DOI 10.37882/2223-2966.2026.02-2.21

АДАПТИВНЫЙ ГИБРИДНЫЙ МЕТОД МИГРАЦИИ БОЛЬШИХ ДАННЫХ С АВТОМАТИЧЕСКИМ ВЫБОРОМ СТРАТЕГИИ ИЗВЛЕЧЕНИЯ И ОБРАБОТКИ ИНФОРМАЦИИ

ADAPTIVE HYBRID METHOD FOR MIGRATING LARGE DATA WITH AUTOMATIC SELECTION OF INFORMATION EXTRACTION AND PROCESSING STRATEGIES

S. Sosenushkin
A. Badanov

Summary. The article discusses the features of big data migration in the context of using heterogeneous storage systems and non-standard information sources. It provides an overview of existing data migration methods and identifies their limitations when working with large amounts of data and various access methods. The article proposes an adaptive hybrid method for big data migration with automated selection of information extraction and processing strategies, based on a combination of screen robots, classical ETL approaches, distributed data processing, and the use of an intermediate storage. It has been shown that the proposed method is a promising approach to organizing large-scale data migration processes and is of interest for further research in the field of building scalable and sustainable migration solutions.

Keywords: big data, data migration, ETL, screen robots, RPA, staging, data transformation.

Сосенушкин Сергей Евгеньевич
Кандидат технических наук, доцент,
ФГБОУ ВО «МГТУ «СТАНКИН»
ss@stankin.ru
Баданов Артем Андреевич
ведущий разработчик ООО «Амбердата»;
ФГБОУ ВО «МГТУ «СТАНКИН»
artem_badanov@inbox.ru

Аннотация. В статье рассматриваются особенности миграции больших данных в условиях использования разнородных хранилищ и нестандартных источников информации. Проведен обзор существующих методов миграции данных и выявлены их ограничения при работе с большими объемами данных и различными способами доступа к источникам. Предложен адаптивный гибридный метод миграции больших данных с автоматизированным выбором стратегии извлечения и обработки информации, основанный на сочетании экранных роботов, классических ETL-подходов, распределенной обработки данных и использования промежуточного хранилища. Показано, что предложенный метод является перспективным подходом к организации процессов миграции больших данных и представляет интерес для дальнейших исследований в области построения масштабируемых и устойчивых миграционных решений.

Ключевые слова: большие данные, миграция данных, ETL, экранные роботы, RPA, промежуточное хранилище, трансформация данных.

Миграция больших данных представляет собой процесс переноса значительных объемов информации между разнородными информационными системами и хранилищами с сохранением корректности, целостности и смысловой интерпретации данных. В условиях современного развития информационных технологий миграция данных становится неотъемлемой частью модернизации инфраструктуры, перехода на новые вычислительные платформы, консолидации источников данных и внедрения аналитических систем нового поколения.

В отличие от классической миграции данных, миграция больших данных характеризуется одновременной обработкой от нескольких десятков до сотен разнородных источников данных, объемами передаваемой информации, достигающими десятков терабайт, а также использованием различных моделей хранения данных, включая реляционные, колоночные и документно-ориентированные хранилища. Процессы миграции при этом должны обеспечивать заданные значения пропускной способности (не менее 10^6–10^7 записей в секунду), допустимого времени простоя целевых систем (не более нескольких минут) и уровня отказоустойчивости при сбоях отдельных компонентов. Совокупность указанных факторов обусловливает необходимость применения специализированных методов извлечения, преобразования и загрузки данных и приводит к усложнению архитектуры миграционных решений.

Одними из первых систематизированных подходов к миграции данных стали методы, основанные на концепции ETL (Extract, Transform, Load), активно развиваемые в рамках построения хранилищ данных. Данный подход получил широкое распространение в работах, посвящённых архитектуре корпоративных хранилищ данных, в частности в исследованиях Р. Кимбалла, где процессы ETL рассматриваются как ключевой механизм

обеспечения качества, согласованности и интеграции данных [16]. Альтернативный архитектурный подход был предложен Б. Инмоном, который рассматривал миграцию данных в контексте централизованных корпоративных хранилищ и подчёркивал важность формализации структур, метаданных и корпоративных моделей данных [17].

С развитием распределённых вычислений и появлением масштабируемых аналитических платформ получил распространение подход ELT (Extract, Load, Transform), в рамках которого основные операции преобразования данных переносятся на сторону целевого хранилища. Данный метод оказался особенно востребованным в условиях использования колоночных и облачных аналитических систем, обладающих значительными вычислительными ресурсами и возможностями параллельной обработки данных.

