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

Кафедра «Прикладная математика и информатика»
(наименование)

09.04.03 Прикладная информатика
(код и наименование направления подготовки)

Управление корпоративными информационными процессами
(направленность (профиль))

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ)

на тему «Методы и технологии хранения корпоративной информации малого предприятия»

Обучающийся А.В. Коновалов
(Инициалы Фамилия) (Личная подпись)

Научный руководитель канд. педагогических наук, доцент, О.В. Оськина
(ученая степень (при наличии), ученое звание (при наличии), Инициалы, Фамилия)

Тольятти 2024

Содержание

Введение

В нашем непрерывно меняющемся мире мы ежедневно принимаем множество решений на основании предыдущего опыта. Мозг человека с рождения имеет безграничную способность хранить информацию о прошлых событиях и использует воспоминания, когда сталкивается с необходимостью принятия решений. Компании, как и люди, проходя свой путь, генерируют и собирают множество данных, которые необходимо сохранить и использовать для принятия более сознательных решений в будущем.

Наш мозг может и обрабатывать, и хранить информацию, а компаниям для работы с данными требуется множество разных инструментов. Малые предприятия и организации различных отраслей экономики, развиваясь, имеют постоянный прирост информации и сталкиваются с проблемой управления нарастающих данных [18].

Ежедневно специалисты работают над решением разнообразных задач по обработке и хранению больших объемов информации, контролю доступа к ней, соблюдению правил поддержки бизнес-аналитики и обеспечению непрерывности работы с данными. При этом структура хранения информации должна обеспечивать эффективное использование ресурсов, быть гибкой и способной снижать расходы на управление данными.

Для любой компании данные являются ключевым элементом её коммерческого успеха. Информация оценивается как наиболее ценный актив компании. В рамках жизненного цикла компания стремится к обеспечению достоверной и своевременной информации. Поэтому разработка новых методов и технологий хранения данных остается актуальной задачей.

Интерес представляют исследования, направленные на разработку методов и технологий создания хранилищ корпоративных данных, отвечающих оптимальным техническим и экономическим показателям на небольших предприятиях.

Объектом исследования являются методы и технологии хранения корпоративной информации.

Предмет исследования – методы создания корпоративного хранилища данных и типы архитектур хранения данных.

Целью работы является создание модели корпоративного хранилища данных, архитектурный подход к построению которого, эффективен для малого предприятия.

Для достижения поставленной цели нужно решить следующие задачи:

- проанализировать современные методы технологии хранения данных;
- проанализировать архитектурные и концептуальные подходы к построению хранилища данных;
- построить архитектурную модель хранилища данных применимого для малого предприятия;
- оценить эффективность применения архитектурной модели хранилища данных на малом предприятии.

Гипотеза исследования: состоит в предположении, что применение грамотно разработанной архитектуры хранилища корпоративной информации существенно повышает эффективность управления данными и соответственно эффективность работы бизнеса в целом. Архитектура хранилища данных является ключевым элементом успешной деятельности любой компании, помогая ей оптимизировать процессы, принимать обоснованные решения и быть конкурентоспособной на рынке.

Методы исследования: системный анализ, моделирования, аналогии.

Научная новизна исследования заключается в разработке гибридной модели архитектуры логической схемы базы данных в хранилище, перспективной для малых предприятий, позволившая обеспечить устойчивость, гибкость, масштабируемость и четкую логику моделирования

данных, что явилось фундаментом хранения информации, существенно уменьшая риски при создании хранилища данных.

Теоретической основой исследования являются научные труды российских и зарубежных ученых, занимающихся проблемами хранения информации. Мировые разработки в сфере хранения и управления данными.

Основные этапы исследования: исследование проводилось с 2022 по 2024 год в несколько этапов.

На первом этапе (констатирующем этапе) – формулировалась тема исследования, выполнялся сбор информации по теме исследования из различных источников, проводилась формулировка гипотезы, определялись постановка цели, задач, предмета исследования, объекта исследования и выполнялось определение проблематики данного исследования.

Второй этап (поисковый этап) – в ходе проведения данного этапа осуществлялся анализ методов и технологий хранения данных. Формирование комплексного подхода и метода хранения данных, выбор архитектурных решений развертывания хранилища данных, применимых в малом предприятии.

Третий этап (реализация) – на данном этапе выполнялась оценка эффективности выбранного метода хранения данных применимо к структурам малого предприятия.

Раздел 1 Современное состояние проблемы хранения информации на малом предприятии

1.1 Современные тенденции в области хранения данных

Малые предприятия аккумулируют различные данные, полученные в бизнес-процессе. Обычно данные поступают из разных источников – структурированных и неструктурированных. Деятельность предприятия связана с различными ситуациями, поэтому данные иногда снимаются в режиме реального времени, а иногда доступны за строго определенные периоды. Все это многообразие необходимо хранить в структурированном виде, чтобы потом его можно было успешно проанализировать, распечатать адекватные отчеты и своевременно заметить изменения и аномалии. Для этих целей конструируют хранилище данных [38].

Хранилище данных – представляет собой логическая и физическая сущность, где сохраняются и хранятся все поступающие и образующиеся данные в организации, некая форма централизованного хранения всей информации бизнеса. Источниками данных могут быть являются различные системы ERP или CRM, таблицы и базы данных с физическими записями и другие неструктурированные файлы.

Хранилище данных требуется для управления жизнедеятельности существующей базы данных и всего предприятия в целом.

Допустим, на предприятии отдельно имеется программа с бухгалтерским учётом, например 1С в которой ведутся все расходы зарплата, аренда и другие хозяйственный расходы. У предприятия возникла задача построить отчётность, которая будет показывать прибыль помесячно, то есть все доходы минус расходы. Информация по доходам находится базе, а расходы в бухгалтерской программе 1С, соответственно для того чтобы построить этот отчёт нужно свести эти данные в одну систему. Данная задача решается созданием отдельной новой системы хранилища данных куда будет

загружаться все необходимые данные из разных систем которые нужны для построения отчётности. При создании хранилища будет решаться две задачи.

Первая задача - это чтобы на предприятии была одна система, где есть все данные, которые нужны, чтобы составлять нужные отчёты, а вторая задача – это минимизировать нагрузку на существующую базу данных источников. Например, если предприятие имеет большую сеть своих представителей и туда был бы постоянный поток новых записей и добавление этих записей должно быть очень быстрым. Если мы начнём строить аналитические отчеты прямо в той самой базе, то это очень сильно затормозит её, потому что аналитические отчеты, как правило, используют большие объёмы данных. Если строить отчет за большой период (год) у нас может быть большое количество строк запрашиваемых из базы данных, по которым происходят расчеты и вычисления. Это очень сильно тормозит систему. Чтобы этого избежать при построении аналитических отчётов эту нагрузку следует вынести на хранилище данных. Более того хранилище данных строится на специальных системах именно для таких аналитических отчётов, эти системы выполняют запросы быстрее [1].

1.2 Анализ построения корпоративного хранилища данных

Существует множество инструментов, используемых для создания корпоративной платформы хранения данных. Рассмотрим каждый из компонентов и его функции.

1.2.1 Компоненты корпоративного хранилища данных

Источники данных – является необработанная информация с внешних источников, которая к нам приходят при перемещении в хранилище. Данные принимаются в виде различных других форм информации, которые отправляются в файловые таблицы, базы данных и хранилища.

Слой передачи данных – несет за собой действие, связанное с передачей данных из источников применяется два взаимозаменяемых метода

для осуществления извлечения и доставки данных в хранилище. Данные инструменты называются ETL (извлечение, трансформация, загрузка) или ELT (извлечение загрузка, трансформация) в зависимости от выбора метода происходит подключение ко всем имеющимся источникам данных и их загрузка в хранилище, а так же их миграция в существующей системе хранилища данных. Существенным отличием двух методов является этап преобразования данных, который происходит до (ETL) загрузки в хранилище либо непосредственно в хранилище после загрузки (ELT) [8].

Преобразовательный этап – на данном этапе происходит упорядочивание потупившей информации, данные подвергаются очистке, в которой избавляются от дублирующих данных, происходит выстраивание порядка. Основная цель данного этапа – это привести поступающее данные к единому формату для последующей совместимостью с построенной архитектурой с имеющимся хранилищем данных.

Центральный слой хранилища – в данном слое загружающиеся данные подвергаются процедуре изменения формата, под существующую СУБД установленной в этом хранилище. Данные загружаются и хранятся по подготовленной модели. Возможно, хранение на этом уровне данные представляются в виде реляционной базы данных без какого-либо контроля или переключения. В то же время в репозитории создается раздел для хранения метаданных полученной информации

Отсек метаданных – несет за собой действие, связанное с хранением информация о данных, описывающая предметную область, техническую информацию и бизнес информацию хранящихся данных. Метаданные хранятся в отдельном отсеке и подвержены управлению отдельной системой. Для сложных структур хранения данных применяется внешний слой управления метаданных, созданный поверх всей системы хранения, который имеет возможность визуализации для простоты использования.

Слой витрины данных. В данном слое формируется отдельно для каждого требуемого вывода информации так называемая витрина. Она

представляет собой структуру, разделенную на отдельные разделы запрашиваемой информации, которая впоследствии имеет визуализацию в слое представления. Архитектура слоя создана и подогнана на отдельную тематику запрашиваемых отчетов и запросов.

Слой представления – в этот слой данные извлекаются. Применяется для извлечения информации в виде запросов представленных в визуальном виде при помощи специальных инструментов. С помощью него решаются задачи запроса и предоставления данных из хранилища в виде визуализации данных, выборки требуемых срезов и отчетов, машинное обучение, machine learning и анализ данных на требуемом уровне.

1.2.2 Концепции и функции корпоративного хранилища данных

Хранилище данных – есть сущность, при которой его можно считать базой данных, которая подключается отдельно от данных к сторонним источникам данных, которые поступают в нее при помощи специализированных инструментов и вывод уже требуемой информации при помощи аналитических средств, применяемых в данном хранилище. Хранилище данных позволяет использование его пользователями по своей предварительно согласованной концепции. На архитектурно уровне выстраивается требуемая модель для работы хранилища. Имеется возможность разделения хранилища на группы и подгруппы отдельных баз банных, что предоставляет удобство и функционал при осуществлении запросов. Обычная база данных уступает построенному хранилищу данных производительности запросов ввиду своей монолитности в архитектуре. Используя выставленную концепцию и функционал при разработке хранилища данных, мы получаем наиболее технически развитой продукт.

За определяющую основу при проектировании хранилища данных берется не только физический размер хранилища, но и сами данные, модели их преобразования и хранения.

Хранилище данных, имея свои функции, возможности и правила предполагает его использование для конкретного предприятия. Это

позволяет небольшой компании с бизнес моделью, построенной на росте и развитии данных, создавать собственную методологию хранения.

Несмотря на разнообразие возможностей, базовые концепции и функции составляют основу любого репозитория. Выделим столбцы, определяющие хранение как техническое событие:

- универсальность в хранении данных. Хранилище данных является основным местом хранения информации в организации. Вся поступающая информация из внешних источников сохраняется в хранилище и используется в корпоративном контексте;
- подготовленность поступающих данных. Данные поступающие в организацию поступают с разнообразных источников. Они не имеют одного формата и привержены определенным системам. Управление и использование поступающей информации предоставляет сложность. Хранилище данных формирует вокруг себя инфраструктуру, позволяющую привести поступающую из источников информацию к единым форматам, до поступления данных в хранилище. Соответственно постоянно поступающий поток информации данных в организации преобразуется к определенному формату для выбранного хранилища;
- сохраняемость отредактированной информации. Хранящаяся информация в хранилище данных имеет постоянно преобразованный и стандартизированный вид. Данная функция позволяет организации производить запросы и выводить отчеты по выбранной форме на данном хранилище. Структурированные и стандартизированные данные в хранилище позволяю с высокой оперативностью их использовать не тратя время на дополнительные операции с ними. Данное преимущественно характерно в сравнением с обычным озером данных, где большой объём информация хранится в хаотичном не упорядоченном виде, что усложняет аналитические и технические действия над ними. Хранимые данные представлены в

хранилище, как стандартизированы и структурированы. Данное представление ведет за собой возможность разрешить конечным пользователям отправлять запросы через интерфейс storefront и формы отчетов. В этом разница между хранилищем данных и озером данных. Озеро данных для хранения больших объемов «сырых» неструктурированных данных в целях анализа.

– ориентированность данных. Назначением базы данных является деловая информация принадлежащая к разным предметным областям. Корпоративная информация организации подразделяется на сферы и предметные области. Для успешной работы с информацией данные выстраиваются на отдельные модели. Так как имеется в хранилище отдельное подразделение хранящихся метаданных имеющейся информации, дающих понятие о том что из себя представляет конкретная данные или группа данных, их тип, информацию от первоисточника и их принадлежность последующему выводу. Функция ориентированности данных при хранении в хранилище позволяет понять к какому конкретному бизнес процессу организации они относятся, структурируя по предметной области их применения [11].
– историчность данных. Бесперебойно поступающая информация в хранилище данных характеризуются событием их поступления, преобразования, сохранения. Данные события имеют историчность, которая показывает когда, как и какой временной период происходили изменения. Разделение на временные отрезки благотворно воздействуют на работу с хранящимися данными.
– постоянство данных. Хранящаяся информация в хранилище данных находится состоянии хранения постоянно, ее существование не нарушается. Она не удаляется и не заменяется. В процессе хранения в хранилище информация может только изменятся и обновляется. Фактор постоянства данных отражает единую сущность

построенного хранилища данных как органичную систему, жизнедеятельность которой непосредственно связана со всеми ее компонентами. Однако устаревшая и не актуальная информация, которая не принимает участие в деятельности хранилища, является избыточной нагрузкой и требует вмешательство для ее удаления.

1.2.3 Типы реализаций корпоративного хранилища данных

Организовывая хранилище данных на предприятии, на начальном этапе имеется возможность выбора в плане проектирования и технической составляющей, для дальнейшего правильного функционирования хранилища. В каждом бизнесе имеется своя специфика и нюансы, поэтому хранения данных должно быть подобрано и интегрировано для конкретного предприятия. Для этого в настоящее время имеется не менее трех методов развертывания хранилища. В зависимости от размера предприятия, потребностей, бизнес аналитики, объема хранения информации, степени требуемой безопасности данных и выделяемых средств, каждое предприятие делает выбор как ему строить свое хранилище данных .

Собственное хранилище данных предприятия.

На предприятиях самым простым и первоочередным способом хранения информации это хранение данных на собственных мощностях. Для этого используются собственное помещение, оборудованное под хранилище данных. В нем присутствуют сервера предприятия купленное или спроектированное программное обеспечение и вся соответствующая инфраструктура, требуемая для оперативной деятельности хранилища. Данный вариант можно назвать классическим и имеет свое начало при образовании предприятия. Хранение собственной информации на физических носителях и у себя на предприятии дает возможность моделировать создаваемые базы данных, не применяя отдельных методов интеграции и модуляции между ними. Поставка информации в хранилище из источников происходит бесперебойно, а изменение самих данных происходит либо на этапе извлечения или уже после попадания в самом

хранилище при помощи созданных инструментов и подготовленной архитектуры хранения. Конечно, хранения данных на собственных носителях и используя свою инфраструктуру, является более удобным для управления, имея собственных подготовленный инженерный состав сотрудников, работающих с данными предприятия. Инженерам существенно проще работать с данными в виду отсутствия уровня абстракции на промежуточном этапе. Поток данных от источника к хранилищу происходит напрямую, а формирование отчетности при желании возможно на любом этапе загрузки данных и работы хранилища.

Все же имеются недостатки данного типа реализации хранения информации на предприятии и зависят от конкретной ситуации, возможностей бизнеса и стороннего влияния. К основным недостаткам относится:

- дороговизна создание и эксплуатация хранилища данных. Требуется вложения в покупку собственного оборудования, программного обеспечения и технологического оснащение. Так же существенные расходы несутся на эксплуатацию всей инфраструктуры хранилища данных;
- требуется иметь в штате предприятия собственную команду инженеров и других специалистов осуществляющие производственную и контрольную деятельность на созданном хранилище. К тому же не малые средства потребуются на создание и поддержание всей инфраструктуры хранения информации.

Классический вариант хранения данных имеет место реализации на предприятии любого размера, уровня и типа. Основное преимущество данной модели это возможность максимально обезопасить свои данные, управляя и изменяя их в процессе жизнедеятельности предприятия. Сохраняя высокую степень защиты конфиденциальности корпоративной информации при использовании хранилища данных, реализованного по классической

схеме, оно может быть модифицировано или даже перестроено в иную форму хранения данных.

Виртуализация хранилища данных предприятия.

Хранилище данных в виртуальном виде используется при классическом построении хранилища, но применяя технологии виртуализации данных. При помощи виртуализации базы данных, созданные отдельно, могут взаимодействовать в целом с помощью виртуализации. Тип реализации виртуального хранилища данных упрощает работу и запросы, позволяя физически хранить данные в источнике без перемещения или копирования. Используется определенный инструмент такого типа для доступа к данным в виртуальном хранилище, не затрагивая физическое расположение информации. Это очень удобно, где использовать данные в не изменённом виде из источника. Использование виртуального хранилища упрощает управление всей инфраструктурой хранилища данных на предприятии [41].

Однако данный подход имеет много недостатков:

- регулярное техническое обслуживание программного обеспечения виртуальных баз данных. Вынужденные расходы на аппаратное обеспечение;
- требуется постоянное преобразование данных находящихся в хранилище для последующей передаче их при запросе конечному потребителю и аналитической отчетности. Данная работа моет быть проделана дополнительным программным обеспечением, что также влияет на затраты при содержании хранилища данных;
- учитывая, что хранилище разделено на отдельные базы данных находящиеся физически в разных местах и виртуализированными в хранилище, запросы информации из базы данных будет занимать значительное количество времени, т.к. отдельные элементы информации могут располагаться в разных источниках или базах данных. В данном случае производительность вывода информации

зависит от качественно подготовленного программного запроса и технического оборудования, что в конечном итоге приведет к дополнительным расходам.

На рисунке 1 представлена схема связей между абстракцией виртуального хранилища данных и исходными базами данных.

