Санкт-Петербургский государственный университет

Кафедра информационно-аналитических систем

Группа 21.Б08-мм

# Вывод метрических функциональных зависимостей в веб-интерфейсе Desbordante

***Белоконный Сергей Александрович***

Отчёт по учебной практике
в форме «Производственное задание»

Научный руководитель:
Ассистент кафедры информационно-аналитических систем Г. А. Чернышев

Санкт-Петербург
2023

# Оглавление

**Введение** **3**

**1. Постановка задачи** **4**

**2. Вводная информация** **5**

2.1. Desbordante . . . . . 5

2.2. GraphQL . . . . . 5

2.3. Обзор технологий создания веб-приложений . . . . . 6

2.4. Метрические функциональные зависимости . . . . . 7

2.5. Веб-приложение Desbordante . . . . . 7

**3. Обзор решения** **9**

**4. Практическая часть** **10**

4.1. Реализация . . . . . 10

4.2. Примеры использования . . . . . 12

**Заключение** **15**

**Список литературы** **16**

# Введение

Удобный для чтения и понимания вывод результата программы позволяет пользователю быстро и легко сделать вывод о входных данных. Так же хорошо оформленный вывод результатов работы программы повышает её презентабельность и может привлечь новых пользователей.

Вывод результата работы программы может быть сделан с помощью диаграмм, схем или таблиц. Но без возможности показать результат работы пользователю, использование этой программы станет невозможным.

Популярными способами вывода результата являются вывод в консоль, в приложении с графическим пользовательским интерфейсом или на веб-сайт. Для разработки веб-сайтов используются фреймворки и библиотеки, которые упрощают написание кода, процесс получения данных с сервера и вывода их на экран.

Desbordante — профайлер данных, состоящий из примитивов. Каждый примитив — это набор алгоритмов для поиска метаданных. У Desbordante есть две версии: консольная и веб-версия, которые позволяют работать с этими примитивами.

Разрабатывая интерфейс для вывода результата работы алгоритма в Desborbante важно помнить об удобстве пользования, об архитектуре веб-сайта и о последующем расширении функционала. Нельзя забывать, что от удобства интерфейса зависит презентабельность и виральность проекта, а выбор правильной архитектуры позволит легко его поддерживать.

# 1. Постановка задачи

Целью работы является создание интерфейса для работы с алгоритмом валидации метрических функциональных зависимостей в веб-приложении Desbordante[1]. Для её выполнения были поставлены следующие задачи:

1. реализовать запросы для получения данных о задаче с сервера с помощью языка запросов GraphQL;
2. реализовать систему хранения и получения информации о задаче и её результате;
3. реализовать интерфейс настройки примитива;
    - создание набора компонентов для создания формы настройки примитива;
    - создание запроса для получения информации об входном наборе данных;
    - добавление формы настройки примитива к уже существующим формам;
4. реализовать интерфейс отображения результатов работы;
    - создание страницы для отображения результатов работы;
    - создание компонента для вывода интерактивной таблицы.

[1] https://github.com/Mstrutov/Desbordante (дата доступа: 8 января 2023 г.).

# 2. Вводная информация

Профилирование данных — это процесс анализа данных, целью которого является получение метаданных.

Метаданные — любая сопутствующая информация о данных. Например: размер файла, дата создания или условия, при которых этот файл был создан. Также метаданными являются закономерности в данных. Валидация этих закономерностей необходима в различных областях науки, таких как физика или медицина, для подтверждения или, наоборот, опровержения гипотез.

## 2.1. Desbordante

Desbordante — профайлер данных, который может обнаруживать различные зависимости в табличных данных с помощью примитивов. Примитив — набор алгоритмов, с помощью которых проверяются закономерности. Ядро Desbordante написано на C++ и работа с ним может вестись через консоль или через веб-приложение с простым в использовании интерфейсом. Больше информации можно найти в блоге Unidata [11, 1].

## 2.2. GraphQL

GraphQL — язык запросов к API сервиса. Он позволяет разработчику получать конкретные данные с помощью настраиваемых запросов, в которых указываются конкретные поля данных, необходимые разработчику, и аргументы, по которым выполняется выборка данных. Это позволяет ускорить работу веб-приложения за счёт уменьшения количества передаваемых данных. Подробнее о GraphQL можно прочитать в статье [4] и на официальном сайте [7].