В то же время рост объёмов данных, увеличение количества разнородных источников и повышение требований к актуальности информации привели к формированию подходов, выходящих за рамки классических схем ETL и ELT. Одним из таких направлений является **потоковая миграция данных**, при которой перенос и обработка информации выполняются в режиме непрерывного поступления данных из источников. В отличие от ETL, ориентированного преимущественно на пакетную обработку, потоковая миграция позволяет существенно снизить задержки доставки данных в целевую систему, однако усложняет обеспечение согласованности и требует дополнительных механизмов управления состоянием и обработки ошибок.

Другим развитием традиционных методов является **миграция на основе захвата изменений (Change Data Capture, CDC)**, предполагающая передачу в целевую систему только изменений, зафиксированных в журналах транзакций исходных систем. По сравнению с классическим ETL данный подход позволяет сократить объём передаваемых данных и снизить нагрузку на источники, а также обеспечить близкую к реальному времени синхронизацию. Вместе с тем CDC, как правило, применяется в сочетании с ETL— или ELT-процессами и не заменяет их полностью, поскольку не решает задачи первичной загрузки и комплексного преобразования исторических данных.

Отдельную группу методов образуют **подходы логической виртуализации данных**, при которых физическое перемещение данных между системами отсутствует либо сводится к минимуму. В рамках данного подхода создаётся логический слой, обеспечивающий унифицированный доступ к распределённым источникам данных без их предварительной миграции. По сравнению с ETL, логическая виртуализация позволяет сократить время интеграции новых источников и снизить затраты на хранение дублирующихся данных. Однако данный метод накладывает дополнительные требования к производительности источников и сетевой инфраструктуры, а также ограничивает возможности глубокой оптимизации и долгосрочного хранения данных в целевой системе. В результате логическая виртуализация чаще применяется как временное или вспомогательное решение либо как часть гибридных архитектур миграции.

На практике всё большее распространение получают **гибридные подходы**, сочетающие элементы ETL, ELT, потоковой обработки, CDC и логической виртуализации. В таких архитектурах первичная миграция и историческая загрузка данных могут выполняться по классической ETL-модели, в то время как актуализация данных обеспечивается средствами CDC или потоковой обработки, а доступ к отдельным источникам реализуется через виртуализированный логический слой. По сравнению с традиционным ETL данный подход позволяет повысить масштабируемость и гибкость миграционных процессов, однако существенно усложняет архитектуру и требует развитых средств оркестрации, мониторинга и управления метаданными.

Таким образом, ETL и ELT следует рассматривать как базовые архитектурные модели миграции данных, на основе которых сформировались более сложные и специализированные методы. В условиях миграции больших данных выбор конкретного подхода или их комбинации определяется объёмами информации, требованиями к задержкам обработки, характеристиками источников и целевых платформ, а также допустимым уровнем архитектурной сложности миграционного решения.

Дальнейшее развитие методов миграции больших данных связано с появлением распределенных фреймворков обработки данных, таких как платформы пакетной и потоковой обработки, ориентированные на работу с внутренними структурами данных и масштабируемое выполнение вычислений. Эти подходы позволили вынести ресурсоемкие операции обработки за пределы систем-источников и существенно повысить производительность миграционных процессов.

Несмотря на разнообразие существующих методов миграции данных, большинство из них ориентированы на использование формализованных источников данных и предполагают наличие стандартных интерфейсов доступа, таких как API или прямое подключение к хранилищам. В то же время в реальных условиях миграции больших данных источниками информации нередко выступают нестандартные системы, включая корпоративные порталы и веб-интерфейсы, не предназначенные для автоматизированного извлечения данных. Кроме того, существующие методы, как правило, не предусма-

