Санкт-Петербургский государственный университет

Кафедра информационно-аналитических систем

Группа 23.Б08-мм

Реализация алгоритма валидации разностных зависимостей в «Desbordante»

Литвинов Юрий Викторович

Отчёт по учебной практике
в форме «Решение»

Научный руководитель:
асситент кафедры ИАС, Чернышев Г. А.

Санкт-Петербург
2025

Оглавление

Введение

Разностная зависимость (далее РЗ, от английского Differential Dependency) это зависимость, которая вводит ограничения на различия атрибутов в таблицах. РЗ задаётся разностными функциями, определяющими ограничения на дистанцию между конкретными атрибутами. Если некоторая РЗ вида $X \longrightarrow Y$ выполняется на некотором массиве данных, это означает, что для любых двух кортежей, у которых расстояния по атрибутам X соответствуют первой разностной функции, расстояния по атрибутам Y также соответствуют второй разностной функции. РЗ является удобным и универсальным паттерном для проверки соблюдения определённых закономерностей в таблицах.

Профилирование данных — это процесс извлечения метаданных из данных. Метаданные — это дополнительная информация об объекте, которая может включать сведения о размере файла, авторстве, времени его создания и другие характеристики. Кроме того, метаданные могут содержать информацию о выполнении различных зависимостей в данных, таких как РЗ.

Профилирование востребовано среди ученых, так как закономерности, найденные с его помощью на массивах экспериментальных данных, позволяют выдвигать гипотезы и делать выводы на их основе, что ускоряет исследования. Кроме того, профилирование может быть полезно для пользователей, желающих повысить качество данных для практических задач, удаляя ошибки, неточные дубликаты и многое другое.

В рамках профилирования данных есть две задачи:

1. Поиск зависимостей;

2. Валидация зависимостей.

Поиск позволяет выявить в данных набор выполняющихся зависимостей. Валидация, в свою очередь, позволяет проверить, выполняется ли конкретная зависимость в этих данных.

Одним из множества инструментов для профилирования данных

является проект «Desbordante[1]». Это наукоёмкий профилировщик данных, который может проверять и находить множество сложных паттернов в таблицах. В отличие от классических профилировщиков, «Desbordante» нацелен на выявление нетривиальных зависимостей в данных, которые задаются математически при помощи примитивов. В «Desbordante» реализован алгоритм для поиска разностных зависимостей, но отсутствует алгоритм для их валидации.

Поэтому в данной работе будет спроектирован алгоритм валидации РЗ. Также на его основе будет реализован класс валидации РЗ в рамках проекта «Desbordante».

[1] https://github.com/Desbordante/desbordante-core

1. Постановка задачи

Целью работы является реализация алгоритма валидации разностных зависимостей на массиве данных. Для её выполнения были поставлены следующие задачи:

1. Спроектировать и описать алгоритм валидации РЗ;
2. Реализовать алгоритм валидации РЗ на языке программирования C++ в рамках проекта «Desbordante»;
3. Разработать концепцию исключений для РЗ, и реализовать их поиск для случаев, когда РЗ не выполняются на массиве данных;
4. С помощью библиотеки pybind11 добавить возможность использования алгоритма валидации в программах на языке Python;
5. Смоделировать таблицу данных и реализовать пример использования валидатора РЗ на ней, используя язык Python;
6. Создать модульные тесты для проверки корректности работы алгоритма валидации;
7. Провести эксперимент для оценки производительности алгоритма.

2. Обзор

2.1. Основные определения

Все нижеприведенные определения взяты из статьи [3].

Пусть R — заданное отношение, B — некоторый атрибут данного отношения, а $dom(B)$ — домен этого атрибута.

Определение 1. *Метрика на $dom(B)$ — отображение $d_B : dom(B) \times dom(B) \to \mathbb{D}$, где $\mathbb{D}$ — значения, которыми может измеряться расстояние между элементами данной метрики.*

Данное отображение должно соответствовать трём аксиомам:

1. $d_B \geq 0$ *— аксиома неотрицательности;*
2. $d_B(a, b) = 0 \Rightarrow a = b$ *— аксиома тождества;*
3. $d_B(a, b) = d_B(b, a)$ *— аксиома симметричности;*

Определение 2. *Разностная функция, заданная на атрибуте B, $\phi[B]$, определяет ограничения на разницу над атрибутом B, то есть устанавливает допустимые значения множества $\mathbb{D}$. Она возвращает булевы значения. Функция возвращает значение «истина», если для любого кортежа данных $t_1, t_2 \in I$, где I — экземпляр отношения R, расстояние между t_1 и t_2 по атрибуту B соответствует заданным ограничениям.*

