Моделирование и анализ данных
2023. Том 13. № 2. С. 180–193
DOI: https://doi.org/10.17759/mda.2023130210
ISSN: 2219-3758 (печатный)
ISSN: 2311-9454 (online)


Modelling and Data Analysis
2023. Vol. 13, no. 2, pp. 180–193
DOI: https://doi.org/10.17759/mda.2023130210
ISSN: 2219-3758 (print)
ISSN: 2311-9454 (online)


## ◇◇◇◇◇◇◇КОМПЛЕКСЫ ПРОГРАММ◇◇◇◇◇◇◇

УДК 004.42

# Разработка ETL процесса на базе open source технологий для решения задачи доставки данных потребителям

***Старков В.В.****
Московский институт стали и сплавов
(национальный исследовательский технологический университет)
(НИТУ МИСиС), г. Москва, Российская Федерация
ORCID: https://orcid.org/0009-0006-0662-4852
e-mail: starkov.viatcheslav@yandex.ru

***Горбатова С.С.****
Московский институт стали и сплавов
(национальный исследовательский технологический университет)
(НИТУ МИСиС), г. Москва, Российская Федерация
ORCID: https://orcid.org/0009-0005-5213-6780
e-mail: ssgorbatova@misis.ru

***Водолага В.И.*****
Московский государственный университет им. М.В. Ломоносова (МГУ)
г. Москва, Российская Федерация
ORCID: https://orcid.org/0009-0003-1816-0088
e-mail: vikavodolaga1@gmail.com

В статье рассматриваются вопросы разработки ETL процесса для хранилища данных на базе open source технологий, взамен частного ПО, поставляемого вендором. Процесс позволяет доставлять данные от источника к потребителю, ориентируясь на скорость доставки, затрачиваемые ресурсы и удобство разработки. Представлена архитектура для решения задачи с описанием заменяемых процессов, реализована передача данных по новому процессу. Задействованы современные инструменты, применяемые для работы с данными, описаны способы взаимодействия с ними и подбор технических характеристик для процесса.

***Ключевые слова:*** хранилище данных, open source, программное обеспечение, ETL процесс, доставка данных.

**Для цитаты:**
*Старков В.В., Горбатова С.С., Водолага В.И.* Разработка ETL процесса на базе open source технологий для решения задачи доставки данных потребителям // Моделирование и анализ данных. 2023. Том 13. № 2. С. 180–193. DOI: https://doi.org/10.17759/mda.2023130210

***Старков Вячеслав Вячеславович,** аспирант, Московский институт стали и сплавов (национальный исследовательский технологический университет) (НИТУ МИСиС), г. Москва, Российская Федерация, ORCID: https://orcid.org/0009-0006-0662-4852,

e-mail: starkov.viatcheslav@yandex.ru

****Горбатова Светлана Сергеевна,** старший преподаватель, Московский институт стали и сплавов (национальный исследовательский технологический университет) (НИТУ МИСиС), г. Москва, Российская Федерация, ORCID: https://orcid.org/0009-0005-5213-6780, e-mail: ssgorbatova@misis.ru

*****Водолага Виктория Игоревна,** магистр, Московский государственный университет имени М.В.Ломоносова (МГУ), г. Москва, Российская Федерация, ORCID: https://orcid.org/0009-0003-1816-0088, e-mail: vikavodolaga1@gmail.com

## 1. ВВЕДЕНИЕ

Процесс ETL (extract, transform, load) призван доставлять данные из источника к приемнику и проводить трансформации данных, нацеленные на получение из массива сырых данных информации необходимой структуры для интерпретации. Источником данных может быть как БД, так и файлы заданной структуры, то же самое справедливо и для приемника. Конечной целью процесса является доставка данных потребителю в удобное для него представление, это может быть, как view в БД, так и предварительно написанная форма в приложении или на сайте [1]. Существует множество программных реализаций для доставки данных, однако большинство из них реализованы проприетарным ПО и подлежат лицензированию. В последнее время получили распространение решения, основанные на open source подходе, что является важным фактором при выборе, так как в текущих условиях наблюдаются проблемы при взаимодействиях с поставщиками ПО, поставкой покупкой лицензий, обновлениями имеющихся решений.

Преимуществами решений с открытым исходным кодом можно назвать возможности: экспериментировать с решениями, доработать продукт, проанализировать исходный код на наличие угроз гибкость разработки, а также отсутствие затрат на лицензии. Однако существуют и минусы – потеря интереса разработчиков к продукту может вызвать необходимость затрат собственных сил на разработку и поддержку, возникновение непроработанных решений внутри продукта, отсутствие необходимых интеграция с другими компонентами, требующее доработок, недостаточность функционала решения, требующее комбинировать ПО между собой. Одно из решений данных проблем описано в статье, также построена архитектура решения и создан реальный рабочий процесс, заменяющий аналогичный от частного вендора. Произведена оценка результатов решения.