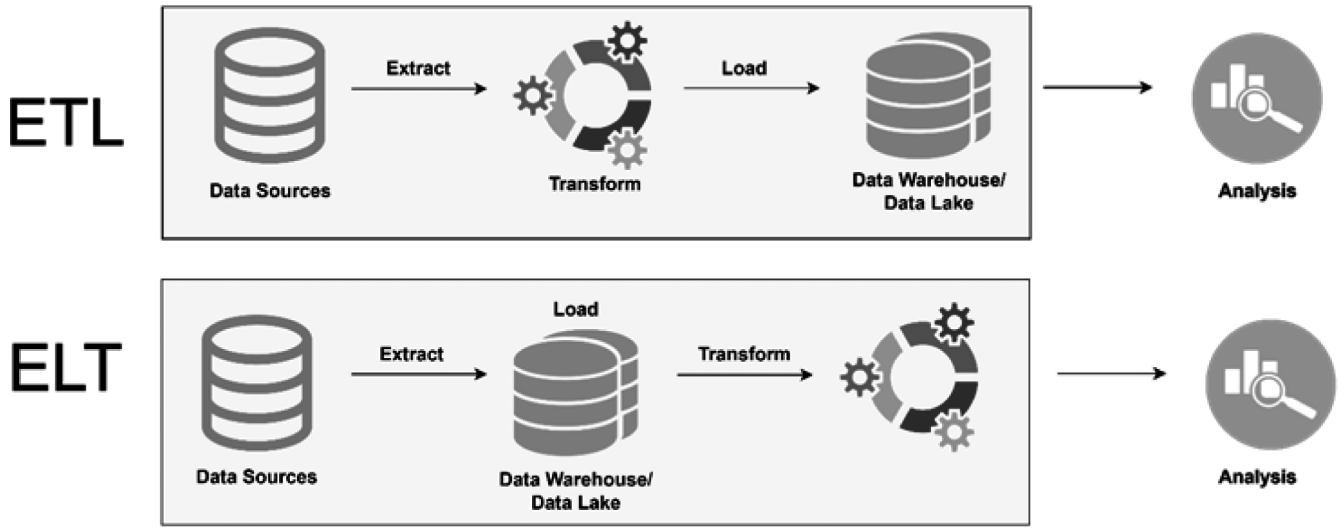

Рис. 1. Визуальный процесс переноса данных в ETL/ELT

тривают адаптивного выбора стратегии миграции в зависимости от характеристик источника и доступных способов извлечения информации.

В связи с этим актуальной является задача разработки адаптивного гибридного метода миграции больших данных, способного автоматически определять оптимальную стратегию извлечения и обработки информации, включая использование как классических ETL/ELT-подходов, так и методов экранного извлечения данных. Решению данной задачи и посвящено дальнейшее изложение.

Прежде чем приступить к рассмотрению предложенного адаптивного гибридного метода миграции больших данных, целесообразно проанализировать особенности применения экранных роботов в контексте извлечения и последующей миграции информации. Данный анализ необходим для корректного понимания ограничений существующих подходов и обоснования необходимости автоматического выбора стратегии извлечения данных.

Экранные роботы представляют собой программные средства, имитирующие действия пользователя при взаимодействии с пользовательским интерфейсом информационной системы. В отличие от классических методов миграции, основанных на использовании формализованных интерфейсов доступа к данным, экранные роботы работают на уровне визуального представления информации и не требуют прямого доступа к внутренним структурам хранилища данных.

В контексте миграции больших данных экранные роботы применяются в тех случаях, когда источник информации не предоставляет API, прямого подключения к базе данных или иных стандартных механизмов извлечения данных. Типичным примером таких источников являются корпоративные порталы, внутренние информационные системы и веб-приложения, доступ к которым ограничен политиками безопасности и реализован исключительно через веб-интерфейс. В подобных условиях использование экранных роботов является единственным возможным способом автоматизированного извлечения информации.

Процесс извлечения данных с использованием экранных роботов, как правило, включает анализ структуры пользовательского интерфейса, идентификацию элементов, содержащих целевую информацию, и последующий сбор данных с использованием методов парсинга. Полученные данные сохраняются во временном или промежуточном хранилище, после чего могут быть включены в основной процесс миграции, включающий этапы обработки, преобразования и загрузки данных [2, 5, 11].

Несмотря на практическую применимость, использование экранных роботов в задачах миграции больших данных сопровождается рядом ограничений. Во-первых, данный подход обладает низкой устойчивостью к изменениям интерфейса источника данных, так как даже незначительные изменения структуры страниц могут привести к нарушению работы робота. Во-вторых, экранное извлечение, как правило, характеризуется меньшей производительностью по сравнению с использованием формализованных интерфейсов доступа, что ограничивает масштабируемость данного метода при работе с большими объемами данных [6, 7, 14].

Кроме того, экранные роботы не обладают встроенными механизмами семантического анализа данных и, как правило, извлекают информацию в виде слабо структурированных наборов значений. Это приводит к необходимости выполнения дополнительных этапов обработки и преобразования данных перед их включе-

нием в процесс миграции. В условиях больших объемов данных данные этапы могут становиться ресурсозатратными и повышать вероятность возникновения ошибок.

