Столыпинский вестник

Научная статья
Original article
УДК 004.415.2

РАЗРАБОТКА АРХИТЕКТУРЫ СИСТЕМЫ УПРАВЛЕНИЯ ИНФОРМАЦИОННЫМИ РЕСУРСАМИ В РАМКАХ МОДЕЛИ ХРАНИЛИЩА ДАННЫХ DATA VAULT 2.0

DEVELOPMENT OF THE ARCHITECTURE OF THE INFORMATION RESOURCE MANAGEMENT SYSTEM IN THE DATA VAULT 2.0 DATA WAREHOUSE MODEL

Конаков Павел Олегович, магистрат, МИРЭА — Российский технологический университет, г. Москва
Смоленцева Татьяна Евгеньевна, доктор технических наук, доцент кафедры практической и прикладной информатики, МИРЭА — Российский технологический университет, г. Москва

Konakov Pavel Olegovich, Graduate Student, MIREA — Russian Technological University, Moscow
Smolenceva Tatjana Evgenjevna, Doctor of Technology, Associate Professor of the Department of Practical and Applied Informatics, MIREA — Russian Technological University, Moscow

Аннотация

При разработке ETL-процессов в рамках корпоративного хранилища данных стоит задача организации доступа к таблицам. В процессе анализа модели Data

Vault 2.0 была выявлена проблема конкурирующего доступа к группе таблиц, связанных с генерацией и получением актуальных суррогатных ключей системы. В качестве решения было предложено спроектировать дополнительный слой управления ресурсами предприятия, основанный на концепции открытия и закрытия транзакций при работе ETL-потоков. Спроектированное решение бесшовно интегрируется в архитектуру корпоративного хранилища данных, поскольку менеджер ресурсов реализуется по сервис-ориентированной архитектуре, и позволит организовать последовательный доступ к ресурсам хранилища, повысив согласованность данных в рамках всего КХД.

Annotation

When developing ETL processes within the corporate data warehouse, the task is to organize access to tables. In the process of analyzing the Data Vault 2.0 model, the problem is competing access to a group of tables related to the generation and receipt of actual surrogate keys of the system. As a solution, it was proposed to design an additional layer of enterprise resource management based on the concept of opening and closing transactions during ETL-flows. The designed solution seamlessly integrates into the architecture of the corporate data warehouse, since the resource manager implemented as a service-oriented architecture, and will allow consistent access to storage resources, increasing data consistency throughout the DWH.

Ключевые слова: большие данные, корпоративное хранилище данных, Data Vault 2.0, ETL-процесс, управление потоками данных

Keywords: big data, corporate data warehouse, Data Vault 2.0, ETL-process, data flow management

Введение

Современное общество находится в среде больших объёмов данных, которые необходимо обрабатывать с целью оперативного и грамотного принятия решения в разных сферах деятельности: производственной, банковской, страховой, финансовой и др.

При разработке корпоративных хранилищ данных стоит задача проектирования и разработки высоконагруженной системы, которая должна обеспечивать стабильную параллельную обработку больших объёмов данных, которые поступают в систему из разных источников. Для решения данной задачи существует множество современных подходов и моделей хранилищ, позволяющих в той или иной степени оптимизировать процесс хранения и обработки данных. Однако перед этими решениями стоит проблематика грамотной организации доступа к данным внутри хранилища, которую можно решить лишь организацией самой модели данных, в рамках которой доступ к таблицам между несколькими потоками не пересекается, или же при помощи проектирования вспомогательного средства по управлению этим доступом между потоками.

Проектирование системы управления ресурсами корпоративного хранилища данных не может иметь универсальной из-за специфики используемых моделей, поэтому стоит рассмотреть на примере конкретной концепции моделирования хранилища данных, например, Data Vault 2.0.

Сущность модели хранилища данных Data Vault 2.0

Модель Data Vault является одной из наиболее гибридных и адаптивных моделей, позволяя вести разработку в командах по гибридным методологиям в процессе постоянно меняющихся требований к системе. Поскольку все новые разрабатываемые сущности создаются отдельно для каждого источника данных, несколько команд могут вести разработку для смежных бизнес-сущностей одновременно, не пересекаясь друг с другом, не образуя блокировок и зависимостей между многочисленными командами разработки. Данная модель хранилища данных имеет свою терминологию и конкретные правила организации хранимых сущностей (Таблица 1).