## 2.3. Обзор технологий создания веб-приложений

Для созданий веб-приложений используется язык программирования JavaScript. Для облегчения разработки веб-приложений вместе с JavaScript может использоваться надстройка TypeScript, разработанная компанией Microsoft, которая добавляет строгую типизацию в JavaScript, и специальные фреймворки. Далее рассмотрим наиболее популярные из них:

- ReactJS — JavaScript-библиотека с открытым исходным кодом, разработанная компанией Facebook, используемая для разработки пользовательских интерфейсов. Цель этой библиотеки — предоставить высокую скорость разработки и масштабируемость. Часто ReactJS используют с другими библиотеками, такими как Apollo GraphQL и NextJS. Последняя библиотека позволяет использовать отрисовку на стороне сервера (Server Side Rendering, SSR). Подробности можно узнать на главных страницах ReactJS [9], Apollo GraphQL [6] и NextJS [8].

- Angular — платформа для разработки веб-приложений на основе шаблона Модель-Представление-Контроллер (Model-View-Controller, MVC), созданная командой из Google написанная на TypeScript. Angular был разработан с использованием идей декларативного программирования для создания пользовательских интерфейсов, так как разработчики убеждены в том, что этот подход лучше всего подходит для этих целей. Подробности можно узнать на официальном сайте [5].

- Vue.js — JavaScript-фреймворк с открытым исходным кодом, разработанный для быстрого прототипирования пользовательских интерфейсов. Подробности можно узнать на официальном сайте [10].

Так как в проекте уже использовались ReactJS с TypeScript и NextJS, было решено использовать именно их.

## 2.4. Метрические функциональные зависимости

Функциональные зависимости позволяют установить зависимости между множествами атрибутов таблицы, например, что нулю в столбце A соответствует единица в столбце B, а единице в A — двойка в B. Таким образом, функциональные зависимости устанавливают строгие зависимости между данными. Но в данных могут быть ошибки, отклонения или шум и в таких данных сложно построить строгие зависимости.

Для решения этой проблемы можно использовать приближённые функциональные зависимости [2] (Approximate Functional Dependency, AFD).

Другим решением является использование метрических функциональных зависимостей [3]. Метрические функциональные зависимости рассматривают не точное равенство между двумя объектами, а некоторую метрику. Говорят, что два объекта соответствуют друг-другу, если метрика между ними не больше некоторого параметра.

Для примера, можно рассмотреть длительность фильма, указанную на разных сайтах: на первом сайте указано, что фильм длится 122 минуты, а на втором, что 120 минут. Если взять метрику

$$d(t, t') = |t - t'|$$

и параметр 3, то можно построить зависимость между фильмом и его длиной.

## 2.5. Веб-приложение Desbordante

Процесс создания и настройки задачи в Desbordante состоит из нескольких шагов:

1. Выбор примитива — например Functional Dependencies, Conditional Functional Dependencies, Association Rules, Error Detection Pipeline или Metric Verification.

2. Выбор набора данных (датасета).

3. Настройка примитива.

4. Получение результатов работы примитива.

В процессе создания задачи ей выдаётся уникальный идентификатор, по которому можно получить результат этой задачи.

# 3. Обзор решения

Для создания GraphQL-запросов была использована IDE GraphiQL[2].
Список созданных запросов:

- Запрос для получения типов данных в столбцах во входных данных.
- Запрос для получения результата работы примитива.

Для создания страницы с результатами выполнения использовался шаблон Per-Page Layouts[3], в котором, для генерации страниц с одинаковым набором меню, но с различным наполнением, в "оболочку" вставляется наполнение страницы.

В разработке компонентов для создания формы настройки примитива использовалась библиотека React-Select[4], которая позволяет создавать настраиваемые меню для выбора опций.

В разработке компонентов для страницы вывода результата работы примитива использовались уже существующие в проекте наработки.
Список созданных компонентов:

- Меню для выбора нескольких вариантов из предложенных (используется для выбора столбцов, для которых будет проверяться зависимость).
- Таблица, которая может выводить данные и позволяет взаимодействовать со своими ячейками.
- Функция, которая подготавливает данные с сервера к выводу в таблице.

