DOI 10.37882/2223–2966.2023.04–2.14

РАЗРАБОТКА МЕТОДОЛГИИ УНИФИКАЦИИ ЗАГРУЗКИ ДАННЫХ В ХРАНИЛИЩЕ ДАННЫХ МОДЕЛИ DATA VAULT

DEVELOPING A METHODOLOGY TO UNIFY DATA LOADING INTO THE DATA VAULT MODEL

P. Konakov

Summary. In the process of working with the corporate data warehouse there is a question of standardization and unification of data loading streams from the sources. A unified approach is necessary to keep the principle of actualization and historicity within the data objects loaded by different streams. This paper proposes to create an additional layer of abstraction for metadata describing the loading principle using ETL-processes. An ETL process is divided into two parts: the control flow, which is based on a pre-designed template with the worked out algorithm for collecting parameters to start the work flow that implements the logic of data loading from the source into the target table. Based on the described meta-information, the corresponding DWH component collects flows for the ETL flow orchestrator.

Keywords: big data, data processing, corporate data warehouse, ETL-process, standardization of development.

Конаков Павел Олегович
МИРЭА — Российский технологический университет
konakov.po@yandex.ru

Аннотация. В процессе работы предприятия с корпоративным хранилищем данных встаёт вопрос стандартизации и унификации потоков загрузки данных с источников. Единый подход необходим для того, чтобы в рамках загруженных объектов данных разными потоками сохранялся принцип актуализации и историчности. В рамках данной статьи предлагается создание дополнительного слоя абстракции для метаданных, описывающих принцип загрузки при помощи ETL-процессов. ETL-процесс подразделяется на две составные части: управляющий поток, создаваемый на основе заранее разработанного шаблона с проработанным алгоритмом сбора параметров запуска рабочего потока, который и реализует логику загрузки данных из источника в целевую таблицу. На основе описанной мета-информации соответствующий компонент КХД собирает потоки для оркестратора ETL-потоков.

Ключевые слова: большие данные, обработка информации, корпоративное хранилище данных, ETL-процесс, стандартизация разработки.

Введение

В процессе обработки больших объёмов данных предприятия всегда встаёт вопрос организации централизованного хранения, для чего зачастую используются такие модели корпоративных хранилищ данных, как Data Vault, якорное моделирование и т.д. Данные модели используют для отображения детальных изменений в данных, на основе которых можно формировать отчётность в процессе бизнес-аналитики деятельности организации.

Такие модели зачастую помимо конкретизированного и статичного набора возможных сущностей могут иметь строгие требования к процессу загрузки в них данных. Наличие данных ограничений и требований накладывают на разработчика потоков загрузки данных дополнительную нагрузку и ответственность за соблюдение этих правил и требований в своей разработке. Для конкретизации возможных проблем, которые могут появиться в процессе разработки, стоит рассмотреть конкретную модель корпоративного хранилища данных, например, Data Vault.

Концепция модели ХД Data Vault

Модель хранилища данных Data Vault является одной из наиболее гибких моделей для разработки, позволяя выполнять её расширение при возникновении новых требований от пользователей системы. Помимо этого, данная модель предполагает, что каждая новая сущность, загружаемая с источника данных, будет грузиться в свой собственный, отдельный объект, не затрагивая изменения в других объектах. Это позволяет вести параллельную разработку сущностей среди нескольких команд, не пересекаясь друг с другом. Data Vault имеет собственную терминологию объектов, используемых для моделирования сущностей.

Хаб (Hub) — сущность, описывающая ключевые информационные объекты организации, позволяющая объединять схожие сущности по смыслу. К Хабам относят те сущности, которые являются ключевыми объектам бизнес-процессов рассматриваемого предприятия, при этом набор сущностей данного типа зачастую бывает очень схож между разными моделями организаций, поскольку описывают базовое взаимодействие:

«Клиент», «Сотрудник», «Договор», «Заявка» и др. Функциональная значимость Хаба — хранение соответствий бизнес-ключа и сгенерированного суррогатного ключа. Это делается с целью обобщения данных из разных источников по одной сущности, при этом сам Хаб не хранит в себе никакой дополнительной информации по данной сущности [1] .