## 2. ПОСТАНОВКА ЗАДАЧИ

Разработать решение на стеке open source технологий, позволяющее заменить частное ПО по доставке данных пользователю. Реализовать работающие ETL процессы с помощью нового ПО. Решение должно соответствовать требованиям отказоустойчивости, времени обработки и доставки данных, количеству передаваемых данных и затрачиваемых на работу ресурсов.

## 3. СУЩЕСТВУЮЩИЕ ЗАДАЧИ

Известно решение, использующее ПО IBM InfoSphere Datastage для доставки данных от поставщиков в хранилище данных. В данном ПО ETL процесс представляет собой поток из заданий, объединенный в единую управляющую последовательность, называемую Job Sequences [2]. Главное задание предоставляет единый интерфейс для передачи значений параметров контролируемым заданиям, управления последовательностью их исполнения, предоставляет механизмы ветвления и запуска заданий. Инструмент обладает широким функционалом и реализует написание последовательности с помощью графического интерфейса, внутри которого возможно параметризировать каждую из стадий, задавать свои запросы и их последовательность исполнения, реализовывать различные проверки данных.

Источниками данных для инструмента могут выступать файлы, последовательности файлов, реляционные базы данных, архивы, приложения. Реализован обширный набор коннекторов к различным базам данных, таких как Oracle, Sybase, Greenplum, HBase, Snowflake и др. Возможна настройка подключений через JDBC и ODBC. Внутри коннекторов доступна настройка правил проверки данных, так возможно проверить типы передаваемых данных, их содержимое и кодировку, наличие NULL значений, определить колонки, на основании которых необходимо формировать ключи. При управлении запросами к БД возможно задать пред-запрос, в котором сформировать временные таблицы для заполнения данными, сам запрос для извлечения данных и пост-запрос, использующийся, например, для очистки временных таблиц. В дополнительных настройках возможно настраивать буфер под запросы, партиционирование данных, ограничивать количество выдаваемых данных, настраивать уровни изоляции.

Работа с данными и их преобразование реализуется в ПО с помощью компонентов: Merge, Modify, Sort, Transformer, ряда генераторов данных, компонента удаления дубликатов и других. Компоненты позволяют из входных источников агрегировать, преобразовывать, очищать и дополнять данные по необходимым правилам. На выходе готовые данные можно как сразу отдавать потребителю для работы, так и использовать для последующих технических этапов подготовки данных.

Выгрузка данных в приемники аналогична по функционалу загрузке и является заключительным этапом в работе процесса. Стоит также отметить наличие контейнеров, позволяющих приводить к единообразию работу со многими источниками

в большом количестве заданий, так как в таком случае на выходе данные записываются в файлы заранее заданной структуры и затем использовать их для дальнейшей передачи по назначению.

Отдельно стоит отметить наличие компонентов онлайн обработки данных, таких как Kafka Connector, работа с IBM MQ, Streams Connector и другими. Основное применение Datastage заключается в обработке данных процессом Batch ETL. Наличие данных компонентов обеспечивает Datastage возможностями работы с Streaming ETL, при котором джоб в Datsatge может работать непрерывно, генерируя в ходе работы файлы для обработки или транслировать данные.

Sequence Job необходимы в DS для организации работы параллельных заданий в последовательность. Так возможно объединить ряд заданий с трансформациями данных в единую цепочку для загрузки во множество таблиц приемника. Есть опции организации циклов загрузки, ветвление последовательностей, реализации параллельности исполнения шагов, настройка вызовов системных скриптов и скриптов языков программирования, определить события, вызывающие прерывание исполнения заданий. Основное задание ставится на расписание с помощью встроенного в DS планировщика или через отдельный сервер – планировщик с помощью cron скриптов.

DS предоставляет инструменты управления выдачей прав пользователям согласно определенной ролевой модели. Так можно выдать права разработчикам на редактирование и запуск джобов, в то время как аналитики будут обладать правами только на чтение дизайна задания. Система логирования предоставляет разработчику полный журнал событий, с помощью которого можно разобраться со всеми ошибками, возникающими во время исполнения, отследить стабильность подключений и передачи данных.

