УДК 004.622

ПОКАЗАТЕЛИ КАЧЕСТВА ЛИНИИ ФОРМИРОВАНИЯ ДАННЫХ НА ОСНОВЕ СТАТИЧЕСКОГО АНАЛИЗА SQL-ЗАПРОСОВ

Конаков П.О.

МИРЭА – Российский технологический университет, 119454, Россия, г. Москва, проспект Вернадского, 78. email: konakovp01@gmail.com

При работе с большими данными важно отслеживать то, откуда данные поступают в систему. Эти знания помогают как в процессах модификации и расширения функциональности отчётности, так и способствуют анализу вычислений с точки зрения эффективности использования вычислительных ресурсов. Для анализа происхождения данных в отчётности применяют линию формирования данных (Data Lineage). Производить формирование линии данных можно при помощи множества средств, одно из наиболее известных средств предполагает использование статического анализа SQL-запросов при помощи модели абстрактного синтаксического дерева (Abstract Syntax Tree, AST). Поскольку линия формирования данных представляет собой ориентированный ациклический граф, её можно анализировать и использовать для расчёта различных показателей. В данной работе предлагается перечень показателей, позволяющих оценить качество формируемой линии данных. Эти показатели позволяют оценить качество как всей линии данных, так и отдельных трансформаций внутри неё. Предлагаемые показатели позволяют проводить процесс оптимизации расчётов более эффективно и наглядно.

Ключевые слова: большие данные, преобразования над большими данными, статический анализ, абстрактное синтаксическое дерево, линия данных, оптимизация вычислительных процессов

QUALITY INDICATORS OF THE DATA LINEAGE BASED ON STATIC ANALYSIS OF SQL QUERIES

Konakov P.O.

MIREA – Russian Technological University, 119454, Moscow, 78 Vernadskogo Avenue, Russia, email: konakovp01@gmail.com

When working with big data, it is important to keep track of where the data is coming from in the system. This knowledge helps both in the processes of modifying and expanding the reporting functionality, and contributes to the analysis of calculations in terms of the efficiency of using computing resources. To analyze the origin of data in reporting, a Data Lineage is used. You can generate a data line using a variety of tools, one of the most well-known tools involves the use of static analysis of SQL queries using the Abstract Syntax Tree (AST) model. Since the data generation line is an oriented acyclic graph, it can be analyzed and used to calculate various indicators. In this paper, we propose a list of indicators that allow us to evaluate the effectiveness of the generated data line. These indicators allow us to evaluate the effectiveness of both the entire data line and individual transformations within it. The proposed indicators make it possible to carry out the calculation optimization process more efficiently and visually.

Keywords: Big Data, transformations over big data, static analysis, abstract syntax tree, data lineage, optimization of computing processes

Введение

На текущем этапе развития информационных технологий и их применения в бизнес-процессах организации формируется большой объём данных, который может быть использован для аналитики и оптимизации внутренних процессов предприятиях [1]. Для формирования корпоративной отчётности на основе больших данных необходимо выстроить процесс аккумулирования и обогащения данных информацией из нескольких систем-источников [1,2]. С целью понимания процессов обогащения данных для отчётов организации используют линию формирования данных (Data Lineage) [3].

Выполнение расчётов с использованием больших объёмов данных требует отслеживания используемых вычислительных ресурсов. Неоптимальное использование ресурсов при параллельном выполнении вычислений

могут привести к снижению производительности всей системы или вовсе привести к неконтролируемым ошибкам и остановкам работы системы. Линия формирования данных в данном случае может использоваться как наглядная модель для анализа и поиска потенциальных мест для последующей оптимизации. Помимо этого, Data Lineage может быть использована как модель, которую можно применить для анализа структуры и оценки эффективности, проводимых в ней расчётов. Показатели качества линии данных позволят провести сравнительный анализ на этапе оптимизации проводимых вычислений и дать направление для последующих шагов улучшения производительности трансформаций над данными. Для построения модели Data Lineage с последующим расчётом показателей будет использоваться статический анализ кода SQL-запросов, который позволит на нужном уровне детализации проводить исследование описанных трансформаций над данными.