Следует также отметить, что экранные роботы в большинстве случаев используются изолированно от классических методов миграции данных и не интегрированы в единый адаптивный механизм выбора стратегии извлечения информации. Отсутствие автоматизированных гибридных решений, способных динамически определять возможность использования API или необходимость экранного извлечения, существенно ограничивает эффективность существующих подходов к миграции больших данных. [1, 8, 9]

Архитектура экранных роботов, применяемых для извлечения данных из нестандартных источников, как правило, имеет многоуровневую структуру, ориентированную на последовательное взаимодействие с пользовательским интерфейсом системы-источника. В общем случае архитектуру экранного робота можно представить в виде набора функциональных компонентов, каждый из которых выполняет строго определенную роль в процессе извлечения информации.

Ключевым элементом архитектуры является модуль управления пользовательским интерфейсом, отвечающий за навигацию по страницам, взаимодействие с элементами интерфейса и эмуляцию действий пользователя. Данный модуль функционирует в строгой последовательности шагов, так как большинство пользовательских интерфейсов не допускают параллельного выполнения действий. Это обстоятельство накладывает фундаментальное ограничение на степень параллелизма экранного извлечения данных.

Следующим компонентом является модуль анализа структуры интерфейса, предназначенный для идентификации элементов, содержащих целевую информацию. Данный модуль опирается на анализ DOM-структуры, визуальных атрибутов или шаблонов страниц. Его работа тесно связана с текущей реализацией интерфейса источника данных, что делает архитектуру экранных роботов чувствительной к изменениям внешнего вида и логики страниц.

Извлеченные данные передаются в модуль парсинга и первичной структуризации, где информация преобразуется из визуального представления в структурированный или полуформализованный вид. На данном этапе, как правило, отсутствует глубокий семантический анализ, вследствие чего данные требуют дополнительной обработки перед включением в процесс миграции [2, 4].

С точки зрения производительности архитектура экранных роботов обладает выраженной последовательной составляющей, что напрямую связано с ограничениями пользовательского интерфейса. Даже при наличии технической возможности запуска нескольких экземпляров робота, значительная часть операций — таких как загрузка страниц, ожидание отклика интерфейса и переходы между состояниями — выполняется последовательно. Данное обстоятельство позволяет напрямую применить закон Амдала, согласно которому ускорение вычислительного процесса ограничено долей его последовательной части [6, 14].

В контексте экранных роботов это означает, что увеличение числа параллельных исполнителей приводит к ограниченному росту производительности, так как критические участки процесса извлечения данных не поддаются эффективному распараллеливанию. Более того, попытки масштабирования экранного извлечения часто приводят к дополнительным накладным расходам, связанным с конкуренцией за ресурсы интерфейса и ограничениями со стороны системы-источника.

Таким образом, архитектура экранных роботов изначально не предназначена для высокопроизводительной обработки больших объемов данных. Ее сильной стороной является возможность извлечения информации из источников, недоступных для классических методов миграции, однако архитектурные особенности приводят к низкой масштабируемости и ограниченной эффективности при росте объемов данных.

Но, все же вернемся к рассмотрению ограничений, возникших ранее. Следует отметить, что часть ограничений экранного извлечения данных может быть существенно смягчена за счет архитектурных и организационных решений.

Так, для приближения к решению проблемы низкой устойчивости экранных роботов к изменениям пользовательского интерфейса целесообразно предварительно выполнять динамический анализ структуры страницы до начала непосредственного извлечения данных. Такой анализ может осуществляться на основе парсинга HTML-разметки, а также связанных с ней JavaScript— и CSS-компонентов, формирующих итоговую структуру пользовательского интерфейса.

Полученное представление структуры страницы целесообразно сопоставлять с эталонным описанием интерфейса, зафиксированным на этапе разработки или предыдущих успешных запусков миграции. Сравнение текущей структуры с эталонной позволяет выявлять изменения в расположении элементов, их атрибутах и взаимосвязях, которые могут повлиять на корректность работы экранного робота. В случае обнаружения существенных расхождений процесс извлечения данных может быть приостановлен либо переведен в безопасный

режим, что предотвращает выполнение некорректных операций в продуктивной среде.

В качестве одного из возможных вариантов реализации данного подхода может рассматриваться использование компактной модели машинного обучения или интеллектуального модуля анализа, способного автоматически сопоставлять текущую структуру страницы с эталонной и выявлять допустимые и критические изменения. Такой модуль может использоваться для частичной автоматизации корректировки сценариев экранного извлечения или формирования рекомендаций по их адаптации.

Применение динамического анализа структуры страницы в сочетании с эталонным сравнением и интеллектуальными механизмами проверки позволяет существенно повысить устойчивость экранных роботов к изменениям пользовательского интерфейса и снизить вероятность ошибок при извлечении данных в процессе миграции больших данных [1, 8, 9] .

