Санкт-Петербургский государственный университет

Математическое обеспечение и администрирование информационных систем

Смирнов Александр Андреевич

Реализация алгоритма DFD поиска функциональных зависимостей

Отчет по учебной практике

Научный руководитель:
ассистент кафедры ИАС Чернышев Г. А.

Санкт-Петербург
2021

Оглавление

Введение

Функциональная зависимость — это бинарное отношение на множестве столбцов некоторой таблицы. Такие зависимости позволяют понять, есть ли некоторая связь (зависимость) между данными из двух наборов столбцов. Функциональные зависимости широко используются при работе с базами данных — например, при поиске и удалении дубликатов, оптимизации, нормализации.

Для поиска функциональных зависимостей был разработан ряд алгоритмов. Каждый алгоритм реализует собственный подход к поиску, из-за чего на различных наборах работает по-своему, потребляя разное количество ресурсов. Поэтому, в зависимости от свойств набора, может возникнуть потребность запустить на нём какой-то конкретный алгоритм, чтобы добиться максимальной производительности.

Для этих целей на языке Java была реализована платформа Metanome [3]. Она предоставляет возможность удобно запускать различные алгоритмы поиска функциональных зависимостей на некотором наборе данных. Однако, производительности Java не всегда достаточно для требовательных к ресурсам вычислений с большим объемом входной информации. Поэтому возникла идея создать альтернативную платформу Desbordante [4]. При этом было решено использовать язык C++.

Таким образом, стало необходимо реализовать существующие алгоритмы поиска функциональных зависимостей на языке C++.

Постановка задачи

Целью работы является реализация на языке C++ и исследование производительности алгоритма DFD поиска функциональных зависимостей в рамках платформы Desbordante.

Для достижения цели были поставлены следующие задачи:

- провести обзор предметной области и работы алгоритма;
- сделать краткий обзор платформ Metanome и Desbordante;
- реализовать алгоритм DFD на языке C++ в рамках платформы Desbordante;
- сравнить производительность реализаций алгоритма на C++ и Java.

1. Обзор

1.1. Функциональные зависимости

Definition 1 *Пусть A — множество атрибутов некоторой таблицы, B — какой-то атрибут. Говорят [2], что между A и B существует функциональная зависимость (Functional Dependency, FD) $A \to B$, если любые два кортежа, совпадающие на атрибутах A, совпадают на атрибуте B. В этом случае, A называется левой частью функциональной зависимости (Left Hand Side, LHS), а B — правой (Right Hand Side, RHS).*

Definition 2 *Если B не зависит функционально ни от одного подмножества A, функциональную зависимость $A \to B$ называют [2] минимальной.*

Definition 3 *Пусть A — множество атрибутов некоторой таблицы, B — какой-то атрибут. Говорят [2], что между A и B не существует функциональной зависимости, если найдутся два кортежа, совпадающие на атрибутах A, но не совпадающие на атрибуте B. В этом случае, отношение $A \not\to B$ называется нефункциональной зависимостью (Non-Functional Dependency, non-FD), A называется левой частью зависимости (Left Hand Side, LHS), а B — правой (Right Hand Side, RHS).*

Definition 4 *Если B зависит функционально от любого надмножества A, нефункциональную зависимость $A \not\to B$ называют [2] максимальной.*

Например, рассмотрим таблицу 1, которая содержит столбцы *Фамилия*, *Имя* и *Возраст*. В таком случае, согласно определению, выполняется функциональная зависимость *Фамилия* $\to$ *Возраст* — оба человека с фамилией Петров имеют одинаковый возраст. А зависимость *Фамилия* $\not\to$ *Имя* не выполняется — люди с фамилией Петров носят разные имена. Таким образом, здесь имеет место нефункциональная зависимость.

