МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«Тольяттинский государственный университет»

Институт математики, физики и информационных технологий
(наименование института полностью)

Кафедра «Прикладная математика и информатика»
(наименование)

09.04.03 Прикладная информатика
(код и наименование направления подготовки, специальности)

Информационные системы и технологии корпоративного управления
(направленность (профиль))

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ)

на тему «Повышение эффективности технологии тестирования миграции данных корпоративных информационных систем»

Студент К.М. Зайцев _____
(И.О. Фамилия) (личная подпись)

Научный руководитель к.п.н., Е.А. Ерофеева
(ученая степень, звание, И.О. Фамилия)

Тольятти 2020

Оглавление

Введение

Предприятия используют хранилища данных для накопления данных из нескольких источников, для анализа данных и исследовательской работы и еще для бесконечного числа задач. Поскольку организационные решения часто принимаются на основе данных, хранящихся в хранилище данных, все его компоненты должны быть тщательно проверены. В этой работе подробно рассмотрен подход к тестированию хранилища данных, а также произведена разработка и оценка алгоритма автоматизированного тестирования процесса Extract-Transform-Load (ETL), который является стандартным решением для энтерпрайз продуктов.

Актуальность темы исследования обусловлена необходимостью создания решения для качественной, быстрой и эффективной проверки пост-мигрированных данных в условиях обширных баз данных с разветвленной иерархической зависимостью.

Объектом исследования является процесс миграции данных из устаревшей базы данных в целевую с использованием промежуточных хранилищ.

Предметом исследования является моделирование алгоритма тестирования миграции баз данных.

Целью данной работы является разработка и внедрение модели эффективной проверки мигрированных данных.

Цель данной работы определила необходимость решения следующих **задач:**

1) Изучить исследования специалистов, связанные с миграцией данных.
2) Произвести обзор и анализ существующих моделей\методик.
3) Предложить алгоритм тестирования миграции данных
4) Разработать алгоритм тестирования миграции данных, которая обеспечит высокую производительность и качеств ПО.

5) Произвести оценку эффективности разработанной модели и доказать гипотезу.

Гипотеза исследования состоит в предположении, что применение разработанного в рамках данного диссертационного исследования метода тестирования миграции данных, обеспечит высокое качество, скорость, эффективность и надежность при работе с большим объемом данных.

Основные **методы исследования**: теоретический анализ, систематизация, объектно-ориентированный подход к моделированию систем управления, статистический анализ.

Новизна исследования заключается в предоставлении нового метода тестирования миграции данных.

Практическая значимость данного исследования состоит в возможности практического применения предлагаемого метода для тестирования миграции данных.

Методологическую базу исследования составляют работы, опубликованные зарубежными и отечественными авторами, П. Джонсон, Фл. Маттес, К, Шульц, К. Халлер, В. Ратика, Л. Аркокиам и др.

Основные этапы исследования: исследование проводилось с 2018 по 2020 годы в несколько этапов:

На первом этапе формулировалась тема исследования и выполнялся сбор и анализ информации по теме исследования, также проводилась постановка цели, задач, предмета, объекта и гипотезы исследования.

На втором этапе осуществлялся анализ методов тестирования, создавался алгоритм получения данных для проверки и соответственно взаимодействия в рамках разрабатываемого метода, публиковались результаты исследования.

На третьем этапе был предложен метод тестирования мигрированных данных и проведен анализ эффективности данного метода, были сформулированы выводы о полученных результатах исследования.

На защиту выносятся:

1. Алгоритм
2. Анализ эффективности разработанной модели.

Диссертация состоит из введения, трех глав, заключения и списка используемой литературы.

В первой главе рассматриваются современное состояние сферы миграции баз данных, её особенности и методологии. Выяснено, что данный вопрос требует проработки в аспекте кастомизации под конкретные задачи.

Во второй главе рассмотрены существующие модели тестирования миграции, их анализ и сделан вывод об их несовершенстве в контексте решения задач проекта

В третьей главе подробно рассмотрен алгоритм валидации данных миграции и сделана оценка его эффективности. Это позволило глубже понять процесс миграции данных.

- Новый алгоритм оказался достаточно гибким, но в тоже время доработанным под конкретные требования рабочей системы.
- Задействованный подход для проверки миграции показал свою эффективность на рабочей системе.
- Анализ результатов работы с предложенным алгоритмом показал его ,в некоторых случаях, многократную эффективность перед предшествующим подходом

В заключении приводятся результаты исследования.

Работа изложена на 71 странице, включает 21 рисунок, 14 таблиц, 47 источников используемой литературы.

Глава 1 Теоретические основы выполнения процесса миграции, методы

1.1Анализ процесса миграции

Наиболее распространенной причиной миграции является необходимость перемещения данных в новую систему для увеличения масштаба и обеспечения роста объема. Однако, другие обстоятельства могут также побудить организации выбрать проект переноса данных. Причины включают в себя:

- замену устаревших систем;
- уменьшение объема хранилища за счет перехода на более ресурсотребовательную систему;
- конкурентоспособность путем внедрения передовых технологий;
- перенос данных в облако, что исключает затраты на локальную ИТ-инфраструктуру.

Исходя из вышесказанного, миграция данных может быть разделена на четыре типа:

- Миграция базы данных

Данный тип миграции включает перемещение данных между двумя ядрами базы данных. Или же можно сказать, что это обновление структуры базы данных от одной версии до другой, обычно более новой.

- Миграция приложений

Данный тип миграции происходит, когда организация переключается с одной платформы или приложения поставщика на платформу или приложение другого поставщика.

- Миграция хранилища

Данный процесс связан с перемещением данных из одной системы хранения в другую, например, на жесткий диск или в облако.

- Облачная миграция

В облачной миграции полные или частичные информационные активы, приложения или службы организации развертываются в облаке.

Методы переноса данных

Метод, который предлагает сочетание надежности, эффективности миграции и минимального воздействия на пользователей и бизнес-процессы, является тем, который может наилучшим образом удовлетворить потребности организации.

Рассмотрим некоторые из них:

- **ETL**

Процессы извлечения, преобразования и загрузки данных позволяют выполнять множество операций в проектах в области информационных технологий. Понимание концепций и практики ETL имеет важное значение для всех специалистов в области данных и технологий.

Извлечение, преобразование, загрузка данных (ETL) - это процесс копирования данных из одного или нескольких источников в целевую систему, которая обычно предназначена для представления данных в отличие от источника (ов). Процессы ETL используются для хранилищ данных, интеграции данных и проектов миграции данных.

Извлечение данных включает извлечение данных из однородных или разнородных источников.

Методы преобразования данных часто очищают, агрегируют, дедуплицируют и другими способами преобразуют данные в правильно определенные форматы хранения для запроса и анализа.

Загрузка данных представляет собой вставку данных в конечное целевое хранилище, такое как хранилище оперативных данных, витрина данных или хранилище данных.

Процессы ETL обычно объединяют данные из нескольких приложений (систем и источников), возможно, разработанные и поддерживаемые различными поставщиками или размещенные на отдельном компьютерном оборудовании. Отдельные системы, содержащие исходные данные, часто управляются и управляются разными командами. Перед проектированием и разработкой ETL-процессов всегда требуется сопоставление данных от источника к цели. Логические карты данных (обычно готовые в виде электронных таблиц) описывают отношения между начальными точками и конечными точками системы ETL. На рисунке 1.1 представлен ETL процесс.

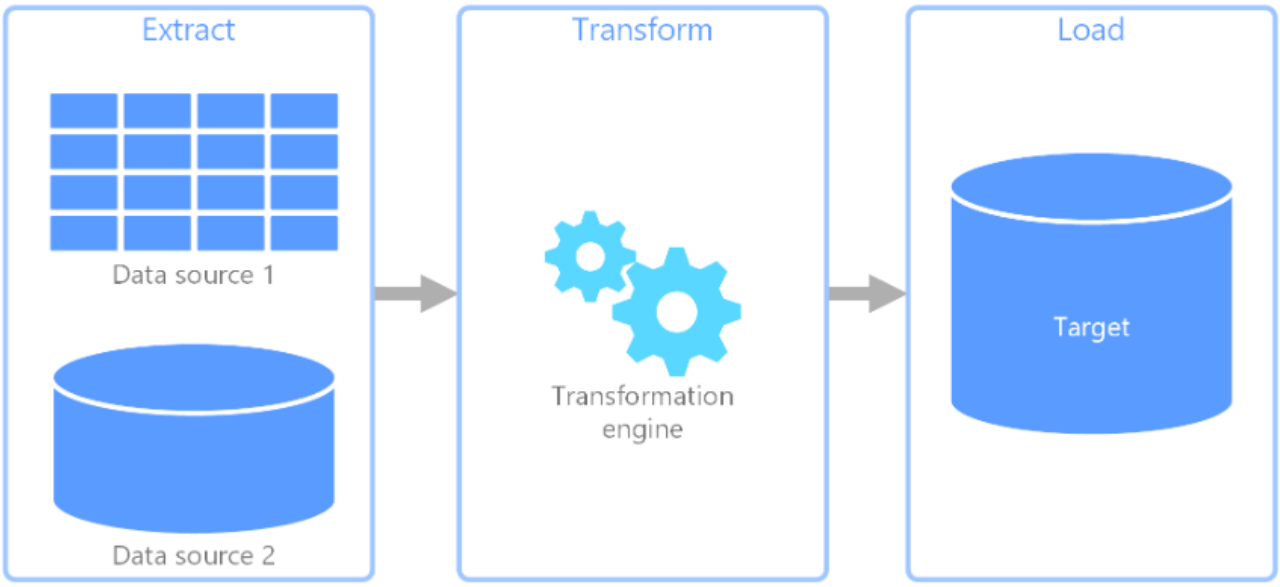

Рисунок 1 - ETL процесс

Шаги для процессов ETL:

Процессы ETL состоят из трех отдельных, но важных функций, которые часто объединяются в единый инструмент программирования, который помогает в подготовке данных и в управлении базами данных.

После обнаружения и записи исходных данных тщательно разработанные процессы ETL извлекают данные из исходных систем,

внедряют задачи качества данных / стандарты согласованности, согласовывают данные, чтобы отдельные источники могли использоваться вместе, и, наконец, доставляют данные в готовом для представления формате, чтобы разработчики приложений могут создавать приложения, а конечные пользователи могут принимать решения.

Обнаружение исходных данных

Некоторые или все исходные системы могли быть идентифицированы во время сеансов моделирования данных проекта, но это не может считаться само собой разумеющимся. Обычно на этапе моделирования данных проекта определяются только ключевые исходные системы. Команда ETL должна углубиться в требования к данным, чтобы определить каждую исходную систему, таблицу и атрибут, требуемые в процессах ETL. Определение необходимых источников данных или систем записи для каждого элемента / таблицы - это задача, которую необходимо решить, прежде чем переходить к извлечениям данных.

Извлечение исходных данных

Фазы извлечения данных представляют собой извлечения из исходных систем, чтобы сделать их доступными для дальнейшей обработки.

Типы извлечения данных:

- полное извлечение - полное извлечение всех данных необходимо каждый раз, когда требуются измененные данные из этих отдельных источников. Полная выписка требует сохранения копии последней выписки в том же формате, чтобы определить изменения, когда станет доступной более поздняя выписка. Команда ETL отвечает за регистрацию изменений содержимого данных во время дополнительных нагрузок после начальной загрузки;

- обновление извлечения - когда исходные системы могут предоставлять уведомления об изменении определенных данных и дополнительно идентифицировать каждое изменение, это самый простой способ извлечения данных;
- инкрементное извлечение. Некоторые исходные системы не могут предоставить уведомление о том, что произошло обновление, но они могут определить, какие записи были изменены, и предоставить выдержку только из этих записей. Во время последующих шагов ETL система должна идентифицировать изменения и распространять их вниз.

Преобразование данных

Данные, извлеченные из источников, часто являются результатом транзакций и поэтому не могут использоваться в целевых базах данных в такой форме. Большая часть таких исходных данных должна быть очищена, дедуплицирована, агрегирована или иным образом преобразована. Это ключевой шаг, когда процесс ETL добавляет ценность и изменяет данные так, чтобы можно было генерировать проницательные отчеты о приложениях.

Загрузка хранилищ данных

Для загрузки данных в хранилища могут быть использованы два следующих метода:

- полная загрузка - полная загрузка данных к целям, которая происходит при первой загрузке источника данных в хранилище;
- дополнительные нагрузки - загрузка данных, которые изменились («дельта-нагрузки») между источником и целью через регулярные промежутки времени. Самая последняя дата извлечения сохраняется, так что загружаются только записи, добавленные после этой даты.

Репликация базы данных

Проще говоря, репликация данных берет данные из исходных баз данных - Oracle, MySQL, Microsoft SQL Server, PostgreSQL, MongoDB и т. д. - и копирует их в облачное хранилище данных. Это может быть одноразовая операция или постоянный процесс, когда ваши данные обновляются. Поскольку ваше хранилище данных является важным механизмом, с помощью которого вы можете получить доступ и анализировать ваши данные, необходима правильная репликация данных, чтобы не потерять, не продублировать или иным образом не испортить ценную информацию.

Существует три распространенных метода репликации данных:

- полный дамп и загрузка. Начиная с самого простого метода, полная репликация дампов и загрузок начинается с определения интервала репликации (может быть два, четыре, шесть часов - в зависимости от ваших потребностей). Затем в каждом интервале запрашиваемые вами таблицы запрашиваются и делается снимок. Новый снимок (дамп) заменяет (загружает) предыдущий снимок в вашем хранилище данных;
- инкремент: с помощью метода инкремента вы определяете индикатор обновления для каждой из ваших таблиц - обычно это столбец, который отслеживает время последнего обновления (обычно что-то вроде «updated_at»). Каждый раз, когда строка в вашей базе данных вставляется или обновляется, индикатор обновления обновляется. Ваши таблицы данных регулярно запрашиваются для сбора изменений. Изменения копируются в ваше хранилище данных и объединяются;
- репликация журналов или сбор данных изменений (CDC). Самый быстрый метод - более или менее золотой стандарт репликации данных - это репликация журналов или CDC. Это включает в себя запрос внутреннего журнала изменений вашей базы данных каждые несколько секунд, копирование изменений в хранилище данных и частое их включение. Все

изменения в указанных вами таблицах и объектах загружаются по умолчанию, включая удаление, поэтому ничего не пропадает.

1.2Методы миграции корпоративных систем

Миграция данных - это передача данных из одного места, носителя или аппаратно-программного комплекса в другое. Миграционные усилия часто вызваны необходимостью обновления технической инфраструктуры или изменениями требований бизнеса агентства.

Обзор лучших практик выявил два принципа, присущих успешной миграции данных:

1. Выполнение миграции данных как проекта, посвященного уникальной цели создания нового (целевого) хранилища данных.

2. Выполнение переноса данных в четыре основных этапа:

- планирование переноса данных;
- анализ и проектирование переноса данных;
- внедрение переноса данных и закрытие переноса данных

Проект миграции данных фокусируется на перемещении данных между унаследованной системой данных и целевой системой, включая все необходимые процедуры для передачи и проверки данных на протяжении всего процесса. Перед перемещением данных часто их необходимо изменить и / или преобразовать. Этот процесс называется преобразованием данных. Планирование и выполнение преобразования данных требует разработки правил и процедур преобразования для осуществления необходимых изменений. Например, если унаследованная система хранит информацию о дате в текстовом формате, но целевая система требует, чтобы эта информация была сохранена как формат даты, то преобразование унаследованных данных необходимо до переноса данных[40].