Структура модели линии формирования данных

В классическом представлении линия данных — это ациклический направленный граф (Directed Acyclic Graph, DAG), где начальные узлы — это таблицы-источники, а конечный узел — целевая таблица-отчёт, которая формируется в результате множества трансформаций данных [4,6].

Промежуточные узлы графа определяют трансформации, такие как соединение таблиц (JOIN), их объединение (UNION), фильтрация (WHERE) и агрегация (GROUP BY). Каждая из трансформаций создаёт отдельный узел в графе. Связи между узлами определяют взаимосвязанные трансформации, которые можно описать как «формируется при помощи» и «используется для». То есть узел, из которого выходит стрелка, используется для выполнения трансформации, в которую стрелка входит. Если одна и та же таблица используется несколько раз в трансформации, создаётся отдельная стрелка для каждого использования. На рисунке 1 приведен пример линии данных.

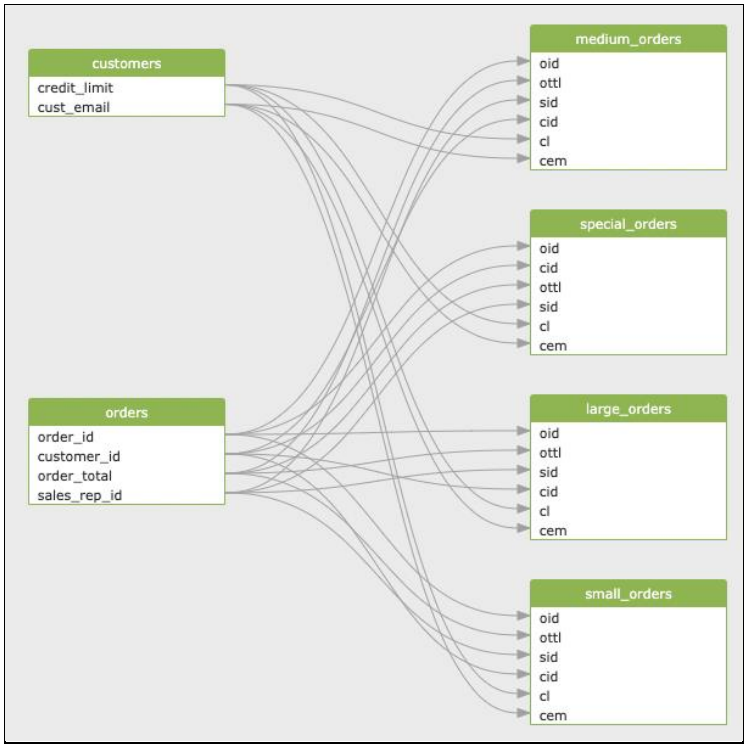

Рисунок 1 – Пример линии данных [5]

Обычно, когда несколько узлов используются для трансформации, говорят о соединении таблиц или их объединении. Для большей детализации линия данных может описывать не только переходные трансформации, но и используемые поля и вычисляемые значения.

Линия формирования данных на основе статического анализа SQL-запросов

Статический анализ кода позволяет преобразовать исходный код программы в промежуточное абстрактное представление, которое можно использовать как для последующего преобразования в произвольный формат выходных данных, так и для дополнительного анализа кода, например, для проверки соответствия кода правилам наименования объектов (naming convention), анализа потенциальных уязвимостей [7,8], анализа

производительности и т.д. Одним из наиболее используемых инструментов статического анализа кода является анализ абстрактного синтаксического дерева (Abstract Syntax Tree, AST) [9]. Абстрактное синтаксическое дерево представляет собой древовидную структуру, где каждый узел представляет синтаксический элемент программного кода, а предки данного узла представляют вложенные в него элементы. За счёт такой реализации AST позволяет проводить статический анализ исходного кода на требуемом уровне детализации и вложенности элементов кода [10, 11].