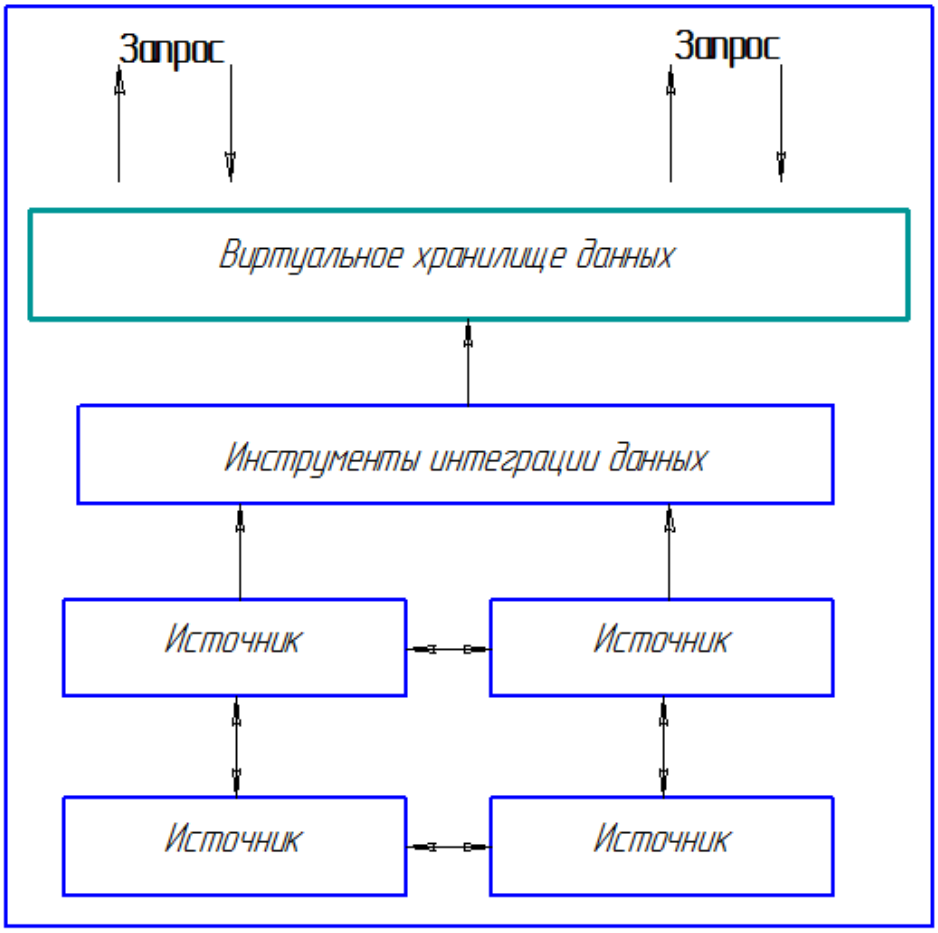

Рисунок 1 – Схема связей между абстракцией виртуального хранилища данных и исходными базами данных

Этот тип предложенного хранилища всецело приносит пользу для компаний со стандартными базами данных, которые не требуют сложных аналитических запросов для своей деятельности. Он также хорошо применим на старт тапах, малых предприятий и организаций, которые не используют бизнес-аналитику на регулярной основе. Для тех кто, возможно, желает использовать ее в будущем [14].

Хранение данных предприятия на облачных ресурсах.

Реализация хранилища данных в виде облачного хранилища предоставляет собой хранение данных на сторонних технологических ресурсах, с предоставлением полного спектра услуг по действию над данными в виде хранения, аналитики и сохранения конфиденциальности. Облако предоставляет собой сторонний сервер, где располагается выделенное место под базы данных хранилища предприятия. Данное место имеет возможность масштабирования, а также предоставления различных сервисов от поставщика облачного ресурса для хранения и управления информацией. Обычно облачный сервис представлен в трех ступенях. Первая это вычислительный кластерный узел производящий множество запросов над базами данных. Следующая ступень отвечает за хранение информации. Третья ступень предоставляет потребителю возможность управления над его данными, это может быть аналитические или технические действия.

В облачной структуре обслуживание инфраструктуры лежит на плечах поставщика облачных ресурсов. Предприятию выбравшему данный вид хранения своей корпоративной информации не требуется заботится о покупке и обслуживании серверов, подготовки программного и аппаратного обеспечения, сдержать штат сотрудников по управлению над данными, управлять хранилищем. Главное это определиться с требуемым объёмом выделяемых мощностей и программных услуг на облаке, толь от этого будет завесить стоимость содержания хранилища данных предприятия. Это неоспоримое преимущество данного типа реализации хранилища данных на предприятие.

Единственная проблема заключается в конфиденциальности хранящихся данных. Информация предприятия, хранящаяся на стороннем ресурсе, очень уязвима от защиты утечек и хакерских атак и не подчиняется в плане безопасности их источнику самого предприятия. В данном случае требуется правильный и доверительный подход к выбору поставщика облачных услуг.

Для небольших и малых предприятий облачное хранение будет иметь лучшую защиту в плане безопасности данных, так как крупные игроки на рынке облачных ресурсов имеют возможность предоставить современные технологии защиты данных, которые малое предприятие не может себе организовать в виду отсутствия средств или знаний. Поэтому при организации хранилища данных на облачном ресурсе следует тщательно сопоставить возможности и требования бизнеса предприятия.

При развертывании хранилища данных качестве облачной платформы хранения этот тип хранилища является отличным выбором для организаций любого размера, если вам нужны все возможности, от внедрения управления над данными до содержания хранилища данных и поддержки аналитической части [24].

1.2.4 Архитектура корпоративного хранилища данных

В настоящее время разработано огромное количество методов и структур построения хранилища данных, на архитектурном уровне функционал которых позволяет максимально благотворно использовать применимые решения в процессе пользования хранилищем. При разработке схематично можно представить процесс поступления, хранения и использования данных на три этапа процесса движения информации:

- этап интеграции данных из источника информации;
- этап хранения информации;
- этап выдачи информации.

До поступления данных в хранилище они проходят через интеграционный этап, применяется два взаимозаменяемых метода для осуществления извлечения и доставки данных в хранилище. Данные инструменты называются ETL (извлечение, трансформация, загрузка) или ELT (извлечение загрузка, трансформация) в зависимости от выбора метода происходит подключение ко всем имеющимся источникам данных и их загрузка в хранилище, а так же их миграция в существующей системе хранилища данных. Существенным отличием двух методов является этап

преобразования данных, который происходит до (ETL) загрузки в хранилище либо непосредственно в хранилище после загрузки (ELT) [28].

Инструменты загрузки и интеграции данных производят работу в процессе загрузки, в экосистеме самого хранилища и вывода данных. При помощи их происходит движение данных по слоям и заданное преобразование на каждом этапе. Поэтому определяющим при построении архитектуры хранилища данных будут факторы, связанные с использование согласованного инструментария в данном процессе. Мы проанализируем проектирование архитектуру хранилища в контексте растущего малого предприятия, делая акцент на временной период. Рассмотрим варианты изменения архитектуры хранилища при росте потребностей предприятия.

Схема архитектуры проектирования хранилища данных представлена в трех видах, увеличивающих свою структуру за счет добавления новых элементов и расширения уже имеющихся.

Вид одноуровневой архитектуры.

Самый простой архитектурный способ построения хранилища данных это создание одноуровневой структуры, где база данных хранится в центе хранилища, к которому непосредственно подключены инструменты отчетности, которые запрашивает пользователь. Данная форма проста в исполнении и использовании.

Прямая последовательность имеет ряд недостатков. Прямое соединение дает погрешность в запросах, что снижает скорость обработки данных и требует дополнительно время на корректировку. При выполнении более сложного запроса для визуализации из хранилища требуется время на интеграцию языковых запросов и сортировки нужных данных. Существуют и ограничения при построении аналитических отчетов.

Одноуровневая архитектура используется в основном на мелких предприятиях, в бизнесе которых не требуется масштабных аналитических запросов. Используя одноуровневую архитектуру, следует учесть ее непредвиденность и малую производительность при выполнении сложных

запросов. Применять данный метод следует в работе с простыми не сложными базами данных при запросе к которым не требуется высокой отдачи [3].

На рисунке 2 представлена архитектура хранилища данных в одноуровневом виде.

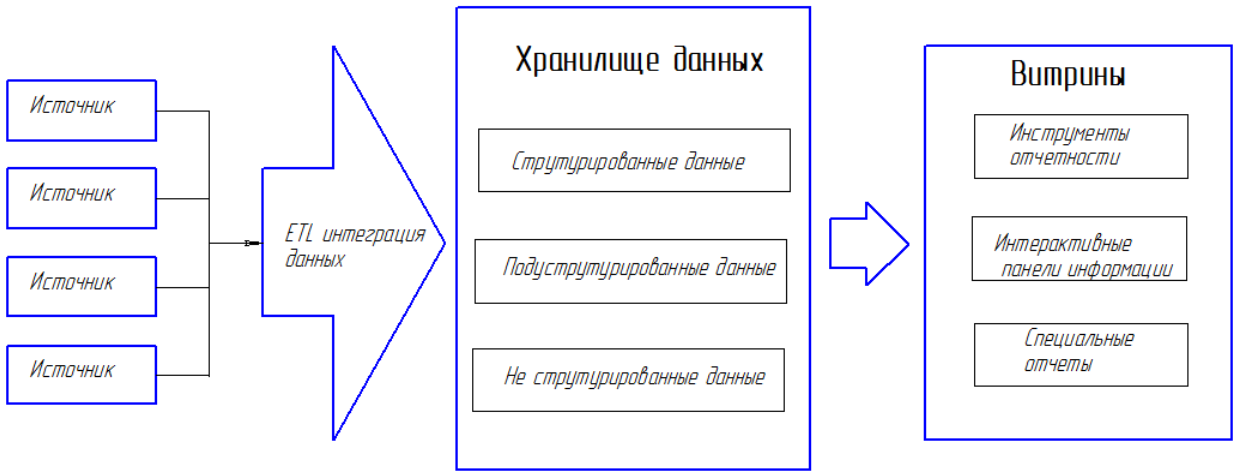

Рисунок 2 – Архитектура хранилища данных в одноуровневом виде

Вид двухуровневой архитектуры.

В двухуровневом виде архитектуры хранилища витрины данных объединяются в отдельный уровень и располагаются между центральным слоем хранилища и дополнительно созданным слоем отчетности (уровень отчетности). Слой уровня витрин данных представляет собой малую базу данных в хранилище, где хранится информация, разделённая на отдельную тематику или разделы. На рисунке 3 представлена архитектура хранилища данных в двухуровневом виде.

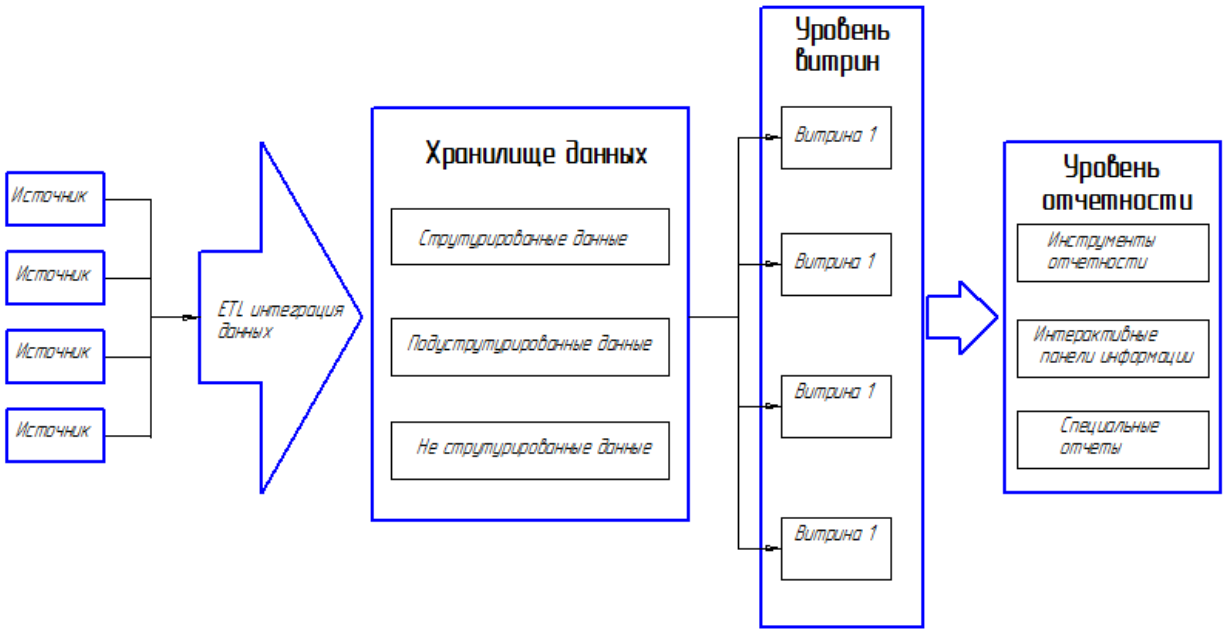

Рисунок 3 – Архитектура хранилища данных в двухуровневом виде

В данную базу данных стекаются данные специально подразделенные по структуре бизнеса предприятия для отдельных запросов. Информация распределяется при хранении на блоки, сгруппированные по областям деятельности предприятия.

Хранение данных по отдельным витринам удобно для последующего вывода при помощи инструментов отчетности. Точечный запрос данных обеспечивает более высокую производительность и предсказуемость получения данных по сравнению с одноуровневым видом.

Недостатком применения витрин данных является увеличение расходов на приобретение дополнительного оборудовании и сведение базы данных уровня витрин с другими составляющими хранилища данных. Преимущество данного подхода видится в решении задачи точечного запроса. Конкретный пользователь по своей области имеет возможность быстро выполнить интересующий его запрос и вывести его из конкретной базы, не затрагивая всю систему хранения. Создание витрин баз данных обеспечивает повышенную безопасность данных, так как имеется

ограничение к доступу для пользователей по конкретным областям бизнеса или хозяйственной деятельности предприятия [7].

Вид трёхуровневой архитектуры.

В трёхуровневый вид архитектуры хранения добавляется Куб OLAP. Он выделяется отдельным слоем связанным непосредственно с витринами данных.

Куб представляет собой отдельную базу данных, в которой информация хранится в трёхмерном измерении. По сравнению с хранение в двухмерном измерении, например таблицы Excel, данный способ имеет преимущество. Агрегируя и перемещая информацию в глубине куба, появляется возможность делать сложные срезы и минимизировать объём хранения данных. Взаимосвязи куба в хранилище позволяет получение данных, как из уровня витрин данных так и сразу из центрального хранилища.

На рисунке 4 представлена архитектура хранилища данных в трехуровневом виде.

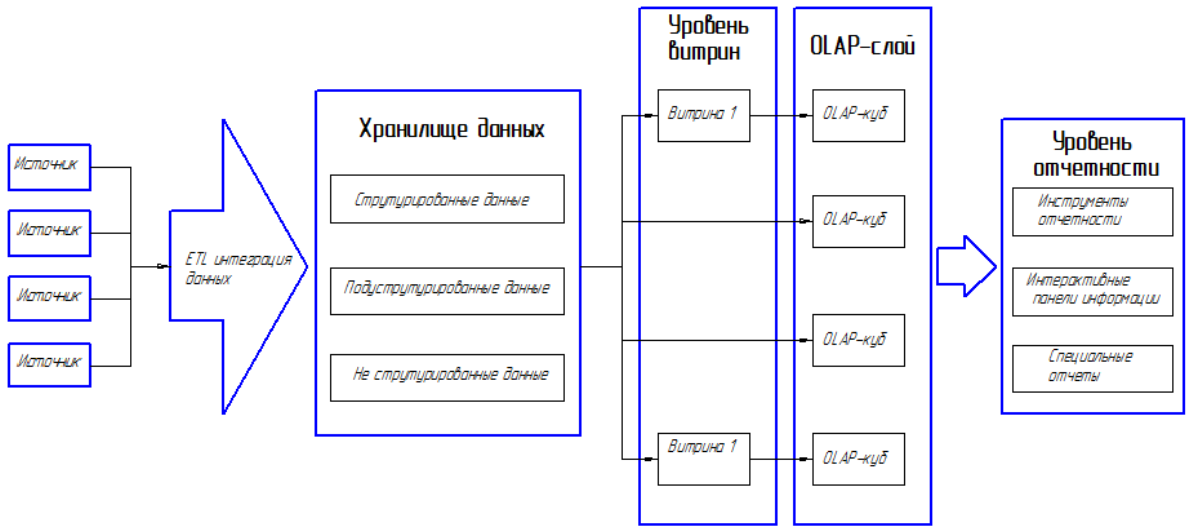

Рисунок 4 – Архитектура хранилища данных в трехуровневом виде

На рисунке 5 представлен OLAP-куб, отражающий в трехмерном измерении хранение данных.

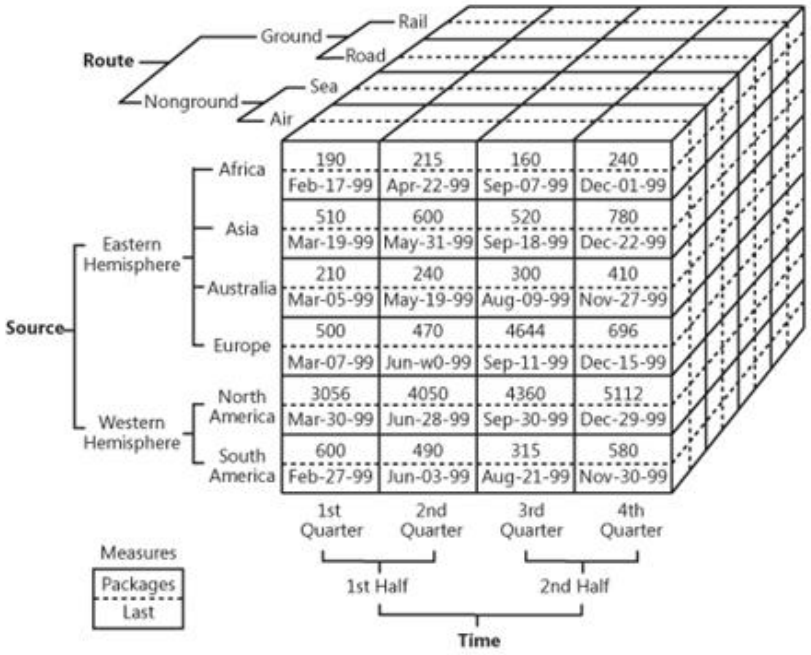

Рисунок 5 – OLAP-куб, отражающий в трехмерном измерении хранение данных

Как видно из рисунка, куб добавляет измерения к данным. Вы можете сравнить его со связанными электронными таблицами Excel. Лицевая грань куба представляет собой стандартную двухмерную таблицу с регионами, отмеченными по вертикали, и продажами по горизонтали. Многомерность данных обнаруживается, когда мы смотрим на верхнюю грань куба, где продажи разделены по маршрутам, а нижняя грань – по временным интервалам [22].