Таблица 1 — Описание сущностей модели Data Vault

Сущность	**Описание**
Хаб (Hub)	Ключевая сущность, которая описывает основные объекты информационного пространства организации. К Хабам относятся в основном те бизнес-сущности, которые чаще всего участвуют в бизнес-

	процессах предприятия (клиент, сотрудник, заявка и пр.). Главное предназначение Хаба – хранение и формирование суррогатных ключей для объединения данных из разных источников по одной сущности в единую систему ассоциации объектов [1].
Комета (Satellite)	Сущность хранилища данных, описывающая характеристики одной из ключевых сущностей. В качестве первичного ключа в данной сущности используется не первичный ключ записи, полученной с источника данных, а суррогатный ключ, полученный из сущности, для которой представляется описание. Комета является одним из источников генерации новых суррогатных ключей, поскольку по сути своей она и описывает сущность с заданным атрибутным составом. Поскольку разные источники данных могут описывать одну и ту же бизнес-сущность разными атрибутами, принято, чтобы для каждого источника данных формировалась отдельная сущность Кометы [1].
Связь (Link)	Сущность, описывающая связь между Хабами. Для описания взаимосвязи между ключевыми бизнес-сущностями, например, «Договор-Клиент», «Заявка-Клиент», «Сотрудник-Подразделение» и других используются сущность Связь. Суррогатный ключ связи формируется на основе набора суррогатных ключей, формируемых в Хабах сущностей, соответствующей формируемой связи. Для описания атрибутов связи также используется Комета, в рамках которой ключом записи является суррогатный ключ, генерируемый в Связи [2].

Поскольку модель достаточно гибкая и позволяет быстро реагировать на изменения требований к формируемым данным, модель также может расширяться дополнительными видами сущностей, которых нет в стандартизированной модели, например, справочная информация. Поскольку эти данные достаточно статичные и не стоит хранить для них историю изменений, их стоит загрузить в хранилище только один раз и не отслеживать их обновление в качестве регламентного процесса, поскольку служат только для формализованного представления атрибутов в рамках отчётов.

Для более детального описания необходимо рассмотреть наглядный пример моделирования сущностей на основе таблицы источника. В качестве абстрактного примера была взята сущность, описывающая заказ из приложения по доставке еды. Некоторые связи с данной сущностью могут показаться очевидными, например, «Заказ-Продукция», но они были опущены с целью наглядности и упрощения рассматриваемого примера. В качестве присутствующих связей были выбраны связи с Клиентом, Сотрудником, а также Статусом заказа, чтобы

продемонстрировать принцип работы со справочной информацией в данной модели. Упрощённая модель заказов сервиса поставки представлена на Рисунке 1.

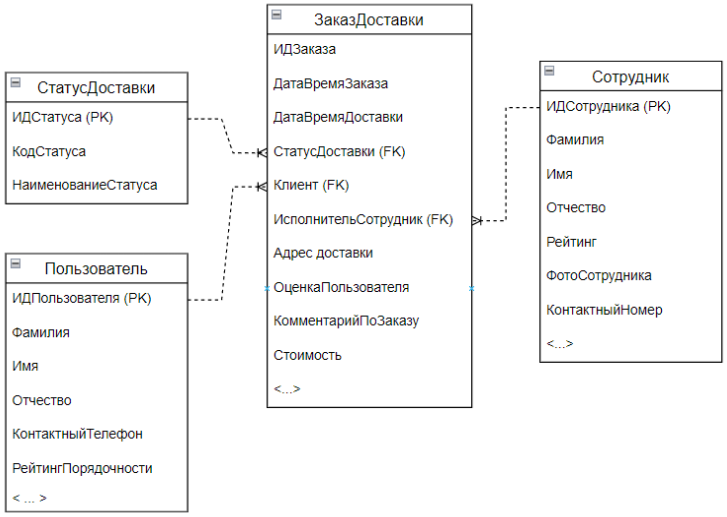

Рисунок 1 — Исходная упрощённая модель данных заказов сервиса доставки

Из упомянутых ранее связей с другими сущностями, такими как Клиент (Пользователь) и Сотрудником (Исполнителем заказа), и самой сущности Заказ можно сформировать ключевые сущности (Хабы). Вокруг них уже необходимо выстроить Связи, а с учётом описательных возможностей атрибутов сформировать соответствующие Кометы. Результат формирования модели в соответствии с Data Vault представлен на Рисунке 2.