Связь (Link) — сущность, связывающая несколько Хабов между собой. Для описания взаимосвязи между ключевыми сущностями, например, «Клиент-Договор», «Заявка-Сотрудник», «Заявка-Клиент», «Сотрудник-Подразделение» и других используется данный вид сущности, обеспечивая описания отношения «много-ко-многим» между ключевыми объектами хранилища. Связь, как и Хаб, не является сущностью, хранящей бизнес-данные хранилища, а служит для хранения суррогатных ключей, генерируемых на основе уникальной связи между суррогатными ключами из Хабов, которые необходимо данной сущностью связать [1] .

Комета (Satellite) — бизнес-сущность, формируемая на основе данных, загружаемых с какого-либо источника данных, тем самым описывает свойства либо одной из ключевых сущностей (Хаба), либо связи между этими сущностями. В качестве первичного ключа выступает сгенерированный в Хабе или Связи суррогатный ключ на основе ключа, приходящего с источника данных, тем самым Комета является одним из источников генерации новых суррогатных ключей хранилища. Поскольку разные источники данных могут по-разному описывать ключевые сущности, то принято, чтобы для каждого источника данных была создана отдельная Комета, куда и будут грузиться данные при помощи соответствующих ETL-процессов [2] .

Как было отмечено ранее, данная модель хранилища данных является адаптивной под постоянно изменяемые требования пользователей, поэтому имеет возможность расширения состава моделей собственными. Единственное условие, которое должно выполняться при модификации модели — новые сущности должны составлять единое целое с хранилищем данных, должны обладать связями, которые необходимы для их структуризации и обобщения с уже имеющейся моделью данных.

В рамках модели Data Vault историчность данных соответствует формату SCD2, отслеживание изменений объектов производится на основе соотнесения вычисляемой хэш-суммы, сформированной из конкатенированных неключевых атрибутов сущности. Такой подход менее гибкий, чем, например, в якорной модели, однако позволяет уменьшить количество соединений при формировании отчёта, поскольку в ней каждый атрибут имеет свой формат историчности и условия соединения таблиц, тем самым уменьшая производительность выполнения запросов к данным).

Одним из ключевых недостатков модели можно выделить избыточность сущностей при разбиении источника, в результате чего модель достаточно сильно усложняется, в том числе и усложняется загрузка данных, поскольку необходимо загрузить данные из одной таблицы-источника в несколько таблиц модели хранилища данных.

Значимость унификации процесса загрузки данных

Data Vault имеет простой набор создаваемых сущностей для хранения информации, однако при этом накладывает на разработчика ETL-потоков дополнительные требования по загрузке данных в таблицы. Большинство требований необходимо соблюдать для консистентности данных хранилища.

Единый подход для формирования и поиска суррогатного ключа. Хаб является таблицей с конкретным набором атрибутов и конкретным типом данных для хранимых полей. Таким образом, для загружаемых данных необходимо обеспечить условие того, что суррогатный ключ должен будет формироваться на одном и том же наборе атрибутов и типов данных. То есть в рамках одного Хаба не может в одном случае генерировать суррогатный ключ на основе строчного значения, а в другом случае на основе числового идентификатора, хранимого в данных на источнике. Или же не может быть такой ситуации, когда в Хабе в одной ситуации генерирует суррогатный ключ на основе одного поля, а в другой — на основе нескольких полей одновременно. Тем самым на разработчика возлагается обязанность соблюдения стандарта генерации суррогатного ключа.