Обычно промежуточная область используется в качестве промежуточного хранилища данных для облегчения тестирования и

проверки этих модификаций / преобразований. Кроме того, промежуточная область может служить областью хранения для проектов интеграции, которые извлекают данные из нескольких исходных систем. Процесс миграции данных высокого уровня отображен на рисунке 1.2.

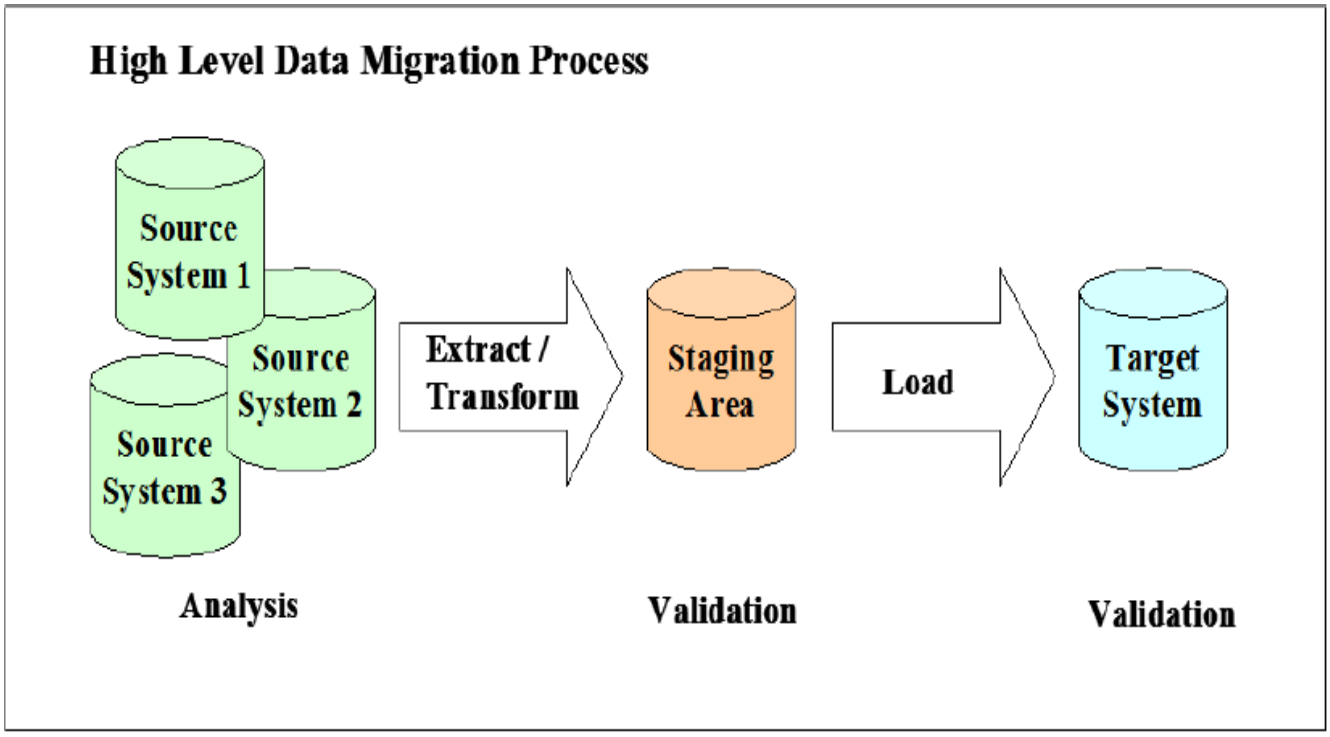

Рисунок 2 - Процесс миграции данных высокого уровня

Выполняется миграция данных в четыре основных этапа:

- планирование миграции данных;
- анализ миграции данных и дизайн;
- реализация миграции данных;
- закрытие переноса данных.

Этап планирования миграции данных.

Этап планирования миграции данных описывает отдельные задачи для:

- планирования проекта миграции данных;

- определения требования к миграции данных;
- оценки текущей среды;
- Разработки плана миграции данных;
- определения и назначения командных ролей и обязанностей.

Чтобы обеспечить успешное выполнение проекта переноса данных и более крупного проекта разработки, рекомендуется выполнять перенос данных в качестве независимого проекта. Тщательное планирование является основой для постоянного успеха в любом процессе, и миграция данных не является исключением. Кроме того, успешная миграция данных требует смягчения проблем и рисков для бизнеса / организации.

План переноса данных содержит подробную информацию, которая должна быть включена для каждого шага в плане. Другие результаты, такие как риски и / или критические факторы успеха, могут быть просто задокументированы в плане. Все этапы последующих этапов миграции включены в план. Также включен способ выполнения шагов в отношении правил, параметров и процедур и так далее.

Кроме того, в программе развития описаны общие результаты проекта, которых должны придерживаться все проекты, такие как:

- план качества;
- план управления изменениями;
- план коммуникаций.

Анализ и дизайн

Тщательный план переноса данных должен включать либо требования к действиям по анализу и проектированию, либо обоснование того, что они не включены в конкретный вид деятельности.

На основе устаревшей (или исходной) архитектуры данных, завершенного анализа и плана миграции данных ТМТ должна спроектировать архитектуру данных миграции. Эта архитектура состоит из:

- устаревшей архитектуры «как есть» (проект);
- архитектуры данных промежуточной области «быть»;
- целевой архитектуры данных «быть»;
- корреляции («отображения»), показывающей межархитектурные отношения и инструкции по превращению данных из одной архитектуры данных в действительные данные в следующей (источник в промежуточную или целевую, промежуточную в целевую)

Структуры данных и процедуры переноса данных должны удовлетворять требованиям переноса данных. Результирующие артефакты состоят из полностью приписанных логических моделей данных, словарей данных, функциональных (или процессных) моделей, отображений между структурами данных «как есть» и «быть» с соответствующими бизнес-правилами и логикой преобразования, а также любым другим определенным применимым артефактом в рамках интегрированной архитектуры федеральной помощи студентам. Затем эти артефакты должны быть проверены через команду EDS и заинтересованные стороны в сфере бизнеса.

Обзор реализации

Этапы внедрения и проверки логически взаимозависимы, так же, как этапы планирования, анализа и проектирования. Хотя не все действия могут выполняться для каждой миграции, выполняемые шаги должны, как правило, соответствовать последовательности, показанной ниже. Есть одно исключение: как обсуждалось в нескольких статьях о передовой практике, в процессе планирования должно быть принято решение относительно наиболее эффективного и действенного времени для очистки данных и преобразования данных[4].

Если миграция данных охватывает несколько исходных систем, повторите шаги по извлечению данных и загрузке их в промежуточную область, пока не будут загружены все исходные данные. Промежуточная

область поддерживает интеграцию всех данных. На рисунке 1.4 отображен план выполнения миграции данных.

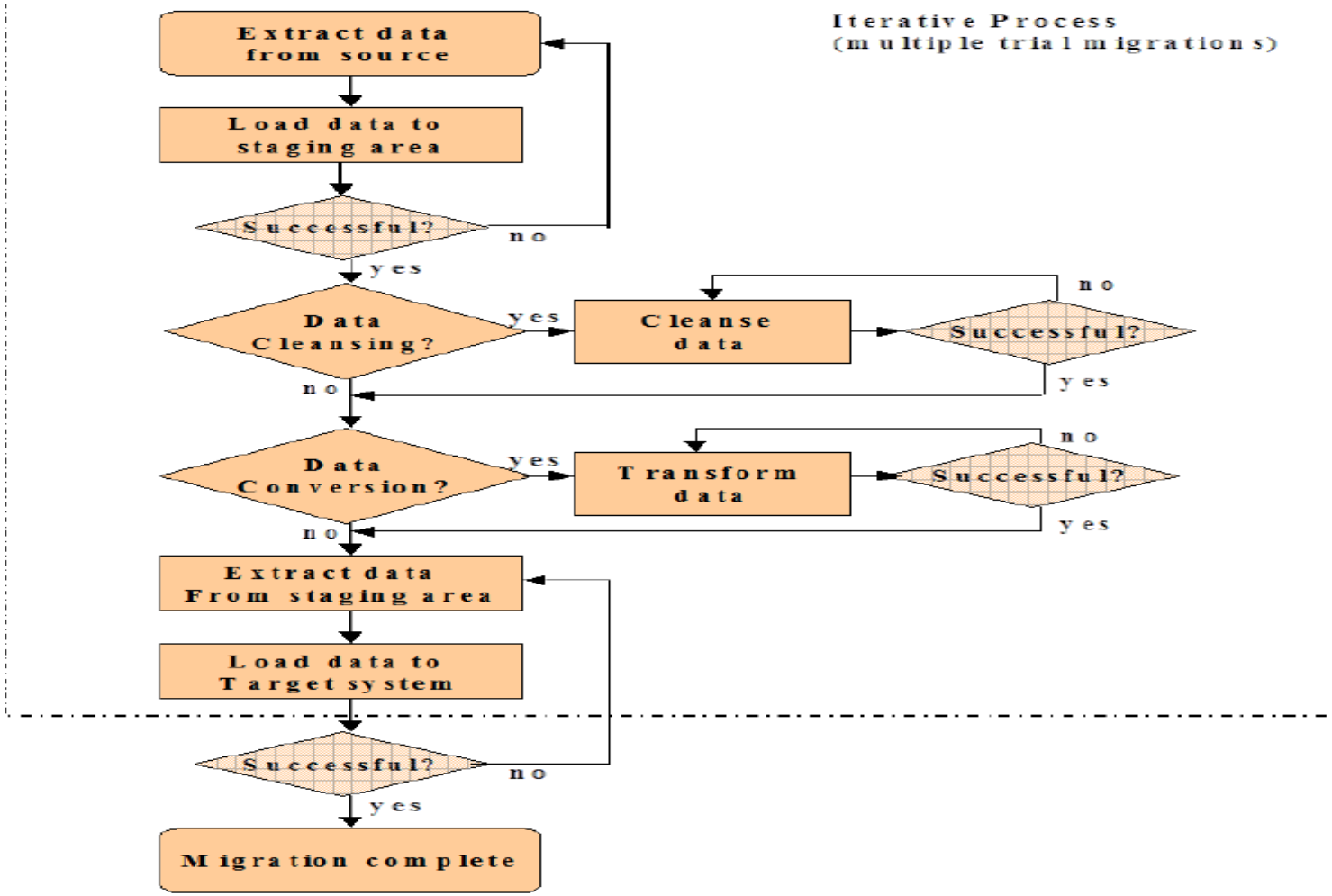

Рисунок 3 - План выполнения миграции данных

Инструменты миграции

Informix от IBM - еще один отличный инструментальный репозиторий, который можно использовать для перемещения данных из одной базы данных IBM в другую базу данных IBM. Он в первую очередь поддерживает однородную миграцию данных.

Informix® предоставляет инструменты, утилиты и операторы SQL, которые можно использовать для перемещения данных из одной базы данных IBM® Informix в другую или из одной операционной системы в другую.

Возможно, пользователь захочет использовать инструмент переноса данных, когда у вас разные размеры страниц или кодовые страницы. Например, UNIX или Linux и Windows хранят данные на страницах разных размеров.

Если миграция включает в себя миграцию между различными операционными системами, пользователь должен экспортировать данные и информацию о их схеме с одного сервера базы данных и импортировать экспортированные данные на другой сервер базы данных.

Обычно, если выполняется миграция в одной операционной системе, не нужно загружать и выгружать данные.

Для перемещения данных могут быть использованы следующие инструменты:

- утилиты dbexport и dbimport;
- утилита dbload;
- утилиты onunload и onload;
- операторы UNLOAD и LOAD;
- the High-Performance Loader (HPL);
- не журналируемые необработанные таблицы.

При импорте данных из источников, отличных от Informix, могут быть использованы следующие инструменты:

- пример
- утилиты dbimport и dbload;
- высокопроизводительный погрузчик (HPL);
- продукты IBM Informix Enterprise Gateway;
- внешние таблицы, которые вы создаете с помощью оператора CREATE EXTERNAL TABLE.

Лучший способ перемещения данных зависит от операционной системы и от того, желает ли пользователь переместить всю базу данных, выбранные таблицы или выбранные столбцы из таблицы. В следующей таблице1 приведены характеристики методов загрузки данных, а также преимущества и недостатки каждого метода. В таблице 1 также показаны серверы баз данных, на которых вы можете использовать инструменты.

Таблица 1 – Методы загрузки данных

Инструмент	Описание	Преимущества	Недостатки
утилиты dbexport и dbimport	Импортирует или экспортирует базу данных в текстовый файл, который хранится на диске или ленте	Можно изменить схему базы данных и изменить формат данных Может перемещать данные между операционными системами Необязательное ведение журнала Может импортировать данные из источников не-Informix	Быстрее, чем утилита dbload, но медленнее, чем утилита onload Перемещает всю базу данных
утилита dbload	Переносит данные из одного или нескольких текстовых файлов в одну или несколько существующих	Может изменить схему базы данных Может перемещать данные между операционными	Более низкая производительность, чем у утилит dbexport, dbimport и onload

Инструмент	Описание	Преимущества	Недостатки
	таблиц	системами Необязательное ведение журнала Умеренно простой в использовании Может импортировать данные из источников не-Informix	
утилиты onunload и onload	Выгружает данные из базы данных в файл на ленте или диске; загружает данные, которые были созданы с помощью команды onunload, на сервер базы данных	Быстрая производительность Дополнительная регистрация	Перемещает данные только между серверами баз данных одной и той же версии в одной операционной системе. Невозможно изменить схему базы данных Ведение журнала должно быть отключено Сложно использовать
Операторы UNLOAD и LOAD	Выгружает и загружает указанные строки	Может изменить схему базы данных Может перемещать данные между операционными системами Легко использовать Необязательное ведение журнала	Принимает только указанные форматы данных
HPL	Загружает данные из любого файла ASCII или COBOL, который соответствует определенным требованиям	Для очень больших баз данных имеет преимущество в производительности по сравнению с другими утилитами переноса данных	Требует значительного времени на подготовку

Инструмент	Описание	Преимущества	Недостатки
	формата	IBM Informix, поскольку оно выполняет параллельные преобразования ввода-вывода и набора кодов Может изменить схему базы данных Может перемещать данные между операционными системами Может импортировать данные из источников не-Informix	
Не журналируемые необработанные таблицы	Загружает определенные виды больших таблиц.	Может быстро загружать очень большие таблицы хранилищ данных.	Не поддерживает первичные ограничения, уникальные ограничения и откат Требуется SQL Не рекомендуется для использования в транзакции

AWS

Набор сервисов AWS охватывает множество методов, которые помогают более эффективно перемещать данные. Их можно разделить на две категории: передача данных по сети и гибридное облачное хранилище, и миграция данных без использования сети на Amazon S3.

Передача данных по сети и гибридное облачное хранилищ

Такие методы позволяют без лишних усилий создавать подключение к VPC, передавать данные в AWS или использовать инстансы S3 для гибридного облачного хранилища с помощью существующих локальных приложений. Эти сервисы помогают однократно переносить большие пакеты данных, а также интегрировать существующие процессы, такие как резервное копирование и восстановление или постоянная потоковая передача данных, непосредственно с облачным хранилищем.