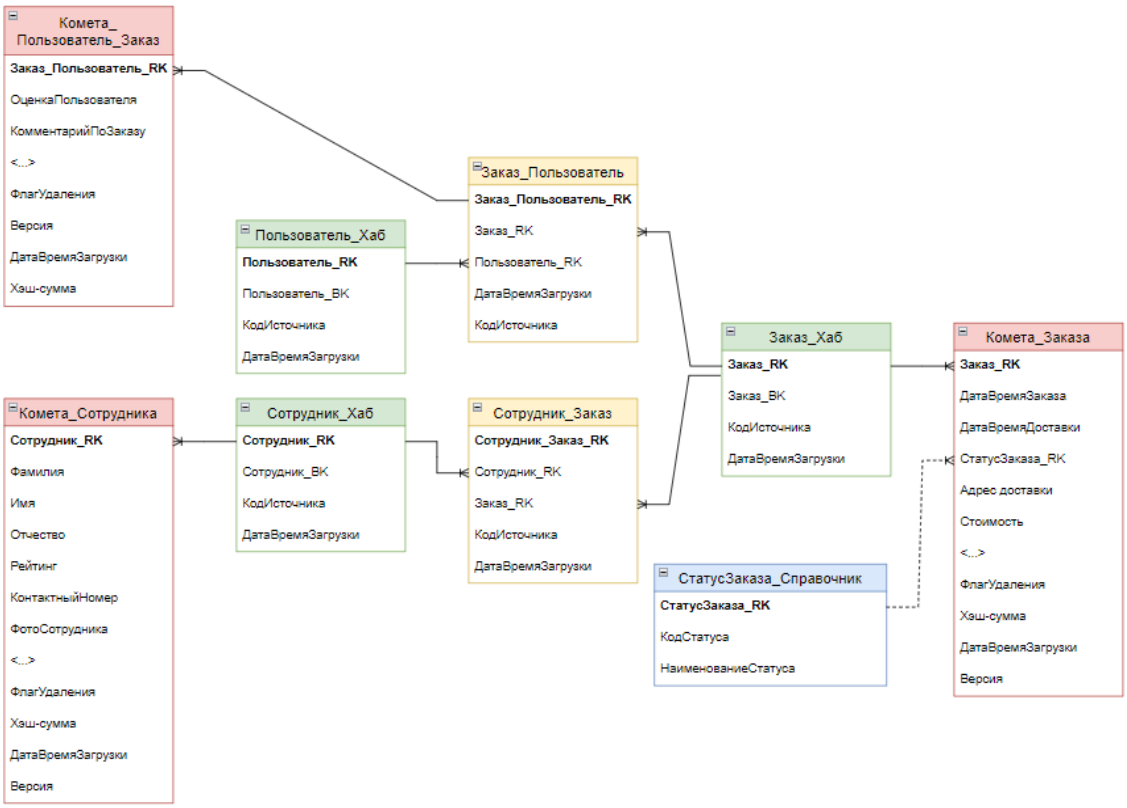

Рисунок 2 — Результат перехода модели к Data Vault

Data Vault позволяет расширять логическую модель данных, добавляя новые описательные сущности, привязывая их к уже существующим Хабам и Связям, например, заказы такси, заказы еды из ресторана и т.д. Если же текущей логической модели не будет хватать для реализации загрузки данных из источника, её всегда можно расширить, добавив новые ключевые и связующие сущности. Стоит отметить, что у данной модели нет строгих правил формирования и распределения сущностей в рамках моделей, что позволяет каждой компании самостоятельно выстраивать правила, которые она будет соблюдать при моделировании новых сущностей в случае появления соответствующей потребности.

Рассматриваемая модель имеет схожие черты с якорным моделированием хранилища данных (Anchor Modeling). Она также имеет ключевые бизнес-сущности, связи между ними и атрибутный состав. Ключевым отличием от модели Data Vault является то, что в якорной модели историчность каждого атрибута сущности прослеживается отдельно, когда в Data Vault историчность соответствует формату историчности SCD2 и отслеживание изменения объектов происходит на основе сверки хэш-сумм значений. В результате такого подхода хоть и достигается

меньшая гибкость хранимых сущностей, однако из-за меньшего количества таблиц уменьшается количество соединений для формирования отчёта (поскольку каждый атрибут имеет свой формат историчности, типы и условия соединения таблиц будут значительно различаться между собой, уменьшая производительность выполнения запросов).

Модель хранилища данных Data Vault 2.0 предполагает использование в ИТ-инфраструктуре источников данных разного рода, не только реляционных. Такая интеграция возможна, поскольку многие модели хранения данных возможно интерпретировать к вычислению хэш-суммы для текущего объекта с целью отслеживания изменения объектов и формирования историчности [4].

Принцип работы ETL-процессов в рамках Data Vault 2.0

ETL-процесс – технологический процесс по выгрузке, обработке и загрузки данных между объектами хранилища данных. Как видно из названия, данных процесс содержит в себе три основных этапа: выгрузка данных (Export), преобразование данных (Transform), загрузка данных (Load), — однако в зависимости от выбранных подходов к модели и способу хранения данных эти этапы могут содержать в себе отличительные функциональные особенности. Поскольку в системе присутствует несколько концептуальных сущностей хранилища данных, они будут иметь некоторые различия по перечню объектов, в которые будут проводиться загрузка данных, генерация суррогатных ключей и пр. Общая последовательность ETL-процессов представлена на Рисунке 3.