Линия формирования данных представляет собой направленный ациклический граф, в котором узлы представлены в виде трансформаций над данными, крайние узлы являются источниками данных, а конечная точка определяет целевой объект формирования данных. Каждый узел представляет собой логически атомарную трансформацию над данными. Трансформация может включать в себя объединение (UNION), соединение (JOIN) таблиц, а также агрегацию (GROUP BY) и фильтрацию (WHERE) над данными. Для проведения детальной аналитики описанных типов трансформаций над данными модель узла трансформации над данными должна включать следующие атрибуты:

- наименование узла трансформации;
- флаг материализации трансформации;
- флаг источника данных;
- перечень используемых трансформаций для текущего шага в порядке их упоминания;
- условия соединения объектов трансформации;
- перечень результирующих полей трансформации и используемые для их формирования поля из источников.

Перечень описанных атрибутов модели трансформации полностью выражается из соответствующего SELECT-запроса, при этом стоит учитывать не только данные из текущего запроса, но и из всех предшествующих, поскольку одним из источников данных для анализируемой трансформации может быть другая трансформация, которая была определена ранее в исходном коде хранимой процедуры. Пример преобразования запроса в модель линии формирования данных представлен на рисунке 2.

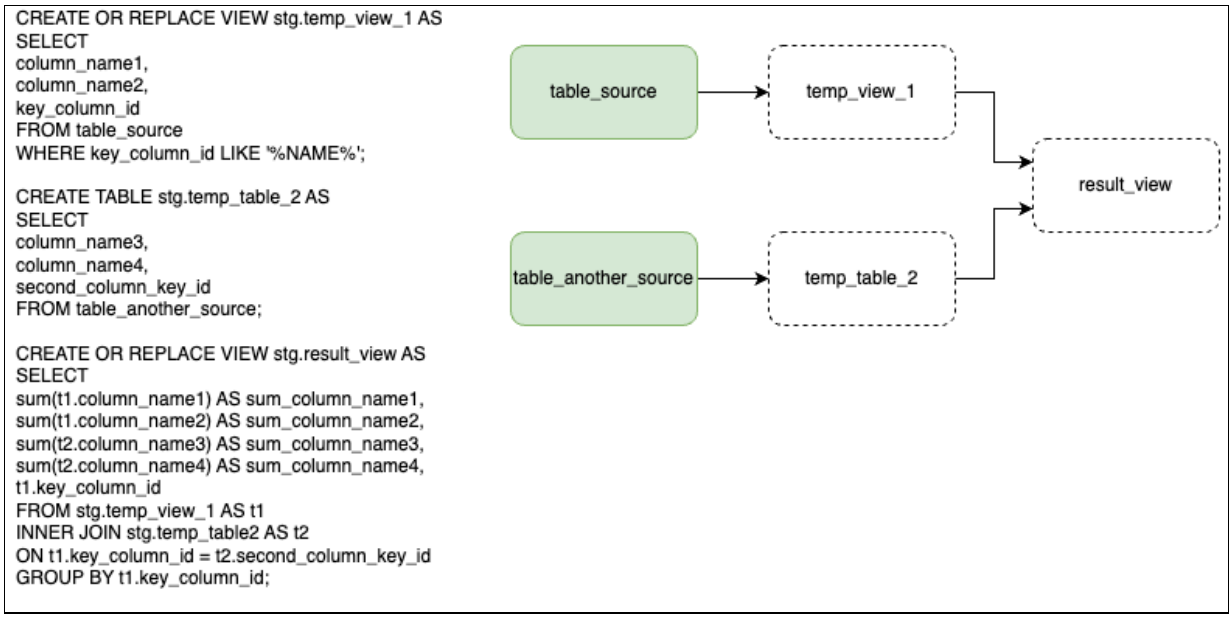

Рисунок 2 – Пример преобразования SQL-запроса в линию формирования данных

Описанная модель линии формирования данных полностью покрывает задачи по анализу её производительности, поскольку учитывает все детали выполняемых трансформаций, которые можно оценить при помощи статического анализа кода, то есть без непосредственного его выполнения.

Показатели качества линии формирования данных

Оценка качества линии формирования данных позволяет сравнивать результаты для всего графа трансформаций и отдельных узлов. Это помогает оптимизировать код и выявлять слабые места в расчёте отчёта. Оценка основана на анализе структуры трансформаций и использует алгоритмы обхода дерева в ширину и глубину для расчёта показателей [12]. Для расчёта показателей качества каждого узла трансформации

необходимо исследовать сам SELECT-запрос, который его формирует и его ближайшие трансформации, на основе которых он формируется. Эта информация позволит понять, насколько эффективно выполнится сам запрос, и какие параметры можно было бы улучшить, чтобы запрос выполнялся эффективнее.