AWS DataSync

AWS DataSync – это сервис передачи данных, который упрощает автоматизацию переноса данных между локальными хранилищами и Amazon S3 или Amazon Elastic File System (Amazon EFS). DataSync автоматически обрабатывает многие из задач при передаче данных, которые могут замедлить перенос или стать излишней нагрузкой для ИТ-отдела (включая запуск собственных инстансов, обработку шифрования, управление скриптами, оптимизацию сети и проверку целостности данных). AWS DataSync позволяет переносить данные в облако AWS до 10 раз быстрее, чем при использовании инструментов с открытым исходным кодом. DataSync можно использовать для копирования данных через подключение AWS Direct Connect или по интернет-ссылкам на AWS при однократном переносе данных, в повторяющихся процессах обработки данных и для автоматической репликации в целях защиты и восстановления данных. На рисунке 4 представлена схема AWS DataSync.

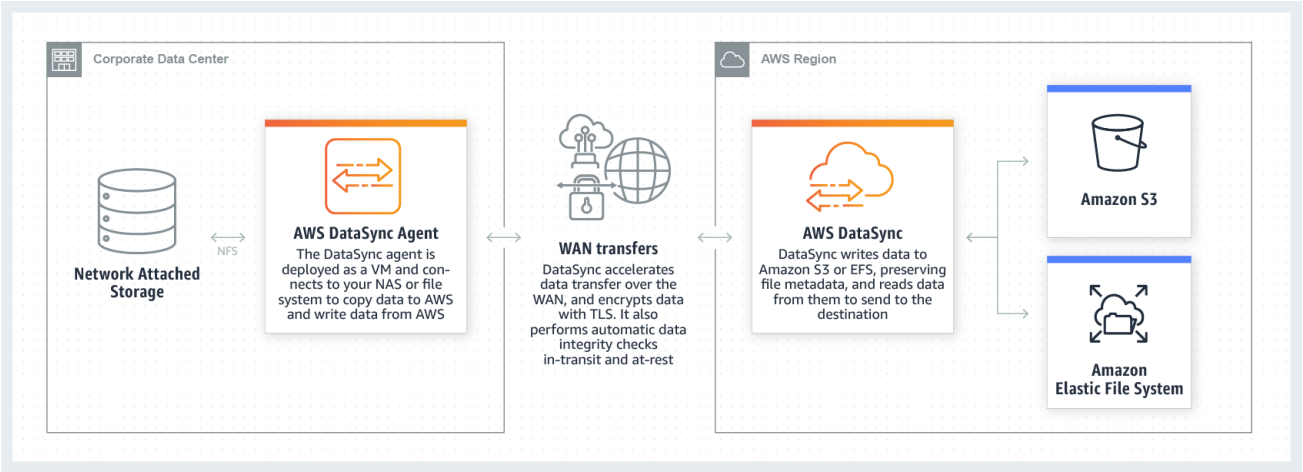

Рисунок 4 - AWS datasync

Перенос данных в Amazon S3 без использования сети

В этих сервисах для передачи данных без использования сети применяются транспортабельные защищенные устройства, которые идеально подходят для переноса крупных архивов, можно сказать, океанов данных или для ситуаций, когда пропускная способность не позволит нужному объему данных пройти по сети за требуемый промежуток времени.

AWS Snowball

Snowball – это решение для перемещения данных, объем которых измеряется в петабайтах, в облако AWS и из него с использованием устройств, разработанных для безопасного переноса больших объемов данных. Использование Snowball решает целый ряд проблем, связанных с передачей больших объемов данных, в том числе проблему высокой стоимости передачи данных по сети, длительной передачи, а также проблемы безопасности. Сегодня клиенты используют Snowball для миграции аналитических данных, геномных данных, библиотек видео, репозиториев изображений, резервных копий, для архивирования информации при остановке центров обработки данных, для замены магнитных лент или в

проектах миграции приложений. На рисунке 5 представлена схема AWS Snowball

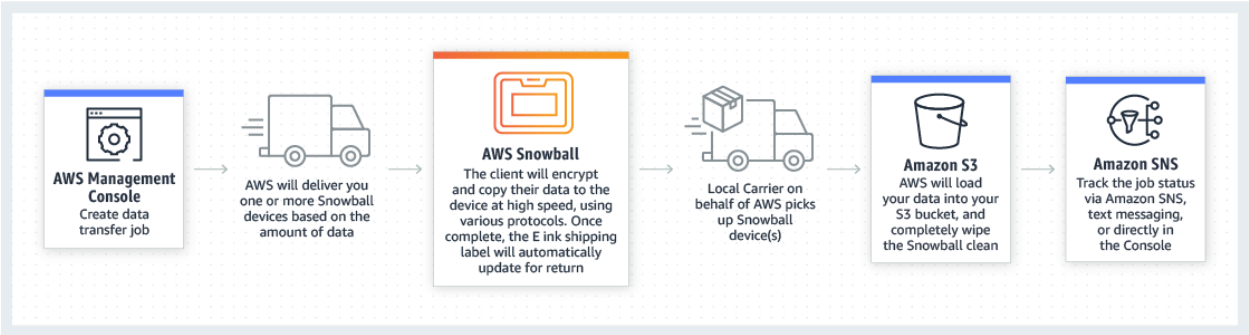

Рисунок 5 – AWS Snowball

1.3 Источники и целевое хранилище данных.

Источники в системе хранилищ данных хранят данные, принадлежащие одной или нескольким организациям для ежедневных транзакций или бизнес-целей. С другой стороны, целевое хранилище данных хранит большие объемы данных для долгосрочного анализа и майнинга. Источники и целевые хранилища данных могут быть спроектированы и реализованы с использованием различных технологий, включая модели данных и системы управления данными.

Модель данных описывает бизнес-термины и их отношения, часто в графической форме .

Следующие модели данных обычно используются для разработки исходной и целевой схем:

Модель реляционных данных:

Такая модель организует данные в виде наборов двумерных таблиц, в которых все данные представлены в виде кортежей (неизменяемые списки). Таблицы представляют собой отношения строк и столбцов с уникальным ключом для каждой строки. Диаграммы отношений сущностей обычно используются для разработки реляционных моделей данных.

Нереляционная модель данных:

такая модель организует данные без структурированного механизма связи данных различных сегментов. Эти модели используют средства, отличные от таблиц, используемых в реляционных моделях. Вместо этого используются разные структуры данных, такие как графики или документы. Эти модели обычно используются для организации чрезвычайно больших наборов данных, используемых для интеллектуального анализа данных, потому что в отличие от реляционных моделей, нереляционные модели не имеют сложных зависимостей между их сегментами.

Модель размерных данных: такая модель использует структуры, оптимизированные для запросов конечных пользователей и инструментов хранилища данных. Эти структуры включают:

- таблицы фактов, в которых хранятся измерения бизнес-процесса
- таблицы измерений, которые содержат описательные атрибуты.

Информация сгруппирована в соответствующие таблицы, называемые измерениями, что упрощает ее использование и интерпретацию. В отличие от реляционных моделей, которые минимизируют избыточность данных и улучшают обработку транзакций, многомерная модель предназначена для поддержки и оптимизации запросов. Размерные модели более масштабируемы, чем реляционные модели, поскольку они устраняют сложные зависимости, существующие между реляционными таблицами. Размерная модель может быть представлена схемами star или Snowflake и часто используется при проектировании хранилищ данных.

Схемы следующие:

- Star: эта схема имеет таблицу фактов в центре. Таблица содержит ключи к таблицам измерений. Каждое измерение включает в себя набор атрибутов и представляется через одну таблицу измерений.

- Snowflake: в отличие от схемы «star», схема «Snowflake» имеет нормализованные измерения, которые разбиты на несколько таблиц

измерений. Схема «star» является частным случаем схемы «Snowflake» с одноуровневой иерархией. Источники и хранилища данных используют различные системы управления данными для сбора и организации своих данных. Ниже приведен список систем управления данными, обычно используемых для реализации исходных и целевых хранилищ данных.

Система управления реляционными базами данных (RDBMS): RDBMS основана на реляционной модели данных, которая позволяет связывать информацию из разных таблиц. Таблица должна содержать то, что называется ключом или индексом, и другие таблицы могут ссылаться на этот ключ для создания связи между своими данными . СУБД обычно используют язык структурированных запросов (SQL) и подходят для управления структурированными данными. СУБД способны обрабатывать запросы и транзакции, которые обеспечивают эффективную, правильную и надежную обработку данных даже в случае сбоев.

Нереляционная система управления базами данных:

Нереляционная СУБД основана на нереляционной модели данных. Наиболее популярной нереляционной базой данных является не только SQL (NoSQL), которая имеет много форм, таких как основанные на документах, основанные на графиках и основанные на объектах.

Нереляционная СУБД обычно используется для хранения и управления большими объемами неструктурированных данных.

Система управления большими данными. Системы управления большими данными должны хранить и обрабатывать большие объемы как структурированных, так и неструктурированных данных. Они включают в себя технологии, которые подходят для управления нетранзакционными формами данных. Большая система управления данными плавно включает в себя системы управления реляционными и нереляционными базами данных.

Устройство хранилища данных (DWA):

Впервые DWA был предложен Хиншоу в качестве архитектуры, подходящей для хранилища данных. DWA предназначены для высокоскоростного анализа

больших объемов данных. DWA объединяет базу данных, сервер, хранилище и аналитику в простую в управлении систему.

Cloud DataWarehouse Appliance: Cloud DWA - это устройство хранилища данных, которое работает на платформе облачных вычислений. Это устройство обладает всеми функциями, предоставляемыми облачными вычислениями, такими как сбор и организация всех данных в сети, получение бесконечных вычислительных ресурсов по требованию и мультиплексирование рабочих нагрузок из разных организаций. В таблице 2.1 представлены некоторые из доступных продуктов, используемых для управления данными в источниках и целевых хранилищах данных. Проектирование и реализация баз данных в источниках обычно основаны на организационных требованиях, тогда как базы данных хранилищ данных основаны на требованиях анализаторов данных и исследователей.

Целевое хранилище данных для данных о работоспособности, возможно, должно соответствовать стандартной модели данных, разработанной для электронных записей о работоспособности, такой как OMOP CDM.

Таблица2 - Доступные продукты для управления данными в источниках и хранилищах данных

Категория	Пример
DBMS	Relational: MySQL, MS-SQL Server , PostgreSQL
Big data management system	Non-relational: Accumulo, ArangoDB , MongoDB
Data warehouse appliance	Apache Hadoop, Oracle
Cloud data warehouse	Google BigQuery, Amazon Redshift

1.4 Проблемы с которыми сталкиваются при

При процессах миграции командам порой приходится решать самые разные неординарные задачи, проблемы с данными могут возникать в самых неожиданных местах. Ниже рассмотрим некоторые из них:

- Разнообразие в исходных системах и платформах
- Несовместимые представления данных
- Сложность преобразований

Разнообразие в исходных системах и платформах

Сегодня на рынке используется буквально более десятка различных технологий систем источников. Пример вида технологий исходной системы, который вы часто видите, показан в Таблице 2. Команда, выполняющая обработку ETL в любой конкретной организации, вероятно, должна понимать по крайней мере 6 из них. Не стоит удивляться, обнаружив крупные банки и телекоммуникационные компании, которые имеют каждую из этих технологий; в одной компании.

Таблица 3 - Разнообразие в исходных системах и платформах

Platform	OS	**DBMS**	**MIS/ERP**
Main Frame	VMS	Oracle	SAP
Computer	Unix	Informix	PeopleSoft
Desktop	Win NT	Access	JD Edwards
	DOS	Text file	

Несовместимые представления данных

Многие системные таблицы из предыдущих версий в настоящее время реализованы в виде набора представлений. Эти представления известны как представления совместимости, и они предназначены только для обратной совместимости. Представления совместимости содержат метаданные, которые были доступны в ранних версиях СУБД. Однако представления совместимости не содержат метаданных, связанных с функциями, появившимися в более поздних версиях.

Сложность преобразований

Существует целый спектр простых, вплоть до очень сложных преобразований, которые можно реализовать. И вероятно, в конечном итоге со многими в большинстве развертываний придется столкнуться. Преобразования, как правило, делятся на три категории следующим образом: Простые скалярные преобразования один-в-один. Отображения одного-во-многих элементов. Сложные преобразования многих-многих-элементов. Например, простое скалярное преобразование -это один-один-один отображение одного набора значений в другой набор значений с использованием простых правил.

Существует шесть основных методов переноса данных. Как правило, они делятся на две широкие категории в зависимости от того, могут ли процедуры выполняться, пока приложение остается работоспособным (онлайн) или приложение должно быть выведено из эксплуатации (автономно) во время фактической миграции. Основными методами являются 18:

• В автономном режиме: резервное копирование и восстановление; восстановить из резервных копий лент; передача ftp и

• Онлайн: репликация на основе массива; управление томами или репликация; и хост-зеркалирование

Во многих случаях гибрид этих методов необходим для удовлетворения требований крупной миграции. Менеджер проекта по миграции данных в тесном сотрудничестве с федеральной командой EDS по студенческой помощи должен определить метод и инструменты, которые будут использоваться для выполнения операций по миграции данных. Метод может отличаться в зависимости от существующих систем. Выбранные метод и инструменты, а также факторы, влияющие на определение, должны быть

включены в План переноса данных. Такие факторы могут включать (но не ограничиваются ими):

• Распределение (местоположение) хранилищ данных

• Ограничения финансирования

• Имеющийся опыт в текущей и целевой среде хранения (например, ограничено ли планирование конкретными вариантами просто из-за имеющегося опыта)

• Выполнение (качественное и количественное) процедур / инструментов

• Защита исходных данных / восстановление

• Гомогенные и гетерогенные требования к хранению

• мультивендорная среда

• Зависимости от внешних деловых партнеров

• Допустимое время простоя

• Время (график) ограничения

• объем данных

• Кадровые ограничения (доступность)

• сложность среды хранения и обработки

• Физическое перемещение

• Несовместимость формата хранения данных (СУБД / СУБД, СУБД / ОС)

• Проблемы конфигурации, связанные с объемом данных

Реконсиляция

Аудит и согласование данных является критическим процессом, который определяет успешную миграцию данных.

Согласование - это этап, на котором целевые данные сравниваются с исходными данными, чтобы убедиться, что данные были преобразованы правильно. Это важно для поддержания миграции на правильном пути, а также для обеспечения качества и количества данных, переносимых на каждом этапе. На уровнях извлечения и загрузки должны быть реализованы программы проверки для сбора количества записей. Анализ и реконсиляция

(A & R) проверяет точность и полноту перенесенных данных и учитывает критерии количества / стоимости (количество перенесенных объектов и т. Д.). A & R не включает проверку данных, которые были перенесены, чтобы убедиться, что данные поддерживают бизнес-процессы. Требованиями сверки руководствуются как функциональные группы - требования владельцев процессов, так и требования технической сверки. Ниже приведены задачи, связанные с процессом A&R.

- Выполнять отчеты в источнике: отчеты A&R могут быть выполнены в прежнем формате, а количество извлеченных записей будет проверено в прежней системе. Количество хэшей для ключевых столбцов отмечено.
- Подтверждение извлечения: затем извлечение данных может быть проверено на соответствие критериям принятия данных. Это также будет включать проверку количества записей и количества хэшей упоминается в отчетах Source A & R
- Запустите промежуточную сверку в слое трансформации: для некоторых объектов промежуточное согласование может быть выполнено на уровне преобразования данных.
- Запускать отчеты по цели: целевые отчеты A & R должны быть выполняется для проверки количества записей и количества хэшей.
-

Выводы по первой главе

1) Миграция данных - процесс переноса данных из исходной базы данных в целевую, в том числе, когда их схемы могут отличаться. Он включает в себя два важных процесса. Первым осуществленным процессом является идентификация, выявление и сбор объектов. Во втором процессе выполняется миграция данных из исходного объекта в целевой объект.