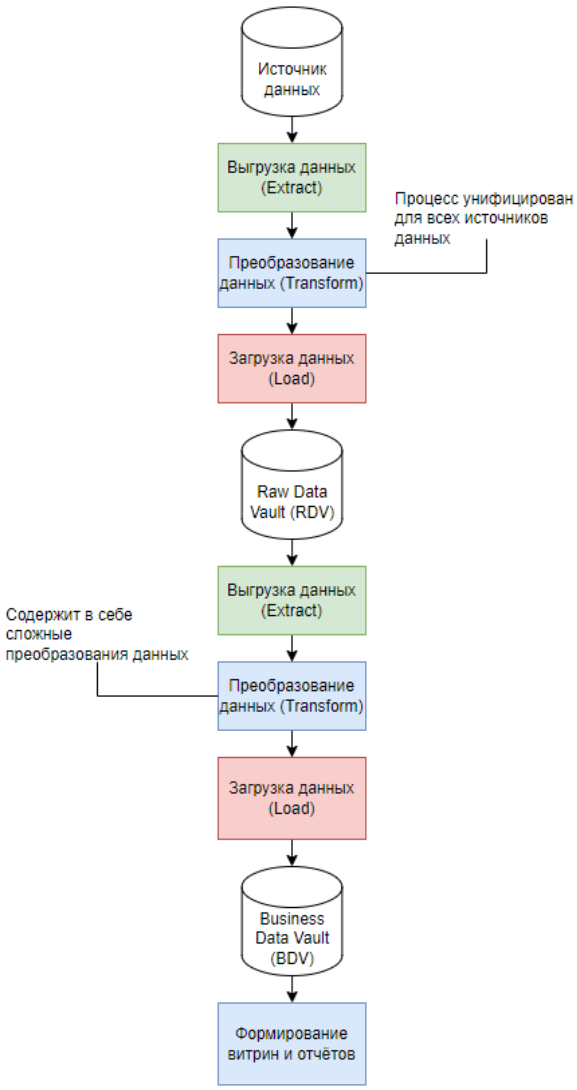

Рисунок 3 — ETL-процесс в рамках модели Data Vault 2.0

Этап выгрузки данных представляет из себя процесс по чтению данных с какого-либо информационного источника во временную таблицу [3]. Область, в которой хранятся временные таблицы для подготовки данных, обычно называют Staging Area (подготовительный слой). В этот слой выгруженные данные попадают «как есть» с попыткой максимального сохранения точности в данных. Данные, выгруженные в подготовительный слой, перед загрузкой в хранилище данных, должны пройти этап преобразований, чтобы между данными из источника и данными в корпоративном хранилище данных не возникало противоречий, дубликатов и других возможных при загрузке аномалий.

Этап преобразований служит этапом трансформирования полученных с источника данных к модели хранилища данных [3]. В случае с моделью Data Vault 2.0 данный этап содержит в себе следующие виды преобразований:

1) **Приведение форматов данных в соответствии со стандартами, установленными в хранилище данных.** Данный аспект в первую очередь касается временных данных (согласованность формата представления дата-времени, обозначение плюс и минус бесконечности и др.)
2) **Разбиение таблицы источника на сущности Data Vault**. Поскольку модель состоит из специфичных сущностей, вряд ли таблицы источника будут иметь схожий атрибутный состав, с этой целью существует данный этап, чтобы разделить исходную таблицу по сущностям, которые содержатся в модели хранилища данных (Хабы, Связи и Кометы).
3) **Генерация суррогатных ключей полученных сущностей.** Для интегрирования данных, полученных с источника, в хранилище необходимо выполнить преобразование внутренних ключей к суррогатным ключам КХД. Это делается с той целью, чтобы унифицировать все данные по одной сущности от разных источников в одном месте. В зависимости от сущности, в которую происходит загрузка данных, генерация суррогатных ключей будет отличаться.
4) **Формирование дельты изменений.** После насыщения данных суррогатными ключами стоит определить, какие данные из загруженной таблицы дублируют то, что есть в хранилище, какие могли поменять значения или вовсе добавиться новые. Для этого необходимо сформировать «дельту» данных, которая будет подгружена в хранилище. Сравнение данных происходить по суррогатным ключам на основе хэш-суммы, сформированной на основе описательных атрибутов сущности.

Этап загрузки данных основан на комплексной загрузке «дельт», сформированных с нескольких потоков, при помощи соответствующего менеджера загрузки данных. В результате загрузки хранилище содержит в себе новые

актуальные данные для формирования данных, используемых в отчётах и витринах [3]. Слой данных, формируемый в результате выгрузки данных с источника “как есть”, называется Raw Data Vault (RDV), а слой витрин и бизнес-моделей данных – Business Data Vault (BDV). Для данного слоя также разрабатывается ETL-процесс, имеющий схожую структуру, однако формируется уже более сложными трансформациями, процесс которых нельзя унифицировать, поскольку бизнес-ценность каждой сущности в хранилище можно оценивать абсолютно по-разному для каждой из задач, в которой требуется аналитика данных.

Как видно из представленного описания, в ETL-процессах участвует сразу несколько объектов для загрузки, поскольку данные надо упорядочить и систематизировать в представлениях модели Data Vault. Поскольку в рассматриваемой модели присутствуют сущности, которые чаще остальных участвуют в процессе (Хабы и Связи), в процессе трансформаций с слоя RDV в слой BDV в системе может возникнуть конкуренция по доступу к ним по причине блокировки транзакций для грамотной загрузки данных.