Консистентная загрузка данных в Хабы, Связи и Кометы. При загрузке данных в хранилище модели Data Vault разработчик всегда должен будет грузить данные больше, чем в одну сущность: в один Хаб и одну Комету, в два Хаба, Связь и Комету и т.д. Таким образом на разработчика накладывается дополнительная ответственность по консистентной загрузке данных во все сущности, которые ему необходимы для связи данных со всем остальным хранилищем. То есть в рамках загрузки будет ошибкой, когда данные в Комету были прогружены по суррогатным ключам, которые не были прогружены в соответствующий Хаб. Также ошибкой будет, если в Связи будет прогружена и сформирована связь по суррогатным ключам, которые не были прогружены в соответствующие Хабы. Помимо этого, под-

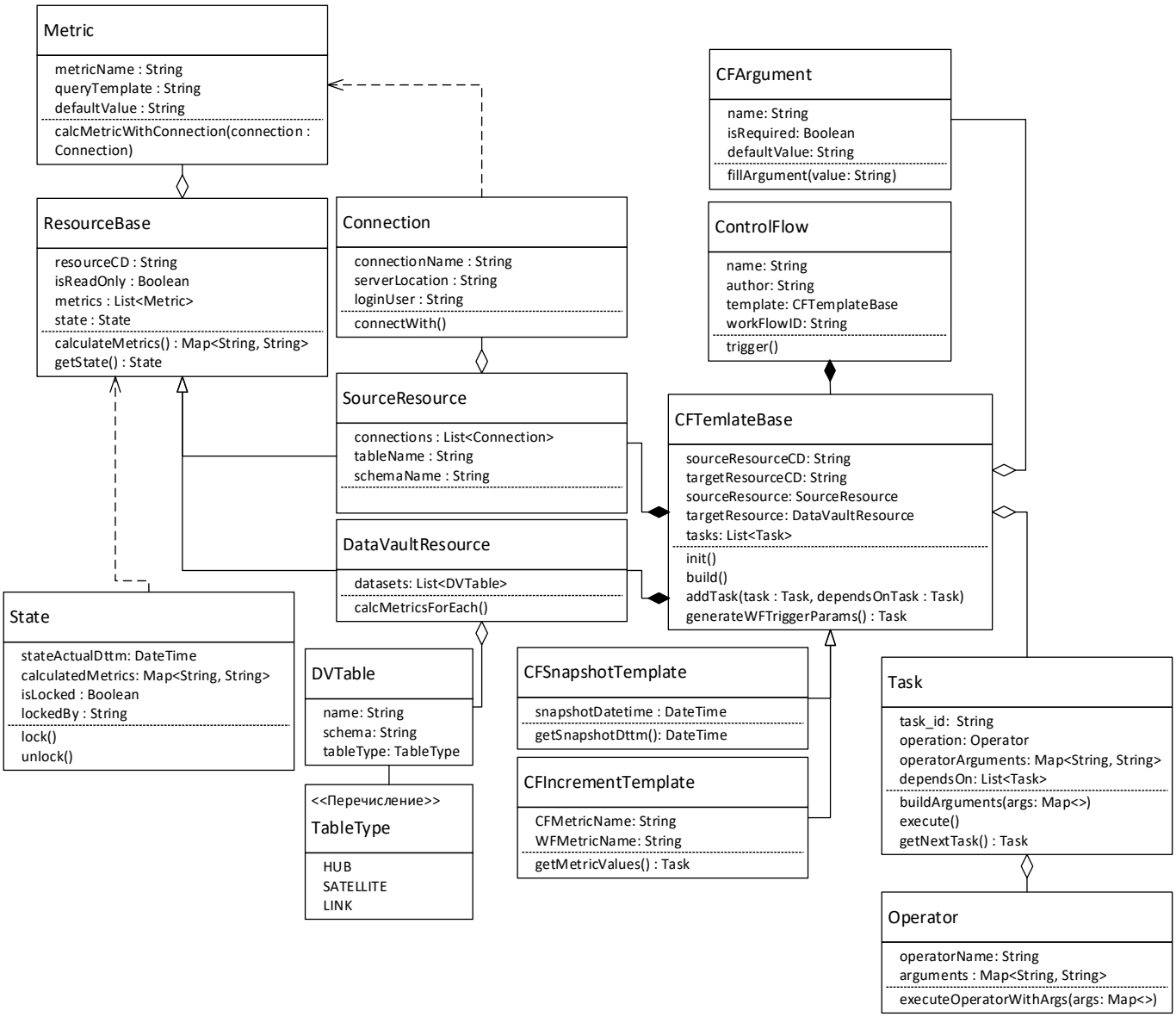

Рис. 1. Диаграмма классов компонентов управляющего потока

ход и порядок загрузки данных должен происходить в соответствии с логикой формирования данных, то есть сначала загружаются генерируемые ключи, а только потом данные, которые используют эти ключи.