Рассмотрим пример реализации решения передачи данных с помощью DS от источника потребителю. Задача состоит в том, чтобы передать данные о клиентах и их операциях из БД источника в БД приемника для построения отчетов об операциях. Для каждой таблицы источника реализуется ETL процесс с помощью параллельных джобов DS, объединяющий данные по смысловой нагрузке. Данные трансформируются в структуру, необходимую приемнику, очищаются от выбросов, возможно дополнение данных или пробелов в данных из других источников. Далее параллельные джобы объединяются в управляющем для построения логики заполнения таблиц (см. Рис. 1).

Инструмент использует внутренние средства и шифронаборы, проприетарный протокол однопроходного симметричного шифрования с использованием одного из тридцати двух 16-и битных ключей, выбираемых циклически. Ключи скрыто встроены в программное обеспечение.

Инструмент полностью реализует задачи работы с данными, однако обладает рядом критических недостатков. Последние версии ПО более не предоставляются для установки на изолированные сервера, а доступны только для облачных решений компании IBM. Таким образом можно сказать, что развитие инструмента прекратилось для компаний, который не имеют возможности использовать чужие облачные решения и должны хранить все данные внутри компании. Продление лицензирования для ПО требует значительных затрат и недоступно в некоторых регионах.

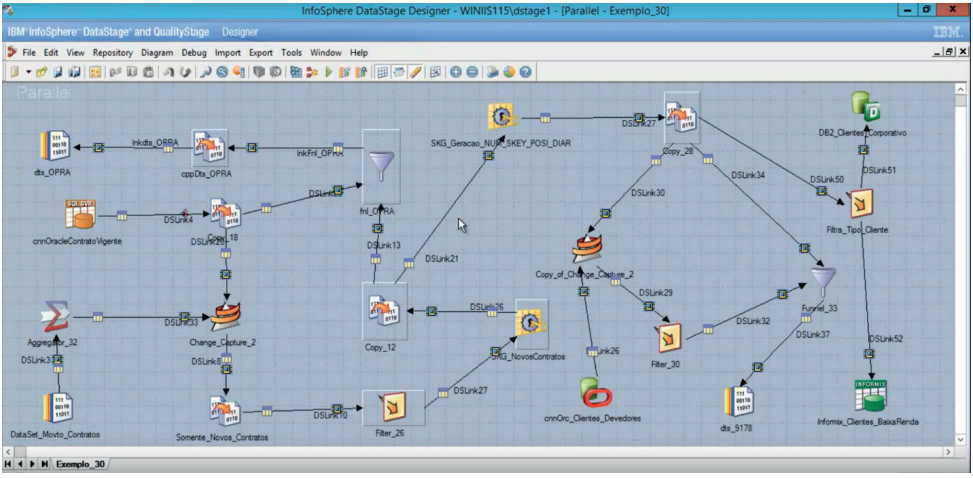


*Рис. 1. Пример ETL задания реализованного на ПО Datastage*

Известно решение, использующее Azure Data Factory (ADF) для передачи данных (см. Рис. 2). ADF – это облачный сервис, предназначенный для организации ETL процесса [3]. ПО требует наличия облачного хранилища данных для использования. Платформа решает задачи анализа, структурирования, обогащения данных из других источников. В ADF создаются конвейеры, которые могут принимать информацию из разрозненных хранилищ. Инструменты разработки реализованы с помощью визуальных представлений. Во многом инструмент завязан на продукты, предоставляемые компанией Microsoft Azure, поддержка в РФ прекращена.

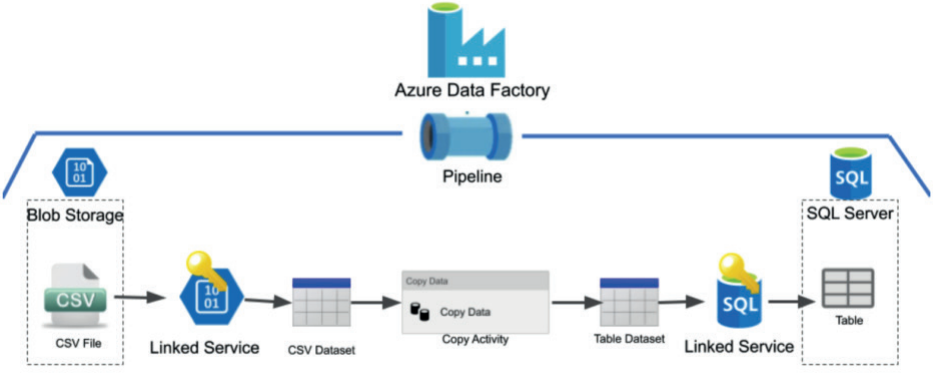


*Рис. 2. Пример ETL на Azure Data Factory*

Другим примером ETL инструмента можно назвать Oracle Data Integrator (ODI). Это комплексная платформа интеграции данных, которая охватывает все требования

к интеграции данных: от высокопроизводительных пакетных загрузок больших объемов до управляемых событиями процессов непрерывной интеграции и сервисов данных с поддержкой SOA. ODI поддерживает работу с большими данными за счет параллелизма при выполнении процессов интеграции данных.

