

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное
учреждение высшего образования

«Южно-Уральский государственный университет»
(национальный исследовательский университет)

Высшая школа экономики и управления

Кафедра «Информационные технологии в экономике»

РАБОТА ПРОВЕРЕНА

Рецензент, начальник отдела прикладного
ПО ПАО «АСКО-СТРАХОВАНИЕ»

_____/ И.В. Косов /

« ____ » _____ 2020 г.

ДОПУСТИТЬ К ЗАЩИТЕ

Заведующий кафедрой,

д.т.н., с.н.с.

_____/ Б.М. Суховилов /

« ____ » _____ 2020 г.

ПРИМЕНЕНИЕ OLAP-ТЕХНОЛОГИЙ ДЛЯ УЧЕТНЫХ СИСТЕМ НА
ПЛАТФОРМЕ 1С

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

ЮУрГУ – 09.04.03.2020.301-461.ВКР

Руководитель, к.т.н., доцент

_____/ В.А. Конов /

« ____ » _____ 2020 г.

Автор

студент группы ЭУ-220

_____/ В.В. Симонов /

« ____ » _____ 2020 г.

Нормоконтролер,

старший преподаватель

_____/ Е.Н. Горных /

« ____ » _____ 2020 г.

Челябинск 2020

АННОТАЦИЯ

Симонов В.В. «Применение OLAP-технологий для учетных систем на платформе 1С». – Челябинск: ЮУрГУ, ЭУ-220, 88 с., 46 ил., 11 табл., библиографический список – 20 наим.

Целью выпускной квалификационной работы является выявление наиболее эффективного метода интеграции учетных систем на платформе 1С с OLAP-системами для построения управленческих отчетов.

В работе поставлены задачи: анализ предметной области, анализ и описание существующих решений, проектирование и реализация OLAP-системы.

Разработанная система выполняет извлечение, обработку и загрузку данных из базы 1С, затрачивая на это минимум времени, разворачивает OLAP-куб и тем самым сокращает время построения аналитических и управленческих отчетов. Так же позволяет бизнес-пользователя создавать отчеты в различных визуализациях, а аналитикам выполнять статистический анализ данных.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ	5
1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ. ПОСТАНОВКА ЗАДАЧИ.....	7
1.1 Предметная область	7
1.2 Постановка требований к OLAP-системе	10
1.3 Описание и анализ существующих решений.....	11
1.3.1 Внешние соединения к платформе 1С	12
1.3.2 Представления в MSSQLServer	12
1.3.3 Web-сервисы и HTTP-сервисы на платформе 1С	15
1.3.4. MSPowerBI и стандартный интерфейс OData платформы 1С	16
Вывод по разделу один	18
2 ПРОЕКТИРОВАНИЕ OLAP-СИСТЕМЫ.....	19
2.1 Выбор и описание технических решений	19
2.2.1 Технология OData.....	19
2.2.2 СУБД MicrosoftSQLServer	20
2.2.3 SQLServerIntegrationServices	20
2.2.4 SQLServerAnalysisServices	20
2.2.5 Веб-сервер Apache	21
2.2.6 MicrosoftPowerBI	21
2.3 Проектирование архитектуры системы	22
2.4 Проектирование хранилища данных.....	23
Вывод по разделу два.....	31
3 РЕАЛИЗАЦИЯ И ТЕСТИРОВАНИЕ OLAP-СИСТЕМЫ.....	32
3.1 Создание хранилища данных	32
3.2 Исследование длительности ETL-процесса существующих решений.....	37
3.3 Публикация интерфейса OData на веб-сервере	45
3.4 Разработка ETL-приложения	46
3.5 Тестирование ETL-приложения	61

3.6Разработка OLAP-куба на основании хранилища данных.....	64
3.7 Разработка аналитических отчетов в приложении Microsoft Power BI.....	66
3.8 Сравнение всех решений	69
Вывод по разделу три	70
ЗАКЛЮЧЕНИЕ	71
БИБЛИОГРАФИЧЕСКИЙ СПИСОК	72
ПРИЛОЖЕНИЕ А	74
ПРИЛОЖЕНИЕ Б.....	77
ПРИЛОЖЕНИЕ В	81

ВВЕДЕНИЕ

Сегодня многие средние и крупные компании анализируют свою деятельность по различным показателям с помощью построения аналитических или управленческих отчетов. Во многих компаниях России используют учетные системы на платформе 1С. С помощью данной системы сотрудники компании фиксируют результат своей деятельности, например, приходные и расходные накладные, перемещения товаров между складами, производство продукции, начисления зарплаты сотрудникам и т.д.

В связи с этим в учетных системах накапливается большой объем информации – это значит, что со временем построение управленческих отчетов будет занимать все больше и больше времени, так как учетная система настроена на эффективную обработку транзакций. Для того, чтобы снизить нагрузку на учетную систему и сократить время формирования отчетов необходимо использовать OLAP-систему.

OLAP-системы обладают не только быстрым построением аналитической отчетности, но и имеют большой инструментарий для визуализации данных. Визуализации дают возможность аналитику смотреть на данные под разными ракурсами. Так же данные системы помогают руководящему звену компании принимать эффективные и правильные бизнес-решения.

В выпускной квалификационной работе я рассмотрю интеграцию учетных систем на платформе 1С с OLAP-системами для построения управленческих отчетов.

Целью выпускной квалификационной работы является выявление наиболее эффективного метода интеграции учетных систем на платформе 1С с OLAP-системами для построения управленческих отчетов.

Эффективность метода интеграции будет рассматриваться по следующим требованиям: временем извлечения, обработки и загрузки данных в хранилище, гибкостью решения и временем построения отчетов.

Для достижения указанной цели в работе решаются следующие задачи:

- 1) Описать и проанализировать предметную область;
- 2) Проанализировать существующие решения;
- 3) Представить решение проблемы;
- 4) Выполнить проектирование OLAP-системы;
- 5) Реализовать OLAP-систему.

При написании выпускной квалификационной работы применялись следующие методы исследования: интервьюирование, сравнительный и описательный анализ, моделирование.

Практическая значимость выпускной квалификационной работы заключается в исследовании существующих решений по интеграции 1С с OLAP-системой и реализации собственного решения.

Структурно работа состоит из введения, трех глав, заключения и библиографического списка.

Введение включает в себя актуальность темы ВКР, характеристику решаемой проблемы.

В первой главе проводится анализ предметной области.

Во второй главе выполняется описание технических решений, проектирование системы в целом и хранилища данных.

В третьей главе выполняется создание хранилища данных, разработка ETL-приложения и развертывание OLAP-куба.

В заключении содержатся выводы о целесообразности внедрения ИС на предприятии.

1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ. ПОСТАНОВКА ЗАДАЧИ

1.1 Предметная область

Платформа 1С в первую очередь «заточена» под эффективную обработку транзакций, т.е. ввод оперативной информации жизнедеятельности организации. Учетная система на платформе 1С строится на регистрах накопления. Данные регистры предназначены для накопления, например, оборотов продаж, и быстрого вывода информации в управленческих отчетах для пользователя. При вводе новой информации – информация в регистрах накопления агрегируется по измерениям, но при накоплении больших объемов данных даже этот механизм не справляется с быстрым выводом информации для пользователя. Так же не вся информация хранится в данных регистрах, некоторая в справочниках или регистрах сведений – это приводит к снижению быстродействия. При этом все еще происходит нагрузка на систему.

Главной проблемой является скорость получения данных из информационной базы при построении отчетов. Так же не мало важная проблема – это вывод данных в различных визуализациях, в учетной системе на платформе 1С визуализации ограничены. Это важно, так как одна визуализация данных может не показать истинную картину для принятия управленческого решения.

В данной проблеме могут помочь OLAP-технологии – это обработка данных, заключающаяся в агрегировании информации на основе больших массивов данных по многомерному принципу. Главная причина использования OLAP для обработки запросов – это скорость.

Рассмотрим на примере построение отчета из учетной системы 1С розничной сети по продаже ювелирных изделий. Каждое изделие в учетной системе уникальное, т.е. оно хранится в разрезе товара и характеристики. Характеристика – состоит из различных свойств изделия, например, артикула поставщика.

В учетной системе данной компании происходит учет всех изделий по различным свойствам. Данные свойства разделяются на predetermined и пользовательские. Предetermined – это размер, металл и цвет металла, вид таблицы хранения этих свойств представлен на рисунке 1.1.

Список Значения свойств изделий

Характеристика	Металл	Цвет металла	Размер
926325409, золото, 585 пробы, 20, 3,91гр	Золото	Комбинированный	20,00
926325456, золото, 585 пробы, 18, 3,56гр	Золото	Комбинированный	18,00
926325468, золото, 585 пробы, 16, 0,7гр	Золото	Комбинированный	16,00
926325490, золото, 585 пробы, 18, 1,77гр	Золото	Комбинированный	18,00
926325542, золото, 585 пробы, 40, 5,56гр	Золото	Родирование	40,00
920818915, золото, 585 пробы, 18, 5,16гр	Золото	Комбинированный	18,00
920818923, золото, 585 пробы, 1,21гр, фианит	Золото	Лимонный	
920819141, золото, 585 пробы, 0,72гр, фианит	Золото	Лимонный	
920819276, золото, 585 пробы, 1,66гр, фианит	Золото	Лимонный	
920819388, золото, 585 пробы, 18, 4,41гр	Золото	Комбинированный	18,00
100049886, золото, 585 пробы, 2,04гр	Золото	Красный	
100050097, золото, 585 пробы, 2,42гр	Золото	Красный	

Рисунок 1.1 – Вид таблицы хранения predetermined свойств изделий

Пользовательские свойства – это свойства, которые создал сам пользователь, т.е. они не хранятся в виде столбцов, вид таблицы хранения этих свойств представлен на рисунке 1.2.

Список Дополнительные значения свойств изделий

Характеристика	Свойство	Значение свойства
926325409, золото, 585 пробы, 20, 3,91гр	Способ производства	Полновес
926325409, золото, 585 пробы, 20, 3,91гр	Плетение	итальянка
926325409, золото, 585 пробы, 20, 3,91гр	Диаметр проволоки	0,60 мм
926325409, золото, 585 пробы, 20, 3,91гр	Вид браслетов	цепочечный
926325409, золото, 585 пробы, 20, 3,91гр	Товарная группа	браслет
926325456, золото, 585 пробы, 18, 3,56гр	Способ производства	Полновес
926325456, золото, 585 пробы, 18, 3,56гр	Плетение	итальянка
926325456, золото, 585 пробы, 18, 3,56гр	Диаметр проволоки	0,60 мм
926325456, золото, 585 пробы, 18, 3,56гр	Вид браслетов	цепочечный
926325456, золото, 585 пробы, 18, 3,56гр	Товарная группа	браслет
926325468, золото, 585 пробы, 16, 0,7гр	Способ производства	Полновес
926325468, золото, 585 пробы, 16, 0,7гр	Плетение	итальянка

Рисунок 1.2 – Вид таблицы хранения пользовательских свойств изделий

Исходя из хранения свойств изделий можно сделать такой вывод: чтобы нам построить отчет в разрезе данных нам необходимо написать запрос с множеством соединений таблиц, а известно, что соединения снижают скорость получения данных в запросе. В данных таблицах могут быть тысячи (миллионы) строк данных, так как каждое изделие уникальное.

Далее создадим отчет на платформе 1С. Данный отчет будет выводить продажи за период в разрезе свойств изделий, которые выберет пользователь, т.е. запрос должен формироваться динамически. Пример запроса представлен в приложении А.

В начале запроса происходит выборка всех predetermined свойств изделий и запись их во временную таблицу. Далее формируется запрос пользовательских свойств изделий, каждое свойство формируется из вложенного запроса. После формирования всех свойств – выполняется формирование оборотов продаж за период с итогами. Итоговый сформированный отчет представлен на рисунке 1.3.

Товарная группа / Цвет металла / Вставка / Размер	Количество	Масса
драгоценные вставки	324,000	634,92
Красный	241,000	448,99
Белый	48,000	123,75
бриллианты россыпь	20,000	64,24
бриллиант одиночный	15,000	27,22
сапфир	11,000	29,51
	8,000	19,62
17,5	1,000	1,88
18	1,000	5,75
16	1,000	2,26
изумруд	1,000	2,26
17,5	1,000	2,26
керамика	1,000	0,52
18	1,000	0,52
Комбинированный	35,000	62,18

Рисунок 1.3 – Пример отчета на платформе 1С

Время формирования данного отчета занимает примерно 1 минуту. Исходя из данного примера можно сделать вывод о необходимости использования OLAP-системы для эффективного построения аналитических отчетов. Так же стоит сказать о том, что данный отчет выводит только продажи, если в данный отчет добавить данные о себестоимости товара, то время формирования соответственно увеличиться.

Далее будет выполнено описание требований, предъявляемых к OLAP-системе.

1.2 Постановка требований к OLAP-системе

OLAP-система влияет на бизнес-процессы аналитики и принятия решений в организации. Исходя из затрагиваемых процессов можно описать требования к OLAP-системе.

Самое важное требование, предъявляемое к OLAP-системе – это скорость получения информации, позволяющая использовать их в процессе работы пользователя.

Система должна быть доступной, т.е. должна описывать очевидное требование к возможности одновременного многопользовательского доступа к информации с интегрированной системой разграничения прав доступа вплоть до уровня конкретной ячейки данных.

Так же у системы должны быть поддержка многомерности, как наиболее логичного пути анализа бизнеса и организации.

Помимо всего этого OLAP-система затрагивает процессы ETL – это процессы извлечения, обработки и загрузки данных в хранилище, без реализации данного процесса OLAP-система не будет функционировать, так как в ней не будет данных.

При проектировании ETL-процесса следует учитывать требования бизнеса по длительности всего процесса. Стоит учесть объем данных и время за которое необходимо загрузить их.

Так же стоит учитывать тот факт, что часто разработкой хранилища данных занимаются специалисты не знающие специфику 1С. Так и при разработке ХД стоит разделять ответственность: программисту 1С всего лишь нужно выдать определенные бизнесом для анализа данные, а ETL-специалисту обработать их и загрузить в хранилище.

Далее будут рассмотрены существующие решения по интеграции учетных систем на платформе 1С и OLAP-системы. Данные решения будут опробованы на розничной сети по продаже ювелирных изделий. Эта компания имеет учетную систему на платформе 1С с СУБД MSSQLServer.

1.3 Описание и анализ существующих решений

При помощи поиска информации в различных интернет-источниках для анализа были выявлены наиболее часто применяемые решения по интеграции учетной системы на платформе 1С с OLAP-системами:

- 1) Внешние соединения к платформе 1С;
- 2) Представления в СУБД MSSQLServer;
- 3) Web-сервисы и HTTP-сервисы на платформе 1С;
- 4) MS Power BI и стандартный интерфейс OData платформы 1С.

Так же были найдены готовые системы, так называемые коннекторы, которые используют один из этих методов интеграции с платформой 1С.

Стоит отметить, что возможных решений гораздо больше. Например, в данной работе не были отмечены такие методы передачи данных, как Excel/XML/DBF-файлы, так как эти методы явно будут медленнее, чем выбранные, и их мало кто использует.

1.3.1 Внешние соединение к платформе 1С

Внешние соединение – это механизм интеграции платформы 1С с другими приложениями. Оно обеспечивает прямой и быстрый способ программного доступа к данным 1С:Предприятия 8 из внешних программ путем создания внешнего сеанса в учетной системе.

Данный метод дает возможность создавать различные объекты платформы 1С, например, писать запросы к данным или обращаться к процедуре и т.п. Внешние соединение является методом, который появился в платформе довольно-таки давно, но все еще им пользуются для интеграции с 1С.

Преимущества:

- 1) Простой и быстрый способ реализации;
- 2) Универсальность – при смене СУБД сервера 1С решение будет работать.

Недостатки:

- 1) Инициализируется полноценный сеанс к платформе 1С;
- 2) Работает только на операционных системах Windows;
- 3) Безопасность чтения данных;
- 4) Нет разделения ответственности в ETL-процессе.

Привести примеры тиражируемых систем очень сложно. Скорее этот механизм применяется в каких-то локальных задачах.

1.3.2 Представления в MSSQLServer

Структура хранения данных учетной системы на платформе 1С в СУБД имеет непонятные названия таблицы и их полей для бизнес-пользователя, пример физической структуры регистра накопления «Продажи» представлен на рисунке 1.4.

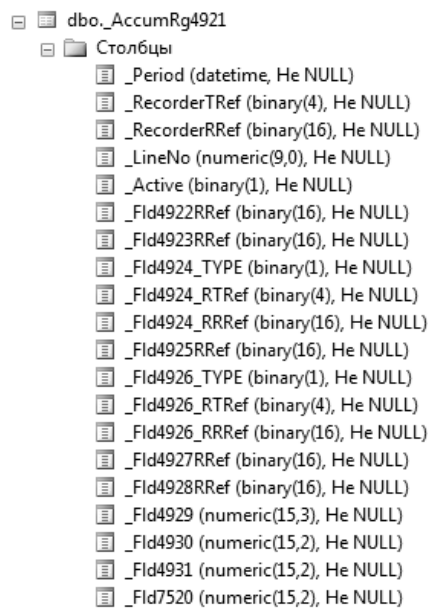


Рисунок 1.4 – Физическая структура РН «Продажи»

Для того, чтобы привести название таблиц и их полей в понятные бизнес-названия можно создать представления (view) в базе данных. После создания представлений к ним можно обращаться из приложений для получения данных.

Существует готовое решение, так называем коннектор, «АТКBiView-1С Коннектор (для Qlik/Tableau/PowerBI)», интерфейс программы представлен на рисунке 1.5.

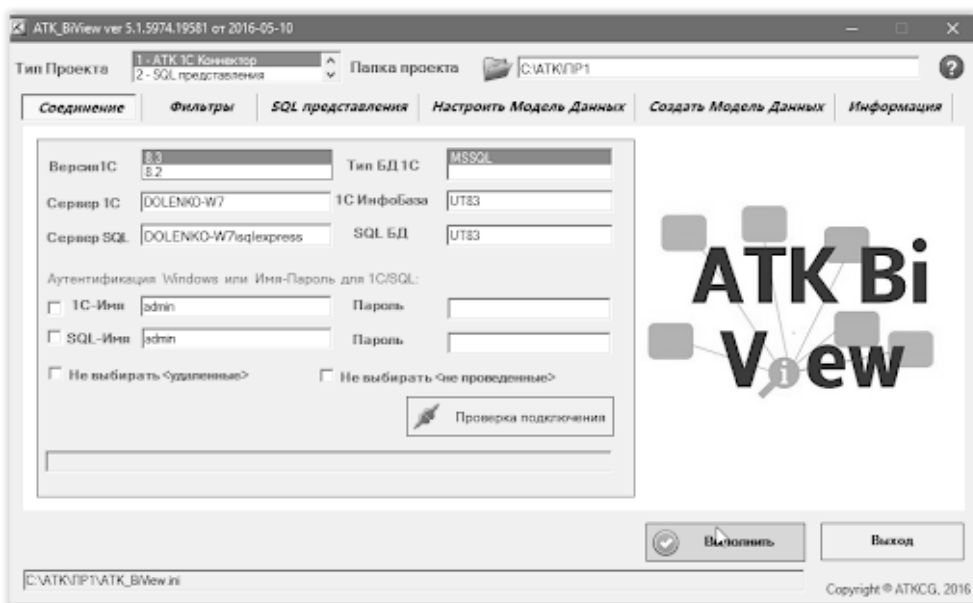


Рисунок 1.5 – Интерфейс программы «АТКBiView»

Данный коннектор подключается к серверу БД и приложения 1С и позволяет автоматически создавать представления на основании физических таблиц в БД, т.е. он переводит названия таблиц и их полей в такие названия, как они именованы в конфигурации 1С. В целом эта программа позволяет создавать удобные и понятные источники данных для приложений BI, работает только с СУБД MSSQL и PostgreSQL.

Преимущества:

- 1) Быстрый способ реализации;
- 2) Не инициализируется сеанс к приложению 1С;
- 3) Высокая скорость получения данных.

Недостатки:

- 1) Нарушает лицензионную политику 1С прямыми запросами к БД;
- 2) Происходит нагрузка на транзакционную БД при выборке данных для построения отчета в случае использования, как источника данных;
- 3) Представление может перестать работать, так как при обновлении конфигурации 1С происходит реструктуризация – влечет за собой изменение внутренних названий таблиц и их полей и необходимость пересоздания представлений;
- 4) Не все платформы по анализу поддерживают запрос данных из представлений;
- 5) При смене СУБД 1С придется пересоздавать представления;
- 6) Безопасность чтения данных;
- 7) Нет разделения ответственности в ETL-процессе;
- 8) Нет возможности для раздельного расположения серверов OLAP и 1С.

Стоит отметить, что в этом решении нужно четко следить за обновлениями конфигурации 1С, так как при каждом изменении структуры будет происходить реструктуризация и будут меняться внутренние названия таблиц и полей – это в свою очередь может привести к неработоспособности OLAP, либо не правильной загрузкой данных в хранилище.

1.3.3 Web-сервисы и HTTP-сервисы на платформе 1С

У платформы 1С существуют два механизма передачи данных по протоколу HTTP. Web-сервисы и HTTP-сервисы, отличаются они технологией передачи данных.

Web-сервисы – это механизм платформы, позволяющий выполнять интеграцию с другими информационными системами путем отправки данных в виде XML-пакета. Он является средством поддержки SOA (Service-Oriented Architecture) — сервис-ориентированной архитектуры, которая является современным стандартом интеграции приложений и информационных систем. Обычно с web-сервисом используют так называемые объекты конфигурации XDTO-пакеты – это схема «XSD» описания объектов передаваемых в виде XML. Данная схема описывает поля объекта, его тип данных, количество и т.п.

HTTP-сервис практически тоже самое, но отличается тем, что объем передаваемых данных потенциально меньше, меньшая вычислительная нагрузка и простота программирования клиента таких сервисов.

HTTP-сервисы ориентированы на «ресурсы», в то время как Web-сервисы ориентированы на «действия». Так же HTTP-сервис способен передавать данные в формате JSON.

Преимущества:

- 1) Универсальность – при смене СУБД сервера 1С решение будет работать;
- 2) Разделение ответственности в ETL-процессе;
- 3) Возможность формировать на стороне сервера 1С специфичные запросы к данным;
- 4) Поддерживает различные операционные системы;
- 5) Безопасность чтения данных.

Недостатки:

1) Дополнительная реализация методов формирования и отправки пакетов данных на сервере 1С.

1.3.4. MSPowerBI и стандартный интерфейс OData платформы 1С

Платформа 1С может автоматически формировать REST интерфейс для всего прикладного решения – это стандартный интерфейс OData. После того, как прикладное решение опубликовано на веб-сервере, сторонние системы могут обращаться к нему через REST интерфейс с помощью HTTP запросов. Благодаря универсальности и кроссплатформенности автоматически генерируемый REST интерфейс является основным инструментом для интеграции со сторонними системами.

OpenDataProtocol (OData) – это веб-протокол для запроса и изменения данных. Данный протокол позволяет выполнять с ресурсами различные действия посредством HTTP-команд и получать ответы в формате XML и JSON.

1С:Предприятие 8 входит в список систем, представленный на интернет-ресурсе «<https://www.odata.org/ecosystem/>», у которых существует OData.

Платформа 1С позволяет гибко настраивать доступ к объектам через OData, например, можно дать доступ на чтение только справочника «Номенклатура».

