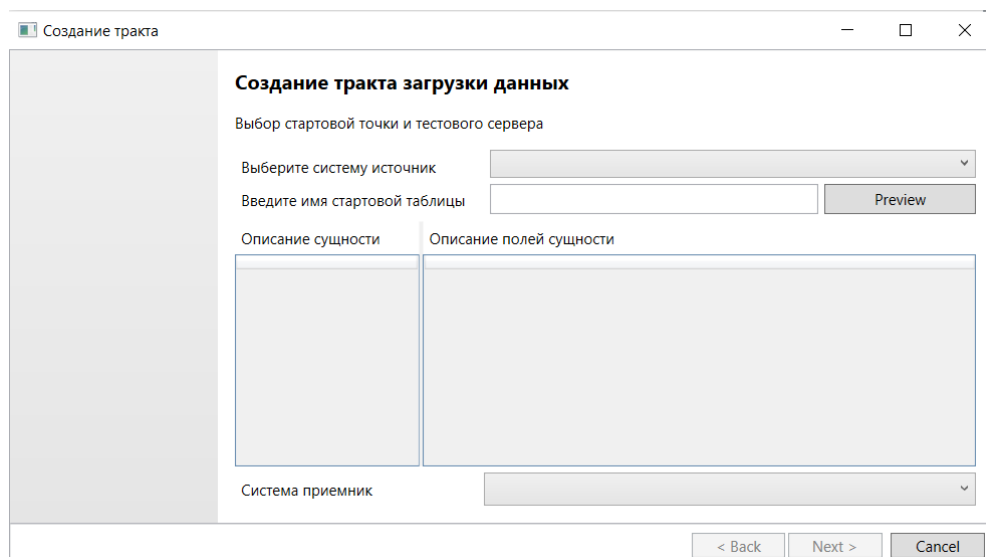


Рисунок 2.16 – Главное окно программы

Для начала формирования скриптов, необходимо выбрать систему источник и ввести имя стартовой таблицы, после чего кликнуть по кнопке «Preview» (рисунок 2.17).

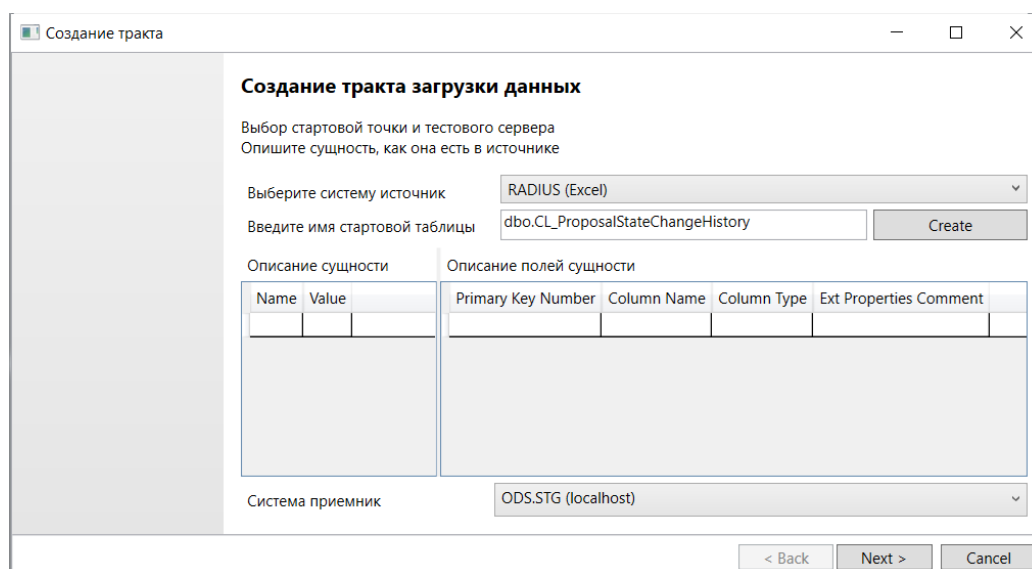


Рисунок 2.17 – Отображение данных из источника

Так как, выбрана система источник Excel, необходимо заполнить описание сущности и описание полей сущности. Кроме того в загружаемой таблице, обязательно должен быть хоть один Primary Key (рисунок 2.18).

Создание тракта

Создание тракта загрузки данных

Выбор стартовой точки и тестового сервера
Опишите сущность, как она есть в источнике

Выберите систему источник: **RADIUS (Excel)**

Введите имя стартовой таблицы: **dbo.CL_ProposalStateChangeHistory** Create

Описание сущности		Описание полей сущности			
Name	Value	Primary Key Number	Column Name	Column Type	Ext Properties Comment
MS_Description	История изменений статуса предложения	1	ProposalStateChangeHistoryID	INT	id изменения статуса
			ProposalID	INT	id предложения
			OldState	TINYINT	старый статус
			NewState	TINYINT	новый статус
			SessionID	UNIQUEIDENTIFIER	сессия пользователя
			ChangeDate	DATETIME	момент изменения
			ID_SECTION	TINYINT	секция загрузки

Система приемник: **ODS.STG (localhost)**

< Back Next > Cancel

Рисунок 2.18 – Окно с заполненными описаниями сущности

После заполнения всех полей, можно продолжить формирование, для этого необходимо нажать на кнопку «Next». На следующей странице в области 1 необходимо ввести или редактировать описания таблицы. В области 2 отражается таблица, которую необходимо установить на сервере. В области 3 автоматически сформируются артефакты для слоя ODS.STG. В области 4 можно выбрать параметр, на основе которого будет сформирован следующий слой. При разном выборе получается разный набор артефактов на следующем слое (рисунок 2.19).

Создание тракта

Внешняя система -> **ODS.STG**

Объекты для создания в слое ODS.STG (localhost)
Основной объект stg_radius.dbo.CL_ProposalStateChangeHistory

Name	Value
MS_Description	История изменений статуса предложения
Source table name	dbo.CL_ProposalStateChangeHistory
Имя таблицы	1
Имя таблицы	RADIUS
Пакет загрузки	
Владелец	

Primary Key Number	Column Name	Column Type	Ext Properties Comment
1	ProposalStateChangeHistoryID	INT	id изменения статуса
	ProposalID	INT	id предложения
	OldState	TINYINT	старый статус
	NewState	TINYINT	новый статус
	SessionID	UNIQUEIDENTIFIER	сессия пользователя
	ChangeDate	DATETIME	момент изменения
	ID_SECTION	TINYINT	секция загрузки
	IS_DEL	BIT	Признак удаления

2

Name	Type	SqlText
INSERT INTO dbo.TBL_LOAD_DATE	INSERT SCRIPT	IF NOT EXISTS (SELECT 1 FROM dbo.TBL_LOAD_DATE WHERE Id(NAME) = 'radius.dbo.CL_ProposalStateChangeHistory' and
		INSERT INTO dbo.TBL_LOAD_DATE (NAME) DATE_FROM DATE_TO (DB_NAME))
		VALUES('radius.dbo.CL_ProposalStateChangeHistory', '19000101', NULL, 'ODS')

3

Next layer props

Increment **4**

Create scripts and Finish < Back Next > Cancel

Рисунок 2.19 – Страница слоя ODS.STG

При двойном клике на строку с артефактом открывается окно для его правки и просмотра. Для сохранения изменений необходимо нажать на кнопку «Save». Для закрытия нажать кнопку «Close» (рисунок 2.20).



Рисунок 2.20 – Окно правки/просмотра артефакта

Далее, по нажатию на кнопку Next, отображается следующая страница (рисунок 2.21).

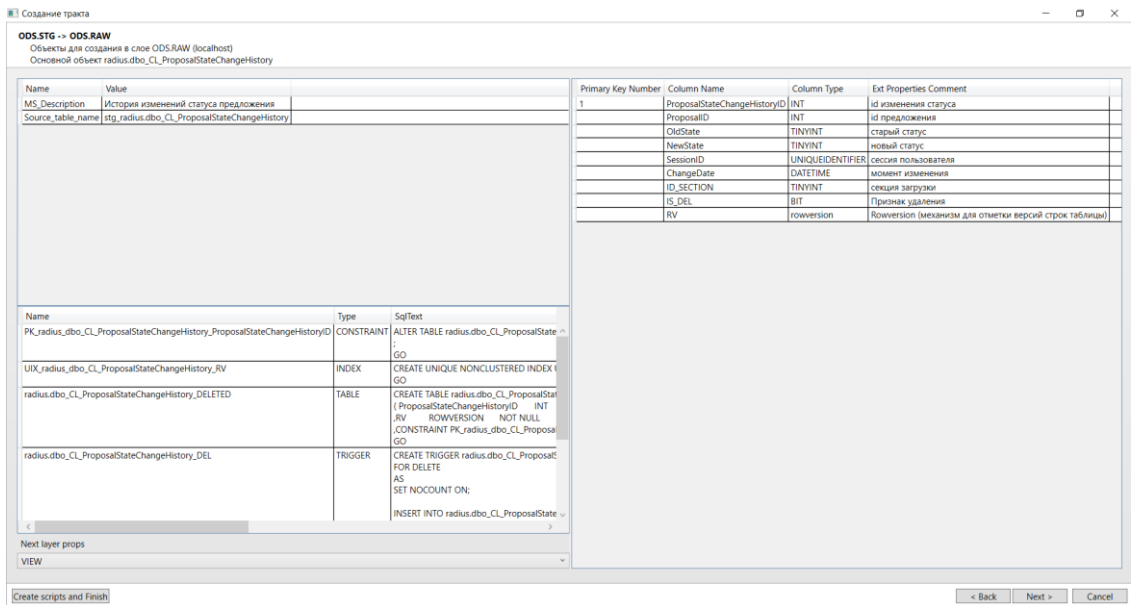


Рисунок 2.21 – Страница слоя ODS.RAW

При нажатии на кнопку «Create scripts and Finish» будет предложено выбрать место для сохранения скриптов и программа завершиться (рисунок 2.22).

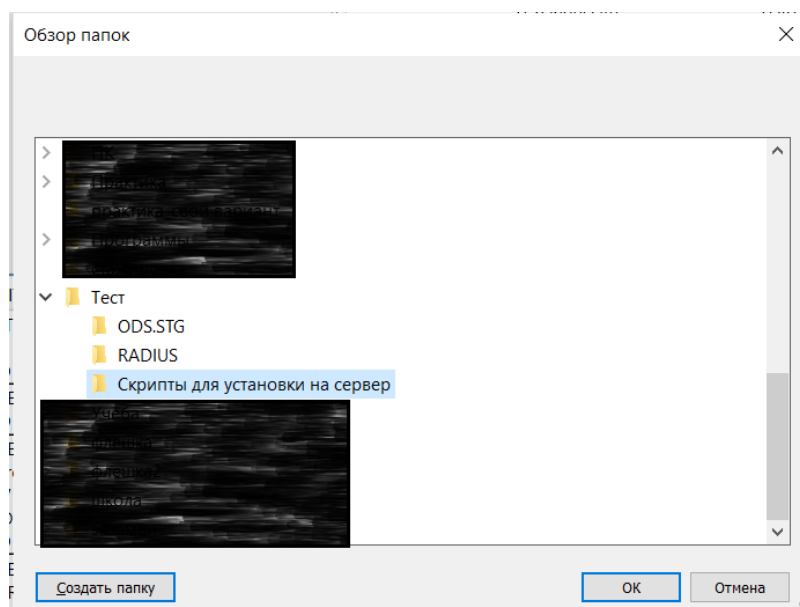


Рисунок 2.22 – Окно выбора места для сохранения скриптов

На рисунках 2.23 – 2.25 показано как сохранились скрипты.

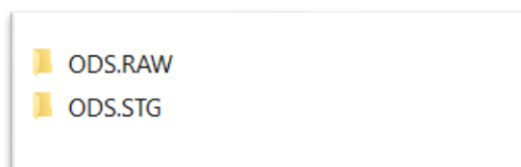


Рисунок 2.23 – Содержимое папки «Скрипты для установки на сервер»

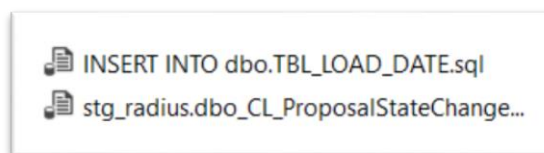


Рисунок 2.24 – Содержимое папки «ODS.STG»

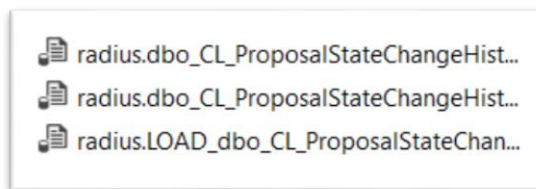


Рисунок 2.25 – Содержимое папки «ODS.RAW»

После установки скриптов на сервере (рисунок 2.26) можно снова запустить программу и продолжить работу с того слоя на котором остановились или со слоя выше (рисунок 2.27 – 2.28).

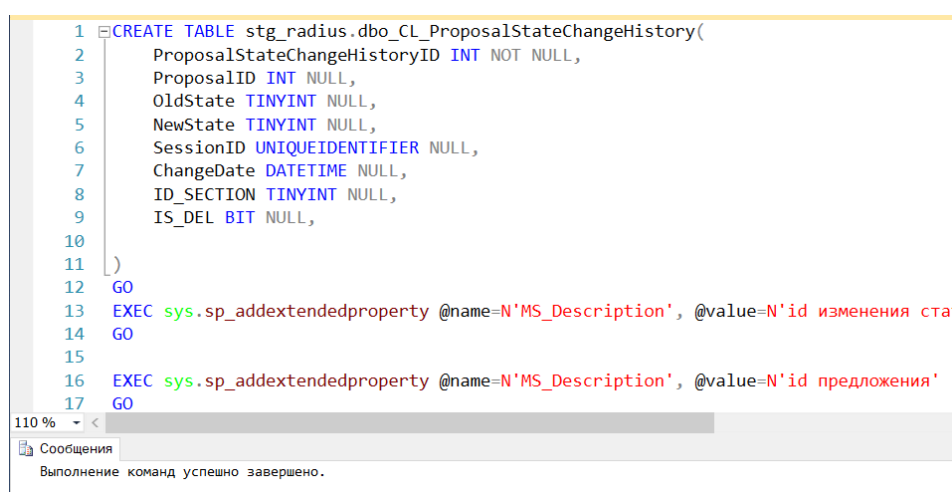


Рисунок 2.26 – Выполнение запроса для слоя «ODS.STG»

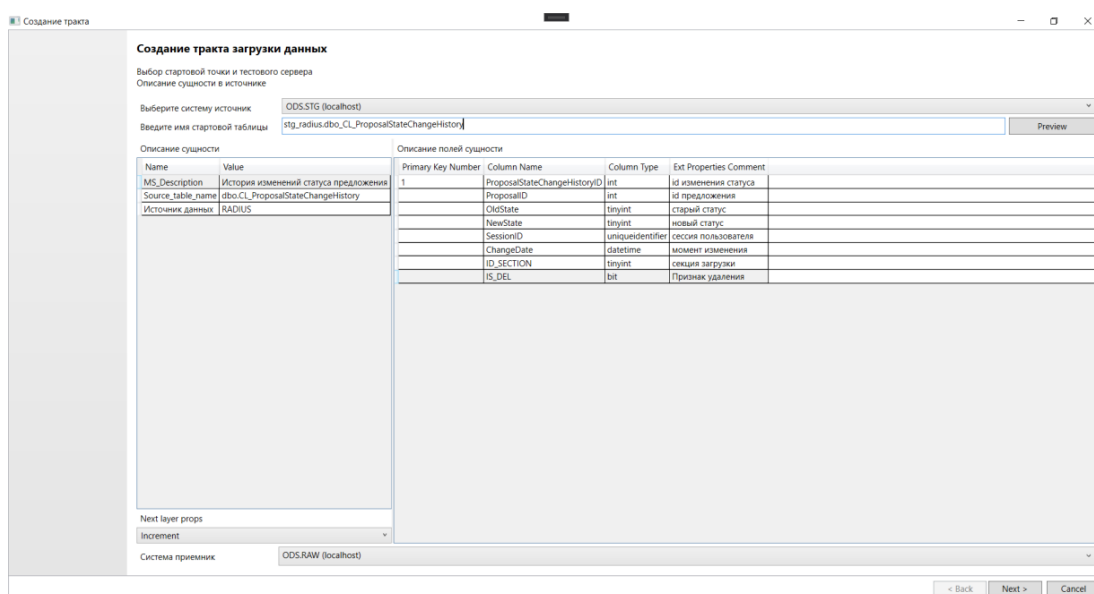


Рисунок 2.27 – Начало работы со слоя «ODS.STG»

Создание тракта

ODSRAW -> ODS.EX

Объекты для создания в слое ODS.EX (localhost)

Основной объект ex_radius.v_dbo_CL_ProposalStateChangeHistory

Name	Value
MS_Description	История изменений статуса предложения
Source_table_name	dbo.CL_ProposalStateChangeHistory
Источник данных	RADIUS
Source_table_name	radius.dbo.CL_ProposalStateChangeHistory

Primary Key Number	Column Name	Column Type
1	ProposalStateChangeHistoryID	INT
	ProposalID	INT
	OldState	TINYINT
	NewState	TINYINT
	SessionID	UNIQUEIDENTIFIER
	ChangeDate	DATETIME
	ID_SECTION	TINYINT
	IS_DEL	BIT
	RV	ROWVERSION

Name	Type	SqlText
INSERT INTO dbo.TBL_LOAD_DATE	INSERT SCRIPT	IF NOT EXISTS (SELECT 1 FROM dbo.TBL_LOAD_DATE WHERE ID(NAME) = 'ex_radius.v_dbo_CL_ProposalStateChangeHistory') INSERT INTO dbo.TBL_LOAD_DATE (NAME) (ID, NAME) RV (FROM: RV_TO) VALUES('ex_radius.v_dbo_CL_ProposalStateChangeHistory', 'STAGEFD', 0x0000000000000000, NULL)
ex_radius.v_dbo_CL_ProposalStateChangeHistory_increment	CREATE SCRIPT	CREATE VIEW ex_radius.v_dbo_CL_ProposalStateChangeHistory_increment AS SELECT s.ProposalStateChangeHistoryID s.ProposalID s.OldState s.NewState s.SessionID s.ChangeDate s.ID_SECTION s.IS_DEL s.RV FROM radius.dbo.CL_ProposalStateChangeHistory s

Create scripts and Finish

< Back

Next >

Cancel

Рисунок 2.28 – Результат работы программы на слое «ODS.EX»

Для завершения работы программы без сохранения нажмите на кнопку «Cancel».

2.7 Вывод по разделу

В данном разделе были разработаны архитектура программной системы и структура данных. Было проведено конструирование пользовательского интерфейса, а также построены схемы алгоритма программы и подпрограмм. Затем, была разработана сама программа с последующей ее отладкой. После завершения отладки программы, было проведено ее тестирование. Тестирование показало, что программа работает корректно. Далее было разработано и описано руководство пользователя.

3 Охрана труда

3.1 Техника безопасности при работе на компьютере

Для работников офиса должен быть проведен устный базовый инструктаж, в дальнейшем его печатный текст должен предоставляться для подробного изучения. Организация в обязательном порядке размещает информационный лист на видном месте. К работе на персональном компьютере допускаются лица, прошедшие инструктаж.

Перед началом работы, даже если речь идет о рабочем месте, которое используется каждый день и регулярно проверяется специалистами, нельзя терять бдительность. Перед тем, как включить компьютер, необходимо уделить пару минут следующим действиям:

1) нужно убедиться в том, что в зоне досягаемости отсутствуют оголенные провода и различные шнуры. Они не только мешают работе, но и несут потенциальную опасность в случае короткого замыкания;

2) предметы на столе не должны мешать обзору, пользованию мышкой и клавиатурой. Поверхность экрана должна быть абсолютно чистой;

3) на системном блоке не должно находиться никаких предметов, так как в результате вибраций может нарушиться работа устройства. Нужно убедиться в том, что никакие посторонние предметы не мешают работе системе охлаждения;

4) недопустимо включать персональный компьютер в удлинители и розетки, в которых отсутствует заземляющая шина;

5) запрещается начинать работу в помещениях с повышенной влажностью, а также в случае, если рядом присутствуют открытые источники влаги (лужи, мокрый пол). Включить технику можно лишь после полного высыхания окружающих предметов;

6) недопустимо часто включать и выключать компьютер в течение рабочего дня без особой нужды. Система просто не справляется с необходимостью быстро сворачивать все процессы.

При выполнении работы, поскольку персональный компьютер обладает всеми свойствами электрического прибора, то на него распространяются основные правила безопасности при взаимодействии с проводниками тока:

1) нельзя размещать какие-либо вещи на проводах, а также самостоятельно менять их расположение без особой нужды;

2) рекомендуется избегать расположения жидкостей рядом с модулями компьютера. Пользователи должны осознавать опасность потенциального замыкания в случае пролития воды на клавиатуру или системный блок. Нельзя работать на ПК с мокрыми руками;

3) нельзя очищать поверхность компьютера от загрязнений, когда он находится во включенном состоянии;

4) в помещении с компьютерами непозволительно употреблять пищу непосредственно на рабочем месте;

5) при ощущении даже незначительного запаха гари, нужно как можно быстрее выключить ПК из сети и обратиться к человеку ответственному за обслуживание компьютерной техники.

Работа с персональным компьютером таит в себе множество угроз для здоровья человека. Чтобы минимизировать это влияние необходимо соблюдать следующие рекомендации:

1) расстояние между глазами пользователя и экраном составляет не менее полуметра. Но пользователь должен быть в состоянии дотянуться кончиками пальцев до верхнего края монитора;

2) стул стоит таким образом, что бы спина лишь немного упиралась в его спинку. Высота сидения позволяет держать ровную осанку;

3) локти не висят в воздухе, а комфортно располагаются на подлокотниках кресла или столешнице. Их позиция существенно не меняется при передвижении мышки;

4) ноги упираются в твердую поверхность, распрямлены вперед, а не подогнуты под себя;

5) чрезвычайно важна периодическая зарядка. Каждый час нужно вставать с кресла, разминать мышцы и суставы. Ведь, несмотря на неподвижность, они

испытывают огромную нагрузку, пребывая в неестественном положении. Обязательно нужно делать разминку для глаз: круговые и линейные движения открытыми глазами, моргание и расфокусирование.