ПО обладает рядом стандартных функций для создания и управления ETL процессом. В общем случае реализовывается механизм копирования данных из различных СУБД, преобразования и передачи в хранилище. Основными компонентами ПО являются дизайнер (средство для создания ETL), оператор (средство мониторинга), менеджер топологий (архив источников и потребителей данных), менеджер безопасности (инструмент выдачи прав пользователям), агент (оркестратор процессов). ODI во многом похож на IBM Datastage (см. Рис. 3), поддерживает использование условных переходов, циклов, обработчиков ошибок. Отличительным моментом можно назвать возможность инструмента ожидать наступления заранее заданного события. Т.е. работающий процесс ожидает появление файла по пути, и при его появлении обрабатывает. В DS же необходимо проверять наличие файла в таком случае по расписанию, каждые 5 минут или др. интервал времени запускать ETL джоб.

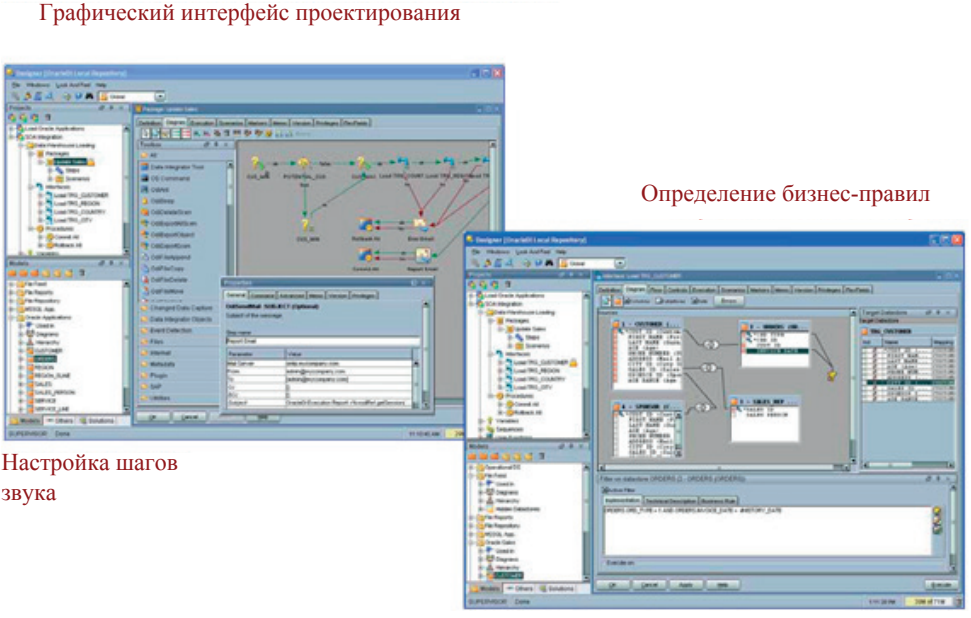


*Рис. 3. ETL процесс на ODI*

Анализ имеющихся решений на рынке показал, что большинство компаний, предоставляющих ETL инструменты, нацелены на внедрение этих инструментов исключительно в облачной среде с привязкой клиента к своему облаку для хранения данных. Такой подход не удовлетворяет все запросы потребителей ввиду необходимости хранения критических данных внутри компаний. Наблюдаются также проблемы

поддержки и сертификации решений в различных регионах, поддержка многих инструментов в РФ не оказывается.

## 4. АРХИТЕКТУРА РЕШЕНИЯ (ПО)

Разработана схема замены проприетарного решения на основе ПО IBM Datastage Open Source аналогами. Потоки 1, 4, 7, 10, 11, 13, 17 (см. Рис. 4) заменяются аналогичными 2, 3, 5, 6, 8, 9, 12, 14, 15, 18, 19. Поток передачи данных от источника в хранилище 1, 7, 10 на основе ПО от IBM заменяется потоками 2, 5, 14 и 3, 6, 15 для передачи стриминговых и пакетных данных. В качестве решения для пакетной передачи предлагается использовать ПО Apache Spark. Apache Kafka отвечает за передачу потоковых сообщений. HDFS предлагается использовать в качестве промежуточного хранилища выгружаемых данных, что позволит реализовать Data Lake подход, т.е. загрузить данные в исходном виде для последующей обработки.