Power BI – это система компании Microsoft, представляющая набор служб бизнес-аналитики с поддержкой облачных технологий для анализа и визуализации данных. Основное преимущество данной технологии – это возможность построения красивых информационных панелей (dashboard), как правило, с ключевыми показателями деятельности компании, доступных на любом устройстве. Так же данная система поддерживает и статический анализ данных с помощью языка программирования «R» или «Python».

В PowerBI есть возможность использовать источник данных «OData», пример подключения представлен на рисунке 1.6.

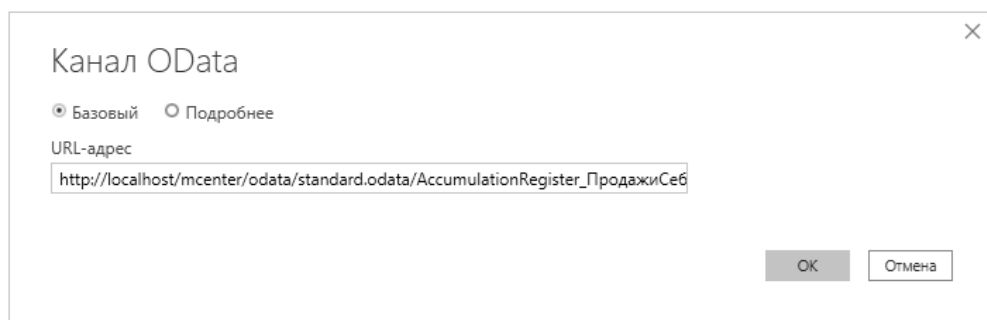


Рисунок 1.6 – Подключение к источнику данных «OData»

Данный канал подключения поддерживает два варианта: постоянный запрос данных, либо их загрузка. Пример загруженных данных, представлен на рисунке 1.7.

Номенклатура_Key	Масса	Стоимость
31e1328e-6e87-11dd-bf25-0080484c172b	0,77	121,43
31e1328e-6e87-11dd-bf25-0080484c172b	1,03	264,71
31e1328e-6e87-11dd-bf25-0080484c172b	2,64	5130,22
31e1328e-6e87-11dd-bf25-0080484c172b	6,41	608,95
77631330-7d20-11db-93be-000c762b3e6a	0,52	4292,1
77631330-7d20-11db-93be-000c762b3e6a	0,85	1688,1
77631330-7d20-11db-93be-000c762b3e6a	0,98	3587,4
77631330-7d20-11db-93be-000c762b3e6a	1,12	5916,46
77631330-7d20-11db-93be-000c762b3e6a	1,2	1874,78
77631330-7d20-11db-93be-000c762b3e6a	1,22	4171,24
77631330-7d20-11db-93be-000c762b3e6a	1,27	3790,33
77631330-7d20-11db-93be-000c762b3e6a	1,28	212,48
77631330-7d20-11db-93be-000c762b3e6a	1,3	2426,48
77631330-7d20-11db-93be-000c762b3e6a	1,34	2629,89
77631330-7d20-11db-93be-000c762b3e6a	1,36	2488,8
77631330-7d20-11db-93be-000c762b3e6a	1,39	2170,44

Рисунок 1.7 – Пример загруженных данных в PowerBI

Преимущества:

- 1) Быстрый способ реализации;
- 2) Возможность загрузки данных «на лету»;
- 3) Универсальность – при смене СУБД сервера 1С решение будет работать;

4) Безопасность чтения данных.

Недостатки:

- 1) Схема данных остается прежней;
- 2) Нет возможности использовать OLAP-куб;
- 3) Скорость построения отчетов не отличается от скорости построения в учетной системе.

Вывод по разделу один

В разделе «Анализ предметной области. Постановка задачи» выполнен анализ предметной области, определены требования для системы. Проанализированы существующие решения и выявлены их преимущества и недостатки.

2 ПРОЕКТИРОВАНИЕ OLAP-СИСТЕМЫ

2.1 Выбор и описание технических решений

При реализации OLAP-системы использованы несколько программных средств и технологий:

- 1) Для доступа к данным 1С:Предприятие 8 REST-интерфейс платформы – OData;
- 2) Для хранения данных использовалась система управления базами данных MSSQLServer;
- 3) Для извлечения, обработки и загрузки данных в хранилище SQLServerIntegrationServices;
- 4) Для создания OLAP-кубов SQL Server Analysis Services;
- 5) Для размещения REST-интерфейса OData использовался веб-сервер Apache;
- 6) Для построения аналитических и управленческих отчетов из OLAP-кубов MS PowerBI.

2.2.1 Технология OData

OpenDataProtocol (OData) – это RESTful-протокол, обеспечивающий эффективные операции запроса и модификации над данными. Выражения и запросы ресурсов базируются на URL HTTP-запроса, а результаты возвращаются в HTTP-ответе.

Данный механизм встроен в технологическую платформу 1С версии 8 и позволяет получать доступ к данным из внешнего приложения без модификации кода прикладного решения.

OData способен возвращать данные в форматах JSON либо XML. В данной работе будем использовать формат JSON, так как это быстрее.

2.2.2 СУБДMicrosoftSQLServer

MSSQLServerявляетсяширокораспространённой системой управления реляционными базами данных. Эта СУБД имеет поддержку языка SQL, который позволяет реализовать запросы к базе данных для извлечения необходимой информации.

MSSQLServerчаще всего используется, как СУБД для 1С:Предприятие 8, так как имеет довольно простое развертывание и администрирование.

В процессе разработки СУБД MSSQLServerзапущена всегда, так как используется для хранения данных, функционирование OLAP-системы без нее было бы невозможным.

2.2.3 SQLServerIntegrationServices

SSIS (SQLServerIntegrationServices) – этоинструмент, которыйпозволяется реализовать в удобном виде процессы извлечения, обработки и загрузки данных в хранилище. Так же он позволяет организовать все эти процессы в многопоточном режиме.

Службы Integration Services могут извлекать и преобразовывать данные из ряда таких источников, как файлы XML-данных, неструктурированные файлы и источники реляционных данных, и затем загружать эти данные в хранилище.

2.2.4 SQLServerAnalysisServices

Analysis Services – это средство аналитических данных, используемое в службе поддержки принятия решений и бизнес-аналитики. Она предоставляет возможности модели семантических данных корпоративного уровня для анализа данных бизнес-аналитики (BI) и приложений для создания отчетов, таких как

Power BI, Excel, Reporting Services отчеты и другие средства визуализации данных.

SSAS выполняет обработку данных в оперативной памяти (in-memory) – это дает пользователям получать данные из OLAP-куба быстрее.

2.2.5 Веб-сервер Apache

Apache – свободно распространяемый web-сервер, который часто используется в Unix-подобных операционных системах. Большое распространение Apache получил за то, что является надежным, имеет гибкую конфигурацию, а также имеет большое сообщество пользователей, которые могут помочь в конфигурировании данного web-сервера.

Публикация доступа к 1С:Предприятие 8 так же поддерживается через веб-сервер Apache.

2.2.6 Microsoft Power BI

Power BI – комплексное программное обеспечение бизнес-анализа (BI), объединяющее несколько программных продуктов, имеющих общий технологический и визуальный дизайн, соединителей (шлюзов), а также web-сервисов.

Данный инструмент может работать просто и быстро, формируя краткие аналитические сведения на базе книги Excel или локальной базы данных. Однако Power BI также является надежным продуктом корпоративного уровня, который пригоден не только для масштабного моделирования в режиме реального времени, а также для разработки индивидуальных решений. Таким образом, он может выступать в качестве вашего личного средства визуализации и ведения отчетов, а также служить подсистемой аналитики и принятия решений для групповых проектов, отделений или целых организаций.

2.3 Проектирование архитектуры системы

Для того, чтобы понять, как взаимодействуют, описанные выше, компоненты – необходимо разработать архитектуру системы в целом, представлена на рисунке 2.1.



Рисунок 2.1 – Архитектура системы

Для начала стоит описать, как будет выглядеть источник данных:

- 1) Будет развернута система управления базами данных MSSQLServer для кластера серверов 1С:Предприятия;
- 2) На сервере 1С будет установлена информационная база с нетиповой конфигурацией «Управление торговлей 10»;
- 3) Будет развернут веб-сервер Apache;
- 4) На веб-сервере будут опубликованы веб-сервисы информационной базы.

Доступ к данным базы 1С будет происходить через веб-сервер.

Архитектура OLAP-системы будет выглядеть следующим образом:

- 1) Извлечение, обработка и загрузка данных в хранилище будет происходить с помощью служб MSSQLIntegrationServices;
- 2) Будет развернута СУБД MSSQLServer для хранилища данных;
- 3) На основе хранилища данных будет создан OLAP-куб с помощью служб MSSQLAnalysisServices;
- 4) Доступ к OLAP-кубу будет осуществляться через приложение PowerBI.

2.4 Проектирование хранилища данных

Хранилище данных должно выглядеть в виде «звезды» или «снежинки» — это специальная организация реляционных таблиц, удобная для хранения многомерных показателей, лежит в основе реляционного OLAP. Модель данных состоит из двух типов таблиц: одной таблицы фактов и нескольких таблиц измерений.

Таблица фактов обычно содержит одну или несколько колонок типа «число», дающих числовую характеристику какому-то аспекту предметной области (например, объём продаж для торговой компании или сумма платежей

для банка), и несколько целочисленных колонок-ключей для доступа к таблицам измерений.

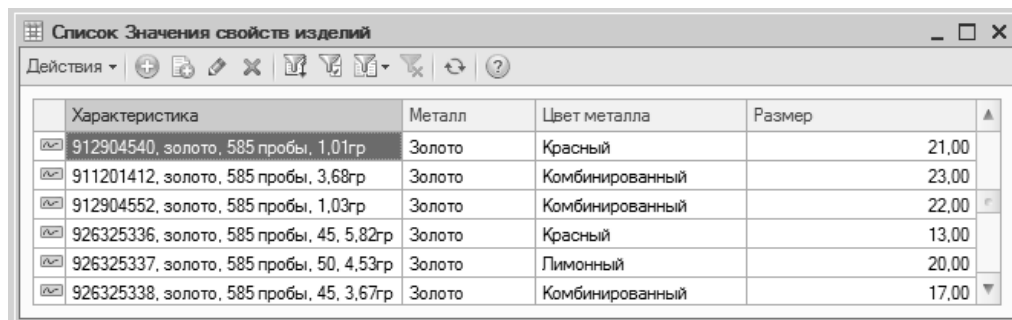
Таблицы измерений расшифровывают ключи, на которые ссылается таблица фактов. Например, измерение «Товары» может содержать сведения о названии товара, его производителе, типе товара. За счёт использования специальной структуры таблицы измерений реализуется иерархия измерений, в том числе ветвящаяся.

Обычно данные в таблицах-измерениях денормализованы: ценой несколько неэффективного использования дискового пространства удастся уменьшить число участвующих в операции соединения таблиц, что обычно приводит к сильному уменьшению времени выполнения запроса.

Для того, чтобы правильно спроектировать хранилище данных необходимо проанализировать и описать источники данных из базы 1С. Начать нужно с фактов о продажах – в базе 1С они хранятся в документах «Отчеты о розничных продажах». Данный документ формируется автоматически после закрытия кассовой смены в магазине продавцами. Из этого документа нам интересны следующие данные: дата, склад, таблица «Товары»: номенклатура, характеристика номенклатуры, сумма, количество, цена, продавец – эти данные будут загружаться в хранилище. Пример документа представлен на рисунке 2.2.

Так же нас будут интересовать свойства товаров: преопределенные и пользовательские. Преопределенные свойства на уровне СУБД имеют собственные столбцы с данными. Пользовательские свойства имеют вид хранения «столбец наименование + столбец значение».

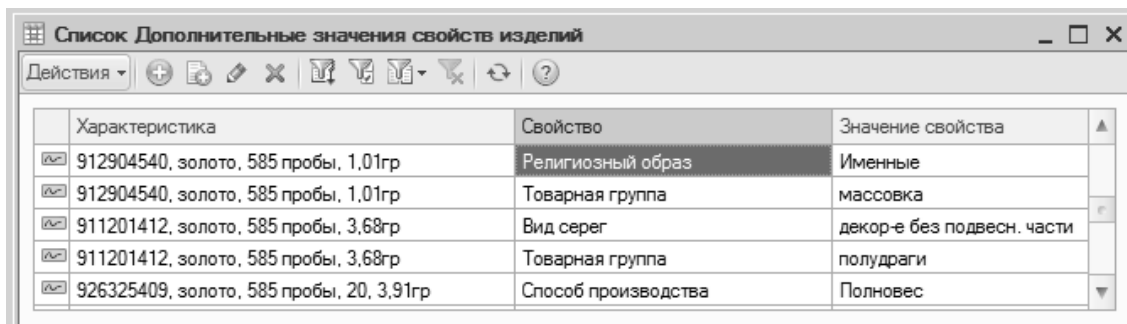
Преопределенные свойства хранятся в регистре сведений «Значения свойств изделий», представлен на рисунке 2.4.



Характеристика	Металл	Цвет металла	Размер
912904540, золото, 585 пробы, 1,01гр	Золото	Красный	21,00
911201412, золото, 585 пробы, 3,68гр	Золото	Комбинированный	23,00
912904552, золото, 585 пробы, 1,03гр	Золото	Комбинированный	22,00
926325336, золото, 585 пробы, 45, 5,82гр	Золото	Красный	13,00
926325337, золото, 585 пробы, 50, 4,53гр	Золото	Лимонный	20,00
926325338, золото, 585 пробы, 45, 3,67гр	Золото	Комбинированный	17,00

Рисунок 2.4 – Регистр сведений «Значения свойств изделий»

Пользовательские свойства хранятся в регистре сведений «Доп. значения свойств изделий», представлен на рисунке 2.5.



Характеристика	Свойство	Значение свойства
912904540, золото, 585 пробы, 1,01гр	Религиозный образ	Именные
912904540, золото, 585 пробы, 1,01гр	Товарная группа	массовка
911201412, золото, 585 пробы, 3,68гр	Вид серег	декор-е без подвесн. части
911201412, золото, 585 пробы, 3,68гр	Товарная группа	полудраги
926325409, золото, 585 пробы, 20, 3,91гр	Способ производства	Полновес

Рисунок 2.5 – Регистр сведений «Доп. значения свойств изделий»

Выполним проектирование хранилища для розничной сети по продаже ювелирных изделий. Таблицей фактов будет «Sales» (Продажи) – содержит факты продаж по различным измерениям. В таблице 1 представлено описание полей таблицы.

Таблица 1 – Описание полей таблицы «Sales»

Поле	Тип данных	Назначение	Комментарий
id	bigint	Идентификатор	Первичный ключ, автоинкремент
date	datetime	Дата	not null
sourceId	nchar(36)	Идентификатор источника	not null
idStore	smallint	Идентификатормагазина	Внешний ключ, not null
idSeller	smallint	Идентификатор продавца	Внешний ключ, not null
idProduct	smallint	Идентификатор товара	Внешний ключ, not null
idProductProperty	bigint	Идентификаторсвойстватовара	Внешний ключ, not null
count	float	Количество	not null
firstCost	float	Себестоимость	not null
cost	float	Стоимость	not null
costWithoutDiscount	float	Стоимость (без скидок)	not null
grossProfit	float	Валовая прибыль	not null
weight	float	Масса	not null

Таблица фактов содержит 5 измерений: дата, магазины, продавцы, товара и свойства товаров. Все поля должны иметь определенные значения для построения OLAP-куба.

Таблица измерение «Stores» (Магазины) – содержит информацию о магазинах компании. В таблице 2 представлено описание полей таблицы.

Таблица 2 – Описание полей таблицы «Stores»

Поле	Тип данных	Назначение	Комментарий
id	smallint	Идентификатор	Первичный ключ, автоинкремент
name	nchar(200)	Наименование	not null
sourceId	nchar(36)	Идентификатор источника	not null

Таблица измерение «Sellers» (Продавцы) – содержит информацию о продавцах компании. В таблице 3 представлено описание полей таблицы.

Таблица 3 – Описание полей таблицы «Sellers»

Поле	Тип данных	Назначение	Комментарий
id	smallint	Идентификатор	Первичный ключ, автоинкремент
name	nchar(200)	Наименование	not null
sourceId	nchar(36)	Идентификатор источника	not null

Таблица измерение «Products» (Товары) – содержит информацию о товарах, продаваемых компанией. В таблице 4 представлено описание полей таблицы.

Таблица 4 – Описание полей таблицы «Products»

Поле	Тип данных	Назначение	Комментарий
id	smallint	Идентификатор	Первичный ключ, автоинкремент
name	nchar(200)	Наименование	not null
sourceId	nchar(36)	Идентификатор источника	not null

Таблица измерение «ProductProperties» (Свойства товаров) – содержит информацию о свойствах товаров. В таблице 5 представлено описание полей таблицы.

Таблица 5 – Описание полей таблицы «ProductProperties»

Поле	Тип данных	Назначение	Комментарий
id	smallint	Идентификатор	Первичный ключ, автоинкремент
name	nchar(200)	Наименование	not null
sourceId	nchar(36)	Идентификатор источника	not null
idMetal	smallint	Идентификатор металла	Внешний ключ
idMetalColor	smallint	Идентификатор цвета	Внешний ключ

Поле	Тип данных	Назначение	Комментарий
		металла	
size	float	Размер	

Таблица «Metals» (Металлы) – содержит информацию о металлах изделий. В таблице 6 представлено описание полей таблицы.

Таблица 6 – Описание полей таблицы «Metals»

Поле	Тип данных	Назначение	Комментарий
id	smallint	Идентификатор	Первичный ключ, автоинкремент
name	nchar(100)	Наименование	not null
sourceId	nchar(36)	Идентификатор источника	not null

Таблица «MetalColors» (Цвета металла) – содержит информацию о цветах металла изделий. В таблице 7 представлено описание полей таблицы.

Таблица 7 – Описание полей таблицы «MetalColors»

Поле	Тип данных	Назначение	Комментарий
id	smallint	Идентификатор	Первичный ключ, автоинкремент
name	nchar(100)	Наименование	not null
sourceId	nchar(36)	Идентификатор источника	not null

Таблица «AdditionalProperties» (Дополнительные свойства) – содержит информацию о дополнительных свойствах товаров. В таблице 8 представлено описание полей таблицы.

Таблица 8 – Описание полей таблицы «AdditionalProperties»

Поле	Тип данных	Назначение	Комментарий
id	smallint	Идентификатор	Первичный ключ, автоинкремент
name	nchar(200)	Наименование	not null
sourceId	nchar(36)	Идентификатор источника	not null

Таблица «AdditionalPropertyValues» (Значения дополнительных свойств) – содержит информацию о возможных значениях дополнительных свойств товаров. В таблице 9 представлено описание полей таблицы.

Таблица 9 – Описание полей таблицы «AdditionalPropertyValues»

Поле	Тип данных	Назначение	Комментарий
id	smallint	Идентификатор	Первичный ключ, автоинкремент
name	nchar(200)	Наименование	not null
sourceId	nchar(36)	Идентификатор источника	not null

Таблица «AdditionalProductPropertyValues» (Значения дополнительных свойств товаров) – является связкой свойств товаров с дополнительными свойствами. Так как свойства может создаваться сам пользователи их хранение реализовано следующим образом – «столбец наименование + столбец значение». В таблице 10 представлено описание полей таблицы.

Таблица 10 – Описание полей таблицы «AdditionalProductPropertyValues»

Поле	Тип данных	Назначение	Комментарий
idProductProperty	bigint	Идентификатор свойства товара	Первичный ключ
idAdditionalProperty	smallint	Идентификатор доп. свойства	
idAdditionalPropertyValue	smallint	Идентификатор значения	Внешний ключ, notnull

Для того, чтобы удобно анализировать данные в разрезе дополнительных свойств товаров после загрузки данных в хранилище будет автоматически формироваться представление, которые преобразует свойства в столбцы при помощи хранимой процедуры.

После описания структуры хранения всех таблиц хранилища создадим ER-диаграмму для более наглядного отражения связей между сущностями. На диаграмме прекрасно видна связь таблицы фактов «Sales» с ее измерениями: «Stores», «Sellers», «Products» и «ProductProperties». Диаграмма представлена на рисунке 2.6.

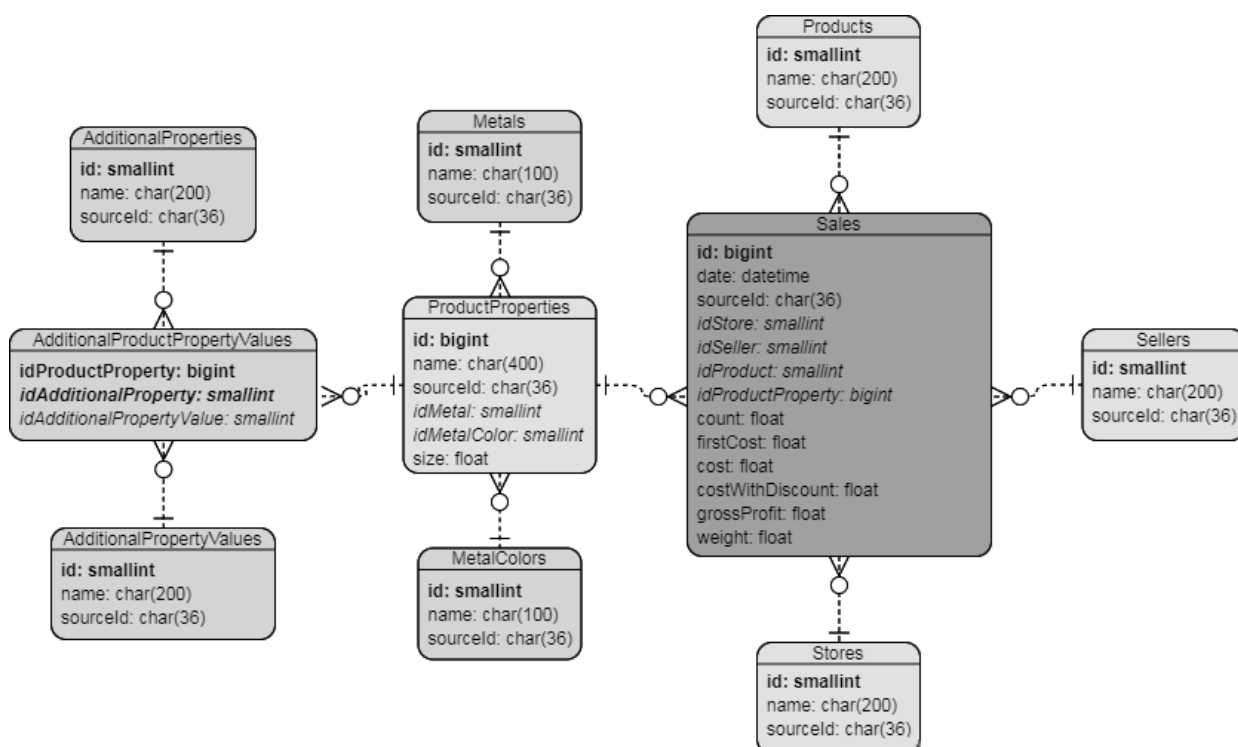


Рисунок 2.6 – ER-диаграмма хранилища данных

После проектирования хранилища данных необходимо будет создать все сущности в СУБД. Так же задать все ключи и ограничения.