Оптимизация производительности экранного извлечения данных может быть достигнута за счет отказа от использования полнофункционального браузера в тех случаях, когда это не является необходимым. Применение облегченных механизмов взаимодействия с веб-страницами, а также минимизация визуальных операций позволяют сократить накладные расходы и частично компенсировать ограничения, связанные с последовательным характером работы экранных роботов.

Ограниченность экранных роботов в части семантического анализа данных не является критической проблемой, а представляет собой архитектурное ограничение, с которым целесообразно работать на последующих этапах миграции. Семантический анализ, очистка и структурирование данных могут быть эффективно выполнены на основном этапе миграции, в рамках операций обработки и преобразования информации. В частности, на этапе ETL данные, извлеченные экранными роботами в сыром виде, могут быть агрегированы, отфильтрованы и приведены к требуемой структуре с использованием более мощных и формализованных инструментов обработки данных.

Таким образом, экранное извлечение данных следует рассматривать как вспомогательный механизм, предназначенный для получения информации из нестандартных источников, тогда как основная логика обработки, агрегации и семантического анализа данных переносится на этапы классической миграции. Такое разделение ответственности позволяет минимизировать влияние ограничений экранных роботов и обеспечить более устойчивую и управляемую миграцию больших данных в рамках гибридного подхода.

Исходя из полученного архитектурного понимания работы экранных роботов, представляется целесообразным интегрировать данный механизм в общий метод «Адаптивный гибридный метод миграции больших данных с автоматическим выбором стратегии извлечения и обработки информации». Ключевой задачей на данном этапе является формализация входных параметров миграции и создание универсального механизма принятия решений о способе извлечения данных.

При выполнении любой миграции больших данных, как правило, формируются конфигурационные файлы (conf), содержащие сведения о доступах к источникам данных, сетевых параметрах, перечне объектов для переноса, а также описания метаданных, подлежащих миграции. Такие конфигурационные файлы могут быть представлены в различных форматах, однако их ключевой особенностью является наличие структурированной информации, достаточной для автоматизированного анализа источника данных [6, 7].

Пример conf файла ниже —

```
{
 «migration_id»: «migration_2026_01»,
 «source»: {
 «engine»: «postgresql»,
 «host»: «source-db.company.local:5432»,
 «db»: «source_db»,
 «user»: «migration_user»,
 «password»: «***»,
 «tables»: [«orders», «customers», «payments»]
 },
 «target»: {
 «engine»: «clickhouse»,
 «host»: «target-ch.company.local:8123»,
 «db»: «analytics»,
 «user»: «ch_user»,
 «password»: «***»
 },
 «intermediate»: {
 «type»: «s3»,
 «endpoint»: «https://s3.internal.company.local»,
 «bucket»: «migration-temp-data»,
 «formats»: [«parquet»]
 },
 «transform»: {
 «enabled»: true,
 «rules»: «git://repo.company.local/migration/transform_
rules»
 },
 «execution»: {
 «strategy»: «adaptive»,
 «screen_allowed»: true,
 «parallelism»: 8
 }
}
```

В представленном примере конфигурационного файла в явном виде описаны параметры источника и приемника данных, способы подключения, перечень объектов для переноса, а также параметры промежуточного хранилища. Наличие раздела, описывающего временное объектное хранилище, позволяет использовать его в качестве универсального промежуточного слоя для хранения результатов преобразования данных и фиксации промежуточных состояний миграции. Для снижения вероятности ошибочного изменения метаинформации и упрощения ввода параметров конфигурации целесообразно использовать пользовательский интерфейс, формирующий конфигурационные файлы в автоматическом режиме.

Выбор файлового хранилища в качестве промежуточного обусловлен его универсальностью и независимостью от конкретных моделей хранения данных. Использование объектного или файлового хранилища позволяет сохранять результаты этапа Transform в нейтральном формате и обеспечивает устойчивость процесса миграции к сбоям на последующих этапах. Промежуточный характер данного хранилища заключается в том, что после выполнения операций преобразования данные требуют временного размещения до их успешной загрузки в итоговую систему [13].

В случае прямой загрузки преобразованных данных в целевое хранилище при возникновении ошибки на этапе Load существует риск появления несогласованных данных, что приводит к необходимости повторного выполнения как этапа Transform, так и этапа Load. Использование промежуточного файлового хранилища позволяет избежать данной ситуации, так как при сбоях на этапе загрузки требуется повторное выполнение только операции Load, тогда как результаты преобразования данных сохраняются и не требуют повторной обработки. При этом вероятность возникновения ошибок на этапе Load существенно ниже, поскольку загрузка данных без выполнения преобразований является малонагруженной и технологически простой операцией.