Фамилия	Имя	Возраст
Иванов	Олег	25
Петров	Никита	19
Сидоров	Артём	35
Петров	Василий	19
Смирнов	Николай	40

Таблица 1: Пример таблицы

1.2. Metanome

Metanome[1] — открытая платформа для профилирования данных, реализованная на языке Java. Содержит множество различных алгоритмов профилирования, в том числе алгоритмы поиска функциональных зависимостей. Платформа позволяет удобно запускать эти алгоритмы на необходимых наборах данных. Благодаря встроенному frontend-клиенту поддерживается управление через браузер.

1.3. Desbordante

Desbordante[2] — консольное приложение, позволяющее находить функциональные зависимости в заданном наборе данных. Как и Metanome, позволяет запускать несколько раличных алгоритмов поиска. Реализовано на языке C++, и на данный момент находится в процессе активной разработки.

1.4. Алгоритм DFD

DFD — один из алгоритмов поиска функциональных зависимостей. Он, как и ряд других алгоритмов, представляет [2, 5] пространство поиска функциональных зависимостей как *алгебраическую решетку* (рисунок 1), состоящую из комбинаций атрибутов таблицы. DFD рассматривает каждый атрибут как RHS, и строит для него свою решетку. От-

[1] `https://hpi.de/naumann/projects/data-profiling-and-analytics/metanome-data-profiling.html`

[2] `https://github.com/Mstrutov/Desbordante`

личительной особенностью данного алгоритма является то, что он исследует решетку методом, аналогичным поиску «в глубину» (depth-first manner), причем каждая следующая из доступных вершин выбирается случайным образом. При проходе решетки алгоритм ищет минимальные FD, максимальные non-FD и отсеивает вершины на основе уже найденных зависимостей. Однако, при таком подходе возникают «островки» — части решетки, до которых поиск в глубину дойти не сможет. Такие места алгоритм обнаруживает благодаря найденным ранее максимальным non-FD, после чего запускает поиск в глубину уже внутри них. Для обнаружения FD/non-FD алгоритм либо применяет определение, исследуя подмножества/надмножества какой-то вершины, либо вычисляет и пересекает партиции.

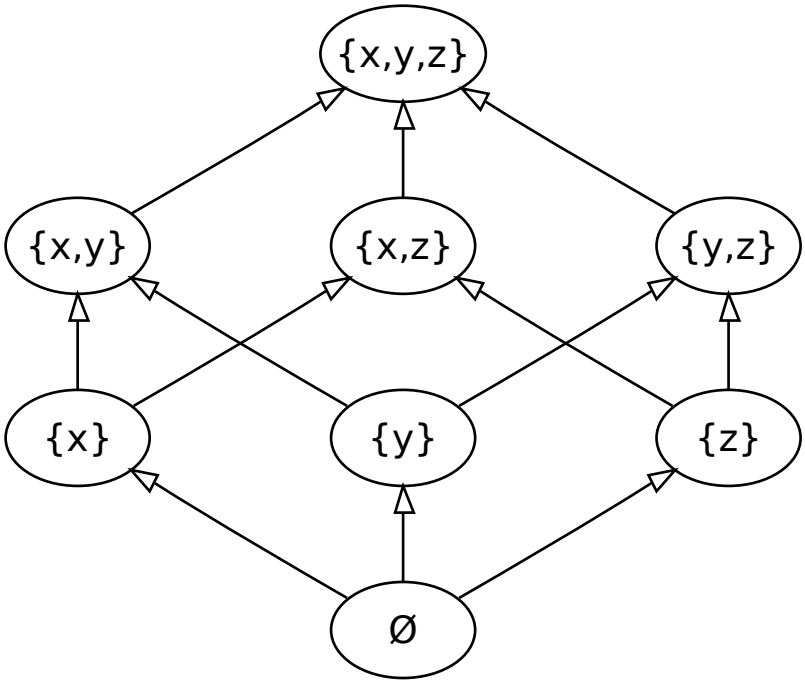

Рис. 1: **Пример алгебраической решетки.** Источник: [1]

2. Описание решения

2.1. Первоначальная реализация

Перед началом работы было решено реализовать алгоритм так, как он был описан в статье, по возможности применяя какие-либо оптимизации. Алгоритм был детально разобран, после чего была спроектирована первая архитектура проекта. Однако, оказалось, что приведенная в статье реализация содержит ряд ошибок, поэтому алгоритм не работал должным образом.