Вывод по разделу два

В разделе «Проектирование OLAP-системы» выполнен выбор и описание технических решений, спроектирована архитектура системы и хранилища данных.

3 РЕАЛИЗАЦИЯ И ТЕСТИРОВАНИЕ OLAP-СИСТЕМЫ

3.1 Создание хранилища данных

Для начала необходимо создать все таблицы измерения хранилища данных, так как таблица фактов содержит внешние ключи на измерения. Так же сразу выполним создание необходимых ключей и индексов таблиц. Создание таблиц в MSSQL Server выполним с помощью скриптов в SQL Server Management Studio.

Для хранения информации о магазинах, продавцах и товаров компании создадим таблицы «Stores», «Sellers» и «Products», код скрипта приведен в листинге 1.

Листинг 1 – Скрипт создания таблиц «Stores», «Sellers» и «Products»

```
CREATE TABLE [dbo].[Stores]
(
    [id] [smallint] IDENTITY(1, 1) NOT NULL,
    [name] [nvarchar](200) NOT NULL,
    [sourceId] [nvarchar](36) NOT NULL,
    CONSTRAINT [PK_Stores] PRIMARY KEY CLUSTERED ([id] ASC)
);
CREATE TABLE [dbo].[Sellers]
(
    [id] [smallint] IDENTITY(1, 1) NOT NULL,
    [name] [nvarchar](200) NOT NULL,
    [sourceId] [nvarchar](36) NOT NULL,
    CONSTRAINT [PK_Sellers] PRIMARY KEY CLUSTERED ([id] ASC)
);
CREATE TABLE [dbo].[Products]
(
    [id] [smallint] IDENTITY(1, 1) NOT NULL,
    [name] [nvarchar](200) NOT NULL,
    [sourceId] [nvarchar](36) NOT NULL,
    CONSTRAINT [PK_Products] PRIMARY KEY CLUSTERED ([id] ASC)
);
```

Для хранения информации о металлах и их цветах свойств товаров создадим таблицы «Metals» и «MetalColors», связь между таблицами будет

проходить через таблицу «Sales», так как у каждого металла может быть несколько цветов. Код скрипта создания таблиц приведен в листинге 2.

Листинг 2 – Скрипт создания таблиц «Metals» и «MetalColors»

```
CREATETABLE [dbo].[Metals]
(
    [id] [smallint] IDENTITY(1, 1)NOTNULL,
    [name] [nvarchar](100)NOTNULL,
    [sourceId] [nvarchar](36)NOTNULL,
    CONSTRAINT [PK_Metals] PRIMARYKEYCLUSTERED ([id] ASC)
);

CREATETABLE [dbo].[MetalColors]
(
    [id] [smallint] IDENTITY(1, 1)NOTNULL,
    [name] [nvarchar](100)NOTNULL,
    [sourceId] [nvarchar](36)NOTNULL,
    CONSTRAINT [PK_MetalColors] PRIMARYKEYCLUSTERED ([id] ASC)
)
```

Для хранения информации о свойствах товаров создадим таблицу «ProductProperties» с двумя внешними ключами на таблицы «Metals» и «MetalColors», код скрипта приведен в листинге 3.

Листинг 3 – Код создания таблицы «ProductProperties»

```
CREATETABLE [dbo].[ProductProperties]
(
    [id] [bigint] IDENTITY(1, 1)NOTNULL,
    [name] [nvarchar](400)NOTNULL,
    [sourceId] [nvarchar](36)NOTNULL,
    [idMetal] [smallint] NULL,
    [idMetalColor] [smallint] NULL,
    [size] [float] NULL,
    CONSTRAINT [PK_ProductProperties] PRIMARYKEYCLUSTERED ([id] ASC)
);

ALTERTABLE [dbo].[ProductProperties] WITHCHECKADDCONSTRAINT
[FK_ProductProperties_MetalColors] FOREIGNKEY([idMetalColor])
REFERENCES [dbo].[MetalColors]([id]);

ALTERTABLE [dbo].[ProductProperties] WITHCHECKADDCONSTRAINT
[FK_ProductProperties_Metals] FOREIGNKEY([idMetal])
REFERENCES [dbo].[Metals]([id]);
```

Для хранения дополнительных свойств товаров необходимо создать три таблицы: «AdditionalProperties» – содержит информацию о дополнительных свойствах, «AdditionalPropertyValues» – содержит возможные значения дополнительных свойств, «AdditionalProductPropertyValues» – является связкой свойства товара с дополнительным свойством и его значением. Код скрипта создания таблиц приведен в листинге 4.

Листинг 4 – Код создания таблиц «AdditionalProperties», «AdditionalPropertyValues» и «AdditionalProductPropertyValues»

```

CREATETABLE [dbo].[AdditionalProperties]
(
    [id] [smallint] IDENTITY(1, 1)NOTNULL,
    [name] [nvarchar](200)NOTNULL,
    [sourceId] [nvarchar](36)NOTNULL,
    CONSTRAINT [PK_AdditionalProperties] PRIMARYKEYCLUSTERED ([id] ASC)
);
CREATETABLE [dbo].[AdditionalPropertyValues]
(
    [id] [smallint] IDENTITY(1, 1)NOTNULL,
    [name] [nvarchar](200)NOTNULL,
    [sourceId] [nvarchar](36)NOTNULL,
    CONSTRAINT [PK_AdditionalPropertyValues] PRIMARYKEYCLUSTERED ([id] ASC)
);
CREATETABLE [dbo].[AdditionalProductPropertyValues]
(
    [idProductProperty] [bigint] NOTNULL,
    [idAdditionalProperty] [smallint] NOTNULL,
    [idAdditionalPropertyValue] [smallint] NOTNULL,
    CONSTRAINT [PK_AdditionalProductPropertyValues_1] PRIMARYKEYCLUSTERED ([idProductProperty] ASC, [idAdditionalProperty] ASC)
);

ALTERTABLE [dbo].[AdditionalProductPropertyValues]
WITHCHECKADDCONSTRAINT
[FK_AdditionalProductPropertyValues_AdditionalProperties]
FOREIGNKEY([idAdditionalProperty])
REFERENCES [dbo].[AdditionalProperties]([id]);
ALTERTABLE [dbo].[AdditionalProductPropertyValues]
WITHCHECKADDCONSTRAINT
[FK_AdditionalProductPropertyValues_AdditionalPropertyValues]
FOREIGNKEY([idAdditionalPropertyValue])
REFERENCES [dbo].[AdditionalPropertyValues]([id]);

```

Окончание листинга 4

```
ALTERTABLE [dbo].[AdditionalProductPropertyValues]  
WITHCHECKADDCONSTRAINT  
[FK_AdditionalProductPropertyValues_ProductProperties]  
FOREIGNKEY([idProductProperty])  
REFERENCES [dbo].[ProductProperties]([id]);
```

Далее выполним создание таблицы фактов «Sales», которая будет накапливать продажи компании в различных измерениях и с различными фактами, код скрипта приведен в листинге 5.

Листинг 5 – Код создания таблицы «Sales»

```
CREATETABLE [dbo].[Sales](  
    [id] [bigint] IDENTITY(1,1)NOTNULL,  
    [date] [datetime] NOTNULL,  
    [sourceId] [nchar](36)NOTNULL,  
    [idStore] [smallint] NOTNULL,  
    [idSeller] [smallint] NOTNULL,  
    [idProduct] [smallint] NOTNULL,  
    [idProductProperty] [bigint] NOTNULL,  
    [count] [float] NOTNULLCONSTRAINT [DF_Sales_count] DEFAULT ((0)),  
    [firstCost] [float] NOTNULLCONSTRAINT [DF_Sales_firstCost_1]  
    DEFAULT ((0)),  
    [cost] [float] NOTNULLCONSTRAINT [DF_Sales_cost] DEFAULT((0)),  
    [costWithoutDiscount] [float] NOTNULLDEFAULT ((0)),  
    [grossProfit] [float] NOTNULLCONSTRAINT [DF_Sales_grossProfit]  
    DEFAULT ((0)),  
    [weight] [float] NOTNULLCONSTRAINT [DF_Sales_weight] DEFAULT ((0)),  
    CONSTRAINT [PK_Sales] PRIMARYKEYCLUSTERED ([id] ASC)  
);
```

```
ALTERTABLE [dbo].[Sales] WITHCHECKADDCONSTRAINT  
[FK_Sales_ProductProperties] FOREIGNKEY([idProductProperty])  
REFERENCES [dbo].[ProductProperties]([id]);
```

```
ALTERTABLE [dbo].[Sales] WITHCHECKADDCONSTRAINT  
[FK_Sales_Products] FOREIGNKEY([idProduct])  
REFERENCES [dbo].[Products]([id]);
```

```
ALTERTABLE [dbo].[Sales] WITHCHECKADDCONSTRAINT  
[FK_Sales_Sellers] FOREIGNKEY([idSeller])  
REFERENCES [dbo].[Sellers]([id]);
```

```
ALTERTABLE [dbo].[Sales] WITHCHECKADDCONSTRAINT  
[FK_Sales_Stores] FOREIGNKEY([idStore])  
REFERENCES [dbo].[Stores]([id]);
```

Для того, чтобы не рассчитывать дополнительные показатели при запросе данных – эффективнее их хранить сразу рассчитанными в таблице, как например поле «grossProfit» (валовая прибыль). Так же все поля должны быть «NOTNULL», так как при создании OLAP-куба будет возникать ошибка и смысла для хранения неопределенного значения в хранилище для анализа нет.

Поле «sourceId» во всех таблицах используется для сопоставления объекта 1С записи в таблице хранилища данных – при загрузке данных ETL-приложение по нему будет находить соответствующую запись в таблице хранилища.

После создания всех таблиц сформируем диаграмму таблиц в SQL ServerManagementStudio, чтобы наглядно увидеть связи между таблицами, представлено на рисунке 3.1.

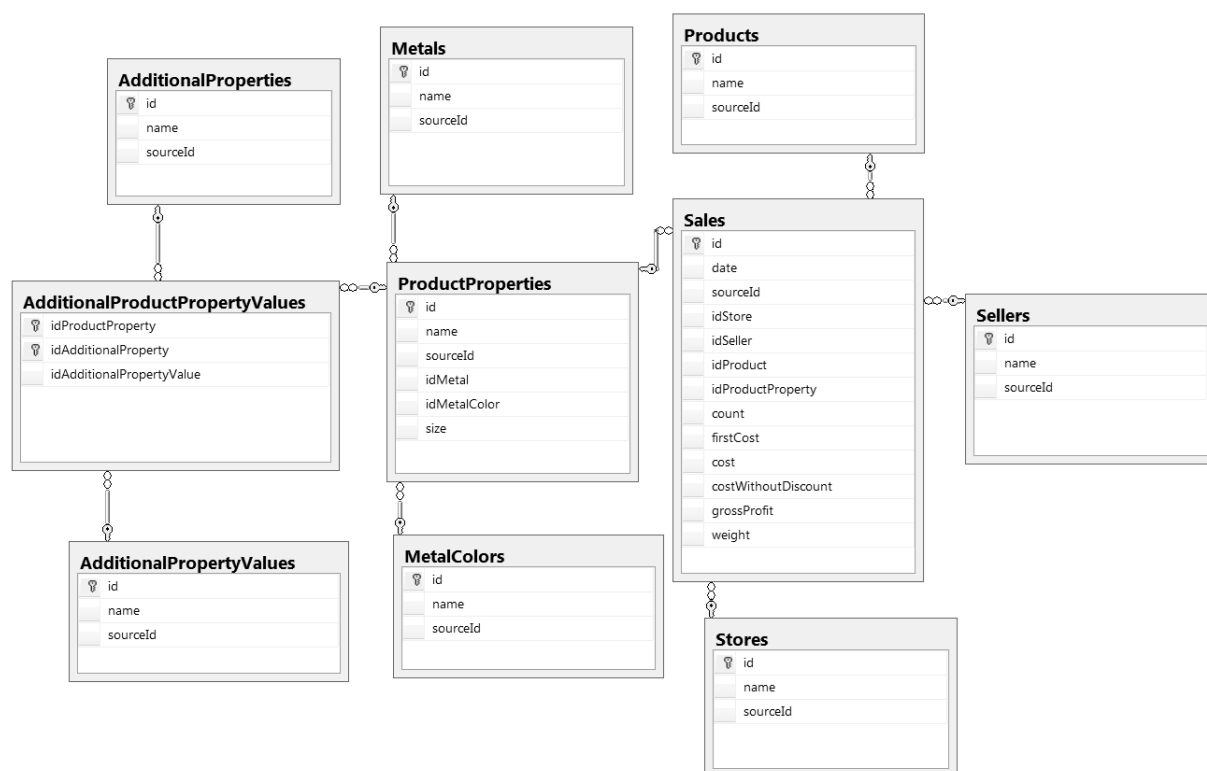


Рисунок 3.1 – Диаграмма таблиц

Далее перейдем к исследованию существующих решений на длительность ETL-процесса.

3.2 Исследование длительности ETL-процесса существующих решений

Реализуем существующие решения для исследования длительности ETL-процесса. Для каждого решения создано свое хранилище данных, количество строк записей для исследования будет 1 000 000 – данный объем данных даст понятное видение времени длительности процесса каждого решения.

Выполним реализацию решения «Внешние соединения к платформе 1С» – для этого создадим проект «WindowsForms» в Visual Studio на языке C#. На основной форме создадим поле для отображения времени и кнопку «Выполнить». Код кнопки приведен в приложении А. В начале выполняется подключение к базе 1С через СОМ-коннектор, после выполняется запрос данных и цикл обходить каждую запись выборки и добавляет в хранилище через хранимую процедуру в СУБД.

Время длительности процесса получилось 50 минут. Довольно долго, при большом количестве данных может работать гораздо дольше.

Выполним реализацию решения «Web-сервисы и HTTP-сервисы на платформе 1С». Для начала необходимо выполнить реализацию методов отправки данных на стороне 1С, их будет два, первый отправляет данные в формате XML – код приведен в приложении Б, второй отправляет данные в формате JSON – код приведен в приложении В. Оба метода выполняют запрос данных в 1С, формируют пакет данных и отправляют получателю.

Для первого метода в конфигурации создан объект «XDTO-пакет» – данный объект представляет собой схему XML-документа, с помощью которого производится формирование пакета данных в формате XML и объект «Web-сервис» с названием «Sales» с методом «Get», представлено на рисунке 3.2.

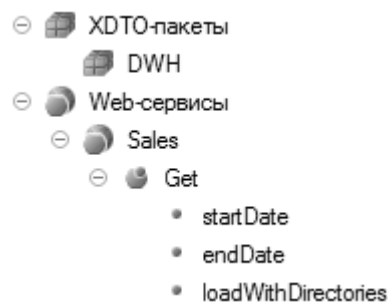


Рисунок 3.2 – «XDTO-пакет» и «Web-сервис» в конфигурации

Для второго метода в конфигурации создан объект «HTTP-сервис» с названием «Sales» с методом «Get». Данный объект позволяет с клиента отправлять HTTP-команды через веб-сервер на сервер 1С, представлено на рисунке 3.3.



Рисунок 3.3 – «HTTP-сервис» в конфигурации

Обязательно нужно опубликовать эти методы на веб-сервере Apache, чтобы к ним можно было обращаться по HTTP.

Далее создадим проект «IntegrationServicesProject» в Visual Studio. Данный проект разрабатывается в виде последовательно или параллельно выполняющихся процессов по извлечению, обработки и загрузки данных в хранилище. Он очень удобно позволяет создавать представления источников данных, после извлечения данных обрабатывать их и загружать в хранилище. Так же данный механизм поддерживает многопоточность, что существенно влияет на производительность приложения.

В данном решении весь процесс ETL будет одинаковым, кроме источника данных. Создадим поток управления – это основной процесс в проекте, в котором задается последовательность выполнения задач потоков данных – это сами процессы извлечения, обработки или загрузки данных. Создадим параметры

пакетаслужб, их использование позволяет вариативно выполнять поток управления не изменяя его, представлены на рисунке 3.4.



Имя	Тип данных	Значение	Конфиденциальный	Обязательное	Описание
 counterYears	Int32	4	False	False	
 loadWithDirectories	String	false	False	False	

Рисунок 3.4 – Параметры пакета служб

Параметр «counterYears» указывает сколько итераций извлечения данных нужно выполнить. Параметр «loadWithDirectories» если он имеет значение «true», тогда при извлечении данных так же извлекаются данные справочников.

Создадим переменные пакета служб для того, чтобы выполнить извлечение данных в цикле, представлены на рисунке 3.5.






Имя	Область	Тип данных	Значение	Выражение	
 endDate	TestREST	String	20130131		...
 endYear	TestREST	Int32	2014		...
 inc	TestREST	Int32	0		...
 startDate	TestREST	String	20130101		...
 startYear	TestREST	Int32	2013		...

Рисунок 3.5 – Переменные пакета служб

Далее в потоке управления создадим контейнер «Цикл по элементам» для извлечения данных пакетам, в нашем случае данные будут извлекаться по годам. Каждая итерация будет содержать данные о продажах за 2 года, итого пройдет 4 итерации. Количество итераций можно менять, все зависит от производительности машины, где выполняется процесс ETL. Создадим задачи и задачи потоков данных: «Установит дату начала», «Установить дату окончания», «Извлечение данных» – извлекает данные из источника, «Обработка дублирующихся данных о продажах» – выполняет удаление данных из хранилища, если по каким-то причинам они попали в пакет второй и более раз,

«Обработка и загрузка данных (без справочников)» – данный поток выполняет саму обработку и загрузку данных в хранилище, «Установить следующий начальный год» – устанавливает следующий начальный год для фильтрации данных, «Установить следующий конечный год»– устанавливает следующий конечный год для фильтрации данных. Задачи потоков данных представлены на рисунке 3.6.

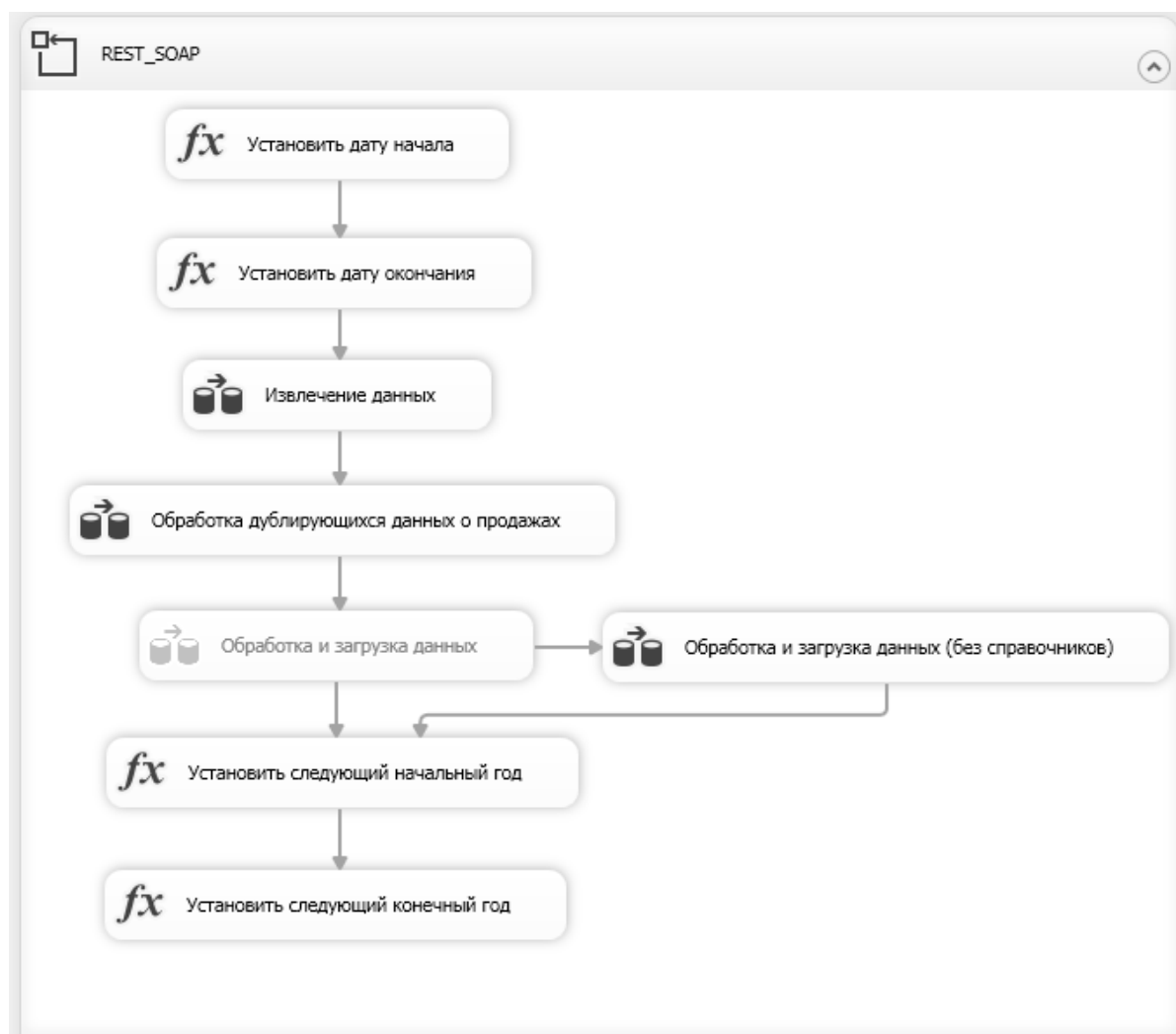


Рисунок 3.6 – Задачи потока данных RESTи SOAPрешения

Реализуем задачу потока данных «Извлечение данных» для источника данных в формате XML. Создадим источник данных «XMLSource», в нем укажем путь к веб-сервису «<http://localhost/mcenter/ws/sales.1cws>», установим HTTP-

команду «POST» и зададим тело запроса в формате XML, в котором укажем имя метода и значения параметров из переменных, код приведен в листинге 6.

Листинг 6 – Тело запроса в формате XML

```
<soap:Envelope xmlns:soap=http://www.w3.org/2003/05/soap-  
envelopexmlns:dwh="http://localhost/dwh">  
<soap:Header/>  
<soap:Body>  
<dwh:Get>  
<dwh:startDate>{{User::startDate}}</dwh:startDate>  
<dwh:endDate>{{User::endDate}}</dwh:endDate>  
<dwh:loadWithDirectories>{{ $Package::loadWithDirectories }}</dwh:loadWithDirectories>  
</dwh:Get>  
</soap:Body>  
</soap:Envelope>
```

Вся настройка источника данных «XMLSource» представлена на рисунке 3.7.