[2] https://github.com/graphql/graphiql (дата доступа: 8 января 2023 г.).
[3] https://nextjs.org/docs/basic-features/layouts#per-page-layouts (дата доступа: 8 января 2023 г.).
[4] https://react-select.com (дата доступа: 8 января 2023 г.).

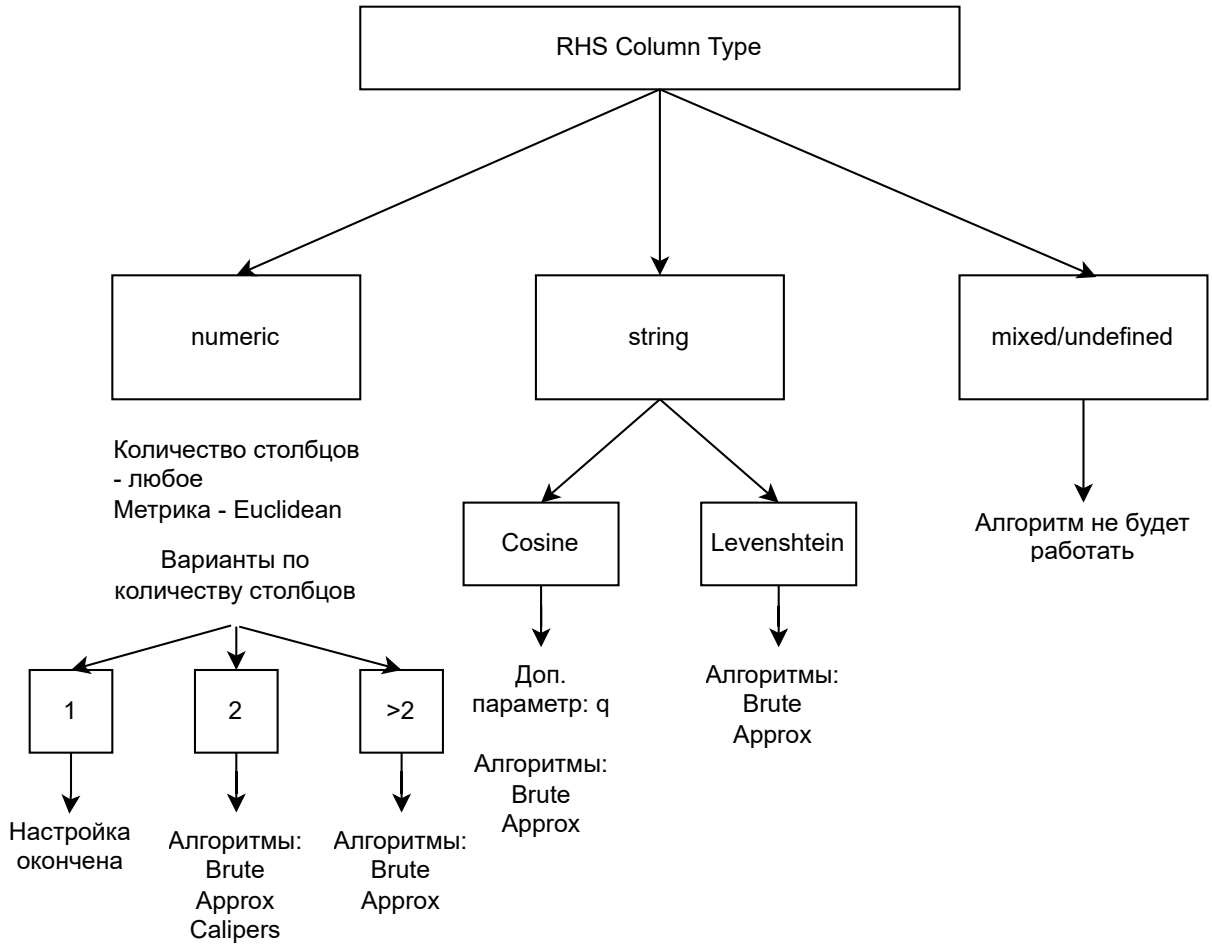



Рис. 1: Схема настройки примитива

# 4. Практическая часть

Основной целью настоящей курсовой работы является реализация интерфейса для примитива “валидация метрических зависимостей”. Примитив принимает в качестве входных параметров индексы правых и левых столбцов, метрику, алгоритм, параметр и датасет. На Рисунке 1 указано, как должен вести себя примитив в зависимости от входных параметров.