Поток 17 возможно заменить потоками 18 и 19. Поток отвечает за транслирование данных от источника к потребителю в обход хранилища.

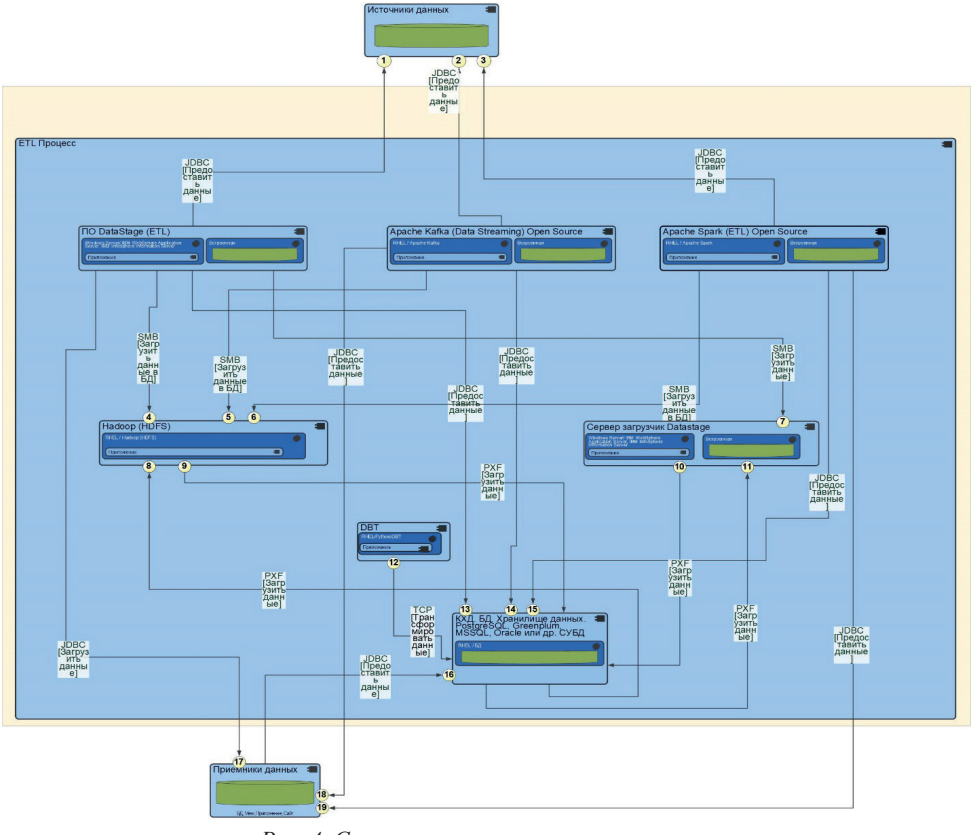


*Рис. 4. Схема архитектуры нового решения*

Потоки 10 и 11 исключаются при замене на новое ПО, вместо них используются потоки 8 и 9. Поток 16 отвечает за доставку данных от хранилища потребителю, может быть реализован как обращение клиента через ПО Aqua Data Studio, DBeaver напрямую к БД с помощью SQL запросов.

Поток 12 необходим для реализации трансформаций внутри хранилища данных. Инструмент DBT используется для облегчения процессов преобразования данных внутри хранилища, упрощения документирования функционала, а также дает возможность строить линии данных в удобном графическом представлении и упрощает процесс написания автотестов.

Установку ПО возможно проводить как на один сервер, так и на кластер серверов. Возможность масштабирования позволяет оптимально подобрать ресурсы под заданную задачу. Рекомендуется следующий порядок установки: Apache ZooKeeper (для настройки кластера), HDFS, Yarn, Hive, Apache Spark, Apache Kafka, DBT, Apache Airflow (для управления и планирования процессов) [4]. DBT и Apache Airflow необходимо разворачивать в разных venv для того, чтобы избежать возможных конфликтов библиотек. К ПО DBT необходимо будет дополнительно установить плагины коннекторы ко всем рабочим базам, пример dbt-postgre. В случае установки в офлайн-режиме необходимо собрать пакеты для DBT и Apache Airflow заранее со всеми зависимостями.