В текущих системах управления данными процесс соединения нескольких таблиц предполагает последовательную подготовку данных под выбранное условие соединения, данные перераспределяются по сегментам для наиболее быстрого выполнения соединения данных. При неправильно сформированной последовательности соединений, когда условия соединения нескольких таблиц могут совпадать, подготовка данных к соединению может происходить большее количество раз, чем могло бы быть, если бы одинаковые условие находились друг за другом [13]. Показатель монотонности соединения $M(T)$ для трансформации T позволит оценить, насколько правильно выстроены условия соединения в таблице:

$$M(T) = \frac{C(T)}{J(T)}, \tag{1}$$

где $C(T)$ – число смены условий соединения для трансформации T, $J(T) = |T.joins|$ – общее число соединений в таблице.

Под сменой условия соединения подразумевается, что в текущем условии соединения данных не используются поля, которые были в предыдущем условии (рисунок 3).

```
function C(T)
    k ← 1
    j_p ← T.joins[0]
    for i ← 1, |T.joins| do          ▷ Обход всех условий соединения
        if j_p ∩ T.joins[i] = ∅ then
            k ← k + 1
        end if
        j_p ← T.joins[i]
    end for
    return k
end function
```

Рисунок 3 – Алгоритм вычисления числа смены условий соединения

При анализе качества трансформаций также важно учитывать используемые при этом объекты данных. Материализованные данные или данные из системы источника используют диск для хранения данных результата, когда представления будут использовать оперативную память системы для сохранения данных. Когда оперативной памяти уже не хватает для вычислений, запрос будет формировать spill-файлы для сохранения промежуточного результата на диск [14, 15]. Показатель материализации источников покажет отношение того, какую часть источников для трансформации занимают материализованные расчёты и таблицы-источники среди всех используемых для расчёта объектов данных:

$$R(T) = \frac{MS(T)}{S(T)}, \tag{2}$$

где $MS(T) = |\{t \mid t.materialized = 1 \wedge t \in T.sources\}|$ – число материализованных источников трансформации, $S(T) = |T.sources|$ – число всех источников трансформации над данными.

С точки зрения формирования запроса также важно, чтобы сформированный результат трансформации над данными был использован в дальнейших трансформациях, иначе в их чтении из таблиц или расчёте значения нет никакого смысла. Показатель полноты используемых расчётов предполагает оценку числа полей источников трансформации, используемых в текущей трансформации:

$$P(T) = \frac{F(T)}{F_S(T)}, \tag{3}$$

где $F(T) = |\{f \mid m \in T.fields \wedge f \in m.fields\}|$ – число уникальных использованных полей в трансформации, $F_S(T) = |\{f \mid s \in T.sources \wedge f \in s.fields\}|$ – число всех полей в используемых для расчёта трансформациях.

При оценке качества всей линии данных необходимо использовать информацию о всех выполняемых трансформациях, при этом для расчёта конкретного показателя могут использоваться только конкретные атрибуты каждой трансформации.

Глубина расчёта линии данных является одним из наглядных показателей, поскольку определяется как глубина соответствующего графа. Чем глубже дерево линии формирования данных, тем больше в графе производится промежуточных расчётов до конечной витрины, что влияет объем и время потребления вычислительных ресурсов. Принцип вычисления глубины расчёта линии данных представлен на рисунке 4.

```
function MAX_DEPTH(T)
    if T.sources = ∅ then          ▷ Обращение к атрибуту sources узла T
        return 0
    else
        U ← {MAX_DEPTH(x)|x ∈ T.sources}
        return max(U) + 1
    end if
end function
```