По окончании работы, нужно выполнить следующие действия:

- 1) правильно закрыть все программы и окна;
- 2) выключить компьютер с использованием алгоритма, установленного производителем;
- 3) обесточить периферийное оборудование;
- 4) убедиться в отключении техники;
- 5) выполнить очистку рабочих поверхностей влажной тканью.

О неисправности оборудования и других замечаний по работе с ПК сообщить непосредственному руководителю или лицам, осуществляющим техническое обслуживание оборудования.

3.2 Требования к помещению (машинного зала, ВЦ)

Помещения ВЦ, их размеры (площадь, объем) должны в первую очередь соответствовать количеству работающих и размещаемому в них комплекту технических средств. В них предусматриваются соответствующие параметры температуры, освещения, чистоты воздуха, обеспечивают изоляцию, от производственных шумов и т.п. Для обеспечения нормальных условий труда санитарные нормы СН 245-71 устанавливают на одного работающего, объем производственного помещения не менее 15 м³, площадь помещения выгороженного стенами или глухими перегородками не менее 4,5 м².

Площадь машинного зала должна соответствовать площади необходимой по заводским техническим условиям данного типа ЭВМ.

Высота зала над технологическим полом до подвесного потолка должна быть 3 - 3,5 м. Расстояние подвесным и основным потолками при этом должно быть 0,5 - 0,8 м. Высоту подпольного пространства принимают равной 0,2 - 0,6 м.

В ВЦ, как правило, применяется боковое естественное освещение. Рабочие комнаты и кабинеты должны иметь естественное освещение. В остальных помещениях допускается искусственное освещение.

В тех случаях, когда одного естественного освещения не хватает, устанавливается совмещенное освещение. При этом дополнительное искусственное освещение применяется не только в темное, но и в светлое время суток.

Искусственное освещение по характеру выполняемых задач делится на рабочее, аварийное, эвакуационное.

Рациональное цветовое оформление помещения направленно на улучшение санитарно-гигиенических условий труда, повышение его производительности и безопасности. Окраска помещений ВЦ влияет на нервную систему человека, его настроение и, в конечном счете, на производительность труда. Основные производственные помещения целесообразно окрашивать в соответствии с цветом технических средств. Освещение помещения и оборудования должно быть мягким, без блеска.

3.3 Мероприятия по противопожарной технике

Пожары в ВЦ представляют особую опасность, так как сопряжены с большими материальными потерями. Как известно пожар может возникнуть при взаимодействии горючих веществ, окисления и источников зажигания.

Горючими компонентами на ВЦ являются: строительные материалы для акустической и эстетической отделки помещений, перегородки, двери, полы, изоляция кабелей и др.

Противопожарная защита - это комплекс организационных и технических мероприятий, направленных на обеспечение безопасности людей, на предотвращение пожара, ограничение его распространения, а также на создание условий для успешного тушения пожара.

Источниками зажигания в ВЦ могут быть электронные схемы от ЭВМ, приборы, применяемые для технического обслуживания, устройства электропитания, кондиционирования воздуха, где в результате различных нарушений образуются перегретые элементы, электрические искры и дуги, способные вызвать загорания горючих материалов.

На рабочем месте запрещается иметь огнеопасные вещества.

В помещениях запрещается:

- 1) зажигать огонь;
- 2) включать электрооборудование, если в помещении пахнет газом;
- 3) курить;
- 4) сушить что-либо на отопительных приборах;
- 5) закрывать вентиляционные отверстия в электроаппаратуре

Источниками воспламенения являются:

- а) искра при разряде статического электричества
- б) искры от электрооборудования
- в) искры от удара и трения
- г) открытое пламя

При возникновении пожароопасной ситуации или пожара персонал должен немедленно принять необходимые меры для его ликвидации, одновременно оповестить о пожаре администрацию.

К средствам тушения пожара, предназначенных для локализации небольших загораний, относятся пожарные стволы, внутренние пожарные водопроводы, огнетушители, сухой песок, асбестовые одеяла и т. п.

В зданиях ВЦ пожарные краны устанавливаются в коридорах, на площадках лестничных клеток и входов. Вода используется для тушения пожаров в помещениях программистов, библиотеках, вспомогательных и служебных помещениях. Применение воды в машинных залах ЭВМ, хранилищах носителей информации, помещениях контрольно-измерительных приборов ввиду опасности повреждения или полного выхода из строя дорогостоящего оборудования возможно в исключительных случаях, когда пожар принимает угрожающе крупные размеры. При этом количество воды должно быть минимальным, а устройства ЭВМ необходимо защитить от попадания воды, накрывая их брезентом или полотном.

Для тушения пожаров на начальных стадиях широко применяются огнетушители. По виду используемого огнетушащего вещества огнетушители подразделяются на следующие основные группы.

Пенные огнетушители, применяются для тушения горящих жидкостей, различных материалов, конструктивных элементов и оборудования, кроме электрооборудования, находящегося под напряжением.

Газовые огнетушители применяются для тушения жидких и твердых веществ, а также электроустановок, находящихся под напряжением.

В производственных помещениях ВЦ применяются главным образом углекислотные огнетушители, достоинством которых является высокая эффективность тушения пожара, сохранность электронного оборудования, диэлектрические свойства углекислого газа, что позволяет использовать эти огнетушители даже в том случае, когда не удастся обесточить электроустановку сразу.

4 Экономическая часть

4.1 Технико-экономические расчеты экономической целесообразности разработки программы

Задача состоит в разработке системы для автоматизации подготовки загрузки первичных данных в хранилище. Программа написана для сокращения затрат времени, что позволяет увеличить производительность труда. В данной программе реализован удобный пользовательский интерфейс, а также возможность просмотра сформированных скриптов перед их сохранением.

Для написания программы использовались языки программирования С# и Т-SQL.

Программа разработана для компьютеров со следующей конфигурацией:

- процессор с тактовой частотой 1,8 ГГц и выше;
- объем оперативного запоминающего устройства – 2 Гб и более;
- объем постоянного запоминающего устройства – 64 Гб и более;
- любая встроенная или дискретная видеокарта.

Для того, чтобы говорить об экономической эффективности разработанного проекта, надо рассчитать его себестоимость.

Себестоимость – это затраты предприятий, связанные с производством и реализацией продукции. Калькуляция себестоимости выпускаемой продукции включает в себя затраты на сырье и материалы, оплату труда, страховые взносы, амортизацию основных фондов и прочие расходы. Расчет себестоимости показан на формуле (4.1).

$$C = M + \text{КИ(ПФ)} + T_{p-z} + Z_{\text{осн}} + Z_{\text{доп}} + C_v + H_{\text{цех}} + H_{\text{зав}} + B_{\text{пр}}, \quad (4.1)$$

где C – себестоимость, руб.;

M – стоимость материалов, основы для полуфабрикатов, разделенных на вспомогательные (не входящие в состав готовой продукции) и основные, руб.;

КИ(ПФ) – стоимость комплектующих материалов, готовых элементов конечного продукта, используемых для сборки, ремонта и упаковки, руб.;

T_{p-z} – транспортно-заготовительные расходы, связанные с заготовкой, доставкой и хранением материальных ценностей (товаров, сырья, материалов, инструментов), руб.;

$Z_{осн}$ – заработная плата основная основных производственных рабочих, определяется в зависимости от трудоемкости, сложности и действующей формы оплаты труда, руб.;

$Z_{доп}$ – заработная плата дополнительная основных производственных рабочих, расходуется на оплату отпусков, отдельных видов премирования, пособия для уходящим в армию, выплат государственных и общественных обязанностей, руб.;

C_v – страховые взносы основных производственных рабочих, расходуется на пенсии и медицину, руб.;

$H_{цех}$ – накладные расходы цеховые, расходуются на амортизацию оборудования цеха, заработную плату всех работников цеха (исключая основных производственных рабочих), ЖКХ, текущий и капитальный ремонт помещений цеха, руб.;

$H_{зав}$ – накладные расходы завода, расходуются на амортизацию общезаводского оборудования, заработную плату всех работников завода (исключая работников цехов), ЖКХ, текущий и капитальный ремонт здания завода, аренду, отчисления вышестоящим организациям, руб.;

$V_{пр}$ – внепроизводственные расходы, расходуются рекламу, упаковку, командировки, презентации новых разработок, руб.

Ввиду специфики работы программиста и отсутствия материальной части формула приобретает вид (4.2):

$$C = Z_{осн} + Z_{доп} + C_v + H_{цех} + H_{зав} + V_{пр}, \quad (4.2)$$

где $Z_{осн}$ – заработная плата основная программистов, определяется в зависимости от трудоемкости задачи и оклада, руб.;

$Z_{доп}$ – заработная плата дополнительная программистов, расходуется на оплату отпусков, отдельных видов премирования, пособия для уходящим в армию, выплат государственных и общественных обязанностей, руб.;

C_v – страховые взносы программистов, расходуется на пенсии и медицину, руб.;

$H_{\text{цех}}$ – накладные расходы отдела, расходуются на амортизацию компьютеров и другого оборудования, заработную плату всех работников отдела (исключая программистов), ЖКХ, текущий и капитальный ремонт помещений отдела, руб.;

$H_{\text{зав}}$ – накладные расходы фирмы, расходуются на амортизацию общefирменного оборудования, заработную плату всех работников фирмы (исключая работников отделов), ЖКХ, текущий и капитальный ремонт здания фирмы, аренду, отчисления вышестоящим организациям, руб.;

$V_{\text{пр}}$ – внепроизводственные расходы, расходуются на антивирусные программы, оплату интернета, покупку нового оборудования, сопровождение программы во время ее выполнения, руб.

В связи с высокой стоимостью оборудования для программиста, необходимостью быстрой амортизации и большим объемом потребляемой электроэнергии формула приобретает вид (4.3):

$$C = Z_{\text{осн}} + Z_{\text{доп}} + C_{\text{в}} + A + \mathcal{E} + H_{\text{цех}} + H_{\text{зав}} + V_{\text{пр}}, \quad (4.3)$$

где A – сумма амортизации оборудования, руб.;

\mathcal{E} – плата за потребленную электроэнергию, руб.

4.2 Расчетная часть

4.2.1 Расчет трудоемкости разработанной программы

Для определения себестоимости решения задачи необходимо, прежде всего, найти трудоемкость решения задачи.

Трудоемкость – это сумма затрат труда (по времени), необходимых для изготовления единицы продукции, которая предназначена для решения разработки программы в соответствии с содержанием задания.

Трудоемкость рассматривается как сумма затрат времени на разных этапах решения задачи. Общая трудоемкость рассчитывается по формуле (4.4).

$$T_o = T_{и} + T_a + T_{бс} + T_{п} + T_{от} + T_{д} + T_{мр} + T_{эвм}, \quad (4.4)$$

где T_o – трудоемкость общая, час;

$T_{и}$ – затраты труда на изучение материала, описание задачи, час;

T_a – затраты труда на разработку алгоритмов решения задачи, час;

$T_{бс}$ – затраты труда на разработку блок-схем алгоритма программы, час;

$T_{п}$ – затраты труда на программирование, час;

$T_{от}$ – затраты труда на отладку программы, час;

$T_{д}$ – затраты на оформление документации, час;

$T_{мр}$ – затраты труда на машинно-ручные работы, час;

$T_{эвм}$ – время машинного счета на ЭВМ, час.

Слагаемые трудоемкости определяются через количество программных команд данной стадии разработки.

Затраты труда на изучение и описание задачи определяются по формуле (4.5).

$$T_{и} = \frac{Q}{B \times K_{кв}} \times \beta, \quad (4.5)$$

где Q – предполагаемое количество команд данной стадии разработки;

β – коэффициент, учитывающий качество описания задачи, равный 1,3 по данным предприятия;

B – скорость программиста, количество команд в час;

K_{KB} – коэффициент квалификации исполнителя, равный 0,8 по данным предприятия.

Расчет трудоемкости на этапе изучения и описания задачи

Число программных команд данной программы равно 1 018 командам. По формуле (4.5) трудоемкость изучения материала и описания задачи.

$$B = 80;$$

$$\beta = 1,3;$$

$$K_{KB} = 0,8.$$

$$T_{и} = \frac{1\,018}{80 \times 0,8} \times 1,3 = 20,68$$

Расчет затрат труда на этапе разработки алгоритма решения задачи

Величина T_a находится по формуле, идентичной $T_{и}$.

$$B = 20;$$

$$\beta = 1,3;$$

$$K_{KB} = 0,8.$$

$$T_a = \frac{1\,018}{20 \times 0,8} \times 1,3 = 82,71$$

Расчет затрат труда на этапе разработки схем алгоритма программы

Величина $T_{бс}$ находится по формуле, идентичной $T_{и}$.

$$B = 14;$$

$$\beta = 1,3;$$

$$K_{KB} = 0,8.$$

$$T_{бс} = \frac{1\,018}{14 \times 0,8} \times 1,3 = 118,16$$

Расчет затрат труда на этапе программирования

Величина T_{Π} находится по формуле, идентичной $T_{и}$.

$$B = 20;$$

$$\beta = 1,3;$$

$$K_{KB} = 0,8.$$

$$T_{\Pi} = \frac{1\ 018}{20 \times 0,8} \times 1,3 = 82,71$$

Расчет затрат труда на этапе отладки

Величина T_o находится по формуле, идентичной $T_{и}$.

$$B = 5;$$

$$\beta = 1,3;$$

$$K_{KB} = 0,8.$$

$$T_o = \frac{1\ 018}{5 \times 0,8} \times 1,3 = 330,85$$

Расчет затрат труда на этапе документации

Величина T_d находится по формуле, идентичной $T_{и}$.

$$B = 15;$$

$$\beta = 1,3;$$

$$K_{KB} = 0,8.$$

$$T_d = \frac{1\ 018}{15 \times 0,8} \times 1,3 = 110,28$$

Расчет затрат труда на этапе машинно-ручных операций

Трудоемкость на этапе машинно-ручных операций рассчитывается по формуле (4.6).

$$T_{м-р} = \frac{t}{3600}, \quad (4.6)$$

где t – время ввода информации, сек.

Время ввода информации находится по формуле (4.7).

$$t = \frac{C \times 1,5}{4}, \quad (4.7)$$

где C – объем программы в символах.

По формуле (4.7) необходимо найти время ввода информации при $C = 69\,489$.

$$t = \frac{69\,489 \times 1,5}{4} = 26\,058,38$$

Зная время ввода информации, можно найти трудоемкость на этапе машинно-ручных операций по формуле (4.6).

$$T_{\text{м-р}} = \frac{26\,058,38}{3600} = 7,24$$

Расчет затрат труда на этапе машинного времени

Время машинного счета на ЭВМ вычисляется по формуле (4.8).

$$T_{\text{ЭВМ}} = t_{\text{ВВ}} + t_{\text{ВЫВ}} + t_{\text{СЧ}}, \quad (4.8)$$

где $t_{\text{ВВ}}$ – время ввода, час;

$t_{\text{ВЫВ}}$ – время вывода, час;

$t_{\text{СЧ}}$ – время счета, час.

По статистическим данным $T_{\text{ЭВМ}} = 0,02$ часа.

Расчет общей трудоемкости

Зная все затраты труда на всех этапах решения задачи, можно вычислить общую трудоемкость решаемой задачи по формуле (4.4).

$$\begin{aligned} T_o &= 20,68 + 82,71 + 118,16 + 82,71 + 330,85 + 110,28 + 7,24 + 0,02 = \\ &= 752,66 \end{aligned}$$

Трудоемкость решения задачи на ЭВМ составляет 752,66 часа.

4.2.2 Расчет себестоимости разработанной программы

Для нахождения себестоимости разработки программы при решении задачи на ЭВМ необходимо использовать формулу (4.9).

$$C = Z_{\text{осн}} + Z_{\text{доп}} + C_{\text{в}} + A + Э + H_{\text{цех}} + H_{\text{зав}} + B_{\text{пр}} \quad (4.9)$$

Расчет основной заработной платы

Заработная плата – это часть национального дохода, переданного в личное распоряжение работника в соответствии с количеством и качеством затраченного им труда. Заработная плата состоит из основной, дополнительной заработной платы и страховых взносов.

Основная заработная плата зависит от степени квалификации работника, уровня сложности, количества и качества выполняемой работы, а также условий, в которых выполняется работа и определяется по формуле (4.10).

$$Z_{\text{осн}} = C_{\text{ч}} \times T_{\text{о}}, \quad (4.10)$$

где $C_{\text{ч}}$ – часовая тарифная ставка, руб.;

$T_{\text{о}}$ – общая трудоемкость решения задачи, час.

Часовая тарифная ставка определяется по формуле (4.11).

$$C_{\text{ч}} = \frac{\text{Оклад}}{22,8 \times 8}, \quad (4.11)$$

где Оклад – месячный штатный оклад программиста по данным предприятия, равен 50 000 рублей;

22,8 – среднее количество рабочих дней в месяц;

8 – количество рабочих часов в смену.

Найдем часовую тарифную ставку по формуле (4.11).

$$C_{\text{ч}} = \frac{50\,000}{22,8 \times 8} = 274,12$$

Для нахождения основной заработной платы необходимо использовать формулу (4.10).

$$З_{\text{осн}} = 274,12 \times 752,66 = 206\,319,16$$

Расчет дополнительной заработной платы

Далее, необходимо найти дополнительную заработную плату по формуле (4.12), которая равна 80% от основной заработной платы по данным предприятия.

$$З_{\text{доп}} = З_{\text{осн}} \times 80\%, \quad (4.12)$$

$$З_{\text{доп}} = 206\,319,16 \times 0,8 = 165\,055,39$$

Расчет страховых взносов

Зная размер основной и заработной платы, можно вычислить сумму страховых взносов, которая составляет 30,2% от размера фонда оплаты труда (ФОТ) и является обязательным государственным налогом. ФОТ рассчитывается по формуле (4.13).

$$\text{ФОТ} = З_{\text{осн}} + З_{\text{доп}}, \quad (4.13)$$

$$\text{ФОТ} = 206\,319,16 + 165\,055,39 = 371\,374,55$$

Зная ФОТ можно рассчитать страховые взносы по формуле (4.14).

$$С_{\text{в}} = \text{ФОТ} \times 30,2\%, \quad (4.14)$$

$$С_{\text{в}} = 371\,374,55 \times 0,302 = 112\,155,11 \text{ руб.}$$

Расчет амортизационных отчислений

Амортизация - это ежемесячное денежное отчисление для возмещения износа оборудования. Амортизация рассматриваемой программы вычисляется в три этапа. Сначала рассчитывается годовая амортизация по формуле (4.15).

$$A_{\text{год}} = \frac{C_{\text{перв}}}{4}, \quad (4.15)$$

где $C_{\text{перв}}$ – первоначальная себестоимость оборудования, использованного для разработки программы, по данным предприятия 80 000 руб;

4 – срок окупаемости оборудования, год.

$$A_{\text{год}} = \frac{80\,000}{4} = 20\,000 \text{ руб.}$$

Далее рассчитывается амортизация дневная по формуле (4.16).

$$A_{\text{дня}} = \frac{A_{\text{год}}}{247}, \quad (4.16)$$

где 247 – количество рабочих дней в 2022 году.

$$A_{\text{дня}} = \frac{20\,000}{247} = 80,97 \text{ руб.}$$

Чтобы найти амортизацию программы, необходимо вычислить амортизацию часовую, исходя из формулы (4.17).

$$A_{\text{час}} = \frac{A_{\text{дня}}}{8}, \quad (4.17)$$

где 8 – количество рабочих часов в смену.

$$A_{\text{час}} = \frac{80,97}{8} = 10,12 \text{ руб.}$$

Итого на каждый час работы программиста приходится 10,12 рубля амортизации. Вычислим амортизацию рассматриваемой программы по формуле (4.18).

$$A_{\text{прог}} = A_{\text{часа}} \times T_o \quad (4.18)$$

$$A_{\text{прог}} = 10,12 \times 752,66 = 7\,616,92 \text{ руб}$$

Расчет затрат на электроэнергию

Затраты на электроэнергию рассчитываются по формуле (4.19).

$$\mathcal{E} = \text{Тариф} \times T_o, \quad (4.19)$$

где Тариф – действующий тариф на электроэнергию в Москве, по данным предприятия 5,92 руб/кВт-час.

$$\mathcal{E} = 5,92 \times 752,66 = 4\,455,75 \text{ руб.}$$

Расчет накладных цеховых расходов

Накладные расходы отдела определяются в процентном отношении от основной зарплаты, составляют 200% по данным предприятия. В состав накладных расходов отдела включаются такие затраты как заработная плата аппарата управления отдела (начальника отдела, заместителя начальника, системного администратора), амортизационные отчисления на текущий и капитальный ремонт заданий и оборудования, сооружений, на охрану труда в данном отделе и на непроизводительные затраты. Рассчитывается по формуле (4.20).

$$H_{\text{цех}} = Z_{\text{осн}} \times 200\% \quad (4.20)$$

$$H_{\text{цех}} = 206\,319,16 \times 2,00 = 412\,638,32$$

Расчет себестоимости цеховой

Себестоимость отдела – это показатель затратности производства, его анализ позволяет выявить фактор отклонения от процентного объема расходов и сравнить

эффективность работы разных отделов. Рассчитывается по формуле (4.21).

$$C_{\text{цех}} = Z_{\text{осн}} + Z_{\text{доп}} + C_{\text{в}} + A + Э + H_{\text{цех}} , \quad (4.21)$$

$$\begin{aligned} C_{\text{цех}} &= 206\,319,16 + 165\,055,39 + 112\,155,11 + 7\,616,92 + 4\,455,75 + 412\,638,32 = \\ &= 908\,240,65 \end{aligned}$$

Расчет накладных заводских расходов

Накладные расходы фирмы определяются в процентном отношении от основной зарплаты и составляют 100% по данным предприятия. Накладные расходы фирмы – это расходы по управлению фирмой, содержание общеприемного персонала с отчислением на страховые взносы, расходы по командировкам, амортизационные отчисления общеприемного оборудования, на текущий и капитальный ремонт зданий, отчисление вышестоящим организациям. Рассчитываются по формуле (4.22).

$$H_{\text{зав}} = Z_{\text{осн}} \times 100\% , \quad (4.22)$$

$$H_{\text{зав}} = 206\,319,16 \times 1,00 = 206\,319,16$$

Расчет себестоимости производственной

Производственная себестоимость определяется путем суммирования общезаводских и целевых расходов с себестоимостью цеховой. Она включает производственные затраты всех отделов, занятых производством продукции или услуг, и расходы по общему управлению предприятием. Себестоимость производственная рассчитывается по формуле (4.23).

$$C_{\text{пр}} = C_{\text{цех}} + H_{\text{зав}} , \quad (4.23)$$

$$C_{\text{пр}} = 908\,240,65 + 206\,319,16 = 1\,114\,559,81$$

Расчет внепроизводственных расходов

Внепроизводственные расходы являются неотъемлемой частью затрат предприятия и могут включать в себя рекламу, сопровождение программы во время ее исполнения на предприятии, оплату интернета, первоначальную настройку оборудования и сетей, поддержание работоспособности системы ЭВМ. По данным предприятия они составляют 10% от стоимости заводской и рассчитываются по формуле (4.24).

$$B_{\text{пр}} = C_{\text{пр}} \times 10\%, \quad (4.24)$$

$$B_{\text{пр}} = 1\,114\,559,81 \times 0,10 = 111\,455,98$$

Расчет полной себестоимости решения задачи

Таким образом, зная все необходимые величины, можно вычислить полную себестоимость. Она включает затраты организации не только на выпуск продукции и организацию производственного процесса, но и на ее реализацию, иначе – на ее поставку на рынок конечного товара и услуг. Она учитывается при формировании цены реализации этой продукции и служит показателем для определения суммы прибыли, получаемой от ее продажи. Вычисляется по формуле (4.25).

$$C_{\text{п}} = C_{\text{пр}} + B_{\text{пр}}, \quad (4.25)$$

$$C_{\text{п}} = 1\,114\,559,81 + 111\,455,98 = 1\,226\,015,79$$

Себестоимость решения задачи на ЭВМ равна 1 226 015,79 руб.