Преимущество внедрения в хранилище данных Куб OLAP как отдельный слой является широкие возможности при создании аналитический отчетов. Оптимизируя кубы под хранение информации предприятия, появляется возможность их использования со всеми составляющими

хранилища. Появляется доступ ко всей информации так и к отдельным данным на уровне витрин данных.

1.2.5 Технические компоненты хранилища данных

Определяясь с реализацией архитектуры хранилища данных, в повседневной деятельности многие не большие предприятия на начальном этапе выбирают, вместо построения хранилища данных делают выбор в пользу образования озера данных или построение витрин данных на своей базе данных. Данная альтернатива актуальна и имеет место, но не следует объединять ее в одно понятие с построением полноценного хранилища данных, поэтому уточним определения каждого.

Хранилища данных хранит информацию в виде табличной структуры. Структурированные данные по столбцам и полям удобны для осуществления запросов на вывод при использовании языковых приложений. Данные упорядочены в таблицах и имеют логическую и физическую взаимосвязь. Применяя инструменты по работе с данными позволяет оптимально быстро и корректно выводить информацию для бизнес анализа предприятия. В хранилище все настроено на структурированные данные, оно легко получает данные с многих структурированных источников разных отделов бизнеса и внешней среды. На создание полноценно работающего хранилища предприятия требуется значительно время, так как при развертывании требуется должная настройка связи с источниками, на преобразования данных в структурированный вид и взаимосвязь меду ними.

Главной отличительно чертой хранилища является это проявления его, в виде физической сущности имея логическую структуру. Управления данными на таком уровне позволяет добиться высоких результатов. Старт хранилища данных можно считать от 100 ГБ информации до любой величины.

Озёра данных хранит информацию в больших объемах в различных видах и форматах. Данные могут иметь структурированный и не структурированный вид, так как для озера данных неважно состояние

данных. В озере данных информация хранится в общей куче, не применяя группировки и подборки данных. Его можно расценить как сток стекающейся информации, к которой не применяются управленческие действия, а просто идет накопление и хранения. Поэтому структурированные данные в озере данных не будут иметь преимущества по сравнению с общей массой хранимой информации и не принесут в дальнейшем удобства ими пользоваться [34].

Все данные хранятся в беспорядочном виде, и применять структурирование данных тут не целесообразно. Накопление огромной не сортированной массы данных удобно для дальнейшего машинного обучения либо может быть применено как долгосрочный архив на случай неопределенных ситуаций, который не требует больших средств на содержание.

Можно отметить, что в последнее время озеро данных начали применять для аналитических действий. Данный метод основан на преобразовании данных в озере после загрузки. Такое домашнее озеро моет быть удобно не большим предприятиям при начале своей деятельности, когда данные нарастающем темпе поступают, а нужная информация на вывод ограничена малой бизнес деятельностью организации.

Витрины данных хранит информацию в виде баз данных разделенных в хранилище данных на отдельные виды, подразделяющееся по типу бизнес или хозяйственной деятельности предприятия. Витрина данных может использоваться как отдельная база данных настроенная на хранение информации в структуре предприятия, не создавая хранилища данных. Система хранения информации в виде витрин данных подразумевает не очень большой размер хранения порядка менее 100 ГБ. Для мелких начинающих бизнес проектов это может быть удобно, этот подход не потребует больших затрат когда как аналитический результат будет на уровне. Так же данный подход применяется для разбиения структуры

предприятия, имеющей различные базы данных, которые не имеют прямой связи между собой и не требующая общего вывода бизнес анализа.

Витрина данных как реляционная база данных оправдывает себя при поступлении данных из мало нагруженной базы структурированного источника. При соблюдении условий витрины данных довольно быстро развёртывается на предприятии, и выдают нужный результат. Витрины данных не могут использоваться крупными предприятиями в виду постоянно нарастающей информацией и большой нагрузки при формирования отчетов и анализа [36].

Хранилище данных и доступный инструментарий для его развёртывания определяет, какая дальнейшая система хранения данных на малом предприятия лучше всего подходит для его нужд.

Вывод по разделу 1

Понимание доступных инструментов корпоративного хранилища данных поможет определить, какая платформа данных для малого бизнеса лучше всего подходит для ваших нужд. Создание хранилища данных может потребовать много времени на проектировку и апробирование, потому что даже самый простой хранилище данных огромно по своей структуре и масштабу. Предприятиям для создания своего хранилища данных требует обращаться в специализированые структуры, которые подберут и разработают верные решения хранения информации. Для грамотного анализа бизнес потребностей предприятия следует, определить цели и задачи в той части бизнеса, которая непосредственно взаимодействует с реальными данными. Популярные в настоящее время облачные технологии предоставляют множество поставщиков в мире. Поставщики услуг по разработке хранения информации предоставляют различный спектр услуг на все потребности, желания и имеющиеся средства.

Проблема выбора решений по хранению корпоративной информации на малом предприятии в условиях постоянно нарастающего объема данных является актуальной и требует внимания со стороны руководства

организации. Продолжительный рост объема данных может привести к таким негативным последствиям, как увеличение времени доступа к информации, ухудшение производительности системы, потеря данных и нарушение безопасности информации.

Предполагается под решение проблемы проведение обоснованного анализа настоящей стадии хранения данных, где потребуется установить объемную составляющую, создать необходимые требования и характеристики для установления доступа, подготовить необходимые действия для обеспечения безопасности информации и резервному копированию. Результат анализа поможет выбрать наиболее эффективные решения по хранению данных [29].

Одним из вариантов решения проблемы может быть использование облачных технологий хранения данных. Облачные хранилища позволяют хранить данные на удаленных серверах, что позволяет освободить место на локальных серверах предприятия. Кроме того, облачные хранилища позволяют масштабировать хранение данных в зависимости от их объема и предоставляют возможность удаленного доступа к информации.

Другим вариантом решения может быть использование сетевого хранилища данных, представляет собой централизованную систему хранения данных, подключенную к компьютерной сети. Она позволяет предоставить доступ к данным различным пользователям внутри организации. Данное хранилище обеспечивает высокую производительность, отказоустойчивость, возможность создания резервных копий данных и защиту информации.

Также вариантом решения проблемы может создание систем хранения информации на жестких дисках собственных мощностей предприятия, обеспечивают быстрый доступ к данным, имеют большую емкость и устойчивы к физическим повреждениям. Однако, стоимость создания хранилища будет высока [12].

Любому постоянно растущему предприятию следует делать выбор исходя из своих потребностей и возможностей, а так же учитывая

меняющуюся ситуацию в мире, стоит определить свой вектор развития хранения информации нацеленный на отечественный продукт или собственные возможности. Владельцы малого бизнеса могут быть ошеломлены количеством доступных вариантов и технологий, поэтому выбор правильного подхода к созданию хранилища имеет решающее значение для определения ваших бизнес целей.

При выборе настоящих и необходимых ответов по решению проектирования и создания структуры по хранению корпоративной информации при создании хранилища данных для малого предприятия предполагается, необходимый последовательный путь, обозначенный определенными требованиями которые необходимо учитывать. Обозначим их:

- количественная составляющая данных. Первым шагом при выборе архитектуры хранилища данных является оценка объема информации, которую необходимо сохранить. Чтобы правильно выбрать систему хранения данных, вы должны определить объем информации, собираемой и хранящейся в хранилище информации;
- структурная составляющая данных. Обязательно необходимо определить структуру данных, используемую хранилищем информации. Например, если информация представлена в табличной форме и имеет жесткую схему, то реляционные базы данных могут быть самым подходящим решением. Если же данные более гибкие и меняются во времени (например, данные о клиентах или продуктах), то подойдет нереляционная база данных;
- интеграция с существующими системами. Если у предприятия уже есть системы, которые используются для хранения информации, то необходимо учесть возможность интеграции хранилища данных с такими системами. Например, если существует система учета продаж, то хорошо бы иметь возможность получать данные из нее и автоматически загружать в хранилище данных;

– инструменты аналитики. Для малого предприятия важно иметь возможность анализировать данные и получать ценную информацию для принятия решений. Поэтому следует выбрать такую архитектуру хранилища данных, которая обеспечит инструменты для аналитики и создания отчетов;
– безопасность. Важно обеспечить безопасность хранения корпоративной информации. При выборе архитектурных решений следует учесть такие аспекты, как доступ к данным, шифрование данных, механизмы аутентификации и авторизации.
– масштабируемость и гибкость. Хранилище данных должно быть масштабируемым и гибким, чтобы легко справляться с увеличением объема данных и изменением требований предприятия. Поэтому следует выбрать архитектуру хранилища данных, которая позволяет легко добавлять новые данные и вносить изменения в существующую структуру данных [17].

Итак, при выборе архитектурных решений по хранению корпоративной информации для создания хранилища данных для малого предприятия необходимо учитывать объем и структуру данных, возможность интеграции с существующими системами, инструменты аналитики, безопасность, масштабируемость и гибкость. Подходящая архитектура хранилища данных должна удовлетворять всем этим требованиям и обеспечивать эффективную работу с данными.

Раздел 2 Анализ методологий проектирования архитектуры логической схемы многомерной базы данных хранилища

При проектировании существует несколько подходов к построению архитектуры логической схемы многомерной базы данных хранилища, помогающих избежать в дальнейшем распространенных проблем.

На небольших предприятиях в процессе деятельности образуется ряд сложностей связанных с хранением и управлением данных. И при росте организации они становятся, все более актуальны. Требуется доработка имеющегося хранилища, поддержка историчности данных, возникают проблемы при сборе данных из нескольких источников. Имеющиеся архитектура не обеспечивает на должном уровне гибкость и расширяемость хранилища. Поэтому следует качественно разработать на логическом уровне схему построения хранилища, подходящего под конкретные задачи на данном предприятии. Учитывая, что малые предприятия обладают потенциальным ростом и развитием, стратегия на реализацию гибкой и масштабируемой структуры имеет приоритетное значение.

2.1 Классические архитектурные подходы к проектированию хранилища данных

Существуют различные способы реализации модели хранилища данных. К самым простым и популярным относятся схемы хранения «Звезда» и «Снежинка», представляющие в себе реляционную базу данных. Создание хранилища данных на основе реляционной модели хранения данных имеет ряд преимуществ, при хранении больших объёмов информации данных. Самое главное чтобы в базе данных не было избыточности и несогласованности.

Избыточность – это хранение одинаковой, подобной информации во множестве разделов или в разных местоположениях. Например,

несогласованность данных возникает, когда определенная информация хранится в нескольких местах. При избыточности, когда одна и та же информация хранится в нескольких местах в базе данных, процесс нормализации пытается устранить эту избыточность, чтобы в принципе не было возможности не согласованных данных. Не согласованность данных и избыточность приводит к тому, что потребляется больше места на жёстком диске. При разработке OLTP базы, как правило, всё максимально нормализовано, то в хранилище данных требует уместной для бизнеса нормализации данных.

При классическом подходе реализации архитектуры хранилища данных применяется первые три формы нормализации (1НФ, 2НФ, 3НФ). Этап нормализации применяется последовательно [25].

Измерения – данные объекты имеют имена и атрибуты, они являются частью процесса работы или же являются категориальными атрибутами, которые имеют отношение к измерениям. Данные о наименованиях товаров, имена людей, которые их производят и продают, географические названия являются измеримыми величинами. В случае, если категория (например, наименование товара) имеет числовой код, то и критерии могут быть также числовыми, но в любом случае это конкретные данные, которые представляют собой оценки из ограниченного количества. В процессе проведения измерений, изучаемый бизнес-процесс качественно описывается.

Факты – данные объекты имеют имена и атрибуты, они являются частью процесса работы или же являются категориальными атрибутами, которые имеют отношение к измерениям. Примерами таких фактов являются цена продукта или стоимость товара, количество продаж и закупок, размер заработной платы сотрудников, размер кредита, сумма страхового покрытия и другие.

Реляционная модель основывается на том, что измеряемые данные хранятся в плоском табличном массиве, а фактическая информация хранится в отдельных таблицах одного и того же базового файла. В таких случаях

таблица фактов служит основой для других таблиц измерения. В нем содержатся количественные характеристики предметов, событий и явлений, которые предполагаются в будущем для каждого из них.

В графической схеме типа «звезда», центральная таблица фактов выполняет роль связующего звена для всех таблиц измерений. Благодаря такому подходу, информация о каждом измерении благоприятно помещается в отдельную таблицу, чтобы обеспечить комфортное изображение. В результате созданная схема логической фигуры становится более читабельной, логика понятна лицу использующему ее.

На рисунке 6 представлена схема «Star schema».

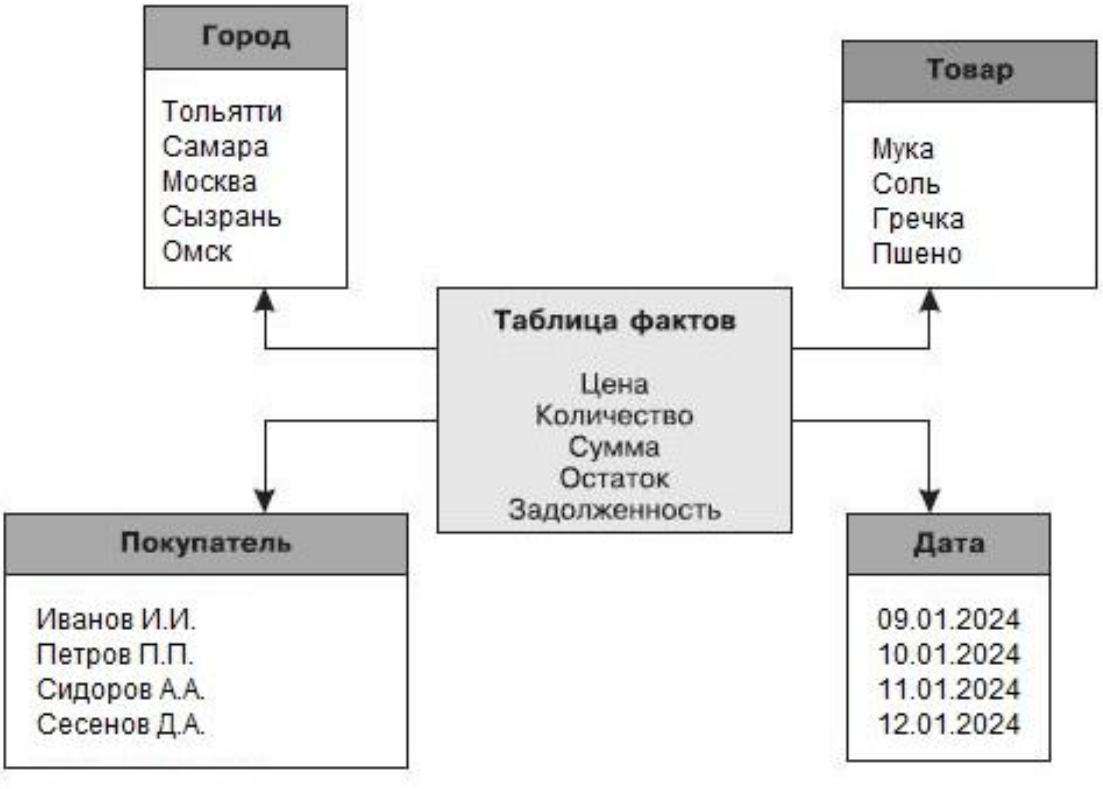

Рисунок 6 – Схема «Star schema»

Стоит отметить, что внесение всей имеющейся информации об измерениях в определенных моментах не может быть сформировано в единственной таблице. Если продаваемый товар имеет иерархию, то необходимо указать, какие товары относятся к той или иной группе, или же

использовать другие способы для обозначения групп. Это может привести к повторениям названий группы. Увеличение избыточности может привести к возникновению конфликтов, например, в случае ошибочного отнесения одного и того же товара к различным группам.

Схема «Snowflake schema» представляет собой видоизмененную в какой то мере разветвленную версию схемы Схема «Star schema» и предназначена для более эффективной работы с иерархическими измерениями данных. Ключевой особенностью этой схемы является то, что данные измерений хранятся в нескольких связанных таблицах, чтобы иерархическая структура могла быть организована более эффективно.

На схеме «Snowflake schema» наблюдается 1 или более таблиц измерений связаны как минимум с 1 другой таблицей измерений, образуя иерархическую структуру. Это позволяет лучше управлять и анализировать данные, особенно в случае использования сложных иерархических структур.

Например, в схеме «снежинка» таблица фактов может быть связана с таблицей общих данных, которая затем связана с таблицей подкатегорий, которая в свою очередь связана с таблицей категорий. Таким образом, данные на каждом уровне иерархии хранятся в отдельной таблице, и между таблицами существуют связи.

Использование схемы «снежинка» позволяет снизить избыточность и повторяемость данных, а также оптимизировать процессы анализа и отчетности. Это особенно важно при работе с большими объемами данных и сложными структурами. Однако, следует отметить, что схема «снежинка» также может быть менее производительной и сложной в реализации и поддержке, поэтому выбор между схемой «звезда» и «снежинка» зависит от конкретных требований и особенностей проекта [31].

На рисунке 7 представлена схема «Snowflake schema».

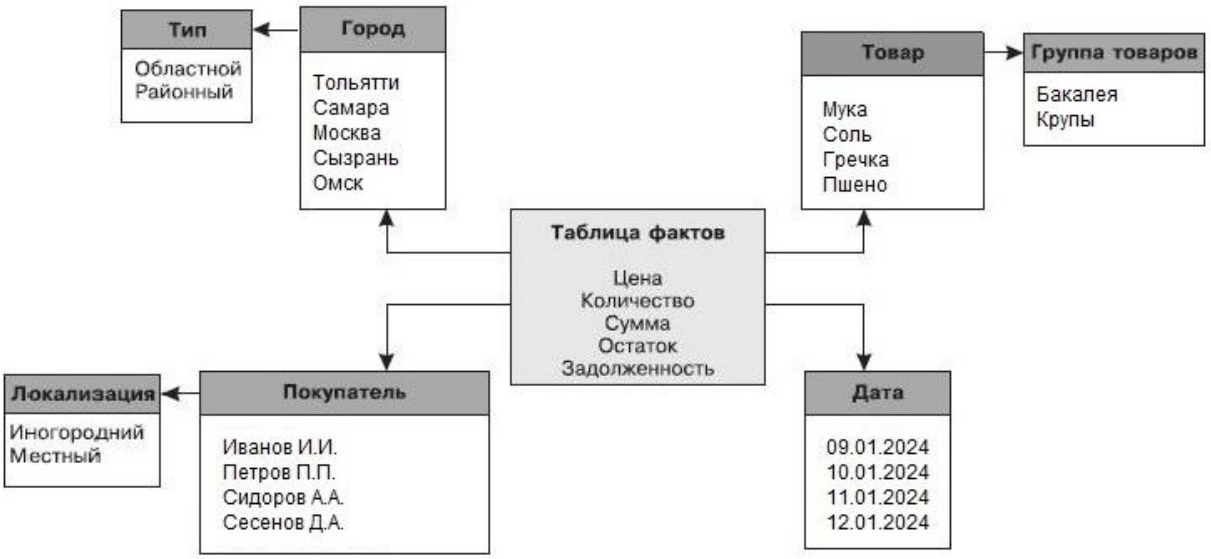

Рисунок 7 – Схема «Snowflake schema»

Основное функциональное различие между схемой "снежинка" и схемой «звезда» заключается в способе обработки иерархических данных.