2) Как правильно заметил в своей работе Klaus Haller – «Что касается обслуживания информационных технологий, как никогда ранее компании

сталкиваются с проблемой переноса данных из исходной структуры в целевую структуру данных, тогда как обе структуры различаются на концептуальном и/или техническом уровне.» В то время как первый уровень относится к типам бизнес-объектов, которые являются осмысленными представлениями объектов в деловом мире (например, контракт, продукт, учетная запись клиента), второй уровень обозначает их техническую реализацию в базах данных.

Таким образом, многие исследователи видят миграцию данных как единовременный процесс с поддержкой инструментов тестирования. Однако любое незапланированное и грубое движение этого в этом направлении в форме непрофессионального миграционного проекта подвергает эту компанию более высокому риску. Поэтому крайне важно следовать строгому и поэтапному подходу, который напрямую учитывает риски миграции данных. Но так как на рынке предлагается огромное количество сервисов, которые помогают выполнять трудоемкие миграционные действия и дальнейшее тестирование, то этот процесс при легкой степени кастомизации может пройти достаточно безболезненно.

3) Как показал анализ работ других исследователей и продуктов представленных на рынке, самым оптимальным решением будет попытка разработать алгоритм на базе рабочего инструментария фирмы и внедрением самых продуктивных и полезных сторонних идей, проверок и логики взаимодействий.

Глава 2 Анализ подходов тестирования миграции данных и моделирование алгоритма

2.1 Анализ существующей модели миграции

Проекты миграции похожи на своеобразный айсберг. Заказчик считает, что данные достаточно хорошего качества, близкие к идеальному, что зачастую оказывается не так. Источники данных могут отсутствовать, оказываются не структурированы, обязательно потребуется поддержка от бизнеса, тех, кто будет разбираться в данных, появляются проблемы с производительностью. Требуется сложное преобразование данных, объектные модели источника и целевая оказываются очень сильно не совместимы.

На рисунке ниже схематично как External databases показаны базы данных источника, которые могут поддерживать совершенно разные системы и представлять дополнительную сложность в процессе миграции.

На основе описанной методологии можно выделить следующие основные этапы тестирования:

На этапе извлечения в процессе присутствуют простые валидации, например на соответствие количества столбцов в таблицах. Primary database отражает структуру source database

На этапе между Primary database и Target database происходит трансформация данных в целевую структуру и полная валидация при помощи библиотеки ошибок.

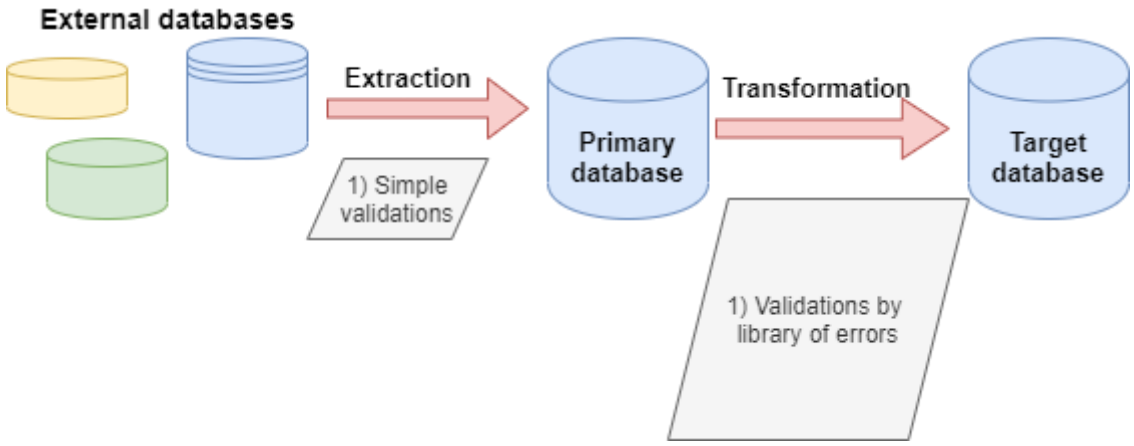

Рисунок 6 - Существующая модель миграции и тестирования базы данных

2.2 Анализ методов тестирования

Сложность преобразований может сделать реализации ETL склонной к сбоям, которые могут поставить под угрозу информацию, хранящуюся в базе данных, что в свою очередь приводит к неправильному анализу результатов. Неисправные сценарии могут привести к неверным данным в базе данных. Таким образом, функциональное тестирование процессов ETL имеет решающее значение. Это тестирование гарантирует, что любые изменения в источнике данных правильно записываются и полностью распространяются в целевое хранилище данных. Способ реализации и выполнения процессов ETL также может привести к неверным данным в целевом хранилище данных. Существует необходимость в систематических, автоматизированных подходах к тестированию, чтобы уменьшить усилия и затраты, связанные с жизненным циклом базы данных.

Факторы, которые влияют на разработку тестов, такие как платформы, операционные системы, сети, СУБД и другие технологии, используемые для реализации хранилищ данных, затрудняют использование общего подхода к тестированию применимого ко всем проектам хранилищ данных. Огромный объем извлеченных данных, трансформированный и загруженный в базу

данных делает полное ручное сравнение данных для тестирования нецелесообразным. Кроме того, тестирование процесса не является разовой задачей, потому что базы данных развиваются, и данные постепенно добавляются, а также периодически удаляются.

Следовательно, тесты должны быть разработаны и реализованы таким образом, чтобы они были повторяемыми. Сбои в любом из компонентов могут привести к неверным данным в целевом хранилище данных это не может быть обнаружено путем оценки целевого хранилища данных изолированно. Выполнение компоненты несколько раз из-за ошибочных настроек, выбранных пользователями, может привести к дублированию данных. Системные сбои или потеря соединения в любом компоненте могут привести к потере данных или дублированию данных в целевом хранилище данных. Ручное участие в запуске процесса может привести к ошибочной настройке параметров, что приведет к неправильным режимам и усечению или дублирование данных или выполнение заданий в неправильном порядке. Используя дубликаты имен для промежуточного хранение файлов может привести к перезаписи важной информации.

Тестирование миграции - это процесс проверки перехода унаследованной системы на новую систему с минимальными перебоями/простоями, сохранением целостности данных и без потери данных, обеспечивая при этом соблюдение всех указанных функциональных и нефункциональных аспектов приложения после миграции. На рисунке 6 изображен процесс обобщенного процесса миграции.

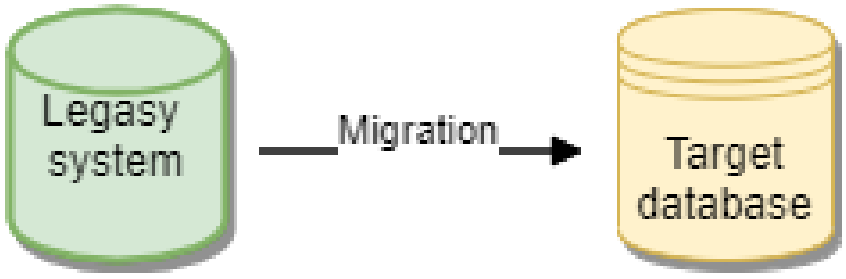

Рисунок 7 – Обобщенная схема миграции

Тестирование должно проводиться как до, так и после миграции.

Различные этапы теста на миграцию, проводимого в лаборатории тестирования, можно классифицировать следующим образом:

- предмиграционное тестирование;
- миграционное тестирование;
- тестирование после миграции.

В дополнение к вышесказанному, следующие тесты также выполняются как часть всей миграции:

- проверка обратной совместимости;
- тестирование отката.

Перед проведением этого тестирования любому тестировщику необходимо четко понимать следующие пункты:

1. Изменения, происходящие как часть новой системы (сервер, интерфейс, БД, схема, поток данных, функциональность и т. д.).

2. Чтобы понять фактическую стратегию миграции, изложенную командой. Как происходит миграция, пошаговые изменения происходят в бэкэнде системы и скриптах, ответственных за эти изменения.

Следовательно, важно тщательно изучить старую и новую системы, а затем соответствующим образом спланировать и спроектировать тестовые наборы и сценарии тестирования, которые будут охвачены в рамках вышеупомянутых этапов тестирования, и подготовить стратегию тестирования.

В информационных технологиях, как никогда ранее компании сталкиваются с проблемой переноса данных из исходной структуры в целевую структуру данных, тогда как обе структуры различаются на концептуальном и/или техническом уровне.

Таким образом, миграция данных понимается как единовременный процесс с поддержкой инструментов тестирования. Однако любое незапланированное и грубое движение этого процесса в форме

непрофессионального миграционного проекта подвергает компанию более высокому риску. Поэтому крайне важно следовать строгому и поэтапному подходу, который напрямую учитывает риски миграции данных.

Обращаясь к алгоритму примененному на проекте на первых итерациях миграции мы увидим, что без соответствующей поддержки команды миграции был применен иной подход с использованием инструмента поддержки, который помогает визуализировать запуск скриптов и чтение отчетов.

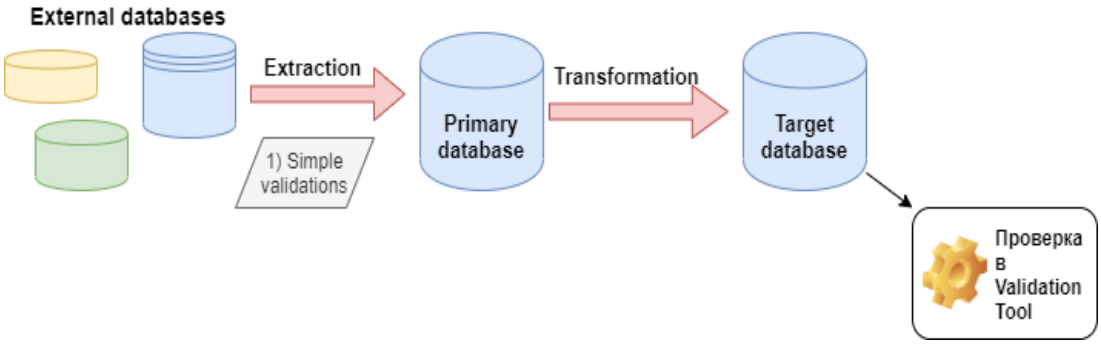

Рисунок 8 – Модель проверки миграции и тестирования базы данных на начальных итерациях

Validation Tool помогает настроить автоматическую проверку данных, в том числе после миграции. Программа сравнивает объекты с источниками данных или по тем правилам, которые задает пользователь системы.

Validation Tool может считывать данные из различных источников. На рисунке 12– Выбор представления отчета показаны несколько форматов доступных вариантов выгрузки отчета с извлеченными данными. В случае предпочтительном для пользователя рассматриваемого в работе, выгрузка происходит в Excel файл, и соответственно был выбран Excel File Reader.

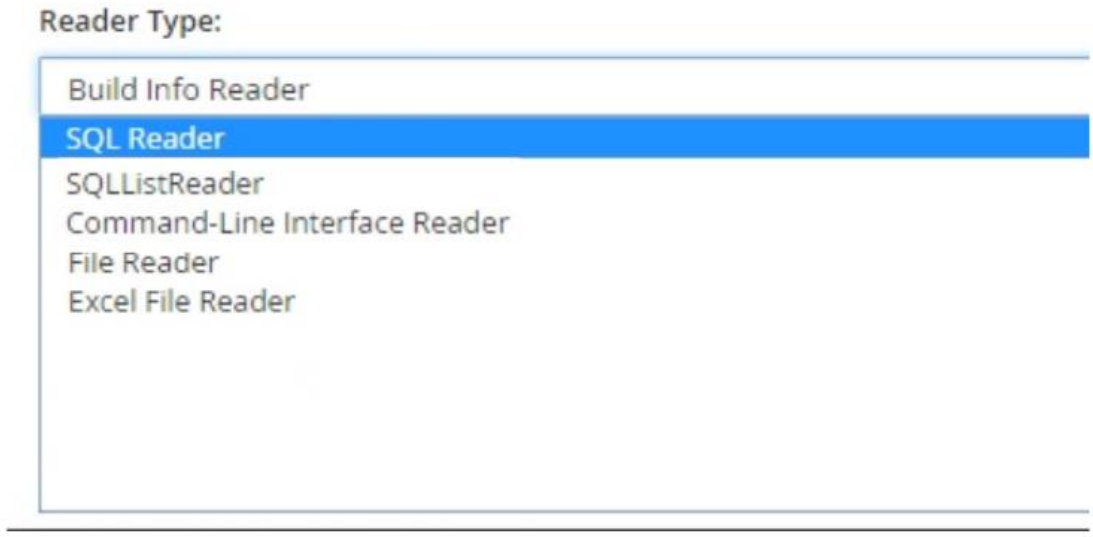

Рисунок 9– Выбор представления отчета

Но стоит учесть, что данный подход был применен уже после загрузки данных в целевую базу, куда они пришли в достаточно некорректном состоянии, что только усложнило все процессы тестирования, и данный метод нельзя принять как продуктивный.

В тоже время И.В. Нефедова в своей работе «Тестирование кода при переносе базы данных» рассматривает иной подход.

Сохраненный образ исходной базы данных определяет то состояние, с которого всегда будет начинаться система запросов. Полученная в результате миграции база данных является результатом данного процесса к исходному состоянию, которое было сохранено. Поэтому процесс работы запросов каждый раз начинается с двух баз, которые находятся в стандартных состояниях. Все ходы записываются на виртуальной машине.

То, что получается на каждом шагу, будет сравниваться как разности множеств A-B и B-A. Если возникнет такая ситуация, что обе разности будут представлять собой пустое множество, результат будет считаться положительным т.е. транслированный код тождественен исходному. В обратном случае, когда множества не являются пустыми, должна

производиться попытка сравнений A-B и B-A, чтобы локализовать ошибку, а результаты сравнений будут записаны в отчет, где будут зафиксированы различные ошибки, возникшие при выполнении переведенного и исходного кодов.