Узкое место реализации ETL-потоков для Data Vault 2.0

Одной из проблем при работе с хранилищем данных модели Data Vault 2.0 является достаточно высокая конкуренция доступа к таблицам Хабов и Связей. Это связано с тем, что поиск взаимосвязей между сущностями в данной модели хранилища данных всегда выносится в отдельную таблицу вне зависимости от того, идентифицирующая ли эта связь или нет. В результате чего, многие ETL-потоки будут требовать для своей работы одной и той же таблицы для чтения и записи. Для лучшего описания ситуации, стоит рассмотреть пример работы нескольких потоков, конкурирующих за доступ к одной таблице.

В рамках примера существует 4 потока, которые запускаются в своё время по расписанию и перезапускаются через фиксированное время, чтобы попробовать выполнить процесс заново. Во время работы с таблицей происходит блокировка транзакций других пользователей, чтобы процесс чтения и записи из таблицы происходил в статичном состоянии (например, это необходимо при чтении и

записи суррогатных ключей из Хаба, чтобы сформировать новый суррогатные ключи для новых записей). Для наглядности на Рисунке 4 представлена схема, на которой демонстрируется пример работы поток с таблицей во времени.

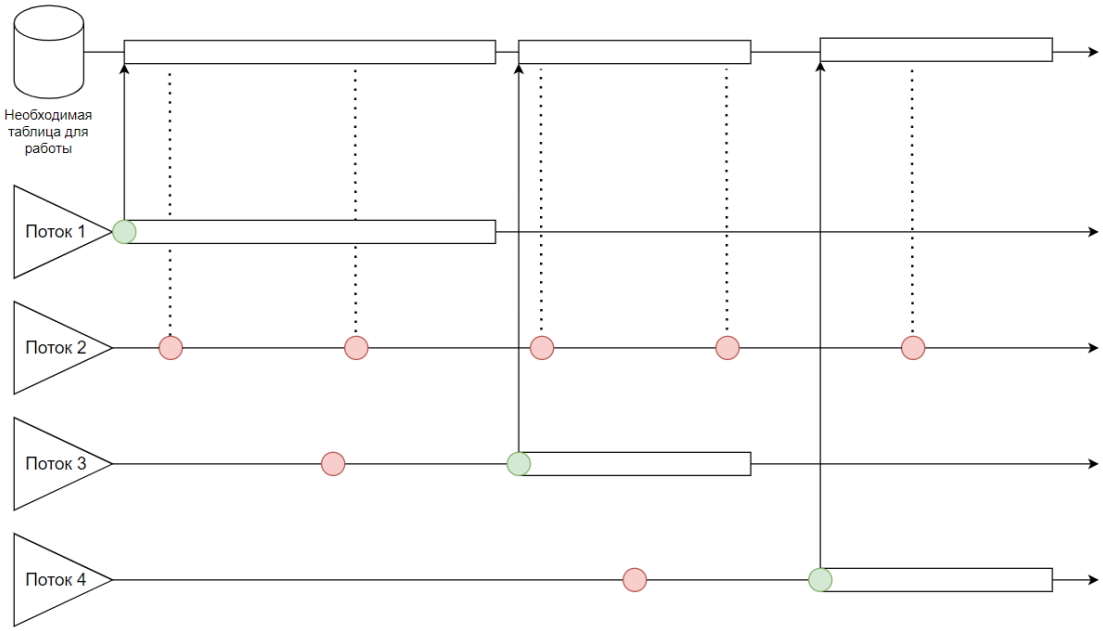

Рисунок 4 — Пример работы потоков с доступом для одной таблицы

Круги на временных линиях потоков обозначают запрос на подключение к таблице для получения доступа к данным. Цвет означает успешность этого действия: зелёный говорит о том, что подключение выполнено успешно, красный – произошла блокировка транзакции. Прямоугольные участки на линиях описывают работы потоков с таблицей в определённый момент времени.

Из данного примера видно, что второй поток, запущенный после первого, постоянно пытается подключиться к базе, но постоянно перезапускается по причине отсутствия доступа к данной таблице по причине блокировки транзакций. Возникла ситуация, в рамках которой третий и четвертые потоки смогли обратиться к базе в тот момент, когда она была свободна от блокировок, в результате чего второй поток снова оказался в ситуации, когда отсутствует доступ к таблице, хотя по сути должен был выполнить свои действия раньше потоков 3 и 4. И данная ситуация может продолжаться достаточно долго, в результате чего некоторые важные для аналитики данные могут дойти до хранилища с большой задержкой, в результате чего могут быть упущены важные детали, которые могли сыграть свою роль при принятии решений.

В рамках заявленной в работе проблемы предлагается проектирование нового информационного слоя управления ресурсами корпоративного хранилища данных, чтобы потоки выгрузки данных могли быть организованы в очередь на доступ к объектам хранилища.