Схема «снежинка» (snowflake schema) позволяет обрабатывать иерархии данных, определяющие уровень детализации. В этой модели данные о каждом товаре или группе товаров связаны с общим представлением. Преимущества схемы "снежинка" включают близкое к данным представление многомерной модели, более эффективные и простые процедуры загрузки в многомерной структуре, легкий доступ к данным. Реализация в виде применения информации в существующей многомерной модели. Недостатки данной реализации логической схемы хранения информации это затруднение в осознании структуры данных по отношении к более простым схемам реализации. Так же появляются определенные затруднения при добавлении новых поступающих измерений в процессе развития бизнеса на предприятии.

Схема «звезда» (star schema) имеет более простую и логичную модель данных. Процедура заполнения измерений проще, поскольку необходимо обрабатывать только одну таблицу. Однако схема «звезда» имеет некоторые недостатки. Во-первых, обработка данных измерений может быть медленной,

так как измерения могут несколько раз появляться в одной таблице. Во-вторых, вероятность несоответствия данных высока из-за возможных ошибок ввода.

При построении реляционной модели на диске преимуществами являются практически неограниченный объем хранимых данных, когда вы добавляете новое измерение данных, вам не требуется сложная физическая реорганизация хранилища или максимально надёжные характеристики безопасности защиты информации и определения доступа к ней. Ухудшающим качеством является использование max увеличенной планки по сбору данных, а иерархические измерения таблицы агрегации могут замедлить выполнение запросов для связанных хранилищ.

Выбор модели объединения зависит от таких факторов, как объемы хранимых данных, сложность иерархии измерений и требования к изменению измерений данных.

2.2 Гибридные подходы построения архитектуры хранилища данных Data Vault и Anchor Model

Классические модели данных основаны на разделении данных на измерения и факты, с использованием внешних ключей для связей между таблицами. Однако такой подход может быть ограничивающим и требовать перепроектирования при появлении новых требований. В концепции Data vault (метод DV) и Anchor Model (метод AM) предлагается хранить все связи в отдельных таблицах и обрабатывать их по принципу многие-ко-многим, что позволяет быть более гибкими и избежать перепроектирования при изменении бизнес процесса в структуре предприятия [2].

Для создания хорошо работающего хранилища данных не требующего дополнительных вложений в процессе его эксплуатации можно использовать гибридный подход при написании логической схемы хранения информации, сочетающий метод DV и метод AM. На этапе проектирования структуры

таблиц очень важно определить, являются ли пары объектов отношениями «многие-ко-многим» или «один-ко-многим». Это позволяет вам определить, какая таблица имеет первичный ключ, а какая таблица имеет внешний ключ. Когда возникают определенные запросы на изменение, вам необходимо полностью переконструировать имеющуюся базу данных, чтобы изменить эти отношения.

При проектировании сущности допустим кассовый чек, инженер данных заложил возможность использование одной акции на один чек, что предполагало, один чек будет поддерживать несколько товаров (и наоборот). Однако со временем возникла необходимость в реализации новой стратегии, когда один и тот же товар может быть связан с несколькими акциями. В результате пришлось оптимизировать таблицы, установить связи между различными элементами и разработать производные элементы, связанные с акциями.

Чтобы избежать подобной ситуации, все связи можно хранить в отдельных таблицах и рассматривать их как связи «многие-ко-многим». Это можно реализовать, применив концепцию построения архитектуры хранилища данных с помощью методов DV и AM.

2.2.1 Моделирование измерений в Data Vault и Anchor Model

Моделирование измерений в методе DV и методе AM хранит отношения между сущностями не в атрибутах родительских сущностей, а в специальных таблицах связей Link, которые в методе DV называются связями, а в методе AM таблицах связей Tie. В обоих методах таблица связей может связывать любое количество сущностей.

Эта, казалось бы, избыточная структура может быть достаточно гибко изменена. Можно добавлять новые связи или изменять кардинальность существующих связей без изменения существующих сущностей или процессов. Пример, представленный на рисунке 8, показывает реализацию данной связи, где если у элемента чека появляется связь с кассиром, который

его выписал, эту связь можно добавить в существующую таблицу, не затрагивая другие объекты [4].

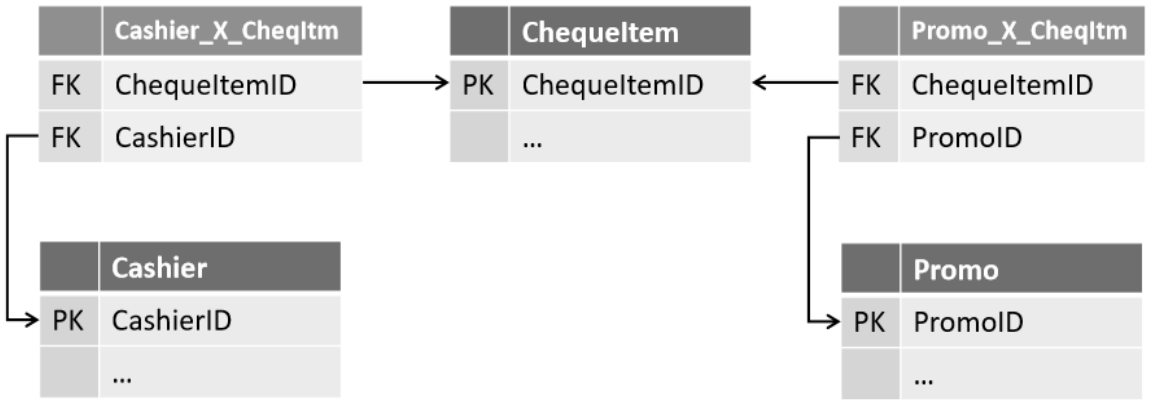

Рисунок 8 – Хранение объектов и атрибутов в Data Vault и Anchor Model

Следует отделить те составляющие данных, которые будут меняться в процессе деятельности и те составляющие данных, которые имеют неизменный вид. Соответственно ключи и атрибуты имеют отдельное хранения. Атрибуты в постоянном неизменном состоянии не имеют отношения к атрибутам имеющие возможность изменяться по мере необходимости, чтобы исправить ошибки ввода или получить новые данные.

В методологиях DV и AM могут быть различия в отношении того, что считается неизменяемой величиной. С архитектурной точки зрения, метод DV позволяет считать естественные и прокси-ключи постоянными, а остальные атрибуты группируются по источнику и частоте изменения, что позволяет каждой группе хранить отдельную версию таблицы.

В методе DV только суррогатный ключ сущности считается неизменяемым значением, в то время как все остальные атрибуты, включая естественный ключ, являются общими атрибутами. По умолчанию все атрибуты независимы друг от друга, поэтому вам необходимо создать отдельную таблицу для каждого атрибута.

В методе DV таблица, содержащая ключ сущности, называется центральной таблицей. Эта таблица всегда содержит фиксированный набор полей, таких как естественный ключ сущности, резервный ключ, ссылка на ресурс и время вставки записи. Записи сущностей в центральной таблице не изменяются и не версируются. Чтобы упростить загрузку ссылок и атрибутов из источника, мы рекомендуем использовать хеш-функции для создания резервного ключа из вашего бизнес-ключа [10].

На рисунке 9 представлено распределение атрибутов по сателлитам.

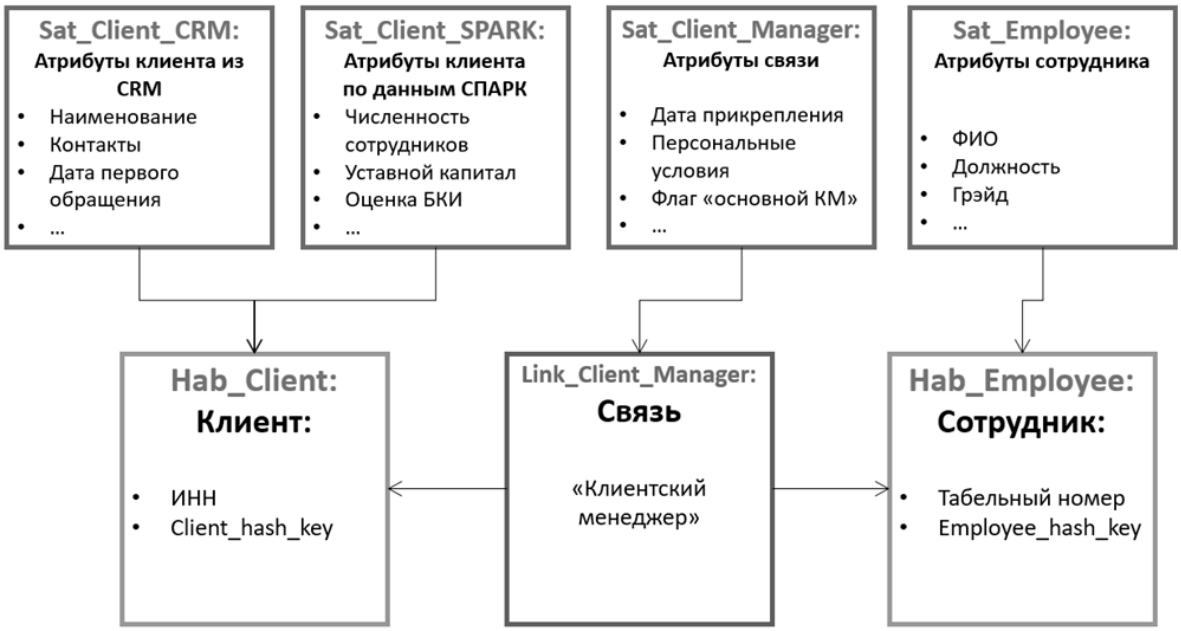

Рисунок 9 – Распределение атрибутов по сателлитам

Все остальные атрибуты объекта хранятся в специальной таблице, называемой сателлитами. Объект может иметь несколько спутников, и каждый спутник содержит определенный набор атрибутов. Атрибуты рассредоточиваются между сателлитами в зависимости от частоты обмена. Этот метод уменьшает дублирование версий и оптимизирует количество сохраненных версий.

В методологии AM таблица, содержащая ключ, называется якорем. Эта таблица содержит только резервный ключ, ссылку на ресурс и время вставки

записи. Естественные ключи считаются нормальными атрибутами. Если сущности объединяют атрибуты из разных систем, может применяться правило «привязки».

В методологии DV, если правило «прокси–Hub» родительской сущности изменяется (или обновляются добавляемые атрибуты), этих правил достаточно, чтобы воссоздать «прокси–Hub», не затрагивая сам первичный естественный ключ и центральную таблицу концентраторов, в которой хранятся атрибуты.

В методологии AM сущности хранятся в общем соединении именуемым якорем. Это означает, что все атрибуты связаны с одним и тем же прокси-сервером, независимо от источника. Такие системы затрудняют идентификацию неправильно объединенных записей и проверку правильности и точности внесения изменений, выраженных в добавлении, особенно когда правила являются переменными и сложными, а одни и те же атрибуты извлекаются из разных источников. Методология AM также включает в себя дополнительный тип объектов, называемый узлом, для хранения «плоских» индексов (пол, семейное положение, категория обслуживания клиентов и т.д.). В отличие от привязки, с узлом не связана таблица атрибутов, и каждый атрибут (имя) всегда хранится в одной и той же таблице с использованием ключа. Узлы связаны с таблицей ссылок привязки таким же образом, как привязки связаны друг с другом [26].

2.2.2 Моделирование фактов в Data Vault и Anchor Model

В методологии DV объекты обрабатываются отдельно, а также xab, содержащие свои именные характеристики. В AM методе связанные якоря именуемые соединяются при помощи отдельной таблицы связи, но их собственные качества не играют предсказуемой роли. Данное отличие приводит к использованию разных подходов к структурированию данных.

Метод DV связывает данные и использует отношения при сохранении фактов. Этот подход кажется нелогичным, поскольку он обеспечивает доступ только к рассматриваемым показателям, аналогично классическим таблицам

фактов для анализа. Чтобы добавить новый информационный ключ к модели, вам также необходимо добавить внешний ключ для установления соединения. Это может разрушить конструкцию и изменить другие части модели.

Метод AM требует сильных якорей, которые должны быть привязаны к определенным местам в зависимости от атрибутов и показателей. Каждый факт должен иметь свой собственный якорь. В некоторых случаях то, что воспринимается как явление, может быть выражено естественным образом. Например, явление покупки можно представить как приобретение товара, а посещение сайта – как процесс сеанса. Существуют факты, затрудняющие определение «естественной цели отношения». Например, наличие и баланс определенного товара в начале дня. Или увеличьте количество информационных ключей способом метода AM не вызывает проблем с модульностью (достаточно добавить новую ссылку к соответствующей ссылке), но дизайн модели может стать неоднозначным, что приведет к созданию «искусственной» модели связей, которая не будет четко представлять объектную модель бизнеса.

Вместе с тем, использование такой структуры требует более сложных запросов для анализа данных, так как данные оригинальной таблицы и соответствующих сателлитов или отдельных таблиц должны быть объединены для получения полной информации. Также, при изменении атрибутов объекта измерения, необходимо обновлять соответствующие записи в сателлитах или отдельных таблицах.

Таким образом, методы с использованием дополнительных таблиц или сателлитов могут быть полезными при определенных условиях, таких как сокращение дискового пространства или уменьшение дублирования значений атрибутов. Однако, они также требуют более сложных запросов и дополнительных усилий при обновлении данных.

Такая гибкость позволяет легко адаптироваться к изменениям требований и добавлять новые функциональности без необходимости

изменения всей структуры данных. Кроме того, раздельное хранение атрибутов упрощает взаимодействие и обработку данных, так как каждый атрибут и предметная область имеют свои собственные правила и операции. Это также способствует повышению производительности и улучшению реализации алгоритмов, так как необходимые данные могут быть быстро и эффективно извлечены и обработаны.

Данные методы проектирования архитектуры хранилища данный соответствуют agile-методологии, где подход к моделированию данных отличается от традиционного подхода. Вместо создания отдельной таблицы для каждого уникального атрибута или сущности, в методах Data Vault и Anchor Modeling используются типовые "детали" или шаблоны таблиц. Это означает, что вместо того, чтобы разрабатывать и поддерживать каждую таблицу индивидуально, можно использовать общие структуры таблиц для различных атрибутов или сущностей, которые имеют схожие свойства. Это позволяет сократить время и усилия, затрачиваемые на разработку и обслуживание модели данных.

Конечно, процесс загрузки и выборки данных может показаться сложным из-за большего количества таблиц, но также стоит учитывать преимущества автоматизации и управления метаданными. Благодаря типовым таблицам можно создавать и изменять сущности или атрибуты гораздо более эффективно и без необходимости изменения каждой отдельной таблицы [5].

Таким образом, методологии Data Vault и Anchor Modeling предлагает более гибкий и эффективный подход к моделированию данных, который упрощает и ускоряет разработку и поддержку баз данных. Недостатком методов можно считать низкую производительность.

Гибридные подходы к построению хранилищ данных, основанные на методе Data Vault и Anchor Modeling, обеспечивают эффективный способ моделирования отношений и управления изменениями данных. Эти методы

обеспечивают гибкость, позволяющую реагировать на изменяющиеся требования, и упрощают процесс загрузки и обработки данных [9].

Вывод по разделу 2

Классические методы «звезда» и «снежинка» являются стандартными подходами к проектированию логической схемы базы данных хранилища. В методе «звезда» используется центральная таблица-факт, которая содержит ключевые показатели и связывается с измерениями через внешние ключи. Метод «снежинка» расширяет подход «звезда» путем добавления более сложных связей между измерениями, что позволяет создавать более гибкие отчеты.

Однако, с появлением гибких методологий «Data Vault» и «Anchor Model», которые базируются на идеях моделирования данных в виде хранилища и управления метаданными, появилась возможность решать более сложные задачи анализа данных.

Методология «Data Vault» подразумевает разделение базы данных хранилища на три основных типа таблиц: хабы, связи и сателлиты. Хабы представляют собой централизованные таблицы, которые содержат уникальные значения и используются для идентификации записей в других таблицах. Связи представляют собой связи между хабами, а сателлиты содержат исторические и контекстные данные. Такой подход обеспечивает гибкость и масштабируемость базы данных.

«Anchor Model» представляет собой набор нормализованных таблиц, которые содержат атрибуты данных и их контекст. Главной идеей «anchor model» является отделение версий данных и контекста данных от самих атрибутов, что позволяет более гибко добавлять новые атрибуты или изменять контекст данных.

Оба подхода – «Data Vault» и «Anchor Model» – обеспечивают гибкость и масштабируемость базы данных хранилища. Они позволяют достичь более высокой степени нормализации и обеспечивают управление метаданными, что упрощает разработку и поддержку базы данных.

Таким образом, анализ существующих подходов проектирования логической схемы базы данных хранилища показывает, что классические методы «звезда» и «снежинка» отлично работают для простых задач анализа данных, однако для сложных и гибких систем рекомендуется использовать методологии «Data Vault» и «Anchor Model». Эти подходы позволяют более эффективно управлять данными и достигать лучших результатов в анализе информации.

Раздел 3 Модель архитектуры слоев хранилища данных применимого для малого предприятия

3.1 Архитектура базы данных хранилища

Для малого предприятия изначально спроектируем классический вариант деления хранилища данных на отдельные части (слои) при проектировании архитектуры хранилища предприятия. При построении логики процесса течения данных от источника к потребителю будем применять отглагольный производящий метод над данными. Таким образом, информация на предприятии собирается, стандартизируются, сохраняется в хранилище данных предприятия, в хранении информации учитывается финансовая возможность предприятия. В последнюю очередь хранящаяся информация передается для осуществления анализа. Отсюда следует произведение следующих действий над данными – собрать, стандартизировать, сохранить, анализировать. По этому принципу будем делить хранилище данных на отдельные функциональные слои данных на логическом уровне.

3.1.1 Архитектура слоев хранилища данных

Начальный уровень действий с информацией происходит в слое RAW (собрать данные) – когда информация собирается и хранится у внешних информаторов (источников данных), информация, полученная в виде различных форматов данных на этом уровне, сохраняется в полученном формате, содержимое не изменяется, изменяется только формат, что делает его пригодным для дальнейшего преобразования.