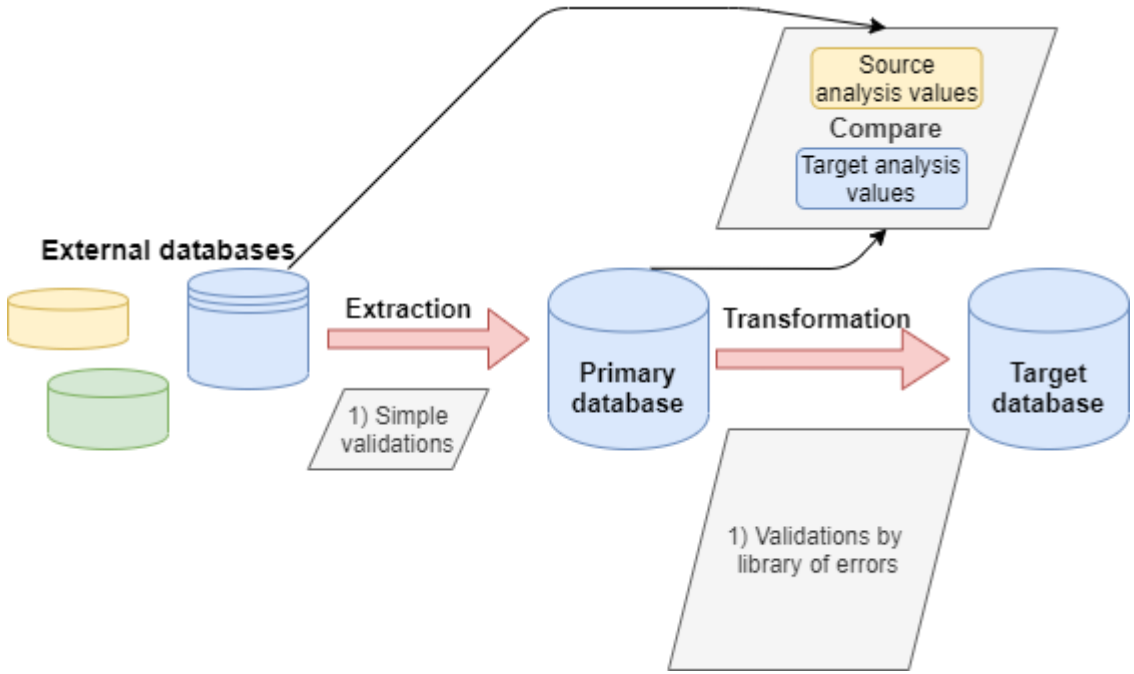

Рисунок 10 –Модель миграции и тестирования базы данных

Обратимся к работе Питера Джонсона «Automated Testing of Database Schema Migrations», где он обращает внимание на тот факт, что обычное тестирование не позволило предотвратить несовместимые миграции в производственной базе данных. Однако он рассматривает несколько методов для выполнения некоторых тестов баз данных и их миграционных схем.

Питер Джонсон ищет способ автоматизировать генерацию предварительных условий благодаря хорошо структурированному формату журнала изменений, который используется в Liquibase.

В **методе утверждения** для проверки журналов изменений Liquibase по состоянию базы данных следует использовать формат Liquibase для создания предварительных условий, которые выполняются в рабочей базе данных без утечки данных. Подмечая его плюс в том, что хотя **метод утверждения** может помешать несовместимым миграциям нарушить производственную базу данных, он делает это на очень поздней стадии традиционной сборки билда.

Следом Питер Джонсон рассматривает другой метод, основанный на сравнении схем. Пользователь должен экспортировать схему производственной базы данных, вставить ее в базу данных в среде тестирования и выполнить Liquibase с логами. Явным минусом **Метода схемы** является невозможность проверить изменения, связанные с записями данных, но он может обеспечить приемлемую надежность для проверки совместимости между миграциями и состояниями базы данных.

Таким образом 108 тестовых случаев, состоящих из миграции и состояния базы данных, были использованы для тестирования всех методов. Были использованы как действительные, так и недействительные контрольные примеры, которые не были совместимы с состоянием базы данных. Распределение прерванных, неудачных и успешных миграций было проанализировано Питером Джонсоном наряду с:

- автоматизацией;
- отслеживаемостью;

- надежностью;
- совместимостью базы данных;
- сохраняемостью;
- масштабируемостью для каждого метода.

И метод подтверждения, и метод схемы могут использоваться для тестирования большинства процессов миграций без доступа к рабочим данным, снизив частоту отказов, благодаря использованию схемы уменьшения сложности с ограничением уникальности. И результаты по работе Питера Джексона вы можете видеть на рисунке 7.

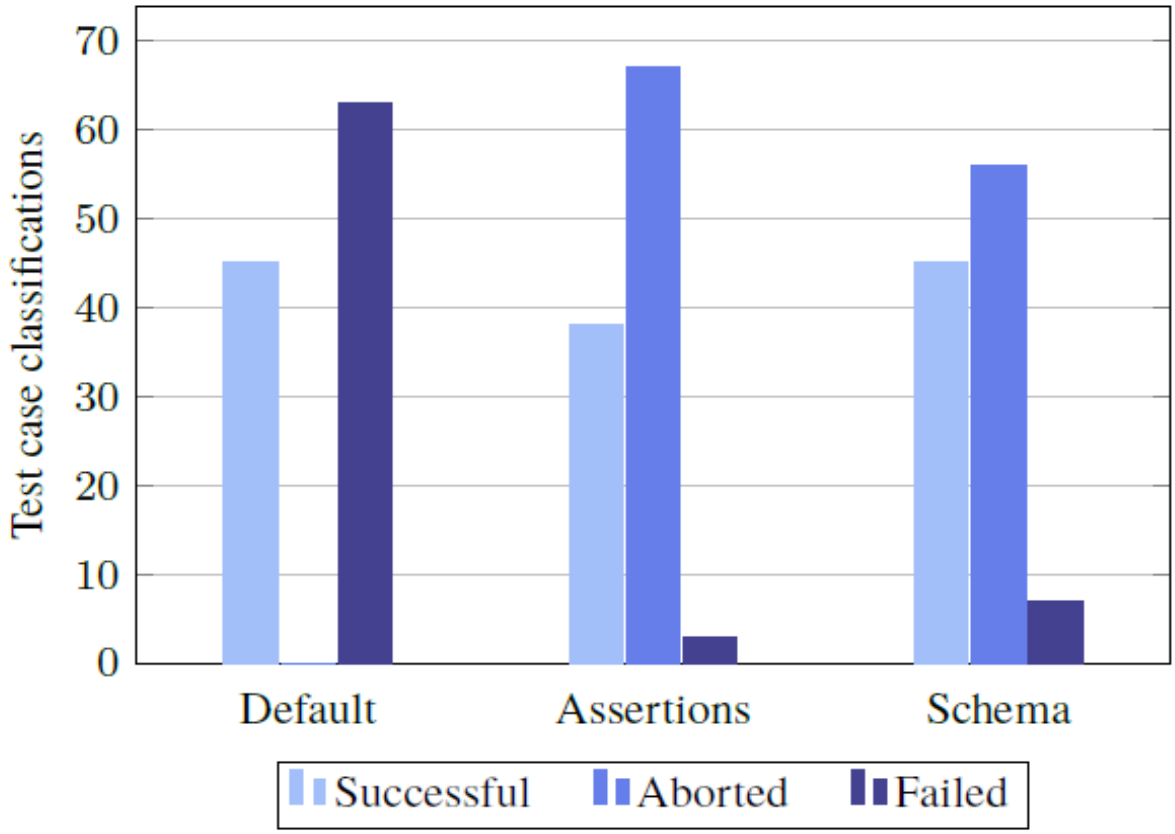

Рисунок 11 - Суммарный результат всех испытаний

Но все же учитывая глубокую кастомизированность методов и невозможность применения в условиях бизнеса компании, следует рассмотреть иные подходы.

Несмотря на кажущуюся точность метода, следует обратиться к собственному алгоритму, так как существующий процесс использует несколько посредников баз данных, в которых исходные данные трансформируются под структуру таблиц, построенных в бизнесе нашей компании, что автоматически отвергает подход предложенный Нефедовой.

2.3Выбор метода тестирования

Консистентность.

Это свойство гарантирует, что содержимое атрибута в целевой таблице соответствует соответствующим в исходной таблице (таблицах) на основе спецификаций преобразования. Свойство консистентности сравнивает содержимое атрибута исходных таблиц с данными целевого хранилища данных для спецификаций, указанных в правилах преобразования. Это обеспечивает семантическую правильность в процессе трансформации. Поскольку не может быть простой один-к-одному соответствие между источником и целевым атрибутом, проверка этого свойства часто является сложной задачей.

Качество данных

Необходимость миграции данных - хорошее качество данных.

для приложения. Если качество данных в целевой базе данных / хранилище данных не соответствует ожиданиям, цель всего приложения не выполняется. Плохое качество данных может привести к появлению приложения задолго до развертывания. Если данные не проходят базовый набор правил проверки, определенных в целевом приложении, загрузка данных завершится сбоем, что увеличит стоимость переделки и займет больше времени для ввода в действие.

Организации должны обеспечить доверие пользователей к данным. к

Полностью доверяя данным, бизнес должен иметь возможность отслеживания каждого элемента данных посредством процессов профилирования, проверки и очистки данных.

Проверка данных затрагивает четыре аспекта:

- полнота;
- правильность;
- согласованность структуры данных;
- совместимость с другими приложениями.

Таким образом, сложное сочетание автоматических и ручных сравнений должны применяться для проверки данных и их структуры. Зачастую основная задача состоит в том, чтобы сравнить данные из двух источников - баз данных, относительно предварительно определенных сравнительных критериев. Тесты на полноту и соответствие типов бизнес-объектов, которые отсутствуют в целевой базе данных (например, клиент, который не был перенесен) или новые объекты (например, клиент в целевом приложении, который не существует в исходное приложение). Такие тесты, как правило, называют сверкой, посмотрите на тысячи бизнес-объектов, требующих для автоматизации сравнения бизнес-объектов в исходной и целевой базе данных. Сценарии сверки автоматически сопоставляют первичные ключи (соответствующие идентификаторы) источника и цели, для обеспечения полноты данных В итоге такая сверка помогает проверить, остаются ли типы бизнес-объектов неизменными. Но в нашем случае, я предлагаю рассмотреть процесс проверки с другой стороны и провалидировать имеющуюся выгрузку на наличие самых часто встречающихся проблем при прохождении End-to-End сценариев.

Проверка будет состоять из сочетания ряда параметров, которые должны быть верны, чтобы миграция была совместимой с состоянием базы данных.

2.4 Моделирование алгоритма

Рассмотрев логику процесса миграции баз данных, подходов к тестированию этого процесса, следует обратиться сформулировать собственный подход к решению задачи.

Предыдущие исследования не дало четкого алгоритма для решения именно нашей задачи, но и не могли это сделать, потому что каждое решение серьезно дорабатывается под конкретную бизнес модель, но мы можем взять некоторые строительные блоки для построения своего алгоритма.

Модель алгоритма представлена на рисунке 8 и основывается на решении проблемы неконсистентности данных адресов, которые были мигрированы в Primary base. Эту проблему было предложено решить сверкой текущих значений данных по адресам пользователей в таблицах с существующими реальными данными ГИС систем. Для чего был реализован интерфейс для запуска SQL-скриптов и сверки с базой реальных данных .

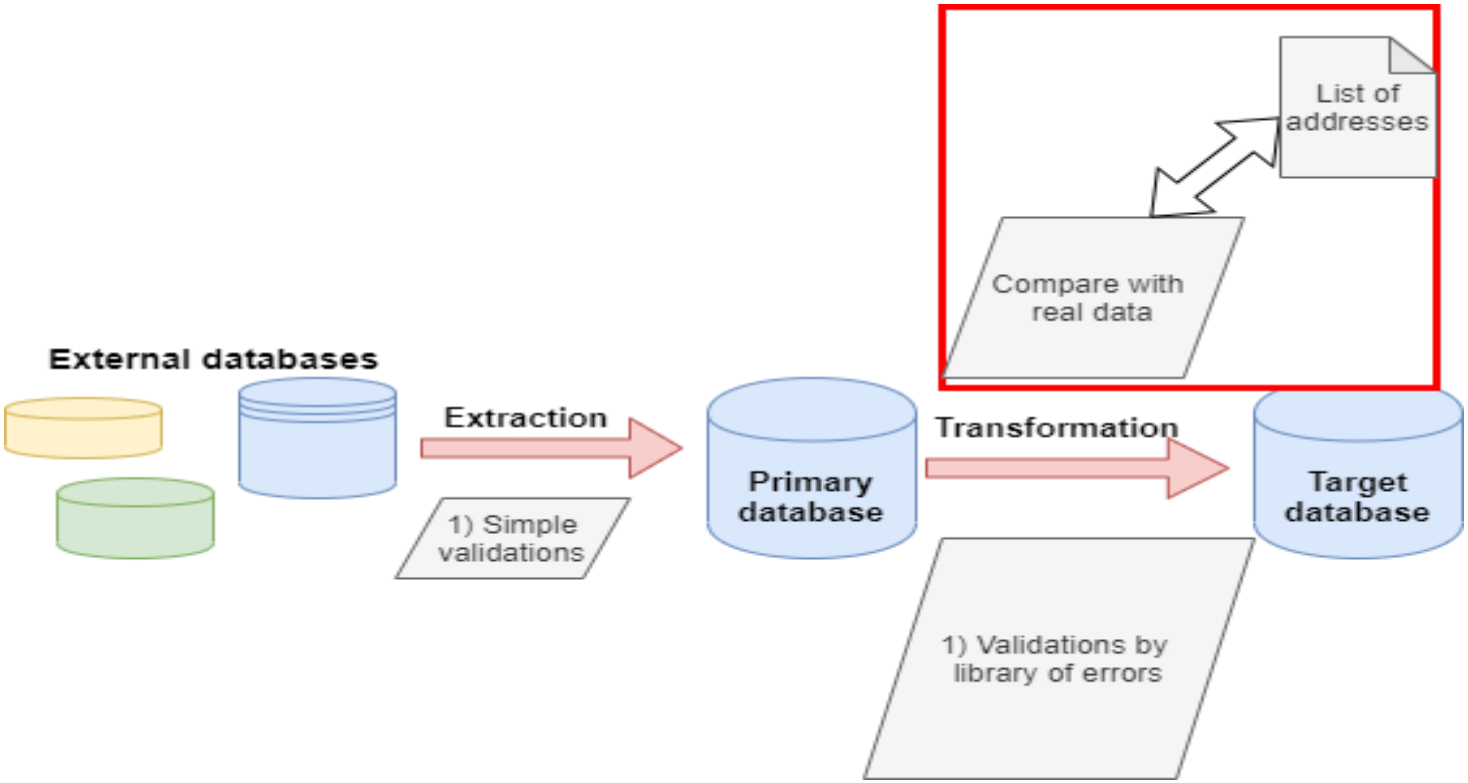

Рисунок 12 – Алгоритм тестирования миграции базы данных

Выводы по 2 главе

1) Самый разный набор платформ, операционных систем, сетей, СУБД и других технологий, используемых для реализации хранилищ данных, делают невозможным использование общего подхода к тестированию применимого ко всем проектам хранилищ данных.
2) Тесты должны быть разработаны и реализованы таким образом, чтобы они были повторяемыми. Сбои в любом из компонентов могут привести к неверным данным в целевом хранилище данных.
3) Большинство методов работают с процессом сравнения запросов с двух баз, то есть базы- источника и целевой
4) Сторонние решения не дают требуемой совместимости с существующим проектом, архитектурой баз данных и бизнес-процессами, поэтому был разработан алгоритм на основе имеющихся инструментов с учетом практики других проектов.

Глава 3 Разработка алгоритма тестирования миграционных данных

В данной главе рассматривается подход к созданию тестов для гарантии того, что полученные данные из исходных баз данных не теряются или неправильно модифицируются процессом ETL или не создаются дубликаты. В ходе исследования предлагается автоматизация генерации тестов для уменьшения ручных проверок, и повышения повторяемость теста.

3.1 Маппинг таблиц

Определить сопоставления источника с целью

Проверка требует наличия сопоставлений источника с целью, которые описывают, как один или несколько атрибутов в исходных таблицах связаны с одним или несколькими атрибутами в целевом хранилище данных. Эти сопоставления источника с целью получены из преобразования ETL. В этой работе предполагается, что преобразование ETL как правило документировано в структурированном виде. В данном хранилище, используемое в нашей работе, правила преобразования ETL описываются как коды представления SQL.

Это хранилище данных включает только однозначные сопоставления записей. В нашем подходе рассматривается каждая таблица или атрибут в целевой базе и валидируется по наличию нулевых значений, дубликатов и некорректно перенесенных varchar данных.