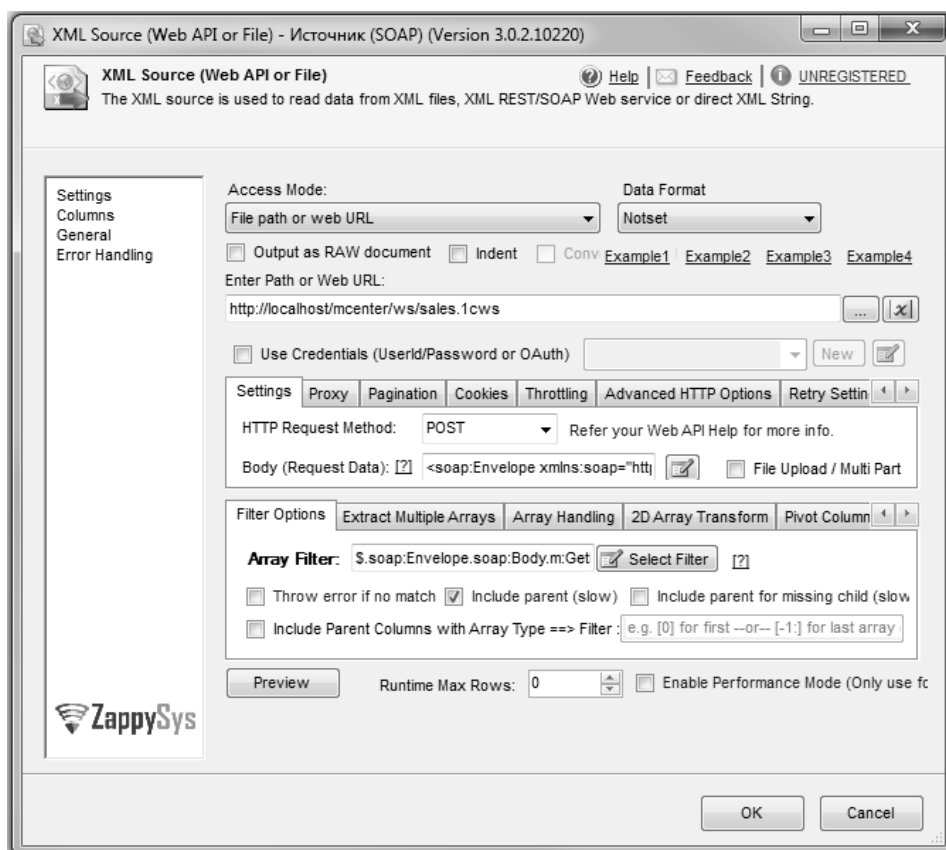


Рисунок 3.7 – Настройка источника данных «XMLSource»

Так же добавим в задачу потока данных назначение «Назначение (временный файл)» – для сохранения данных во временный файл, чтобы их можно было прочитать из разных задач потока данных и чтобы не выполнять запрос к веб-сервису еще раз. Вся задача потока данных представлена на рисунке 3.8.

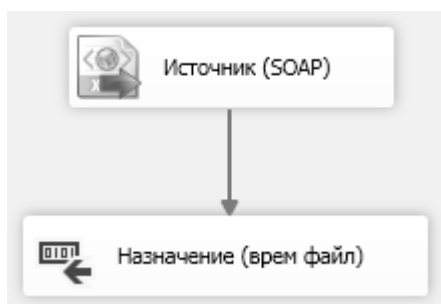


Рисунок 3.8 – Задача потока данных «Извлечение данных» XML

Реализуем задачу потока данных «Извлечение данных» для источника данных в формате JSON. Создадим источник данных «JSONSource», в нем укажем путь к веб-сервису «`http://localhost/mcenter/hs/Sales?startDate={{User::startDate}}&endDate={{User::endDate}}&loadWithDirectories={{Package::loadWithDirectories}}`».

Вся задача потока данных представлена на рисунке 3.9.



Рисунок 3.9 – Задача потока данных «Извлечение данных» JSON

После извлечения данных реализуем задачу потока данных «Обработка и загрузка данных», в нем создадим задачи: «Источник (временный файл)» – это извлеченные данные о продажах, источники данных таблиц измерений в хранилище, выполним сортировку данных по «sourceId», чтобы впоследствии

выполнить соединение источников и получить внутренние идентификаторы измерений в хранилище данных, после отсортируем все данные по столбцу «date» и загрузим их в хранилище с помощью задачи «Добавление данных о продажах».

Вся задача потока данных «Обработка и загрузка данных» представлена на рисунке 3.10.

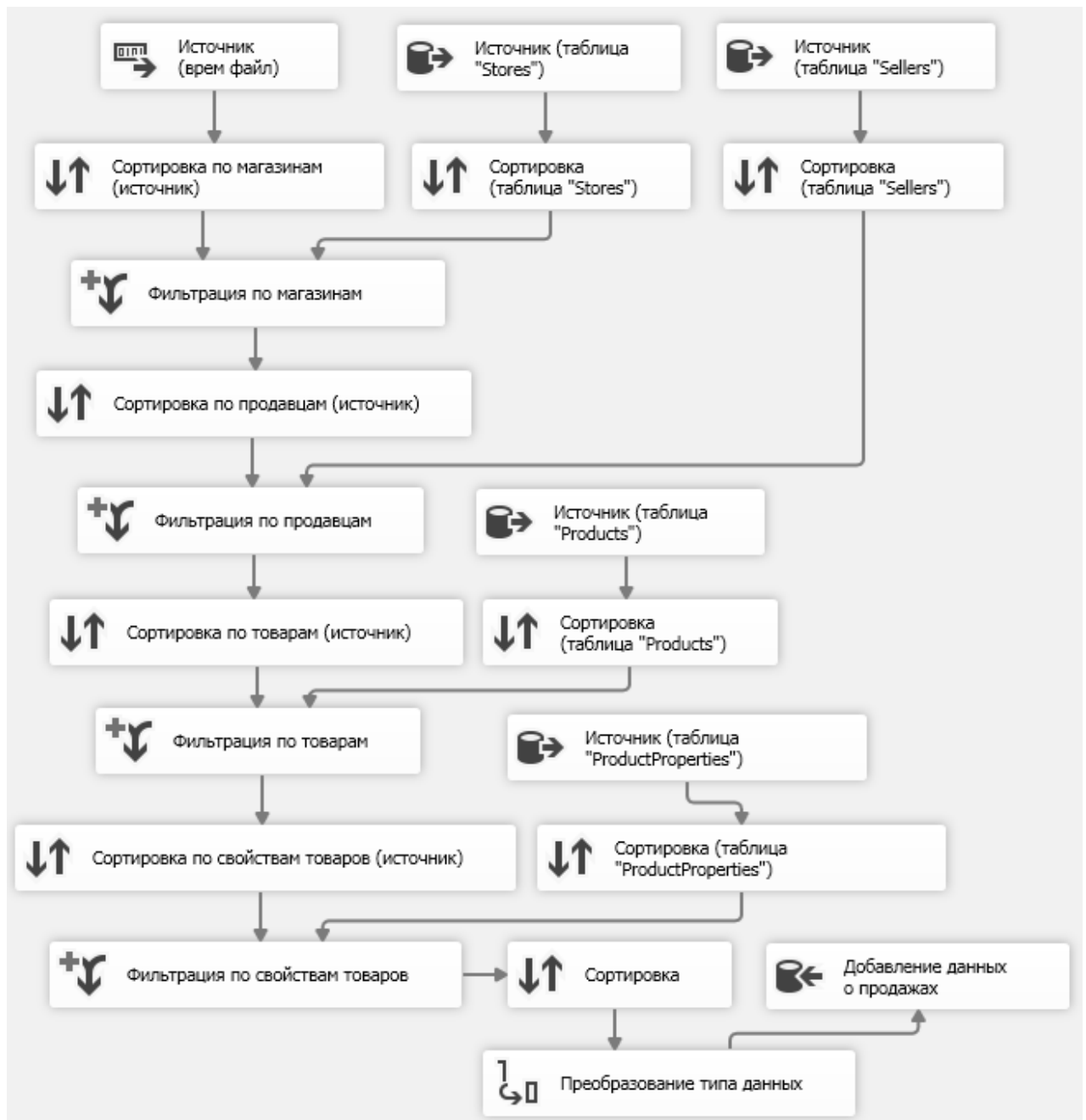


Рисунок 3.10 – Задача потока данных «Обработка и загрузка данных»

Выполним пакеты служб для обеих форматов данных XMLи JSON для того, чтобы протестировать решение и определить время длительности всего процесса, представлено на рисунке 3.11.

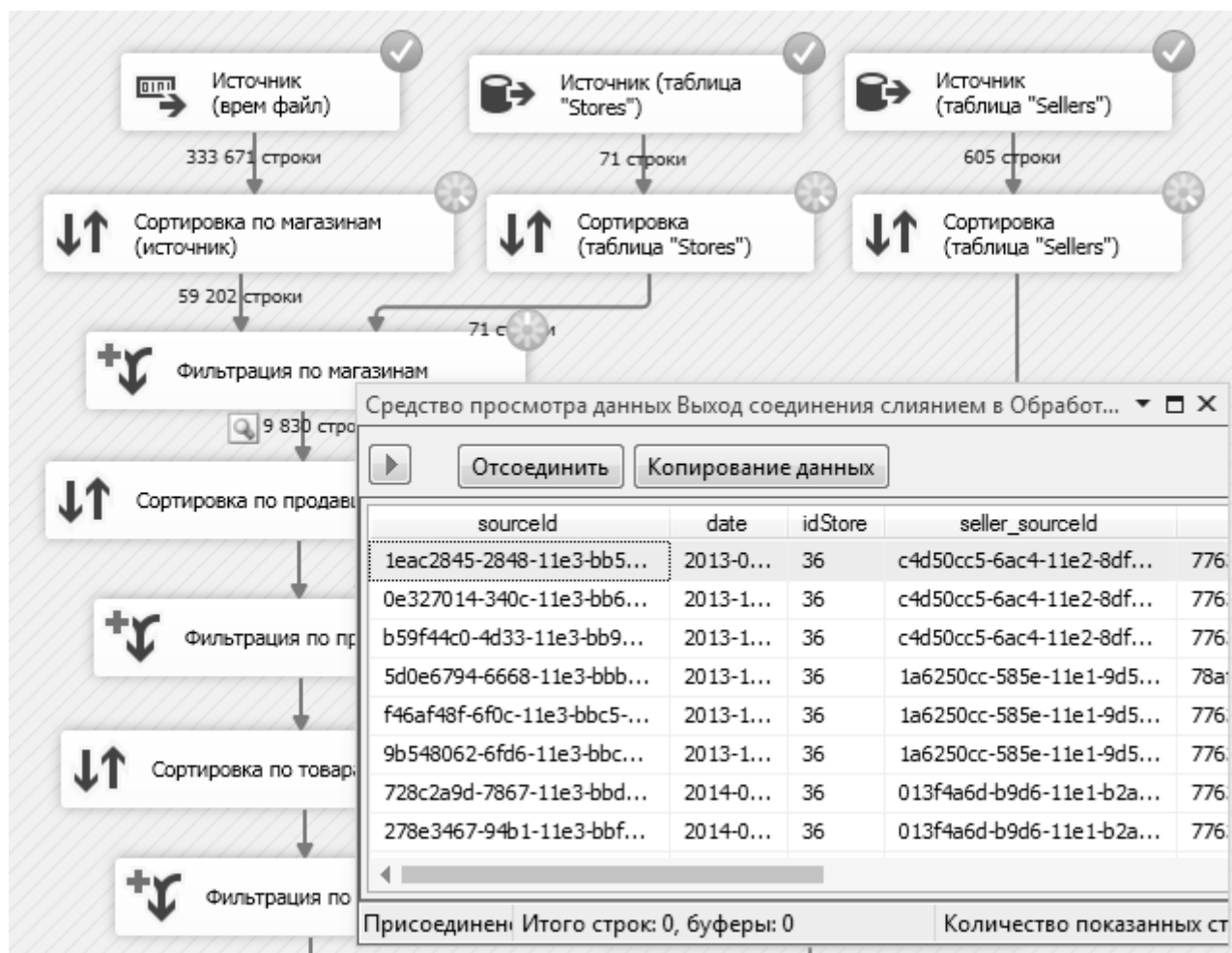


Рисунок 3.11 – Выполнение пакета служб исследуемых решений

Для тестирования была установлена точка останова после задачи «Фильтрация по магазинам» – она позволила увидеть промежуточные данные в процессе.

В итоге время всего процесса ETL, в котором источник представлен в формате данных XMLсоставило 17 минут, а в формате данныхJSON – 15 минут.

3.3 Публикация интерфейса OData на веб-сервере

Для выполнения публикации интерфейса OData необходимо запустить конфигуратор платформы 1С под правами администратора. Перейти в «Администрирование» - «Публикация на веб-сервере», в открывшемся окне выбрать веб-сервер, указать каталог и установить галочку «Публиковать стандартный интерфейс OData». Публикация представлена на рисунке 3.12.

Публикация на веб-сервере

Основные OpenID Прочие

Имя: mcenter

Веб-сервер: Apache 2.4

Каталог: C:\1c\mcenter\

☐ Публиковать тонкий клиент и веб-клиент

☒ Публиковать стандартный интерфейс OData

Web-сервисы HTTP сервисы

☐ Публиковать Web-сервисы по умолчанию

☒ Публиковать Web-сервисы:

	Имя	Адрес
<input type="checkbox"/>	Sales	Sales.1cws

☐ Публиковать Web-сервисы расширений по умолчанию

☐ Публиковать дистрибутив

Расположение публикуемого дистрибутива:

x86:

x86_64:

Адрес перехода при окончании работы веб-клиента:

Опубликовать

Отключить

Сохранить

Загрузить

Закрыть

Справка

Рисунок 3.12 – Публикация OData

После публикации автоматически сформируется файл конфигурации на веб-сервере со всеми указанными настройками и доступ к данным базы 1С будет открыт через REST-интерфейс. Код конфигурационного файла приведен в листинге 7.

Листинг 7 – Код конфигурационного файла веб-сервера

```
<?xml version="1.0" encoding="UTF-8"?>
<point xmlns="http://v8.1c.ru/8.2/virtual-resource-system"
        xmlns:xs="http://www.w3.org/2001/XMLSchema"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        base="/mcenter"
        ib="Srvr=&quot;localhost&quot;;Ref=&quot;mcenter&quot;;Usr=&quot;Администратор&quot;;Pwd=&quot;&quot;;"
        enable="false">
    <debug enable="true"
        protocol="tcp"
        url="tcp://localhost"/>
    <standardOdata enable="true"
        reuseSessions="autouse"
        sessionMaxAge="60"
        poolSize="10"
        poolTimeout="50"/>
</point>
```

В данном файле конфигурации можно увидеть настройки стандартного интерфейса OData.

3.4 Разработка ETL-приложения

В платформе 1С существует объект под названием «План обмена» – он используется для реализации механизмов обмена данными с внешними системами. План обмена содержит информацию об узлах, которые могут участвовать в обмене данными, определяет состав данных, которыми будет производиться обмен. Смысл этого механизма заключается в том, что когда мы добавляем в состав новый объект метаданных, например справочник «Номенклатура», то на уровне СУБД создается новая таблица, когда фиксирует любое изменение элементов этого справочника, т.е. к этой таблице можно обращаться и получить измененные объекты и загрузить их во внешнюю систему.

Для нашего решения создадим новый план обмена «Хранилище данных «Справочники» и наполним его состав для того, чтобы все новые изменения выгружались в хранилище:

1) Справочники: «ЗначенияСвойствОбъектов» – содержит значения дополнительных свойств товаров, будет использоваться для загрузки их в хранилище; «Номенклатура» – содержит информацию о товарах компании; «Склады» – содержит информацию о магазинах компании; «СР_Металл» – содержит информацию о металлах изделий; «СР_ЦветМеталла» – содержит информацию о цветах металлов изделий; «ФизическиеЛица»– содержит информацию о продавцах компании; «ХарактеристикиНоменклатуры» – содержит информацию о свойствах товаров.

2) План видов характеристик: «СвойстваОбъектов» – содержит информацию о дополнительных свойствах.

3) Регистры сведений: «СР_ДопЗначенияСвойствИзделий» – содержит информацию о дополнительных свойствах товаров; «СР_ЗначенияСвойствИзделий»– содержит информацию о преопределенных свойствах товаров.

Состав плана обмена представлен на рисунке 3.13.

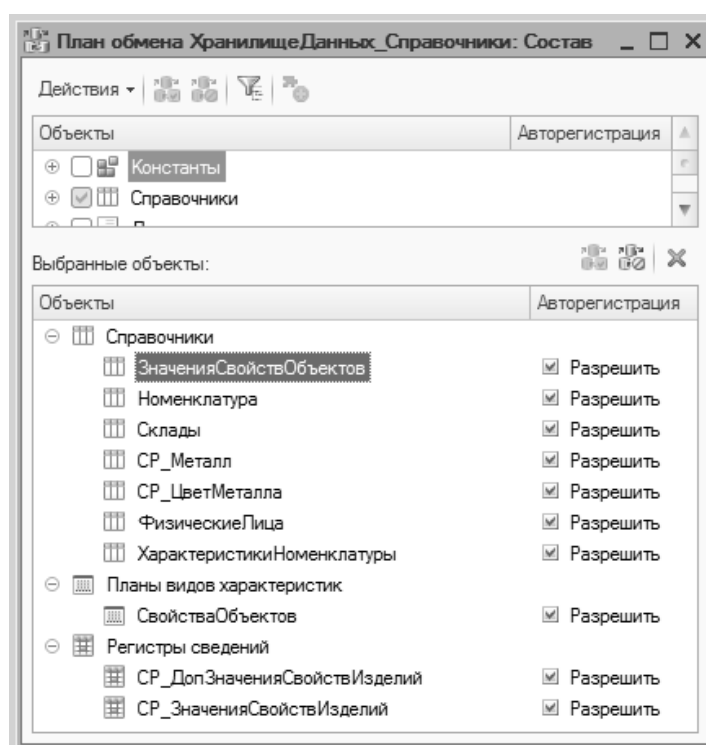


Рисунок 3.13 – Состав плана обмена

Стандартный интерфейс OData имеет свою схему описания данных в формате XML, которую можно получить следующим HTTP-запросом «[http://localhost/mcenter/odata/standard.odata/\\$metadata](http://localhost/mcenter/odata/standard.odata/$metadata)». Она содержит в себе описание каждого объекта конфигурации, его реквизитов и их типов, а так же методы по которым можно к ним обращаться. Приведем пример описания метаданных регистра накопления «ПродажиСебестоимость», который потребуется нам в дальнейшем, чтобы получать себестоимость проданных товаров, код приведен в листинге 8.

Листинг 8 – Метаданные регистра накопления «ПродажиСебестоимость»

```
<EntityType Name="AccumulationRegister_ПродажиСебестоимость_RecordType"
  OpenType="true">
  <Key>
    <PropertyRef Name="Recorder"/>
    <PropertyRef Name="LineNumber"/>
    <PropertyRef Name="Recorder_Type"/>
  </Key>
  <Property Name="Recorder" Type="Edm.String" Nullable="false"/>
  <Property Name="Period" Type="Edm.DateTime" Nullable="true"/>
  <Property Name="LineNumber" Type="Edm.Int64" Nullable="false"/>
  <Property Name="Active" Type="Edm.Boolean" Nullable="true"/>
  <Property Name="Номенклатура_Key" Type="Edm.Guid" Nullable="true"/>
  <Property Name="ХарактеристикаНоменклатуры_Key" Type="Edm.Guid"
    Nullable="true"/>
  <Property Name="ЗаказПокупателя" Type="Edm.String" Nullable="true"/>
  <Property Name="ДокументОприходования" Type="Edm.String" Nullable="true"/>
  <Property Name="Подразделение_Key" Type="Edm.Guid" Nullable="true"/>
  <Property Name="Количество" Type="Edm.Double" Nullable="true"/>
  <Property Name="Стоимость" Type="Edm.Double" Nullable="true"/>
  <Property Name="Масса" Type="Edm.Double" Nullable="true"/>
  <Property Name="СписаниеПартий" Type="Edm.Boolean" Nullable="true"/>
  <Property Name="ДокументДвижения" Type="Edm.String" Nullable="true"/>
  <Property Name="ДокументДвиженияПериод" Type="Edm.DateTime"
    Nullable="true"/>
  <Property Name="Recorder_Type" Type="Edm.String" Nullable="false"/>
  <Property Name="ЗаказПокупателя_Type" Type="Edm.String" Nullable="true"/>
  <Property Name="ДокументОприходования_Type" Type="Edm.String"
    Nullable="true"/>
  <Property Name="ДокументДвижения_Type" Type="Edm.String" Nullable="true"/>
</EntityType>
```


По данному описанию специалист ETL может понять внутреннюю структуру таблиц.

Интерфейс ODataв HTTP-запросе поддерживает написание следующих параметров, которые нам пригодятся при извлечении данных:

- 1) «\$format» – указывает формат возвращаемых данных: XMLили JSON;
- 2) «\$filter» – отбор при извлечении данных;
- 3) «\$select» – перечисление свойств (столбцов) объекта метаданных, которые попадут в результат запроса;
- 4) «\$orderby» – сортировка результата запроса.

Создадимпроект «IntegrationServicesProject» вVisualStudio и в нем создадим пакет служб «ETL». Для начала разработаем поток управления по извлечению, обработки и загрузки справочников в хранилище, так как сначала мы должны наполнить наши измерения в хранилище, а потом уже загружать факты. Данный поток будет находится в контейнере «Последовательность», так как всезадачи потока данных нужно выполнить последовательно.

Для начала необходимо создать параметры пакета:

- 1) «startDate» – дата начала извлечения данных о продажах;
- 2) «endDate» – дата окончания извлечения данных о продажах;
- 3) «isService» – признак указывает в каком режиме находится приложение: в режиме службы или нет;
- 4) «forLastCountDays» – указывает за какое количество дней извлекаем данные о продажах.

Так же создадим необходимые переменные:

- 1) «startDate» – дата начала извлечения данных о продажах;
- 2) «endDate» – дата окончания извлечения данных о продажах;
- 3) «messageNumberDirectory» – номер сообщения пакета данных справочников из плана обмена.

Наполним поток управления по извлечению, обработки и загрузки справочников задачами потока данных.

Задача «Получить номер последнего загруженного пакета данных справочников» – выполняет запрос «`http://localhost/mcenter/odata/standard.odata/ExchangePlan_ХранилищеДанных_Справочники?$format=json;odata=nometadata&$select=SentNo&$filter=Code eq '002'`» к плану обмена, чтобы получить последний номер отправленного пакета данных для установки следующего номера.

Задача «Установить следующий номер сообщения пакета данных справочников» – увеличивает значение переменной «`messageNumberDirectory`» на один.

Задача потока данных «Извлечение, обработка и загрузка справочников» – основной поток, так как он выполняет извлечение данных и отправку их по таблицам хранилища. Создадим и опишем задачи этого потока.

Создадим задачу с источником «Источник (план обмена "ХранилищеДанных_Справочники")» для извлечения данных с запросом «`http://localhost/mcenter/odata/standard.odata/SelectChanges?DataExchangePoint='http://localhost/mcenter/odata/standard.odata/ExchangePlan_ХранилищеДанных_Справочники(guid'f7d37419-6911-11ea-815c-40618617ecfa')&MessageNo={{User::messageNumberDirectory}}&$format=json;odata=nometadata`» – извлекает все измененные элементы справочников из плана обмена.

Так как наш источник содержит в себе сразу несколько справочников, чтобы разделить их на несколько потоков для загрузки в разные таблицы хранилища создадим задачу «Разделение по таблицам хранилища данных» – позволяет с помощью отборов распределить данные на разные потоки. После выполнения этой задачи получаться 11 потоков:

- 1) «Products» – данные о товарах;
- 2) «Stores» – данные о магазинах;

- 3) «ProductProperties» – данные о свойствах товаров;
- 4) «Metals» – данные о металлах изделий;
- 5) «MetalColors» – данные о цветах металлов изделий;
- 6) «DetailedProductProperties» – данные о predetermined свойствах товаров;
- 7) «Sellers» – данные о продавцах;
- 8) «AdditionalProperties» – данные о дополнительных свойствах;
- 9) «AdditionalPropertyValues» – данные о значениях дополнительных свойств;
- 10) «AdditionalProductPropertyValues» – данные о значениях дополнительных свойств товаров;
- 11) «DeletedInformationRegisterEntries» – данные об удаленных записей таблиц.