4.2.3 Анализ возможных путей снижения себестоимости

Калькуляция – это исчисление себестоимости продукции по основным затратам, которые входят в состав себестоимости изделия.

Себестоимость оценивает общую сумму затрат предприятия не только на производство, но и на реализацию продукции. Она учитывается при формировании цены реализации этой продукции и служит показателем для определения суммы прибыли, получаемой от ее продажи.

Данная программа является трудоемкой, так как затраты, связанные с составлением и реализацией программы высоки.

Прямые расходы составляют 39,5% (16,8% + 13,5% + 9,2%). Снижение себестоимости возможно путем:

- повышение квалификации разработчика программы;
- использование современного оборудования;
- поддержание оборудования в хорошем состоянии;
- использование более подходящего языка программирования или более эффективное использование его возможностей.

По косвенным расходам наибольший удельный вес себестоимости полной занимают накладные цеховые = 33,7%. Снизить себестоимость в данном случае можно при помощи:

- внедрения современного энергосберегающего оборудования;
- пересмотра штатного расписания.

Внепроизводственные расходы при разработке данной программы составляют 9% (по данным предприятия) от производственной себестоимости.

4.3 Графическая часть

В таблице 4.1 представлена производительность труда программиста.

Таблица 4.1 – Производительность труда программиста

Характер работы	Производительность, Количество команд/час
Изучение описания задачи	80
Разработка алгоритмов решения	20
Разработка схем алгоритмов	14
Программирование по готовой схеме алгоритма с использованием алгоритмического языка	20
Автономная отладка программы	5
Оформление документации	15

В таблице 4.2 представлена трудоемкость при решении задачи на ЭВМ.

Таблица 4.2 – Трудоемкость при решении задачи на ЭВМ

Наименование затрат	Единица измерения	Трудоемкость в часах
Трудоемкость решения задачи на ЭВМ	час	752,68
В том числе:		
Время изучения описания задачи	час	20,68
Время на разработку алгоритма задачи	час	82,71
Время на разработку схемы алгоритма	час	118,16
Время на программирование	час	82,71
Время на отладку программы	час	330,85
Время на оформление документации	час	110,28
Время на машинно-ручные работы	час	7,24
Время машинного счета	час	0,02

В таблице 4.3 представлена калькуляция разработки программы.

Таблица 4.3 – Таблица калькуляции

Наименование статьи калькуляции	Сумма в руб.	Процент к итогу	Процент к основной зарплате
Прямые затраты:			
Основная заработная плата	206 319,16	16,8	100,0
Дополнительная заработная плата	165 055,39	13,5	80,0
Страховые взносы	112 155,11	9,2	54,4
Амортизация оборудования	7 616,92	0,6	3,7
Затраты на электроэнергию	4 455,75	0,4	2,2
Косвенные затраты:			
Накладные цеховые расходы	412 638,32	33,7	200,0
Себестоимость цеховая	908 240,65	-	-
Накладные заводские расходы	206 319,16	16,8	100,0
Себестоимость производственная	1 114 559,81	-	-
Внепроизводственные расходы	111 455,98	9,1	54,0
Себестоимость полная	1 226 015,79	100,0	594,2

В таблице 4.4 представлены технико-экономические показатели.

Таблица 4.4 – Техничко-экономические показатели

Техничко-экономический показатель	Единица измерения	Формула	Результат
Трудоёмкость решения задачи на ЭВМ	час	$T_o = T_{и} + T_{а} + T_{бс} + T_{п} + T_{от} + T_{д} + T_{мр} + T_{эвм}$	752,68
Себестоимость решения задачи на ЭВМ	руб.	$C = Z_{осн} + Z_{доп} + C_{в} + A + Э + H_{цех} + H_{зав} + B_{пр}$	1 226 015,79

ЗАКЛЮЧЕНИЕ

В ходе выполнения дипломного проекта была разработана программная система для автоматизации подготовки загрузки первичных данных в хранилище. Использование программного продукта созданного в рамках дипломного проекта позволит сократить время, затрачиваемое на написание скриптов необходимых для создания объектов на слоях корпоративного хранилища данных Промсвязьбанка при загрузке в него данных. Программный продукт прост в использовании и выполняет все необходимые функции.

При разработке программы использовался высокоуровневый язык программирования C# с использованием платформы WPF, который позволяет создавать клиентские приложения для настольных систем Windows с привлекательным пользовательским интерфейсом.

В ходе выполнения данного дипломного проекта были отработаны навыки моделирования программных систем, процессов отладки и тестирования, составления документации. Также был проведен экономический анализ данного дипломного проекта: подсчитана производительность труда программиста, составляющая 752,66 часов, и себестоимость решения задачи на ЭВМ, составляющая 1 226 015,79 руб.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

- 1 Бондарев А.Г. Microsoft SQL Server 2014. «БХВ-Петербург», 2015
- 2 Борисов Е. Ф., Волков Ф. М. Основы экономической теории. «Высшая школа», 2016
- 3 Грофф Д., Вайнберг П., Оппель Э. SQL. Полное руководство. «Диалектика», 2019
- 4 Макконелл С. Совершенный код. «БХВ-Петербург», 2017
- 5 Меняев М., Цифровая экономика предприятия, «ИНФРА-М», 2020
- 6 Молинаро. Э. SQL. Сборник рецептов. «БХВ-Петербург», 2021
- 7 Нагел К., Ивьен Б. C# 5.0 и платформа .NET 4.5 для профессионалов. «Вильямс», 2014
- 8 Натан А. WPF4. Подробное руководство. «Символ-Плюс», 2011
- 9 Райзберг Б. А. Основы экономики и предпринимательства. «Просвещение», 2016
- 10 Редмонд Э., Джим Р. Семь баз данных за семь недель. Введение в современные базы данных и идеологию NoSQL. «ДМК Пресс», 2018
- 11 Роберт М. Идеальный программист. Как стать профессионалом разработки ПО. «Питер», 2022
- 12 Роберт М. Чистый код: создание, анализ и рефакторинг. «Питер», 2022
- 13 Скит Д. C# для профессионалов. Тонкости программирования. «Вильямс», 2019
- 14 Стивен С. Скиена. Алгоритмы. Руководство по разработке. «БХВ-Петербург», 2018
- 15 Троелсен Э., Джепикс Ф. Язык программирования C# 7 и платформы .NET и .NET Core. «Диалектика», 2018
- 16 Фримен А. ASP.NET Core MVC 2 с примерами на C# для профессионалов. Руководство. «Диалектика», 2019
- 17 Документация по Visual Studio [Электронный ресурс]. – URL: <https://docs.microsoft.com/ru-ru/visualstudio/?view=vs-2019> (дата обращения: 12.04.2022)

- 18 Язык программирования C# и платформа .NET [Электронный ресурс]. – URL: <https://metanit.com/sharp/> (дата обращения: 18.04.2022)
- 19 Реляционные базы данных и язык SQL [Электронный ресурс]. – URL: <https://metanit.com/sql/> (дата обращения: 22.04.2022)
- 20 Документация по Microsoft SQL [Электронный ресурс]. – URL: <https://docs.microsoft.com/ru-ru/sql/?view=sql-server-ver15> (дата обращения: 25.04.2022)

ПРИЛОЖЕНИЕ А

(обязательное)

Листинги программы

A.1 Листинг класса MainWindow

```
/* Дипломный проект
* Тема: "Разработка системы для автоматизации подготовки загрузки первичных
* данных в хранилище".
* Название программы: OdsWizard.exe.
* Разработал: Никифоров Макар Александрович, группа ТИП-81.
* Дата и номер версии: 15.05.2022 v1.3.
* Язык программирования: C#.
* Среда разработки: Visual Studio 2019.
*****
* Задание:
*     Разработать программную систему для автоматизирования процесса
* подготовки загрузки первичных данных в хранилище данных, путем
* автоматического формирования sql скриптов объектов, необходимых при
* импорте данных. Реализовать понятный пользовательский интерфейс.
* Ожидаемые входные данные:
*     1) наименование системы источника;
*     2) метаданные таблицы источника;
*     3) путь сохранения файлов.
*
* MainWindow – класс главного окна приложения.
* Используемые переменные:
*     currentPageChanged - флаг, отвечает за состояние текущей страницы;
*     currentSystem - объект класса TractSystem. Хранит текущую систему,
* отображаемую в интерфейсе;
*     pageChanged - флаг, отвечает за состояние страницы Wizard.
*/
using System;
using System.Collections.Generic;
using System.Linq;
```



```

using System.Windows;
using System.Windows.Controls;
using System.Windows.Input;
using System.Data.SqlClient;
using System.Configuration;
using System.Data;
using Xceed.Wpf.Toolkit;
using System.Collections.ObjectModel;
using WinForms = System.Windows.Forms;
using System.IO;
using Oracle.DataAccess.Client;

namespace OdsWizard
{
    public partial class MainWindow : Window
    {
        private Boolean currentPageChanged = false;
        private Boolean canNext = false;
        private TractSystem currentSystem;
        private bool pageChanged;
        public MainWindow()
        {
            try
            {
                InitializeComponent();
                pageChanged = false;
                MainWizard.CanSelectNextPage = false;
            }
            catch (Exception ex)
            {
                System.Windows.MessageBox.Show(ex.InnerException.Message);
            }
        }
    }
}

```

```

TractData.TractSystems = new List<TractSystem>();

string connString =
ConfigurationManager.AppSettings.Get("configConnectionStr");

try
{
    using (SqlConnection conn = new SqlConnection(connString))
    {

// Получение всех доступных систем из БД

        string query = @"
select
    id_layer as TR_SYSTEM_ID,
    name as TR_SYSTEM_NAME,
    name_s,
    GET_METADATA_SCRIPT,
    CONN_STRING,
    TR_SYSTEM_TYPE,
    isnull(IS_EXTERNAL, 0) IS_EXTERNAL,
    READONLY_DT_COLS,
    HIDDEN_DT_COLS,
    isnull(ADD_ROW_CFG, '0,0,0') ADD_ROW_CFG,
    isnull(DELETE_ROW_CFG, '0,0,0') DELETE_ROW_CFG,
    isnull(SHOW_INC, 0) SHOW_INC
from
    dbo.DIC_LAYER
order by IS_EXTERNAL desc, id_layer asc;";

        SqlCommand cmd = new SqlCommand(query, conn);
        conn.Open();
        SqlDataReader dr = cmd.ExecuteReader();
        if (dr.HasRows)
        {
            startPointCB.Items.Clear();

```

```

startPointCB.SelectedValuePath = "Key";
startPointCB.DisplayMemberPath = "Value";
while (dr.Read())

// Добавление полученных данных в
// переменную списка таблиц

{
    TractSystem ts = new TractSystem()
    {
        TractSystemId = Convert.ToInt32(dr["TR_SYSTEM_ID"]),
        TractSystemName = dr["TR_SYSTEM_NAME"].ToString(),
        ScriptGetTableMetadata = dr["GET_METADATA_SCRIPT"].ToString(),
        ConnStr = (Boolean)dr["IS_EXTERNAL"] ? dr["CONN_STRING"].ToString() :
Tools.ReplaceServer(dr["CONN_STRING"].ToString()),
        TractSystemType = dr["TR_SYSTEM_TYPE"].ToString(), IsExternal =
(Boolean)dr["IS_EXTERNAL"], TractSystemDestinations = new List<TractSystem>(),
        RoColumns = Array.ConvertAll(dr["READONLY_DT_COLS"].ToString().
Split(",", StringSplitOptions.RemoveEmptyEntries), s =>
int.Parse(s)).ToList<int>(),
        HiddenColumns = Array.ConvertAll(dr["HIDDEN_DT_COLS"].ToString().
Split(",", StringSplitOptions.RemoveEmptyEntries), s =>
int.Parse(s)).ToList<int>(),
        IsShowInc = (bool)dr["SHOW_INC"],
        CanAddRowField =
Convert.ToBoolean(Convert.ToInt32(dr["ADD_ROW_CFG"].ToString().
Split(",", StringSplitOptions.RemoveEmptyEntries)[0])),
        CanAddRowComment = Convert.ToBoolean(
Convert.ToInt32(dr["ADD_ROW_CFG"].ToString().Split(",", StringSplitOptions.
RemoveEmptyEntries)[1])),
        CanAddRowArtifact = Convert.ToBoolean(Convert.ToInt32
(dr["ADD_ROW_CFG"].ToString().Split(",", StringSplitOptions.
RemoveEmptyEntries)[2])),

```

```

        CanDeleteRowField = Convert.ToBoolean(
Convert.ToInt32(dr["DELETE_ROW_CFG"].ToString().Split(",".ToCharArray(),
StringSplitOptions.RemoveEmptyEntries)[0])),
        CanDeleteRowComment = Convert.ToBoolean(
Convert.ToInt32(dr["DELETE_ROW_CFG"].ToString().Split(",".ToCharArray(),
StringSplitOptions.RemoveEmptyEntries)[1])),
        CanDeleteRowArtifact = Convert.ToBoolean(
Convert.ToInt32(dr["DELETE_ROW_CFG"].ToString().Split(",".ToCharArray(),
StringSplitOptions.RemoveEmptyEntries)[2]))
    };
    TractData.TractSystems.Add(ts);
    if (ts.TractSystemName != "OLAP+KIHFD")
    {
        startPointCB.Items.Add(new KeyValuePair<int,
String>(ts.TractSystemId, ts.NameWithServer));
    }
}
dr.Close();

// Заполнение поля система назначения
// каждой из систем
foreach (TractSystem ts in TractData.TractSystems)
{
    query = "select id_layer_dst from [dbo].[DIC_TRANSITION_LAYER]
where id_layer_src = " + ts.TractSystemId.ToString();
    SqlCommand subCmd = new SqlCommand(query, conn);
    SqlDataReader sdr = subCmd.ExecuteReader();
    while (sdr.Read())
    {
        TractSystem dst = TractData.TractSystems.FirstOrDefault(t =>
t.TractSystemId == Convert.ToInt32(sdr["id_layer_dst"]));
        if (dst != null)
            ts.TractSystemDestinations.Add(dst);
    }
}

```

```

        }
        sdr.Close();
    }
}
else
{
    System.Windows.MessageBox.Show(@"В данный момент таблица с
системами пуста!");
}
TractData.TypeTrans = new List<KeyValuePair<string, string>>();
cmd = new SqlCommand("select type_src, type_dst from
DIC_TRANSITION_TYPE", conn);
SqlDataReader typeTransDr = cmd.ExecuteReader();
while (typeTransDr.Read())
{
    TractData.TypeTrans.Add(new KeyValuePair<string,
string>(typeTransDr["type_src"].ToString().ToUpper(),
typeTransDr["type_dst"].ToString().ToUpper()));
}
typeTransDr.Close();
}
}
catch (Exception Ex)
{
    System.Windows.MessageBox.Show(@"Нет связи с сервером проверьте
файл App.Config!");
    System.Windows.MessageBox.Show(Ex.Message);
}
}

```

```

private void startPreviewBtn_Click(object sender, RoutedEventArgs e)
{
    // Формирование тракта

    if (startPointCB.SelectedItem == null ||
String.IsNullOrEmpty(tableNameTB.Text))
    {
        System.Windows.MessageBox.Show("Необходимо выбрать систему и
ввести наименование таблицы");
        return;
    }

    var rslt = MessageBoxResult.Cancel;

    if (TractData.TractTablesBuffer != null && TractData.TractTablesBuffer.Count()
> 1 && currentPageChanged)
        rslt = System.Windows.MessageBox.Show("Уверены, что хотите очистить
весь тракт?", "Внимание!!!", MessageBoxButton.OKCancel);

    if (TractData.TractTablesBuffer == null || TractData.TractTablesBuffer.Count()
== 0 || rslt == MessageBoxResult.OK)
    {
        if (InitTract())
        {
            TractTable currentTable = TractData.TractTablesBuffer.First(t =>
t.WzrdPage == IntroPage);
            drawPage(IntroPage, currentTable);
            MainWizard.CanSelectNextPage = true;
        }
    }
}

private void Wizard_Next(object sender,
Xceed.Wpf.Toolkit.Core.CancelRoutedEventArgs e)
{

```

```

var currentPage = ((Wizard)sender).CurrentPage;

TractTable tt = TractData.TractTablesBuffer.First(t => t.Layer ==
currentSystem.TractSystemName);

// Перезаполнение значения TractTable по
// странице

String prefix = currentPage.Name.Replace("Page", "");
ComboBox incCB = (ComboBox)currentPage.FindName(prefix + "IncCB");
tt.IsIncrement = incCB.Text == "Increment";

if (currentPage == OdsRawPage || currentPage == CalcfdPage)
{
    ComboBox syncCB = (ComboBox)currentPage.FindName(prefix +
"SyncCB");
    tt.SyncType = syncCB.Text;
}

// Проверка корректности sql скриптов

if (currentPage == IntroPage)
{
    TractData.StartLayerName = currentSystem.TractSystemName;
}
else
{
    if (currentPage != IntroPage)
    {
        foreach (TractLayerArtifact art in tt.Artifacts)
        {
            using (SqlConnection conn = new
SqlConnection(currentSystem.ConnStr))
            {
                var sqlArtText = art.SqlText.Replace("\nGO", "\n;");
                if (art.Type == "PROCEDURE" || art.Type == "CREATE SCRIPT" ||
art.Type == "TRIGGER")

```



```

String tableName = tableNameTB.Text.Contains(".") ?
tableNameTB.Text.Split('.')[1].Replace("[", "").Replace("]", "").Replace("\\", "") :
tableNameTB.Text;

TractTable tt = new TractTable()
{
    TSchema = tableSchema,
    TName = tableName,
    TExtProps = new ObservableCollection<TractLayerExtProp>(),
    TColumns = new ObservableCollection<TractTableColumn>(),
    Artifacts = new ObservableCollection<TractLayerArtifact>(),
    IsExternal = currentSystem.IsExternal,
    WzrdPage = IntroPage,
    Layer = currentSystem.TractSystemName
};

initTract = true;

// Инициализация тракта в
// зависимости от типа системы

if (currentSystem.TractSystemType != "Excel")
{
    if (currentSystem.TractSystemType == "mssql")
    {
        using (SqlConnection conn = new SqlConnection(currentSystem.ConnStr))
        {
            String query =
currentSystem.ScriptGetTableMetadata.Replace("#schema#",
tableSchema).Replace("#tname#", tableName);

            SqlCommand cmd = new SqlCommand(query, conn);
            conn.Open();
            SqlDataReader dr = cmd.ExecuteReader();
            if (dr.HasRows)
            {

```

```

while (dr.Read())
{
    if (String.IsNullOrEmpty(dr["tp"].ToString()))
    {
        tt.TExtProps.Add(new TractLayerExtProp() { Name =
dr["c_name"].ToString(), Value = dr["value"].ToString() });
    }
    else
    {
        tt.TColumns.Add(new TractTableColumn()
        {
            CName = dr["c_name"].ToString(),
            CType = dr["tp"].ToString(),
            CTypeNew = dr["type_new"].ToString(),
            PrimaryKeyNumber = System.DBNull.Value == dr["pk"] ? null
: (Int32?)dr["pk"],
            CComment = dr["value"].ToString()
        });
    }
}
else
{
    System.Windows.MessageBox.Show("Запрос не вернул данных");
    initTract = false;
}
}
else if (currentSystem.TractSystemType == "oracle")
{
    String oraConnStr = currentSystem.ConnStr;
    foreach (string key in ConfigurationManager.AppSettings)