2.2. Последующая реализация

После неудачной попытки было принято решение взять некоторые детали реализации из Java-кода алгоритма для Metanome. После объединения их с предыдущей реализацией получился следующий набор классов.

- `DFD` — основной класс алгоритма. Унаследован от класса `FDAlgorithm`, который является базовым для всех алгоритмов в реализации Desbordante. Метод `execute()` запускает работу алгоритма.
- `LatticeObservations` — представляет собой ассоциативный контейнер на основе хеш-таблицы, который содержит информацию о посещенных вершинах. В процессе работы алгоритм относит посещенную вершину к одной из категорий, которые представлены перечислением `NodeCategory`. Вершина является ключом, по которому хранится её категория.
- `DependenciesMap` и `NonDependenciesMap` — две хеш-таблицы. Хранят информацию об уже ранее найденных FD/non-FD, не обязательно минимальных/максимальных. Позволяют эффективно отсекать вершины, которые являются подмножествами/надмножествами ранее найденных зависимостей.

- `PartitionStorage` — хранит ранее пересеченные партиции, чтобы избежать повторного их вычисления.

- `Vertical` — представляет собой вершину алгебраической решетки. Хранит в себе bitset, который отображает список атрибутов, составляющих вершину.

2.3. Особенности реализации

Способ хранения вычисленных партиций немного отличается от способов, предложенных в статье с алгоритмом и в релизации алгоритма для Metanome. Для этого используется PartitionStorage — измененная структура *PLI Cache*. Данная структура описана в статье [7], представляющей алгоритм Pyro.

Кроме того, каждая вершина решетки (объект класса Vertical) хранится в объекте умного указателя. Поэтому, чтобы использовать умные указатели на объекты класса Vertical в качестве ключей, были определены следующие структуры:

- `custom_comparator` — сравнивает два умных указателя на объекты типа `Vertical`. Полагаем, что умные указатели равны, если они указывают на объекты `Vertical`, представляющие из одинаковый набор атрибутов;

- `hash` — переопределение функтора для использования умных указателей в качестве ключа хеш-таблицы. Полагаем, что хеш умного указателя совпадает с хешем объекта типа `Vertical`, который в нем хранится.

3. Эксперименты и сравнение с Metanome

Тестирование реализованного алгоритма производилось с помощью библиотеки GoogleTest [6]. Алгоритм запускался на ряде наборов данных, для которых известны выполняющиеся функциональные зависимости. После этого результаты работы алогоритма сверялись с ожидаемыми.

Затем были выполнены замеры производительности. Они проводились на машине со следующими характеристиками:

- операционная система: Manjaro Linux 20.2.1 (Linux kernel 5.4.97), gcc version 10.2.0;
- процессор: IntelCore i5-8300H 4.00 GHz 8 MB cache;
- оперативная память: 8 ГБ.

В процессе экспериментов выяснилось, что реализованный в рамках Metanome алгоритм DFD не срабатывает на некоторых наборах данных. Поэтому на данном этапе замеры удалось произвести лишь на трех наборах. На каждом из наборов время работы алгоритма было измерено 10 раз, после чего были посчитаны и нанесены на логарифмическую шкалу доверительные интервалы. Результаты представлены на рисунке 2.

Исходя из полученного графика, можно сделать вывод, что реализация алгоритма на C++ выигрывает по времени у реализации на Java.