Определение 3. *Разностная функция, заданная на множестве атрибутов L, возвращающая булевое значение, представляет собой логическое умножение этой функции на каждом из $B_i \in L$:*

$$\phi[L] = \phi[B_0] \wedge \cdots \wedge \phi[B_n].$$

Определение 4. *РЗ — это утверждение, обозначаемое $\phi_L[X] \longrightarrow \phi_R[Y]$, где $\phi_L[X]$ и $\phi_R[Y]$ — разностные функции, а $X \subseteq R$, $Y \subseteq R$. Если РЗ выполняется, то для любых кортежей данных t_1, t_2 экземпляра I отношения R, для которых разностная функция $\phi_L[X]$ истинна, следует, что разностная функция $\phi_R[Y]$ также истинна.*

Рассмотрим пример некоторой РЗ. В таблице, хранящей цены на авиарейсы, соблюдается следующая РЗ:

$$[flight\_id(=0)];\ [date(\leq 7)] \longrightarrow [price(\leq 250)].$$

Данная запись обозначает, что для любых двух одинаковых рейсов в течение недели разница в цене не превышает 250 платежных единиц. Пример таблицы, в которой данная зависимость выполняется, приведен ниже.

flight_id	date	price
1123	06.08.2023	1000
1123	11.08.2023	1120
1123	14.08.2023	1200
25	19.08.2023	370
25	22.08.2023	200
11	01.09.2023	850
25	02.09.2023	120
11	07.09.2023	700
11	12.09.2023	460
25	11.10.2023	200

2.2. Обзор существующих решений

На данный момент в «Desbordante» присутствует возможность поиска РЗ, но отсутствует функциональность валидации РЗ. Кроме того, по итогу поиска, проведенного автором настоящей работы, не было обнаружено готовых решений для валидации РЗ. Также автор работы [4] утверждает, что не существует сторонних инструментов-аналогов, которые позволили бы проводить валидацию РЗ.

Поднимая тему актуальности валидаторов зависимостей, необходимо отметить, что ученые по всему миру занимаются валидацией различных закономерностей в данных. Например, в статье [2] представлено решение для валидации запрещающих ограничений (Denial Constraints,

DC) с участием исследователей из Microsoft. Также есть статья [1] опубликованная в журнале IEEE Access про решение для валидации функциональных зависимостей (Functional Dependency, FD), условных функциональных зависимостей (Conditional Functional Dependency, CFD) и DC. Кроме того, в рамках проекта «Desbordante» реализованы валидаторы множества различных паттернов. В связи с этим данная работа приобретает актуальность.

В ходе работы над алгоритмом поиска РЗ [5] в «Desbordante», Михаилом Синельниковым были реализованы основные структуры данных для работы с РЗ, за что автор настоящей работы выражает ему благодарность. Это следующие классы и структуры данных:

1. Тип `DF` — предназначен для хранения разностной функции;
2. Структура `DD` — предназначена для хранения разностной зависимости в виде пары разностных функций типа `DF`;
3. Структура `DFConstraint` — упорядоченная тройка, состоящая из названия атрибута, верхней и нижней границ расстояний;
4. Структура `DDString` — предназначена для хранения разностной зависимости в виде массива структур `DFConstraint`. В отличие от структуры `DD`, данная структура, помимо границ разностной функции, содержит название атрибута.

3. Описание решения

3.1. Алгоритм

Автором настоящей статьи был спроектирован и предложен следующий алгоритм для валидации РЗ в таблицах.

Перед выполнением алгоритму передаётся таблица и разностная зависимость.

Algorithm 1 Алгоритм валидации РЗ

```
1:  Input: φ_L[X] ⟶ φ_R[Y], table
2:  Output: false/true
3:  correct_rows = empty
4:  for row1 in table do
5:      for row2 in table do
6:          for attr in X do
7:              if φ_L(row1[attr], row2[attr]) then
8:                  correct_rows.append({row1, row2})
9:              else
10:                 break
11:             end if
12:         end for
13:     end for
14: end for
15: for [row1, row2] in correct_rows do
16:     for attr in Y do
17:         if not φ_R(row1[attr], row2[attr]) then
18:             return false
19:         end if
20:     end for
21: end for
22: return true
```

Формально алгоритм состоит из следующих шагов:

1. На первом шаге алгоритма производится поиск пар строк, для которых первая разностная функция по атрибутам X истинна. Этот шаг выполняется с 4 по 14 строку алгоритма. В этих строках мы перебираем все пары строк в таблице и проверяем на истинность первой разностной функции для атрибутов в них;

2. Далее, с 14 по 21 строку алгоритма, выполняется второй шаг, который заключается в проверке выбранных пар кортежей на истинность второй разностной функции по атрибутам Y. Если для каждой пары разностная функция истинна, то РЗ выполняется на массиве данных, в противном случае — не выполняется.