```

```

        {
            oraConnStr = oraConnStr.Replace("#" + key + "#",
ConfigurationManager.AppSettings[key]);
        }
        using (OracleConnection conn = new OracleConnection(oraConnStr))
        {
            String query =
currentSystem.ScriptGetTableMetadata.Replace("#schema#",
tableSchema.ToUpper()).Replace("#tname#", tableName.ToUpper());
            OracleCommand cmd = new OracleCommand(query, conn);
            conn.Open();

            OracleDataReader ord = cmd.ExecuteReader();
            if (ord.HasRows)
            {
                while (ord.Read())
                {
                    if (String.IsNullOrEmpty(ord["tp"].ToString()))
                    {
                        tt.TExtProps.Add(new TractLayerExtProp() { Name =
ord["c_name"].ToString(), Value = ord["value"].ToString() });
                    }
                    else
                    {
                        tt.TColumns.Add(new TractTableColumn()
                        {
                            CName = ord["c_name"].ToString(),
                            CType = ord["tp"].ToString(),
                            CTypeNew = ord["type_new"].ToString(),
                            PrimaryKeyNumber = System.DBNull.Value == ord["pk"] ?
null : (Int32?)(Decimal.ToInt32((decimal)ord["pk"])),
                            CComment = ord["value"].ToString()
                        });
                    }
                }
            }
        }
    }
}

```

```

        });
    }
}
}
else
{
    System.Windows.MessageBox.Show("Запрос не вернул данных");
    initTract = false;
}
}
}
}
if (initTract) TractData.TractTablesBuffer.Add(tt);
return initTract;
}

```

```

private void Wizard_Previous(object sender,
Xceed.Wpf.Toolkit.Core.CancelRoutedEventArgs e)
{
    currentPageChanged = false;
}

```

```

private void IntroPage_Enter(object sender, RoutedEventArgs e)
{
    currentPageChanged = true;
}

```

```

private void startPointCB_SelectionChanged(object sender, SelectionChangedEventArgs
e)
{
    pageChanged = true;
    currentPageChanged = true;
}

```

```
currentSystem = TractData.TractSystems.First(t => t.TractSystemId ==  
((KeyValuePair<int, String>)startPointCB.SelectedItem).Key);
```

// Изменение системы приемника

```
IntroPage_destinationComboBox.Items.Clear();  
IntroPage_destinationComboBox.SelectedValuePath = "Key";  
IntroPage_destinationComboBox.DisplayMemberPath = "Value";  
foreach (TractSystem ts in currentSystem.TractSystemDestinations)  
{  
    IntroPage_destinationComboBox.Items.Add(new KeyValuePair<int,  
String>(ts.TractSystemId, ts.NameWithServer));  
}  
IntroPage_destinationComboBox.SelectedIndex = 0;
```

// Выбор следующей страницы в

// зависимости от системы приемника

```
switch (currentSystem.TractSystemName)  
{  
    case "ODS.STG":  
        IntroPage.NextPage = OdsRawPage;  
        OdsRawPage.PreviousPage = IntroPage;  
        break;  
    case "ODS.RAW":  
        IntroPage.NextPage = OdsExPage;  
        OdsExPage.PreviousPage = IntroPage;  
        break;  
    case "ODS.EX":  
        IntroPage.NextPage = StagefdPage;  
        StagefdPage.PreviousPage = IntroPage;  
        break;  
    case "STAGEFD":  
        IntroPage.NextPage = CalcfdPage;
```

```

        CalcfdPage.PreviousPage = IntroPage;
        break;
    case "CALCFD":
        IntroPage.NextPage = OlapPage;
        OlapPage.PreviousPage = IntroPage;
        break;
    default:
        IntroPage.NextPage = OdsStgPage;
        break;
};
if (currentSystem.TractSystemType == "Excel")
{
    IntroFieldsDG.IsReadOnly = false;
    IntroFieldsDG.CanUserAddRows = true;
    startPreviewBtn.Content = "Create";
    IntroCommentLB.CanUserAddRows = true;
    IntroPage.Description =
        "Выбор стартовой точки и тестового сервера" + Environment.NewLine +
        "Опишите сущность, как она есть в источнике";
}
else
{
    IntroFieldsDG.IsReadOnly = true;
    IntroFieldsDG.CanUserAddRows = false;
    startPreviewBtn.Content = "Preview";
    IntroCommentLB.CanUserAddRows = true;
    IntroPage.Description =
        "Выбор стартовой точки и тестового сервера" + Environment.NewLine +
        "Описание сущности в источнике";
}
if (currentSystem.IsExternal)
    IntroIncCB.Visibility = Visibility.Collapsed;

```

```

else
    IntroIncCB.Visibility = currentSystem.IsShowInc ? Visibility.Visible :
Visibility.Collapsed;
    IntroNextLbl.Visibility = IntroIncCB.Visibility;
    anyChangeOnPage();
}

private void anyChangeOnPage()
{
    PageProps pp = (PageProps)this.Resources["pageProperties"];
    if (this.MainWizard.CurrentPage == IntroPage)
    {
        if (IntroFieldsDG.Items.Count > 0 &&
IntroPage_destinationComboBox.SelectedItem != null)
        {
            pp.CanNext = true;
        }
    }
}

private void drawPage(WizardPage wp, TractTable currentTable)
{
    String prefix = wp.Name.Replace("Page", "");
    // Отрисовка интерфейса страницы
    DataGrid dg = (DataGrid)wp.FindName(prefix + "FieldsDG");
    dg.ItemsSource = currentTable.TColumns;
    dg.Columns[0].Header = "Primary Key Number";
    dg.Columns[1].Header = "Column Name";
    dg.Columns[2].Header = "Column Type";
    dg.Columns[3].Header = "New Column Name";
    dg.Columns[4].Header = "New Column Type";
    dg.Columns[5].Header = "Ext Properties Comment";
}

```

```

dg.CellEditEnding += Dg_CellEditEnding;

DataGrid commentLB = (DataGrid)wp.FindName(prefix + "CommentLB");
commentLB.ItemsSource = currentTable.TextProps;
commentLB.CellEditEnding += Dg_CellEditEnding;

DataGrid artifactLB = prefix == "Intro" ? new DataGrid() :
(DataGrid)wp.FindName(prefix + "ArtifactLB");
artifactLB.ItemsSource = currentTable.Artifacts;
commentLB.CellEditEnding += Dg_CellEditEnding;
if (prefix == "Intro")
{
    // Отрисовка страницы IntroPage

    dg.Columns[3].Visibility = Visibility.Hidden;
    dg.Columns[4].Visibility = Visibility.Hidden;

    if (currentSystem.TractSystemType != "Excel")
    {
        commentLB.CanUserAddRows = false;
        commentLB.CanUserDeleteRows = false;

        dg.CanUserAddRows = false;
        dg.CanUserDeleteRows = false;

        dg.IsReadOnly = false;
        for (int i = 0; i < 6; i++)
        {
            dg.Columns[i].IsReadOnly = i == 5 || i == 0 ? false : true;
        }
    }
}

```



```

else
{
    commentLB.CanUserAddRows = true;
    commentLB.CanUserDeleteRows = true;
    dg.IsReadOnly = false;

    dg.CanUserAddRows = true;
    dg.CanUserDeleteRows = true;
    dg.IsReadOnly = false;
}
}
else
{
    // Отрисовка страниц слоев хранилища
    wp.Description = $"Объекты для создания в слое
{currentSystem.NameWithServer}{Environment.NewLine}Основной объект
{currentTable.TSchema}.{currentTable.TName}";
    foreach (int cIndex in currentSystem.RoColumns)
    {
        dg.Columns[cIndex].IsReadOnly = true;
    }

    foreach (int cIndex in currentSystem.HiddenColumns)
    {
        dg.Columns[cIndex].Visibility = Visibility.Hidden;
    }
    dg.Columns[3].Visibility = Visibility.Hidden;
    dg.Columns[4].Visibility = Visibility.Hidden;

    Style rowStyle = new Style(typeof(DataGridRow));
    rowStyle.Setters.Add(new
EventSetter(DataGridRow.MouseDoubleClickEvent,

```

```

        new MouseButtonEventHandler(Row_DoubleClick)));
artifactLB.RowStyle = rowStyle;

if (currentSystem.TractSystemName != "OLAP+KIHFD")
{
    if (currentSystem.IsShowInc)
        ((ComboBox)wp.FindName(prefix + "IncCB")).Visibility =
Visibility.Visible;
    else
        ((ComboBox)wp.FindName(prefix + "IncCB")).Visibility =
Visibility.Collapsed;
}
}
if (currentSystem.TractSystemName != "OLAP+KIHFD")
{
    // Отрисовка страницы на последнем слое
    // хранлища
    Label lb = (Label)wp.FindName(prefix + "NextLbl");
    if (currentSystem.IsShowInc || currentSystem.TractSystemName ==
"ODS.RAW" || currentSystem.TractSystemName == "CALCFD")
        lb.Visibility = Visibility.Visible;
    else
        lb.Visibility = Visibility.Collapsed;
}
}

private void Dg_CellEditEnding(object sender, DataGridCellEditEndingEventArgs e)
{
    pageChanged = true;
}

```

```

private void Row_DoubleClick(object sender, MouseButtonEventArgs e)
{
    DataGridView row = sender as DataGridView;
    TractTable currentTable = TractData.TractTablesBuffer.First(t => t.WzrdPage ==
MainWizard.CurrentPage);
    ArtifactWindow aw = new ArtifactWindow(currentTable, row.GetIndex());
    aw.ShowDialog();
}

private void generateTractTable(TractTable previousTable, WizardPage pg)
{
    // Объявление и инициализация
    // переменных хранящих метаданные
    // начальной таблицы

    DataTable tableDT = new DataTable();
    tableDT.Columns.Add("id_tbl", typeof(int));
    tableDT.Columns.Add("schema_name_src", typeof(string));
    tableDT.Columns.Add("table_name_src", typeof(string));
    tableDT.Columns.Add("is_increment", typeof(int));
    tableDT.Columns.Add("schema_name_dst", typeof(string));
    tableDT.Columns.Add("table_name_dst", typeof(string));

    tableDT.Rows.Add(new object[] { 1, previousTable.TSchema,
RenameExtSysName(previousTable.TName), previousTable.IsIncrement ? 1 : 0 });

    DataTable columnDT = new DataTable();
    columnDT.Columns.Add("id_table", typeof(int));
    columnDT.Columns.Add("clmn_src", typeof(string));
    columnDT.Columns.Add("ordr", typeof(int));
    columnDT.Columns.Add("typ_src", typeof(string));
    columnDT.Columns.Add("pk", typeof(int));
    columnDT.Columns.Add("clmn_dst", typeof(string));

```

```

columnDT.Columns.Add("typ_dst", typeof(string));

DataTable descrDT = new DataTable();
descrDT.Columns.Add("id_table", typeof(int));
descrDT.Columns.Add("id_column", typeof(int));
descrDT.Columns.Add("ep_name", typeof(string));
descrDT.Columns.Add("ep_value_src", typeof(string));
descrDT.Columns.Add("ep_value_dst", typeof(string));

int i = 1;
foreach (var clmn in previousTable.TColumns)
{
    columnDT.Rows.Add(new object[] { 1, RenameExtSysName(clmn.CName), i,
RenameType(clmn.CType), clmn.PrimaryKeyNumber });
    descrDT.Rows.Add(new object[] { 1, i, "MS_Description", clmn.CComment
});

    i += 1;
}

i = 1;
foreach (var exprop in TractData.TractTablesBuffer[0].TExtProps)
{
    descrDT.Rows.Add(new object[] { 1, 0, exprop.Name, exprop.Value });
    i += 1;
}

// Подключение к БД, вызов хранимой
// процедуры для формирования тракта для
// следующего слоя хранилища

using (SqlConnection conn = new
SqlConnection(ConfigurationManager.AppSettings.Get("configConnectionStr")))
{

```

```

SqlCommand cmd = new SqlCommand("dbo.create_layer_mdata
@src_table_descr, @src_column_descr, @ep, @layer_name_src, @layer_name_dst,
@ods_ex , @sync_type", conn);

SqlParameter tblParam = cmd.Parameters.AddWithValue("@src_table_descr",
tableDT);

tblParam.SqlDbType = SqlDbType.Structured;
tblParam.TypeName = "dbo.list_table";

SqlParameter columnParam =
cmd.Parameters.AddWithValue("@src_column_descr", columnDT);

columnParam.SqlDbType = SqlDbType.Structured;
columnParam.TypeName = "dbo.list_column";

SqlParameter propsParam = cmd.Parameters.AddWithValue("@ep", descrDT);
propsParam.SqlDbType = SqlDbType.Structured;
propsParam.TypeName = "dbo.ext_props";

SqlParameter srcParam = cmd.Parameters.AddWithValue("@layer_name_src",
previousTable.Layer);

srcParam.SqlDbType = SqlDbType.VarChar;

SqlParameter dstParam = new SqlParameter();

dstParam = cmd.Parameters.AddWithValue("@layer_name_dst",
currentSystem.TractSystemName);

dstParam.SqlDbType = SqlDbType.VarChar;
if (previousTable.SyncType != null)
{
    SqlParameter odsExParam = cmd.Parameters.AddWithValue("@ods_ex",
previousTable.SyncType);

    dstParam.SqlDbType = SqlDbType.VarChar;

    SqlParameter syncTypeParam =
cmd.Parameters.AddWithValue("@sync_type", previousTable.SyncType);

    dstParam.SqlDbType = SqlDbType.VarChar;
}

```

```

else
{
    SqlParameter odsExParam = cmd.Parameters.AddWithValue("@ods_ex",
""");

    dstParam.SqlDbType = SqlDbType.VarChar;
    SqlParameter syncTypeParam =
cmd.Parameters.AddWithValue("@sync_type", "");
    dstParam.SqlDbType = SqlDbType.VarChar;
}

conn.Open();
SqlDataReader dr = cmd.ExecuteReader();

// Парсинг полученных данных

if (dr.HasRows)
{
    TractTable tt = new TractTable()
    {
        TExtProps = new ObservableCollection<TractLayerExtProp>(),
        TColumns = new ObservableCollection<TractTableColumn>(),
        Artifacts = new ObservableCollection<TractLayerArtifact>(),
        IsExternal = false,
        WzrdPage = pg,
        Layer = currentSystem.TractSystemName
    };

    var isCreate = true;
    while (dr.Read())
    {
        if (isCreate)
        {
            tt.TSchema = dr["SCHEMA_DST"].ToString();
            tt.TName = dr["TABLE_DST"].ToString();

```

```

        isCreate = false;
    }
    if (String.IsNullOrEmpty(dr["TYPE"].ToString()))
    {
        tt.TExtProps.Add(new TractLayerExtProp() { Name =
dr["COLUMN_NAME"].ToString(), Value = dr["VALUE"].ToString() });
    }
    else if (String.IsNullOrEmpty(dr["COLUMN_ID"].ToString()))
    {
        tt.Artifacts.Add(new TractLayerArtifact()
        {
            Name = dr["COLUMN_NAME"].ToString(),
            Type = dr["TYPE"].ToString(),
            SqlText = dr["VALUE"].ToString(),
            IsHandChanged = false
        });
    }
    else
    {
        tt.TColumns.Add(new TractTableColumn()
        {
            CName = dr["COLUMN_NAME"].ToString(),
            CType = dr["TYPE"].ToString(),
            PrimaryKeyNumber = System.DBNull.Value == dr["PK"] ? null :
(Int32?)dr["PK"],
            CComment = dr["VALUE"].ToString()
        });
    }
}
if (TractData.TractTablesBuffer.Count(t => t.Layer ==
currentSystem.TractSystemName) > 0)
{

```

```

        TractData.TractTablesBuffer[TractData.TractTablesBuffer.FindIndex(ind
=> ind.Layer == currentSystem.TractSystemName)] = tt;
    }
    else
        TractData.TractTablesBuffer.Add(tt);
    }
}
}

```

```

private bool needRunGenerateScript(WizardPage page)
{
    return (MainWizard.CurrentPage == page &&
TractData.TractTablesBuffer.Where(t => t.WzrdPage == page).Count() == 0) ||
pageChanged;
}

```

```

private void OdsStgPage_Enter(object sender, RoutedEventArgs e)
{
    // Присвоение переменной curentSysteam
    // значение следующей системы и
    // генерация тракта загрузки для нее

    WizardPage pg = (WizardPage)sender;
    TractTable previousTable = TractData.TractTablesBuffer.First(t => t.WzrdPage
== pg.PreviousPage);
    currentSystem = TractData.TractSystems.First(t => t.TractSystemName ==
"ODS.STG");

    if (needRunGenerateScript(pg))
    {
        generateTractTable(previousTable, pg);
    }
}

```



```

        TractTable currentTable = TractData.TractTablesBuffer.First(t => t.WzrdPage ==
pg);
        drawPage(OdsStgPage, currentTable);
    }

private void OdsRawPage_Enter(object sender, RoutedEventArgs e)
{
    // Присвоение переменной curentSysteam
    // значение следующей системы и
    // генерация тракта загрузки для нее

    WizardPage pg = (WizardPage)sender;
    TractTable previousTable = TractData.TractTablesBuffer.First(t => t.WzrdPage
== pg.PreviousPage);
    currentSystem = TractData.TractSystems.First(t => t.TractSystemName ==
"ODS.RAW");
    if (needRunGenerateScript(pg))
    {
        generateTractTable(previousTable, pg);
    }
    TractTable currentTable = TractData.TractTablesBuffer.First(t => t.WzrdPage ==
pg);
    drawPage(OdsRawPage, currentTable);
}

private void OdsExPage_Enter(object sender, RoutedEventArgs e)
{
    // Присвоение переменной curentSysteam
    // значение следующей системы и
    // генерация тракта загрузки для нее

    WizardPage pg = (WizardPage)sender;

```

```

        TractTable previousTable = TractData.TractTablesBuffer.First(t => t.WzrdPage
== pg.PreviousPage);
        currentSystem = TractData.TractSystems.First(t => t.TractSystemName ==
"ODS.EX");
        if (needRunGenerateScript(pg))
        {
            generateTractTable(previousTable, pg);
        }
        TractTable currentTable = TractData.TractTablesBuffer.First(t => t.WzrdPage ==
pg);
        drawPage(OdsExPage, currentTable);
    }

    private void StagefdPage_Enter(object sender, RoutedEventArgs e)
    {
        // Присвоение переменной curentSysteam
        // значение следующей системы и
        // генерация тракта загрузки для нее

        WizardPage pg = (WizardPage)sender;
        TractTable previousTable = TractData.TractTablesBuffer.First(t => t.WzrdPage
== pg.PreviousPage);
        currentSystem = TractData.TractSystems.First(t => t.TractSystemName ==
"STAGEFD");
        if (needRunGenerateScript(pg))
        {
            generateTractTable(previousTable, pg);
        }
        TractTable currentTable = TractData.TractTablesBuffer.First(t => t.WzrdPage ==
pg);
        drawPage(StagefdPage, currentTable);
    }

```

```

private void CalcfdPage_Enter(object sender, RoutedEventArgs e)
{
    // Присвоение переменной curentSysteam
    // значение следующей системы и
    // генерация тракта загрузки для нее

    WizardPage pg = (WizardPage)sender;
    TractTable previousTable = TractData.TractTablesBuffer.First(t => t.WzrdPage
== pg.PreviousPage);
    currentSystem = TractData.TractSystems.First(t => t.TractSystemName ==
"CALCFD");
    if (needRunGenerateScript(pg))
    {
        generateTractTable(previousTable, pg);
    }
    TractTable currentTable = TractData.TractTablesBuffer.First(t => t.WzrdPage ==
pg);
    drawPage(CalcfdPage, currentTable);
}

private void OlapPage_Enter(object sender, RoutedEventArgs e)
{
    // Присвоение переменной curentSysteam
    // значение следующей системы и
    // генерация тракта загрузки для нее

    WizardPage pg = (WizardPage)sender;
    TractTable previousTable = TractData.TractTablesBuffer.First(t =>
t.WzrdPage == pg.PreviousPage);
    currentSystem = TractData.TractSystems.First(t =>
t.TractSystemName == "OLAP+KIHFD");
    if (needRunGenerateScript(pg))
    {

```

```

        generateTractTable(previousTable, pg);
    }
    TractTable currentTable = TractData.TractTablesBuffer.First(t =>
t.WzrdPage == pg);
    drawPage(OlapPage, currentTable);
}

private void MainWizard_Help(object sender, RoutedEventArgs e)
{
    SaveScripts();
}

private void SaveScripts()
{
    // Инициализация окна выбора директории
    // для сохранения
    WinForms.FolderBrowserDialog fd = new WinForms.FolderBrowserDialog();
    WinForms.DialogResult result = fd.ShowDialog();
    if (result == WinForms.DialogResult.OK)
    {
        // Сохранения скриптов из буфера
        foreach (TractTable table in TractData.TractTablesBuffer)
        {
            if (table.Layer != TractData.StartLayerName)
            {
                // Создание папки с названием системы
                var layerDir =
Directory.CreateDirectory(System.IO.Path.Combine(fd.SelectedPath, table.Layer));

                TractTable previousTable = TractData.TractTablesBuffer.TakeWhile(
x => x != table).DefaultIfEmpty(
TractData.TractTablesBuffer[TractData.TractTablesBuffer.Count - 1]).LastOrDefault();

```

```

// Запись скриптов в файл

if (previousTable.SyncType != "VIEW")
{
    using (StreamWriter sw =
File.CreateText(System.IO.Path.Combine(fd.SelectedPath, layerDir.Name,
String.Format("{0}.{1}.sql", table.TSchema, table.TName))))
    {
        sw.WriteLine(table.SqlText);
    }
}

foreach (TractLayerArtifact art in table.Artifacts)
{
    if (!new String[] { "CONSTRAINT", "INDEX", "TRIGGER"
}.Contains(art.Type))
    {
        using (StreamWriter sw =
File.CreateText(System.IO.Path.Combine(fd.SelectedPath, layerDir.Name,
String.Format("{0}.sql", art.Name))))
        {
            sw.WriteLine(art.SqlText);
        }
    }
}

this.Close();
}
}

private void IntroCommentLB_AddingNewItem(object sender,
AddingNewItemEventArgs e)
{

```

```

DataGrid dg = (DataGrid)sender;
if (dg.Items.Count == 1)
{
    dg.CanUserAddRows = false;
    dg.CanUserDeleteRows = false;
}
}

private String RenameExtSysName(String seqName)
{
    // Приведение к общему вид наименования
    // системы

    String res = seqName;
    if (res.Contains("\") || res.Contains("/") || res.Contains("."))
    {
        res = res.Replace("\", "");
        if (res.StartsWith("/"))
        {
            res = res.Remove(0, 1);
        }
        res = res.Replace("/", "_");
        res = res.Replace(".", "_");
    }
    return res;
}

private String RenameType(String oldType)
{
    // Приведение к общему виду типов
    // столбцов таблицы

    String res = oldType.ToUpper();