Стандартизированный подход к вычислению дельты. Необходимо выдерживать стандартизированный в рамках всего хранилища подход к вычислению дельты загрузки данных. В том случае, когда подходы к формированию дельты загрузки различаются, нельзя гарантировать консистетность данных между разными таблицами, которые используются для дальнейшей работы с инкрементным формированием витрин и отчётов [3]. Тем самым, все разработчики, участвующие в процессе интеграции потоков загрузки данных в хранилище, должны договориться о едином подходе по вычислению дельты загрузки и строго соблюдать его.

В рамках данного вопроса можно выделить ещё множество аспектов, которые необходимо стандартизировать и поддерживать со стороны разработки ETL-потоков. Поскольку в процессе разработки присутствует человеческий фактор, многие требования к ETL-потокам разработчиками могут интерпретироваться по-разному или вовсе упущены. В результате таких упущений могут возникнуть инциденты при работе с данными в хранилище.

По данной причине предлагается унифицировать процесс загрузки данных в хранилище данных Data

Vault посредством формирования дополнительного слоя абстракции, через который разработчики будут задавать параметры для загрузки данных через ETL-потоки. Таким образом, разработчики могут управлять только информацией о таблице-источнике, целевых таблицах и принципах заполнения полей (маппинг), а логика работы потока будет формироваться системой самостоятельно на основе представленной информации о необходимых объектах.

Описание структурной модели реализации построения ETL-потоков

Как было сказано ранее, для унификации загрузки данных в хранилище модели Data Vault предлагается сформировать дополнительный слой абстракции метаданных, на основании которого система уже будет выстраивать ETL-потоки. Система сборки ETL-потоков, опираясь на информацию о таблице источника, целевых таблицах и маппинге, будет выстраивать последовательность действий при загрузке в соответствии обговоренным стандартом. Таким образом потоки будут шаблонизированы, структура потоков будет генерироваться в соответствии с тем, какой объект для загрузки был указан в метаданных потока.

Для загрузки данных необходимо два потока: управляющий поток (Control Flow) и рабочий поток (Work Flow). Управляющий поток служит для предварительной проверки информации о загрузке и подготовки параметров для дальнейшей работы ETL-потока. После проверки необходимой информации управляющий поток передаёт параметры запуска в рабочий поток, который реализует ETL-процесс по загрузке данных из источника в целевую таблицу [4] .

Управляющий поток предлагается реализовать с использованием ресурсной модели. Control Flow во время своей работы будет обращаться к менеджеру ресурсов хранилища с целью проверки доступности данных таблиц для захвата и загрузки данных. В том случае, когда ресурсы доступны для работы, их внутренние состояния с рассчитанными метриками передаются в качестве входных параметров рабочего потока. Поскольку в зависимости от типа загрузки данных (снимок или инкремент) требуется разная информация, то и управляющий поток будет шаблонизирован и формировать нужный набор параметров запуска Work Flow [4]. Структурная модель управляющего потока в виде диаграммы классов UML представлена на Рисунке 1.

Управляющий поток генерируется при помощи базового шаблона, на основе которого команда разработки хранилища данных может разработать реализации для сборки конкретных параметров для управляющего потока. Для шаблонов управляющего потока также можно задать набор параметров, которые разработчик ETL-потока должен указать в качестве входного набора данных, на основе которого будет генерироваться управляющий поток по шаблону. Поток состоит из задач, которые образуют собой последовательность взаимосвязанных действий, некоторые из которых могут выполняться параллельно. Задачи являются конкретной реализацией выполнения какого-либо действия, описываемого при помощи оператора. Оператором может быть запрос в базу данных, запрос к веб-сервису, выполнение команды в Unix-терминале и т.д.