Данный алгоритм завершает свою работу сразу же, если на втором шаге найдена пара кортежей, для которых разностная функция не истинна. В ходе работы алгоритм будет изменен так, чтобы на втором шаге он не завершал свою работу, а сохранял информацию о парах кортежей, нарушающих выполнение зависимости.

3.2. Реализация алгоритма на языке C++ в проекте «Desbordante»

Для реализации алгоритма валидации РЗ был создан класс `DDVerifier` в проекте «Desbordante». Его структура представлена на диаграмме ниже.

Этот класс наследует функциональность класса `Algorithm`. Это позволяет легко инициализировать данные для алгоритма с помощью переопределения нескольких методов родительского класса.

В классе `DDVerifier` были реализованы методы `GetRowsWhereLhsHolds`, `CheckDFOnRhs` и `VerifyDD` которые выполняют алгоритм валидации.

Метод `GetRowsWhereLhsHolds` возвращает объект типа `std::vector<std::p` Это массив пар номеров строк данной таблицы, для которых истинна первая разностная функция зависимости.

Метод `CheckDFOnRhs` проверяет истинность второй разностной функции зависимости для полученных пар номеров строк таблицы. Если разностная функция не истинна для всех пар, то пары, нарушающие выполнение РЗ, добавляются в поле `highlights_`, а их количество сохраняется в поле `num_error_rhs_`.

В методе `VerifyDD` выполняется алгоритм валидации с использованием двух вышеописанных функций. В этом методе по результатам

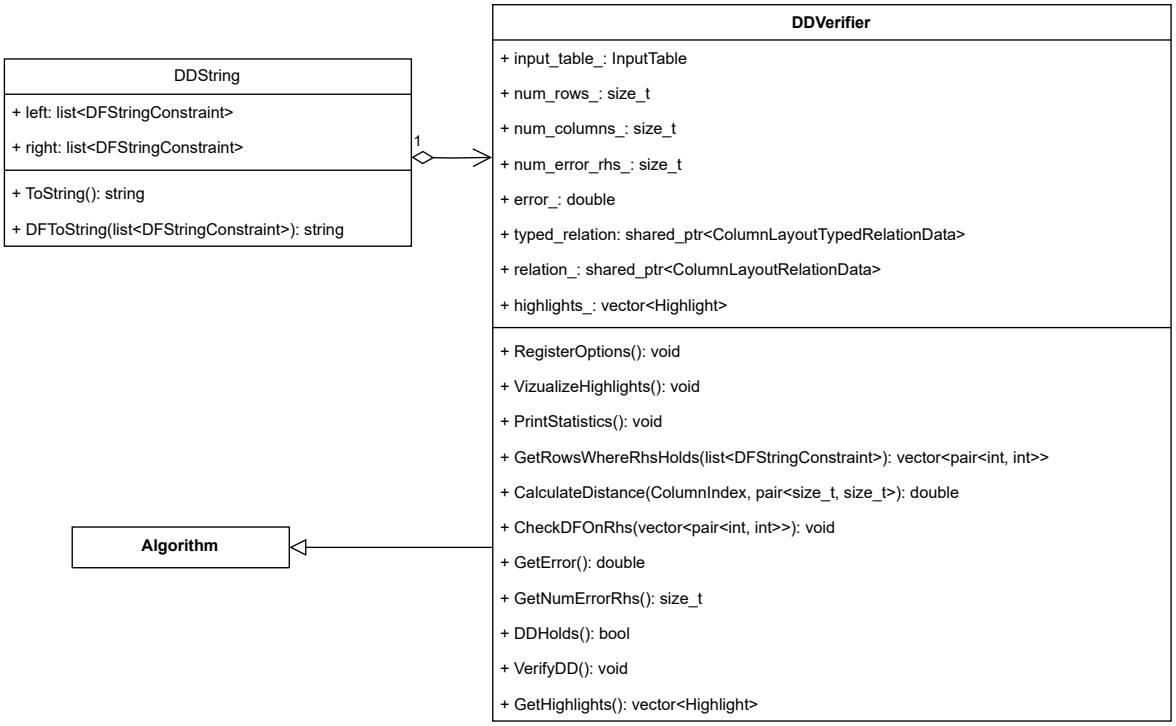

Рис. 1: Структура класса DDVerifier

валидации заполняется поле класса `error_`, которое содержит отношение количества пар кортежей, нарушающих РЗ, к количеству пар, для которых проверялась истинность разностных функций.