```

```

foreach (KeyValuePair<String, String> pair in TractData.TypeTrans)
{
    if (pair.Key == res)
    {
        res = res.Replace(pair.Key, pair.Value);
    }
    else if (res.StartsWith(pair.Key + "("))
    {
        res = res.Replace(pair.Key, pair.Value);
    }
}
return res;
}

private void MainWizard_Finish(object sender,
Xceed.Wpf.Toolkit.Core.CancelRoutedEventArgs e)
{
    // Обработка события нажатия на кнопку
    // SaveScripts and Finish

    SaveScripts();
}
}
}

```

A.2 Листинг класса ArtifactWindow

```
/* ArtifactWindow - класс окна приложения, для просмотра и правки текста
* сформированных скриптов.
* Используемые переменные:
*     table - таблица для которой создан скрипт;
*     artifact - сформированный скрипт;
*     system - система для которой создан скрипт.
*/

public partial class ArtifactWindow : Window
{
    private TractTable table;
    private TractLayerArtifact artifact;
    private TractSystem system;
    public ArtifactWindow(TractTable tbl, Int32 rowNumber)
    {
        InitializeComponent();

                                                // Отображение скрипта

        table = tbl;
        artifact = tbl.Artifacts[rowNumber];
        system = TractData.TractSystems.First(t => t.TractSystemName == tbl.Layer);

        var art = tbl.Artifacts[rowNumber].SqlText;
        var flowDoc = new FlowDocument();

        Paragraph para = new Paragraph();
        para.Inlines.Add(new Run(art));
        flowDoc.Blocks.Add(para);
        richTextBox.Document = flowDoc;
    }
}
```



```

private void CloseBtn_Click(object sender, RoutedEventArgs e)
{
    this.Close();
}

private void SaveBtn_Click(object sender, RoutedEventArgs e)
{
    // Подключение к БД
    using (SqlConnection conn = new SqlConnection(system.ConnStr))
    {
        var sqlText = new TextRange(richTextBox.Document.ContentStart,
richTextBox.Document.ContentEnd).Text;

        if (artifact.Type == "PROCEDURE" || artifact.Type == "CREATE SCRIPT")
        {
            sqlText = "exec (" + sqlText + ")";
        }

        sqlText = "set NOEXEC ON" + Environment.NewLine + sqlText +
Environment.NewLine + "set NOEXEC OFF";

        SqlCommand cmd = new SqlCommand(sqlText, conn);
        conn.Open();

        // Выполнение скрипта на сервере

        try
        {
            cmd.ExecuteNonQuery();
            artifact.SqlText = new TextRange(richTextBox.Document.ContentStart,
richTextBox.Document.ContentEnd).Text;
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
        }
    }
}

```

A.3 Листинг класса TractData

```
/* TractData - класс представляющий данные тракта загрузки.
* Используемые переменные:
*     TractSystems - список систем;
*     TractTablesBuffer - буфер, хранящий таблицы;
*     StartLayerName - наименование начальной системы;
*     TypeTrans - тип транзакции.
*/

public static class TractData
{
    public static List<TractSystem> TractSystems;
    public static List<TractTable> TractTablesBuffer;
    public static String StartLayerName;
    public static List<KeyValuePair<String, String>> TypeTrans;
}
```

A.4 Листинг класса TractLayerArtifact

```
/* TractLayerArtifact - класс представляющий артефакт слоя.
* Используемые переменные:
*     Name - наименование;
*     Type - тип артефакта;
*     SqlText - текст sql скрипта.
*/

public class TractLayerArtifact
{
    public String Name { get; set; }
    public String Type { get; set; }
    public String SqlText { get; set; } }
}
```

A.5 Листинг класса TractTableColumn

```
/* TractTableColumn - класс представляющий данные колонок таблиц.  
* Используемые переменные:  
*   PrimaryKeyNumber - номер первичного ключа;  
*   CName - наименование столбца;  
*   CType - тип столбца;  
*   CComment - комментарии.  
*/  
  
public class TractTableColumn  
{  
    public Int32? PrimaryKeyNumber { get; set; }  
    public String CName { get; set; }  
    public String CType { get; set; }  
    public String CComment { get; set; }  
}
```

А.6 Листинг класса TractLayerExtProp

```
/* TractLayerExtProp - класс представляющий артефакт слоя.  
* Используемые переменные:  
* _name - приватная переменная, наименование;  
* Name - открытая переменная, наименование;  
* Value - текст описания;  
*/  
  
public class TractLayerExtProp  
{  
    private String _name;  
    public String Name  
    {  
        get  
        {  
            if (String.IsNullOrEmpty(_name))  
            {  
                _name = "MS_Description";  
            }  
            return _name;  
        }  
        set { _name = value; }  
    }  
    public String Value { get; set; }  
}
```

A.7 Листинг класса TractSystem

```
/* TractSystem - класс представляющий данные о системе .  
* Используемые переменные:  
*   TractSystemId - id системы;  
*   TractSystemName - наименование системы;  
*   ScriptGetTableMetadata - скрипт получения метаданных таблицы;  
*   ConnStr - строка подключения;  
*   TractSystemType - тип ситемы;  
*   IsExternal - признак системы, внешняя/не внешняя;  
*   TractSystemDestinations - система назначения;  
*   RoColumns - список столбцов таблциы;  
*   HiddenColumns - список скрытых столбцов;  
*   CanAddRowComment - признак строки, можно ли добавить комментарий;  
*   CanAddRowField - признак строки, можно ли добавить поле;  
*   CanAddRowArtifact - признак строки, можно ли добавить артефакт;  
*   CanDeleteRowComment - признак строки, можно ли удалить комментарий;  
*   CanDeleteRowField - признак строки, можно ли удалить поле;  
*   CanDeleteRowArtifact - признак строки, можно ли удалить поле;  
*   IsShowInc - признак строки, отображать или нет;  
*   NameWithServer - наименование сервера .  
*/  
  
public class TractSystem  
{  
    public Int32 TractSystemId;  
    public String TractSystemName;  
    public String ScriptGetTableMetadata;  
    public String ConnStr;  
    public String TractSystemType;  
    public Boolean IsExternal;  
    public List<TractSystem> TractSystemDestinations;
```

```

public List<int> RoColumns;
public List<int> HiddenColumns;
public bool CanAddRowComment;
public bool CanAddRowField;
public bool CanAddRowArtifact;
public bool CanDeleteRowComment;
public bool CanDeleteRowField;
public bool CanDeleteRowArtifact;
public bool IsShowInc;
public String NameWithServer
{
    get
    {
        if (this.TractSystemType.ToLower() != "mssql")
            return $"{this.TractSystemName} ({this.TractSystemType})";
        else
        {
            SqlConnectionStringBuilder builder = new
SqlConnectionStringBuilder(this.ConnStr);
            return $"{this.TractSystemName} ({builder.DataSource})";
        }
    }
}
}

```

A.8 Листинг класса TractTable

```
/* TractTable - класс представляющий данные таблицы.
 * Используемые переменные:
 * TSchema - наименование схемы таблицы;
 * TName - наименование таблицы;
 * TExtProps - описание таблицы;
 * TColumns - колонки таблицы;
 * IsExternal - признак внешней системы;
 * WzrdPage - страница соответствующая данной таблице;
 * Artifacts - артефакты таблицы;
 * Layer - система;
 * TSchemaNew - новое наименование схемы таблицы;
 * TNameNew - новое наименование таблицы;
 * SyncType - тип синхронизации;
 * SqlText - sql текст скрипта для создания таблицы.
 */

public class TractTable
{
    public String TSchema;
    public String TName;
    public ObservableCollection<TractLayerExtProp> TExtProps;
    public ObservableCollection<TractTableColumn> TColumns;
    public Boolean IsExternal;
    public WizardPage WzrdPage;
    public ObservableCollection<TractLayerArtifact> Artifacts;
    public String Layer;
    public String TSchemaNew;
    public String TNameNew;
    public String SyncType;
```

```

public String SqlText
{
    get
    {
        string stext = "";
        string fieldComments = "";
        stext = "CREATE TABLE " + TSchema + "." + TName + "(" +
Environment.NewLine;
        foreach (TractTableColumn clmn in this.TColumns)
        {
            stext = stext + String.Format("    {0} {1}{2} NULL," , clmn.CName,
clmn.CType, clmn.PrimaryKeyNumber == null ? "" : " NOT") + Environment.NewLine;
            if (!String.IsNullOrEmpty(clmn.CComment))
                fieldComments = fieldComments +
                    String.Format("EXEC sys.sp_addextendedproperty
@name=N'MS_Description', @value=N'{0}',
@level0type=N'SHEMA',@level0name=N'{1}',
@level1type=N'TABLE',@level1name=N'{2}',
@level2type=N'COLUMN',@level2name=N'{3}'", clmn.CComment, this.TSchema,
this.TName, clmn.CName)
                    + Environment.NewLine + "GO" + Environment.NewLine +
Environment.NewLine;
        }
        stext = stext + Environment.NewLine + ")" + Environment.NewLine + "GO" +
Environment.NewLine;
        stext = stext + fieldComments;

        bool msDescrAdded = false;
        foreach (TractLayerExtProp prp in this.TExtProps)
        {

```



```

        if (!String.IsNullOrEmpty(prp.Name) &&
!String.IsNullOrEmpty(prp.Value) && ((prp.Name == "MS_Description" &&
!msDescrAdded) || prp.Name != "MS_Description"))
        {
            stext = stext +
                String.Format(@"EXEC sys.sp_addextendedproperty @name=N'{0}',
@value=N'{1}', @level0type=N'SHEMA',@level0name=N'{2}',
@level1type=N'TABLE',@level1name=N'{3}'", prp.Name, prp.Value, this.TSchema,
this.TName);

            stext = stext + Environment.NewLine + "GO" + Environment.NewLine +
Environment.NewLine;

            msDescrAdded = prp.Name == "MS_Description";
        }
    }
    foreach (TractLayerArtifact art in this.Artifacts)
    {
        if (new String[] { "CONSTRAINT", "INDEX", "TRIGGER"
}.Contains(art.Type))
        {
            stext = stext + art.SqlText + Environment.NewLine + "GO" +
Environment.NewLine + Environment.NewLine;
        }
    }

    return stext;
}
}
}

```

A.9 Листинг класса Tools

// Tools - класс содержащий вспомогательные методы

```
public static class Tools
{
    public static String GetCleanScript(String ScriptText)
    {
        // Отчистка скрипта от лишних символов
        String res = ScriptText;
        res = res.
            Replace("[", "").
            Replace("]", "").
            Replace(Environment.NewLine, "").
            Replace("\t", "").
            Replace(" ", "");
        return res;
    }

    public static String ReplaceServer(String ConnStr)
    {
        // Вставка в строку подключения значений
        // из файла App.config

        SqlConnectionStringBuilder builder = new
        SqlConnectionStringBuilder(ConnStr);

        if
        (!String.IsNullOrEmpty(ConfigurationManager.AppSettings.Get(builder.DataSource
        e)))
        {
```

```
        builder.DataSource =  
ConfigurationManager.AppSettings.Get(builder.DataSource);  
    }  
    return builder.ToString();  
}  
}
```

A.10 Листинг хранимой процедуры dbo.create_layer_mdata

```
USE [UTILITY]
GO

SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE PROCEDURE [dbo].[create_layer_mdata]
    @src_table_descr dbo.list_table readonly,
    @src_column_descr dbo.list_column readonly,
    @ep dbo.ext_props readonly,
    @layer_name_src varchar(255),
    @layer_name_dst varchar(255),
    -- view или proc, по дефолту создается вьюшка
    @ods_ex varchar(10) = "",
    -- sds или standart, по дефолту создается станд синхр-я
    @sync_type varchar(50) = ""
AS
BEGIN

SET XACT_ABORT ON;
SET NOCOUNT ON;

-- Список таблиц, по которым надо получить скрипты >

DECLARE @LIST TABLE (
                                ID_TBL          INT,
```

```

        SCHEMA_SRC VARCHAR(255),
        TABLE_SRC VARCHAR(255),
        IS_INCREMENT BIT,
        SCHEMA_ORIG VARCHAR(255),
        TABLE_ORIG VARCHAR(255),
        PRIM_SYS VARCHAR(255),
        SCHEMA_DST VARCHAR(255),
        TABLE_DST VARCHAR(255)
    );

INSERT INTO @LIST (ID_TBL,
        SCHEMA_SRC,
        TABLE_SRC,
        IS_INCREMENT
    )

SELECT td.ID_TBL,
        td.SCHEMA_NAME_SRC,
        td.TABLE_NAME_SRC,
        td.IS_INCREMENT
FROM @SRC_TABLE_DESCR AS td;

IF @LAYER_NAME_DST = 'CALCFD'
BEGIN
-- Если на STAGEFD у таблицы есть поле IS_DEL, то обязательно должен быть
заполнен параметр IS_INCREMENT = 1
        IF EXISTS (SELECT CD.CLMN_DST, CD.CLMN_SRC
                FROM @SRC_COLUMN_DESCR CD
                WHERE IIF(ISNULL(CD.CLMN_DST, '') = '',
cd.CLMN_SRC, CD.CLMN_DST) = 'IS_DEL')
        BEGIN
                UPDATE L
                SET L.IS_INCREMENT=1

```

```

FROM @LIST L
END
END

```

-- Создание таблицы для результатов

```

IF OBJECT_ID('tempdb..#CREATE_OBJ') IS NOT NULL
BEGIN
    drop table #CREATE_OBJ
END
CREATE TABLE #CREATE_OBJ
(
    ID_CREATE_OBJ          INT IDENTITY          NOT NULL,
    ID_TABLE               INT                  NOT NULL,
    SCHEMA_DST             VARCHAR(50)          NULL,
    TABLE_DST             VARCHAR(255)         NULL,
    COLUMN_NAME            VARCHAR(255)         NULL,
    COLUMN_ID              INT                  NULL,
    TYPE                   VARCHAR(255)         NULL,
    PK                     INT                  NULL,
    VALUE                  VARCHAR(MAX)         NULL
)

```

-- Определение шаблона схемы и шаблона имени таблицы для таблицы-источника
 -- и таблицы-приемника

```

DECLARE @SCHEMA_PATTERN_SRC VARCHAR(255) = "",
        @PREFIX_TABLE_SRC  VARCHAR(255) = "",
        @SCHEMA_PATTERN_DST VARCHAR(255),
        @PREFIX_TABLE_DST  VARCHAR(255)

SELECT @SCHEMA_PATTERN_SRC = pn.SCHEMA_PATTERN,
       @PREFIX_TABLE_SRC = pn.PREFIX_TABLE

```

```

FROM dbo.DIC_PATTERN_NAMING pn
    INNER JOIN dbo.DIC_LAYER l
        ON l.ID_LAYER = pn.ID_LAYER
    INNER JOIN dbo.DIC_TYPE_OBJECT tob
        ON tob.ID_TYPE_OBJECT = pn.ID_TYPE_OBJECT
WHERE l.NAME = @LAYER_NAME_SRC
    AND tob.NAME IN ('table', 'view')

```

```

SELECT @SCHEMA_PATTERN_DST = pn.SCHEMA_PATTERN,
       @PREFIX_TABLE_DST = pn.PREFIX_TABLE
FROM dbo.DIC_PATTERN_NAMING pn
    INNER JOIN dbo.DIC_LAYER l
        ON l.ID_LAYER = pn.ID_LAYER
    INNER JOIN dbo.DIC_TYPE_OBJECT tob
        ON tob.ID_TYPE_OBJECT = pn.ID_TYPE_OBJECT
WHERE l.NAME = @LAYER_NAME_DST
    AND tob.NAME IN ('table', 'view')

```

-- Определение схемы и названия таблицы-источника

```

;with cte as
    (SELECT l.ID_TBL,
            iif(CHARINDEX('#', @PREFIX_TABLE_SRC)>0,
stuff(l.TABLE_SRC, 1, CHARINDEX('#', @PREFIX_TABLE_SRC)-1, ''), '') s1,
            iif(CHARINDEX('#', @SCHEMA_PATTERN_SRC)>0,
stuff(l.SCHEMA_SRC, 1, CHARINDEX('#', @SCHEMA_PATTERN_SRC)-1, ''), '') s2,
            iif(CHARINDEX('$', @PREFIX_TABLE_SRC)>0,
stuff(l.TABLE_SRC, 1, CHARINDEX('$', @PREFIX_TABLE_SRC)-1, ''), '') ps1,
            iif(CHARINDEX('$', @SCHEMA_PATTERN_SRC)>0,
stuff(l.SCHEMA_SRC, 1, CHARINDEX('$', @SCHEMA_PATTERN_SRC)-1, ''), '') ps2
    FROM @LIST l)
UPDATE l
SET SCHEMA_ORIG = iif(CHARINDEX('_', c.s1)>0, left(c.s1, CHARINDEX('_',
c.s1)- 1), '') + left(c.s2, iif(CHARINDEX('_', c.s2)>0, CHARINDEX('_', c.s2)-1, len(c.s2))),

```

```

        PRIM_SYS = iif(CHARINDEX('_', c.ps1)>0, left(c.ps1, CHARINDEX('_',
c.ps1)- 1), '') + c.ps2
FROM cte c
        INNER JOIN @LIST l
                ON l.ID_TBL = c.ID_TBL

UPDATE l
SET PRIM_SYS = lower(l2.NAME_S)
FROM @LIST l
        CROSS JOIN dbo.DIC_LAYER l2
WHERE l2.NAME = @LAYER_NAME_SRC
        AND l.PRIM_SYS = ''

UPDATE @LIST
SET TABLE_ORIG = stuff(TABLE_SRC, 1,
len(replace(replace(@PREFIX_TABLE_SRC, '#', SCHEMA_ORIG), '$', PRIM_SYS)),
'')

-- Определение схемы и названия таблицы-приемника
UPDATE @LIST
SET SCHEMA_DST = REPLACE(REPLACE(@SCHEMA_PATTERN_DST,
'#', SCHEMA_ORIG), '$', PRIM_SYS),
        TABLE_DST = REPLACE(REPLACE(@PREFIX_TABLE_DST, '#',
SCHEMA_ORIG), '$', PRIM_SYS)+TABLE_ORIG

-- Для таблиц из Афины - исключение, для всех одна схема на STAGEFD
IF @LAYER_NAME_DST = 'STAGEFD'
BEGIN
        UPDATE @LIST
        SET SCHEMA_DST = iif(SCHEMA_DST in ('psb', 'psbe', 'psbf'), 'afina',
SCHEMA_DST)
END

```



```

UPDATE L
SET SCHEMA_DST = S.SCHEMA_NAME_DST
FROM @LIST L
      INNER JOIN @SRC_TABLE_DESCR S
            ON L.ID_TBL = S.ID_TBL
WHERE ISNULL(S.SCHEMA_NAME_DST, "") <> ""

```

```

UPDATE L
SET TABLE_DST = S.TABLE_NAME_DST
FROM @LIST L
      INNER JOIN @SRC_TABLE_DESCR S
            ON L.ID_TBL = S.ID_TBL
WHERE ISNULL(S.TABLE_NAME_DST, "") <> ""

```

-- В таблицу @PRIM вставляется список колонок, заданных на вход

```

DECLARE @PRIM TABLE (
                                ID_TBL          INT,
                                CLMN            VARCHAR(255),
                                ORDR            INT,
                                TYP             VARCHAR(255),
                                PK              INT
                                );

INSERT INTO @PRIM (
                                ID_TBL,
                                CLMN,
                                ORDR,
                                TYP,
                                PK
                                )

SELECT cd.ID_TBL,
      IIF(ISNULL(CD.CLMN_DST, "") = "", cd.CLMN_SRC, CD.CLMN_DST),

```

```

cd.ORDR,
    IIF(ISNULL(CD.TYP_DST, "") = "", cd.TYP_SRC, CD.TYP_DST),
    IIF(PK = 0, NULL, PK)
FROM @SRC_COLUMN_DESCR cd;

-- Если в первичной есть поле RV, то оно превращается в RV_SYSTEM с типом
binary(8)

IF @layer_name_dst = 'ODS.STG'
BEGIN
    UPDATE @PRIM
    SET CLMN = 'RV_SYSTEM'
    WHERE CLMN = 'RV'
    UPDATE @PRIM
    SET TYP = 'binary(8)'
    WHERE TYP in ('timestamp', 'rowversion')
END

IF @LAYER_NAME_DST = 'CALCFD'
BEGIN
--Переименование ПК из первичной, если не задано новое название
    IF NOT EXISTS ( SELECT P.clmn
                    FROM @PRIM P
                    INNER JOIN @LIST L ON P.ID_TBL =
L.ID_TBL

                    WHERE P.PK>0
                    AND P.CLMN =
'ID_'+REPLACE(L.TABLE_DST,'TBL_',")+ '_SYSTEM')
    BEGIN
        UPDATE P
        SET P.CLMN =
'ID_'+REPLACE(L.TABLE_DST,'TBL_',")+ '_SYSTEM'
        FROM @PRIM P

```

```

INNER JOIN @LIST L ON P.ID_TBL = L.ID_TBL
        WHERE P.PK>0
    END

-- ID_CONNECT_CODE становится ПК
    UPDATE P
    SET P.PK =1
    FROM @PRIM P
    WHERE P.clmn = 'ID_CONNECT_CODE'
    END

-- При переходе из CALCFD на OLAP поле RV не переносится
    IF @LAYER_NAME_DST = 'OLAP+KIHFD'
    BEGIN
        DELETE P
        FROM @PRIM p
        WHERE p.CLMN = 'RV'
    END

-- В таблицу @EXT_PROPS вставляется описание колонок, заданные на вход, а
-- также название таблицы-источника
    DECLARE @EXT_PROPS AS TABLE (
        ID_TABLE      INT,
        ID_COLUMN     INT,
        EP_NAME       VARCHAR(8000),
        EP_VALUE      VARCHAR(8000))
    INSERT INTO @EXT_PROPS (ID_TABLE,
                                ID_COLUMN,
                                EP_NAME,
                                EP_VALUE)

    SELECT ep.ID_TABLE,
           ep.ID_COLUMN,
           ep.EP_NAME,