## 4.1. Реализация

В веб-приложении Desbordante для получения результатов работы примитивов пользователь должен сделать следующую последова-

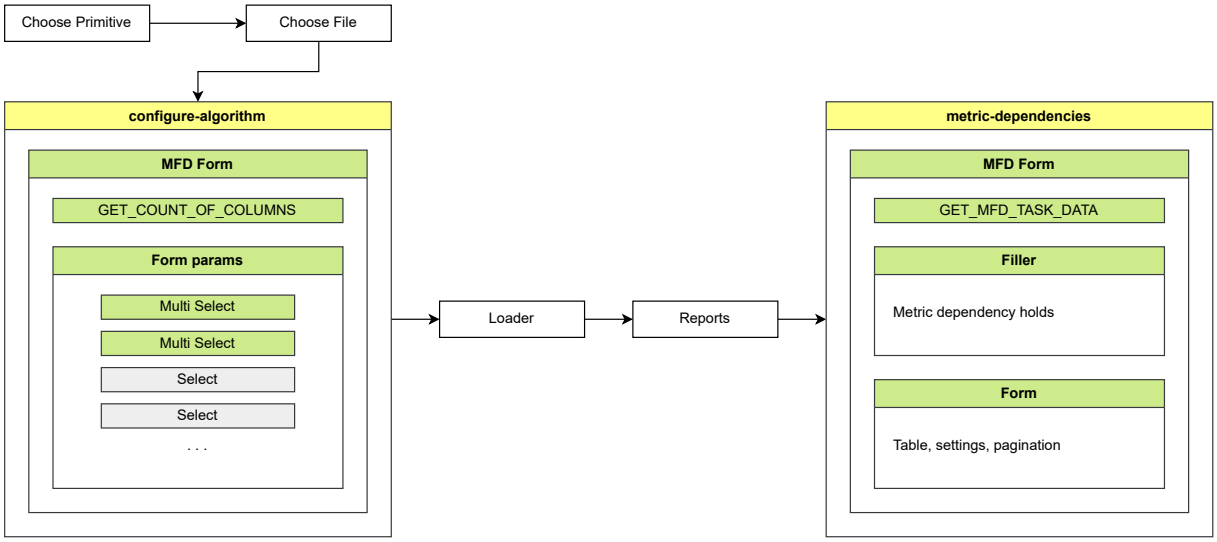


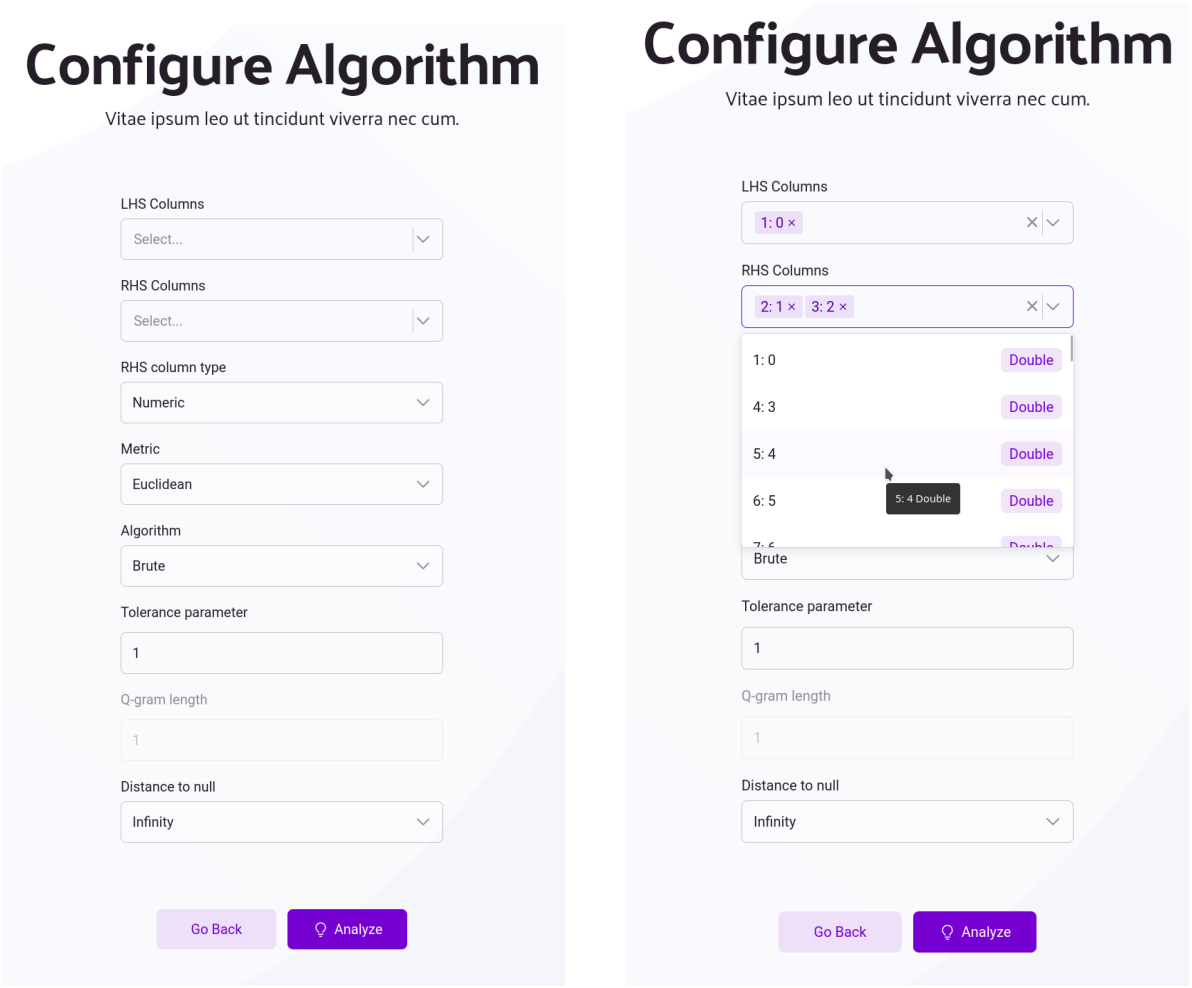

Рис. 2: Схема веб-интерфейса Desbordante

тельность действий:

1. выбрать примитив;
2. выбрать датасет;
3. настроить примитив;
4. подождать, пока сервер обработает запрос;
5. посмотреть результаты работы.

Каждый шаг в этой последовательности реализован как отдельный компонент и схема этих компонентов отображена на Рисунке 2. При разработке были написаны компоненты “MFD Form” и “metric-dependecies”, которые отвечают за настройку примитива и отображение результатов соответственно.

Все реализованные компоненты не зависят друг от друга, что позволяет использовать их и вне рамок текущих задач. Такой подход к решению задачи позволяет крайне легко масштабировать веб-приложение и даёт возможность легко добавлять новый функционал не изменяя уже существующий код.




(a) Пример стандартных значений (b) Пример выбора столбцов

Рис. 3: Форма настройки примитива

Написанный интерфейс позволяет конечному пользователю воспользоваться алгоритмами для верификации метрических зависимостей. Это расширяет функционал веб-приложения и повышает узнаваемость проекта.

## 4.2. Примеры использования

На Рисунке 3a показаны стандартные значения формы, когда пользователь открывает страницу для настройки примитива.

- В первых двух полях пользователь выбирает индексы столбцов.
- В поле “RHS Column type” пользователь в ручную или в автоматическом режиме устанавливает тип колонок.