Также в классе `DDVerifier` реализован ряд служебных функций, которые позволяют получать доступ к некоторым приватным полям класса или выводить результаты валидации РЗ. Таковыми являются:

1. `VizualizeHighlights` — выводит информацию об исключениях;
2. `PrintStatistics` — выводит информацию о том, выполняется ли РЗ в таблице;
3. `GetError` — предоставляет доступ к полю `error_`;
4. `GetNumErrorEhs` — предоставляет доступ к полю `num_error_rhs_`;
5. `DDHolds` — показывает, выполняется ли РЗ или нет;
6. `GetHighlights` — предоставляет доступ к полю `highlights_`.

3.3. Реализация поиска исключений

Важной особенностью «Desbordante» является наличие поиска исключений. Исключение — это концепция элемента данных, нарушающего выполнение зависимости. В случаях, когда проверяемый в таблице примитив не соблюдается, исключения позволяют получить больше информации о данных, не соответствующих данной зависимости, а также провести их анализ.

В «Desbordante» предусмотрено два вида представления исключений: в виде пар или кластеров. Кластеры являются более удобным представлением исключений, так как они сразу предоставляют весь контекст поиска ошибок. Однако, если для хранения исключений некоторого паттерна кластер не подходит, используется способ хранения в виде пар. Недостатками пар являются сложность в работе с ними и их трактовка.

Автор предлагает использовать в валидаторе РЗ структуру данных, которая хранит информацию о паре строк и атрибуте для которого разностная функция не истинна.

Данной структуре данных соответствует класс `Highlight`. Структура данного класса изображена на Рисунке 2. В этом классе также хранится значение метрики для данных атрибутов. Для хранения исключений в классе валидатора создано поле `std::vector<Highlight> highlights_`. Массив `highlights_` состоит из экземпляров класса `Highlight`.

Это поле заполняется на этапе выполнения метода `CheckDDOnRhs`. Если значения атрибутов в некоторой паре строк имеют расстояние, не соответствующее ограничениям, заданным разностной функцией, то номера такой пары строк и столбца помещаются в экземпляр класса `Highlight`, а затем в поле `highlights_`.

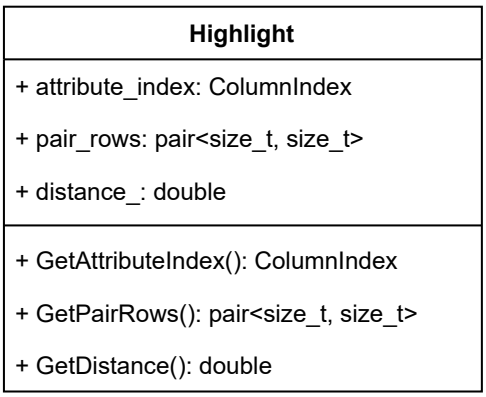

Рис. 2: Структура класса Highlight

3.4. Реализация возможности использования алгоритма валидации в программах на языке программирования Python

`Pybind11` — это библиотека, позволяющая интегрировать библиотеки из `C++` в `Python`. Для добавления возможности использования класса `DDVerifier` в языке `Python` была написана функция `BindDDVerification`, которая с помощью инструментов `pybind11` в библиотеке `desbordante` добавляет объект `DDVerifier`, соответствующий классу валидатора. К нему привязаны следующие функции:

1. `dd_holds` — показывает, удерживается ли РЗ в таблице;
2. `get_error` — возвращает отношение количества пар строк, в которых атрибуты не удовлетворяют разностной функции, ко всем парам строк, на которых проверялось выполнение зависимости;
3. `get_num_error_pairs` — возвращает количество пар строк, в которых атрибуты не соответствуют разностной функции;
4. `get_highlights` — возвращает массив с исключениями.