Управляющий поток для сбора параметров запуска рабочего потока должен использовать информацию о таблице источника и таблице приёмника, куда будет загружаться данные при помощи разрабатываемого ETL-процесса. В качестве модели для представления данной информации будет использоваться ресурсная модель, где структурной единицей ресурса является таблица, участвующая в потоке либо в качестве источника, либо в качестве приёмника [5]. В рамках ресурса рассчитываются конкретные метрические показатели (метрики), отражающие текущее состояние ресурса на момент времени запроса к данному ресурсу.

Метрики необходимы для того, чтобы сопоставить показатели актуальности данных между последней загрузкой данных и данными, которые на данный момент находятся на источнике данных. Если данные показатели отличаются, то можно считать, что на источнике появились новые данные, которые стоит прогрузить при помощи запуска рабочего потока, в ином случае запуск потока смысла не имеет [6] .

Метрика рассчитывается на основе запроса, который выполняется при помощи параметров подключения к конкретному объекту расчёта. Это может быть как само хранилище данных, в которое производится загрузка данных, так и сервер с метаданными, куда предварительно записывается информация о загружаемых данных. На основе рассчитываемых метрических показателей ресурса формируется его состояние, по которому можно отслеживать доступность ресурса. Также при помощи состояния конкретного ресурса есть возможность блокировки его использования с целью построения грамотного управления доступом к ресурсам при асинхронной работе ETL-потоков, использующих одни и те же объекты для загрузки. Описанного набора метаданных достаточно, чтобы на основе него сформировать полноценный управляющий поток, выполняющий функцию формирования параметров запуска рабочего потока.

Рабочий поток ETL-процесса служит реализацией загрузки данных из таблицы-источника в целевые объ-