Рисунок 4 – Описание алгоритма глубины расчёта линии данных

Показатель материализации расчётов также связан с общим качеством сформированной линии данных. Данный показатель позволяет отслеживать общее состояние Data Lineage и потенциальные проблемы, связанные с слишком большими промежуточными расчётами, не имеющих никакую промежуточную материализацию. При выполнении таких расчётов для хранения будет использоваться оперативная память, однако если для расчёта её будет не хватать, СУБД будет формировать spill-файлы для частичного сохранения запроса на диск, что приведёт к неконтролируемому со стороны разработчиков потреблению ресурсов диска. Показатель материализации расчётов определяется следующим образом:

$$N(G) = 1 - \frac{max\_unmat\_depth(\text{G.head})}{max\_depth(\text{G.head})}, \quad (4)$$

где $max\_unmat\_depth(T)$ – глубина нематериализированного расчёта линии данных, $max\_depth(T)$ – глубина расчёта линии данных, G.head – указание на голову графа линии данных (целевого объекта расчёта).

Глубина нематериализованного расчёта линии данных определяется как максимальная глубина непрерывной цепочки трансформаций без материализации. Принцип определения глубины схож с показателем глубины расчёта линии данных, единственным отличием является то, что условие остановки дальнейшего прохода алгоритма является встреченный узел материализации (рисунок 5).

```
function MAX_UNMAT_DEPTH(T)
    if T.is_materialized = 1 ∧ |T.sources| = 0 then
        return 0
    else
        U ← {MAX_UNMAT_DEPTH(x)|x ∈ T.sources}
        return max(U) + 1
    end if
end function
```

Рисунок 5 – Описание алгоритма глубины нематериализованного расчёта линии данных

Показатель перерасчёта линии данных определяет то, как часто таблицы-источники используются повторно в запросах для формирования итогового отчёта.

Также этот показатель позволяет повысить ценность промежуточной материализации трансформаций над данными, которые могут быть использованы в линии данных несколько раз. Сначала необходимо ввести метод расчёта числа использований таблиц в линии данных.

Предполагается, что каждая нематериализованная трансформация учитывает в себе все использования трансформаций и таблиц-источников, использованных в её расчёте. Алгоритм расчёта представлен на рисунке 6.

function SOURCES_USAGE_COUNT(T)
  if $T.is\_materialized = 1$ **then**
    return 1
  else
    $V \leftarrow \{c | c \in T.sources\}$
    $S = \sum_{v \in V} \text{SOURCES\_USAGE\_COUNT}(v)$
    return S
  end if
end function

Рисунок 6 – Описание алгоритма расчёта числа использований таблиц для каждого узла

Таким образом, если посчитать число использований источников для вершины графа, то есть для конечного объекта линии формирования данных, полученное значение будет характеризовать общее число использований таблиц-источников. На основе данного значения можно определить показатель перерасчёта источников как отношение числа источников данных в анализируемой линии данных к числу использований таблиц-источников в анализируемом графе:

$$P(G) = \frac{O(G)}{sources\_usage\_count(\text{G.head})}, \quad (5)$$

где $O(G) = |\{t \mid t \in G \wedge t.sources = \emptyset\}|$ – число источников линии формирования данных, $sources\_usage\_count(T)$ – число использований таблиц для каждого узла.

В оптимальном случае линия данных должна использовать каждый из источников трансформации единожды, чтобы описанный показатель был равен единице. Стоит отметить, что помимо общей оценки всей линии данных данный показатель также можно использовать для оценки оптимальности отдельных узлов, если один из узлов использует слишком много нематериализованных трансформаций над данными.