Также для комфортной работы с ним были реализованы и добавлены в язык `Python` функции для создания разностных функций (`create_df`)

и зависимостей (`create_dd`), аналогичные соответствующим структурам данных в языке `C++`.

Для иллюстрации примеров использования валидатора была спроектирована таблица `stores_dd.csv`:

Таблица 2: stores_dd.csv

store_name	product_name	category	stock_quantity	price_per_unit
Bestbuy NY	Apple iPhone 15	Smartphones	50	999
Bestbuy LA	Apple iPhone 15	Smartphones	30	1029
Walmart TX	Apple iPhone 15	Smartphones	40	989
Bestbuy NY	Samsung Galaxy S23	Smartphones	25	899
Bestbuy LA	Samsung Galaxy S23	Smartphones	20	920
Walmart TX	Samsung Galaxy S23	Smartphones	35	880
Bestbuy NY	Sony WH-1000XM5	Headphones	15	399
Bestbuy LA	Sony WH-1000XM5	Headphones	18	410
Walmart TX	Sony WH-1000XM5	Headphones	10	395
BestBuy NY	Apple MacBook Air	Laptops	10	1299
Bestbuy LA	Apple MacBook Air	Laptops	8	1349
Walmart TX	Apple MacBook Air	Laptops	12	1289

Пример использования алгоритма на `Python` представлен ниже.

```python
import desbordante as db
TABLE = 'examples/datasets/stores_dd.csv'

#differential functions
lhs1 = [db.dd_verification.create_df('product_name', 0, 0)]
rhs1 = [db.dd_verification.create_df('stock_quantity', 0, 20),
        db.dd_verification.create_df('price_per_unit', 0, 60)]
#differential dependency
dd1 = db.dd_verification.create_dd(lhs1, rhs1)

lhs2 = [db.dd_verification.create_df('category', 0, 0)]
rhs2 = [db.dd_verification.create_df('stock_quantity', 0, 10)]

dd2 = db.dd_verification.create_dd(lhs2, rhs2)

algo = db.dd_verification.algorithms.DDVerifier()
algo.load_data(table=(TABLE, ',', True))
algo.execute(dds=dd1)

#indicates whether the dependency is holds.
holds = algo.dd_holds()

if holds:
    print('Fisrt DD holds!')
else:
    print('First DD not holds!')

holds = algo.execute(dds=dd2)

if holds:
    print('Second DD holds!')
else:
    print('Second DD not holds!')

highlights = algo.get_highlights()

print(f'Second dd has {len(highlights)} pairs of rows, which not satisfy do df.')
print(f'There are:')
for hl in highlights:
    print(f'In {hl.pair_rows[0]} and {hl.pair_rows[1]} rows on attribute with number {hl.attribute_index}')
    print(f'Distance does not satisfy DF with value of metric {hl.distance}.\n')
```

Listing 1: Пример валидации РЗ на языке программирования Python

Данный код выведет следующее:

```
Fisrt DD holds!
Second DD not holds!
Second dd has 7 pairs of rows, which not satisfy do df.
There are:
In 0 and 1 rows on attribute with number 3
Distance does not satisfy DF with value of metric 20.0.

In 0 and 3 rows on attribute with number 3
Distance does not satisfy DF with value of metric 25.0.

```

```
In 0 and 4 rows on attribute with number 3
Distance does not satisfy DF with value of metric 30.0.

In 0 and 5 rows on attribute with number 3
Distance does not satisfy DF with value of metric 15.0.

In 2 and 3 rows on attribute with number 3
Distance does not satisfy DF with value of metric 15.0.

In 2 and 4 rows on attribute with number 3
Distance does not satisfy DF with value of metric 20.0.

In 4 and 5 rows on attribute with number 3
Distance does not satisfy DF with value of metric 15.0.
```

Следовательно, в таблице stores_dd.csv выполняется следующая РЗ:

$$product\_name[0,0] \longrightarrow stock\_quantiy[0,20]; price\_per\_unit[0,60]$$

И не выполняется:

$$category[0,0] \longrightarrow storck\_quatity[0,10]$$

.

3.5. Пример использования валидатора РЗ

В «Desbordante», помимо алгоритмов валидации и поиска РЗ, реализованы алгоритмы, обеспечивающие аналогичную функциональность для других зависимостей. Большинство из этих примитивов являются сложными для понимания, поэтому для снижения порога входа пользователей в проекте предусмотрены примеры для каждого из них. Соответственно, автором данной работы также был создан и реализован пример валидации РЗ на языке программирования `Python` в рамках проекта «Desbordante»[2].