Цель первого этапа действия над данными в этом слое сохранить подступающий поток информации от поставщика данных для дальнейших действий над данными. Для этого следует собрать данные с источника и преобразовать их в удобный и стандартизированный формат, используемый в данном хранилище данных [6].

На рисунке 10 представлено место слоя RAW в структуре хранилища данных.

Рисунок 10 – Место слоя RAW в структуре хранилища данных

Следующий ODS-слой (стандартизировать данные) – операционный слой. В представленном слое происходит стандартизация и хранение поступающей информации данных от поставщика, историчность не соблюдается. Учитывая поступившую информацию данных и важности поставщика, возможна замена и обновление данных (дата, состав).

Цель операционного слоя – определить и создать сущности и группы сущностей поступающей информации от поставщика данных. Для работы с данными в данном слое должна быть возможность предоставление стандартного доступа к ним, предоставляя требуемый интерфейс в не зависимости от внешних особенностей начального поставщика данных.

На рисунке 11 представлено место слоя ODS в структуре хранилища данных.

Рисунок 11 – Место слоя ODS в структуре хранилища данных

Здесь хранятся операционные данные. Формируется подбор сущностей от начального поставщика данных и происходит распределение данных по сущностям.

Следующий DDS-слой – центральный слой в хранилище данных именуемый как детальный. Данный слой является главным слоем в хранилище данных. В детальном слое происходит накопление информации данных, управление взаимодействия между всеми начальными поставщиками данных. Детальный слой в хранилище данных можно представить как его ядро хранения информации.

Основной цель данного слоя является централизованное накопление и хранение информации данных о выделенных сущностях в предприятие. Отличительная функция детального слоя это логически выстроенная консолидация всех данных предприятия между выделенными сущностями бизнеса происходящего бизнес процесса. Для работы с данными в данном слое предоставляется возможность стандартного доступа к сущностям данных бизнеса по требуемому интерфейсу.

На рисунке 12 представлено место детального слоя в структуре хранилища данных.

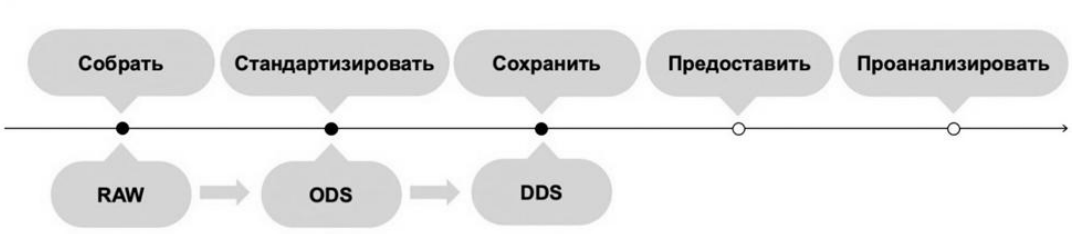

Рисунок 12 – Место детального слоя в структуре хранилища данных

Следующий раздел архитектуры хранилища данных CDM-слой. Это слой предоставляемых витрин данных. При помощи данного слоя формируются стандартные табличные витрины и оптимизируется различные

SQL запросы или другое предоставление доступа на чтение информации данных.

Цель CDM-слоя в хранилище данных предприятия состоит в том, чтобы предоставить больше данных для анализа в визуальном виде. Предоставлять информацию по требованию бизнес-потребителя. Улучшать оптимизацию и производительность при запросе на чтение информации данных.

На рисунке 13 представлено место слоя CMD в структуре хранилища данных.

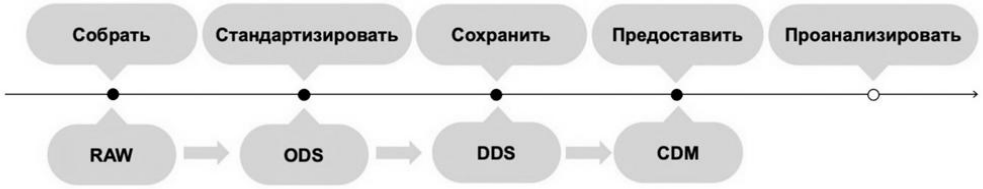

Рисунок 13 – Место слоя CMD в структуре хранилища данных

Следующий раздел архитектуры хранилища данных REP-слой. Это раздел отчетов и аналитики. При помощи данного слоя формируются и хранятся подготовленные заранее отчетные заготовки для конечных потребителей информации данных. Цель слоя сохранять аналитические заготовки и срезы по интересующей информации данных. Формировать данные в контексте бизнес потребителей. Готовить агрегированные отчеты.

На рисунке 14 представлено место слоя REP в структуре хранилища данных.

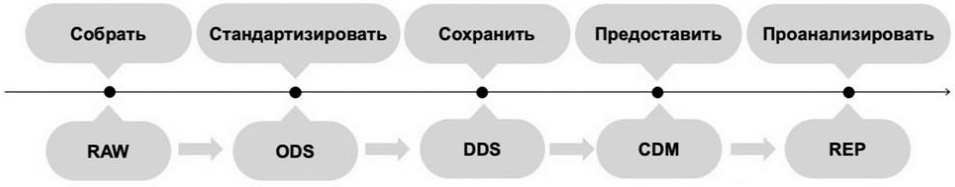

Рисунок 14 – Место слоя REP в структуре хранилища данных

3.1.2 Проектирование детального слоя в хранилище данных

Критически важным пространством для создания всего хранилища данных будет уровень DDS-слой, который именуется детальным. На последовательной структуре передвижения информации данных он находится в центре и является ядром хранилища. Данное расположение имеет показательное значение и будет нести показательное значение при построении хранилища данных [16].

Главным требованием к DDS-слою во всей системе хранилища данных это накопление информации данных, управление взаимодействия между всеми начальными поставщиками данных по всей структуре бизнес деятельности предприятия. В процессе функционирования детального слоя в нем хранится история изменений сущностей бизнес информации данных и связей между этими сущностями.

Причем детальный слой обязан принимать различные изменения бизнеса, не разрушая своей логической структуры. Данное свойство очень важно для бизнеса и достигается сложным путем. Потребность к устойчивости к изменению и проявленною гибкости часто встречается на различных этапах деятельности предприятия. Это можно наблюдать, когда в бизнесе появляются новые сущности или связи между сущностями, в определенный момент поменялись. Либо предприятие кардинально поменяло свой подход к ведению бизнес процессов.

В той же мере угоду изменений важно постоянно не переделывать всю модель хранилища данных. От детального слоя требуется определенной функциональности и гибкости. Для добавления дополнительных элементов в детальный слой он должен быть масштабируемым. Для изменения или исключения сущностей и связей DDS-слой обязан быть представлен модульном виде. При росте бизнеса предприятия и нарастании хранилища данных этот функционал детального слоя позволит сохранять всю проделанную до этого работу над хранилищем данных [30].

При проектировании детального слоя, следует провести развернутое исследование имеющихся подходов для его построения. Изучить все возможные варианты, начиная с начального этапа при котором подхода к проектированию не существует. Это можно описать, как нет ни детального слоя, ни подхода.

В данном случае при использовании без DDS-слоя все манипуляции происходят сразу на операционном слое, построение отчётов и витрин происходит прямо на ODS-слое. Для начинающих и малых предприятий это приемлемо, так как в самом начале деятельности у бизнеса сравнительно мало источников с поступающими данными. В зависимости от нарастания поставщиков информации данных и взаимодействие между этими источниками данных, как и построение витрин, увеличивается и требует большего места и мощности обработки запросов. На этом этапе уже требуется выделение отдельного слоя под решение данных задач и устранение излишнего многообразия создания витрин данных.

Вторым подходом при проектировании детального слоя хранилища данных можно считать применение классической нормализации данных до 3НФ, такие как построение по методу «звезда» и «снежинка». Данные подходы просты в исполнении и дальнейшем использовании. Требуется определиться с понятием первичного и внешнего ключа, join генерацией таблиц в бизнес-процессе хранения и управления данных на предприятии.

Из данной методики в последствие пользования вытекают такие недостатки, как сложность в перестройки, когда изменяются кардинальности связей меду сущностями бизнес данных предприятия. Например, один участник имеет множество заказов, но один заказ может быть предоставлен только одному участнику. В бизнесе происходят кардинальные изменения и теперь связи поменялись. У одного заказа появилась возможность множества участников и сами участники получили возможность разделять один заказ между собой.

Третьим подвидом применяемых подходов к проектированию детального слоя хранилища данных это гибкие методологии Data Vault (DV) и Anchor Modeling (AM). Данные методы схожи по своей сущности, предлагают строгую нормализацию и правила на создание таблиц. Взамен эти методологии предполагают в процессе работы хранилища данных отсутствие его пристраивания при решении бизнес задач, максимальную гибкость в управлении. Данные подходы в настоящее время являются современными. Постоянно модернизируются и все шире находят свое применение на практике [39].

На рисунке 15 проанализированы эксплуатационные свойства при применении упомянутых выше подходов.

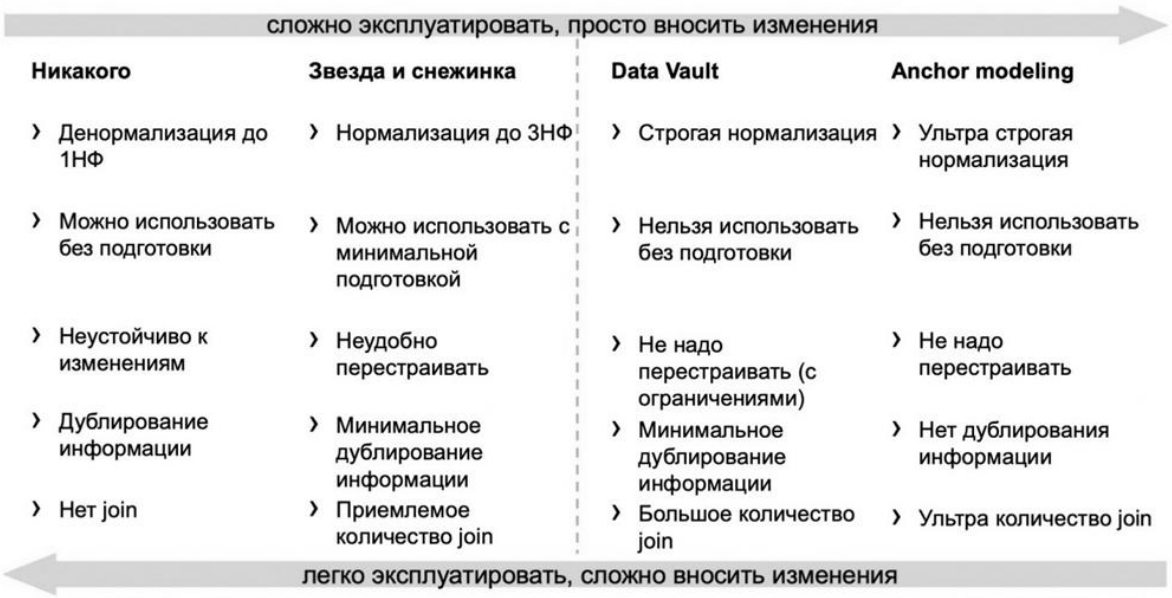

Рисунок 15 – Анализ эксплуатации классических подходов с DV/AM при проектировании детального слоя

Правая часть рисунка относится к эксплуатационным свойствам гибких подходов DV и AM. Мы видим у обоих указано огромное количество генерации таблиц, что в конечном итоге сильно усложнит использование хранилища. Увеличение создаваемых таблиц усложнит написание SQL-

запросов, что в сою очередь негативно повлияет на производительность выдачи информации данных. Но благодаря этому, имея расширенную модульную структуру можно просто и точечно вносить изменения не влияя на модель хранилища и соответственно не подвергая бизнес в проблемы.

Левая часть рисунка показывает нам зависимость простого и классического подхода построения архитектуры детального слоя хранилища данных при его эксплуатации. В данном случае мы видим либо полное отсутствие join или совсем малое количество. Малое количество таблиц в логической структуре модели увеличивает производительность при получении запроса из базы данных, упрощает работу с хранимой информацией данных. Все данные хранятся в обычной плоской таблице и нормализованы. Но при всей простоте создания и использования данной модели, наблюдается сложность в перестройке системы либо в определенных случаях она может быть не возможна .

3.2 Выбор подхода проектирования хранилища данных из гибких методологий Data Vault и Anchor Model

Сравнивая подходы проектирования DV и AM, сопоставляя модель их построения, мы видим много общего.

Нормализации данных более 3НФ. Каждый подход предполагает свой тип таблиц в структуре, которые наложены строгие ограничения на то, что они могут содержать, и какие действия над ними возможны. Применяя данные подходы в процессе деятельности хранилища данных, воспроизводится огромное количество таблиц, требующих качественного управления над ними.

Отсутствует или уменьшается до минимума дублирование информации данных при изменении одного атрибута. Простое и устойчивое изменение и расширение хранилища данных при использовании данных моделей. При

подходе исключатся вредные изменения, влияющие на структуру связей данных.

Подход DV предлагает определенные типы таблиц, распределяя ответственность на каждую -– это Hub, Link и Satellite:

- таблица Hub ответственная за хранение сущности;
- таблица Link ответственная за обеспечение связей между сущностями;
- таблица Satellite ответственная за содержание атрибутов и описаний в методологии DV.

Талица Hub (сущность) представляет собой отдельно содержащую бизнес ключи таблицу. Мы имеем внешний ключ от сущности, суррогатный ключ, определяемый в хранилище данных. Плюс имеется техническая информация о времени загрузки информации данных и ID поставщика информации данных (источник).

Таблица Link (связь) определяет отношение на физическом уровне взаимоотношений между сущностями в хранилище данных. Данная связь между таблицами определена только как многие ко многим (M-M) и выделена в отдельную таблицу. В таблице связей атрибуты содержат суррогатный ключ, относящийся к таблице связи Link ключи, относящиеся к таблице сущности Hub, учитывая зависимость какие сущности имеют связь через эту таблицу. Обязательные технические данные времени загрузки информации данных и ID поставщика информации данных (источник).

Таблица Satellite хранит в себе атрибуты как историческую информацию о сущностях. Это описательные данные в виде ключа связи или сущности относительно характеристик загруженного в данном Satellite. Непосредственно данные и технические данные времени загрузки информации данных из источника [15].

Подход AM предлагает более строгое моделирование выше нормализованность данных по сравнению с методом DV. В данном подходе

имеются определенные типы таблиц, распределяя ответственность на каждую – это якорь, атрибут, связь и узел.

Таблица якорь – сопоставима с таблицей сущностей Hub в методологии DV, но отличающаяся тем, что в ней нет бизнес ключей, а имеется лишь суррогатный ключ и технические данные времени загрузки информации данных и ID поставщика информации данных (источник). В данном подходе Бизнес-ключ представлен как атрибут и помещается в отдельную таблицу.

Таблица связь – эта таблица копирует таблицу соединений подхода DV, которая описывает взаимосвязь на физическом уровне взаимосвязи между сущностями в хранилище данных. Это отношение между таблицами определяется только как много-ко-многим (M-M) и присваивается другой таблице. Единственное отличие это строгое отсутствие атрибутов в данной таблице.

Таблица атрибута – в данной таблице хранятся атрибуты, описывая информацию о сущностях. Отличатся от метода DV в том, что в подходе AM применяется 6НФ где предусмотрена строгость 1 атрибут – 1 таблица. Данное условие позволяет достаточно легко и быстро масштабировать атрибуты для сущностей, расширяя их состав. При добавлении атрибута происходит добавление новой таблицы под него. В этом случае ничего в модели не меняется и не удаляется. Огромным недостатком данного подхода будет являться высокая генерация новых таблиц, их становится огромное количество, что значительно усложнит пользование хранилищем данных.

3.3 Создание гибридной модели детального слоя хранилища данных

Существуют проблемы классического подхода к проектированию хранилища данных – это, как правило, сочетание вариантов схем «звезда/ снежинка» с 3НФ – основной проблемой данного подхода является

отсутствие гибкости. Причина этого является: жёсткая кардинальность связей, дублирование данных, нелинейная сложность доработки.

Подходы звезда и снежинка просты в использовании, основным требование является понятие о первичном ключе, внешнем ключе, join таблица. Недостаток сложность в перегруппировке кардинальности связей часто встречается при изменении бизнеса, и перестраивать уже будет очень проблематично.

Либо проблема связана, когда при проектировании хранилища данных не используется каких либо методологий и подходов, здесь информация хранится в одной массе и при извлечении ее приходятся сталкиваться с серьезными проблемами. Единственным плюсом может служить это применение данного метода, без какой либо подготовки, но при каких либо изменениях в структуре приходится перестраивать все заново.

Подходы гибких методологий Data Vault и Anchor modeling можно рассмотреть в едином ключе, т.к. они схожи повышенной нормализацией, для применения требуется понимание и осведомленность в правилах структурной составляющей метода [27].

В современном мире малое предприятие постоянно обновляется и со временем изменения в хранилище данных будут. Поэтому к построению хранилища данных на малом предприятии стоит подходить с гибкостью. Данными методами это возможно. Плюсы применения – увеличение (выше 3НФ) нормализации, при использовании имеется возможность применять большое количество таблиц, на которые в свою очередь действуют жёсткие ограничения действия с ними и их содержание. Уменьшается вредное дублирование данных. Тем самым хранилище данных в процессе изменений не приводит к большой трудоемкости. Оно расширяется и преобразуется просто и быстро.

Самый результативный вариант решения проблем гибкости хранилища данных это выбрать лучшее из каждого подхода и правильно применить на каждом этапе. Нам нужен оптимальный вариант хранения данных.

На рисунке 16 представлена архитектура слоев данных на начальном этапе проектирования.

Рисунок 16 – Начальная архитектура проектируемого хранилища данных

Цели на каждом этапе:

- RAW – сохранить информацию с источника;
- ODS – хранить операционные данные;
- DDS – накапливать данные о сущностях доменной модели;
- CDM – предоставлять витрины данных для анализа;
- REP – хранить отчетные срезы.

Для поставленной цели формируем гибридную модель на основе Data Vault и Anchor modeling. В данной модели не происходит строгих ограничений по подходам к проектированию. Имеется возможность выбора одного или другого. Имея такую возможность, мы можем разделить и закрепить уровни проектирования моделирование отдельно на логический уровень и физический уровень.

Определяем важные требования для решения поставленной задачи:

- устойчивость к изменениям;
- модульность;
- масштабируемость;
- удобное построение витрин, кубов таблиц;
- способность интегрироваться или Data Vault или Anchor modeling;
- идемпотентность повторной загрузки;

– высокая нормализация;

– загрузка из источников параллельная.