Ресурсный менеджер как система управления доступом ETL-потоков к данным

Под термином «ресурс» понимается набор данных, который может быть использован в рамках ETL-процессов или же является конечным результатом его выполнения. Он не ограничивается просто таблицей в той или иной РСУБД, поскольку модель Data Vault 2.0 подразумевает работу с разными источниками данных, в том числе и с NoSQL базами данных, статическими файлами, вычисляемыми таблицами при помощи других данных и т. д. Ресурс является целостным описанием объекта данных, поскольку включает всю информацию о расположении и определении набора данных, а также перечень свойств, среди которых статус блокировки объекта и информация о предыдущем использовании объекта. Система, обращаясь за статусом ресурса, сможет понять, сможет ли она им воспользоваться в рамках работы ETL-потока или нет.

Если же ресурс заблокирован другим потоком, поток включается в “очередь” на доступ. В результате такой организации доступа потоки будут получать доступ к необходимым данным в правильном порядке – в порядке обращения за доступом.

Помимо механизма предоставления доступа к ресурсам должна быть определена и методика блокировки ресурсов. Для этого была использована концепция, которая и так реализована в СУБД, однако будет организована на более высоком уровне управления данными — транзакции. Под транзакцией будет пониматься единый ETL-процесс, затрагивающий все целевые таблицы сразу. По сути в рамках работающего потока создаётся одна транзакция, которая и будет определять процесс блокировки ресурсов. Транзакция, получая через поток доступ к ресурсу, изменяет его на соответствующее состояние. В момент окончания выполнения всех манипуляций над ресурсами, вне зависимости от результата

отработки потока (успешного или ошибочного), транзакция закрывается, все использованные ресурсы разблокируются для доступа другими потоками, которые находились в момент работы потока в ожидании данных ресурсов. Стоит также отметить, что для утверждения закрытия транзакции в целях безопасности необходимо использовать дополнительный ключ транзакции, которые формируется при создании данной транзакции. Это сделано с той целью, чтобы у других пользователей или потоков не было возможности преждевременно закрыть транзакцию и получить доступ к ресурсу. Помимо этого, у всех транзакций существует время жизни. То есть, если транзакция будет существовать дольше определённого времени, то она будет считаться автоматически завершённой.

Взаимодействие между транзакциями и ресурсами можно организовать при помощи паттерна проектирования “Наблюдатель” (Observer), в рамках которого объекты, подписанные на изменения другого объекта (в данном случае Ресурсы “подписываются” на изменения Транзакции) будут выполнять действия в случае возникновения определённого события у Наблюдаемого. Однако в данном случае существует логическое ограничение того, что наблюдатель может быть подписан только на одного наблюдаемого, что фиксируется в состоянии ресурса (Рисунок 5).