В примере проводится валидация следующих зависимостей в таблице `stores_dd.csv`:

[2] https://github.com/unfadingDawn/desbordante-core-my/blob/main/examples/basic/verifying_dd.py

1. $product\_name[0,0] \longrightarrow stock\_quantity[0,20];\ price\_per\_unit[0,60]$

2. $store\_name[0,0];\ category[0,0] \longrightarrow stock\_quantity[0,25];$

3. $store\_name[0,0] \longrightarrow stock\_quantity[0,25];$

Первые две зависимости в данной таблице выполняются, а последняя — нет.

В результате запуска кода, проверяющего описанные зависимости, выводится следующая информация:

```

This is an example of validating differential dependencies.

Differential dependencies were introduced by Song, Shaoxu,
and Chen, Lei in their 2011 article, "Differential
Dependencies: Reasoning and Discovery," published in ACM
Transactions on Database Systems (Vol. 36, No. 3).

A differential dependency (DD) defines constraints on the
differences between attribute values within a table.
These dependencies are formalized using differential
functions, which specify permissible distance ranges
between attribute values.

For instance, consider the following differential
dependency:

flight_id[0,0]; date[0, 7] -> price[0, 250].

This expression is composed of three differential functions:
flight_id[0,0], date[0, 7], and price[0, 250]. If this
dependency is applied to a flight schedule table, it
implies that for any two tuples where the difference
for identical flights (sharing the same id) in the date
attribute is no more than 7 days, the corresponding
difference in the price attribute must not exceed 250 units.
In simpler terms, this means that the price difference
between any two identical flights scheduled within the
same week will be within a 250-unit margin.

Flight schedule table:

   flight_id        date  price
0         25  2023-08-19    370
```

```
1          25  2023-08-22    200
2          11  2023-09-01    850
3          25  2023-09-02    120
4          11  2023-09-07    700
5          11  2023-09-12    460
6          25  2023-10-11    200

It can be observed that this dependency holds true for the
flights table.

To illustrate this further, we will examine the
stores_dd.csv dataset and validate a specified
differential dependency.

It is worth noting that this validator implements
standard distance metrics for numerical data types
and dates, as well as the Levenshtein distance for
strings.

Finally, an additional example of differential dependency
mining, using the Split algorithm, is also available in
the Desbordante project.

    store_name        product_name     category  stock_quantity
   price_per_unit
0   BestBuy NY     Apple iPhone 15  Smartphones              50
   999
1   BestBuy LA     Apple iPhone 15  Smartphones              30
   1029
2   Walmart TX     Apple iPhone 15  Smartphones              40
   989
3   BestBuy NY  Samsung Galaxy S23  Smartphones              25
   899
4   BestBuy LA  Samsung Galaxy S23  Smartphones              20
   920
5   Walmart TX  Samsung Galaxy S23  Smartphones              35
   880
6   BestBuy NY     Sony WH-1000XM5   Headphones              15
   399
7   BestBuy LA     Sony WH-1000XM5   Headphones              18
   410
8   Walmart TX     Sony WH-1000XM5   Headphones              10
   395
9   BestBuy NY   Apple MacBook Air      Laptops              10
   1299
10  BestBuy LA   Apple MacBook Air      Laptops               8
```

```
   1349
11  Walmart TX   Apple MacBook Air      Laptops              12
   1289

--------------------------------------------------------------------------------

Example #1

To better understand the differential dependency concept,
let's examine a practical example.

Consider the following DD:

product_name[0, 0] -> stock_quantity[0, 20]; price_per_unit[0, 60]

This DDholds.

This dependency requires that for any two records
with the same product_name, the difference in
their stock_quantity must not exceed 20, and
the difference in price_per_unit must not
exceed 60.

In other words, for the same product sold
across different stores, stock levels cannot
vary by more than 20 units, and the price
cannot vary by more than 60 units.
--------------------------------------------------------------------------------

Example #2

Now, let`s check the DD:

store_name[0, 0] -> stock_quantity[0, 25]

This means that for a single product, the
difference in stock quantity between any
two stores cannot exceed 25 units.

This DDdoesn‘thold.

Desbordante can automatically detect pairs of
violating tuples. ’Lets do it.

Desbordante returned 4 pairs of records that
violate the stock_quantity threshold.
```

The error threshold of a differential dependency is the ratio of record pairs that satisfy the left-hand side (LHS) but violate the right-hand side (RHS), to the total number of record pairs that satisfy the LHS.