Применение показателей к линии формирования данных

Для экспериментальной оценки писанных показателей качества предлагается рассмотреть конкретный пример линии формирования данных, для которого будут приведены результаты расчётов всех показателей. Показатели позволят провести анализ и определить, какие конкретные действия в трансформациях над данными с точки зрения SQL влияют на значения конкретных показателей качества, и рассмотреть действия, которые могут повлиять на улучшение показателей после проведения оптимизации написанного кода. Для анализа будет использоваться скрипт на языке SQL, представленный на рисунке 7, последовательность выполнения запросов определена слева направо.

```sql
create or replace view stg.stage_temp_st1 as
select
    col_name,
    sum(val1) AS sum_val1,
    sum(val2) AS sum_val2,
    sum(val3) AS sum_val3,
    sum(val4) AS sum_val4,
    sum(val5) AS sum_val5
from source_first
where col_name IN ('string1', 'string2', 'string3')
GROUP BY col_name;
```

```sql
create or replace view stg.stage_temp2 as
with stage_temp_2_1 as (
    select
        t1.col1,
        t1.col2,
        t1.col3,
        t1.col4,
        t1.col5,
        t1.col6,
        t1.col7,
        t1.col8
    from source_second AS t1
        inner join stg.stage_temp_st1 AS t2
            on t1.col1 = t2.col_name
),
stage_temp_2_2 as (
    select
        t1.col1,
        t1.col9,
        t1.col10,
        t1.col11,
    from source_third as t1
        inner join stg.stage_temp_st1 AS t2
            ON t1.col1 = t2.col_name
)
select
    t1.col1,
    t1.col2,
    t1.col3,
    t2.col9,
    t2.col10
from stage_temp_2_1 as t1
    inner join stage_temp_2_2 as t2
        on t1.col1 = t2.col1
```

```sql
create or replace view stg.final_step as
select
    t1.col1,
    t1.col2,
    t1.col3,
    t1.col9,
    t1.col10,
    t2.an_col1,
    t2.an_col2,
    t3.some_col1,
    t3.some_col2,
    t5.col_name_1
    t5.col_name_2
from stage_temp2 as t1
    inner join some_source_1 as t2
        on t1.col1 = t2.my_col
    inner join some_source_2 as t3
        on t1.col2 = t3.col_key
    inner join some_source_3 as t4
        on t1.col1 = t4.col
    inner join some_source_4 as t5
        t4.another_col = t5.key;
```

Рисунок 7 – Пример скрипта на языке SQL для анализа

В результате статического анализа SQL-запросов при помощи модели абстрактного синтаксического дерева и определённых правил формирования модели линии формирования данных был получен направленный граф, описывающий последовательность трансформаций над данными и взаимосвязи между ними (рисунок 8).

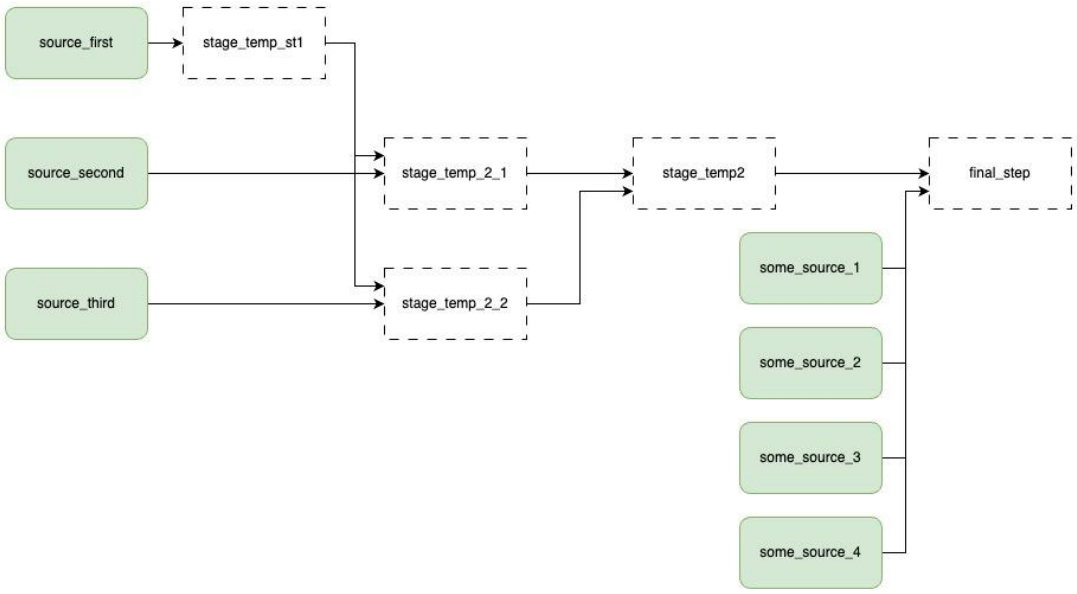

Рисунок 8 – Сформированная линия формирования данных

Необходимо уточнить, что зелёные фигуры являются источниками данных, которые уже есть в системе, остальные фигуры являются трансформациями над данными, при этом фигуры с пунктирными краями являются трансформациями без материализации и создаются как VIEW, а фигуры со сплошной линией – материализуются и создаются как таблицы. В таблицах 1 и 2 представлены расчёты общих показателей линии данных и показатели для каждой трансформации соответственно.