```

```

        IIF(ISNULL(EP.EP_VALUE_DST,"")="", ep.EP_VALUE_SRC,
EP.EP_VALUE_DST)
FROM @ep ep
UNION ALL
SELECT l.ID_TBL ID_TABLE,
       NULL ID_COLUMN,
       'Source_table_name' EP_NAME,
       l.SCHEMA_SRC + '.' + l.TABLE_SRC EP_VALUE
FROM @LIST l

```

-- В первом слое необходимо добавить источник данных, пакет загрузки, владелец

```

IF @LAYER_NAME_DST = 'ODS.STG'
BEGIN
    INSERT INTO @EXT_PROPS (ID_TABLE,
                           ID_COLUMN,
                           EP_NAME,
                           EP_VALUE)

    SELECT l.ID_TBL,
           NULL,
           'Источник данных',
           @LAYER_NAME_SRC

    FROM @LIST l
    UNION ALL
    SELECT l.ID_TBL,
           NULL,
           'Пакет загрузки',
           "

    FROM @LIST l
    UNION ALL
    SELECT l.ID_TBL,
           NULL,
           'Владелец',

```

```

"

FROM @LIST L

END

-- Переменные для оформления скриптов
DECLARE @r CHAR(1)= CHAR(10),
        @rr CHAR(2) = CHAR(10) + CHAR(10),
        @t CHAR(1) = CHAR(9)

-- Наполнение временных переменных для таблиц, триггеров
declare @view table (ALIAS char(1),
                     SELECT_COLUMNS varchar(8000))

insert into @view (ALIAS,
                  SELECT_COLUMNS)

select ALIAS,
       SELECT_COLUMNS
FROM @list l
outer apply (
    Select vl.ALIAS,
           @r+'UNION ALL' + @r + 'SELECT
'+stuff(xv.clmn,1,2,"")+@r+','+vl.d+' IS_DEL' + @r + 'FROM ' + l.schema_src +
'+l.table_src + vl.nm + ' ' + vl.alias + @r
           ('NULL'+@t, 'I', '_DELETED', 'd'))
    vl (v, d, nm, ALIAS)
outer apply
(
    SELECT @r+', ' + iif(p.pk>0 or p.clmn='RV', vl.alias + '.', vl.v) + p.clmn
    FROM @PRIM p
    WHERE p.id_tbl = l.id_tbl
    ORDER BY p.ordr FOR XML PATH("")
) xv (clmn)
) x (ALIAS, SELECT_COLUMNS)

```

```

IF OBJECT_ID('tempdb..#tbl_body') IS NOT NULL
    BEGIN
        drop table #tbl_body
    END
select l.id_tbl AS ID_TBL,
       x1.clms AS TBL_COLUMNS,
       x2.pk AS PK,
       x3.clms AS PK_COLUMNS,
       x4.i AS PK_FOR_TRIGGER,
       (SELECT v.SELECT_COLUMNS + 'WHERE EXISTS (SELECT 1
FROM dbo.TBL_LOAD_DATE ld WHERE ld.[NAME] = ''+ l.SCHEMA_DST +
'+stuff(l.TABLE_DST, 1, 2, '')+'' and [DB_NAME] = "STAGEFD" and
isnull(ld.RV_FROM, 0x0000000000000000) <' + v.alias + '.RV and
isnull(ld.RV_TO,0xFFFFFFFFFFFFFFFF) >= ' + v.alias + '.RV) '
       from @view v
       for xml path(''))+ ' AND' v_inc,
       (select v.SELECT_COLUMNS + " from @view v for xml path('')) + '
WHERE' v_full,
       x6.c,
       ex_prop.s AS EX_PROPS
into #tbl_body
FROM @list l
    CROSS APPLY
    (
        SELECT @r+','+p.clmn + @t + UPPER(p.typ)
        FROM @PRIM p
        WHERE p.id_tbl = l.id_tbl
        ORDER BY p.ordr FOR XML PATH('')
    ) x1(clms)
    OUTER APPLY
    (
        SELECT ','+p.clmn

```

```

FROM @PRIM p
WHERE p.id_tbl = l.id_tbl
      AND p.pk > 0
ORDER BY p.pk FOR XML PATH("")
) x2(pk)
OUTER APPLY
(
  SELECT @r+', '+p.clmn+@t+UPPER(p.typ)
  FROM @PRIM p
  WHERE p.id_tbl = l.id_tbl
        AND p.pk > 0
  ORDER BY p.pk FOR XML PATH("")
) x3(clms)
OUTER APPLY
(
  SELECT ')'+@r+'SELECT '+STUFF(ins.c,1,2,")
  FROM (VALUES("),(d.)) prf(alias)
OUTER APPLY
(
  SELECT ', '+prf.alias+p.clmn
  FROM @PRIM p
  WHERE p.id_tbl = l.id_tbl
        AND p.pk > 0
  ORDER BY p.ordr FOR XML PATH(")) ins(c)
ORDER BY prf.alias FOR XML PATH(")) x4(i)
OUTER APPLY
(
  SELECT ' AND t.' + p.clmn+ '=' + 'd.' + p.clmn
  FROM @PRIM p
  WHERE p.id_tbl = l.id_tbl AND p.pk > 0
  ORDER BY p.pk FOR XML PATH("")
) x6(c)

```

```

outer apply (
    SELECT @rr + 'EXEC sys.sp_addextendedproperty' + @r +
        '@name = N''' + ep.ep_name + ''', ' + @r +
        '@value = N''' + ep.ep_value + ''', ' + @r +
        '@level0type = N"SCHEMA",' + @r +
        '@level0name = N''' + l.schema_dst + ''', ' + @r +
        '@level1type = N"TABLE",' + @r +
        '@level1name = N''' + l.table_dst + '''+
        iif(ep.id_column is null, ', ' + @r +
        '@level2type = N"COLUMN",' + @r + '@level2name = N''' + p.clmn + ''')+@r + 'GO'
    from @ext_props ep
        inner join @list l
            on l.id_tbl = ep.id_table
        left join @PRIM p
            on p.id_tbl = ep.id_table
            and p.ordr = ep.id_column
    FOR XML PATH('')
) ex_prop (s)

-- Наполнение временных переменных для ETL-процедуры
IF OBJECT_ID('tempdb..#proc_body') IS NOT NULL
    BEGIN
        drop table #proc_body
    END
select l.id_tbl,
       l.schema_dst,
       l.table_dst,
       l.is_increment,
       x1.crt,
       x2.ins,
       x3.mpk,
       x4.e,
       x5.u,

```



```

        x6.i
into #proc_body
FROM @list l
    CROSS APPLY (
        SELECT @r+','+p.clmn+@t+ p.typ
        FROM @PRIM p
        WHERE p.id_tbl = l.id_tbl
        ORDER BY p.ordr
        FOR XML PATH("")
    ) x1(crt)
    CROSS APPLY (
        SELECT ')'+@r+'SELECT '+@r+STUFF(xi.ins,1,2,")
FROM(VALUES("),( 's.')) prf(alias)
        CROSS APPLY
        (
            SELECT @r+',' + prf.alias+p.clmn
            FROM @PRIM p
            WHERE p.id_tbl = l.id_tbl
            ORDER BY p.ordr
            FOR XML PATH("")
        ) xi(ins)
        ORDER BY prf.alias FOR XML PATH("")
    ) x2(ins)
    CROSS APPLY (
        SELECT ' AND t.'+p.clmn+' = '+ 's.'+p.clmn
        FROM @PRIM p
        WHERE p.id_tbl = l.id_tbl
        AND p.pk > 0
        ORDER BY p.pk FOR XML PATH("")
    ) x3(mp)
    CROSS APPLY (

```

```

SELECT @r+'EXCEPT'+@r+'SELECT'+STUFF(xm.ex,1,1,")
FROM(VALUES('t.'),('s.')) prf(alias)
    CROSS APPLY
    (
        SELECT ', '+prf.alias+p.clmn
        FROM @PRIM p
        WHERE p.id_tbl = l.id_tbl
            AND ISNULL(p.pk, 0) = 0
        ORDER BY p.ordr FOR XML PATH("")
    ) xm(ex)
    ORDER BY prf.alias FOR XML PATH("")
) x4(e)
OUTER APPLY (
    SELECT @r + ', ' + p.clmn + ' = s.' + p.clmn
    FROM @PRIM p
    WHERE p.id_tbl = l.id_tbl
        AND ISNULL(p.pk,0) = 0
    ORDER BY p.ordr FOR XML PATH("")
) x5 (u)
CROSS APPLY (
    SELECT ')'+@r+'VALUES('+STUFF(ins.c,1,2,")
    FROM(VALUES("),('s.')) prf(alias)
        CROSS APPLY
        (
            SELECT ', '+prf.alias+p.clmn
            FROM @PRIM p
            WHERE p.id_tbl = l.id_tbl
            ORDER BY p.ordr FOR XML PATH("")
        ) ins(c)
    ORDER BY prf.alias FOR XML PATH("")
) x6(i)

```

-- Создание таблиц, представлений, триггеров

IF @layer_name_dst = 'ODS.STG'

begin

if exists (select l.is_increment

FROM #tbl_body t

inner join @list l

on l.id_tbl = t.id_tbl

where l.is_increment = 1)

begin

insert into @PRIM (id_tbl, clmn, ordr, typ, pk)

select max(id_tbl) as id_tbl,

'IS_DEL' as clmn,

max(ordr)+1 as ordr,

'BIT' as typ,

null as pk

from @PRIM

insert into @ext_props (id_table, id_column, ep_name, ep_value)

select id_tbl, ordr, 'MS_Description', 'Признак удаления'

from @PRIM

where clmn='IS_DEL'

end

INSERT INTO #CREATE_OBJ (

ID_TABLE,

SCHEMA_DST,

TABLE_DST,

COLUMN_NAME,

COLUMN_ID,

TYPE,

PK,

VALUE)

```

select ep.id_table,
       l.schema_dst,
       l.table_dst,
       isnull(p.clmn, ep.ep_name) as c_name,
       p.ordr as column_id,
       p.typ as tp,
       p.pk as pk,
       ep.ep_value as [value]
from @PRIM p
      right join @ext_props ep
        on p.id_tbl = ep.id_table
        and p.ordr = ep.id_column
      right join @list l
        on l.id_tbl = ep.id_table
UNION ALL
SELECT t.id_tbl, l.schema_dst, l.table_dst,
       'INSERT INTO dbo.TBL_LOAD_DATE' as COLUMN_NAME,
       null as COLUMN_ID,
       'INSERT SCRIPT' as TYPE,
       null as PK,
       'IF NOT EXISTS (SELECT 1 FROM dbo.TBL_LOAD_DATE ld
WHERE ld.[NAME] = ''+ replace(l.schema_dst, 'stg_', '')+'.'+l.table_dst+'' and
ld.[DB_NAME] = "ODS")' + @r +
       'INSERT INTO dbo.TBL_LOAD_DATE ([NAME], DATE_FROM,
DATE_TO, [DB_NAME])'+@r+
       'VALUES(''+ replace(l.schema_dst, 'stg_', '') +'.'+l.table_dst +'',
"19000101", NULL, "ODS")' as VALUE
FROM #tbl_body t
      inner join @list l
        on l.id_tbl = t.id_tbl
end

```

```
IF @layer_name_dst = 'ODS.RAW'
```

```
begin
```

--В этом слое добавляется поле RV, если только этого поля уже не существует

```
if NOT EXISTS (SELECT * FROM @PRIM WHERE clmn='RV')
```

```
begin
```

```
insert into @PRIM (id_tbl, clmn, ord, typ, pk)
```

```
select max(id_tbl) as id_tbl,
```

```
      'RV' as clmn,
```

```
      max(ord)+1 as ord,
```

```
      'rowversion' as typ,
```

```
      null as pk
```

```
from @PRIM
```

```
insert into @ext_props (id_table, id_column, ep_name, ep_value)
```

```
select id_tbl, ord, 'MS_Description', 'Rowversion (механизм для отметки  
версий строк таблицы)'
```

```
from @PRIM
```

```
where clmn='RV'
```

```
end
```

```
INSERT INTO #CREATE_OBJ (
```

```
ID_TABLE,
```

```
SCHEMA_DST,
```

```
TABLE_DST,
```

```
COLUMN_NAME,
```

```
COLUMN_ID,
```

```
TYPE,
```

```
PK,
```

```
VALUE
```

```
)
```

```
select ep.id_table,
```

```

        l.schema_dst,
        l.table_dst,
        isnull(p.clmn, ep.ep_name) as c_name,
        p.ordr as column_id,
        p.typ as tp,
        p.pk as pk,
        ep.ep_value as [value]
from @PRIM p
    right join @ext_props ep
        on p.id_tbl = ep.id_table
        and p.ordr = ep.id_column
    right join @list l
        on l.id_tbl = ep.id_table

UNION ALL

-- Скрипт для создания CONSTRAINT по первичным ключам
SELECT t.id_tbl,
        l.schema_dst,
        l.table_dst,
        'PK_'+l.schema_dst+'_'+l.table_dst+REPLACE(t.pk, ',', '_') as
COLUMN_NAME,
        null as COLUMN_ID,
        'CONSTRAINT' as TYPE,
        null as PK,
        isnull('ALTER TABLE '+l.schema_dst+'.'+l.table_dst+' ADD
CONSTRAINT PK_'+l.schema_dst+'_'+l.table_dst+REPLACE(t.pk, ',', '_') + '
PRIMARY KEY CLUSTERED ('+STUFF(t.pk,1,1,")+')'-- PRIMARY KEY
NEEDED!') + @r+';' + @r + 'GO' as VALUE
FROM #tbl_body t
    inner join @list l
        on l.id_tbl = t.id_tbl

```

UNION ALL

-- Скрипт для создания индекса по RV

```
SELECT t.id_tbl,
       l.schema_dst,
       l.table_dst,
       'UIX_'+l.schema_dst+'_'+l.table_dst+'_RV' as COLUMN_NAME,
       null as COLUMN_ID,
       'INDEX' as TYPE,
       null as PK,
       'CREATE UNIQUE NONCLUSTERED INDEX
UIX_'+l.schema_dst+'_'+l.table_dst+'_RV ON '+l.schema_dst+'.'+l.table_dst+' (RV)' +
@r +
       'GO' as VALUE
FROM #tbl_body t
     inner join @list l
     on l.id_tbl = t.id_tbl
```

UNION ALL

-- Скрипт для создания таблицы логов

```
SELECT t.id_tbl,
       l.schema_dst,
       l.table_dst,
       l.schema_dst+'.'+l.table_dst+'_DELETED' as COLUMN_NAME,
       null as COLUMN_ID,
       'TABLE' as TYPE,
       null as PK,
       'CREATE TABLE
'+l.schema_dst+'.'+l.table_dst+'_DELETED'+@r+
       '('+STUFF(t.PK_COLUMNS,1,2,"")+@r+
       ',RV'+@t+'ROWVERSION'+@t+'NOT NULL'+@r+
```

```

        ',CONSTRAINT
PK_'+l.schema_dst+'_'+l.table_dst+'_DELETED_RV PRIMARY KEY CLUSTERED
(RV)'+');' + @r +

        'GO' as VALUE

FROM #tbl_body t
    inner join @list l
        on l.id_tbl = t.id_tbl

UNION ALL

-- Скрипт для создания триггера логов
SELECT t.id_tbl,
        l.schema_dst,
        l.table_dst,
        l.schema_dst+'_'+l.table_dst+'_DEL' as COLUMN_NAME,
        null as COLUMN_ID,
        'TRIGGER' as TYPE,
        null as PK,
        'CREATE TRIGGER '+l.schema_dst+'_'+l.table_dst+'_DEL ON
'+l.schema_dst+'_'+l.table_dst+'@r+
        'FOR DELETE'+@r+'AS'+@r+
        'SET NOCOUNT ON;'+@rr+
        'INSERT INTO '+l.schema_dst+'_'+l.table_dst+'_DELETED (' +
STUFF(t.PK_FOR_TRIGGER, 1, 9, ") + @r +
        'FROM deleted d' + @rr +
        'GO' as VALUE

FROM #tbl_body t
    inner join @list l
        on l.id_tbl = t.id_tbl

-- Скрипт для создания процедуры загрузки
SELECT p.id_tbl, p.schema_dst,p.table_dst,

```



```

p.schema_dst+'.LOAD_'+p.table_dst as COLUMN_NAME,
null as COLUMN_ID,
'PROCEDURE' as TYPE,
null as PK,
'CREATE PROCEDURE '+ p.schema_dst
+'.LOAD_'+p.table_dst+@r+
'AS'+@r+
'SET NOCOUNT ON;'+@rr+
'DECLARE @dt DATETIME2(0)= GETDATE()'+@rr+
'DROP TABLE IF EXISTS #src'+@rr+
'CREATE TABLE #src'+@r+'('+UPPER(STUFF(p.crt,1,2,"))
+@r+')'+@rr+
'INSERT INTO #src ('+STUFF(p.ins,1,9,")+@r+
'FROM '+l.schema_src+'.'+l.table_src+' s'+@r+
'BEGIN TRY' + @r + 'BEGIN TRANSACTION' +@rr+
'MERGE '+p.schema_dst+'.'+p.table_dst+' t'+@r+
'USING #SRC s'+@r+
'ON '+STUFF(p.mpk,1,5,")+@r+
    iff(p.u is null, ", 'WHEN MATCHED AND
EXISTS'+@r+'('+STUFF(p.e,1,8,")+ @r + '')+ IIF(p.is_increment = 0, ", +@r + 'AND
s.IS_DEL = 0') + @r
    'THEN UPDATE SET' + @r + STUFF(p.u, 1, 2, ") + @r) +
    'WHEN NOT MATCHED' + IIF(p.is_increment = 0, ", ' AND
s.IS_DEL = 0') + ' THEN' + @r + 'INSERT(' + STUFF(p.i, 1, 9, ") + ')+ @r +
    IIF(p.is_increment = 0, 'WHEN NOT MATCHED BY SOURCE',
'WHEN MATCHED AND s.IS_DEL = 1') + @r + 'THEN DELETE' + @r + ';' + @rr +
'UPDATE dbo.TBL_LOAD_DATE' + @r +
'SET DATE_FROM = DATE_TO,' + @r +
'DATE_TO = NULL' + @r +
'WHERE [NAME] = '"+p.schema_dst+'.'+p.table_dst + "' AND
DATE_TO IS NOT NULL' + @rr +

```

```

        'COMMIT TRANSACTION' + @rr +
        'END TRY' + @r +
        'BEGIN CATCH' + @r +
        'IF @@trancount <> 0' + @r +
        'ROLLBACK;' + @r +
        'THROW' + @r +
        'END CATCH;' + @r +
        'GO' as VALUE
FROM #proc_body p
    inner join @list l
        on l.id_tbl = p.id_tbl
end

IF @layer_name_dst = 'ODS.EX'
begin
    -- Добавление поля IS_DEL, если только этого поля уже не существует
    if NOT EXISTS (SELECT * FROM @PRIM WHERE clmn='IS_DEL')
    begin
        insert into @PRIM (id_tbl, clmn, ordr, typ, pk)
        select max(id_tbl) as id_tbl,
            'IS_DEL' as clmn,
            max(ordr)+1 as ordr,
            'BIT' as typ,
            null as pk
        from @PRIM

        insert into @ext_props (id_table, id_column, ep_name, ep_value)
        select id_tbl, ordr, 'MS_Description', 'Признак удаления'
        from @PRIM
        where clmn='IS_DEL'
    end
end

```

```

IF @ods_ex = 'proc'
BEGIN
    UPDATE L
    SET TABLE_DST = 'LOAD_'+REPLACE(l.table_dst,'v_',")
    FROM @LIST L
    WHERE L.table_dst NOT LIKE 'LOAD%'

    INSERT INTO #CREATE_OBJ (
        ID_TABLE,
        SCHEMA_DST,
        TABLE_DST,
        COLUMN_NAME,
        COLUMN_ID,
        TYPE,
        PK,
        VALUE
    )

    select ep.id_table,
           l.schema_dst,
           l.table_dst,
           isnull(p.clmn, ep.ep_name) as c_name,
           p.ordr as column_id,
           p.typ as tp,
           p.pk as pk,
           ep.ep_value as [value]
    from @PRIM p
        right join @ext_props ep
            on p.id_tbl = ep.id_table
            and p.ordr = ep.id_column
        right join @list l
            on l.id_tbl = ep.id_table

```

UNION ALL

```
SELECT t.id_tbl,
       l.schema_dst,
       l.table_dst,
       'INSERT INTO dbo.TBL_LOAD_DATE' as
COLUMN_NAME,
       null as COLUMN_ID,
       'INSERT SCRIPT' as TYPE,
       null as PK,
       'IF NOT EXISTS (SELECT 1 FROM
dbo.TBL_LOAD_DATE ld WHERE ld.[NAME] = ''+ l.schema_dst +'.'+l.table_dst+
'' and ld.[DB_NAME] = "STAGEFD")' + @r +
       'INSERT INTO dbo.TBL_LOAD_DATE ([NAME],
[DB_NAME], RV_FROM, RV_TO)+'@r+
       'VALUES(''+ l.schema_dst +'.'+l.table_dst+',
"STAGEFD", 0x0000000000000000, NULL)'
FROM #tbl_body t
      inner join @list l
      on l.id_tbl = t.id_tbl
```