In our example error threshold is: 0.2222222222222222

Now, let us look at the pairs of violating tuples:

1) *BestBuyNY* Apple iPhone 15 Smartphones 50999
7) *BestBuyNY* Sony WH-1000XM5 Headphones 15 399

1) *BestBuyNY* Apple iPhone 15 Smartphones 50 999
10) *BestBuy* NY Apple MacBook Air Laptops 10 1299

3) *WalmartTX* Apple iPhone 15 Smartphones 40 989
9) *WalmartTX* Sony WH-1000XM5 Headphones 10 395

3) *WalmartTX* Apple iPhone 15 Smartphones 40 989
12) *WalmartTX* Apple MacBook Air Laptops 12 1289

Clearly, this DD has no practical significance, however, it remains useful for demonstration purposes.

Example #3

Our previous dependency failed because it didn't account for key operational factors. For instance, stock levels are influenced by product demand and store size, not just the store location.

To address this, we refine the dependency by adding a constraint on the product_category. Next DD, which we are going to check:

$store_name[0,0]; category[0,0] -> stock_quantity[0,25]$

This *DDholds.*

This differential dependency states:

For tuples with the same store_name and category,
the stock_quantity must not differ by more than
25 units.

Example #4

Validation of Differential Dependencies can
also be utilized for mitigating data inaccuracies.

Consider the grades_dd.py table.

	student_id	student_name	course	exam_date	grade_score
0	1	Alice	Math	2025-06-15	95
1	1	Akice	Math	2025-06-20	92
2	1	Alice	Physics	2025-06-18	80
3	2	Alice	Math	2025-06-21	42
4	2	Bob	Math	2025-06-16	70
5	2	Bob	Math	2025-06-21	68
6	3	Charlie	Chemistry	2025-06-17	55

We will check the DD:

$student_id[0,0] -> student_name[0,0]$

This *DDdoesn'thold.*

1) 1 *Alice* Math 2025-06-15 95
2) 1 *Akice* Math 2025-06-20 92

2) 1 *Akice* Math 2025-06-20 92
3) 1 *Alice* Physics 2025-06-18 80

4) 2 *Alice* Math 2025-06-21 42
5) 2 *Bob* Math 2025-06-16 70

4) 2 *Alice* Math 2025-06-21 42
6) 2 *Bob* Math 2025-06-21 68

Error threshold: 0.6666666666666666

We have two pairs of rows that do not conform to
the constraints imposed by the DD. Let's rectify
this data error by changing "Akice" to "Alice"
in the first row of the "student_name" column

```
and then re-evaluate the DD's over this table.

   student_id student_name     course   exam_date  grade_score
0           1        Alice       Math  2025-06-15           95
1           1        Alice       Math  2025-06-20           92
2           1        Alice    Physics  2025-06-18           80
3           2        Alice       Math  2025-06-21           42
4           2          Bob       Math  2025-06-16           70
5           2          Bob       Math  2025-06-21           68
6           3      Charlie  Chemistry  2025-06-17           55

After correcting the error, the error threshold
dropped to 0.3333333333333333

A potential error may also exist in the left-hand
side of the DD. For instance, in rows 3, 4 and 5
of the table, we have three entries with an identical
student_id but a different student_name. Let's
correct this error and observe the subsequent changes.

   student_id student_name     course   exam_date  grade_score
0           1        Alice       Math  2025-06-15           95
1           1        Alice       Math  2025-06-20           92
2           1        Alice    Physics  2025-06-18           80
3           1        Alice       Math  2025-06-21           42
4           2          Bob       Math  2025-06-16           70
5           2          Bob       Math  2025-06-21           68
6           3      Charlie  Chemistry  2025-06-17           55

After correcting the error, the error threshold
dropped to 0.0 and the DDholds.
```

Listing 2: Результат валидации некоторых РЗ на спроктированной таблице

Данная версия примера является базовой и в дальнейшем будет улучшена для повышения её содержательности.

3.6. Тестирование

Во время работы над валидатором РЗ было разработано 14 тестов для отладки и проверки корректности работы алгоритма. Тестирование

проводилось на таблице `TestDD.csv`[3], так как в ней можно вручную проверить выполнение конкретной РЗ. Была проверена корректность валидации зависимостей, а также поиск исключений для них.