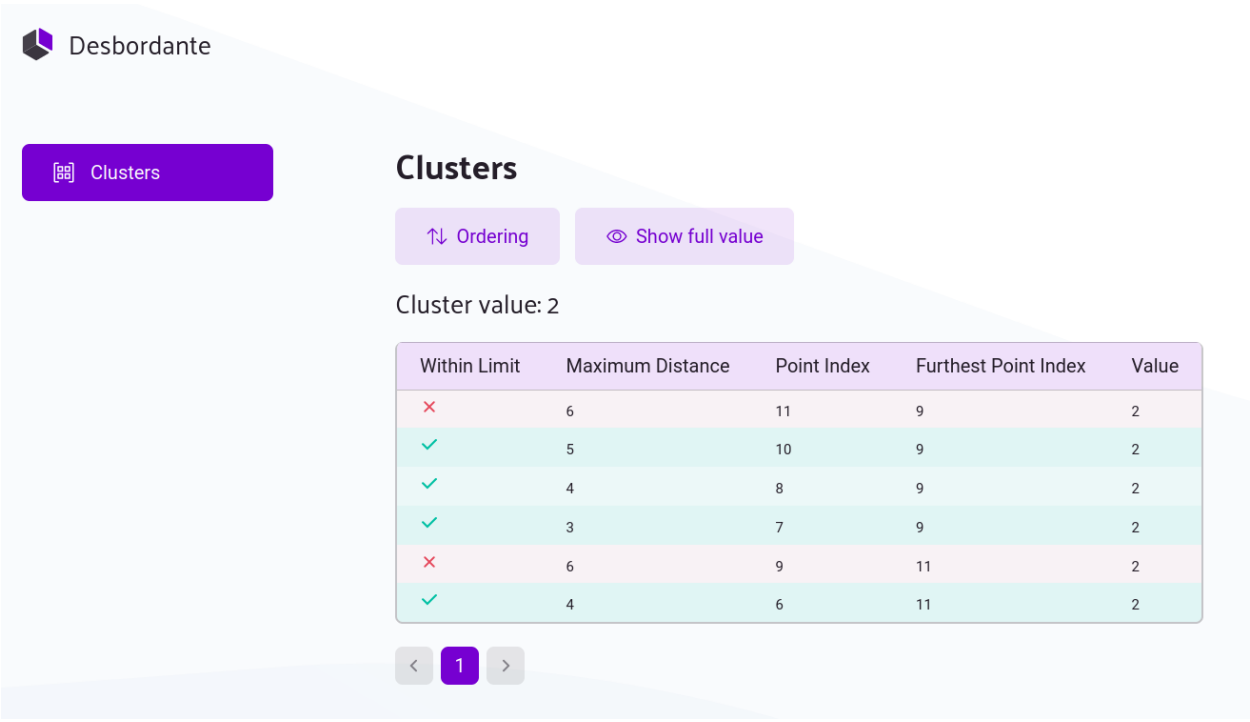



Рис. 4: Пример результата работы примитива

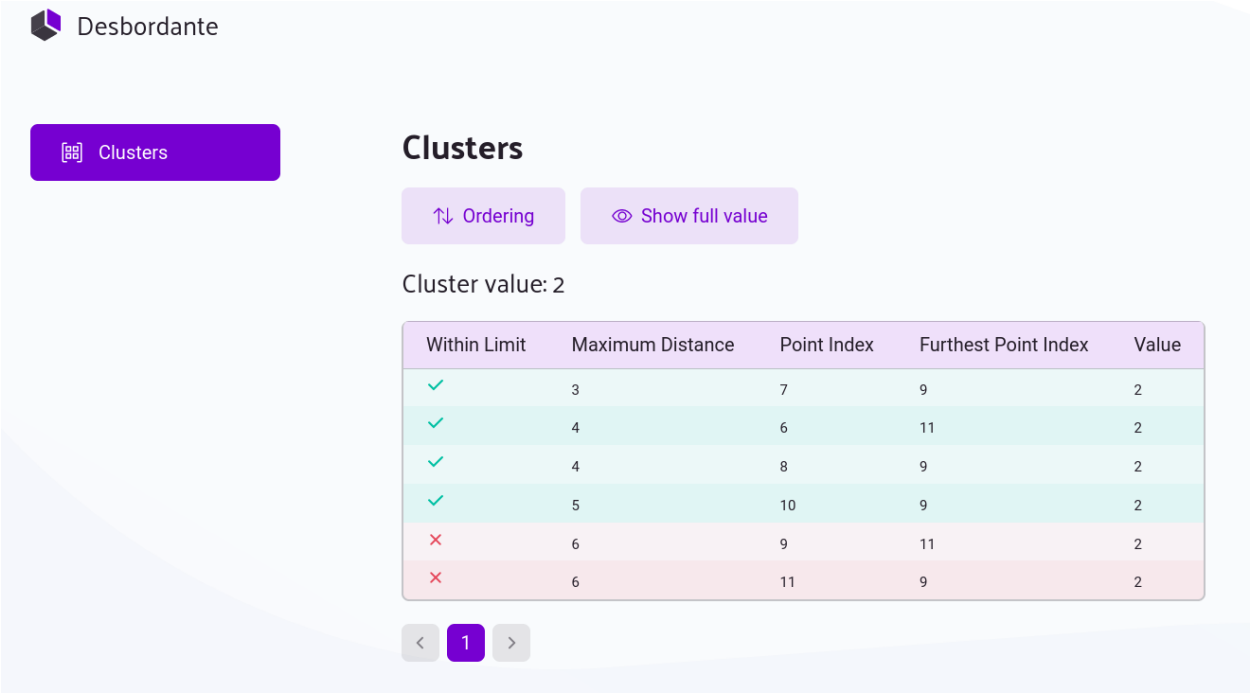



Рис. 5: Пример работы сортировки данных

- В четвёртом и пятом полях пользователь указывает, какие будут использованы метрика и алгоритм соответственно.
- В поле “Tolerance parameter” указывается число — порог для метрики.
- Поле “Q-gram length” — опциональный параметр.
- Последнее поле указывает на то, как рассматривается расстояние до пропущенных данных в столбцах.

На Рисунке 3b показан выбор индексов столбцов. Есть возможность выбрать один или несколько столбцов. Если это возможно, указывается тип столбцов.

На Рисунке 4 показывается результат работы примитива. Есть возможность отсортировать точки в кластере, показать полное значение точки. Показывается значение кластера, точки в кластере. Есть возможность выбрать кластер.

На Рисунке 5 показывается сортировка точек внутри кластера.

# Заключение

В результате работы был создан интерфейс для работы с алгоритмом валидации метрических функциональных зависимостей и были реализованы следующие задачи:

- Созданы запросы для получения данных с сервера.
- Создана система хранения данных задачи.
- Созданы интерфейс настройки алгоритма и сопутсвующие им компоненты.
- Созданы интерфейс отображения результатов работы и сопутсвующие им компоненты.

И появились задачи на будущее:

- Написать документацию на проделанную работу.
- Переписать код интерфейса настройки алгоритма для упрощения последующего развития проекта.

Подробнее о проделанной работой можно ознакомится по ссылке: https://github.com/vs9h/Desbordante/pull/59