Для тестовой сборки был реализован кластер из одной мастер нодыдефис? и трех рабочих нод. На мастер ноду были установлены – DBT, Apache Airflow (Standalone версия), мастер ноды Kafka и Spark. На всех нодах был установлен HDFS, Apache Spark и Apache Kafka на рабочих нодах. В качестве хранилища данных выступали СУБД Greenplum и Postgre [5].

Определить правильно количество рабочих нод для Spark можно с помощью таблицы 1. В общем случае число зависит от объема обрабатываемых данных [6].

Таблица 1

**Методика определения количества рабочих нод для Spark**

| | |
|---|---|
| Объем обрабатываемых данных за сутки | x |
| Годовой объем обрабатываемых данных | z = 365*x |
| Процентный рост обрабатываемых данных за год | y % |
| Объем архивных данных | a |
| Общий объем данных за год | t = z + (y * z/100) + a |
| Коэффициент репликации 3 | r = t * 3 |
| Объем диска на рабочей ноде | w |
| Пространство задействованное под ОС | o = 0.2 * w |
| Доступный объем дискового пространства для рабочих нод | c = w – o |
| Количество рабочих нод | t/c |

Если в сутки обрабатывается 100 Гб данных, то в год эта цифра будет 36,5 ТБ. При проектировании кластера необходимо учитывать увеличение объемы данных, так как в общем случае количество данных все время растет. Для примера возьмем

значение в y = 20 %, а для a = 10 Тб архивных данных (необходимы в том случае, если возникнут пробелы при обработке). Таким образом t за год составляет 54 Тб. При проектировании кластера осуществляется установка коэффициента репликации, который копирует данные между несколькими рабочими узлами. При таком коэффициенте, равном 3, потребуется 162 Тб места для HDFS (если привязать Spark к Hadoop для хранения данных, альтернативный вариант S3).

Таким образом для сервера с 8 Тб дискового пространства нужно оставить 20 % объема под систему и ее нужды, тогда будет доступно 6 Тб дискового пространства. Результатом деления t на c получится, что для обработки такого объема данных потребуется 27 серверов. Добавим к этому числу 1 мастер ноду и 1 ноду Standby для резервирования мастера и получим 29 серверов для формирования рабочего промышленного кластера. Количество CPU и RAM можно подобрать экспериментально, основываясь на работе кластера. В общем случае минимальные требования для одного сервера: 8 CPU и 8 GB RAM; рекомендованные значения: 8 CPU и 16 GB RAM.

## 5. ПРОГРАММНАЯ РЕАЛИЗАЦИЯ ЗАДАЧИ

Был разработан аналог процесса, работающего на ПО IBM Datastage с помощью нового стека технологий. Обрабатывается информация о клиентах и их операциях за сутки, сырые данные выкладываются на HDFS, информация загружается из необработанного формата в КХД, в третьей нормальной форме соответствующей формату хранилища.

Управляет процессом Airflow, в нем определена последовательность запуска заданий, так называемых DAG-ов. Согласно расписанию запускаются задачи, отвечающие за выгрузку данных из источника с помощью Spark (см. Рис. 5). Далее запускается задание по загрузке файлов из HDFS в само хранилище, аналогично первому заданию.

Процесс также можно запускать вручную. Механизм работы Airflow позволяет гибко управлять заданиями, на всех этапах идет подробное логирование, в случае неисполнения какой-либо части логики можно вручную запустить только недостающие процессы, экономя время выполнения предыдущих шагов.

После отработки ряда первых двух заданий на выходе получаем данные в БД, готовые к дальнейшей обработке. При загрузке возможно точечно отфильтровать только необходимые колонки и записи. За дальнейшие трансформации внутри хранилища отвечает DBT. Данные распределяются внутри хранилища согласно его структуре, обогащаются при наличии информации от других источников. На основе итоговых таблиц строятся представления для конечного потребителя. Управляет последовательностью исполнения трансформаций Airflow (см. Рис. 6).

Обработка данных с помощью ПО Kafka во многом аналогична процессу, описанному для Spark [7]. Создается DAG с подключением к Kafka, далее задается потребитель данных, в нашем случае это хранилище, можно записывать информацию в файлы json для последующей обработки. Далее прописывается стадия приема сообщений от брокера [8].

```
from airflow import DAG
from airflow.providers.postgres.operators.postgres import PostgresOperator
from airflow.providers.apache.hdfs.operators.hdfs import HdfsPutFileOperator
from datetime import datetime, timedelta

default_args = {
    'owner': 'airflow',
    'depends_on_past': False,
    'start_date': datetime(2023, 3, 26),
    'retries': 1,
    'retry_delay': timedelta(minutes=5),
}

dag = DAG('postgresql_to_hdfs', default_args=default_args, schedule_interval='@dail

pg_operator = PostgresOperator(
    task_id='extract_data_from_postgresql',
    postgres_conn_id='my_postgresql_connection',
    sql='SELECT * FROM clients_operations;',
    dag=dag
)

hdfs_operator = HdfsPutFileOperator(
    task_id='load_data_to_hdfs',
    hdfs_conn_id='my_hdfs_connection',
    source='/tmp/clients_operations.csv',
    destination='/user/hadoop/clients_operations.csv',
    dag=dag
)

pg_operator >> hdfs_operator
```