Для тестирования использовалась библиотека googletest, которая позволяет удобно тестировать программы, написанные на языке `C++`.

Также все тесты автоматически включаются в CI проекта для удобства проверки работоспособности валидатора при внесении изменений в код в будущем.

[3] https://github.com/Desbordante/desbordante-core/blob/main/test_input_data/TestDD.csv

4. Эксперимент

Нагрузочное тестирование проводилось тестовом стенде с следующими характеристиками:

1. Процессор — Intel Core i5 12450H с тактовой частотой 2.1 ГГц;
2. 16GiB оперативной памяти;
3. Операционная система — Manjaro linux;
4. Ядро — 6.12.17-1-MANJARO (64-bit);
5. Компилятор — gcc 14.2.1.

Проверялось выполнение следующих РЗ в таблице `Workshop.csv` (945 строк, 6 столбцов):

1. $job\_post[0,0] \longrightarrow salary[0,1100]$;
2. $job\_post[0,0] \longrightarrow salary[0,300]$.

В следующей таблице представлено время выполнения валидации каждой из зависимостей:

Разностная зависимость	Удерживается	Время(мс)
$job\_post[0,0] \longrightarrow salary[0,1100]$	да	1166
$job\_post[0,0] \longrightarrow salary[0,300]$	нет	1154
Функциональная зависимость	Удерживается	Время(мс)
$job\_post \longrightarrow salary$	нет	1

Разница во времени валидации различных разностных зависимостей незначительна, поскольку независимо от того, выполняется ли РЗ на массиве данных, необходимо проверять истинность второй разностной функции для каждой пары строк таблицы, для которой истинна первая разностная функция. Это необходимо для сбора информации об исключениях в данной таблице.

Также в таблицу добавлена строка с функциональной зависимостью $job\_post \longrightarrow salary$. Если функциональная зависимость выполняется

на массиве данных, то для каждого поля из левой части зависимости соответствует единственное поле из правой части. Функциональная зависимость является частным случаем РЗ (если у обеих частей РЗ разностная функция имеет вид $X[0, 0]$), и можно заметить, что валидация разностных зависимостей является более ресурсозатратной, чем валидация функциональных зависимостей.

Заключение

В результате выполнения данной работы стала возможной валидация РЗ в таблицах. По результатам учебной практики:

1. Спроектирован алгоритм валидации РЗ;
2. Данный алгоритм реализован в ядре проекта «Desbordante» на языке программирования C++;
3. Предложена концепция исключений в «Desbordante», а также реализован их поиск для случаев, когда РЗ не выполняется на массиве данных;
4. Добавлена возможность использования алгоритма валидации РЗ в программах на Python с помощью библиотеки `pybind11`;
5. Реализован пример использования валидатора и массив данных для него;
6. Алгоритм протестирован на корректность;
7. Проведено нагрузочное тестирование алгоритма.

Исходный код алгоритма[4] доступен на GitHub, PR 543. Новая функциональность, реализованная в данном проекте, находится на стадии рассмотрения.

[4]https://github.com/Desbordante/desbordante-core/pull/543

Список литературы

[1] One-Pass Inconsistency Detection Algorithms for Big Data / Meifan Zhang, Hongzhi Wang, Jianzhong Li, Hong Gao // IEEE Access. — 2019. — Vol. 7. — P. 22377–22394.

[2] Liu Zifan, Deep Shaleen, Fariha Anna et al. Rapidash: Efficient Constraint Discovery via Rapid Verification. — 2023. — 2309.12436.

[3] Song Shaoxu, Chen Lei. Differential dependencies: Reasoning and discovery // ACM Trans. Database Syst. — 2011. — Aug.. — Vol. 36, no. 3. — 41 p. — URL: https://doi.org/10.1145/2000824.2000826.

[4] Примитивы профилирования данных в Desbordante : Rep. / Saint Petersburg State University ; Executor: Даниил Гончаров : 2024. — https://github.com/Desbordante/desbordante-core/blob/main/docs/papers/Data%20profiling%20survey%20-%20Daniil%20Goncharov%20-%202024%20spring.pdf.

[5] Реализация алгоритма поиска дифференциальных зависимостей в рамках платформы Desbordante : Rep. / Saint Petersburg State University ; Executor: Михаил Синельников : 2024. — https://github.com/Desbordante/desbordante-core/blob/main/docs/papers/SPLIT%20-%20Michael%20Sinelnikov%20-%202023%20autumn.pdf.