На рисунке 17 представлено многомерное представление архитектур слоев данных хранилища [21].

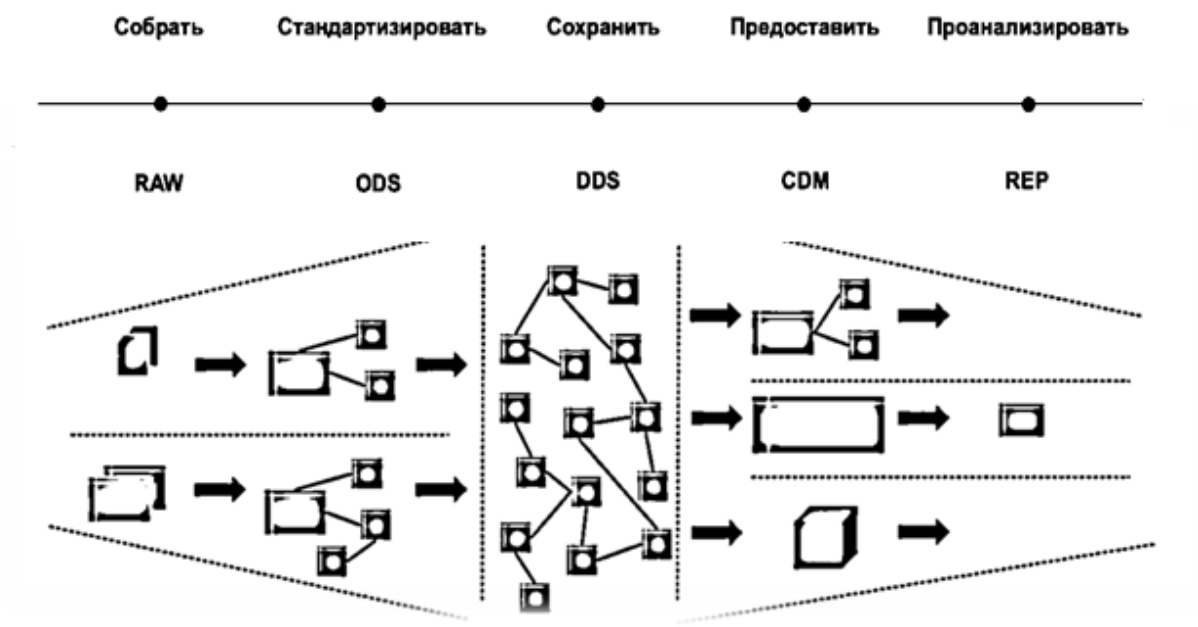

Рисунок 17 – Многомерное представление архитектур слоев данных хранилища

Представлена схема каркаса хранилища данных, разделенного на отдельные слой, каждый отвечающий за свою задачу при загрузке, хранении и выгрузки данных.

– RAW – сохранить информацию с источника;

– ODS – хранить операционные данные;

– DDS – накапливать данные о сущностях доменной модели;

– CDM – предоставлять витрины данных для анализа;

– REP – хранить отчетные срезы.

В данном случае (RAW+ODS) – озеро данных, которые были захвачены (RAW) и сохранены в первоначальном виде. После чего в слое

ODS они немного преобразовались, но все еще это операционные данные, не имеющие историю.

В центральном слое (детальный слой DDS) произошло сливание данных путем описание сущностей и связей меду ними на логическом уровне. Далее на основании детального слоя мы имеем возможность построения витрин, плоских таблиц и кубов для аналитических действий с помощью запросов.

Разделенная на слои данная модель представляет собой возможность конструировать на основе слоя. В нашем случае нам интересен детальный слой DDS. При построении архитектуры детального слоя применяем гибрид гибких подходов к проектированию Data Vault и Anchor modeling.

3.3.1 Разделение логического и физического моделирования

На начальном этапе происходит разделение архитектуры детального слоя на логический и физический уровень.

Логический уровень представляет собой ER-диаграмм сущность-связь и их ключи, атрибуты сущностей и их историзм. Проектирование логического уровня зависит бизнеса, для которого оно проектируется, требуется выделить изначально сущности и домены в этом бизнесе.

Физический уровень представляет собой набор команд (скрипты DDL) в базе данных. На данном уровне происходит разбиение таблиц сущностей по критериям, объединение атрибутов в отдельные группы, параллельное управление хранения данных, применение индексов ускорения. Физический уровень напрямую зависит от СУБД и технологического оснащения. Требуется сокрыть физический уровень для того что бы имелась возможность при необходимости переходить от Data Vault к Anchor model и наоборот.

Разделяя на логический и физический уровень, происходит распределение на две задачи. На рисунке 18 представлено разделение уровней проектирования при создании гибридной модели хранилища.

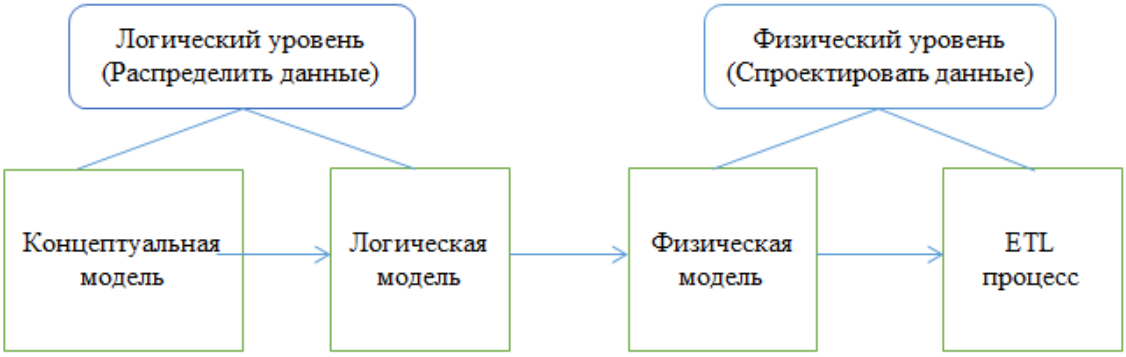

Рисунок 18 – Разделение уровней проектирования при создании гибридной модели хранилища

Распределить данные. Строим на логическом уровне концептуальную модель, основываясь на направления деятельность бизнеса, его возможности. Она должна определить, как будут меняться атрибуты.

Концептуальная модель формирует общее представления имеющегося на начальном этапе направлений бизнеса, обозначаются взаимоотношение между ними и потребляющих систем.

Логическая модель определяет сущности существующего бизнеса, связи между ними, частота изменения их атрибутов.

Проектирование данных.Строим физическую модель. Она должна решить, как хранить атрибуты, осуществить при необходимости разбиение данных. В итоге осуществляет ETL процесс [37].

3.3.2 Закрепление разделение уровней проектирования

Основываясь на ER-диаграмме, используемой СУБД, технических ограничений, определяем сущность и связь на логическом и физическом уровне.

В таблице 1 представлено описание сущности гибридной модели хранилища данных.

Таблица 1 – Описание сущности гибридной модели хранилища данных

<table>
<tr><td colspan="2">Сущность</td></tr>
<tr><td>Логический уровень</td><td>Физический уровень</td></tr>
<tr><td>Сущности описываются:</td><td>Физическая сущность состоит:</td></tr>
<tr><td>- Именованием и комментарием.
- Бизнес ключом, поле определяющее сущность.
- Атрибуты.</td><td rowspan="3">- Hub – таблица, содержащая техническую информацию и суррогатный ключ.
- Attribute – таблица, содержащая информацию об атрибуте, может содержать или нет историю.
- Group – таблица, содержащая информацию о нескольких атрибутах, может содержать или нет историю, приходят от одного источника и имеют один историзм.</td></tr>
<tr><td>Атрибуты описываются:</td></tr>
<tr><td>- Именованием и комментарием.
- Типом сущности.
- Хранение истории по атрибуту.</td></tr>
</table>

Сущность базовая часть любого описания домена сферы деятельности бизнеса, на логическом уровне имеет описание: именем, бизнес ключ и атрибутный состав. Атрибуты так е описываются. Прописывается историзм.

На физическом уровне отдельная сущность представлена в виде таблиц содержащая техническую информацию и суррогатный ключ. Атрибуты в отдельных таблицах со своей информацией об этом атрибуте. Атрибуты мы можем сгруппировать в отдельную таблицу группы атрибутов со своей информацией из каждого объединённого в таблицу атрибута.

В таблице 2 представлено описание связи гибридной модели хранилища данных.

Таблица 2 – Описание связи гибридной модели хранилища данных

Связь	
Логический уровень	Физический уровень
Связи описываются:	Физически связь - это таблица многие ко многим, которая содержит:
- Списком связанных сущностей. - Типом каждой из связи (M - M, 1-M). - У связи не может быть атрибутов- если есть необходимость добавить в связь атрибут, то необходимо выделить сущность.	- Суррогатные ключи всех входящих в связь сущностей. - Поля историзма SCD2. - Поля - партиции сущностей.

Связь на логическом уровне это список связанных сущностей с координацией связей меду ними. Отдельно атрибуты не хранятся.

На физическом уровне связь это отдельная таблица «многие ко многим», которая содержат суррогатные ключи связываемых через нее сущностей и поля историзма.

Таким образом, наша модель представлена на логическом уровне описанием сущностей в виде ER-диаграммы, а на физическом уровне набором таблиц.

3.3.3 Сокрытие физической модели

Следующей задачей требуется сокрытие физической реализации, что бы иметь возможность предоставить интерфейс работы с сущностями и связями. Данную процедуру описываем в виде загрузки данных и построение витрин.

В таблице 3 представлено описание сокрытия физической реализации гибридной модели хранилища данных.

Таблица 3 – Описание сокрытия физической реализации гибридной модели хранилища данных

Скрытие физической реализации	
Загрузка данных	Построение витрин
- Идемпотентность если мы загружаем одни и те же данные несколько раз, то результат не меняется - Параллельность - все таблицы физической реализации могут грузиться параллельно - Сокрытие сложности SCD2 загрузки - Во всех загружаемых данных должна быть бизнес-дата, атрибуты одной группы грузятся из одного источника	- Работа с сущностями на логическом уровне, а не с таблицами на физическом - Сокрытие сложности исторической обработки записей - Максимизация простоты работы с инкрементом.

Физическая реализация загрузки срыта. Загрузка таблиц происходит параллельно. При построении витрин скрыта сложность построения таблиц сущностей, связей, атрибутов и групп таблиц. Работа с сущностями и

связями происходит на логическом уровне, которые в свою очередь связаны с таблицами на уровне физическом. Это позволило максимально просто работать с инкрементом и упростило историческую обработку записи данных. Таким образом, имеем возможность составить максимально удобный формат хранения данных для определенной ситуации или случая. Можем производить группировку атрибутов.

При сопоставлении Data Vault и Anchor model можем выделить то, что было применено из каждой методологии.

Модель Data Vault представлена «группой атрибутов» так называемые сателлиты. И нет строгости где 1 атрибут это 1 таблица. Атрибуты группируются в отдельные таблицы по совместимости изменения источника или пользователя или одновременно.

Модель Anchor model представлена, где каждая сущность имеет свой так называемый якорь – таблица с суррогатным ключом. Бизнес ключи не хранятся в отдельной таблице, они представлены как атрибуты. Связь происходит через отдельную таблицу, которая не моет быть связана с атрибутами [19].

3.3.4 Описание метаданных

Если сконструировать представленную модель с помощью DDL скриптов будет очень сложно впоследствии управлять, т.к. образуется огромное количество таблиц и метаданных. Для управления нам потребуется подборка отдельной платформы определенная СУБД, на которой будут представлены слои и ответственный детальный слой, находящийся в самом хранилище. Подборка сервисов и СУБД возможно индивидуально в зависимости от потребностей и возможности бизнеса. Для примера представим оптимальную структуру платформы для осуществления поставленной цели в проектировании хранилища данных.

На рисунке 19 представлена схема полноценной платформы для реализации хранилища данных.

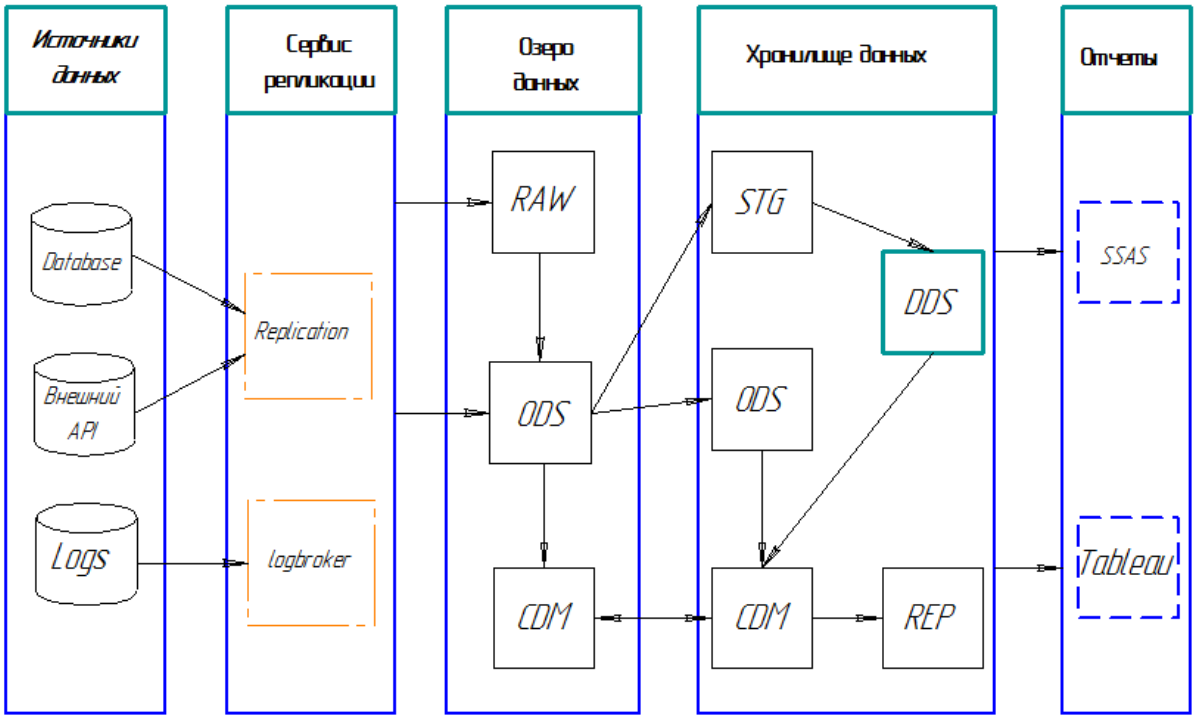

Рисунок 19 – Схема полноценной платформы для реализации хранилища данных

Левая часть схемы представлена СУБД предоставляющая сбор данных с источников.

В середине находится озеро данных, в котором хранятся слои RAW и ODS. На этом кластере храниться основной объем хранения данных под управление определенной СУБД.

Справа - хранилище данных, в котором находится детальный слой DDS, он является главным элементом хранилища данных, с помощь него происходит построение всех запросов, витрин и других аналитических действий [13].

3.3.5 Генерация объектов базы данных

Генерация объектов базы данных происходит при помощи DDL операций. Применяемый язык программирования индивидуален в зависимости от технических условий, СУБД и предпочтений. В настоящее время для удобной реализации описания сущностей и связей в детальном

слое DDS логической части на физическом уровне желательно применение высокоуровневого языка Python.

После объявления сущностей и связей согласно разработанной модели на основе гибридного подхода к построению взятого из Data Vault и Anchor model, требуется загрузить эти сущности и связи. Для этого описывается на программном языке (в данном случае применяется Python, который имеет удобные инструменты для этого) ETLкуб который участвует в процессе загрузки. Здесь происходит описание, что попадает на входе, дальнейшее распространение по атрибутам. Указывается, какое поле является бизнес датой [20].

На рисунке 20 представлен скрипт объявление сущности «Сотрудник» в DDS-слое. На рисунке 21 представлена ER диаграмма физической модели объявленной сущности. На рисунке 22 представлен скрипт объявление сущности «Сотрудник» с группами атрибутов в DDS-слое. На рисунке 23 представлена ER диаграмма физической модели объявленной сущности с объединенными в группу атрибутами.

```
class Person(Entity):

    __layout__ = Layout(layer='dds', name='person', group='staff')

    person_id = Int(comment='ID в Стаффе', change_type=ChangeType.IGNORE)
    first_name_ru = String(comment='Имя сотрудника', change_type=ChangeType.UPDATE)
    last_name_ru = String(comment='Фамилия сотрудника', change_type=ChangeType.NEW)
    login = String(comment='Рабочий login', change_type=ChangeType.IGNORE)
    gender = String(comment='Пол', change_type=ChangeType.UPDATE)
    tshirt_size = String(comment='Размер футболки', change_type=ChangeType.UPDATE)
    birthday_dt = Date(comment='Дата рождения', change_type=ChangeType.UPDATE)
    is_dismissed_flg = Boolean(comment='Был уволен', change_type=ChangeType.NEW)
    is_homeworker_flg = Boolean(comment='Надомник', change_type=ChangeType.NEW)
    is_robot_flg = Boolean(comment='Робот', change_type=ChangeType.NEW)
    is_trainee_flg = Boolean(comment='Стажер', change_type=ChangeType.NEW)

    __keys__ = [login]
```