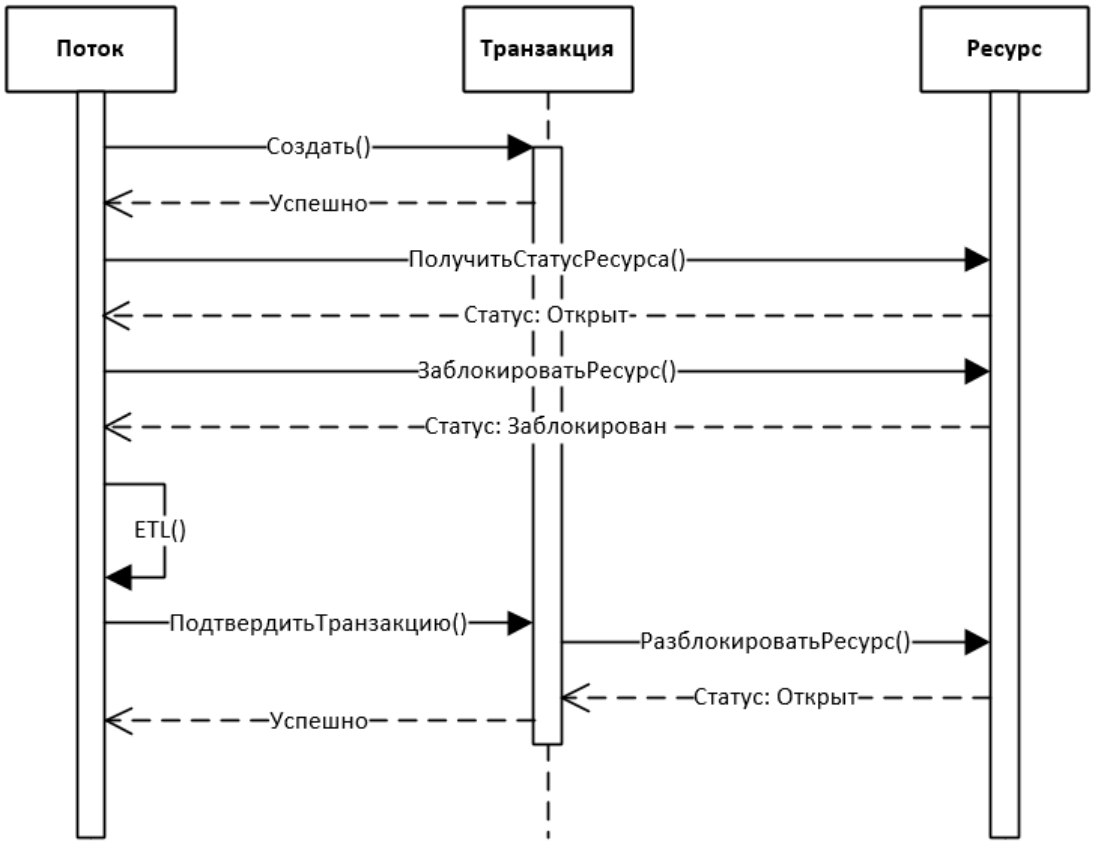

Рисунок 5 — Последовательность работ в рамках ETL-потоков при работе ресурсами

Менеджер ресурсов представляет собой дополнительный логический слой, занимающейся управлением работой ETL-процессов в Корпоративных Хранилищах Данных, который будет отвечать за взаимодействие потоков и ресурсов. Менеджер ресурсов будет занимать одно из ключевых мест среди текущих компонентов системы, поскольку именно он будет отвечать за правильное распределение доступов к источникам данных.

Концептуально-техническая реализация ресурсного менеджера

Как и было упомянуто ранее, ресурсный менеджер представляет из себя дополнительный слой управления ETL-процессами, на основе которого будет выстраиваться доступ к источникам данных на всех слоях хранилища данных. Это сделано с той целью, чтобы организовать корректное распределение доступов к источникам данных между потоками в правильной очередности их запуска. Для упрощения описания архитектуры предлагаемого решения далее представлена

схема коммуникации компонентов ресурсного менеджера и других программно-структурных единиц КХД (Рисунок 6).

Рисунок 6 — Интеграция менеджера ресурсов в инфраструктуру КХД

Компоненты системы управления транзакциями и менеджера ресурсов, которые отвечают за коммуникацию между потоками и друг другом, предлагается реализовать при помощи веб-сервиса с применением технологии REST, тем самым обеспечить целостность сервис-ориентированной архитектуры корпоративного хранилища данных.

При запуске потока создаётся транзакция и при помощи ресурсного менеджера блокирует перечень используемых в ней таблиц при помощи соответствующего состояния. В случае успешной отработки потока транзакция помечается для подтверждения (commit), состояние ресурсов обновляется, и они снова становятся доступными для работы. В том случае, когда поток увидит заблокированный для работы ресурс он будет дожидаться в очереди его разблокировки. Ресурсный менеджер запоминает потоки, которые обращаются к нему за доступом к ресурсу и организует из них очередь, передавая в качестве состояния ресурса идентификатор версии потока, которая сейчас использует данный источник данных. Как только идентификатор версии потока совпадёт с текущим состоянием ресурса, поток запускает ветку использования данного

ресурса. Таким образом, конкуренция потоков за доступ к данным будет устранена, поскольку будет напрямую зависеть от того, когда был поток запущен и в каком временном порядке произошло обращение к таблице. Новый порядок доступа к таблице показан на Рисунке 7.