*Рис. 5. Выгрузка данных из источника*

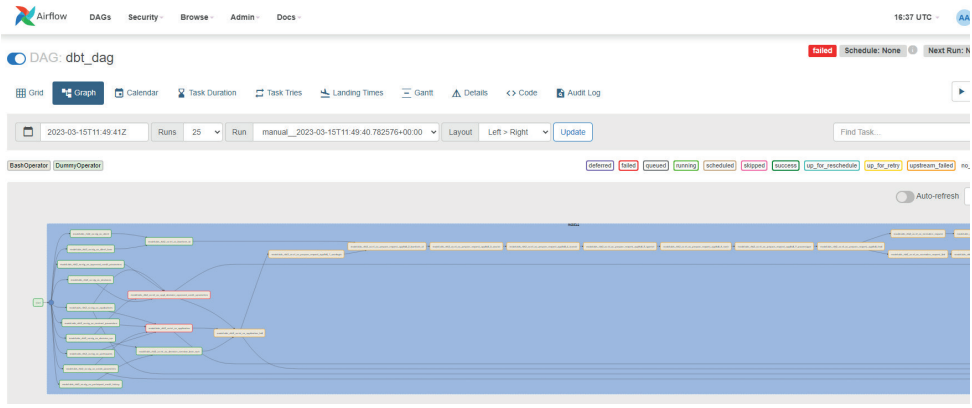


*Рис. 6. Обработка данных с помощью DBT*

## 6. ЗАКЛЮЧЕНИЕ

Работоспособность системы была проверена на тестовом наборе данных. Работа заданий показала, что обработка данных осуществляется согласно требованиям ко времени и к скорости выполнения. Управлять производительностью можно с помощью изменения характеристик серверов или добавлением новых нод к кластеру. Таким образом ресурсы, затрачиваемые на сертификацию, экономятся и позволяют использовать их для расширения рабочего кластера.

В работе представлена архитектура решения, потоки данных и способы их замещения, примеры реализации заданий с помощью open source ПО. Решение можно использовать взамен проприетарного ПО, ввиду наличия ограничений на его использование, аналог является полностью работоспособным и пригодным к применению.

Разработанная система может иметь широкое применение из-за своей доступности, открытости и наличия подробной документации в различных областях, ставящих своей задачей обработку данных. Сами программные продукты активно развиваются, обновления ускоряют обработку данных, расширяют функционал, добавляют новые удобные инструменты для разработки и автоматизации процессов.

***Литература***

1. *David Loshin.* ETL (Extract, Transform, Load) // Business Intelligence. – 2nd. – Morgan Kaufmann, 2012. – 400 p
2. *Ralph Kimball, Joe Caserta.* The Data Warehouse ETL Toolkit: Practical Techniques for Extracting, Cleaning, Conforming, and Delivering Data. – John Wiley & Sons, 2004. – 528 p.
3. *David Haertzen.* ETL Tools // The Analytical Puzzle: Profitable Data Warehousing, Business Intelligence and Analytics. – Technics Publications, 2012. – 346 p.
4. *С. Риза, У. Лезерсон, Ш. Оуэн, Д. Уиллс.* Spark для профессионалов: современные паттерны обработки больших данных = Advanced Analytics with Spark. Patterns for Learning from Data at Scale (O'Reilly, 2015). 2017. – 272 с.

5. *Уоррен Р., Карау Х.* Эффективный Spark. Масштабирование и оптимизация = High Performance Spark. Best Practices for Scaling and Optimizing Apache Spark. 2018. – 352 с.
6. *Х. Карау, Э. Конвински, П. Венделл, М. Захария.* Изучаем Spark. Молниеносный анализ данных = Learning Spark: Lightning-Fast Big Data Analytics (O'Reilly, 2015). 2015. – 304 с.
7. *Нархид Ния, Шапира Гвен, Палино Тодд.* Apache Kafka. Потоковая обработка и анализ данных. – СПб., 2019 320 с.
8. *Vohra, Deepak* (October 2016). Practical Hadoop Ecosystem: A Definitive Guide to Hadoop-Related Frameworks and Tools (1st ed.). Apress. p. 429.