Настройка задачи представлена на рисунке 3.14.

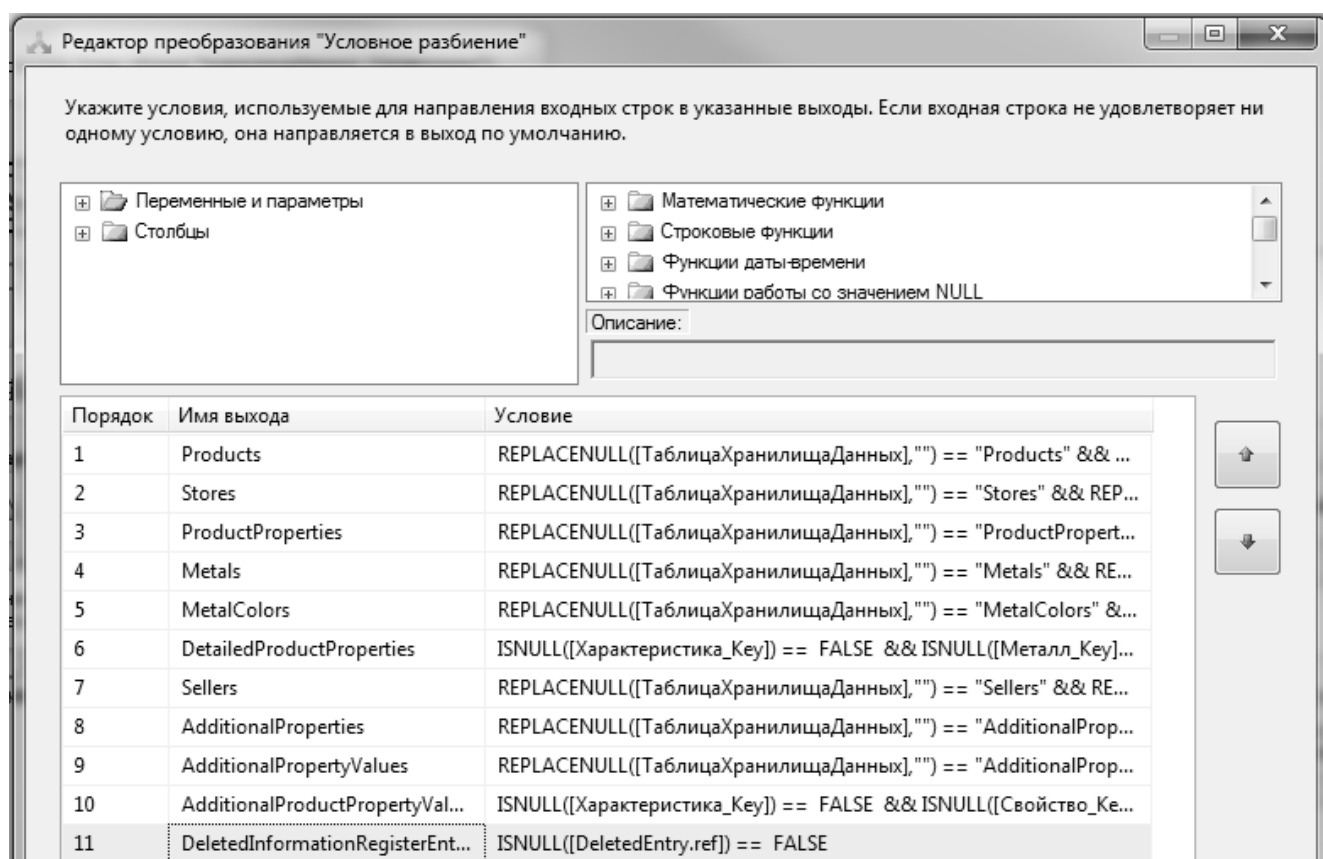


Рисунок 3.14 – Настройка задачи «Разделение по таблицам хранилища данных»

Далее нам необходимо понять, есть ли данное значение в таблице хранилища данных и если оно есть, то изменить его, иначе добавить новое. Для этого создадим задачу «Разделение на добавление и изменение» для потоков данных: «Products», «Sellers», «Stores», «Metals», «MetalColors», «AdditionalProperties» и «AdditionalPropertyValues», так как они не имеют внешних ключей. Поиск данных в таблице хранилища выполняется по полю «sourceId». Настройка задачи представлена на рисунке 3.15.

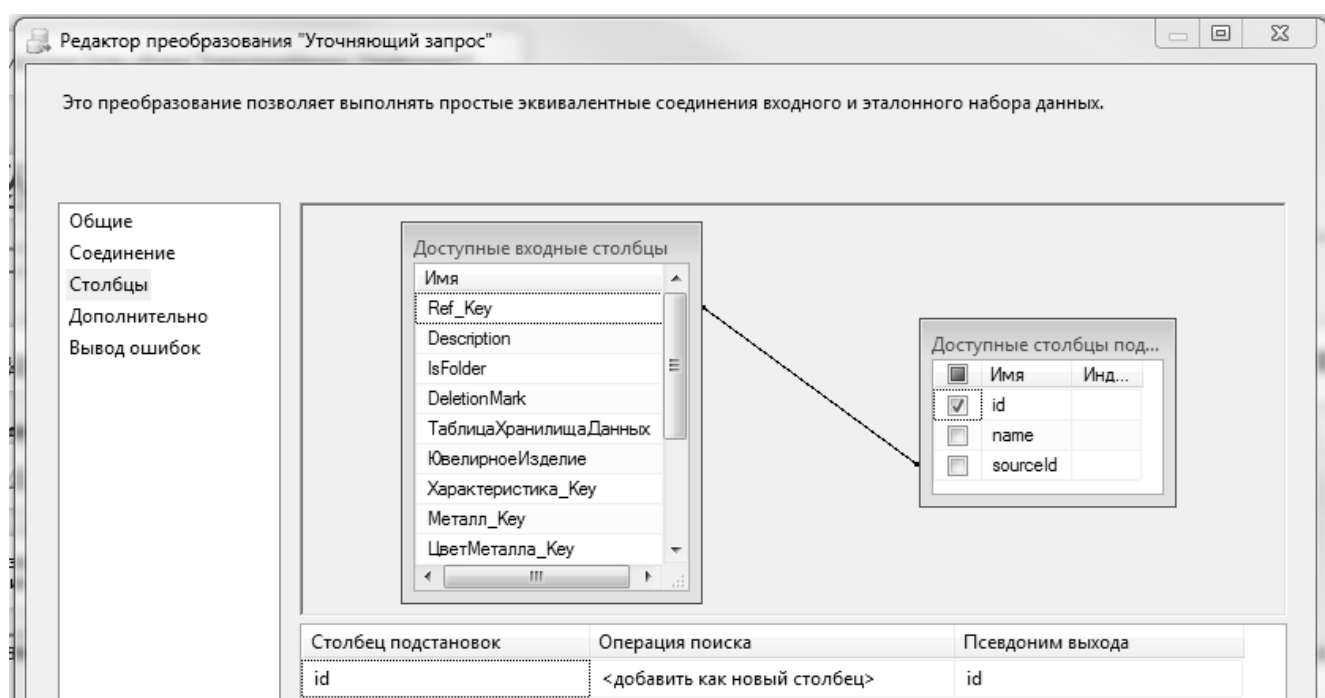


Рисунок 3.15 – Настройка задачи «Разделение на добавление и изменение»

После выполнения этой задачи получаться два потока данных: данные которые нужно изменить и добавить. Для этого создадим задачи: «Добавление» и «Изменение» в соответствующие таблицы хранилища данных. Потоки данных «ProductProperties», «DetailedProductProperties», «AdditionalProductPropertyValues» запишем во временные файл и загрузим их в следующих задачах потока данных, так как они имеют внешние ключи.

Вся задача потока данных «Извлечение, обработка и загрузка справочников» представлена на рисунке 3.16.

Создадим задачу потока данных «Обработка и загрузка свойств товаров» для обработки и загрузки свойств в таблицу хранилища, представлено на рисунке 3.17.

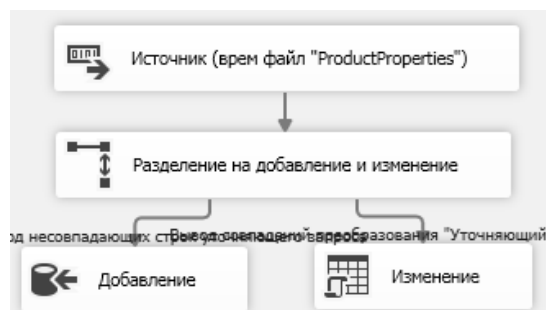


Рисунок 3.17 – Задача потока данных «Обработка и загрузка свойств товаров»

Создадим задачу потока данных «Обработка и загрузка типовых свойств товаров» для обработки и загрузки металла и его цвета в таблицу «ProductProperties», представлено на рисунке 3.18.

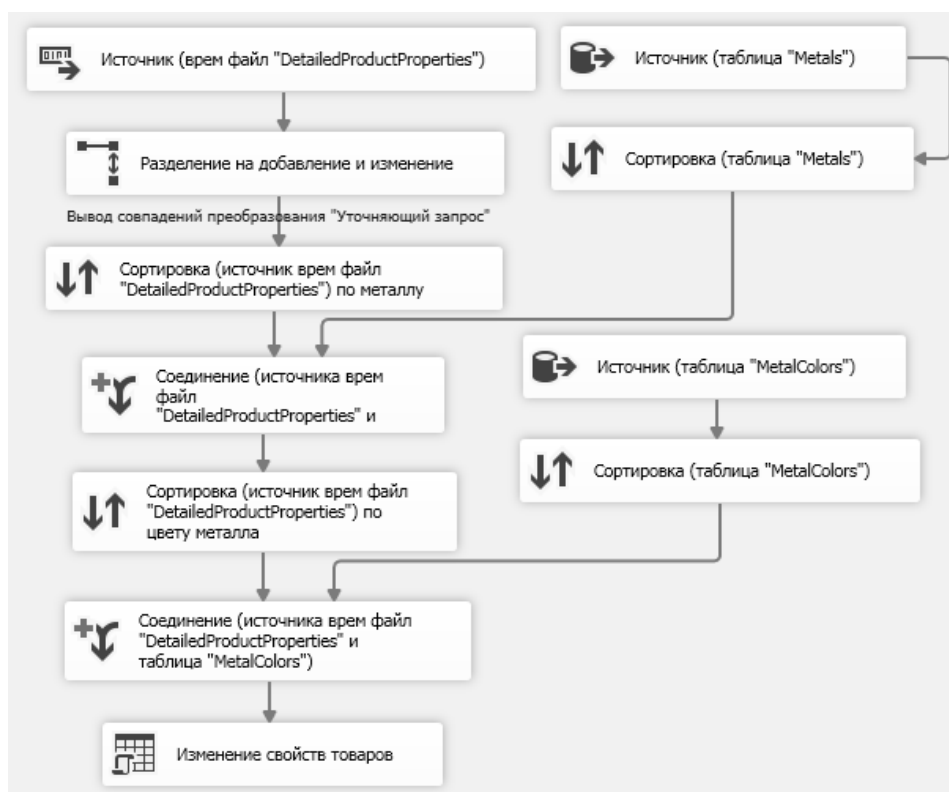


Рисунок 3.18 – Задача потока данных «Обработка и загрузка типовых свойств товаров»

Создадим задачу потока данных «Обработка и загрузка доп свойств товаров» для обработки и загрузки дополнительных свойств товаров, представлено на рисунке 3.19.

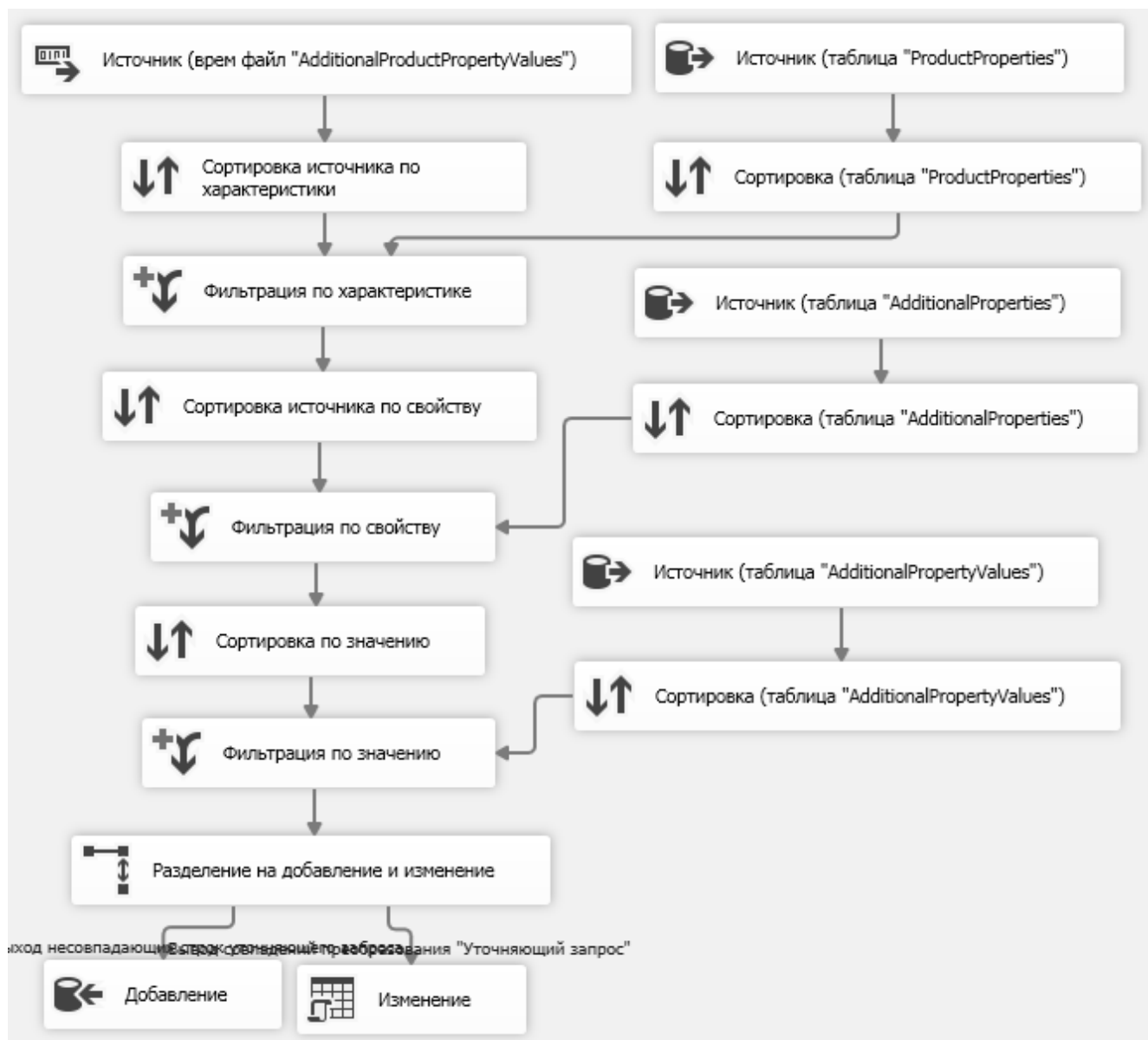


Рисунок 3.19 – Задача потока данных «Обработка и загрузка доп свойств товаров»

После загрузки всех справочников в хранилище необходимо оповестить план обмена базы 1С об успешном окончании загрузки. Для этого создадим задачу «Уведомление источника о загрузке данных справочников» с запросом «<http://localhost/mcenter/odata/standard.odata/NotifyChangesReceived?DataExchangeP>

oint='http://localhost/mcenter/odata/standard.odata/ExchangePlan_ХранилищеДанных_Справочники(guid'f7d37419-6911-11ea-815c-40618617ecfa')&MessageNo={{User::messageNumberDirectory}}», который отправить сообщение в план обмена и снимет все зарегистрированные изменения справочников. На этом процесс по извлечению, обработки и загрузки справочников в хранилище окончен, параллельно с этим процессом должно выполняться извлечение данных о продажах – перейдем к его разработке.

Для того, чтобы получать данные о продажах за какой-либо период создадим две выполняющиеся последовательно задачи «Установить дату начала» и «Установить дату окончания». Далее создадим задачу потока данных «Извлечение данных о продажах» – в ней будет два источника данных:

1) «Источник (документы "Отчет о розничных продажах")» с запросом «http://localhost/mcenter/odata/standard.odata/Document_ОтчетОРозничныхПродажах?\$format=json;odata=nometadata&\$select=Ref_Key, Date, Склад_Key, Товары/Номенклатура_Key, Товары/ХарактеристикаНоменклатуры_Key, Товары/Количество, Товары/Цена, Товары/Сумма, Товары/Продавец1&\$filter=Posted eq true and Склад_Key ne guid'00000000-0000-0000-0000-000000000000' and Товары/Номенклатура_Key ne guid'00000000-0000-0000-0000-000000000000' and Товары/ХарактеристикаНоменклатуры_Key ne guid'00000000-0000-0000-0000-000000000000' and cast(Товары/Продавец1, 'Catalog_ФизическиеЛица') ne guid'00000000-0000-0000-0000-000000000000' and Товары/Количество gt 0 and Date ge datetime'{{\$Package::startDate}}' and Date le datetime'{{\$Package::endDate}}'&\$orderby=Date asc», который извлекает данные о продажах из документов за определенный период с отбором по заполненным данным и с сортировкой по дате;

2) «Источник (регистр накопления "Продажи себестоимость")» с запросом «http://localhost/mcenter/odata/standard.odata/AccumulationRegister_ПродажиСебестоимость_RecordType?\$format=json;odata=nometadata&\$select=Recorder,

Номенклатура_Key, ХарактеристикаНоменклатуры_Key, Стоимость, Магга&\$filter=Period ge datetime'{{\$Package::startDate}}' and Period le datetime'{{\$Package::endDate}}' and Номенклатура_Key ne guid'00000000-0000-0000-0000-000000000000' and ХарактеристикаНоменклатуры_Key ne guid'00000000-0000-0000-0000-000000000000' and Номенклатура/ЮвелирноеИзделие eq true and Количество gt 0&\$orderby=Period asc», который извлекает данные о себестоимости проданных товаров.

Далее создадим задачу «Соединение», чтобы соединить два источника данных по документу, номенклатуре и ее характеристики. После рассчитаем стоимость продажи без скидок и запишем во временный файл, так как процесс выполняется параллельно и данные о продажах нам нужно загрузить после загрузки справочников. Вся задача потока данных представлена на рисунке 3.20.



Рисунок 3.20 – Задача потока данных «Извлечение данных о продажах»

Далее создадим задачу потока данных «Обработка дублирующихся данных о продажах» – это очистка данных, если по каким-то причинам в выборку попал документ, который уже был загружен в хранилище, представлено на рисунке 3.21.

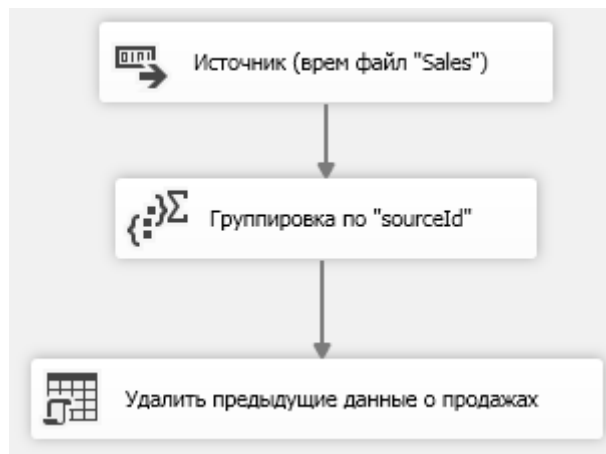


Рисунок 3.21 – Задача потока данных «Обработка дублирующихся данных о продажах»

После обработки дублей создадим задачу потока данных «Обработка и загрузка данных о продажах», представлена на рисунке 3.10. После загрузки всех данных нам необходимо сформировать представление «СвойстваТоваров», чтобы оно содержало в себе так же дополнительные свойства развернутые по столбцам. Для этого создадим хранимую процедуру «CreateViewProductProperties», которую будем вызывать заданием. Данная процедура составляет текст запроса по дополнительным свойствам товаров. Каждое свойство – это вложенная таблица. После составления запроса процедура создает представление с этим запросом. Код процедуры приведен в листинге 9.