UNION ALL

```
SELECT
L.id_tbl,
l.schema_dst,
l.table_dst,
l.schema_dst +'.'+l.table_dst as COLUMN_NAME,
null as COLUMN_ID,
'PROCEDURE' as TYPE,
null as PK,
```

```

'CREATE PROCEDURE '+l.schema_dst+'.'+l.table_dst+
'AS'+@r+
'SET NOCOUNT ON;'+@rr+
'DECLARE @RV_FROM BINARY(8)
DECLARE @RV_TO BINARY(8)
SELECT
    @RV_FROM = RV_FROM,
    @RV_TO = RV_TO
FROM ODS.DBO.TBL_LOAD_DATE
WHERE NAME="'+l.schema_dst+'.'+l.table_dst+'"
DROP TABLE IF EXISTS #CHANGES
CREATE TABLE #CHANGES(ID INT);

--TABLE1 - ОСНОВНАЯ ТАБЛИЦА, ID_TABLE1 -
ПЕРВИЧНЫЙ КЛЮЧ ОСН ТАБЛИЦЫ
INSERT INTO #CHANGES
SELECT T1.ID_TABLE1
FROM TABLE1 AS T1
WHERE RV > @RV_FROM AND RV <= @RV_TO
UNION
SELECT T1.ID_TABLE1
FROM TABLE1 AS T1
    INNER JOIN TABLE2 AS T2 ON T1.ID_TABLE1 =
T2.ID_TABLE1
WHERE T2.RV > @RV_FROM AND T2.RV <= @RV_TO
SELECT T1.ID_TABLE1
FROM TABLE1 AS T1
    INNER JOIN TABLE3 AS T3 ON T1.ID_TABLE1 =
T3.ID_TABLE1
WHERE T3.RV > @RV_FROM AND T3.RV <= @RV_TO
UNION
SELECT T1.ID_TABLE1

```

```

FROM TABLE1 AS T1
        INNER JOIN TABLE4 AS T4 ON T1.ID_TABLE1 =
T4.ID_TABLE1
        INNER JOIN TABLE5 AS T5 ON T4.ID_TABLE4 =
T5.ID_TABLE4
WHERE T5.RV > @RV_FROM AND T5.RV <= @RV_TO
OPTION(RECOMPILE)

SELECT
        T1.ID_TABLE1,
        T2.ID_TABLE2,
        T3.ID_TABLE3,
        T5.ID_TABLE5
FROM TABLE1 T1
        INNER JOIN #CHANGES AS C ON C.ID =
T1.ID_TABLE1
        INNER JOIN TABLE2 T2 ON T1.ID_TABLE1 =
T2.ID_TABLE1
        LEFT JOIN TABLE3 T3 ON T1.ID_TABLE1 =
T3.ID_TABLE1
        INNER JOIN TABLE4 T4 ON T1.ID_TABLE1 =
T4.ID_TABLE1
        INNER JOIN TABLE5 T5 ON T4.ID_TABLE4 =
T5.ID_TABLE4
OPTION(RECOMPILE) ' AS VALUE
FROM #TBL_BODY T
        INNER JOIN @LIST L
        ON T.ID_TBL = L.ID_TBL

END
ELSE
BEGIN

```

```

INSERT INTO #CREATE_OBJ (
    ID_TABLE,
    SCHEMA_DST,
    TABLE_DST,
    COLUMN_NAME,
    COLUMN_ID,
    TYPE,
    PK,
    VALUE)

select ep.id_table,
       l.schema_dst,
       l.table_dst,
       isnull(p.clmn, ep.ep_name) as c_name,
       p.ordr as column_id,
       p.typ as tp,
       p.pk as pk,
       ep.ep_value as [value]
from @PRIM p
      right join @ext_props ep
        on p.id_tbl = ep.id_table
       and p.ordr = ep.id_column
      right join @list l
        on l.id_tbl = ep.id_table

-- Если приходит инкремент, то добавляем записи в TBL_LOAD_DATE и создаем
-- инкрементальное представление
if exists (select l.is_increment
           FROM #tbl_body t
          inner join @list l
            on l.id_tbl = t.id_tbl
          where l.is_increment = 1 )

BEGIN

```

```

INSERT INTO #CREATE_OBJ (
    ID_TABLE,
    SCHEMA_DST,
    TABLE_DST,
    COLUMN_NAME,
    COLUMN_ID,
    TYPE,
    PK,
    VALUE)

SELECT t.id_tbl,
    l.schema_dst,
    l.table_dst,
    'INSERT INTO dbo.TBL_LOAD_DATE' as
COLUMN_NAME,
    null as COLUMN_ID,
    'INSERT SCRIPT' as TYPE,
    null as PK,
    'IF NOT EXISTS (SELECT 1 FROM
dbo.TBL_LOAD_DATE ld WHERE ld.[NAME] = ''+ l.schema_dst + '.'+ l.table_dst +
    '' and ld.[DB_NAME] = "STAGEFD")' + @r +
    'INSERT INTO dbo.TBL_LOAD_DATE ([NAME],
[DB_NAME], RV_FROM, RV_TO)'+@r+
    'VALUES(''+ l.schema_dst + '.'+ l.table_dst + ',
"STAGEFD", 0x0000000000000000, NULL)'
FROM #tbl_body t
    inner join @list l
        on l.id_tbl = t.id_tbl

UNION ALL

SELECT t.id_tbl,

```



```

        l.schema_dst,
        l.table_dst,
        l.schema_dst+'.'+l.table_dst+'_increment' as
COLUMN_NAME,

        null as COLUMN_ID,
        'CREATE SCRIPT' as TYPE,
        null as PK,
        @rr+'CREATE VIEW

'+l.schema_dst+'.'+l.table_dst+'_increment'+@r+
        'AS'+replace(replace(stuff(t.v_inc,1,10,''), '<','>'), '>','<')
        +@r +
        'NOT EXISTS (SELECT 1 FROM '+ l.schema_src
        +'.'+l.table_src+
        ' t WHERE'+STUFF(t.c,1,4, ''))+@rr+
        'GO' [view]
FROM #tbl_body t
        inner join @list l
        on l.id_tbl = t.id_tbl

END
ELSE
BEGIN
        INSERT INTO #CREATE_OBJ (
                ID_TABLE,
                SCHEMA_DST,
                TABLE_DST,
                COLUMN_NAME,
                COLUMN_ID,
                TYPE,
                PK,
                VALUE)
        SELECT t.id_tbl,

```

```

        l.schema_dst,
        l.table_dst,
        l.schema_dst+'.'+l.table_dst as COLUMN_NAME,
        null as COLUMN_ID,
        'CREATE SCRIPT' as TYPE,
        null as PK,
        @rr+'CREATE VIEW

'+l.schema_dst+'.'+l.table_dst+@r+
        'AS'+stuff(t.v_full,1,10,'')+ @r +
        'NOT EXISTS (SELECT 1 FROM '+
l.schema_src+'.'+l.table_src+
        ' t WHERE'+STUFF(t.c,1,4, ''')+@rr+
        'GO' [view]

FROM #tbl_body t
        inner join @list l
        on l.id_tbl = t.id_tbl

END

END

end

```

```

IF @layer_name_dst = 'STAGEFD'
BEGIN
-- Добавление поля ID_CONNECT_CODE
        if NOT EXISTS (SELECT * FROM @PRIM WHERE
clmn='ID_CONNECT_CODE')
        begin
                insert into @PRIM (id_tbl, clmn, ord, typ, pk)
                select max(id_tbl) as id_tbl,
                        'ID_CONNECT_CODE' as clmn,
                        max(ord)+1 as ord,
                        'SMALLINT' as typ, null as pk

```

```

from @PRIM

insert into @ext_props (id_table, id_column, ep_name, ep_value)
select id_tbl, ordr, 'MS_Description', 'Код исходной информационной
системы'
from @PRIM
where clmn='ID_CONNECT_CODE'
end

```

```

INSERT INTO #CREATE_OBJ (
                                ID_TABLE,
                                SCHEMA_DST,
                                TABLE_DST,
                                COLUMN_NAME,
                                COLUMN_ID,
                                TYPE,
                                PK,
                                VALUE
                                )

select ep.id_table,
       l.schema_dst,
       l.table_dst,
       isnull(p.clmn, ep.ep_name) as c_name,
       p.ordr as column_id,
       p.typ as tp,
       p.pk as pk,
       ep.ep_value as [value]
from @PRIM p
right join @ext_props ep
on p.id_tbl = ep.id_table
and p.ordr = ep.id_column

```

```

        right join @list l
        on l.id_tbl = ep.id_table

WHERE ISNULL(ep.ep_value, '') NOT LIKE '%rowversion%'

END

IF @layer_name_dst = 'CALCFD'
BEGIN

    UPDATE P
    SET P.PK = CAST(NULL AS INT)
    FROM @PRIM P
    WHERE P.PK > 0

-- Добавление поля являющегося первичным ключом в КИХФД и поля RV
INSERT INTO @PRIM (id_tbl, clmn, ord, typ, pk)
SELECT max(P.id_tbl) as id_tbl,
        'ID_'+REPLACE(max(L.table_dst), 'DIC_', '') as clmn,
        max(ord)+1 as ord,
        'BIGINT' as typ,
        1 as pk
FROM @PRIM P
        INNER JOIN @LIST L ON P.id_tbl = L.id_tbl
UNION ALL
SELECT max(id_tbl) as id_tbl,
        'RV' as clmn,
        max(ord)+2 as ord,
        'rowversion' as typ,
        null as pk
FROM @PRIM

INSERT INTO @ext_props (id_table, id_column, ep_name, ep_value)
SELECT id_tbl, ord, 'MS_Description', 'Идентификатор в КИХФД'

```

```

FROM @PRIM
WHERE clmn LIKE 'ID%' AND PK = 1
UNION ALL
SELECT id_tbl, ordr, 'MS_Description', 'Rowversion (механизм для отметки
версий строк таблицы)'
FROM @PRIM
WHERE clmn='RV'

INSERT INTO #CREATE_OBJ (

                                ID_TABLE,
                                SCHEMA_DST,
                                TABLE_DST,
                                COLUMN_NAME,
                                COLUMN_ID,
                                TYPE,
                                PK,
                                VALUE
                                )

select ep.id_table,
       l.schema_dst,
       l.table_dst,
       isnull(p.clmn, ep.ep_name) as c_name,
       p.ordr as column_id,
       p.typ as tp,
       p.pk as pk,
       ep.ep_value as [value]
from @PRIM p
      right join @ext_props ep
      on p.id_tbl = ep.id_table
      and p.ordr = ep.id_column
      right join @list l

```

```

        on l.id_tbl = ep.id_table
where ISNULL(p.clmn, "") <> 'IS_DEL'

UNION ALL
-- Скрипт для создания процедуры загрузки
SELECT p.id_tbl,
       l.schema_dst,
       l.table_dst,
       'etl.LOAD_' + l.table_dst as COLUMN_NAME,
       null as COLUMN_ID,
       'PROCEDURE' as TYPE,
       null as PK,
       'CREATE PROCEDURE etl.LOAD_' + l.table_dst + @r +
       'AS' + @r +
       'SET NOCOUNT ON;' + @rr +
       'DROP TABLE IF EXISTS #src' + @rr +
       'CREATE TABLE
#src' + @r + '(' + UPPER(STUFF(p.crt, 1, 2, '')) + IIF(p.is_increment = 1, @r +
',IS_DEL' + @t + 'BIT', '') + @r + ')' + @rr +
       'INSERT INTO #src (' + STUFF(p.ins, 1, 9, '') + @r +
       'FROM ' + l.schema_src + '.' + l.table_src + ' s' + @rr +
       'BEGIN TRY' + @r + 'BEGIN TRANSACTION' + @rr +
       'MERGE ' + p.schema_dst + '.' + p.table_dst + ' t' + @r +
       'USING #SRC s' + @r +
       'ON ' + STUFF(p.mpk, 1, 5, '') + @r +
       iif(p.u is null, '', 'WHEN MATCHED AND
EXISTS' + @r + '(' + STUFF(p.e, 1, 8, '') + @r + ')' + IIF(p.is_increment = 0, '', @r + 'AND
s.IS_DEL = 0') + @r + -- если нет не ключевых полей, то выкидываем UPDATE
       'THEN UPDATE SET' + @r + STUFF(p.u, 1, 2, '') + @r +
       'WHEN NOT MATCHED' + IIF(p.is_increment = 0, '', ' AND
s.IS_DEL = 0') + ' THEN' + @r + 'INSERT(' + STUFF(p.i, 1, 9, '') + @r +

```

```

        IIF(p.is_increment = 0, 'WHEN NOT MATCHED BY SOURCE',
'WHEN MATCHED AND s.IS_DEL = 1') + @r + 'THEN DELETE' + @r + ';' + @rr +
        'UPDATE dbo.TBL_LOAD_DATE' + @r +
        'SET DATE_FROM = DATE_TO,' + @r +
        'DATE_TO = NULL' + @r +
        'WHERE [NAME] = ''' + l.schema_src + '.' + l.table_src + ''' AND
DATE_TO IS NOT NULL' + @rr +
        'COMMIT TRANSACTION' + @rr +
        'END TRY' + @r +
        'BEGIN CATCH' + @r +
        'IF @@trancount <> 0' + @r +
        'ROLLBACK;' + @r +
        'THROW' + @r +
        'END CATCH;' + @r +
        'GO' as VALUE

```

```

FROM #proc_body p
    inner join @list l
        on l.id_tbl = p.id_tbl

```

UNION ALL

-- Скрипт для создания CONSTRAINT по первичным ключам

```

SELECT l.id_tbl,
        l.schema_dst,
        l.table_dst,
        'PK_' + l.schema_dst + '_' + l.table_dst + REPLACE(x.pk, ',', '_') as
COLUMN_NAME,
        null as COLUMN_ID,
        'CONSTRAINT' as TYPE,
        null as PK,
        isnull('ALTER TABLE ' + l.schema_dst + '.' + l.table_dst + ' ADD
CONSTRAINT PK_' + l.schema_dst + '_' + l.table_dst + REPLACE(x.pk, ',', '_') + '

```

```
PRIMARY KEY CLUSTERED ('+STUFF(x.pk,1,1,')+'),'-- PRIMARY KEY
NEEDED!')+ @r+';' + @r + 'GO' as VALUE
```

```
FROM @list l
```

```
OUTER APPLY (SELECT '+'p.clmn
```

```
FROM @PRIM p
```

```
WHERE p.id_tbl = l.id_tbl
```

```
AND p.pk > 0
```

```
ORDER BY p.pk FOR XML PATH(''
```

```
) x(pk)
```

```
UNION ALL
```

```
-- Скрипт для создания индекса по RV
```

```
SELECT t.id_tbl,
```

```
l.schema_dst,
```

```
l.table_dst,
```

```
'UIX_'+l.schema_dst+'_'+l.table_dst+'_RV' as COLUMN_NAME,
```

```
null as COLUMN_ID,
```

```
'INDEX' as TYPE,
```

```
null as PK,
```

```
'CREATE UNIQUE NONCLUSTERED INDEX
```

```
UIX_'+l.schema_dst+'_'+l.table_dst+REPLACE(T.PK,',','_')+ ' ON
```

```
'+l.schema_dst+'_'+l.table_dst + ' ('+STUFF(T.PK,1,1,NULL)+')' + @r +
```

```
'GO' as VALUE
```

```
FROM #tbl_body t
```

```
inner join @list l
```

```
on l.id_tbl = t.id_tbl
```

```
END
```

```
IF @layer_name_dst = 'OLAP+KIHFD'
```

```
BEGIN
```

```
INSERT INTO #CREATE_OBJ (
```

```
ID_TABLE,
```



```

        SCHEMA_DST,
        TABLE_DST,
        COLUMN_NAME,
        COLUMN_ID,
        TYPE,
        PK,
        VALUE
    )

SELECT ep.id_table,
       l.schema_dst,
       l.table_dst,
       isnull(p.clmn, ep.ep_name) as c_name,
       p.ordr as column_id,
       p.typ as tp,
       p.pk as pk,
       ep.ep_value as [value]
FROM @PRIM p
      RIGHT JOIN @ext_props ep
        ON p.id_tbl = ep.id_table
        AND p.ordr = ep.id_column
      RIGHT JOIN @list l
        ON l.id_tbl = ep.id_table
WHERE ISNULL(ep.ep_value, '') NOT LIKE '%rowversion%'
UNION ALL
SELECT l.ID_TBL,
       l.SCHEMA_DST,
       l.TABLE_DST,
       'KIHFD.'+l.SCHEMA_DST+'.'+l.TABLE_DST as
COLUMN_NAME,
       null as COLUMN_ID,
       'CREATE SCRIPT' as COLUMN_TYPE, null as PK,

```

```

'CREATE VIEW '+l.SCHEMA_DST+'.'+l.TABLE_DST+ @r+
'AS'+ @r+
'SELECT '+STUFF(x.clms,1,1,")+ @r+
'FROM OLAP.'+l.SCHEMA_DST+'.'+l.TABLE_DST+@r+
'GO' AS VALUE
FROM @LIST 1
CROSS APPLY
(
    SELECT '+'p.CLMN
    FROM @PRIM p
    WHERE p.ID_TBL = l.ID_TBL
    ORDER BY p.ORDR FOR XML PATH("")
) x(clms)

IF @sync_type = 'SDS'
BEGIN
    INSERT INTO #CREATE_OBJ (
                                ID_TABLE,
                                SCHEMA_DST,
                                TABLE_DST,
                                COLUMN_NAME,
                                COLUMN_ID,
                                TYPE,
                                PK,
                                VALUE
                                )

```

-- Скрипт для создания триггера на таблице на калке

```

SELECT l.id_tbl,
       l.schema_dst,
       l.table_dst,
       'TR_'+l.table_src+'_SDS' as COLUMN_NAME,
       null as COLUMN_ID, 'TRIGGER' as TYPE,

```

```

null as PK,
'CREATE TRIGGER
'+l.schema_src+'.'+'TR_'+l.table_src+'_SDS ON '+l.schema_src+'.'+l.table_src+'@r+
'WITH EXECUTE AS "FDAdmin"'+'@r+
'AFTER INSERT, DELETE, UPDATE'+'@r+'AS'+'@r+
'SET NOCOUNT ON;'+'@rr+
'declare @inserted xml = ('+'@r+
'select '+STUFF(x.clms,1,1,')+'@r+
'from inserted for xml raw("ins"),XMLSCHEMA, BINARY
BASE64,type),'+'@r+
'@deleted xml = ('+'@r+
'select '+STUFF(x.clms,1,1,')+'@r+
'from deleted for xml raw("del"),XMLSCHEMA, BINARY
BASE64,type),'+'@r+
'@DB_NAME sysname = db_name();'+'@rr+
'exec
UTILITY.sds.PUBLISH_DICTIONARY_CHANGES'+'@r+
'@OBJECT_ID = @@PROCID,'+'@r+
'@DB_NAME = @DB_NAME,'+'@r+
'@INS = @inserted,'+'@r+
'@DEL = @deleted;' as VALUE
FROM @list l
CROSS APPLY
(
    SELECT '+'p.clmn
    FROM @PRIM p
    WHERE p.id_tbl = l.id_tbl
    ORDER BY p.ordr FOR XML PATH("")
) x(clms)
UNION ALL
-- Скрипт для добавления записей в
SELECT l.id_tbl, l.schema_dst,

```

```

l.table_dst,
INSERT INTO
sds.DIC_PUBLICATIONS_sds.DIC_SUBSCRIPTIONS' as COLUMN_NAME,
null as COLUMN_ID,
INSERT SCRIPT' as TYPE,
null as PK,
declare
    @SERVER_NAME_SRC sysname = "MASERATI",
    @DATABASE_NAME_SRC sysname = "CALCFD",
    @TABLE_SCHEMA_NAME_SRC sysname =
""+l.schema_src+",
    @TABLE_NAME_SRC sysname = ""+l.table_src+",

    @SERVER_NAME_TRG sysname = "LEXUS",
    @DATABASE_NAME_TRG sysname = "OLAP",
    @TABLE_SCHEMA_NAME_TRG sysname =
""+l.schema_dst+",
    @TABLE_NAME_TRG sysname = ""+l.table_dst+"

declare @ID_PUBLICATION int;
declare @ID_SERVER_ROUTE tinyint;
set @ID_PUBLICATION =(
    select ID_PUBLICATION
    from sds.DIC_PUBLICATIONS
    where DATABASE_NAME =
@DATABASE_NAME_SRC
and TABLE_SCHEMA_NAME =
@TABLE_SCHEMA_NAME_SRC
and TABLE_NAME =
@TABLE_NAME_SRC
);

```

```

        if @ID_PUBLICATION is null
        begin
            exec [sds].[DIC_PUBLICATIONS_INSERT]
                @DATABASE_NAME =
@DATABASE_NAME_SRC
                ,@TABLE_SCHEMA_NAME =
@TABLE_SCHEMA_NAME_SRC
                ,@TABLE_NAME =
@TABLE_NAME_SRC;

            set @ID_PUBLICATION =(
                select ID_PUBLICATION
                from sds.DIC_PUBLICATIONS
                where DATABASE_NAME =
@DATABASE_NAME_SRC
                    and TABLE_SCHEMA_NAME =
@TABLE_SCHEMA_NAME_SRC
                    and TABLE_NAME =
@TABLE_NAME_SRC
            );
        end

        set @ID_SERVER_ROUTE = (
            select ID_SERVER_ROUTE
            from sds.DIC_SERVER_ROUTES
            where SERVER_ROUTE_NAME =
@SERVER_NAME_TRG
        );

        if not exists(
            select * from sds.DIC_SUBSCRIPTIONS
            where ID_PUBLICATION =
@ID_PUBLICATION
                and DATABASE_NAME =

```

```

@DATABASE_NAME_TRG
                                and OBJ_SCH_NAME =
@TABLE_SCHEMA_NAME_TRG
                                and OBJ_NAME = @TABLE_NAME_TRG
                                and ID_OBJECT_TYPE = 1
                                )
                                begin
                                exec [sds].[DIC_SUBSCRIPTIONS_INSERT]
                                    @ID_PUBLICATION =
@ID_PUBLICATION
                                    ,@ID_SERVER_ROUTE =
@ID_SERVER_ROUTE
                                    ,@DATABASE_NAME =
@DATABASE_NAME_TRG
                                    ,@OBJ_SCH_NAME =
@TABLE_SCHEMA_NAME_TRG
                                    ,@OBJ_NAME =
@TABLE_NAME_TRG
                                    ,@ID_OBJECT_TYPE = 1;
                                end
                                GO' as VALUE
                                FROM @list l
END
ELSE
BEGIN
    INSERT INTO #CREATE_OBJ (
                                ID_TABLE,
                                SCHEMA_DST,
                                TABLE_DST,
                                COLUMN_NAME,
                                COLUMN_ID,