Рисунок 20 – Скрипт объявление сущности «Сотрудник» в DDS-слое

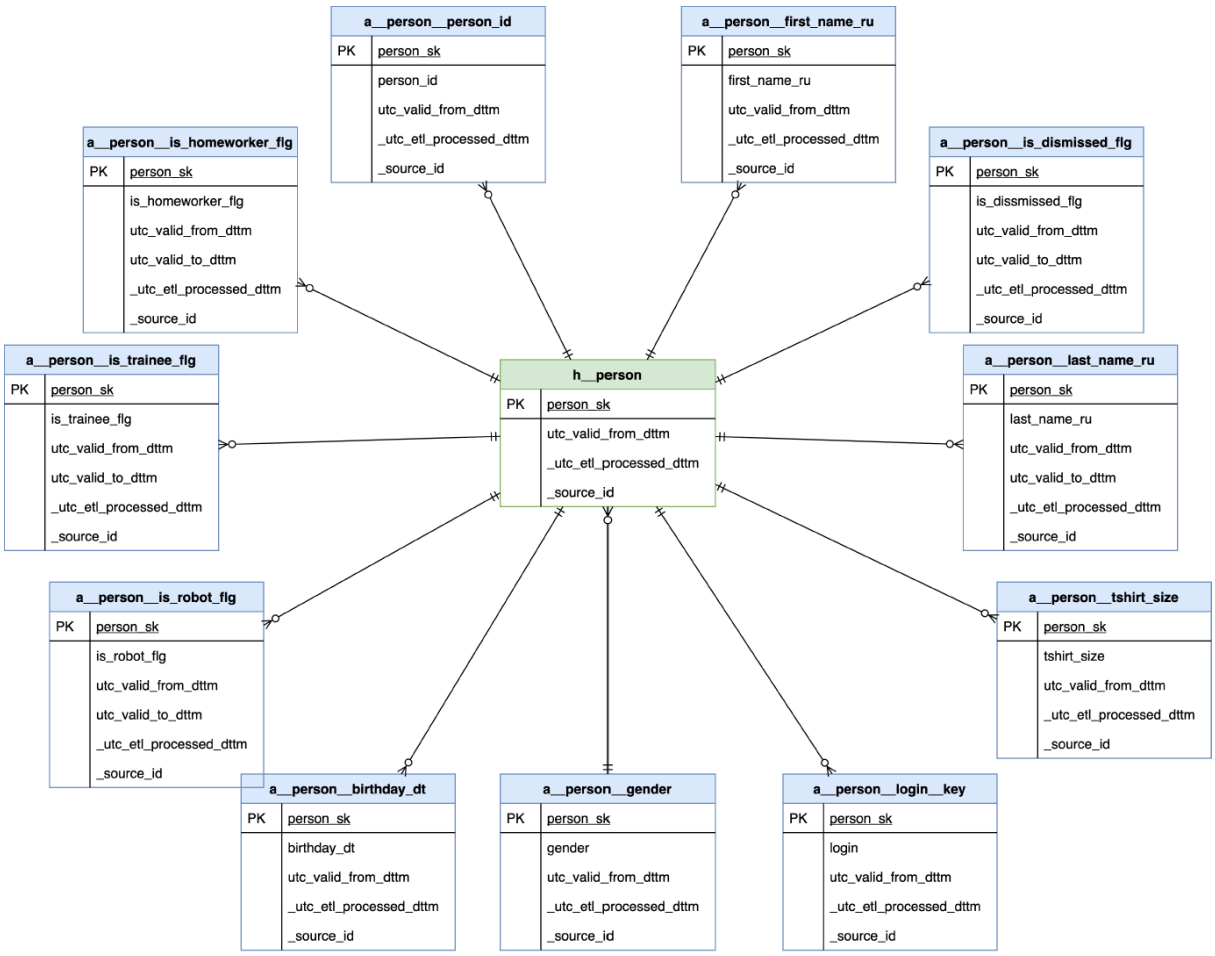

Рисунок 21 – ER диаграмма физической модели объявленной сущности

```python
class Person(Entity):

    __layout__ = Layout(layer='dds', name='person', group='staff')

    person_id = Int(comment='ID в Стаффе', change_type=ChangeType.IGNORE, group='key')
    first_name_ru = String(comment='Имя сотрудника', change_type=ChangeType.UPDATE)
    last_name_ru = String(comment='Фамилия сотрудника', change_type=ChangeType.NEW)
    login = String(comment='Рабочий login', change_type=ChangeType.IGNORE, group='key')
    gender = String(comment='Пол', change_type=ChangeType.UPDATE, group='info')
    tshirt_size = String(comment='Размер футболки', change_type=ChangeType.UPDATE, group='info')
    birthday_dt = Date(comment='Дата рождения', change_type=ChangeType.UPDATE, group='info')
    is_dismissed_flg = Boolean(comment='Был уволен', change_type=ChangeType.NEW, group='flg')
    is_homeworker_flg = Boolean(comment='Надомник', change_type=ChangeType.NEW, group='flg')
    is_robot_flg = Boolean(comment='Робот', change_type=ChangeType.NEW, group='flg')
    is_trainee_flg = Boolean(comment='Стажер', change_type=ChangeType.NEW, group='flg')

    __keys__ = [login]
```

Рисунок 22 – Скрипт объявление сущности «Сотрудник» с группами атрибутов в DDS-слое

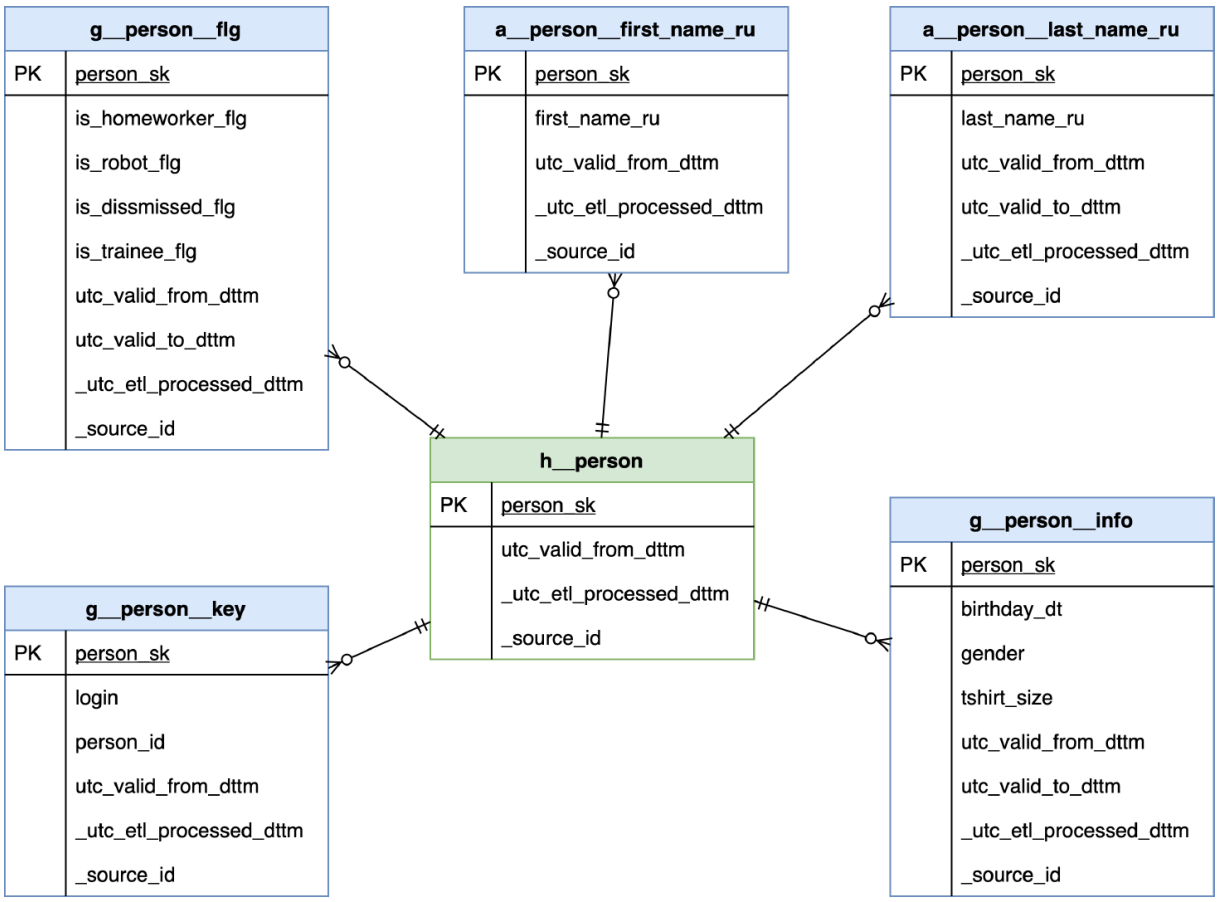

Рисунок 23 – ER диаграмма физической модели объявленной сущности с объединенными в группу атрибутами

```
class Department(Entity):

    __layout__ = DdsLayout(name='department', group='staff')
    department_id = Int(comment='ID в Стаффе', change_type=ChangeType.IGNORE)
    department_type = String(comment='Тип подразделения', change_type=ChangeType.NEW, group='base')
    name = String(comment='Название подразделения', change_type=ChangeType.NEW, group='base')
    level = Int(comment='Уровень в иерархии', change_type=ChangeType.NEW, group='base')
    description = String(comment='Описание', change_type=ChangeType.UPDATE, group='info')
    url = String(comment='Ссылка на стафф', change_type=ChangeType.UPDATE, group='info')

    __keys__ = [department_id]


class PersonDepartment(Link):

    __layout__ = DdsLayout(name='person_department', group='link')
    department = Element(entity=Department(), comment='департамент')
    person = Element(entity=Person(), comment='персона')

    __keys__ = [person]
```

Рисунок 24 – Скрипт объявление связи «Департамент» - «Сотрудник» в DDS-слое

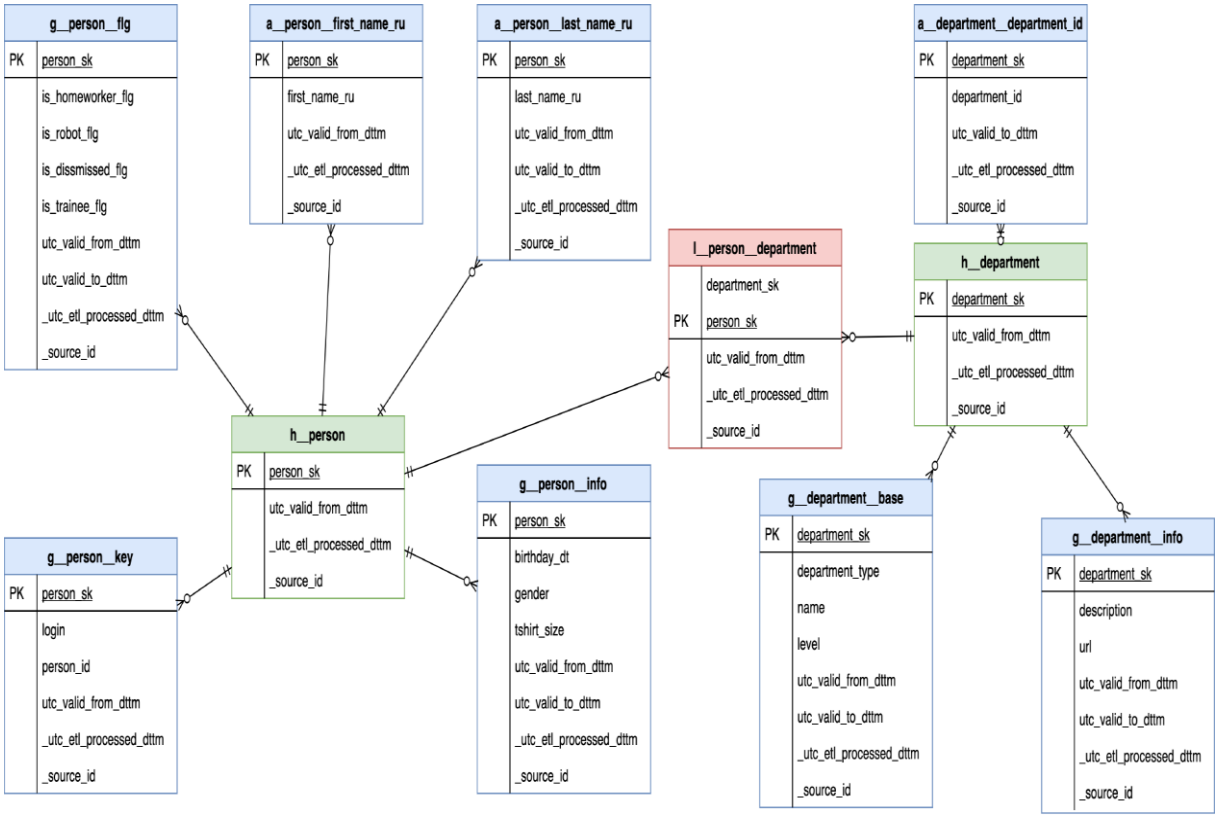

Рисунок 25 – ER диаграмма физической модели связи Департамент» - «Сотрудник»

3.3.6 Загрузка данных

Рассмотрим пример, как происходит загрузка данных.

На рисунке 26 представлена загрузка сущностей, на рисунке 27 – загрузка связей, на рисунке 28 – граф загрузки.

```
        HybridLoadGenerator(
            source=Stg,
            business_date_field=Stg.utc_business_dttm,
            source_system=SourceSystem.staff
        ).target(Person).mappings({
            Person.person_id: Stg.person_id,
            Person.first_name_ru: Stg.first_name_ru,
            Person.last_name_ru: Stg.last_name_ru,
            Person.login: Stg.login,
            Person.is_robot_flg: Stg.is_robot_flg,
            Person.gender: Stg.gender,
            Person.birthday_dt: Stg.birthday_dt,
            Person.tshirt_size: Stg.tshirt_size,
        })
```

Рисунок 26 – Загрузка сущностей

```
HybridLoadGenerator(
    source=Stg,
    business_date_field=Stg.utc_business_dttm,
    source_system=SourceSystem.staff
).target(Person).mappings({Person.login: Stg.login})\
    .target(Department).mappings({Department.department_id: Stg.department_id})\
    .target(Telegram).mappings({Telegram.login: Stg.telegram})\
    .target(Email, 'personal').mappings({Email.email: Stg.personal_email})\
    .target(Email, 'work').mappings({Email.email: Stg.email})\
    .target(LinkPersonDepartment)\
    .target(LinkPersonTelegram)\
    .target(LinkPersonOffice)\
    .target(LinkDepartmentOrganization)\
    .target(LinkPersonPersonalEmail).mappings(
    {LinkPersonPersonalEmail.person: Person,
     LinkPersonPersonalEmail.personal_email: 'personal'})\
    .target(LinkPersonWorkEmail).mappings(
    {LinkPersonWorkEmail.person: Person,
     LinkPersonWorkEmail.work_email: 'work'}
)
```

Рисунок 27 – Загрузка связей

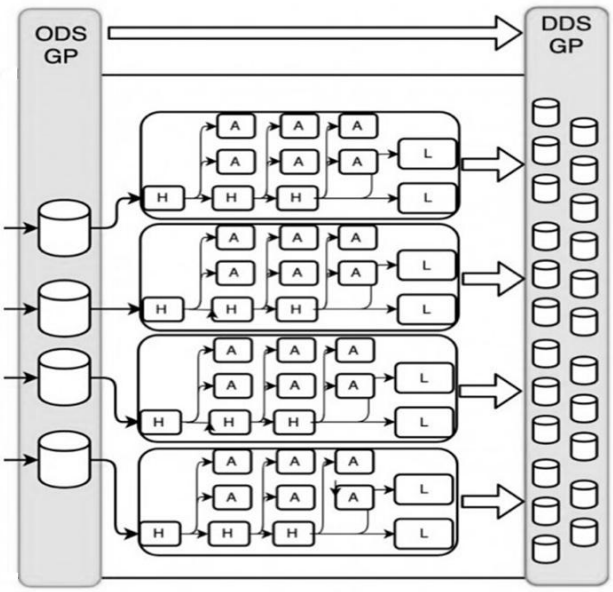

Рисунок 28 – Граф загрузки

Формируется модель загрузки, где каждая загрузка происходит с помощью сгенерированного для нее графа загрузки, представляющего собой атомарные загрузки атрибутов, соответствующим сущностям и связям. Хэш суррогатного ключа генерируется на каждом этапе загрузки [40] .

3.4 Тестирование разработанной модели хранилища данных

Применяя на практике разработанную модель корпоративного хранилища данных, пользователь имеет возможность конкретно обращаться к интересующей сущности, делать выборку нужной информации данных. При построении витрин обращение производится к DDS-слою.

Построение витрин на основе созданной схемы хранения данных возможно при помощи определенной СУБД или SQL запросов.

СУБД дает нам представление о сущности в виде полноценной таблицы, в которой срыты на какие группы и атрибуты разбиты данные. Применяя разработанную модель хранения удобно на предприятие использующего у себя СУБД.

На рисунке 29 представлена СУБД доступ к сущностям.

```
SELECT *
FROM get_entity_act(in_entity := 'person'
  , in_cols := '{person_id, gender, tshirt_size}'
  , in_result_table := 'tmp_person'
  , in_recreate_flg := TRUE
);

SELECT * FROM tmp_person;
```

	person_id	gender	tshirt_size
1	1	m	L
2	2	f	XS
3	3	f	S
4	4	m	M
5	5	m	L

Рисунок 29 – СУБД доступ к сущностям

Потребление данных из DDS-слоя происходит через визуализацию, что удобно. Для пользователя физическая реализация схемы построения базы данных скрыта. Доступ к сущностям происходит при использовании специальных процедур. Применяя специальные процедуры, мы получаем доступ к сущностям и имеем возможность посредством связи Link добавить другую сущность, расширяя таблицу. Учитывая, что запрашиваемые сущности могут быть большими, в данных процедурах указывается необходимое количество столбцов и только требуемые атрибуты.

На рисунке 30 представлена СУБД доступ к объединенным сущностям.

```
SELECT *
FROM add_entity_act(in_table := 'tmp_person'
  , in_link := 'person_department'
  , in_entity := 'department'
  , in_cols := '{department_id, department_type, description}'
  , in_result_table := 'tmp_person'
  , in_recreate_flg := TRUE
);

SELECT * FROM tmp_person;
```

	person_id				department_id	description
1	1	m	L	2	tmp	test
2	2	f	XS	2	tmp	test
3	3	f	S	1	tmp1	test
4	4	m	M	1	tmp1	test
5	5	m	L	1	tmp1	test

Рисунок 30 – СУБД доступ к объединенным сущностям

Применение SQL запросов при использовании данной модели происходит через запрос из DDS-слоя требуемой информации данных. На начальном этапе своего развития малое предприятие имеет не сильно загруженную базу данных. Поэтому пользователь может производить специальные запросы для конкретного случая и построение витрин через функции SQL. Для пользователя физическая реализация схемы построения базы данных скрыта.

Но в процессе роста бизнеса происходит расширение базы данных в хранилище, в данном случае использование простого SQL запроса может привести к сложности. Довольно трудно будет написать оптимальный запрос при большой генерации таблиц. Для этого на базе Python предлагается подготовить классы данных позволяющие производить SQL запрос к требующейся сущности и уже после применять запрос в построении витрины данных. В данном случае в коде указывается интересующие сущность и атрибуты, требуемые при дальнейшем выводе витрины данных [33].

На рисунке 31 представлено создание классов «департамент», «сотрудник» и класс их связи. На рисунке 32 представлен SQL запрос к сущностям.

```
def load():
    department = HistoryEntityCte(Department).project(
        Department.department_id,
        Department.department_type,
        Department.description
    )
    person = ActualEntityCte(Person).project(
        Person.person_id,
        Person.gender,
        Person.tshirt_size,
        Person.is_robot_flg,
        staff_login=Person.login
    )
    link_person_department = ActualLinkCte(LinkPersonDepartment)
```

Рисунок 31 – Создание классов «департамент», «сотрудник» и класс их связи

```
WITH department AS (
    {department}
), person AS (
    {person}
), link_person_department AS (
    {link_person_department}
)
SELECT *
FROM department d
  LEFT JOIN link_person_department lpdr
         ON d.id = lpdr.department
  LEFT JOIN person p
         ON p.id = lpdr.person
```

Рисунок 32 – SQL запрос к сущностям

При использовании разработанной схемы базы данных хранения данных в корпоративном хранилище решается задача оптимизации дискового пространства хранения информации.