Листинг 9 – Код процедуры создания представления

```

CREATEPROCEDURE [dbo].[CreateViewProductProperties]
AS
BEGIN
    DECLARE
        @viewName nvarchar(max)="",
        @queryView nvarchar(max)="",
        @counter smallint= 0,
        @querySelect nvarchar(max)="",
        @queryFrom nvarchar(max)="",
        @totalQuery nvarchar(max)="
    'SELECT
        ProductProperties.id AS Идентификатор,
        LTRIM(RTRIM(ProductProperties.name)) AS Наименование,
        LTRIM(RTRIM(COALESCE(Metals.name, ""))) AS Металл,
  
```

Продолжение листинга 9

```

        LTRIM(RTRIM(COALESCE(MetalColors.name, ''))) AS [Цветметалла],
        ProductProperties.size AS Размер,
        @selectAdditionalProperties
FROM
    dbo.ProductProperties AS ProductProperties LEFT OUTER JOIN
    dbo.MetalColors AS MetalColors ON ProductProperties.idMetalColor =
    MetalColors.id LEFT OUTER JOIN
    dbo.Metals AS Metals ON ProductProperties.idMetal = Metals.id LEFT OUTER
    JOIN
    @fromAdditionalProperties';
SELECT
    @querySelect = @querySelect +'COALESCE(AdditionalProperty'+
    RTRIM(LTRIM(CAST(@counter ASchar)))+' .name, ''') AS
    ['+RTRIM(LTRIM(AdditionalProperties.name))+'],
    ',
    @queryFrom = @queryFrom +
    '(SELECT
        AdditionalProductPropertyValues.idProductProperty AS id,
        RTRIM(LTRIM(AdditionalPropertyValues.name)) AS name
    FROM
        dbo.AdditionalProductPropertyValues AS AdditionalProductPropertyValues
        LEFT OUTER JOIN
        dbo.AdditionalPropertyValues AS AdditionalPropertyValues ON
        AdditionalProductPropertyValues.idAdditionalPropertyValue =
        AdditionalPropertyValues.id
    WHERE
        idAdditionalProperty = '+RTRIM(LTRIM(CAST(AdditionalProperties.id
        ASchar)))+' AS AdditionalProperty'+RTRIM(LTRIM(CAST(@counter
        ASchar)))+' ON ProductProperties.id =
        AdditionalProperty'+RTRIM(LTRIM(CAST(@counter ASchar)))+' .id LEFT
        OUTER JOIN
    ',
    @counter += 1
FROM
    dbo.AdditionalPropertiesAS AdditionalProperties
IF @querySelect = "
    SET @querySelect = @querySelect;
ELSE
    SET @querySelect =LEFT(@querySelect,LEN(@querySelect)- 4);
IF @queryFrom = "
    SET @queryFrom = @queryFrom;
ELSE
    SET @queryFrom =LEFT(@queryFrom,LEN(@queryFrom)- 19);
IF @querySelect = "
    SET @totalQuery =REPLACE(@totalQuery,',
    @selectAdditionalProperties','');
ELSE
    SET @totalQuery =REPLACE(@totalQuery,'@selectAdditionalProperties',
    @querySelect);

```

```

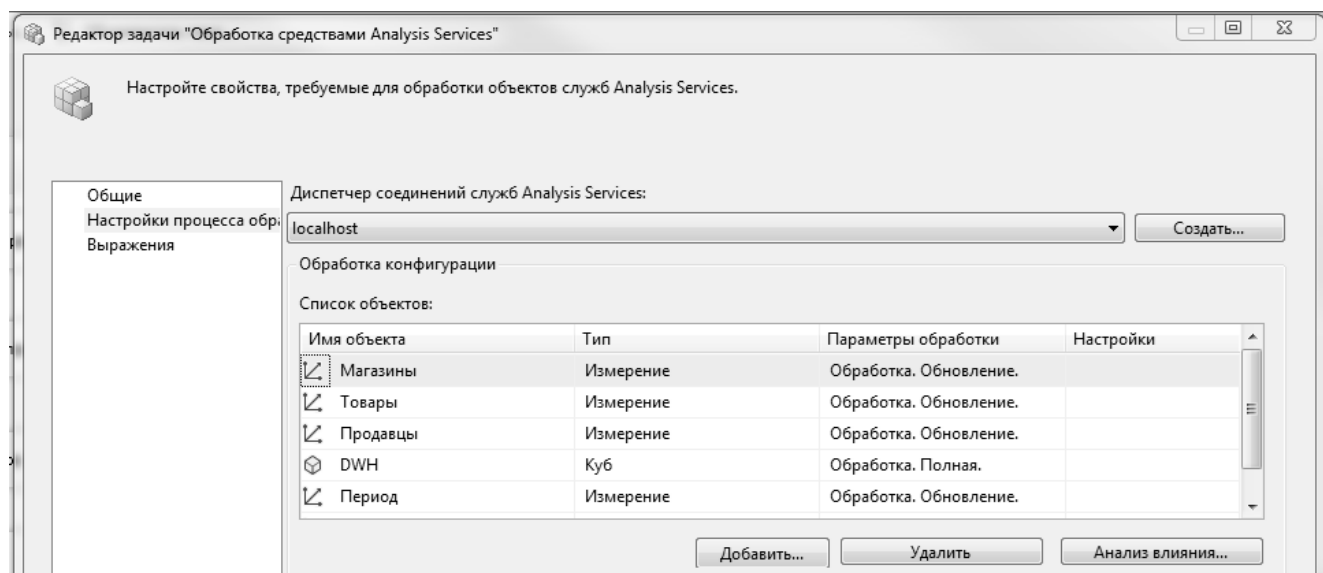
IF @queryFrom="
Окончание листинга 9

SET @totalQuery =REPLACE(@totalQuery,' LEFT OUTER JOIN
@fromAdditionalProperties','');
ELSE
SET @totalQuery =REPLACE(@totalQuery,'@fromAdditionalProperties',
@queryFrom);
SET @viewName =
'SELECT TABLE_NAME FROM INFORMATION_SCHEMA.VIEWS
WHERE TABLE_NAME = "СвойстваТоваров"';
IF @viewName ISNULL
SET @queryView =
'CREATE VIEW СвойстваТоваров AS @select';
ELSE
SET @queryView =
'ALTER VIEW СвойстваТоваров AS @select';
SET @queryView =REPLACE(@queryView,'@select', @totalQuery);

EXECUTEsp_executesql@queryView;
END

```

Так же для того, чтобы обновить данные в OLAP-кубе создадим задачу «Обновление OLAP-куба», так как автоматически новые добавленные данные в хранилище не попадут в куб. Настройки задачи представлены на рисунке 3.22.



Разработка ETL-приложения завершена. Весь разработанный пакет служб SSIS представлен на рисунке 3.23.

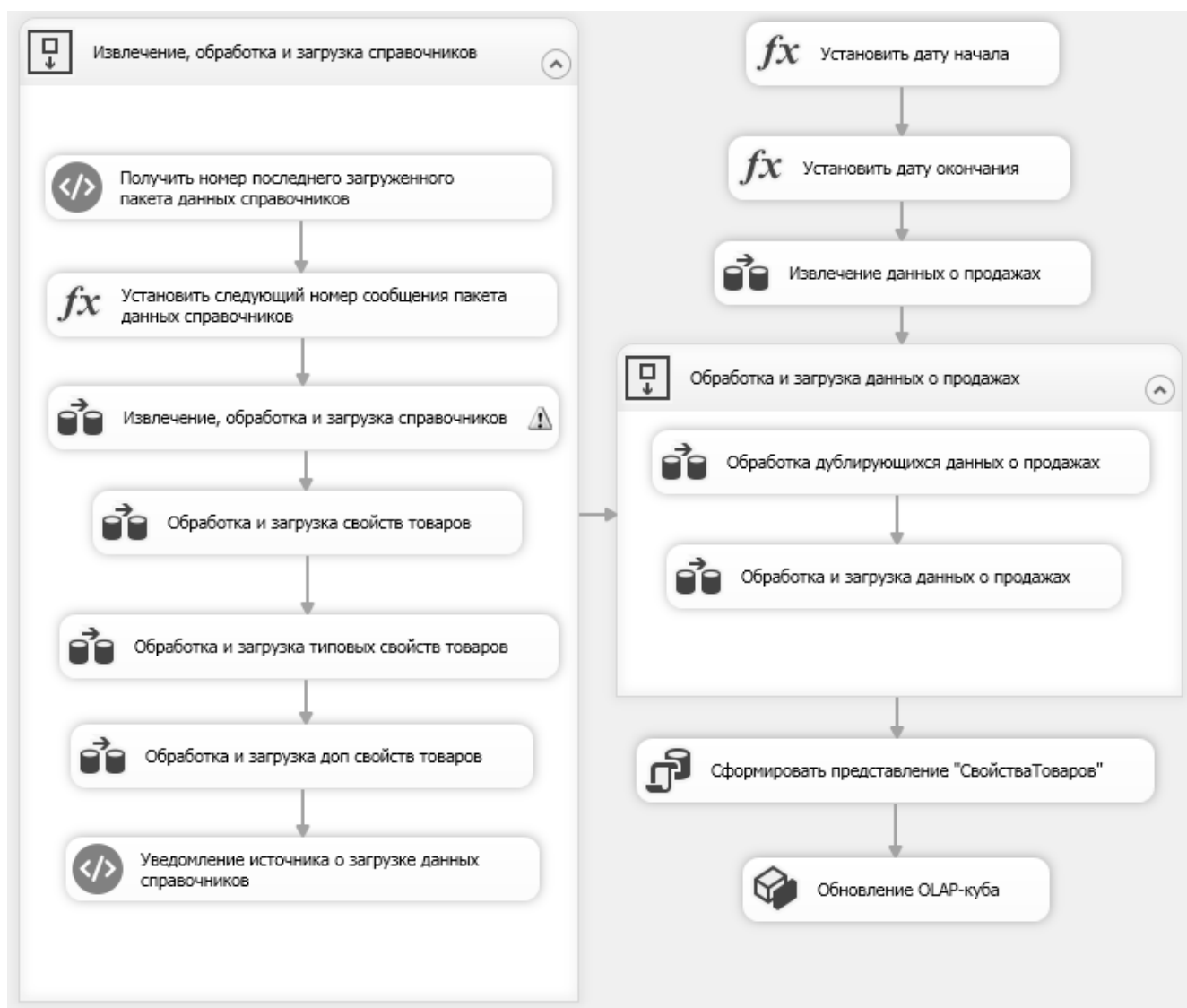


Рисунок 3.23 – Пакет служб SSIS

После разработки приступим к тестированию приложения по извлечению, обработки и загрузки данных в хранилище.

3.5 Тестирование ETL-приложения

Для тестирования извлечения, обработки и загрузки справочников установим точку основа внутри задачи потока данных «Извлечение, обработка и загрузка справочников» после задачи «Источник (план обмена

"ХранилищеДанных_Справочники")» для того, чтобы увидеть все извлеченные справочники, представлено на рисунке 3.24.

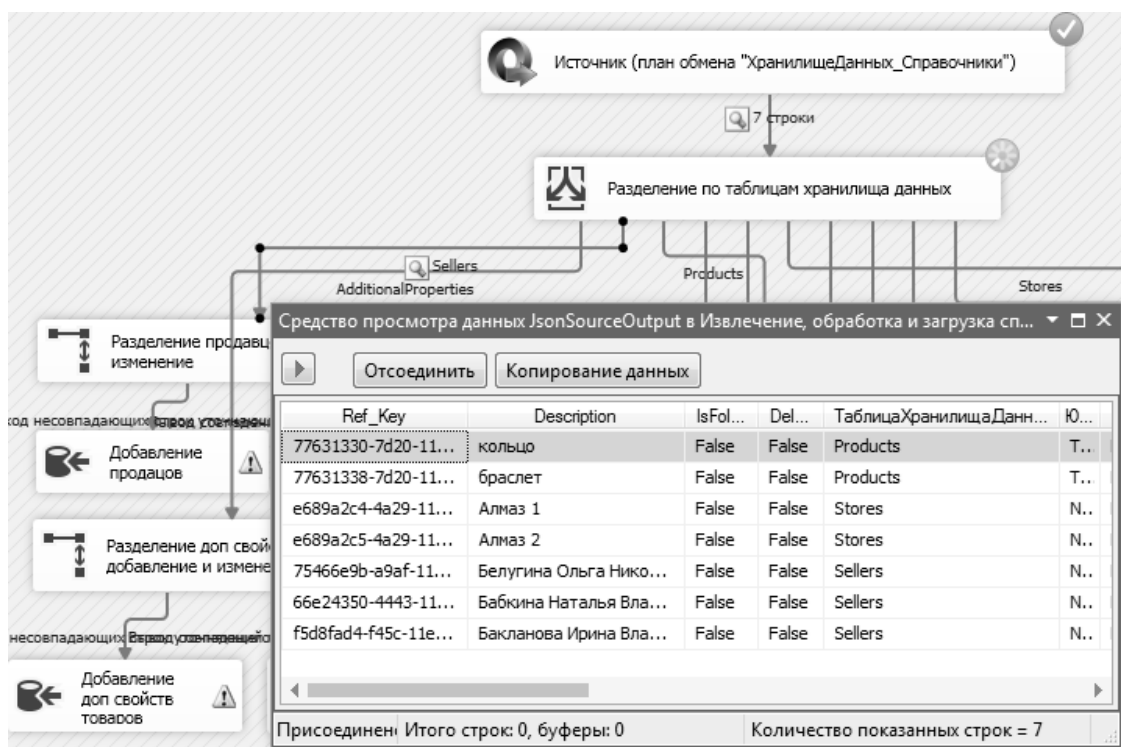


Рисунок 3.24 – Тестирование загрузки справочников 1

Как мы можем увидеть извлеклись данные о товарах, магазинах и продавцах. Поставим точку останова после задачи «Разделение по таблицам хранилища данных», чтобы увидеть, что данные разделились, представлено на рисунке 3.25.

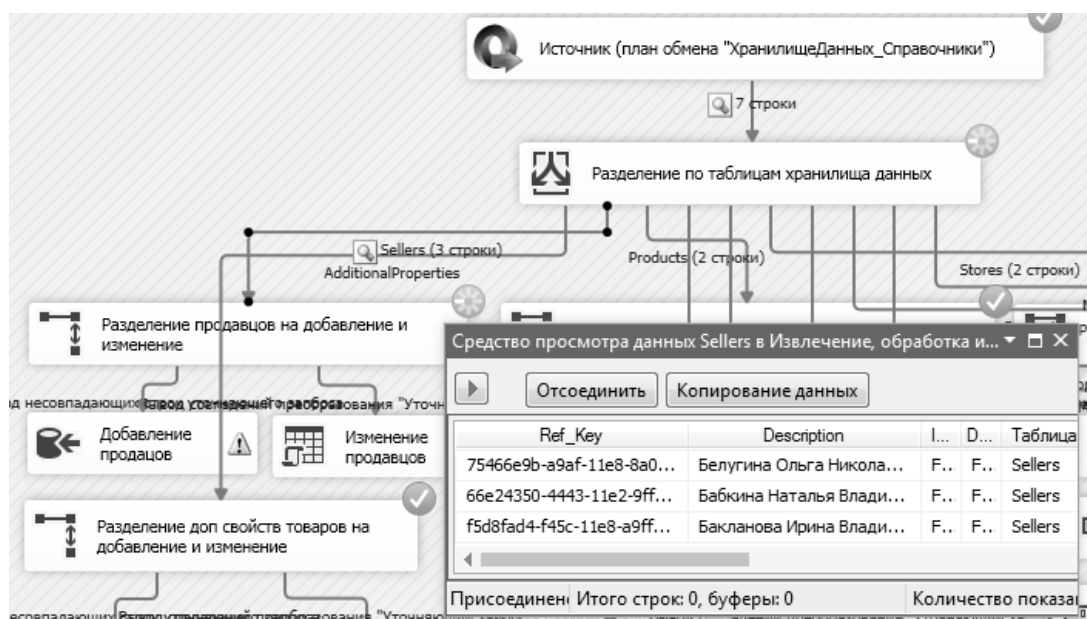


Рисунок 3.25 – Тестирование загрузки справочников 2

На рисунке видно, что 3 строки попали в таблицу «Sellers», 2 в таблицу «Stores» и 2 в таблицу «Products».

Так же проведем тестирование извлечения данных о продажах. В задаче потока данных «Извлечение данных о продажах» поставим точку останова после задачи «Расчет стоимости без скидки», представлено на рисунке 3.26.

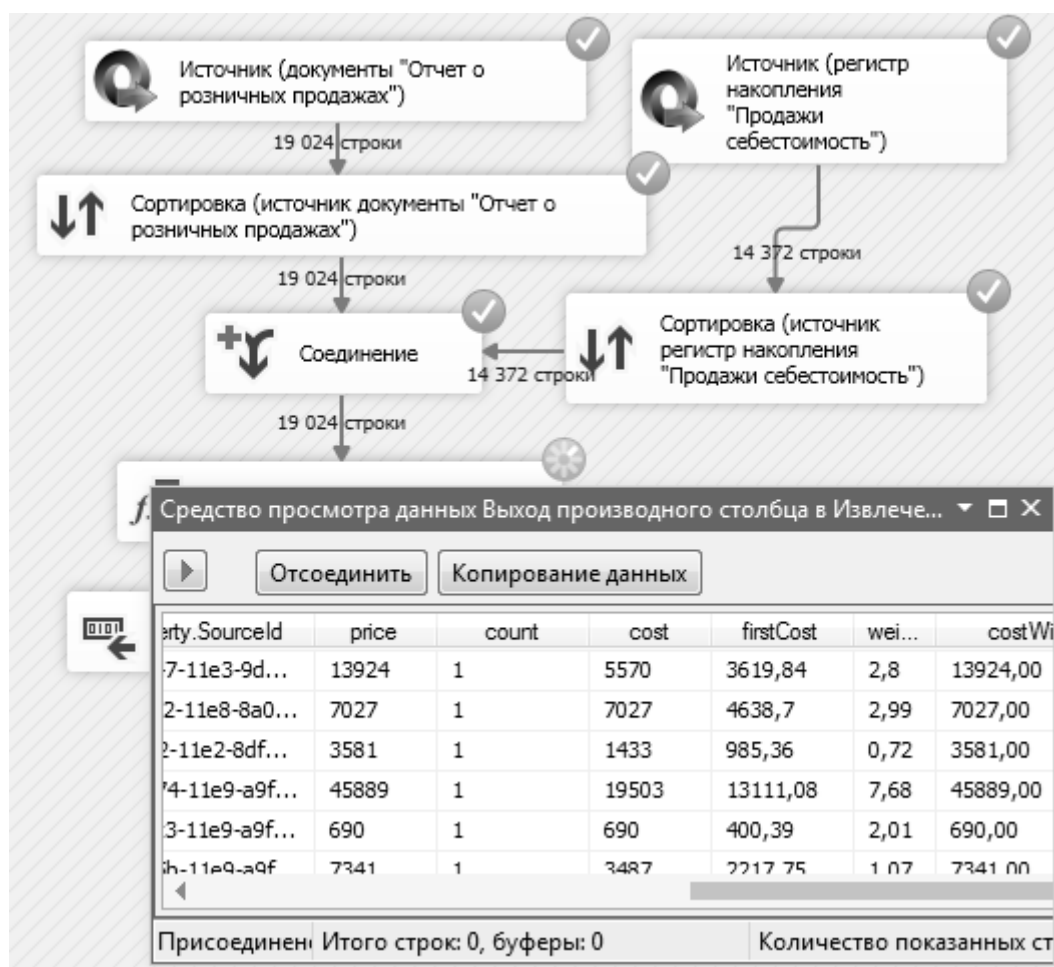


Рисунок 3.26 – Тестирование загрузки данных о продажах

На рисунке видно, что для каждого товара так же извлеклась его себестоимость. Время загрузки 1 000 000 строк записей составило 13 минут. После наполнения хранилища данных приступим к разработке OLAP-куба.

3.6 Разработка OLAP-куба на основании хранилища данных

Перед разработкой проекта OLAP-куба необходимо создать понятные для пользователя представления данных: «Магазины», «Продавцы», «Товары», «Свойства Товаров» и «Продажи».

Создадим новый проект многомерных данных и интеллектуального анализа данных служб AnalysisServices в Visual Studio. Для начала необходимо выбрать источники данных и задать логические ключи между ними. В нашем случае – это представления данных, которые мы создали ранее, представлено на рисунке 3.27.

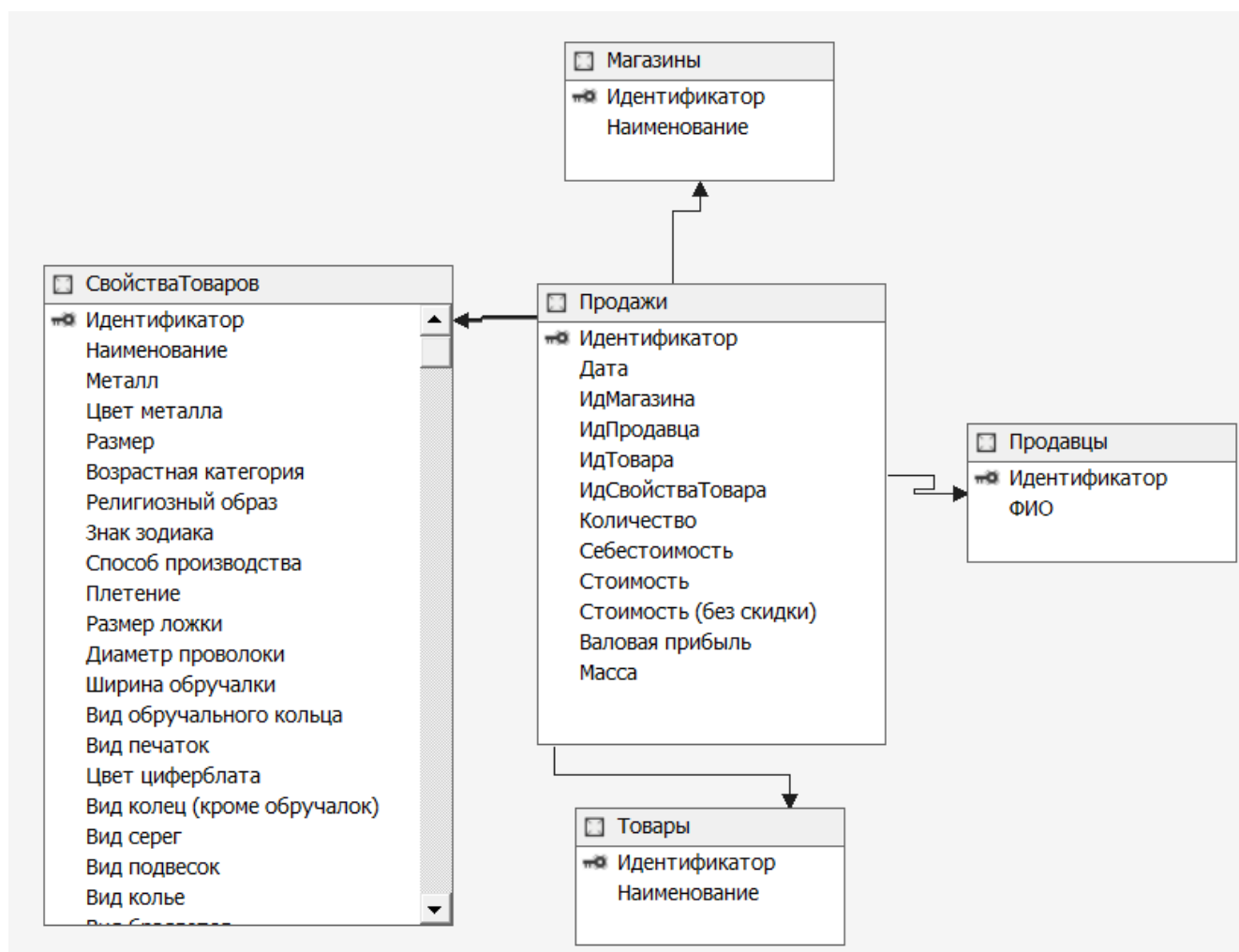


Рисунок 3.27 – Представления источников данных

На основе этого представления источников создадим куб. Таблицами измерения будут: «Магазины», «Продавцы», «Товары», «Свойства товаров» и «Период» – будет создана на сервере SSAS. Таблица фактов – «Продажи». Все таблицы представлены на рисунке 3.28.

Группы мер	
Измерения	Продажи
Магазины	Идентификатор
Продавцы	Идентификатор
Товары	Идентификатор
Период	Дата
Свойства товаров	Идентификатор

Рисунок 3.28 – Структура куба

Сформируем куб и проведем тестирование выборки данных из него. Выберем измерение «Магазин» и меры «Стоимость», «Себестоимость» и «Валовая прибыль», результат представлен на рисунке 3.29.

<div><Все></div> <div>DWH</div> <div>Measures</div> <div>Продажи</div> <div>Валовая прибыль</div> <div>Количество</div> <div>Количество продаж</div> <div>Масса</div> <div>Себестоимость</div> <div>Стоимость</div> <div>Стоимость без скидки</div> <div>Ключевые показатели эффективности</div> <div>Магазины</div> <div>Идентификатор</div> <div>Магазин</div>	Магазин	Стоимость	Себестоимость	Валовая прибыль
	Алиса	52696573	32038337,53	20658235,47
	Алмаз 1	93345235,48	58072685,18	35272550,3
	Алмаз 2	74908209,5	46592706,45	28315503,05
	Весна	89295202	55869117,48	33426084,52
	Воробьевы горы	6248890	3491335,14	2757554,86
	Ворошиловский	3607476	2351906,5	1255569,5
	Гипермаркет Gold & Briliants	171412702,15	106597775,74	64814926,41
	ГИПЕР-СИТИ	5098538	3354755,95	1743782,05
	Гипер-Сити (старый)	63453608,38	38273244,37	25180364,01
	Горки	166516573,57	105193590,74	61322982,83
	Горки 2	94715955,1	51471766,04	43243590,06
	Горки 3	94715955,1	51471766,04	43243590,06
	Горки 4	94715955,1	51471766,04	43243590,06

Рисунок 3.29 – Результат тестовой выборки из куба

Разработка OLAP-куба завершена, в последствии мы будем использовать этот куб, как источник данных в приложении PowerBI.