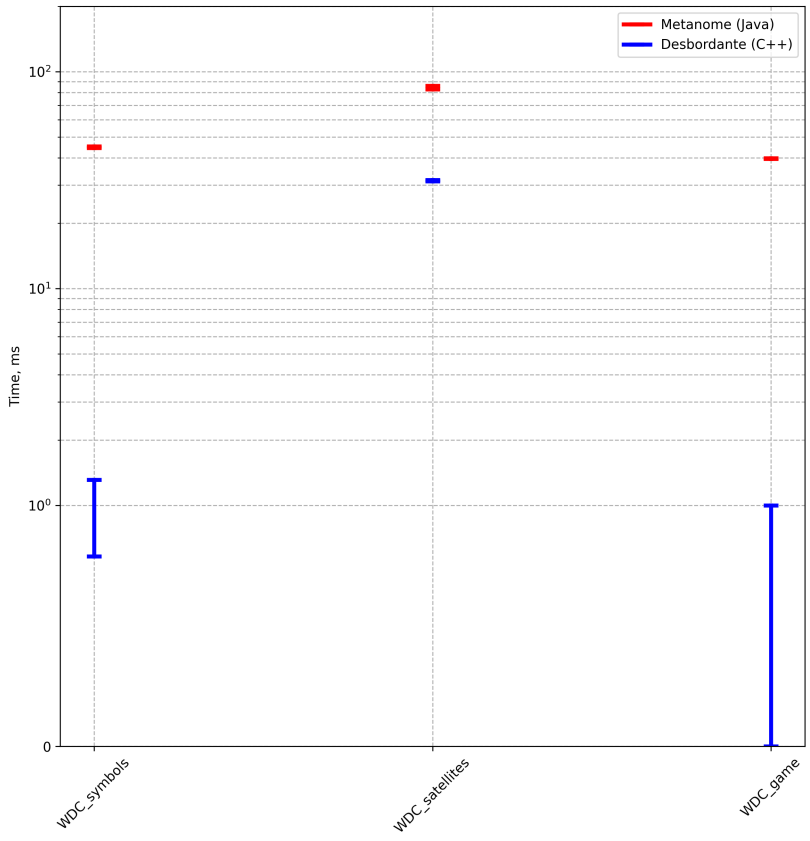

Рис. 2: **Результаты замеров**

4. Заключение

В ходе работы над реализацией алгоритма были выполнены следующие задачи:

- сделан обзор предметной области и работы алгоритма;
- сделан краткий обзор инструментов Metanome и Desbordante;
- реализован алгоритм DFD на языке C++;
- произведено сравнение производительности реализаций алгоритма на языках C++ и Java.

Исходный код алгоритма доступен на Github [8].

Список литературы

[1] By I, KSmrq, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=2118211.

[2] Abedjan Ziawasch, Schulze Patrick, Naumann Felix. DFD: Efficient Functional Dependency Discovery // Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management. — CIKM '14. — New York, NY, USA : Association for Computing Machinery, 2014. — P. 949–958. — Access mode: https://doi.org/10.1145/2661829.2661884.

[3] Data Profiling with Metanome / Thorsten Papenbrock, Tanja Bergmann, Moritz Finke et al. // Proc. VLDB Endow. — 2015. — Aug. — Vol. 8, no. 12. — P. 1860–1863. — Access mode: http://dx.doi.org/10.14778/2824032.2824086.

[4] Desbordante: a Framework for Exploring Limits of Dependency Discovery Algorithms / Maxim Strutovskiy, Nikita Bobrov, Kirill Smirnov, George Chernishev // 2021 29th Conference of Open Innovations Association (FRUCT). — 2021. — P. 344–354.

[5] Functional Dependency Discovery: An Experimental Evaluation of Seven Algorithms / Thorsten Papenbrock, Jens Ehrlich, Jannik Marten et al. // Proc. VLDB Endow. — 2015. — Jun. — Vol. 8, no. 10. — P. 1082–1093. — Access mode: https://doi.org/10.14778/2794367.2794377.

[6] GoogleTest: C++ testing and mocking framework. — https://github.com/google/googletest.

[7] Kruse Sebastian, Naumann Felix. Efficient Discovery of Approximate Dependencies // Proc. VLDB Endow. — 2018. — Mar. — Vol. 11, no. 7. — P. 759–772. — Access mode: https://doi.org/10.14778/3192965.3192968.

[8] Исходный код. Находится в ветке DFD. — https://github.com/alexandrsmirn/Desbordante.