На рисунке 33 представлено преобразование физической модели схемы базы данных.

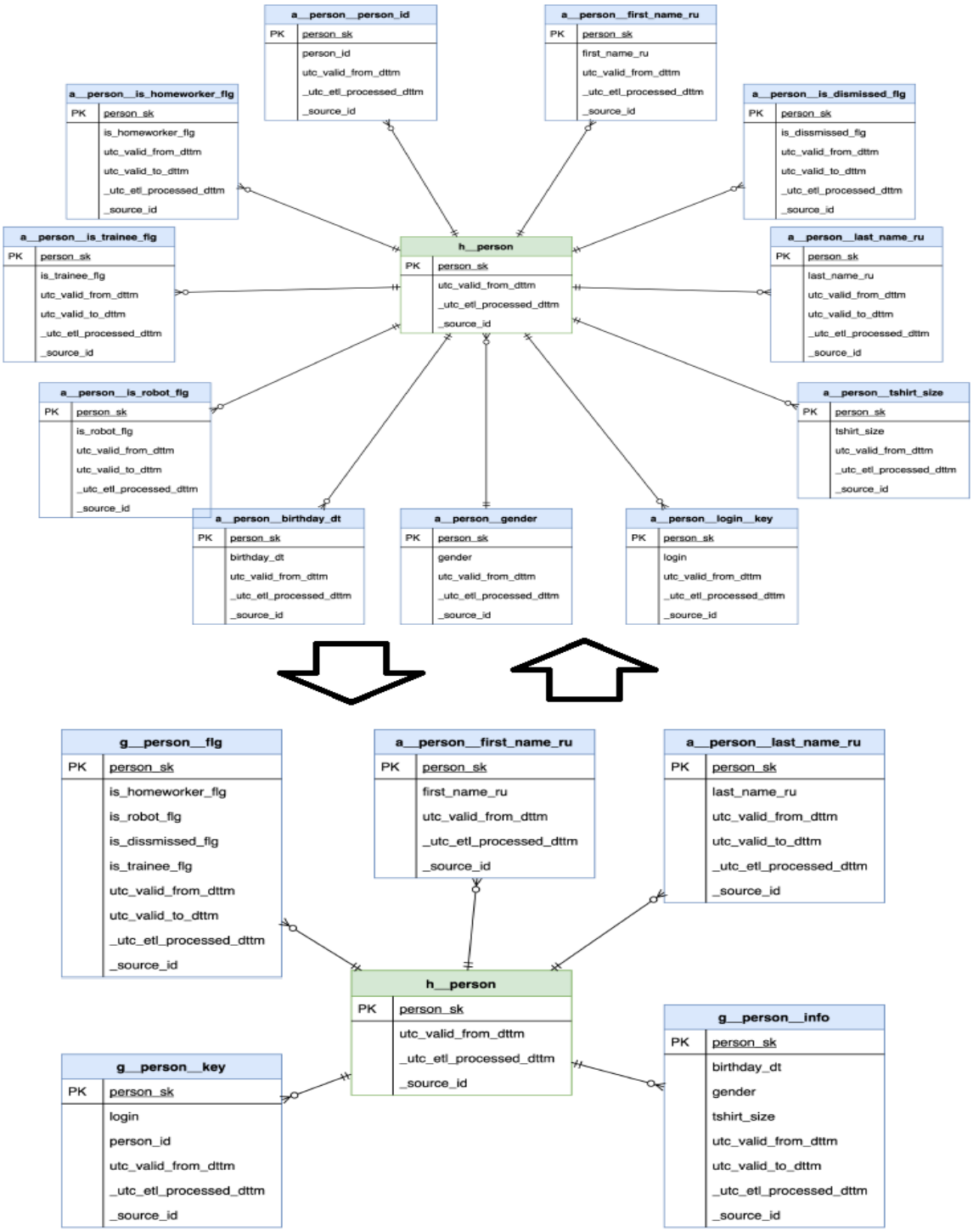

Рисунок 33 – Преобразование физической модели схемы базы данных

Построение витрин и аналитических запросов в данной модели реализуется на основе DDS слоя. Пользователи имеют возможность напрямую обращаться к данным через сущности. Архитектура DDS слоя решает задачи оптимизации, оптимально объединяя атрибуты по группам путем преобразования физической модели слоя DDS слоя, уменьшая объем хранения данных, что минимизирует занимаемое место на диске. Для этого вводятся операции, меняющие физическую схему, но логическая составляющая не изменяется. К примеру, объединение групп или атрибутов в группу, разъединение группы в набор группы или атрибутов. Меняется схема, но не меняют сама логика. Простыми действиями объединение, группирование, разъединение мы можем получить требуемое состояние физической модели, не изменяя логической модели [32].

Вывод по разделу 3

Применяя данную модель на практике, реализуется построение витрин, запросов и аналитики на основе детального слоя DDS. Пользователи имеют возможность напрямую обращаться к данным через сущности.

Построение витрин на основе модели может производиться через СУБД – все сущности представлены в виде виртуальной логической таблицы, которая скрывает все, что скрыто на физическом уровне, предоставляет значительное удобство при работе внутри СУБД.

При помощи запросов используя язык python, где доступ осуществляется с помощью генерации SQL на основе описания. Для этого вам нужно разработать класс, который генерирует запросы к определенному объекту, и использовать его для создания в хранилище визуализации и витрин запросов. Это удобно для использования в процессах загрузки.

Применяя модель, имеется возможность производить решение задач оптимизации, оптимально объединяя атрибуты по группам путем преобразования физической модели детального слоя DDS, чтобы уменьшить объем хранения данных. Таким образом, минимизируем занимаемое место на

диске. Для этого вводим операции, меняющие физическую схему, но логическая составляющая не изменяется. К примеру:

– объединение групп или атрибутов в группу;

– или разъединение группы в набор группы или атрибутов.

В методе применяются те операции, которые меняют схему, но не меняют саму логику. Данные действия простые (объединение, группирование, разъединение). Но с помощью их мы можем получить разное (требуемое) состояние физической модели, не изменяя логической модели.

В результате проделанного исследования были поставлены и решены 3 задачи:

– применение логического и физического проектирования;

– сокрытие физической составляющей созданной модели;

– оптимизация хранения информации.

При решении данных задач и потребовалось применения гибких подходов проектирования хранилища данных Data Vault и Anchor Modeling, а именно было взято то лучшее, на мой взгляд, из каждой методологии. Важно отметить, что дальнейший вектор развития разработанной модели перспективен, используя данную идею в перспективе можно применить не только для оптимизации хранения информации и требований по устойчивости к изменениям, масштабируемости и т.п., но и применить данный подход для оптимизации чтения данных, что существенно повлияет на жизнедеятельность хранилища данных.

Заключение

Создание корпоративного хранилища данных для малого предприятия может быть эффективным, если используется правильный архитектурный подход при его проектировании. В целом, корпоративное хранилище данных представляет собой централизованную базу данных, которая объединяет и интегрирует данные из различных источников, позволяя предприятию получить централизованный и консолидированный обзор своей информации.

Одним из архитектурных подходов, рекомендуемых для малых предприятий, является реляционная модель хранения данных. В этом случае данные организованы в таблицы с жестко определенными отношениями между сущностями. Реляционные базы данных, такие как MySQL, PostgreSQL или Microsoft SQL Server могут использоваться для создания корпоративного хранилища данных.

Чтобы построить эффективное корпоративное хранилище данных, необходимо следовать нескольким шагам:

- анализ бизнес требований. Сначала необходимо определить, какие данные требуется хранить и анализировать в хранилище данных. Это может включать в себя данные о продажах, клиентах, финансах и т.д. Важно также определить требования к производительности и масштабируемости;
- проектирование схемы данных. На основе анализа бизнес-требований, необходимо разработать структуру базы данных для хранения данных. Моделирование данных помогает определить таблицы, столбцы и связи между ними;
- интеграция данных. Для создания централизованного хранилища данных, предстоит полностью наладить взаимосвязь по интеграции информации данных из имеющихся источников. Данная процедура предусматривает извлечение информации данных с различных баз данных, определённые файлы и системные объекты;

– увеличение и балансировка производительности. Для обеспечения эффективной работы хранилища данных, необходимо провести оптимизацию производительности. Это может включать в себя индексирование таблиц, оптимизацию запросов и настройку оборудования;
– разработка ETL-процессов. Разработка эффективных ETL-процессов позволяет обновлять и обрабатывать данные в хранилище данных;
– разработка отчетов и аналитических инструментов. Наконец, необходимо разработать отчеты и инструментую часть для анализа данных в хранилище данных. Современные хранилища данных в своем арсенале имеют способность предоставить для анализа созданные отчеты, информационные панели мониторинга и аналитические запросы.

Создание корпоративного хранилища данных может быть сложным процессом, но правильный архитектурный подход позволяет малому предприятию получить значительные преимущества в организации данных, анализе и принятии решений. Для успешного внедрения корпоративного хранилища данных требуется хорошая планировка, анализа бизнес требований и оптимизации производительности.

Включая в разработку гибкие методологии Data Vault и Anchor Model на начальном этапе создания хранилища данных на малом предприятии, имеет ряд значимых преимуществ. Эти методологии позволяют создать гибкую и масштабируемую базу данных, которая способна адаптироваться к изменениям требований и бизнес-процессов предприятия. Методы предоставляют возможность добавлять, изменять и удалять данные без необходимости перестраивать всю структуру базы данных. Это позволяет предприятию быстро реагировать на изменения внешних условий и требований рынка. Применение этих методологий помогает сформировать четкую и однозначную модель данных. Data Vault и Anchor Model используют строгие правила и стандарты при проектировании базы данных,

что помогает избежать путаницы и неоднозначности в структуре данных. Это значительно облегчает работу с базой данных, упрощает разработку запросов и отчетов, а также улучшает качество данных.

Методологии Data Vault и Anchor Model способствуют созданию гибкой и расширяемой архитектуры данных. Они позволяют создать хранилище данных, которое может интегрировать большое количество источников данных и оперировать большим объемом информации. Такая архитектура облегчает доступ к данным и улучшает их качество. Применение Data Vault и Anchor Model позволяет предприятию минимизировать риски при создании хранилища данных. Эти методологии предусматривают разделение данных на независимые компоненты, которые могут разрабатываться и внедряться поэтапно. Это позволяет уменьшить возможные затраты и риски при разработке и внедрении базы данных.

Таким образом, применение гибких методологий Data Vault и Anchor Model на начальном этапе создания хранилища данных на малом предприятии имеет существенную значимость. Они позволяют создать гибкую, масштабируемую и высококачественную базу данных, которая способствует эффективному управлению данными и успешному развитию предприятия.

Предложенная модель хранения информации в разделе №3 данного исследования, основанная на гибридном подходе проектирования архитектуры хранилища данных из подходов Data Vault и Anchor Model, обладает несколькими преимуществами.

Во-первых, подход Data Vault обеспечивает гибкость и масштабируемость хранилища данных, что позволяет легко добавлять новые источники данных и вносить изменения в существующую структуру. Это особенно полезно в случае быстрого развития бизнеса, когда требуется быстрая интеграция новых данных.

Во-вторых, подход Anchor Model предоставляет четкую и гибкую семантику моделирования данных. Он позволяет определить основные

понятия и связи между ними, что облегчает понимание структуры данных и упрощает разработку запросов. Кроме того, Anchor Model обеспечивает эффективное использование памяти и процессора, что повышает производительность хранилища данных.

Комбинация методов создания логической схемы базы данных в хранилище data vault и anchor model позволяет построить стойкую и гибкую архитектуру. Data vault создает нужную структуру для сохранения всех данных без потерь, в то время как anchor model придает ясность и определенность в логике модели. Данный подход обеспечивает эффективное управление данными и структурными изменениями, обеспечивая отказоустойчивость и возможность масштабирования.

Применение гибридного подхода к проектированию архитектуры хранилища данных на основе методов data vault и anchor model приносит много преимуществ, таких как гибкость, расширяемость, ясность семантики модели и эффективное управление изменениями. Эта концепция идеально подходит для компаний, желающих создать продуктивное и надежное хранилище данных для бизнес-приложений на начальной стадии своего развития.

Создание эффективно и грамотно спроектированной модели корпоративного хранилища данных на начальном этапе станет фундаментом успеха и развития предприятия. Она обеспечит надежное хранение информации, позволит предприятию экономить средства на обслуживании и развитии системы, а также обеспечит доступность точной и необходимой информации в ходе бизнес-процессов.

Список используемой литературы

1. Разработчик моделей данных для профессионалов Data Vault 2.0 Modeling Basics [Электронный ресурс]. – Режим доступа: vertabelo.com – Дата доступа: 18.11.2022.

2. IT-Эксперт журнал, Москва, №04 (292) апрель-май 2020г.

3. Linstedt D. Building a Scalable Data Warehouse with Data Vault 2.0 / M. Kaufmann. – Elsevier, 2015. – 684 с.

4. Арутюнян А. П. Методология проектирования архитектуры многомерной базы данных. Москва: Издательский дом «Фазис», 2008.

5. Базинский Ю. А., Безсмертный А. К. Современные тенденции в области хранения данных: монография. М.: Изд-во Национального исследовательского университета высшая школа экономики, 2019.

6. Бизяев Н.О. Инновационные методы сохранения информации в информационных системах [Текст] / Н.О. Бизяев, И.С. Заварзин. - М. : Издательство РГО, 2016. - 195 с.

7. Волков Б. В. Анализ методологий проектирования архитектуры логической схемы хранилища данных. М. : Издательский дом «Лоссарио», 2010.

8. Гришин С. Хранение данных в распределенных системах: учебное пособие. М.: Издательский дом «Русская редакция», 2017.

9. Джонсон Р. В. Проектирование хранилищ данных: методология и практика. М. : Издательский дом «БИНОМ», 2012.

10. Ершов С. П., Бестубаев М. В. (2019). Анализ современных тенденций развития систем хранения данных. Журнал Технологии банковского дела, (4), 47-51.

11. Иванов А. А. Архитектура и проектирование базы данных. М. : Издательство «Форум», 2011.

12. Карташов А.Е. Многомерные методы защиты информации [Текст] / А.Е. Карташов, В.В. Ионова. - СПб. : Издательство "СПбГУ", 2016. - 202 с.

13. Карташов А. С. Методология создания инфологической модели данных для хранилища данных. М. : Издательский дом «Академия», 2013.

14. Караханян А. В., Петров В. М. (2018). Методы и алгоритмы хранения и обработки данных. М.: Издательский дом Кодекс.

15. Ляжко И. А., Ивченко Г. Л., Журавлева Е. В. (2016). Распределенные системы хранения данных: учебник. М.: Издательстко-торговый дом «Дашков и Ко».

16. Маслова Е. В. Методология проектирования многомерной базы данных: теория и практика. М. : Издательство «Академический проект», 2006.

17. Новиков А. Ю. Проектирование многомерной базы данных: архитектура и методология. СПб. : Издательство «Питер», 2015.

18. От хранения данных к управлению информацией. 2-е изд. – СПб.: Питер, 2016. – 544 с.: ил.

20. Петров К. В. Архитектура многомерной базы данных: основы проектирования и разработки. М. : Издательский дом «Диалектика», 2010.

21. Работа с BigData в облаках. Обработка и хранение данных с примерами из Microsoft Azure. — СПб.: Питер, 2019. — 448 с.: ил. — (Серия «Для профессионалов»).

22. Радзиховский О. М. Проектирование схем многомерной базы данных: методы и подходы. М. : Издательство «Физматлит», 2009.

23. Смирнов В. Д. Методология проектирования логической схемы многомерной базы данных. СПб : Издательство СПбГУ, 2013.

24. Скляр, А. (2010). Системы хранения и обработки информации. М.: Издательский дом "Вильямс".

25. СУБД (OLAP) технологии, конспект лекций для студентов, Алматы 2019.

26. Третьяков Д. И. Анализ архитектурных методологий проектирования многомерной базы данных. Москва: Издательство «Знание», 2011.

27. Телегина, И. В. (2018). Моделирование распределенных систем хранения данных: монография. Уфа: РПТЦ УГНТУ.

28. Филатов, В. (2016). Методы и технологии организации хранения и обработки данных в информационных системах. Вестник Мордовского университета, (1), 69-74.

29. Хранилища данных и средства бизнес-аналитики: учебное пособие / Т.Е. Точилкина, А.А. Громова – М.: Финансовый университет, 2017. – 161 с.

30. Широков А.И. Технологии и методы защиты информации в локальных сетях [Текст] / А.И. Широков. - М.: Питер, 2015. - 336 с.

31. [Электронный ресурс]. – Режим доступа: https://www.osp.ru/lan/2014/10/13043206?ysclid=laqwgxnml6969877135. Дата доступа: 24.11.2022.

32. [Электронный ресурс]. – Режим доступа: http://iso.ru/ru/press-center/journal/1861.phtml. Дата доступа 21.09.2023

33. [Электронный ресурс]. – Режим доступа: https://timi.eu/blog/data-vaulting-from-a-bad-idea-to-inefficient-implementation/ Дата доступа 18.11.2022

34. [Электронный ресурс]. – Режим доступа: https://www.altexsoft.com/blog/best-bi-tools-comparison/ для EWD, Дата доступа 01.11.2023

35. [Электронный ресурс]. – Режим доступа: https://www.vmware.com/ru/topics/glossary/content/hybrid-cloud.html Дата доступа 18.11.2023

36. [Электронный ресурс]. – Режим доступа: https://cyberpedia.su/16xad34.html Дата доступа 28.11.2023

37. "Information Storage and Management: Storing, Managing, and Protecting Digital Information in Classic, Virtualized, and Cloud Environments" by EMC Education Services: Wiley, 496 с.

38. "Managing Gigabytes: Compressing and Indexing Documents and Images" by Ian H. Witten, Alistair Moffat, and Timothy C. Bell: Morgan Kaufmann, 560 c.

39. "Enterprise Content Management: A Business and Technical Guide" by Anne Lapkin: Auerbach Publications, 384 c.

40 "Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems" by Martin Kleppmann: O'Reilly Media, 616 c.

41 "Information Technology for Management: Advancing Sustainable, Profitable Business Growth" by Efraim Turban, Carol Pollard, and Gregory Wood: Wiley, 512 c.