Таблица 1 – Рассчитанные показатели качества для анализируемой линии формирования данных

Показатель	**Значение**
Глубина расчёта линии данных	5
Глубина нематериализованного расчёта линии данных	4
Показатель материализации расчётов	0,2
Число использований таблиц-источников для конечного объекта линии данных	8
Число источников линии данных	7
Перерасчёт источников	0,875

Таблица 2 – Рассчитанные показатели качества для трансформаций

Показатель	**stage_temp_st1**	**stage_temp_2_1**	**stage_temp_2_2**	**stage_temp2**	**final_step**
Монотонность соединения таблиц	0	1	1	1	1
Материализация источников трансформации	1	0,5	0,5	0	0,8
Полнота расчётов	1	0,57	0,4	0,42	1
Число использований таблиц-источников	1	2	2	4	8

На основе представленных расчётов показателей качества линии данных можно сделать вывод о слабой

степени материализации отчётов, поскольку все шаги трансформаций не имеют материализацию, что также сказалось на перерасчёте источников. Взглянув на показатели для трансформаций, можно заметить, что для узла stage_temp2 отсутствует какая-либо материализация источников, так и самого шага, что привело к значительному росту числа использований таблиц-источников. Данную трансформацию стоит рассмотреть в первую очередь на предмет оптимизации.

Заключение

В рамках работы были определены основные шаги статического анализа SQL-запросов для построения линии формирования данных (Data Lineage) с целью дальнейшего анализа с точки зрения оптимизации. Анализ данной модели позволил определить перечень показателей, которые можно использовать для сравнительного анализа линии данных на этапах оптимизации рассматриваемой цепочки трансформаций над данными. Такие показатели могут быть полезными для аудита информационных систем, управления качеством данных, а также для оптимизации использования вычислительных ресурсов.

Данные показатели в перспективе могут использоваться для анализа на этапе автоматического тестирования трансформаций над данными и мониторинга «неоптимальных» запросов, написанных разработчиками. Далее, при формулировке набора подходов по оптимизации линии данных, эти показатели можно использовать для оценки качества инструментов автоматической оптимизации Data Lineage с использованием популяционных алгоритмов и алгоритмов машинного обучения. Развитие исследований в данном направлении позволит развивать процессы поставки данных, а не только поддерживать их стабильное состояние через оптимизацию потребления вычислительных ресурсов.