Структурированность данной информации делает возможным ее использование в рамках предлагаемого метода за счет создания управляющей подсистемы, реализующей логику адаптивного выбора стратегии миграции. Данная подсистема осуществляет чтение и интерпретацию конфигурационных файлов, извлекая из них сведения о типе источника данных, доступных способах подключения, наличии формализованных интерфейсов доступа и характеристиках передаваемой информации.

Благодаря строго структурированному описанию параметров миграции в конфигурационных файлах обеспечивается возможность формализованного принятия решений без участия оператора. Это позволяет унифицировать процесс миграции, снизить вероятность ошибок, связанных с ручной настройкой, и обеспечить масштабируемость метода при работе с большим количеством разнородных источников данных.

Однако даже при отсутствии оператора на этапах определения источника данных и выбора стратегии извлечения, участие разработчика остается необходимым на этапе Transform. Это обусловлено тем, что логика преобразования данных, включая правила фильтрации, агрегации и семантической интерпретации информации, напрямую зависит от требований предметной области и особенностей целевого использования данных.

В отличие от этапов извлечения и загрузки данных, которые могут быть формализованы и автоматизированы на основе конфигурационных параметров, этап преобразования данных требует учета бизнес-логики, контекста использования информации и неявных зависимостей, которые, как правило, не отражены в технических описаниях источников данных. Определение корректных правил трансформации данных невозможно без участия разработчика или эксперта предметной области, обладающего знанием о смысловой нагрузке обрабатываемой информации.

При этом следует отметить, что необходимость участия разработчика на этапе Transform возникает не во всех сценариях миграции данных. В случае, если задача сводится к прямому переносу данных из источника А в источник В без изменения структуры, логики обработки и семантического содержания информации, процесс миграции может быть выполнен полностью в автоматическом режиме [2, 15].

В подобных сценариях отсутствует необходимость в разработке сложных правил преобразования данных, а этап Transform либо минимален, либо сводится к базовому приведению форматов и типов данных, которое может быть формализовано в конфигурационных файлах. Благодаря этому адаптивный гибридный метод позволяет выполнять простые миграции данных без привлечения разработчика, используя автоматический выбор стратегии извлечения и стандартные механизмы загрузки данных в систему-приемник.

Для того, чтобы описанный выше пайплайн был более понятен, ниже доступна его визуализация (см. рис. 2).

Важно отметить, что в рамках предлагаемого метода на этапе Transform может применяться использование внешних инструментов параллельной обработки данных, таких как распределённые вычислительные фреймворки. Привлечение подобных инструментов обусловлено необходимостью распараллеливания операций обработки информации и сокращения времени выпол-

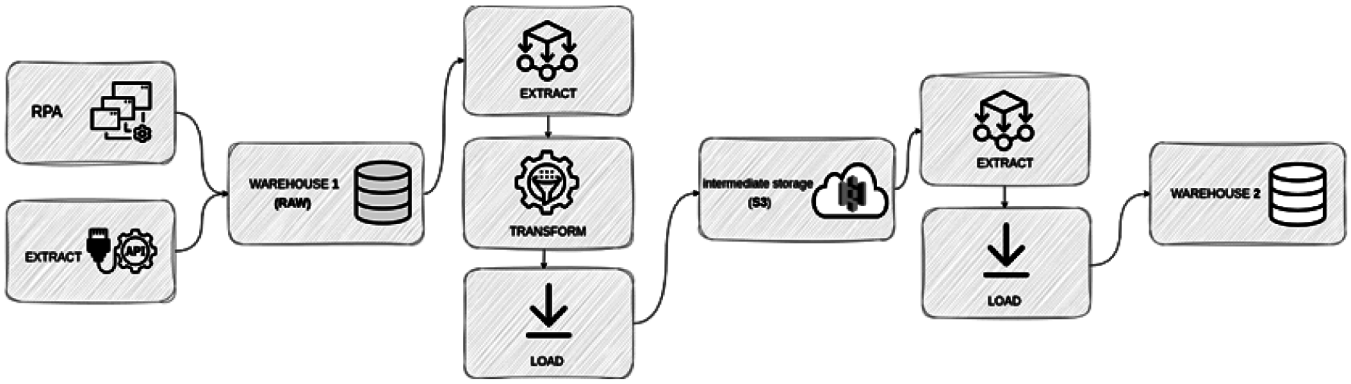

Рис. 2. Визуализация пайплайна миграции больших данных

нения вычислений при работе с большими объемами данных.