3.7Разработка аналитических отчетов в приложении Microsoft Power BI

Приложение PowerBIподдерживает различные источники данных. Нам необходимо создать источник данных из служб AnalysisServices, будем получать данные в режиме реального времени, представлено на рисунке 3.30.



Рисунок 3.30 – Источник данных AnalysisServices

После создания источника данных в правой стороне приложения появятся доступные таблицы OLAP-куба, которые можно использовать для создания аналитических или управленческих отчетов.

Выберем измерение «Магазин» и меру «Валовая прибыль» и сформируем диаграмму 5 самых продающих магазинов, представлено на рисунке 3.31.

Валовая прибыль по магазинам

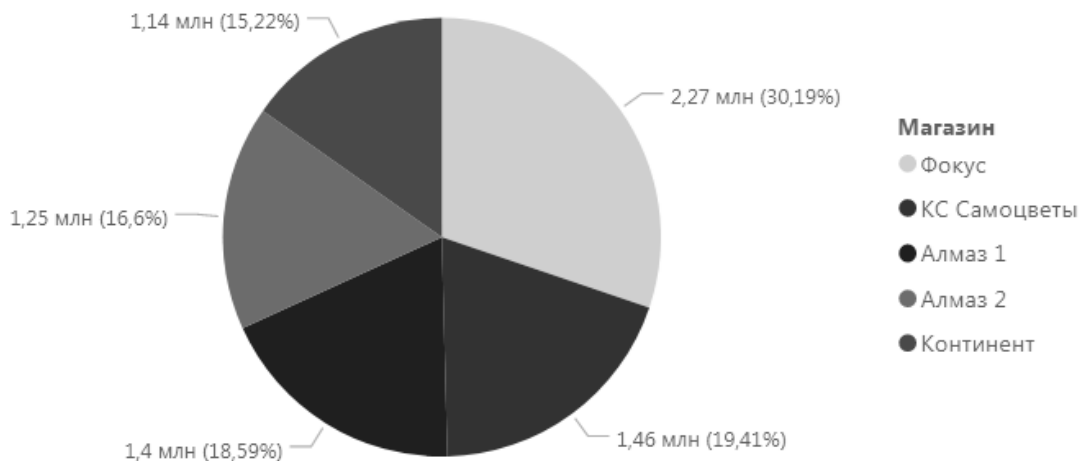


Рисунок 3.31 – Диаграмма «5 самых продающих магазинов»

Выберем измерение «Месяц» и меру «Количество» и сформируем график для того, чтобы проанализировать сезонность продажи ювелирных изделий, представлено на рисунке 3.32.

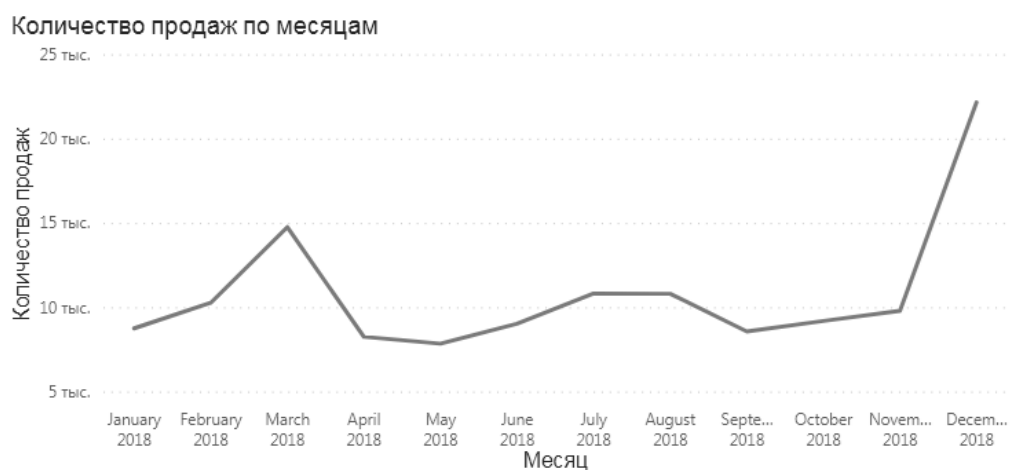


Рисунок 3.32 – График «Сезонность продаж ювелирных изделий»

PowerBI позволяет гибко менять настройки отчетов, фильтры и т.п. Стоит лишь нажать на значение в одном отчете, как данные в другом отчете сразу обновятся под выбранное значение.

Сформируем сводный отчет по измерениям «Металл», «Цвет металла» и «Размер» с мерой «Количество», представлено на рисунке 3.33.

Металл	Белый	Комбинированный	Красный	Лимонный	Всего
без проб	339,00	394,00	283,00	380,00	1 396,00
13,00			127,00		127,00
14,00	1,00		113,00		114,00
15,00	109,00	1,00	1,00	3,00	114,00
16,00	101,00	3,00	14,00	4,00	122,00
17,00	126,00	1,00	18,00	1,00	146,00
18,00			5,00	107,00	112,00
19,00			4,00	137,00	141,00
20,00				126,00	126,00
21,00	1,00	125,00	1,00		127,00
22,00	1,00	122,00			123,00
23,00		142,00		2,00	144,00
Золото	11 228,00	14 739,00	31 154,00	13 925,00	71 046,00
13,00	6,00	6,00	4 373,00	9,00	4 394,00
14,00	6,00	12,00	4 415,00	10,00	4 443,00
14,50			2,00		2,00
15,00	3 669,00	259,00	387,00	221,00	4 536,00
15,50	1,00	11,00	317,00		329,00
Всего	23 920,00	28 714,00	40 695,00	28 147,00	121 476,00

Рисунок 3.33 – Сводная таблица

Данная таблица позволяет понять, сколько продается изделий в разрезах: «Металл», «Цвет металла» и «Размер».

3.8 Сравнение всех решений

Для того, чтобы наглядно сравнить все решения составим таблицу сравнения по функциональным требованиям к системе, представлено в таблице 11.

Таблица 11 – Сравнение всех решений

Требование	COM	Представления	Веб-сервисы	MS Power BI + OData	OData
Быстрый способ реализации	+	+	-	+	-
Универсальность (смена СУБД)	+	-	+	+	+
Поддержка различных ОС	-	+	+	-	+
Разделение ответственности в ETL-процессе	-	-	+	-	+
Безопасность чтения данных	-	-	+	+	+
Соответствие лицензионной политики	+	-	+	+	+
Возможность использования OLAP-куба	+	+	+	-	+
Удаленное расположение сервера OLAP	-	-	+	+	+
Обновление IC не влияет на работоспособность	+	-	+	+	+
Универсальность запросов к данным	+	+	+	-	-
Время длительности ETL-процесса (минут)	50	-	XML 20, JSON 15	-	13

Внешние соединение и представления данных лучше использовать только, если загрузку данных нужно произвести один раз. Внешние соединение не годится даже на единоразовую загрузку, если объем данных большой. Представления являются наиболее быстрыми в плане длительности процесса загрузки данных в хранилище.

MSPowerBIи ODataне годится, так как данные постоянно нужно будет получать по веб-сервису и в этом случае не будет возможности использования OLAP-куба, соответственно, скорость построения отчетов не увеличится.

В итоге выбор стоит сделать в пользу веб-сервисов: RESTили OData. Так как время длительности ETL-процесса является наименьшим, существует разграничение ответственности в процессе и сами решения являются универсальными.

Вывод по разделу три

В разделе «Разработка и тестирование OLAP-системы» выполнено создание хранилища данных, проведено исследование длительности ETL-процесса существующих решений, выполнена реализация OLAP-системы и произведено сравнение всех решений.

ЗАКЛЮЧЕНИЕ

Разработанная OLAP-система позволила повысить производительность построения аналитических и управленческих отчетов, а так же дала возможность проводить статистический анализ данных.

В первой главе выпускной квалификационной работы был проведен анализ предметной области: сформулирована проблема, описаны существующие решения и их преимущества и недостатки.

Вторая глава посвящена проектированию системы в целом и хранилища данных.

Проводится описание выбранных технических решений. Для хранения данных выбрана СУБД MSSQLServer, для создания ETL-приложения SQLServerIntergrationServices, для создания OLAP-куба SQLServerAnalysisServices. Описывается архитектура всей системы, на которой видно, как взаимодействуют между собой различные компоненты. Выполняется проектирование хранилища данных, какие таблицы она в себя включает.

В третьей главе выполняется создание хранилища данных, исследование длительности ETL-процесса существующих решений, разработка и тестирование OLAP-системы. После разработки и тестирования проводится сравнение всех решений по функциональным требованиям. Выявляется наиболее подходящее решение.

Таким образом, цель работы – выявление наиболее эффективного метода интеграции учетных систем на платформе 1С с OLAP-системами для построения управленческих отчетов достигнута. Данная цель была достигнута путем решений следующих задач: описанием предметной области, анализом существующих решений, проектированием и реализацией OLAP-системы.

Данная работа имеет практическую значимость, и разработка и внедрение OLAP-системы является целесообразным.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

- 1) OpenDataProtocol[Электронный ресурс]. – Режим доступа :<https://www.odata.org/documentation/odata-version-3-0/odata-version-3-0-core-protocol/>, свободный.
- 2) Стандартный интерфейс ODataплатформы 1С [Электронный ресурс]. – Режим доступа :<https://its.1c.ru/db/v837doc#bookmark:dev:TI000001357>, свободный.
- 3) MSSQLServerIntegrationServices[Электронный ресурс]. – Режим доступа :<https://docs.microsoft.com/ru-ru/sql/integration-services/sql-server-integration-services?view=sql-server-ver15>, свободный.
- 4) MSSQLServerAnalysisServices[Электронный ресурс]. – Режим доступа :<https://docs.microsoft.com/ru-ru/analysis-services/ssas-overview?view=asallproducts-allversions>, свободный.
- 5) Документация повеб-серверу Apache[Электронный ресурс]. – Режим доступа :https://wiki.archlinux.org/index.php/Apache_HTTP_Server, свободный.
- 6) Документация по приложению «АТКBiView-1С Коннектор»[Электронный ресурс]. – Режим доступа :<https://biview.atkcg.ru/knowledge-base/>, свободный.
- 7) ETLи 1С. Извлечение данных[Электронный ресурс]. – Режим доступа :<https://habr.com/ru/post/334798/>, свободный.
- 8) Стандартный интерфейс ODataи MSPowerBI[Электронный ресурс]. – Режим доступа :<https://infostart.ru/public/914689/>, свободный.
- 9) Основные функции ETL-систем[Электронный ресурс]. – Режим доступа :<https://habr.com/ru/post/248231/>, свободный.
- 10) Внешние соединение платформы 1С [Электронный ресурс]. – Режим доступа :<https://v8.1c.ru/platforma/vneshnee-soedinenie/>, свободный.
- 11) Регистры накопления платформы 1С [Электронный ресурс]. – Режим доступа :<https://v8.1c.ru/platforma/registr-nakopleniya/>, свободный.

12) Разработка веб-сервисов (SOAP) на платформе 1С [Электронный ресурс]. – Режим доступа :<https://infostart.ru/public/327963/>, свободный.

13) Создание и отладка HTTP-сервисов (REST) на платформе 1С [Электронный ресурс]. – Режим доступа :<https://its.1c.ru/db/metod8dev#content:5756:hdoc>, свободный.

14) Создание OLAP-куба в VisualStudio[Электронный ресурс]. – Режим доступа :<https://docs.microsoft.com/ru-ru/analysis-services/multidimensional-models/create-a-cube-using-a-data-source-view?view=asallproducts-allversions>, свободный.

15) Создание проекта WindowsForms в VisualStudio[Электронный ресурс]. – Режим доступа :<https://metanit.com/sharp/windowsforms/1.1.php>, свободный.

16) Разработка проекта SQLServerIntegrationServices[Электронный ресурс]. – Режим доступа :<https://habr.com/ru/post/330618/>, свободный.

17) R. Kimball, J. Caserta. The Data Warehouse ETL Toolkit: Practical Techniques for Extracting, Cleaning, Conforming, and Delivering Data. 2011. 330-350 pp.

18) E. Sperley. The Enterprise Data Warehouse. Planning, Building, and Implementation. 1999. 250-311 pp.

19) Erik Thomsen, OLAP Solution: Building Multidimensional Information Systems. Letters 2, 3 -5 (2002).

20) Симонов В.В., Конов В.А. Применение OLAP-технологий для учетных систем на платформе 1С. <https://elibrary.ru/item.asp?id=38569053>.

ПРИЛОЖЕНИЕ А

Код решения «Внешнее соединение к платформе 1С»

```
private void Load_Click(object sender, EventArgs)
{
    string interval = Convert.ToString(DateTime.Now) + " - ";
    string connectionString = "srvr='localhost'; ref='mcenter'; usr='Администратор'; pwd='';";
    dynamic connection;
    dynamic query;
    dynamic resultQuery;
    V83.COMConnector connector = new V83.COMConnector();
    try
    {
        connection = connector.Connect(connectionString);
    }
    catch (Exception ex)
    {
        MessageBox.Show(string.Format("Ошибка подключения к 1С \'{0}\'",
            (ex.InnerException.ToString()), ex.Message));
        return;
    }
    query = connection.NewObject("Запрос");
    query.Текст =
    "ВЫБРАТЬ " +
    "ОтчетОРозничныхПродажахТовары.Ссылка КАК Ссылка, " +
    "ОтчетОРозничныхПродажахТовары.Ссылка.Дата, " +
    "ОтчетОРозничныхПродажахТовары.Ссылка.Склад, " +
    "ОтчетОРозничныхПродажахТовары.Номенклатура КАК Номенклатура, " +
    "ОтчетОРозничныхПродажахТовары.ХарактеристикаНоменклатуры КАК
    ХарактеристикаНоменклатуры, " +
    "ВЫРАЗИТЬ(ОтчетОРозничныхПродажахТовары.Продавец1 КАК
    Справочник.ФизическиеЛица) КАК Продавец, " +
    "ОтчетОРозничныхПродажахТовары.Количество, " +
    "ОтчетОРозничныхПродажахТовары.Цена, " +
    "ОтчетОРозничныхПродажахТовары.Сумма " +
    "ПОМЕСТИТЬ ВТ_ОтчетыОРозничныхПродажах " +
    "ИЗ " +
    "Документ.ОтчетОРозничныхПродажах.Товары
    КАК ОтчетОРозничныхПродажахТовары " +
    "ГДЕ " +
    "ОтчетОРозничныхПродажахТовары.Ссылка.Проведен " +
    "И ОтчетОРозничныхПродажахТовары.Ссылка.Склад <>
    ЗНАЧЕНИЕ(Справочник.Склады.ПустаяСсылка) " +
    "И ОтчетОРозничныхПродажахТовары.Номенклатура <>
    ЗНАЧЕНИЕ(Справочник.Номенклатура.ПустаяСсылка) " +
    "И ОтчетОРозничныхПродажахТовары.Номенклатура.ЮвелирноеИзделие " +
    "И ОтчетОРозничныхПродажахТовары.ХарактеристикаНоменклатуры <>
    ЗНАЧЕНИЕ(Справочник.ХарактеристикиНоменклатуры.ПустаяСсылка) " +
```

Продолжение приложения А

```

"И ОтчетОРозничныхПродажахТовары.Продавец1 ССЫЛКА
Справочник.ФизическиеЛица " +
"И ОтчетОРозничныхПродажахТовары.Продавец1 <
ЗНАЧЕНИЕ(Справочник.ФизическиеЛица.ПустаяСсылка) " +
"И ОтчетОРозничныхПродажахТовары.Количество > 0 " +
"ИНДЕКСИРОВАТЬ ПО " +
"Ссылка, " +
"Номенклатура, " +
"ХарактеристикаНоменклатуры" +
"; " +
"ВЫБРАТЬ " +
    "ВЫРАЗИТЬ(ПродажиСебестоимость.Регистратор КАК
    Документ.ОтчетОРозничныхПродажах) КАК Ссылка, " +
"ПродажиСебестоимость.Номенклатура КАК Номенклатура, " +
    "ПродажиСебестоимость.ХарактеристикаНоменклатуры КАК
    ХарактеристикаНоменклатуры, " +
"ПродажиСебестоимость.Стоимость, " +
"ПродажиСебестоимость.Масса " +
"ПОМЕСТИТЬ ВТ_ПродажиСебестоимость " +
"ИЗ " +
    "РегистрНакопления.ПродажиСебестоимость КАК ПродажиСебестоимость " +
"ГДЕ " +
    "ПродажиСебестоимость.Номенклатура <
    ЗНАЧЕНИЕ(Справочник.Номенклатура.ПустаяСсылка) " +
"И ПродажиСебестоимость.Номенклатура.ЮвелирноеИзделие " +
    "И ПродажиСебестоимость.ХарактеристикаНоменклатуры <
    ЗНАЧЕНИЕ(Справочник.ХарактеристикиНоменклатуры.ПустаяСсылка) " +
"И ПродажиСебестоимость.Количество > 0 " +
"ИНДЕКСИРОВАТЬ ПО " +
"Ссылка, " +
"Номенклатура, " +
"ХарактеристикаНоменклатуры" +
"; " +
"ВЫБРАТЬ " +
"ВТ_ОтчетыОРозничныхПродажах.Ссылка КАК Ссылка, " +
"ВТ_ОтчетыОРозничныхПродажах.Дата КАК Дата, " +
"ВТ_ОтчетыОРозничныхПродажах.Склад КАК Склад, " +
"ВТ_ОтчетыОРозничныхПродажах.Номенклатура КАК Номенклатура, " +
    "ВТ_ОтчетыОРозничныхПродажах.ХарактеристикаНоменклатуры КАК
    ХарактеристикаНоменклатуры, " +
"ВТ_ОтчетыОРозничныхПродажах.Продавец КАК Продавец, " +
"ВТ_ОтчетыОРозничныхПродажах.Количество КАК КоличествоТовара, " +
    "ЕСТЬNULL(ВТ_ПродажиСебестоимость.Стоимость, 0) КАК Себестоимость, " +
"ВТ_ОтчетыОРозничныхПродажах.Сумма КАК Стоимость, " +
    "ВТ_ОтчетыОРозничныхПродажах.Количество *
    ВТ_ОтчетыОРозничныхПродажах.Цена КАК СтоимостьБезСкидки, " +

```

Окончание приложения А

```

"BT_ОтчетыОРозничныхПродажах.Сумма -
ЕСТЬNULL(BT_ПродажиСебестоимость.Стоимость, 0) КАК ВаловаяПрибыль, " +
"ЕСТЬNULL(BT_ПродажиСебестоимость.Масса, 0) КАК Масса " +
"ИЗ " +
"BT_ОтчетыОРозничныхПродажах КАК BT_ОтчетыОРозничныхПродажах
ЛЕВОЕ СОЕДИНЕНИЕ " +
"BT_ПродажиСебестоимость КАК BT_ПродажиСебестоимость ПО
BT_ОтчетыОРозничныхПродажах.Ссылка = BT_ПродажиСебестоимость.Ссылка "
+
"И BT_ОтчетыОРозничныхПродажах.Номенклатура =
BT_ПродажиСебестоимость.Номенклатура " +
"И BT_ОтчетыОРозничныхПродажах.ХарактеристикаНоменклатуры =
BT_ПродажиСебестоимость.ХарактеристикаНоменклатуры " +
"УПОРЯДОЧИТЬ ПО " +
"Дата";
resultQuery = query.Выполнить().Выбрать();
while (resultQuery.Следующий() == true)
{
    stringsourceId =
        connection.XMLString(resultQuery.Ссылка.УникальныйИдентификатор());
    DateTime date = Convert.ToDateTime(resultQuery.Дата);
    string sourceIdStore = connection.XMLString
        (resultQuery.Склад.УникальныйИдентификатор());
    string sourceIdProduct = connection.XMLString
        (resultQuery.Номенклатура.УникальныйИдентификатор());
    string sourceIdProductProperty = connection.XMLString
        (resultQuery.ХарактеристикаНоменклатуры.УникальныйИдентификатор());
    string sourceIdSeller = connection.XMLString
        (resultQuery.Продавец.УникальныйИдентификатор());
    double count = Convert.ToDouble(resultQuery.КоличествоТовара,
        System.Globalization.CultureInfo.GetCultureInfo("en-US"));
    double firstCost = Convert.ToDouble
        (resultQuery.Себестоимость, System.Globalization.CultureInfo.GetCultureInfo("en-
        US"));
    double cost = Convert.ToDouble
        (resultQuery.Стоимость, System.Globalization.CultureInfo.GetCultureInfo("en-US"));
    double costWithoutDiscount =
        Convert.ToDouble(resultQuery.СтоимостьБезСкидки,
        System.Globalization.CultureInfo.GetCultureInfo("en-US"));
    double grossProfit = Convert.ToDouble
        (resultQuery.ВаловаяПрибыль, System.Globalization.CultureInfo.GetCultureInfo("en-
        US"));
    double weight = Convert.ToDouble
        (resultQuery.Масса, System.Globalization.CultureInfo.GetCultureInfo("en-US"));
    queriesTableAdapter1.InsertUpdateSale(date, sourceId, sourceIdStore, sourceIdSeller,
        sourceIdProduct, sourceIdProductProperty,
        count, firstCost, cost, costWithoutDiscount, grossProfit, weight);
}
tbTimeCOM.Text = interval + Convert.ToString(DateTime.Now);}

```

ПРИЛОЖЕНИЕ Б

Код формирования пакета данных в формате XML

Функция Get(startDate, endDate, loadWithDirectories)

Если Не ЗначениеЗаполнено(startDate) Тогда

Тип = ФабрикаXDTO.Тип("http://localhost/dwh", "Error");

Ошибка = ФабрикаXDTO.Создать(Тип);

Ошибка.Сообщение = "Не заполнен параметр 'startDate'";

Возврат Ошибка;

КонецЕсли;

Если Не ЗначениеЗаполнено(endDate) Тогда

Тип = ФабрикаXDTO.Тип("http://localhost/dwh", "Error");

Ошибка = ФабрикаXDTO.Создать(Тип);

Ошибка.Сообщение = "Не заполнен параметр 'endDate'";

Возврат Ошибка;

КонецЕсли;