# Development of an ETL Process Based on Open Source Technologies to Solve the Problem of Data Delivery to Consumers

***Vyacheslav V. Starkov****
Moscow Institute of Steel and Alloys (National Research Technological University) (NUST MISIS), Moscow, Russia
ORCID: https://orcid.org/0009-0006-0662-4852
e-mail: starkov.viatcheslav@yandex.ru

***Svetlana S. Gorbatova ******
Moscow Institute of Steel and Alloys (National Research Technological University) (NUST MISIS), Moscow, Russia
ORCID: https://orcid.org/0009-0005-5213-6780
e-mail: ssgorbatova@misis.ru

***Victoria I. Vodolaga*****
Lomonosov Moscow State University (MSU), Moscow, Russia
ORCID: https://orcid.org/0009-0003-1816-0088
e-mail: vikavodolaga1@gmail.com

The article discusses the issues of developing an ETL process for a data warehouse based on open source technologies, instead of private software supplied by the vendor. The process allows you to deliver data from the source to the consumer, focusing on the speed of delivery, the resources spent and the convenience of development. The architecture for solving the problem with a description of the processes being replaced is presented, data transmission over a new process is implemented. Modern tools used to work with data are involved, methods of interaction with them and selection of technical characteristics for the process are described.

***Keywords:*** database, open source, software, ETL process, data delivery.

**For citation:**
Starkov V.V., Gorbatova S.S., Vodolaga V.I. Development of an ETL Process Based on Open Source Technologies to Solve the Problem of Data Delivery to Consumers. *Modelirovanie i analiz dannykh = Modelling and Data Analysis*, 2023. Vol. 13, no. 2, pp. 180–193. DOI: 10.17759/mda.2023130210 (In Russ., abstr. in Engl.).

***Vyacheslav V. Starkov,** Postgraduate Student, Moscow Institute of Steel and Alloys (National Research Technological University) (NUST MISIS), Moscow, Russia, ORCID: https://orcid.org/0009-0006-0662-4852 ,e-mail: starkov.viatcheslav@yandex.ru

****Svetlana S. Gorbatova,** Senior Lecturer, Moscow Institute of Steel and Alloys (National Research Technological University) (NUST MISIS), Moscow, Russia, ORCID: https://orcid.org/0009-0005-5213-6780 , e-mail: ssgorbatova@misis.ru

*****Victoria I. Vodolaga,** Master's Degree, Lomonosov Moscow State University (MSU), Moscow, Russia, ORCID: https://orcid.org/0009-0003-1816-0088, e-mail: vikavodolaga1@gmail.com

***References***

1. David Loshin. ETL (Extract, Transform, Load) . Business Intelligence. – 2nd. – Morgan Kaufmann, *2012. – 400 p*
2. Ralph Kimball, Joe Caserta. The Data Warehouse ETL Toolkit: Practical Techniques for Extracting, Cleaning, Conforming, and Delivering Data. – John Wiley & Sons, *2004. – 528 p.*
3. David Haertzen. ETL Tools . The Analytical Puzzle: Profitable Data Warehousing, Business Intelligence and Analytics. – *Technics Publications, 2012. – 346 p.*
4. S. Riza, U. Lezerson, Sh. Ouen, D. Uills. Spark dlya professionalov: sovremennye patterny obrabotki bol'shikh dannykh = Advanced Analytics with Spark. Patterns for Learning from Data at Scale (O'Reilly, 2015). *2017. – 272 p.*
5. Uorren R., Karau Kh. Effektivnyi Spark. Masshtabirovanie i optimizatsiya = High Performance Spark. Best Practices for Scaling and Optimizing Apache Spark. *2018. – 352 s.*
6. Kh. Karau, E. Konvinski, P. Vendell, M. Zakhariya. *Izuchaem Spark. Molnienosnyi analiz dannykh* = Learning Spark: Lightning-Fast Big Data Analytics (O'Reilly, 2015). *2015. – 304 s.*
7. Narkhid Niya, Shapira Gven, Palino Todd. Apache Kafka. Potokovaya obrabotka i analiz dannykh. – SPb., 2019 *p = 320.*
8. Vohra, Deepak (October 2016). Practical Hadoop Ecosystem: A Definitive Guide to Hadoop-Related Frameworks and Tools (1st ed.). *Apress. p. 429.*

Получена 12.04.2023 Received 12.04.2023
Принята в печать 12.05.2023 Accepted 12.05.2023