Список литературы

1. Пузикова С. И. Цифровые основы роста: цифровая трансформация и управление, основанное на данных // Использование Big Data в официальной статистике Using Big Data in official statistics. – 2022. – С. 260-263
2. Singh S. et al. (ed.). Data-driven decision making for long-term business success. – IGI Global, 2023.
3. Ikeda R., Widom J. Data lineage: A survey // Stanford University Publications. http://ilpubs. stanford. edu. – 2009. – Т. 8090. – №. 918. – С. 1.
4. What Is Data Lineage? // IBM. URL: https://www.ibm.com/topics/data-lineage#:~:text=Data%20lineage%20is%20the%20process,any%20ETL%20or%20ELT%20processes (Дата обращения: 14.02.2024)
5. SQLFlow: Visualise column impact and data lineage to track columns across transformations by analysing SQL query // SQLFlow. URL: https://sqlflow.gudusoft.com/#/ (Дата обращения: 19.02.2024)
6. Data lineage overview and concepts // Astronomer Documentation. URL: https://docs.astronomer.io/astro/data-lineage-concepts (Дата обращения: 19.02.2024)
7. Аветисян А., Белеванцев А., Бородин А., Несов В. Использование статического анализа для поиска уязвимостей и критических ошибок в исходном коде программ // Труды Института системного программирования РАН. – 2011. – Т. 21. – С. 23-38.
8. Al Azhar M. F., Harwahyu R. Detection of sql injection vulnerability in codeigniter framework using static analysis //Multitek Indonesia. – 2023. – Т. 17. – №. 1. – С. 69-78.
9. Sun W. et al. Abstract Syntax Tree for Programming Language Understanding and Representation: How Far Are We? // arXiv preprint arXiv:2312.00413. – 2023.
10. Горчаков А. В. Исследование подхода к преобразованию программ в векторные представления на основе цепи Маркова // Электронный научный журнал «ИТ-Стандарт». – 2023. – №2. – С. 40-50
11. Фомин И. С., Тагилов В. М. Статический анализ кода //LVI Всероссийская конференция по проблемам динамики, физики частиц, физики плазмы и оптоэлектроники. – 2020. – С. 222-225
12. Scheffler R. On the recognition of search trees generated by BFS and DFS //Theoretical Computer Science. – 2022. – Т. 936. – С. 116-128.
13. How to optimize SQL Queries with multiple joins in Oracle // process.st. URL: https://www.process.st/how-to/optimizing-sql-queries-with-multiple-joins-in-oracle/ (Дата обращения 10.03.2024)
14. Spill-файлы // Arenadata Docs, URL: https://docs.arenadata.io/ru/ADB/current/concept/data-model/spill-files.html (Дата обращения 18.03.2024)
15. Managing Spill Files Generated by Queries // VMware Docs. URL: https://docs.vmware.com/en/VMware-Greenplum/7/greenplum-database/admin_guide-query-topics-spill-files.html (Дата обращения: 18.03.2024)

References

1. Puzikova S. I. Digital fundamentals of growth: digital transformation and data-based management // Using Big Data in official statistics, – 2022. – C.260-263
2. Singh S. et al. (ed.). Data-driven decision making for long-term business success. – IGI Global, 2023.
3. Ikeda R., Widom J. Data lineage: A survey //Stanford University Publications. http://ilpubs. stanford. edu. – 2009. – Т. 8090. – №. 918. – С. 1.
4. What Is Data Lineage? // IBM. URL: https://www.ibm.com/topics/data-lineage#:~:text=Data%20lineage%20is%20the%20process,any%20ETL%20or%20ELT%20processes
5. SQLFlow: Visualise column impact and data lineage to track columns across transformations by analysing SQL query // SQLFlow. URL: https://sqlflow.gudusoft.com/#/
6. Data lineage overview and conepts // Astronomer Documentation. URL: https://docs.astronomer.io/astro/data-lineage-concepts
7. Avetisyan A., Belevantsev A., Borodin A., Nesov V. Using static analysis for finding security vulnerabilities and critical errors in source code // Proceedings of the Institute for System Programming of the RAS. – 2011. – Т. 21. – С. 23-38
8. Al Azhar M. F., Harwahyu R. Detection of sql injection vulnerability in codeigniter framework using static analysis //Multitek Indonesia. – 2023. – Т. 17. – №. 1. – С. 69-78
9. Sun W. et al. Abstract Syntax Tree for Programming Language Understanding and Representation: How Far Are We? //arXiv preprint arXiv:2312.00413. – 2023.
10. Gorchakov A. V. A Study of the Approach to Conversion of Program Texts into Vector Representations Based on Markov Chains // Electronic Scientific Journal “IT-Standart”. – 2023. – №2. – С. 40-50
11. Fomin I. S., Tagilov V. M. Static code analysis // LVI All-Russian Conference on Problems of Dynamics, Particle Physics, Plasma Physics and Optoelectronics. – 2020. – С.222-225
12. How to optimize SQL Queries with multiple joins in Oracle // process.st. URL: https://www.process.st/how-to/optimizing-sql-queries-with-multiple-joins-in-oracle/
13. Scheffler R. On the recognition of search trees generated by BFS and DFS //Theoretical Computer Science. – 2022. – Т. 936. – С. 116-128.
14. Spill-files // Arenadata Docs, URL: https://docs.arenadata.io/ru/ADB/current/concept/data-model/spill-files.html
15. Managing Spill Files Generated by Queries // VMware Docs. URL: https://docs.vmware.com/en/VMware-Greenplum/7/greenplum-database/admin_guide-query-topics-spill-files.html