ТекстЗапроса =

"ВЫБРАТЬ

| ОтчетОРозничныхПродажахТовары.Ссылка КАК Ссылка,

| ОтчетОРозничныхПродажахТовары.Ссылка.Дата,

| ОтчетОРозничныхПродажахТовары.Ссылка.Склад,

| ОтчетОРозничныхПродажахТовары.Номенклатура КАК Номенклатура,

| ОтчетОРозничныхПродажахТовары.ХарактеристикаНоменклатуры КАК

ХарактеристикаНоменклатуры,

| ВЫРАЗИТЬ(ОтчетОРозничныхПродажахТовары.Продавец1 КАК

Справочник.ФизическиеЛица) КАК Продавец,

| ОтчетОРозничныхПродажахТовары.Количество,

| ОтчетОРозничныхПродажахТовары.Цена,

| ОтчетОРозничныхПродажахТовары.Сумма

ПОМЕСТИТЬ ВТ_ОтчетыОРозничныхПродажах

ИЗ

| Документ.ОтчетОРозничныхПродажах.Товары КАК

ОтчетОРозничныхПродажахТовары

ГДЕ

| ОтчетОРозничныхПродажахТовары.Ссылка.Проведен

| И ОтчетОРозничныхПродажахТовары.Ссылка.Дата МЕЖДУ

&НачалоПериода И &КонецПериода

| И ОтчетОРозничныхПродажахТовары.Ссылка.Склад <>

ЗНАЧЕНИЕ(Справочник.Склады.ПустаяСсылка)

| И ОтчетОРозничныхПродажахТовары.Номенклатура <>

ЗНАЧЕНИЕ(Справочник.Номенклатура.ПустаяСсылка)

| И ОтчетОРозничныхПродажахТовары.Номенклатура.ЮвелирноеИзделие

| И ОтчетОРозничныхПродажахТовары.ХарактеристикаНоменклатуры <>

ЗНАЧЕНИЕ(Справочник.ХарактеристикиНоменклатуры.ПустаяСсылка)

| И ОтчетОРозничныхПродажахТовары.Продавец1 ССЫЛКА

Справочник.ФизическиеЛица

| И ОтчетОРозничныхПродажахТовары.Продавец1 <>

ЗНАЧЕНИЕ(Справочник.ФизическиеЛица.ПустаяСсылка)

| И ОтчетОРозничныхПродажахТовары.Количество > 0

ИНДЕКСИРОВАТЬ ПО

Ссылка,
Номенклатура,
ХарактеристикаНоменклатуры

;

////////////////////////////////////

ВЫБРАТЬ

ВЫРАЗИТЬ(ПродажиСебестоимость.Регистратор КАК
Документ.ОтчетОРозничныхПродажах) КАК Ссылка,
ПродажиСебестоимость.Номенклатура КАК Номенклатура,
ПродажиСебестоимость.ХарактеристикаНоменклатуры КАК
ХарактеристикаНоменклатуры,
ПродажиСебестоимость.Стоимость,
ПродажиСебестоимость.Масса

ПОМЕСТИТЬ ВТ_ПродажиСебестоимость

ИЗ

РегистрНакопления.ПродажиСебестоимость КАК ПродажиСебестоимость

ГДЕ

ПродажиСебестоимость.Период МЕЖДУ &НачалоПериода И
&КонецПериода

И ПродажиСебестоимость.Номенклатура <>

ЗНАЧЕНИЕ(Справочник.Номенклатура.ПустаяСсылка)

И ПродажиСебестоимость.Номенклатура.ЮвелирноеИзделие

И ПродажиСебестоимость.ХарактеристикаНоменклатуры <>

ЗНАЧЕНИЕ(Справочник.ХарактеристикиНоменклатуры.ПустаяСсылка)

И ПродажиСебестоимость.Количество > 0

ИНДЕКСИРОВАТЬ ПО

Ссылка,
Номенклатура,
ХарактеристикаНоменклатуры

;

////////////////////////////////////

ВЫБРАТЬ

ВТ_ОтчетыОРозничныхПродажах.Ссылка КАК Ссылка,
ВТ_ОтчетыОРозничныхПродажах.Дата КАК Дата,
ВТ_ОтчетыОРозничныхПродажах.Склад КАК Склад,
ВТ_ОтчетыОРозничныхПродажах.Номенклатура КАК Номенклатура,
ВТ_ОтчетыОРозничныхПродажах.ХарактеристикаНоменклатуры КАК
ХарактеристикаНоменклатуры,
"""" КАК Металл,
"""" КАК ЦветМеталла,
0 КАК Размер,
ВТ_ОтчетыОРозничныхПродажах.Продавец КАК Продавец,
"""" КАК СкладПредставление,
"""" КАК НоменклатураПредставление,

```

|      """" КАС ХарактеристикаНоменклатурыПредставление,
|      """" КАС МеталлПредставление,
|      """" КАС ЦветМеталлаПредставление,
|      """" КАС ПродавецПредставление,
|      ВТ_ОтчетыОРозничныхПродажах.Количество КАС Количество,
|      ЕСТЬNULL(ВТ_ПродажиСебестоимость.Стоимость, 0) КАС
Себестоимость,
|      ВТ_ОтчетыОРозничныхПродажах.Сумма КАС Стоимость,
|      ВТ_ОтчетыОРозничныхПродажах.Количество *
ВТ_ОтчетыОРозничныхПродажах.Цена КАС СтоимостьБезСкидки,
|      ВТ_ОтчетыОРозничныхПродажах.Сумма -
ЕСТЬNULL(ВТ_ПродажиСебестоимость.Стоимость, 0) КАС ВаловаяПрибыль,
|      ЕСТЬNULL(ВТ_ПродажиСебестоимость.Масса, 0) КАС Масса
|ИЗ
|      ВТ_ОтчетыОРозничныхПродажах КАС ВТ_ОтчетыОРозничныхПродажах
|      ЛЕВОЕ СОЕДИНЕНИЕ ВТ_ПродажиСебестоимость КАС
ВТ_ПродажиСебестоимость
|      ПО ВТ_ОтчетыОРозничныхПродажах.Ссылка =
ВТ_ПродажиСебестоимость.Ссылка
|      И ВТ_ОтчетыОРозничныхПродажах.Номенклатура =
ВТ_ПродажиСебестоимость.Номенклатура
|      И
ВТ_ОтчетыОРозничныхПродажах.ХарактеристикаНоменклатуры =
ВТ_ПродажиСебестоимость.ХарактеристикаНоменклатуры
|
|УПОРЯДОЧИТЬ ПО
|      Дата";
Запрос = Новый Запрос;
Запрос.Текст = ТекстЗапроса;
Запрос.УстановитьПараметр("НачалоПериода", НачалоДня(startDate));
Запрос.УстановитьПараметр("КонецПериода", КонецДня(endDate));
РезультатЗапроса = Запрос.Выполнить();
Если РезультатЗапроса.Пустой() Тогда
    Тип = ФабрикаХДТО.Тип("http://localhost/dwh", "Error");
    Ошибка = ФабрикаХДТО.Создать(Тип);
    Ошибка.Сообщение = "Данные о продажах не найдены";
    Возврат Ошибка;
Иначе
    Тип = ФабрикаХДТО.Тип("http://localhost/dwh", "Sales");
    Продажи = ФабрикаХДТО.Создать(Тип);
    Тип = ФабрикаХДТО.Тип("http://localhost/dwh", "Sale");

    Выборка = РезультатЗапроса.Выбрать();
    Пока Выборка.Следующий() Цикл
        СсылкаGUID = ?(ЗначениеЗаполнено(Выборка.Ссылка),
        Строка(Выборка.Ссылка.УникальныйИдентификатор()), "");
        СкладGUID = ?(ЗначениеЗаполнено(Выборка.Склад),
        Строка(Выборка.Склад.УникальныйИдентификатор()), "");

```

```

НоменклатураGUID = ?(ЗначениеЗаполнено(Выборка.Номенклатура),
Строка(Выборка.Номенклатура.УникальныйИдентификатор()), "");
ХарактеристикаНоменклатурыGUID=? (ЗначениеЗаполнено(Выборка.Хара
ктеристикаНоменклатуры),
Строка(Выборка.ХарактеристикаНоменклатуры.УникальныйИдентификат
ор()), "");
МеталлGUID = ?(ЗначениеЗаполнено(Выборка.Металл),
Строка(Выборка.Металл.УникальныйИдентификатор()), "");
ЦветМеталлаGUID = ?(ЗначениеЗаполнено(Выборка.ЦветМеталла),
Строка(Выборка.ЦветМеталла.УникальныйИдентификатор()), "");
ПродавецGUID = ?(ЗначениеЗаполнено(Выборка.Продавец),
Строка(Выборка.Продавец.УникальныйИдентификатор()), "");
    Продажа = ФабрикаXDTO.Создать(Тип);
    Продажа.СсылкаGUID = СсылкаGUID;
    Продажа.Дата = Выборка.Дата;
    Продажа.СкладGUID = СкладGUID;
    Продажа.НоменклатураGUID = НоменклатураGUID;
    Продажа.ХарактеристикаНоменклатурыGUID =
    ХарактеристикаНоменклатурыGUID;
    Продажа.ПродавецGUID = ПродавецGUID;
    Продажа.Количество = Выборка.Количество;
    Продажа.Себестоимость = Выборка.Себестоимость;
    Продажа.Стоимость = Выборка.Стоимость;
    Продажа.СтоимостьБезСкидки = Выборка.СтоимостьБезСкидки;
    Продажа.ВаловаяПрибыль = Выборка.ВаловаяПрибыль;
    Продажа.Масса = Выборка.Масса;
    Продажи.Продажа.Добавить(Продажа);
    КонецЦикла;
КонецЕсли;
    Возврат Продажи;
КонецФункции

```


ПРИЛОЖЕНИЕ В

Код формирования пакета данных в формате JSON

Функция SalesGET(Запрос)

```
    Ответ = Новый HTTPСервисОтвет(200);
    Ответ.Заголовки.Вставить("Content-Type", "text/html; charset=utf-8");
    НачалоПериода = Запрос.ПараметрыЗапроса.Получить("startDate");
    КонецПериода = Запрос.ПараметрыЗапроса.Получить("endDate");
    ЗагрузкаСоСправочниками =
        Запрос.ПараметрыЗапроса.Получить("loadWithDirectories");
    Если Не ЗначениеЗаполнено(НачалоПериода) Тогда
        Ответ.КодСостояния = 404;
        Ответ.УстановитьТелоИзСтроки("Не заполнен параметр 'startDate'",
            КодировкаТекста.UTF8);
        Возврат Ответ;
    КонецЕсли;
    Если Не ЗначениеЗаполнено(КонецПериода) Тогда
        Ответ.КодСостояния = 404;
        Ответ.УстановитьТелоИзСтроки("Не заполнен параметр 'endDate'",
            КодировкаТекста.UTF8);
        Возврат Ответ;
    КонецЕсли;

    Если Не ЗначениеЗаполнено(ЗагрузкаСоСправочниками) Тогда
        Ответ.КодСостояния = 404;
        Ответ.УстановитьТелоИзСтроки("Не заполнен параметр 'loadWithDirectories'",
            КодировкаТекста.UTF8);
        Возврат Ответ;
    КонецЕсли;
    Попытка
        НачалоПериода = Дата(НачалоПериода);
    Исключение
        Ответ.КодСостояния = 404;
        Ответ.УстановитьТелоИзСтроки("Ошибка преобразования параметра 'startDate' в
            тип данных 'Дата'", КодировкаТекста.UTF8);
        Возврат Ответ;
    КонецПопытки;
    Попытка
        КонецПериода = Дата(КонецПериода);
    Исключение
        Ответ.КодСостояния = 404;
        Ответ.УстановитьТелоИзСтроки("Ошибка преобразования параметра 'endDate' в
            тип данных 'Дата'", КодировкаТекста.UTF8);
        Возврат Ответ;
    КонецПопытки;
    Попытка
        ЗагрузкаСоСправочниками = Булево(ЗагрузкаСоСправочниками);
    Исключение
        Ответ.КодСостояния = 404;
```

Продолжение приложения В

```

Ответ. Установить ТелоИзСтроки("Ошибка преобразования параметра
'loadWithDirectories' в тип данных 'Булево'", КодировкаТекста.UTF8);
    Возврат Ответ;
КонецПопытки;
Попытка
    ТекстЗапроса =
    "ВЫБРАТЬ
    |      ОтчетОРозничныхПродажахТовары.Ссылка КАК Ссылка,
    |      ОтчетОРозничныхПродажахТовары.Ссылка.Дата,
    |      ОтчетОРозничныхПродажахТовары.Ссылка.Склад,
    |      ОтчетОРозничныхПродажахТовары.Номенклатура КАК
    Номенклатура,
    |      ОтчетОРозничныхПродажахТовары.ХарактеристикаНоменклатуры
    КАК ХарактеристикаНоменклатуры,
    |      ВЫРАЗИТЬ(ОтчетОРозничныхПродажахТовары.Продавец1 КАК
    Справочник.ФизическиеЛица) КАК Продавец,
    |      ОтчетОРозничныхПродажахТовары.Количество,
    |      ОтчетОРозничныхПродажахТовары.Цена,
    |      ОтчетОРозничныхПродажахТовары.Сумма
    ПОМЕСТИТЬ ВТ_ОтчетыОРозничныхПродажах
    ИЗ
    |      Документ.ОтчетОРозничныхПродажах.Товары КАК
    ОтчетОРозничныхПродажахТовары
    ГДЕ
    |      ОтчетОРозничныхПродажахТовары.Ссылка.Проведен
    |      И ОтчетОРозничныхПродажахТовары.Ссылка.Дата МЕЖДУ
    &НачалоПериода И &КонецПериода
    |      И ОтчетОРозничныхПродажахТовары.Ссылка.Склад <>
    ЗНАЧЕНИЕ(Справочник.Склады.ПустаяСсылка)
    |      И ОтчетОРозничныхПродажахТовары.Номенклатура <>
    ЗНАЧЕНИЕ(Справочник.Номенклатура.ПустаяСсылка)
    |      И
    ОтчетОРозничныхПродажахТовары.Номенклатура.ЮвелирноеИзделие
    |      И
    ОтчетОРозничныхПродажахТовары.ХарактеристикаНоменклатуры <>
    ЗНАЧЕНИЕ(Справочник.ХарактеристикиНоменклатуры.ПустаяСсылка)
    |      И ОтчетОРозничныхПродажахТовары.Продавец1 ССЫЛКА
    Справочник.ФизическиеЛица
    |      И ОтчетОРозничныхПродажахТовары.Продавец1 <>
    ЗНАЧЕНИЕ(Справочник.ФизическиеЛица.ПустаяСсылка)
    |      И ОтчетОРозничныхПродажахТовары.Количество > 0
    |
    ИНДЕКСИРОВАТЬ ПО
    |      Ссылка,
    |      Номенклатура,
    |      ХарактеристикаНоменклатуры
    ;
    ///////////////////////////////////////////////////////////////////

```

```

|ВЫБРАТЬ
|      ВЫРАЗИТЬ(ПродажиСебестоимость.Регистратор КАК
Документ.ОтчетОРозничныхПродажах) КАК Ссылка,
|      ПродажиСебестоимость.Номенклатура КАК Номенклатура,
|      ПродажиСебестоимость.ХарактеристикаНоменклатуры КАК
ХарактеристикаНоменклатуры,
|      ПродажиСебестоимость.Стоимость,
|      ПродажиСебестоимость.Масса
|ПОМЕСТИТЬ ВТ_ПродажиСебестоимость
|ИЗ
|      РегистрНакопления.ПродажиСебестоимость КАК
ПродажиСебестоимость
|ГДЕ
|      ПродажиСебестоимость.Период МЕЖДУ &НачалоПериода И
&КонецПериода
|      И ПродажиСебестоимость.Номенклатура <>
ЗНАЧЕНИЕ(Справочник.Номенклатура.ПустаяСсылка)
|      И ПродажиСебестоимость.Номенклатура.ЮвелирноеИзделие
|      И ПродажиСебестоимость.ХарактеристикаНоменклатуры <>
ЗНАЧЕНИЕ(Справочник.ХарактеристикиНоменклатуры.ПустаяСсылка)
|      И ПродажиСебестоимость.Количество > 0
|
|ИНДЕКСИРОВАТЬ ПО
|      Ссылка,
|      Номенклатура,
|      ХарактеристикаНоменклатуры
|;
|
|////////////////////////////////////
|ВЫБРАТЬ
|      ВТ_ОтчетыОРозничныхПродажах.Ссылка КАК Ссылка,
|      ВТ_ОтчетыОРозничныхПродажах.Дата КАК Дата,
|      ВТ_ОтчетыОРозничныхПродажах.Склад КАК Склад,
|      ВТ_ОтчетыОРозничныхПродажах.Номенклатура КАК
Номенклатура,
|      ВТ_ОтчетыОРозничныхПродажах.ХарактеристикаНоменклатуры
КАК ХарактеристикаНоменклатуры,
|      """" КАК Металл,
|      """" КАК ЦветМеталла,
|      0 КАК Размер,
|      ВТ_ОтчетыОРозничныхПродажах.Продавец КАК Продавец,
|      """" КАК СкладПредставление,
|      """" КАК НоменклатураПредставление,
|      """" КАК ХарактеристикаНоменклатурыПредставление,
|      """" КАК МеталлПредставление,
|      """" КАК ЦветМеталлаПредставление,
|      """" КАК ПродавецПредставление,
|      ВТ_ОтчетыОРозничныхПродажах.Количество КАК Количество,

```

```

|      ЕСТЬNULL(BT_ПродажиСебестоимость.Стоимость, 0) КАК
Себестоимость,
|      BT_ОтчетыОРозничныхПродажах.Сумма КАК Стоимость,
|      BT_ОтчетыОРозничныхПродажах.Количество *
BT_ОтчетыОРозничныхПродажах.Цена КАК СтоимостьБезСкидки,
|      BT_ОтчетыОРозничныхПродажах.Сумма -
ЕСТЬNULL(BT_ПродажиСебестоимость.Стоимость, 0) КАК
ВаловаяПрибыль,
|      ЕСТЬNULL(BT_ПродажиСебестоимость.Масса, 0) КАК Масса
|ИЗ
|      BT_ОтчетыОРозничныхПродажах КАК
BT_ОтчетыОРозничныхПродажах
|      ЛЕВОЕ СОЕДИНЕНИЕ BT_ПродажиСебестоимость КАК
BT_ПродажиСебестоимость
|      ПО BT_ОтчетыОРозничныхПродажах.Ссылка =
BT_ПродажиСебестоимость.Ссылка
|      И BT_ОтчетыОРозничныхПродажах.Номенклатура =
BT_ПродажиСебестоимость.Номенклатура
|      И
BT_ОтчетыОРозничныхПродажах.ХарактеристикаНоменклатуры =
BT_ПродажиСебестоимость.ХарактеристикаНоменклатуры
|
|УПОРЯДОЧИТЬ ПО
|      Дата";
Запрос = Новый Запрос;
Запрос.Текст = ТекстЗапроса;
Запрос.УстановитьПараметр("НачалоПериода",
НачалоДня(НачалоПериода));
Запрос.УстановитьПараметр("КонецПериода", КонецДня(КонецПериода));
РезультатЗапроса = Запрос.Выполнить();
Если РезультатЗапроса.Пустой() Тогда
    Ответ.КодСостояния = 404;
Ответ.УстановитьТелоИзСтроки("Данные о продажах не найдены",
КодировкаТекста.UTF8);
    Возврат Ответ;
Иначе
    Счетчик = 1;
    ЗаписьJSON = Новый ЗаписьJSON;
    ЗаписьJSON.УстановитьСтроку();
    ЗаписьJSON.ЗаписатьНачалоМассива();
    Выборка = РезультатЗапроса.Выбрать();
    Пока Выборка.Следующий() Цикл
        СсылкаGUID = ?(ЗначениеЗаполнено(Выборка.Ссылка),
Строка(Выборка.Ссылка.УникальныйИдентификатор()), "");
        СкладGUID = ?(ЗначениеЗаполнено(Выборка.Склад),
Строка(Выборка.Склад.УникальныйИдентификатор()), "");
        НоменклатураGUID =
        ?(ЗначениеЗаполнено(Выборка.Номенклатура),
Строка(Выборка.Номенклатура.УникальныйИдентификатор()), "");

```

Продолжение приложения В

```
ХарактеристикаНоменклатурыGUID =  
?(ЗначениеЗаполнено(Выборка.ХарактеристикаНоменклатуры),  
Строка(Выборка.ХарактеристикаНоменклатуры.УникальныйИденти  
фикатор()), "");  
МеталлGUID = ?(ЗначениеЗаполнено(Выборка.Металл),  
Строка(Выборка.Металл.УникальныйИдентификатор()), "");  
ЦветМеталлаGUID =  
?(ЗначениеЗаполнено(Выборка.ЦветМеталла),  
Строка(Выборка.ЦветМеталла.УникальныйИдентификатор()), "");  
ПродавецGUID = ?(ЗначениеЗаполнено(Выборка.Продавец),  
Строка(Выборка.Продавец.УникальныйИдентификатор()), "");  
    ЗаписьJSON.ЗаписатьНачалоОбъекта();  
    ЗаписьJSON.ЗаписатьИмяСвойства("sourceId");  
ЗаписьJSON.ЗаписатьЗначение(СсылкаGUID);  
    ЗаписьJSON.ЗаписатьИмяСвойства("date");  
ЗаписьJSON.ЗаписатьЗначение(Строка(Выборка.Дата));  
    ЗаписьJSON.ЗаписатьИмяСвойства("store_sourceId");  
ЗаписьJSON.ЗаписатьЗначение(СкладGUID);  
    ЗаписьJSON.ЗаписатьИмяСвойства("product_sourceId");  
ЗаписьJSON.ЗаписатьЗначение(НоменклатураGUID);  
  
ЗаписьJSON.ЗаписатьИмяСвойства("productProperty_sourceId");  
ЗаписьJSON.ЗаписатьЗначение(ХарактеристикаНоменклатурыGUID  
);  
    ЗаписьJSON.ЗаписатьИмяСвойства("seller_sourceId");  
  
ЗаписьJSON.ЗаписатьЗначение(ПродавецGUID);  
    ЗаписьJSON.ЗаписатьИмяСвойства("count");  
ЗаписьJSON.ЗаписатьЗначение(Выборка.Количество);  
    ЗаписьJSON.ЗаписатьИмяСвойства("firstCost");  
ЗаписьJSON.ЗаписатьЗначение(Выборка.Себестоимость);  
    ЗаписьJSON.ЗаписатьИмяСвойства("cost");  
ЗаписьJSON.ЗаписатьЗначение(Выборка.Стоимость);  
    ЗаписьJSON.ЗаписатьИмяСвойства("costWithoutDiscount");  
ЗаписьJSON.ЗаписатьЗначение(Выборка.СтоимостьБезСкидки);  
    ЗаписьJSON.ЗаписатьИмяСвойства("grossProfit");  
ЗаписьJSON.ЗаписатьЗначение(Выборка.ВаловаяПрибыль);  
    ЗаписьJSON.ЗаписатьИмяСвойства("weight");  
ЗаписьJSON.ЗаписатьЗначение(Выборка.Масса);  
    ЗаписьJSON.ЗаписатьКонецОбъекта();  
    Счетчик = Счетчик + 1;  
КонецЦикла;  
ЗаписьJSON.ЗаписатьКонецМассива();  
Ответ.Заголовки.Вставить("Content-type", "application/json;  
charset=utf-8");  
Ответ.УстановитьТелоИзСтроки(ЗаписьJSON.Закреть(),  
КодировкаТекста.UTF8, ИспользованиеByteOrderMark.НеИспользовать);  
КонецЕсли;  
Исключение
```

```
    Ответ.КодСостояния = 404;  
    Ответ.Заголовки.Вставить("Content-type", "application/json; charset=utf-8");  
    Ответ.УстановитьТелоИзСтроки(ОписаниеОшибки(),  
        КодировкаТекста.UTF8);  
    КонецПопытки;  
    Возврат Ответ;  
КонецФункции
```