Использование параллельной обработки на этапе Transform позволяет вынести ресурсоемкие операции агрегации, фильтрации и преобразования данных за пределы исходного хранилища, снижая нагрузку на его вычислительные ресурсы. Однако применение таких инструментов не является безусловным: в случае отсутствия в используемом фреймворке необходимых методов обработки данных либо при наличии различий в семантике их работы по сравнению с методами исходного хранилища, обработка данных выполняется непосредственно на ресурсах системы-источника.

Решение о применении параллельной обработки принимается автоматически на основе интегрированной библиотеки методов, входящей в состав предлагаемого метода. Данная библиотека осуществляет сопоставление функциональных возможностей инструментов параллельной обработки с методами, используемыми в исходном хранилище, и проверяет эквивалентность их поведения. В случае, если функциональность методов совпадает и не приводит к искажению результатов обработки данных, для выполнения этапа Transform используется внешняя параллельная среда. В противном случае обработка выполняется в исходном хранилище данных.

Дополнительным преимуществом применения параллельных инструментов обработки является возможность динамического управления вычислительными ресурсами. Параметры вычислительной сессии могут адаптивно изменяться в зависимости от объема обрабатываемых данных: при увеличении объема данных ресурсы масштабируются, при снижении — сокращаются. Такой подход позволяет не только эффективно выполнять параллельную обработку данных в рамках одной миграции, но и запускать несколько миграционных процессов одновременно в условиях ограниченных вычислительных ресурсов.

Использование параллельных инструментов обработки данных в составе адаптивного гибридного метода обеспечивает баланс между производительностью, корректностью обработки информации и рациональным использованием вычислительных ресурсов при миграции больших данных.

В рамках данной статьи был предложен адаптивный гибридный метод миграции больших данных с автоматическим выбором стратегии извлечения и обработки информации, ориентированный на использование в условиях разнородных источников данных, архитектурных ограничений и больших объемов обрабатываемой информации.

Ключевой особенностью предлагаемого метода является интеграция нескольких подходов и инструментов в рамках единой архитектуры. Метод предусматривает использование экранных роботов (RPA) для извлечения данных из нестандартных источников, не предоставляющих формализованных интерфейсов доступа, что позволяет расширить область применимости миграции за пределы традиционных хранилищ данных. Для обеспечения устойчивости и управляемости процесса миграции используется промежуточное хранилище, выступающее в качестве универсального слоя хранения результатов извлечения и преобразования данных и обеспечивающее возможность восстановления процесса миграции в случае сбоев.

Дополнительно в методе используется распределённый фреймворк обработки данных, позволяющий выполнять параллельную обработку информации на этапе Transform и снижать нагрузку на исходные хранилища. Применение данного фреймворка осуществляется адаптивно на основе сопоставления доступных методов обработки и эквивалентности их поведения, что позволяет сохранить корректность обработки данных и повысить производительность вычислений.

Автоматизация ключевых этапов миграции достигается за счет использования структурированных конфи-

гурационных файлов, на основе которых управляющая подсистема метода осуществляет анализ характеристик источников данных и выбирает оптимальную стратегию извлечения и обработки информации. Такой подход снижает зависимость процесса миграции от ручной настройки, минимизирует вероятность ошибок и повышает масштабируемость метода при работе с большим количеством источников данных.

Таким образом, предлагаемый адаптивный гибридный метод может рассматриваться как **перспективный подход** к решению задач миграции больших данных, поскольку он ориентирован на автоматизированный выбор стратегии извлечения данных, рациональное использование вычислительных ресурсов и интеграцию специализированных инструментов обработки информации.

Полученные результаты носят **предварительный характер** и указывают на целесообразность дальнейших исследований, направленных на количественную оценку эффективности предлагаемого метода в различных сценариях миграции данных.