Первоначальные преобразования один-к-одному и многие-к-одному и «многие ко многим» остаются прежними. Таким образом, остается несколько однозначных многозначных отображенияй. Ниже последует описание процесса работы с различными отображениями типов в хранилище данных. В таблице 3 представлено

Таблица 4 – Маппинг реквестов

Маппинг	**Таблицы источника**	**Запрос на анализ**	**Утверждения**
Individual_address_data_id	Individual_address_data_g_cst	select group_id from (select group_id, ADR_ELEM8, upper(ADR_ELEM1), upper(ADR_ELEM2) from Individual_address_data_g_cst where ind.groupping_id is not null and ind.customer_type in ('o', 'n') and addr.address_id = ind.address_id and addr.ADDR_TYPE = 'S' and ENTITY_TYPE group by groupping_id, ADR_ELEM8))	One individual should have only 1 address. GROUPPING_ID = <%GROUP_ID%>
Individual_address_data_id	Individual_address data_rca_one	Select group_id from (select group_id_elem5,upper(adr_elem8) From individual_address_data_rca_one Where group_id is not null and ind.customer_type in ('o', 'n') and addr.address_id = ind.address_id)	One individual should have only 1 address. GROUP_ID = <%GROUPPING_ID%>
Customer_contract_Name	Customer_contract	Select group_id from (select group_id_elem5,upper(adr_elem8) From Customer_contract Where group_id is not null and ind.customer_type in ('o', 'n') and addr.address_id = ind.address_id)	
Individual_address_data_id	Individual_address data_bca	select group_id from (select group_id, BCA_GROUP_ID, ADR_ELEM8, upper(ADR_ELEM1), upper(ADR_ELEM2) from Individual_address data_bca where ind.group_id is not null and ind.BCA_GROUP_ID is not null and addr.address_id = ind.address_id and addr.ADDR_TYPE = 'S' and ENTITY_TYPE	GROUP_ID = <%GROUP_ID%> One BCA should have only 1 address

		= 'CUSTOMER' and LINK_TYPE = 'C' group by group_id, BCA_GROUP_ID, ADR_ELEM8, upper(ADR_ELEM1), upper(ADR_ELEM2))	
Individual_address_data_id	Individual_address data_not_found_cst	select rowid from Individual_address data_not_found_cst where address_id is not null and customer_type <> 'R' and group_id is null and not exists(select 1 from ADDR_DATA where ADDRESS_ID_AD = ind.address_id)	There is no address data for this ADDRESS_ID in source table (SDB_HU2_ADDR_DATA.ADDRESS_ID_AD). GROUP_ID = <%GROUP_ID%>, ADDRESS_ID=<%ADDRESS_ID%>, CUSTOMER_ID=<%CUSTOMER_ID%>
Individual_address_data_id	INDIVIDUAL_DATA_ADDRESS_ID_NNL	select rowid from INDIVIDUAL_DATA_ADDRESS_ID_NNL where address_id is null and group_id is null union all select rowid from INDIVIDUAL_DATA_ADDRESS_ID_NNL where group_id in (select grouping_id from SDB_HU2_MIGR_INDIVIDUAL_DATA where address_id is null and group_id is not null))	ADDRESS_ID should be not null. In a case of group. all records are marked invalid.

На рисунке 9 представлена сама схема базы данных

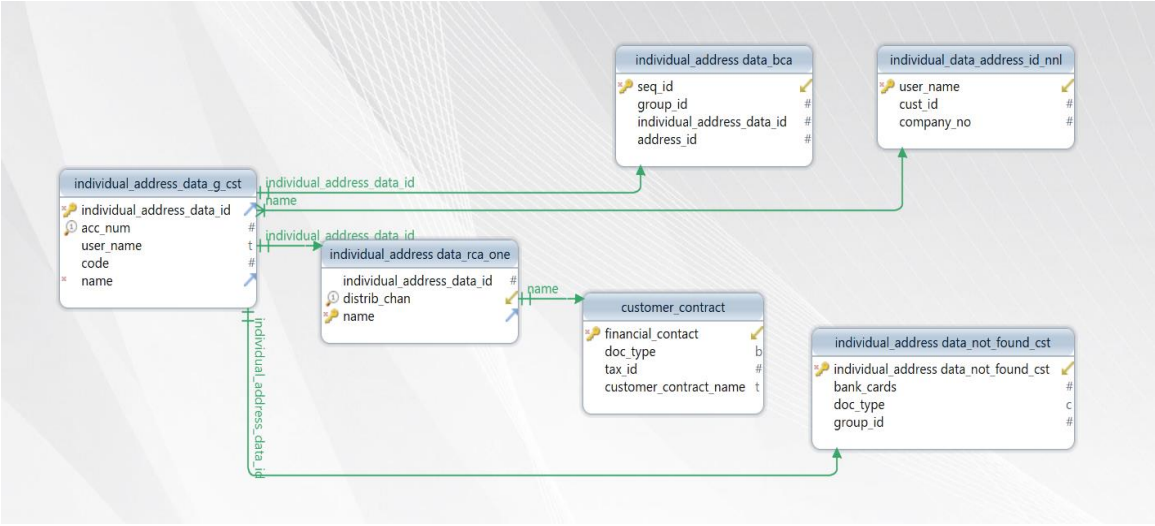

Рисунок 13- Упрощенная схема базы данных

Краткие спецификации полей сущностей приведены в таблицах 2.2 – 2.17. Буквой (О) обозначены те поля, значения которых могут быть получены из других открытых информационных систем.

Таблица 5.1- Individual_address_data_g_cst

№ п/п	Название поля	Описание поля
1	- Individual_address_data_id	Идентификационный номер адреса (первичный ключ)
2	acc_num	Идентификационный номер клиента
3	user_name	Идентификационное имя клиента
4	code	Номер договора
5	name	Реальное имя клиента

Таблица 5.2-- Individual_address_data_rca_one

№ п/п	Название поля	Описание поля
1	Individual_address_data_id	Идентификационный номер адреса (первичный ключ)
2	Distrib_chan	Способ доставки пакетов
3	name	Реальное имя клиента

Таблица 5.3- Individual_address_data_bca

№ п/п	Название поля	Описание поля
1	Seq_id	(первичный ключ)
2	group_id	Идентификационный номер группы пользователей
3	Individual_address_data_id	Идентификационный номер адреса (первичный ключ)
4	Address_id	Идентификационный номер адреса

Таблица 5.4—Customer_contact

№ п/п	Название поля	Описание поля
	financial_contact	Финансовые контактные данные
	doc_type	Тип договора
	tax_id	Идентификационный номер тарифной ставки
	customer_contact_name	Контактное имя заказчика

Таблица 5.5— Individual_address_data_nnl

№ п/п	Название поля	Описание поля
1	User_name	Идентификационное имя клиента
2	Cust_id	Идентификационный номер заказчика
3	Company_no	

Таблица 5.6 Individual_address_data_not_found_cst

№ п/п	Название поля	Описание поля
1	Individual_address_data_not_found_cst	Идентификационный номер не найденнго заказчика
2	bank_cards	Идентификационный данные карты
3	doc_type	Тип договора
4	group_id	Идентификационный номер группы пользователей

3.2 Генерация запросов

Табличные отображения один к одному.

Для каждой строки сопоставления таблицы «один к одному» генерируются запросы для свойства соответствия количества записей. Запрос на анализ для целевого хранилища данных возвращает единую таблицу плюс количество записей в целевой таблице. На рисунке 10 – Связь «Один-к-одному»

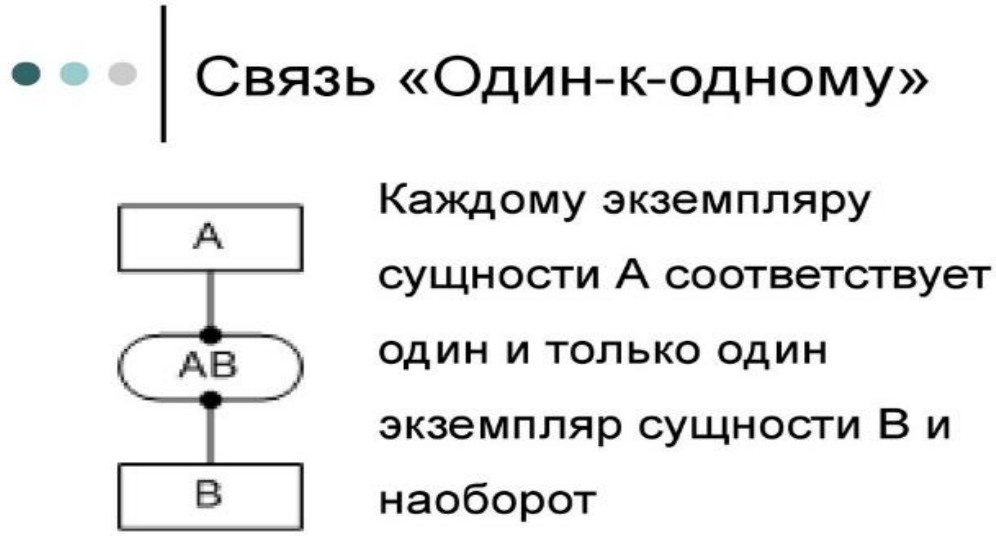

Рисунок 14 – Связь «Один-к-одному»

Отображение таблиц "многие к одному".

Для каждой строки сопоставления таблицы «многие к одному» генерируются запросы на анализ для соответствующего анализа типа данных. Анализ запроса для цели возвращает различное количество записей в целевой таблице. Запрос анализа для источника возвращает количество записей в самой левой таблице в левом JOIN, и количество записей в самой правой таблице в операции RIGHT JOIN, которые при условии выбора условия. Для более сложных случаев, таких как операция INNER JOIN или комбинация различных типов операций JOIN, для запроса анализа источника не может подсчитать число записи, полученные из объединения, если оно не выполняет все операции соединения. В представленном случае имеется только цепочка операций LEFT JOIN. В этом случае количество записей в крайнем левой таблице – это минимальное количество записей, которые будут возвращены. В результате сравнивания числа отдельных записей в целевой таблице с количеством записей в самой левой таблице объединения операция при условии выбора условия. На рисунке 11 – «Один-ко-многим»

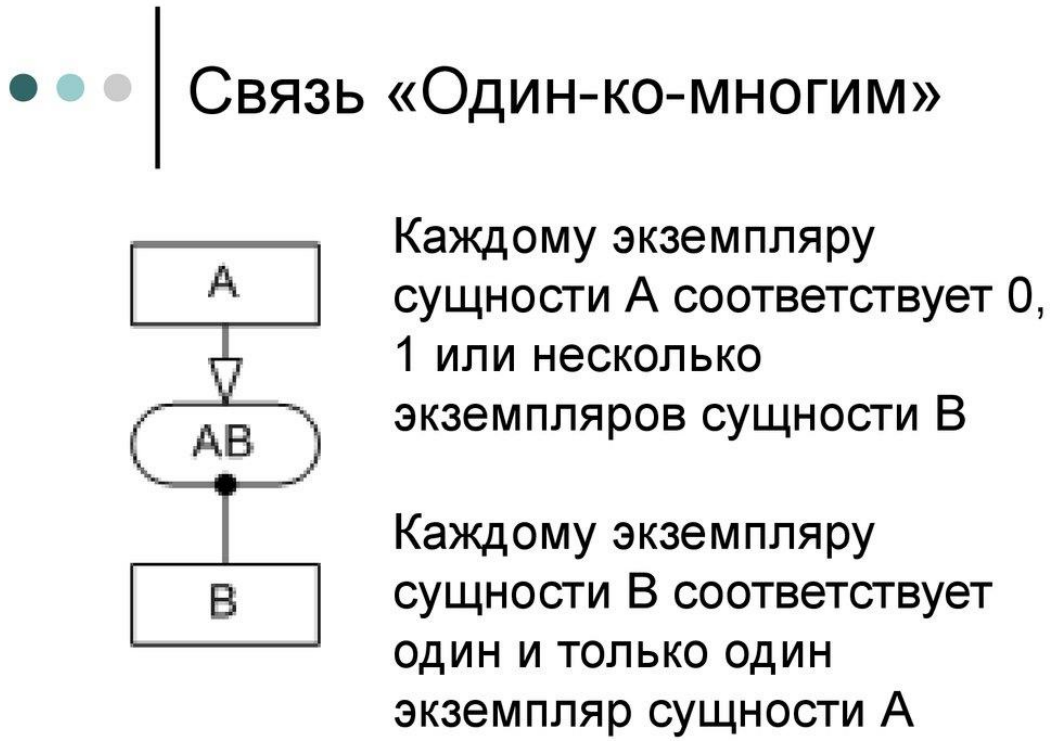

Рисунок 15 – «Один-ко-многим»

Отображение атрибутов.

Для каждой строки в таблице сопоставления мы генерируем запрос для анализа соответствия атрибутов.

Соответствие значения атрибута.

Этот анализ выполняется для целевых атрибутов с любым типом данных. Если это сопоставление атрибутов «один к одному», то запрос возвращает все значения целевых атрибутов.

Соответствие атрибута.

Если это сопоставление атрибутов «один к одному», то запрос возвращает число NULL. значения исходного и целевого атрибутов. Если это сопоставление атрибута «многие к одному», то цель Запрос на анализ

возвращает количество пустых значений целевого атрибута и исходного анализа запрос возвращает количество значений NULL соответствующих атрибутов источника, применяя операция: Count (ISNULL (операция (исходные столбцы))).

Соответствие длины атрибута.

Если это сопоставление атрибутов «один к одному», то запрос возвращает среднее значение длина исходного и целевого атрибутов. Если это сопоставление атрибутов многие-к-одному, то запрос анализа цели возвращает среднюю длину целевого атрибута и исходный анализ Запрос возвращает среднюю длину соответствующих исходных атрибутов, применяя операцию: Avg (длина (операция (исходные столбцы))).

Атрибут типа данных соответствует.

Если это сопоставление атрибутов «один к одному», то запрос возвращает тип исходных и целевых атрибутов. Если это сопоставление атрибута «многие к одному», то цель запрос анализа возвращает тип целевого атрибута, а запрос анализа источника возвращает тип соответствующих исходных атрибутов путем применения операции: DATA_TYPE (операция (источник столбцы)).

3.2 Реализация проверки адресов

Реализация результатов исследования была осуществлена на базе ИТ-проекта компании NetCracker.

Проверки различной глубины глубоко интегрированы во все бизнес-процессы миграции компании. Таким образом, качество преобразованных данных является критично важным для конечного результата, что

обуславливает необходимость внедрения более эффективного алгоритма проверки данных.

Проверка данных производилось другим инструментом что имело свои недостатки:

- скорость обработки данных была низкой, из-за чего многие инциденты решались со значительными задержками;
- часть инцидентов не рассматривалась;

Предложенный алгоритм был реализован при помощи программной системы с открытым исходным кодом - **Jenkins.** П роцесс обращается к предзагруженной гис-базе и сравнивает мигрированные адреса с реально существующими.

На рисунке ниже отображено рабочее окно для настройки запуска сессии сверки адресов.