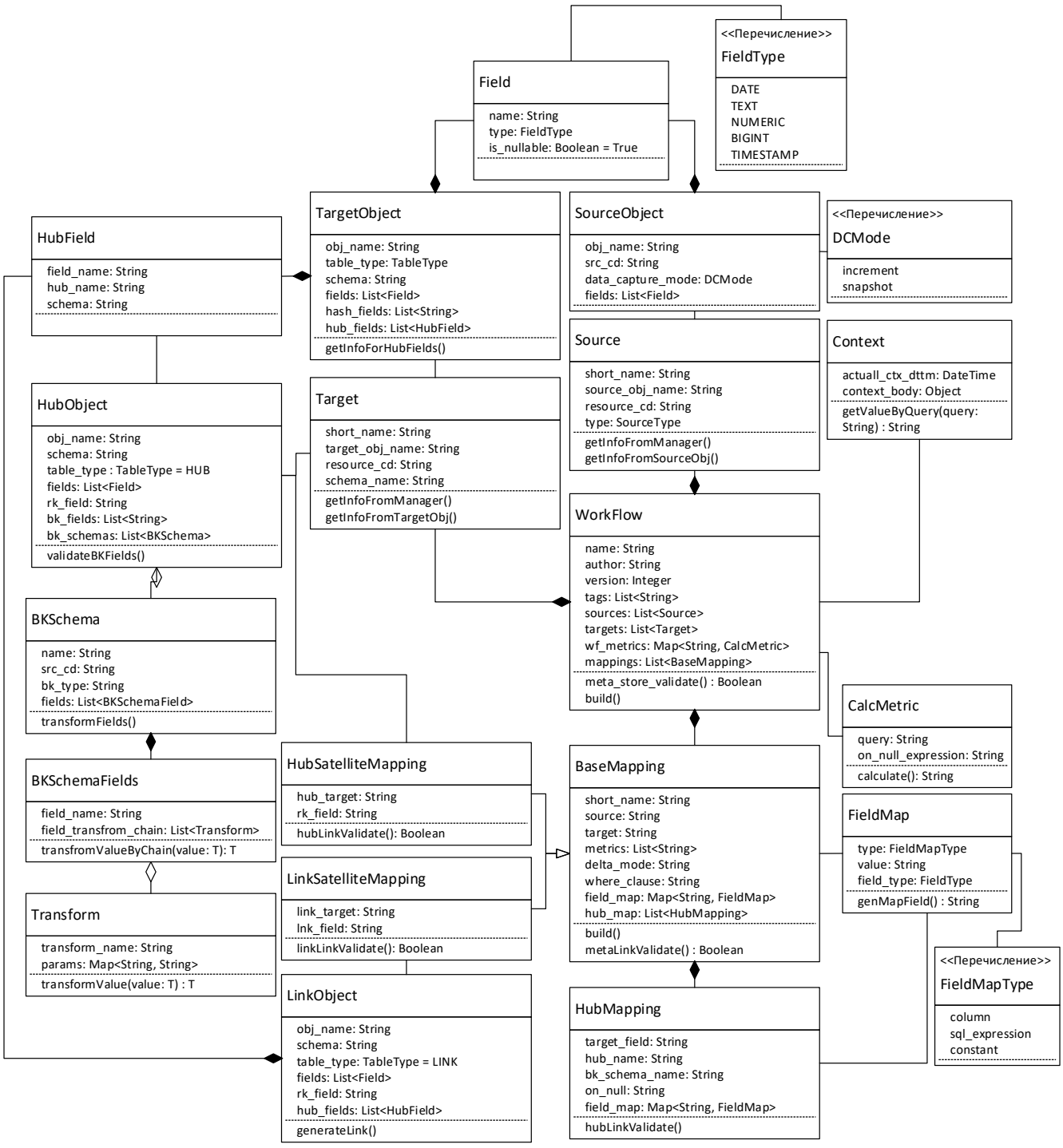

Рис. 2. Диаграмма классов компонентов рабочего потока

екты с сопутствующими шагами по генерации суррогатных ключей в связующие объекты хранилища данных. Как отмечалось ранее, шаблонизация рабочего потока необходима для формирования стандартизированного подходапо загрузке данных в хранилище. На основе описания абстрактных моделей потока, включающих в себя источник, целевой объект и объект маппинга, система может формировать поток с корректной последовательностью действий по загрузке данных и генерации суррогатных ключей в соответствующие таблицы хранилища данных. По сути своей рабочий поток также представлен в виде направленной последовательности задач, выполняющих конкретные действия по реализации ETL-процесса с дополнительной интеграцией с дру-