ЛИТЕРАТУРА

1. Garcia-Molina H., Ullman J.D., Widom J. Database Systems: The Complete Book. 2nd ed. // Pearson, 2008. 1248 p. URL: https://djvu.online/file/WUTyQnUI8qhKb?ysclid=mkasqdj11p512840887 (дата обращения: 14.01.2025)
2. Connolly T.M., Begg C.E. Database Systems: A Practical Approach to Design, Implementation, and Management. 6th ed. // Pearson, 2014. 1440 p. URL: https://www.researchgate.net/publication/228831514_Database_systems_A_practical_approach_to_design_implementation_and_management (дата обращения: 14.01.2025)
3. Мартишин С.А., Симонов В.Л., Храпченко М.В. Проектирование и реализация баз данных в СУБД MySQL с использованием MySQL Workbench: Методы и средства проектирования информационных систем и технологий. Инструментальные средства информационных систем. Учебное пособие // ИНФРА-М, 2022. 159 с. URL: https://www.litres.ru/book/vladimir-lvovich-sim/proektirovanie-i-realizaciya-baz-dannyh-v-subd-mysql-71724910/?utm_source=yandex&utm_medium=cpc&utm_campaign=mixed_web_dsa_drr_feed+607619718+%7Cspecial%7C47441382&utm_content=13971846835&utm_term---autotargetingutm_term=---autotargeting&etext=&yclid=11982121547679662079&lfrom_processed=607619718 (дата обращения: 14.01.2025)
4. Silberschatz A., Korth H.F., Sudarshan S. Database System Concepts. 7th ed. // McGraw-Hill, 2019. 1344 p. URL: https://www.db-book.com/ (дата обращения: 14.01.2025)
5. Лаврентчук Е.А. Технологии анализа больших данных: учебное пособие // СПб.: Университет ИТМО, 2019. 96 с. URL: https://books.ifmo.ru/book/1179/tehnologii_analiza_bolshih_dannyh.htm (дата обращения: 14.01.2025)
6. Zikopoulos P., deRoos D., Bienko C. et al. Big Data Beyond the Hype: A Guide to Conversations for Today's Data Center // McGraw-Hill, 2015. 256 p. URL https://archive.org/details/bigdatabeyondhyp0000ziko_m1u7 (дата обращения: 14.01.2025)
7. Машнин Т. Технология хранения и обработки больших данных Hadoop // ЛитРес 2021 136 с. URL: https://www.litres.ru/book/timur-mashnin-301845/tehnologiya-hraneniya-i-obrabotki-bolshih-dannyh-hado-65077172/?utm_source=yandex&utm_medium=cpc&utm_campaign=mixed_web_dsa_drr_feed+607619718+%7Cspecial%7C47441382&utm_content=13971846835&utm_term---autotargetingutm_term=---autotargeting&etext=&yclid=9381017690509148159&lfrom_processed=607619718 (дата обращения: 14.01.2025)
8. Coronel C., Morris S. Database Systems: Design, Implementation, & Management. 14th ed. // Cengage Learning, 2022. 848 p. URL: https://www.cengage.com/c/database-systems-design-implementation-management-14e-coronel/9780357673034/ (дата обращения: 14.01.2025)
9. Гарсиа-Молина Г., Ульман Д., Уидом Д. Системы баз данных. Полный курс = Database Systems: The Complete Book // М.: Вильямс, 2014. 1088 с. URL: https://djvu.online/file/WUTyQnUI8qhKb?ysclid=mkat8env1l686620576 (дата обращения: 14.01.2025)
10. Han J., Kamber M., Pei J. Data Mining: Concepts and Techniques. 3rd ed. // Morgan Kaufmann, 2011. 744 p. URL: https://www.oreilly.com/library/view/data-mining-concepts/9780123814791/ (дата обращения: 14.01.2025)
11. НОУ ИНТУИТ. Основы баз данных : онлайн-курс // intuit.ru . URL: https://intuit.ru/studies/courses/599/455/lecture/10159 (дата обращения: 14.01.2025)
12. Stonebraker M., Hellerstein J. M. What Goes Around Comes Around // Readings in Database Systems. 4th ed. // MIT Press, 2005. P. 2–41. URL: https://people.csail.mit.edu/tdanford/6830papers/stonebraker-what-goes-around.pdf (дата обращения: 14.01.2025)
13. Орлов С.А. Администрирование и оптимизация баз данных. Учебник и практикум для вузов // М.: Юрайт, 2020. 323 с. URL: https://urait.ru/bcode/449745 (дата обращения: 14.01.2025)
14. White T. Hadoop: The Definitive Guide. 4th ed. // O'Reilly Media, 2015. 756 p. URL: https://www.oreilly.com/library/view/hadoop-the-definitive/9781491901687/ (дата обращения: 14.01.2025)
15. Мейер Д. Теория реляционных баз данных = The Theory of Relational Databases // М.: Мир, 1987. 608 с. URL: https://search.rsl.ru/ru/record/01001116473 (дата обращения: 14.01.2025)
16. Kimball R., Ross M. The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling. 3rd ed. // Wiley, 2013. 600 p. URL: https://www.wiley.com/en-us/The+Data+Warehouse+Toolkit%2C+3rd+Edition-p-9781118530801 (дата обращения: 23.01.2026)
17. Inmon W.H. Building the Data Warehouse. 4th ed. // Wiley, 2005. 576 p. URL: https://www.wiley.com/en-us/Building+the+Data+Warehouse%2C+4th+Edition-p-9780764599446(дата обращения: 23.01.2026)