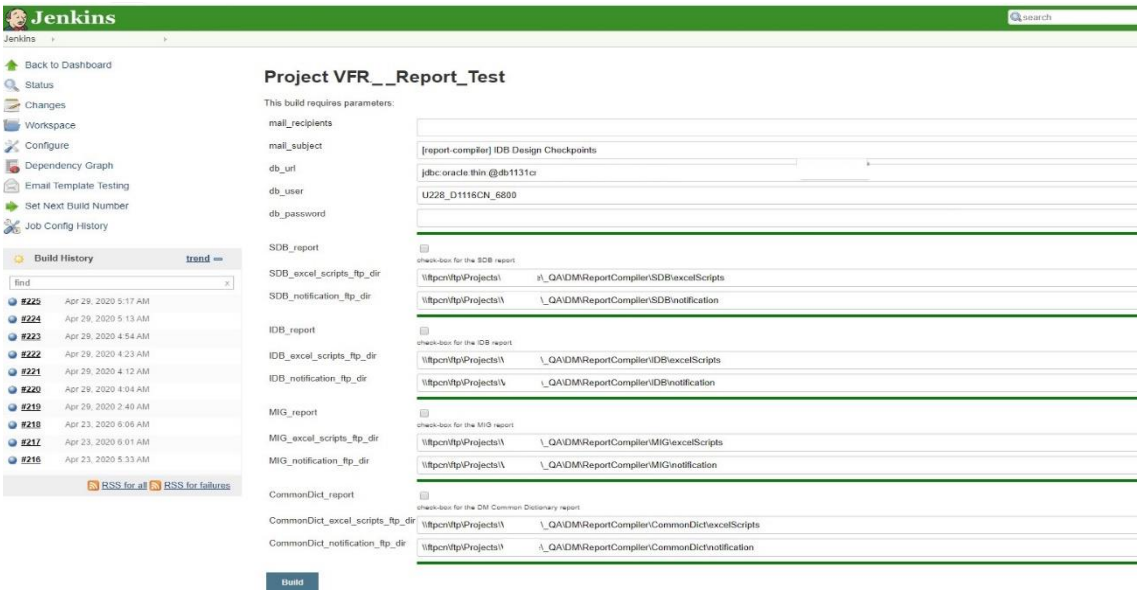

Рисунок 16 – Настройка сессии в Jenkins

На следующем рисунке отображен результат проверки, выгруженный в формате Excel-файла.

ACCOUNT_NUMBER	Att_name	USER_NAME	Entity_id	Sales code	Current warehouse	value_report
AGR	Address line ID	11603	9156203452013074068	11603	AGR001	#N/A
AGR	Address line ID	11786	9156209321713062196	11786	AGR001	#N/A
AGR	Address line ID	11977	9156245430013022807	11977	AGR001	#N/A
AGR	Address line ID	80192	9156010647413697004	80192	AGR001	#N/A
AGR	Address line ID	11553	9156028938213734263	11553	AGR002	#N/A
AGR	Address line ID	12123	9155611872213269021	12123	AGR002	#N/A
AGR	Address line ID	11677	9155888133913650396	11677	AGR002	#N/A
AGR	Address line ID	11580	9155890940813237786	11580	AGR005	#N/A
AGR	Address line ID	10170	9155391909013559944	10170	AGR005	#N/A
AGR	Address line ID	11181	9155389416313037499	11181	AGR005	#N/A
AGR	Address line ID	10842	9156208033113801821	10842	AGR006	#N/A
AGR	Address line ID	11759	9156208230713945059	11759	AGR006	#N/A
AGR	Address line ID	11833	9155389478813091876	11833	AGR006	#N/A
AGR	Address line ID	11761	9155389530413126514	11761	AGR006	#N/A
AGR	Address line ID	10082	9155604084713516905	10082	AGR006	#N/A
AGR	Address line ID	12144	9155391765813474540	12144	AGR006	#N/A
AGR	Address line ID	10045	9155391950113575438	10045	AGR007	#N/A
AGR	Address line ID	11552	9156139326813454223	11552	AGR007	#N/A
AGR	Address line ID	10050	9156156684813406704	10050	AGR007	#N/A
AGR	Address line ID	11695	9155391973113579945	11695	AGR007	#N/A
AGR	Address line ID	11584	9156217491113971667	11584	AGR007	#N/A
AGR	Address line ID	11402	9156246175513360501	11402	AGR007	#N/A
AGR	City/region	11120	9155613808913947125	11120	AGR008	#N/A
AGR	City/region	70613	9156269956613705982	70613	AGR008	#N/A
AGR	City/region	11823	9156098157513558465	11823	AGR009	#N/A
AGR	City/region	11996	9156089639113724089	11996	AGR009	#N/A
AGR	City/region	11415	9156064162113944064	11415	AGR009	#N/A
AGR	City/region	11619	9156087699313975532	11619	AGR010	#N/A
AGR	City/region	11805	9155708801913132058	11805	AGR010	#N/A

Рисунок 17 – Отчет проверки

3.5 Оценка эффективности

Оценка эффективности данной работы была проведена в сравнении с данными по предыдущим итерациям проекта компании с сопоставимым объемом данных для миграции и задействованных параметров.

Был использован другой инструмент, который уже был упомянут во 2 главе.

Validation Tool помогает настроить автоматическую проверку данных, в том числе после миграции. Программа сравнивает объекты с источниками данных или по тем правилам, которые задает пользователь системы.

В таблице 6 – Искомые дефекты, представлены типы расхождений.

Несоответствие данных	**Описание**	**Пример**

Недостающие данные	Значения некоторых полей данных отсутствуют в целевой базе данных.	Пропущенные значения данных при передаче данных из источника в целевую базу данных или значения данных не полностью передаются в целевую базу данных.
Null	Неправильное преобразование исходных значений NULL в целевую базу данных.	Значения NULL исходного поля данных должны преобразовываться по умолчанию в целевое поле данных. Однако из-за неправильной логики реализации это приводит к тому, что целевое поле данных содержит значения NULL.
Ошибки логики преобразования	Логика преобразования не соблюдается, что приводит к ошибкам в значениях данных	Целочисленное значение данных по умолчанию исходной базы данных должно быть перенесено в целевую базу данных в процентном формате, что не выполняется должным образом в процессе миграции, что приводит к неправильным данным
Дубликаты записей	Записи, которые похожи на две или более записи, называются дублирующими записями.	Запись имен сущностей и атрибутов с уникальным идентификатором повторяется более одного раза

Validation Tool может считывать данные из различных источников. На рисунке 12– Выбор представления отчета показаны несколько форматов доступных вариантов выгрузки отчета с извлеченными данными. В случае предпочтительном для пользователя рассматриваемого в работе, выгрузка происходит в Excel файл, и соответственно был выбран Excel File Reader.

Далее добавляются валидационные параметры, которые определены как:

- **поиск нулевых значений**
- **дубликатов**
- **некорректно перенесенных данных**
- **недостающие данные**.

Затем в рамках разработанного тест-кейса применяется код, который обрабатывает excel таблицу в рамках поиска некорректных данных.

Как итоговый результат программа отображает данные по каждому запуску отдельно по каждой таблице, как представлено на рисунке 15.

поиск нулевых значений

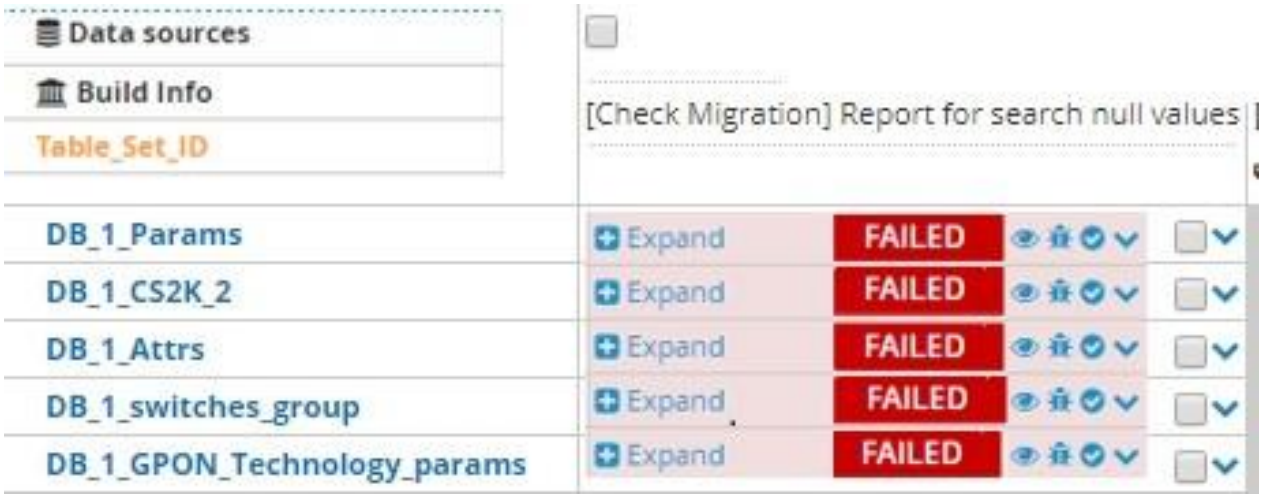

Рисунок 18 – Отчет по поиску нулевых значений

Сама метрика в контексте ПО — это выражение какого-либо качества самого продукта, свойства или процесса его разработки. Другими словами,

это то, с помощью чего мы можем измерить, сравнить и оценить качество миграции в нашем случае.

Для оценки выделим такие основополагающие параметры как:

- плотность дефектов;
- коэффициент регрессии;
- скорость обработки;

Где плотность дефектов – это количество дефектов в отдельном модуле. Коэффициентом регрессии является количество дефектов обнаруженных после стадии тестирования миграции, но относящиеся к ошибкам процесса миграции к общему количеству регрессионных дефектов. Скорость же обработки показывает отношение количества проверенных атомарных модулей к количеству итераций.

Теперь рассмотрим подробнее каждую характеристику.

3.5.1. Плотность дефектов

Эту метрику можно считать базовой для принятия решения по эффективности методики. Рассчитывается как доля дефектов от общего их числа, приходящихся на отдельный модуль в рамках итерации.

Таблица 7 - Метрика плотности дефектов

Модуль	Первые итерации	Последующие итерации	% Соотношение
Module 1	35	48	37%
Module 2	20	31	55%
Module 3	38	49	28%
Module 4	15	17	13%

В данной таблице 5 можно видеть такую картину, что при тестировании каждого модуля с использованием предложенного алгоритма

количество найденных дефектов существенно увеличилось. А именно в процентном соотношении от 13% до 55%. Разительная разница в количественной плотности дефектов может насторожить касаемо верности подсчетов, но тут же стоит учитывать, что был подобран максимально схожий по функционалу проект, с аналогичным набором предоставленных модулей заказчику, хотя и было протестировано разными командами людей, что безусловно вносит некую плавающую погрешность.

На диаграмме, представленной на рисунке 19 более наглядно представлена картина распределения обнаруженных дефектов.

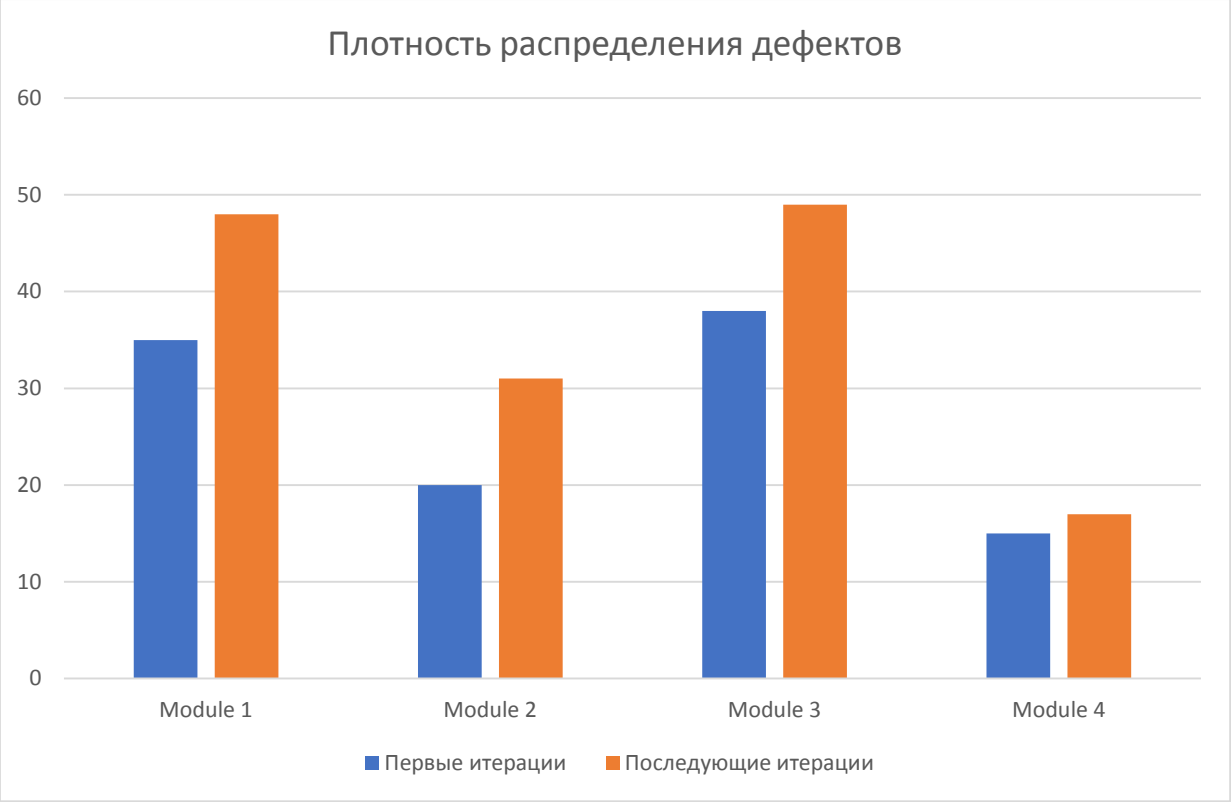

Рисунок 19 - Плотность распределения дефектов

3.5.2 Коэффициент регрессии

Косвенно по этому показателю также можно посмотреть, на что уходят усилия команды — происходит ли тестирование новых фич и модулей, или основное время уходит на исправления некачественной миграции данных.

Как уже указывалось выше, коэффициент регрессии показывает количество дефектов, обнаруженных после стадии тестирования миграции (но относящиеся к ошибкам процесса миграции), к общему количеству регрессионных дефектов.

Таблица 8 – метрика коэффициента регрессии

Модуль	Коэффициент на прошлом проекте	Коэффициент на новом проекте
Module 1	0.15	0.1
Module 2	0.19	0.18
Module 3	0.19	0.15
Module 4	0.11	0.12

Как можно заметить из таблицы 6 выше, чем выше коэффициент, тем хуже результат. Количество дефектов стадии регрессии, на взятом за основу проекте, было больше в 3-х протестированных модулях из 4-х, что так же подтверждает эффективность предложенного алгоритма.