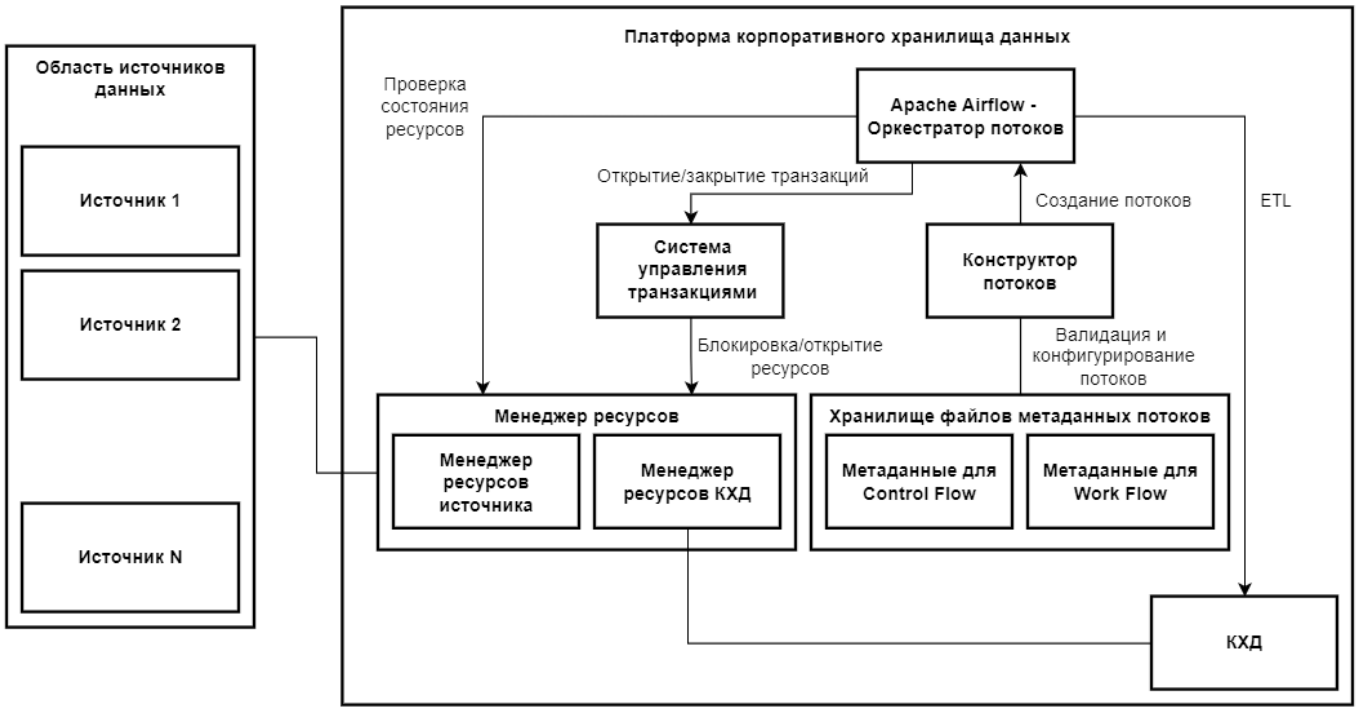

Рис. 3. Диаграмма компонентов платформы корпоративного хранилища данных

гими компонентами системы, управляющих состояниями ресурсов хранилища данных. Структурная модель рабочего потока в виде диаграммы классов представлена на рисунке 2.

Стоит отметить, что в данной модели отсутствуют явно определение задач и выполняемых операций потоком, поскольку вся логика работы будет основываться на описанных разработчиком ETL-потока метаданных: описание источника данных, описания целевой таблицы данных (а также сопутствующих для загрузки таблиц Связей и Хабов) и описания основных параметров рабочего потока.

Work Flow объединяет в себе все метаданные потока загрузки данных, проверяет ссылочную целостность между используемыми в данном потоке объектами. Ссылочная целостность и взаимосвязь между объектами достигается при помощи коротких названий объектов и их использовании при описании составных частей потока. Таким образом, в объекте маппинга рабочего потока указываются источник и целевой объект через их короткое название из sources и targets. Также короткие наименования используются для самих потоков загрузки конкретных объектов, чтобы выделять задачи по загрузке для объекта в отдельную смысловую часть.

Наличие конкретных объектов метаданных в потоке отвечает за соответствующую часть работы ETL-процесса: файл метаданных таблицы источника отвечает за процесс захвата данных с источника, метаданные маппинга отвечают за трансформацию данных и генерацию необходимых для целостности данных суррогатных ключей, метаданные целевых таблиц, соответственно, отвечают за процесс загрузки данных в соответствующие объекты хранилища. Стоит отметить, что некоторые атрибуты объектов метаданных являются опциональными, то есть их наличие или отсутствие в объекте напрямую влияет на набор выполняемых задач в рамках соответствующего раздела потока.

Структуру метаданных можно выделить в отдельных компонент системы генерации ETL-потоков, к которому система будет обращаться для формирования новых потоков и поддержания работы старых в случае изменения внутренних принципов работы ETL-процесса. «Конструктор» потоков будет предварительно собирать всю метаинформацию о потоке, производить проверку ссылочной и структурной целостности объектов. В том случае, когда проверка была выполнена без ошибок, выполняется сборка потока в соответствии с содержанием метаданных. Как было упомянуто ранее, поток может взаимодействовать с внешними сервисами по управлению информационными ресурсами хранилища данных [5]. Диаграмма компонентов, описывающая интегрирование описанной методологии унификации разработки потоков, представлена на Рисунке 3.