# Список литературы


[1] Data profiling, и с чем его едят / Георгий Чернышев, Максим Струтовский, Никита Бобров и др. — URL: https://habr.com/ru/company/unidata/blog/667636 (дата обращения: 8 января 2023 г.).

[2] Kruse Sebastian, Naumann Felix. Efficient Discovery of Approximate Dependencies // Proc. VLDB Endow. — 2018. — mar. — Vol. 11, no. 7. — P. 759–772. — URL: https://doi.org/10.14778/3192965.3192968 (дата обращения: 31 января 2023 г.).

[3] Metric Functional Dependencies / Nick Koudas, Avishek Saha, Divesh Srivastava, Suresh Venkatasubramanian // 2009 IEEE 25th International Conference on Data Engineering. — 2009. — P. 1275–1278.

[4] Send Alik. Что же такое этот GraphQL? — URL: https://habr.com/ru/post/326986 (дата обращения: 8 января 2023 г.).

[5] Официальный сайт Angular. — URL: https://angular.io (дата обращения: 8 января 2023 г.).

[6] Официальный сайт Apollo GraphQL. — URL: https://www.apollographql.com (дата обращения: 8 января 2023 г.).

[7] Официальный сайт GraphQL. — URL: https://www.graphql.org (дата обращения: 8 января 2023 г.).

[8] Официальный сайт NextJS. — URL: https://nextjs.org (дата обращения: 8 января 2023 г.).

[9] Официальный сайт ReactJS. — URL: https://www.graphql.org (дата обращения: 8 января 2023 г.).

[10] Официальный сайт VueJS. — URL: https://vuejs.org (дата обращения: 8 января 2023 г.).

[11] Чернышев Георгий, Полынцов Михаил, Бобров Никита. Примитивы Desbordante: Функциональные зависимости и их применение в эксплорации и очистке данных. — URL: https://www.habr.com/ru/company/unidata/blog/679390 (дата обращения: 8 января 2023 г.).