```

TYPE,
PK,
VALUE)

-- Скрипт для создания триггера на удаление

```
SELECT l.id_tbl,
       l.schema_dst,
       l.table_dst,
       l.schema_src+'.'+l.table_src+'_DEL' as COLUMN_NAME,
       null as COLUMN_ID,
       'TRIGGER' as TYPE,
       null as PK,
       'CREATE TRIGGER '+l.schema_src+'.'+l.table_src+'_DEL
ON '+l.schema_src+'.'+l.table_src+'@r+
   'FOR DELETE'+@r+'AS'+@r+
   'SET NOCOUNT ON;'+@rr+
   'DELETE td'+@r+
   'FROM deleted d'+@r+
   'JOIN dbo.TBL_DELETED_OBJECTS td on td.ID=d.ID and
td.[OBJECT_NAME]=''+ l.table_src +'''+@rr+
   'insert
dbo.TBL_DELETED_OBJECTS([OBJECT_NAME],ID,LASTDATE)'+@r+
   'select '''+l.table_src +'''+',d.ID, getdate()'+@r+
   'from deleted d;'+@rr+
   'update dbo.TBL_PUBLICATIONS'+@r+
   'set RV_LAST=@ @DBTS'+@r+
   'WHERE
OBJECT_PATH="CALCFD.'+l.schema_src+'.'+l.table_src+'''+@r+
   'AND ID_EVENT_TYPE = 4;'+@rr+
   'GO' as VALUE
FROM @list l
UNION ALL
```

-- Скрипт для создания триггера на update, insert

```
SELECT l.id_tbl,
       l.schema_dst,
       l.table_dst,
       l.schema_src+'.'+l.table_src+'_CT' as COLUMN_NAME,
       null as COLUMN_ID,
       'TRIGGER' as TYPE,
       null as PK,
       'CREATE TRIGGER '+l.schema_src+'.'+l.table_src+'_CT ON
'+l.schema_src+'.'+l.table_src+'@r+
       'FOR INSERT, UPDATE'+@rr+'AS'+@rr+'SET NOCOUNT
ON'+@rr+
       'update dbo.TBL_PUBLICATIONS'+@r+
       'set RV_LAST=@ @DBTS'+@r+
       'WHERE
OBJECT_PATH="CALCFD.'+l.schema_src+'.'+l.table_src+'"'@r+
       'AND ID_EVENT_TYPE = 4;'@r+
       'GO' as VALUE

FROM @list L
UNION ALL
```

-- Скрипт для добавления записей в TBL_PUBLICATIONS и

-- TBL_SUBSCRIPTIONS

```
SELECT l.id_tbl,
       l.schema_dst,
       l.table_dst,
       'INSERT INTO
dbo.TBL_PUBLICATIONS_dbo.TBL_SUBSCRIPTIONS' as COLUMN_NAME,
       null as COLUMN_ID,
       'INSERT SCRIPT' as TYPE,
       null as PK,
```



```

        'IF NOT EXISTS (SELECT 1 FROM
UTILITY.dbo.TBL_PUBLICATIONS WHERE
OBJECT_PATH="CALCFD.'+l.schema_src+'.'+l.table_src+'"')'+@r+
        'BEGIN'+@r+
        'INSERT INTO UTILITY.dbo.TBL_PUBLICATIONS
(ID_EVENT_TYPE, OBJECT_PATH, LAST_CHANGE_DATE)'+@r+
        'VALUES (4, "CALCFD.'+l.schema_src+'.'+l.table_src+',
"19000101")'+@r+
        'END'+@r+
        'IF NOT EXISTS (SELECT 1 FROM
UTILITY.dbo.TBL_SUBSCRIPTIONS WHERE SYNC_PROC
="OLAP.sync.LOAD_'+l.table_src+'"')'+@r+
        'BEGIN'+@r+
        'DECLARE @ID_PUBLICATION INT=(SELECT
ID_PUBLICATION FROM UTILITY.dbo.TBL_PUBLICATIONS WHERE
OBJECT_PATH="CALCFD.'+l.schema_src+'.'+l.table_src+'"')'+@r+
        'INSERT INTO UTILITY.dbo.TBL_SUBSCRIPTIONS
(ID_PUBLICATION,SERVER_NAME,SYNC_PROC,
LAST_SYNC_DATE,IS_ACTIVE)'+@r+
        'VALUES (@ID_PUBLICATION,"KIHFD-
DB","OLAP.sync.LOAD_'+l.table_src+'", "19000101",1)'+@r+
        'END' as VALUE
FROM @list L
-- Скрипт процедуры синхронизации
SELECT l.id_tbl,
       l.schema_dst,
       l.table_dst,
       'sync.LOAD_'+l.table_src as COLUMN_NAME,
       null as COLUMN_ID,
       'PROCEDURE' as TYPE,
       null as PK,

```

```

CREATE PROCEDURE
sync.LOAD_'+l.table_src+'@r+'AS'+@r+'BEGIN'+@r+'SET NOCOUNT ON;'+@r+
    DECLARE @RV_LAST BINARY(8),
    @RV_PUBLICATION BINARY(8)+'@r+
    SELECT @RV_LAST =
ISNULL(RV,0x0),'+@r+'@RV_PUBLICATION =
ISNULL(RV_PUBLICATION,0x0)+'@r+
    FROM sync.DICTIONARY_LASTDATE'+@r+
    WHERE DICTIONARY_NAME = '"+L.table_src+"'"+@r+
    DROP TABLE IF EXISTS #CHANGED'+@r+
    SELECT '+STUFF(x.clms,1,1,")+'@r+
    INTO #CHANGED'+@r+
    FROM [CALCFD-
DB].CALCFD.'+l.schema_src+'.'+l.table_src+'@r+
    WHERE RV > @RV_LAST AND RV <=
@RV_PUBLICATION'+@rr+
    DROP TABLE IF EXISTS #DELETED'+@r+
    SELECT OBJECT_NAME, ID'+@r+'INTO
#DELETED'+@r+
    FROM [CALCFD-
DB].[CALCFD].[dbo].[TBL_DELETED_OBJECTS]'+@r+
    WHERE OBJECT_NAME = '"+L.table_src+"'"+@r+' AND
RV > @RV_LAST and RV <= @RV_PUBLICATION;'+@rr+
    UPDATE T'+@r+'SET'+@r+' STUFF(P.u,1,1,")+'@r+
    FROM '+l.schema_dst+'.'+l.table_dst+'@r+' AS T'+@r+
    INNER JOIN #CHANGED S ON '+STUFF(P.mpk,1,4,")
+@rr+
    INSERT '+l.schema_dst+'.'+l.table_dst+'
('+STUFF(x.clms,1,1,")+'@r+
    SELECT '+STUFF(x.clms,1,1,")+'@r+'FROM #CHANGED
T'+@r+

```

```

'LEFT JOIN '+l.schema_dst+'.'+l.table_dst+' AS D'+@r+'ON
'+STUFF(P.mpk,1,4,"")+@rr+
'WHERE '+STUFF(xxx.pk,1,4,"")+@r+
'IF EXISTS (SELECT TOP 1 1 FROM #DELETED)'+@r+
'DELETE T'+@r+'FROM #DELETED D'+@r+
'INNER JOIN '+l.schema_dst+'.'+l.table_dst+' AS T'+@r+'ON
'+STUFF(P.mpk,1,4,"")+@rr+
'IF EXISTS (SELECT TOP 1 1 FROM
#CHANGED)'+@r+'BEGIN'+@r+
'UPDATE sync.DICTIONARY_LASTDATE'+@r+'SET RV
= @RV_PUBLICATION'+@r+
'WHERE DICTIONARY_NAME
='+L.table_src+''''+@r+'END'+@rr+
" as VALUE
FROM @list L
CROSS APPLY ( SELECT '+p.clmn
FROM @PRIM p
WHERE p.id_tbl = l.id_tbl
ORDER BY p.ordr FOR XML PATH("")
) x(clms)
CROSS APPLY (
SELECT ' AND D.'+p.clmn+' IS NULL'
FROM @PRIM p
WHERE p.id_tbl = l.id_tbl
AND p.pk > 0
ORDER BY p.pk FOR XML PATH("")
) xxx(pk)
INNER JOIN #proc_body P
ON P.id_tbl = L.id_tbl
END
END

```

```
SELECT
ID_CREATE_OBJ
, ID_TABLE
, SCHEMA_DST
, TABLE_DST
, COLUMN_NAME
, COLUMN_ID
, TYPE
, PK
, VALUE
FROM #CREATE_OBJ
order by column_id
end
```

ПРИЛОЖЕНИЕ Б

(обязательное)

Результаты выполнения программы

Главное окно программы представлено на рисунке Б.1.

Рисунок Б.1 – Главное окно программы

На рисунке Б.2 продемонстрировано отображение метаданных начальной таблицы из слоя ODS.STG.

Рисунок Б.2 – Отображение метаданных начальной таблицы

На рисунке Б.3 продемонстрирован результат работы программы на слое ODS.RAW.

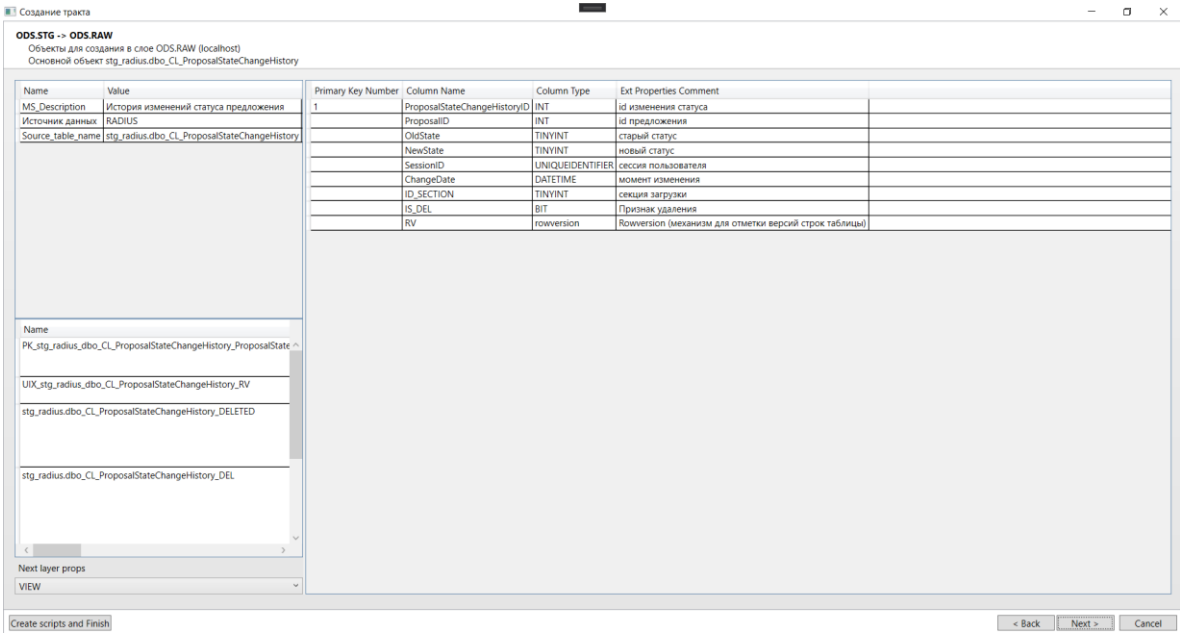


Рисунок Б.3 – Результат работы программы на слое ODS.RAW

На рисунке Б.4 продемонстрирован результат работы программы на слое ODS.EX.

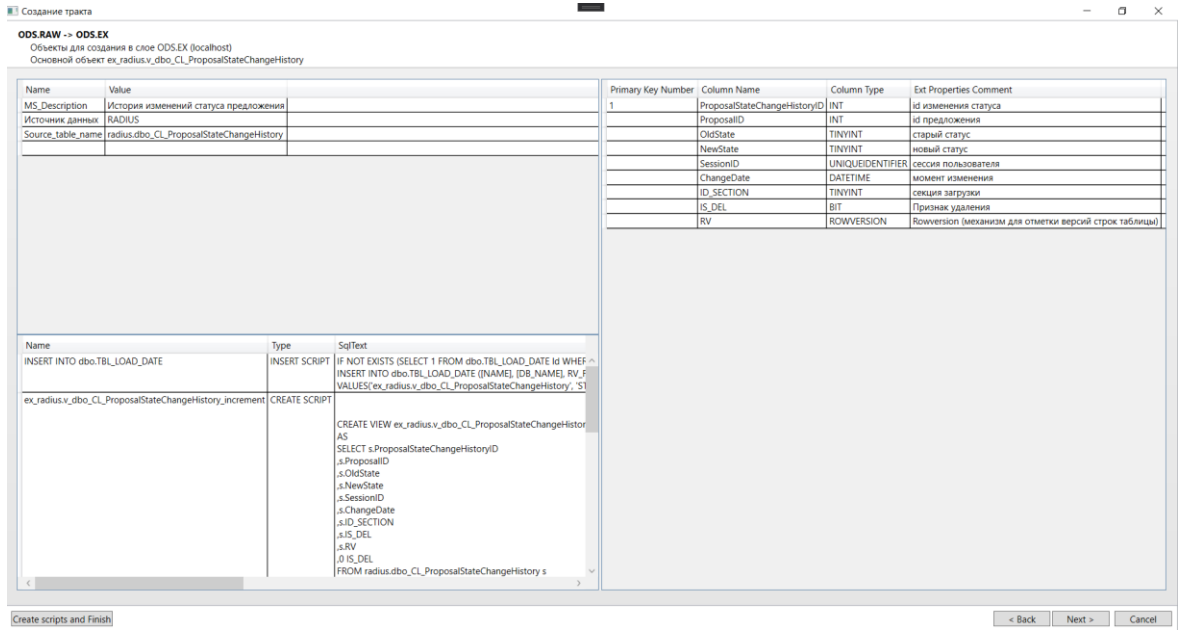


Рисунок Б.4 – Результат работы программы на слое ODS.EX

На рисунке Б.5 продемонстрирован результат работы программы на слое STAGEFD.

Name	Value	Primary Key Number	Column Name	Column Type	Ext Properties	Comment
MS_Description	История изменений статуса предложения	1	ProposalStateChangeHistoryID	INT		ид изменения статуса
Источник данных	RADIUS		ProposalID	INT		ид предложения
Source_table_name	ex.radius.v.dbo.CL_ProposalStateChangeHistory		OldState	TINYINT		старый статус
			NewState	TINYINT		новый статус
			SessionID	UNIQUEIDENTIFIER		сессия пользователя
			ChangeDate	DATETIME		момент изменения
			ID_SECTION	TINYINT		секция загрузки
			IS_DEL	BIT		Признак удаления
			ID_CONNECT_CODE	SMALLINT		Код исходной информационной системы

Рисунок Б.5 – Результат работы программы на слое STAGEFD

На рисунке Б.6 продемонстрирован результат работы программы на слое CALCFD.

Name	Value	Primary Key Number	Column Name	Column Type	Ext Properties	Comment
MS_Description	История изменений статуса предложения		ID_CL_ProposalStateChangeHistory_SYSTEM	INT		ид изменения статуса
Источник данных	RADIUS		ProposalID	INT		ид предложения
Source_table_name	radius.CL_ProposalStateChangeHistory		OldState	TINYINT		старый статус
			NewState	TINYINT		новый статус
			SessionID	UNIQUEIDENTIFIER		сессия пользователя
			ChangeDate	DATETIME		момент изменения
			ID_SECTION	TINYINT		секция загрузки
			ID_CONNECT_CODE	SMALLINT		Код исходной информационной системы
		1	ID_TBL_CL_ProposalStateChangeHistory	BIGINT		Идентификатор в KИFD
			Rowversion	Rowversion		Rowversion (механизм для отметки версий строк таблицы)

Рисунок Б.6 – Результат работы программы на слое CALCFD

На рисунке Б.7 продемонстрирован результат работы программы на слое OLAP + KIHFD.

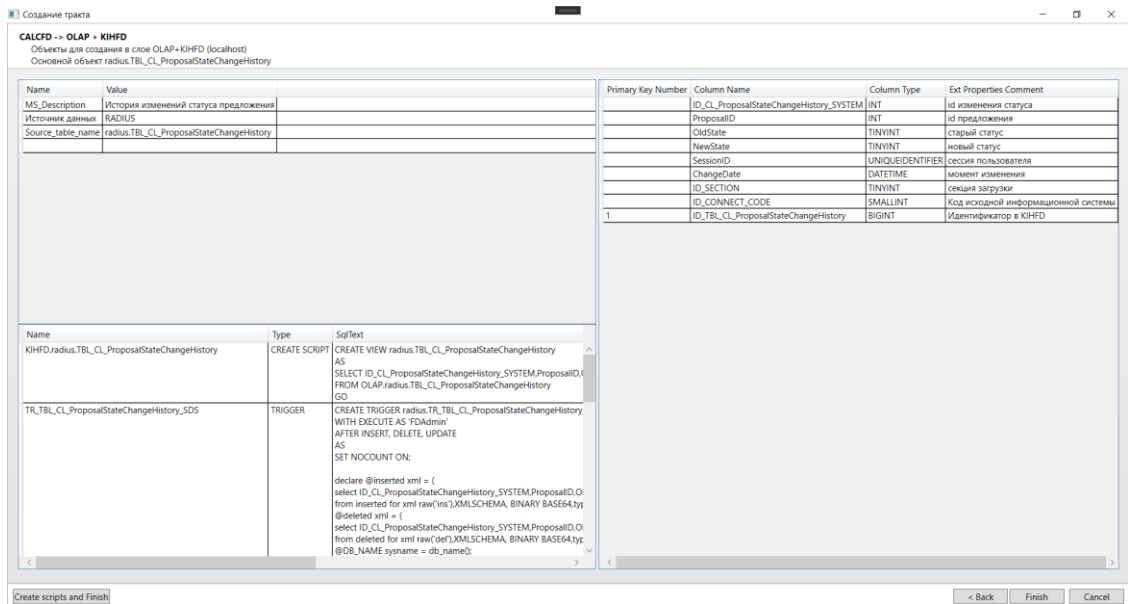


Рисунок Б.7 – Результат работы программы на слое OLAP + KIHFD

На рисунке Б.8 продемонстрировано окно правки и редактирования сформированных скриптов.

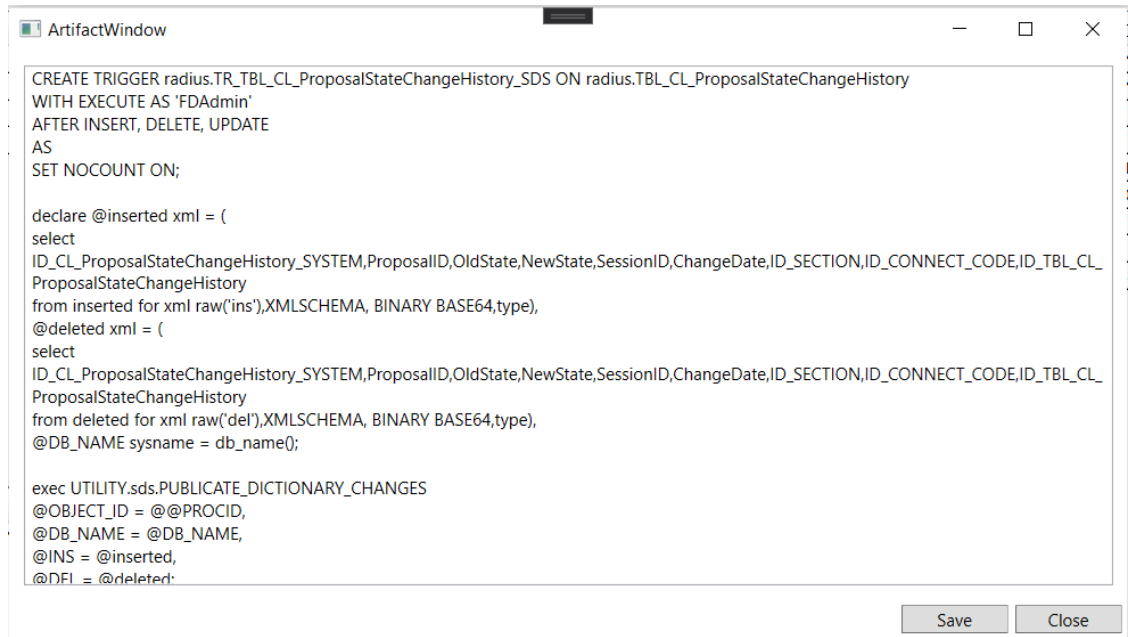


Рисунок Б.8 – Окно правки и редактирования скриптов

На рисунках Б.9 и Б.10 продемонстрирован результат сохранения сформированных скриптов в выбранную директорию.

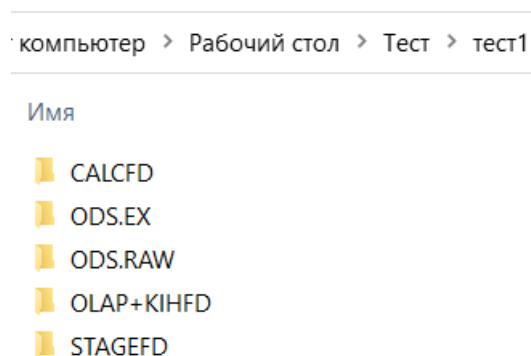


Рисунок Б.9 – Отображение папок в выбранной директории

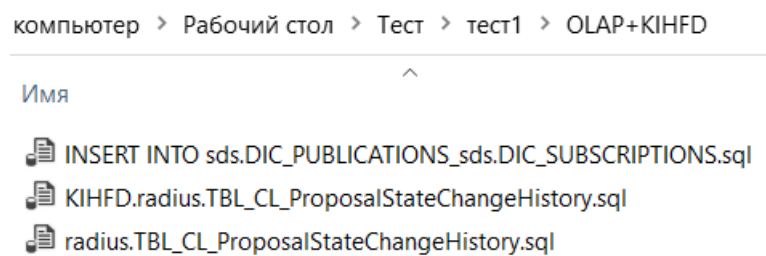


Рисунок Б.10 – Отображение сохраненных файлов