В платформе корпоративного хранилища данных реализуется инструмент централизованного построения ETL-потоков, соответствующих стандартам их разработки. От прикладных пользователей будет требоваться только описание объектов источника, целевых объектов и объекта настройки потоков (управляющего и рабочего). Построение потоков на основе данной информации ложится на соответствующий компонент-конструктор, реализуемый разработчиками платформы.

В качестве средства описания метаданных потоков можно применять как графические, так и файловые средства описания. Для описания объектов подходят данные в формате JSON (Java Sctipt Object Notation) и YAML (Yet Another Markup Language), однако второй вариант можно рассматривать в большем приоритете, поскольку имеет ряд внутренних ограничений по описанию (например, нельзя описывать несколько атрибутов с одним названием в рамках одного узла). Таким образом, можно добиться более строгого подхода к описанию объектов метаданных.

Заключение

Как упоминалось ранее, при разработке корпоративных хранилищ данных часто встаёт вопрос стандартизации подхода к загрузке данных. Это важная составляющая при организации процесса загрузки данных в хранилище, особенно для модели Data Vault, поскольку при работе большой команды может возникнуть фактор рассогласования подхода к загрузке данных (последовательности выполнения шагов, принципы генерации суррогатных ключей, подход к вычислению дельты изменений и др.). В качестве реализации предлагается спроектировать дополнительный слой абстракции метаданных, на основе будут конструироваться ETL-потоки.

В данном решении ETL-процесс подразделяется на два потока: управляющий поток (Control Flow), формирующий параметры запуска рабочего потока (Work Flow), который и реализует захват, обработку и загрузку данных. Для управляющего потока концептуально предлагается реализовать подход к описанию сущностей через параметры заранее разработанного шаблона, который будет собирать параметры запуска рабочего в соответствии с предполагаемым принципом загрузки (инкрементальная загрузка или загрузка снимка данных). Шаблоны могут расширяться и варьироваться в зависимости от необходимостей пользователей.

Описание работы рабочего потока происходит посредством описания соответствующих объектов источника, целевой таблицы и алгоритма загрузки данных из источника в целевой объект. Принцип загрузки и структура формируемого потока напрямую зависит от описываемой структуры объектов. Так, потоки, загружающие данные в объекты одного типа, будут иметь общий подход к загрузке.

Такое решение не только позволит формализовать и стандартизировать загрузку данных из разных источников в хранилище модели Data Vault, но и значительно сократить затраты на разработку ETL-потоков, поскольку разработчики смогут сконцентрировать своё внимание на детальном описании сущностей, которые участвуют в данном процессе, а не описывать повторяющиеся действия между потоками.

ЛИТЕРАТУРА

1. Linstedt, Dan. “Data Vault Series 2 — Data Vault Components”. Data Vault Series. The Data Administration Newsletter. [Электронный ресурс] URL: https://tdan.com/datavault-series-2-data-vault-components/5155 (Дата обращения: 10.01.2022)
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: 10.01.2022)
3. Tim Mitchell. The What, Why, When, and How of Incremental Loads. Tim Mitchell — Data Solution Architect [Электронный ресурс] URL: https://www.timmitchell.net/post/2020/07/23/ incremental-loads/ (Дата обращения: 20.11.2022)
4. 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: 10.01.2022)
5. Конаков П.О., Смоленцева Т.Е. Разработка архитектуры системы управления ресурсами в рамках модели хранилища данных DATA VAULT 2.0 // Сетевой журнал «Столыпинский вестник», 2022, № 9, URL: https://stolypin-vestnik.ru/stolypinskij-vestnik-9–2022/
6. Конаков П.О., Смоленцева Т.Е. Разработка методологии применения метрик алгоритмов для процессов актуализации данных // Научный журнал «НАУЧНЫЙ АСПЕКТ», 2022, № 6, URL: https://na-journal.ru/6–2022-informacionnye-tekhnologii/

Журнал «Современная наука: актуальные проблемы теории и практики»