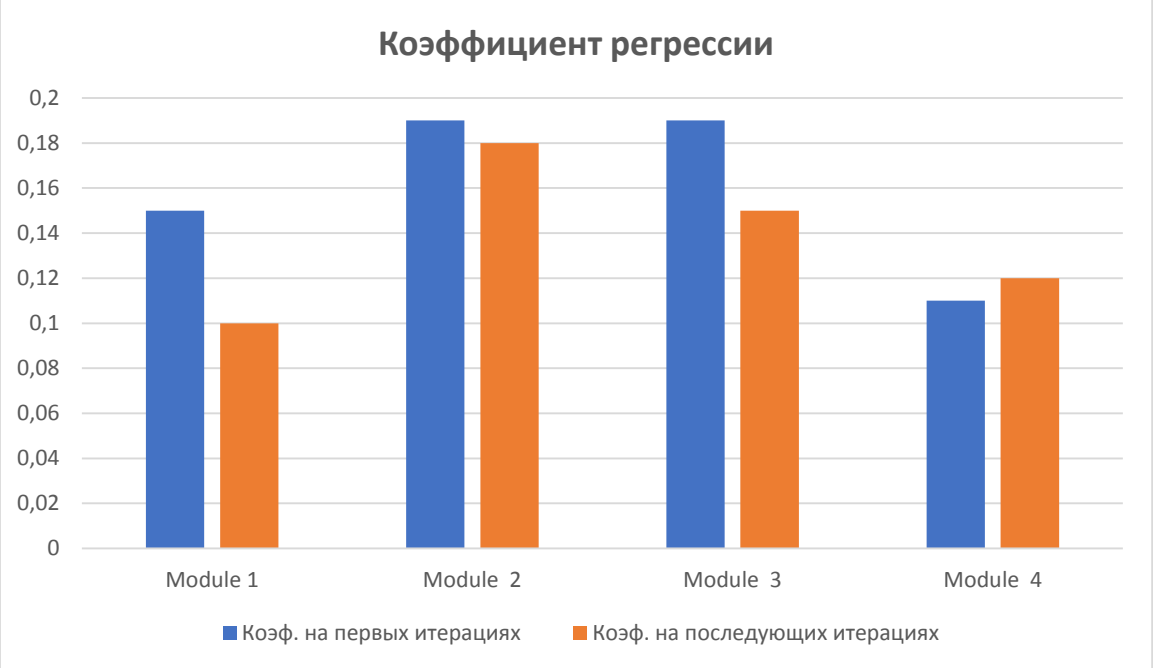

Рисунок 20 - Коэффициент регрессии

3.5.3 Скорость обработки

Достаточно важный показатель, который так же помогает определить эффективность методики, плюс к тому помогает менеджменту численно выразить возможности, скорость работы команды для дальнейшего планирования объема работ и анализа трендов развития. Метрика позволяет следить за скоростью работы QA, наблюдать за тем, какие внутренние или внешние воздействия на команду влияют на эту скорость.

Таблица 9 – Метрика скорости обработки

Модуль	Прошлый проект	Новый проект
Module 1	10	15.5
Module 2	5	6
Module 3	5	9
Module 4	1.5	2.5

На графике наглядно видно, что скорость обработки атомарных сценариев увеличилась для QA команды практически на 50% для всех модулей, что безусловно положительно складывается на общей производительности тестирования.

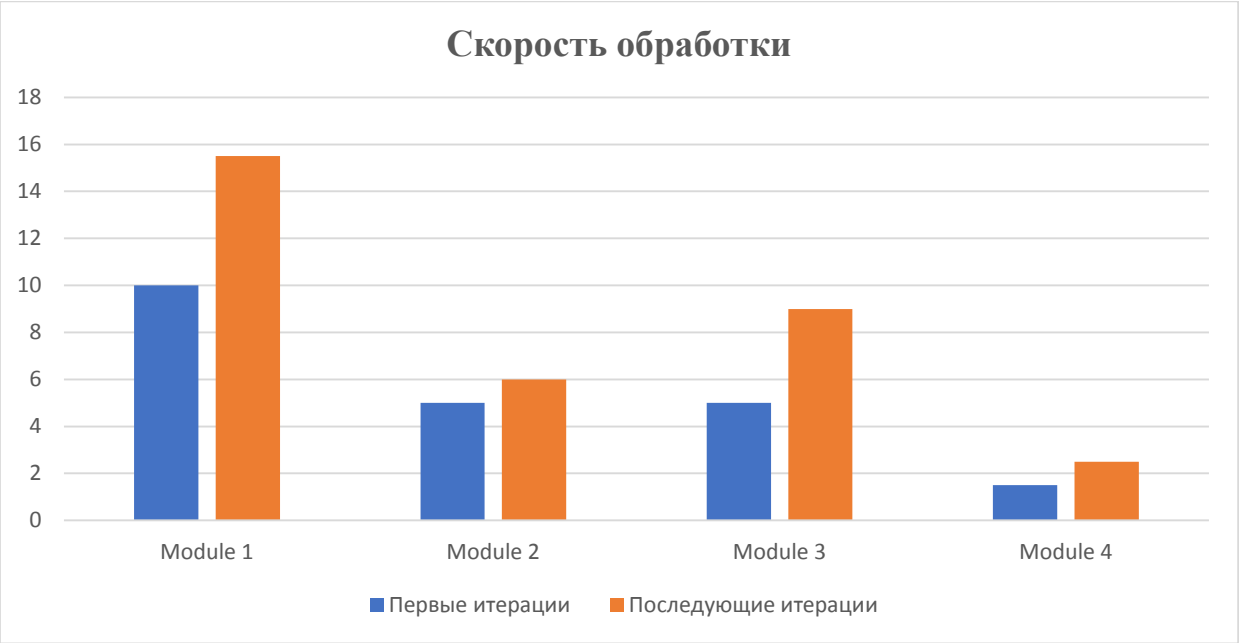

Рисунок 21 – Скорость обработки

Выводы по третьей главе

1) В третьей главе был более подробно рассмотрен более процесс предложенного алгоритма, что позволило глубже понять процесс миграции данных и дальнейшей проверки данных на новом окружении. Из чего можно наблюдать достаточно гибкий подход, но в тоже время доработанный под конкретные требования рабочей системы.
2) Задействованный подход для проверки миграции показал свою эффективность на рабочей системе, доказал наличие необходимой гибкости для решения срочных и глобальных задач, включая тестирование миграции данных.
3) Анализ результатов работы с предложенным алгоритмом показал его, в некоторых случаях, многократную эффективность перед предшествующим подходом, что положительно сказалось на качестве продукта продуктивности в работе команды.

Заключение

В процессе работы над выпускной диссертацией был проведен анализ исследований по теме миграции баз данных, рассмотрены различные подходы к выполнению задач. Так же учтены базово встречающиеся проблем и недостатки техник. По итогам было установлено, существуют самые различные техники решения задачи тестирования миграции баз данных, но каждая из них имеет как свои недостатки, так и достоинства, но в тоже время не оказалось возможным выбрать существующие решения

Для достижения поставленной цели исследования (данной работы является разработка и внедрение модели эффективной проверки мигрированных данных.) были решены все поставленные задачи, а именно:

1) Изучить исследования специалистов, связанные с миграцией данных.
2) Произвести обзор и анализ существующих моделей\методик.
3) Предложить алгоритм тестирования миграции данных
4) Разработать модель тестирования миграции данных, которая обеспечит высокую производительность и качеств ПО.
5) Произвести оценку эффективности разработанной модели и доказать гипотезу.

Из чего можно сделать вывод о том, что цель исследования была достигнута.

Таким образом, можно сделать вывод о том, что цель исследования была достигнута.

Основной научный результат магистерской диссертации заключается в том, что реализованный в ходе исследования алгоритм проверки мигрированных данных показал свою высокую эффективность, в некоторых случаях в разы превосходя предшествующий ему подход по результатам.

Список используемой литературы

1. Akond Ashfaque Ur Rahman et al. “Synthesizing continuous deployment practices used in software development.” In: Agile Conference (AGILE), 2015. IEEE. 2015, pp. 1–10.
2. Amir Hassan Bahmani, Mahmoud Naghibzadeh, and Behnam Bahmani. “Automatic database normalization and primary key generation.” In: 2008 Canadian Conference on Electrical and Computer Engineering. IEEE. 2008, pp. 000011–000016.
3. Barry E. Jacobs. “On database logic.” In: Journal of the ACM (JACM) 29.2 (1982), pp. 310–332. [10] Henryk Rybiński. “On first-order-logic databases.” In: ACM Transactions on Database Systems (TODS) 12.3 (1987), pp. 325–349.
4. Ben Shneiderman and Glenn Thomas. “An architecture for automatic relational database system conversion.” In: ACM Trans. Database Syst. 7.2 (1982), pp. 235–257.
5. C. Burry and D. Mancusi, “How to plan for data migration,” 2004.
6. Carlo Curino et al. “Automating the database schema evolution process.” In: The VLDB Journal—The International Journal on Very Large Data Bases 22.1 (2013), pp. 73–98.
7. Edgar F Codd. “A relational model of data for large shared data banks.” In: Communications of the ACM 13.6 (1970), pp. 377–387.
8. Endava, “Data Migration - The Endava Approach,” London, United Kingdom, p. 11, 2007.
9. Florian Matthes, Christopher Schulz, Klaus Haller, “Testing and Quality Assuarance in data migration projects,” 2011 27th IEEE International Conference on Software Maintenanace(ICSM).
10. Golubchik L., Khanna S., Khuller S., Thurimella R. and Zhu A. Approximation Algorithms for Data Placement on Parallel Disks // Proc. of ACM-SIAM SODA, 2000.

11. Golubchik L., Khuller S., Kim Y. A., Shargorodskaya S., Wan Y. Data Migration on Parallel Disks: Algorithms and Evaluation // Algorithmica. – 2006. – №45(1). – P. 137–158.
12. H.D. Foster, A.C. Krolnik, and D.J. Lacey. Assertion-Based Design. Springer US, 2006.. [19] Alan Turing. “Checking a large routine.” In: The early British computer conferences. MIT Press. 1989, pp. 70–72. [20] Charles Antony Richard Hoare. “An axiomatic basis for computer programming.” In: Communications of the ACM 12.10 (1969), pp. 576– 580.
13. Hall J., Hrtline J., Karlin A., Saia J., Wilkes J. On Algorithms for Efficient Data Migration // ACM Symposium on Discrete Algorithms. – 2001. – P. 620–629.
14. Haller K. Data Migration Project Management and Standard Software – Experiences in Avaloq Implementation Projects // Proceedings of the DW2008 Conference, St. Gallen, Switzerland, 2008.
15. Hudicka J. The Complete Data Migration Methodology // Dulcian Inc., June.
16. IBM, “Best practices for data migration - Methodologies for assessing, planning, moving and validating data migration,” Somers, NY, USA, p. 16, 2009.
17. Javier Tuya et al. “A Controlled Experiment on White-box Database Testing.” In: SIGSOFT Softw. Eng. Notes 33.1 (Jan. 2008).
18. Jez Humble and David Farley. Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation. Pearson Education, 2010.
19. Joachim Biskup. “Achievements of relational database schema design theory revisited.” In: International Workshop on Semantics in Databases. Springer. 1995, pp. 29–54.
20. John F Roddick. “A survey of schema versioning issues for database systems.” In: Information and Software Technology 37.7 (1995), pp. 383–393.

21. John Hess, “Dealing With Missing Values in The Data Warehouse” A Report of Stonebridge Technologies, Inc-1998.
22. Katarina Grolinger and Miriam AM Capretz. “A unit test approach for database schema evolution.” In: Information and Software Technology 53.2 (2011), pp. 159–170.
23. Khuller S., Kim Y. A. Algorithms for Data Migration with Cloning // SIAM Journal on Computing. – 2004. – Vol. 33, №2. – P. 448–461.
24. Maatuk A. Migrating Relational Databases into Object-Based and XML Databases // Doctoral thesis, Northumbria University, 2009.
25. Manik Jain. “Automated Liquibase Generator And Validator(ALGV).” In: International Journal of Scientific & Technology Research 4 (Sept. 2015), pp. 248–256.
26. Manjunath T N, Ravindra S Hegadi and Archana R A, “A study on sampling techniques for data testing”, International Journal of Computer Science and Communication, Vol. 3, No. 1, January-June 2012, pp. 13-16.
27. Manjunath T. N., Ravindra S. Hegadi, Mohan H. S., ”Automated Data Validation for Data Migration Security”, IJCA Online.
28. Manjunath T., Ravindra S.,Mohan H. Automated Data Validation for Data Migration Security // International Journal of Computer Applications (0975 – 8887)Volume 30– No.6, September 2011.
29. Matthes F., Schulz C., Haller K. Testing & quality assurance in data migration projects // Software Maintenance (ICSM), 2011 27th IEEE International Conference.
30. Michael de Jong and Arie van Deursen. “Continuous deployment and schema evolution in SQL databases.” In: 2015 IEEE/ACM 3rd International Workshop on Release Engineering. IEEE. 2015, pp. 16–19.
31. Michael de Jong, Arie van Deursen, and Anthony Cleve. “Zero-downtime SQL database schema evolution for continuous deployment.” In: 2017 IEEE/ACM 39th International Conference on Software Engineering:

Software Engineering in Practice Track (ICSE-SEIP). IEEE. 2017, pp. 143–152.

32. P. Howard and C. Potter, “Data migration in the global 2000 - research, forecasts and survey results,” London, United Kingdom, p. 29, 2007.
33. Paolo Guagliardo and Leonid Libkin. “A formal semantics of SQL queries, its validation, and applications.” In: Proceedings of the VLDB Endowment 11.1 (2017), pp. 27–39.
34. Paul Jorgensen. Software testing : a craftman’s approach. eng. 2. ed.. Boca Raton, FL: CRC Press, 2002.
35. Radoslaw Dziadosz et al. “Liquibase: Version Control for Database Schema.” In: Jyväskylän ammattikorkeakoulu (2017).
36. Sagar Khandelwal, Kannan Subramanian and Rohit Garg, “Next Generation Cross Technology Test Data Solution for M&A”, 2011 27th IEEE International Conference on Software Maintenance (ICSM).
37. Scott W. Ambler and Pramod J. Sadalage. Refactoring Databases: Evolutionary Database Design (paperback). 2006.
38. Sirkka-Liisa Jämsä-Jounela. “Future trends in process automation.” In: IFAC Proceedings Volumes 40.1 (2007), pp. 1–10.
39. Wikipedia contributors. SQL. [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/SQL (дата обращения: 03.02.2020).
40. Зайцев К.М. Методы миграции корпоративных данных // Материалы научного электронного журнала «Меридиан». - Выпуск №11(45) - 2020.
41. ИНТУИТ. Интернет университет информационных технологий. [Электронный ресурс]. – Режим доступа: http://intuit.ru (дата обращения 18.03.2020).
42. Королева Ю.А., Маслова В.О., Козлов В.К. Разработка концепции миграции данных между реляционными и нереляционными системами БД // Программные продукты и системы. 2019. Т. 32. № 1. С. 63–67.
43. Нефёдова И.В. Тестирование кода при переносе базы данных// Системные технологии. – 2016. – 3(20). – N. 96–100.

44. Петров Д.Л. Оптимальный алгоритм миграции данных в масштабируемых облачных хранилищах. [Электронный ресурс]. – Режим доступа: https://cyberleninka.ru/article/n/optimalnyy-algoritm-migratsii-dannyh-v-masshtabiruemyh-oblachnyh-hranilischah/viewer (дата обращения: 03.02.2020).
45. Петров Д.Л., Татаринов Ю. Data migration in the scalable storage cloud // IEEE, International Conference on Ultra Modern Telecommunications, ICUMT, St. Petersburg, 2009.
46. Хабр - ресурс для IT-специалистов. [Электронный ресурс]. – Режим доступа: https://habr.com/hub/api/ (дата обращения 02.04.2019).
47. Чернова Е.В. Миграция базы данных. [Электронный ресурс]. – Режим доступа: https://cyberleninka.ru/article/n/migratsiya-bazy-dannyh/viewer (дата обращения 02.01.2020).