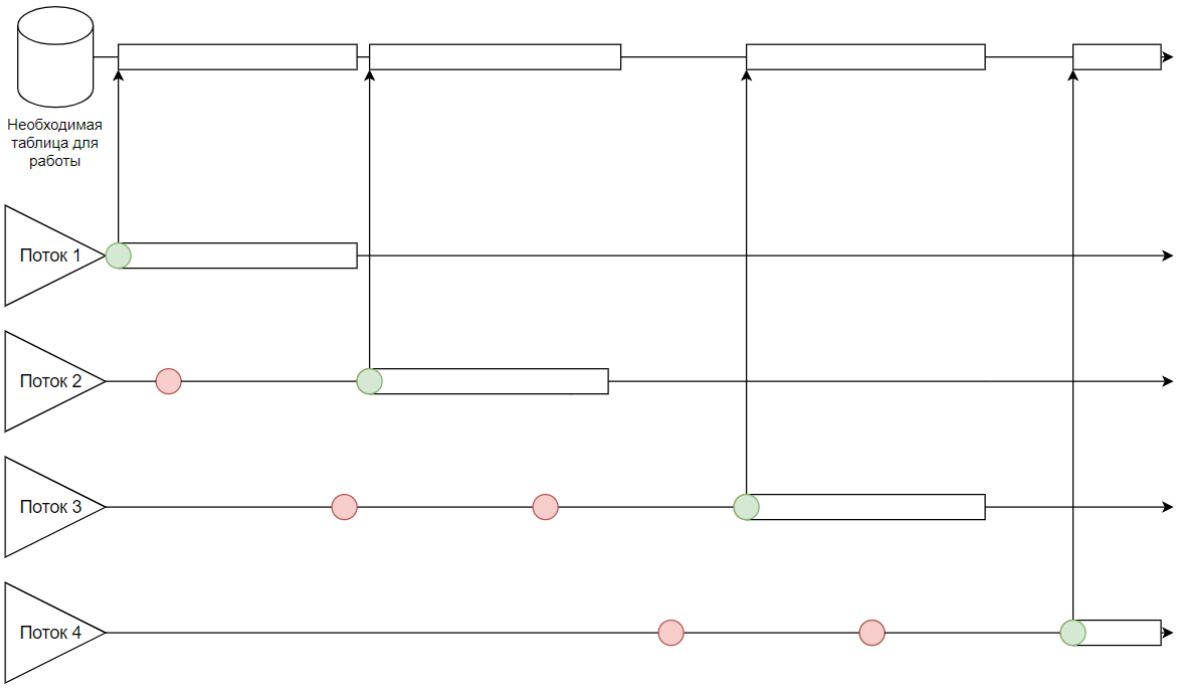

Рисунок 7 — Пример работы поток в соответствии с принципом работы менеджера ресурсов

Хоть в данном случае некоторым потокам приходится ждать другие, хотя в момент запроса доступа таблица может быть доступна, что увеличивает время ожидания запуска некоторых потоках, теперь не возникает такой ситуации, что какой-либо поток никак не может выполнить свою работу из-за того, что у него перехватывают доступ раньше, чем он сделал запрос.

Заключение

Рассмотренная в работе архитектура системы управления ресурсами корпоративным хранилищем данных позволит решить задачу управления конкурирующим доступом к объектам данных в результате контроля ресурсов посредством транзакций. Такой подход позволит передавать доступ к данным потокам оптимальным образом, в результате чего данные будут поступать в хранилище в нужном порядке, соблюдая условие консистентности данных.

Список литературы:

1. Linstedt, Dan. "Data Vault Series 2 – Data Vault Components". Data Vault Series. The Data Administration Newsletter. [Электронный ресурс] URL: https://tdan.com/data-vault-series-2-data-vault-components/5155 (Дата обращения: 11.11.2022)
2. Linstedt, Dan. "Data Vault Series 4 – Link Tables". Data Vault Series. The Data Administration Newsletter. [Электронный ресурс] URL: https://tdan.com/data-vault-series-4-link-tables/5172 (Дата обращения: 11.11.2022)
3. Linstedt, Dan. "Data Vault Series 5 – Loading Practices". Data Vault Series. The Data Administration Newsletter. [Электронный ресурс] URL: https://tdan.com/data-vault-series-5-loading-practices/5285 (Дата обращения: 11.11.2022)
4. Kent Graziano. Data Vault 2.0 Modeling Basics [Электронный ресурс] URL: https://www.vertabelo.com/blog/data-vault-series-data-vault-2-0-modeling-basics/ (Дата обращения: 11.11.2022)
5. Эрик Фримен, Элизабет Фримен. Паттерны проектирования = Head First Design Patterns. — СПб.: Питер, 2011. — 656 с.

References:

1. Linstedt, Dan. "Data Vault Series 2 – Data Vault Components". Data Vault Series. The Data Administration Newsletter. [Electronic resource] URL: https://tdan.com/data-vault-series-2-data-vault-components/5155 (Date of application: 11.11.2022)
2. Linstedt, Dan. "Data Vault Series 4 – Link Tables". Data Vault Series. The Data Administration Newsletter. [Electronic resource] URL: https://tdan.com/data-vault-series-4-link-tables/5172 (Date of application: 11.11.2022)
3. Linstedt, Dan. "Data Vault Series 5 – Loading Practices". Data Vault Series. The Data Administration Newsletter. [Electronic resource] URL: https://tdan.com/data-vault-series-5-loading-practices/5285 (Date of application: 11.11.2022)

4. Kent Graziano. Data Vault 2.0 Modeling Basics [Electronic resource] URL: https://www.vertabelo.com/blog/data-vault-series-data-vault-2-0-modeling-basics / (Date of request: 11.11.2022)
5. Eric Freeman, Elizabeth Freeman. Design Patterns = Head First Design Patterns. — St. Petersburg: Peter, 2011. — 656 p.

Для цитирования: Конаков П.О., Смоленцева Т.Е. РАЗРАБОТКА АРХИТЕКТУРЫ СИСТЕМЫ УПРАВЛЕНИЯ ИНФОРМАЦИОННЫМИ РЕСУРСАМИ В РАМКАХ МОДЕЛИ ХРАНИЛИЩА ДАННЫХ DATA VAULT 2.0// Научный сетевой журнал «Столыпинский вестник» №8/2022.