Высокопроизводительный поиск функциональных зависимостей в данных

Отчёт по учебной практике

Салью Артур Кристофович
группа 19.Б11-мм

Научный руководитель:
асс. кафедры ИАС Чернышев Г.А.

Введение

Представим привычную нам таблицу с данными.

Пусть два её столбца имеют имена A и B соответственно.

Если каждому элементу столбца A *однозначно* сопоставлен элемент из B,

тогда говорят, что соблюдается **функциональная зависимость**

$$FD: A \longrightarrow B$$

Пример функциональных зависимостей

Отношение R			
Имя	Фамилия	Марка автомобиля	Цвет автомобиля
Александр	Петров	Desbordauto	Navy blue
Андрей	Серый	Desbordauto	Snakeskin Green
Василиса	Шавел	Fdepus	Ghost White
Дмитрий	Петров	Desbordauto	Navy blue
Василиса	Шавел	FdepuZ	Ghost White

Алгоритмы поиска

- Поиском ФЗ занимаются достаточно давно;
- ФЗ применяются при нормализации БД, поиске ошибок, дубликатов, анализе данных;
- Алгоритмы поиска ресурсоёмки.

Существующее решение

Metanome — первая платформа с алгоритмами поиска.

Metanome написан на Java:

- необходимость тонкой настройки на уровне JVM;
- затруднён анализ производительности;
- высокие затраты памяти (по сравнению с C++).

Решение — реализуем алгоритм на C++ для проекта Desbordante

Цели и задачи

Цель работы — реализовать стартовую версию алгоритма поиска функциональных зависимостей (FDep) на языке программирования C++ для платформы Desbordante и измерить её производительность.

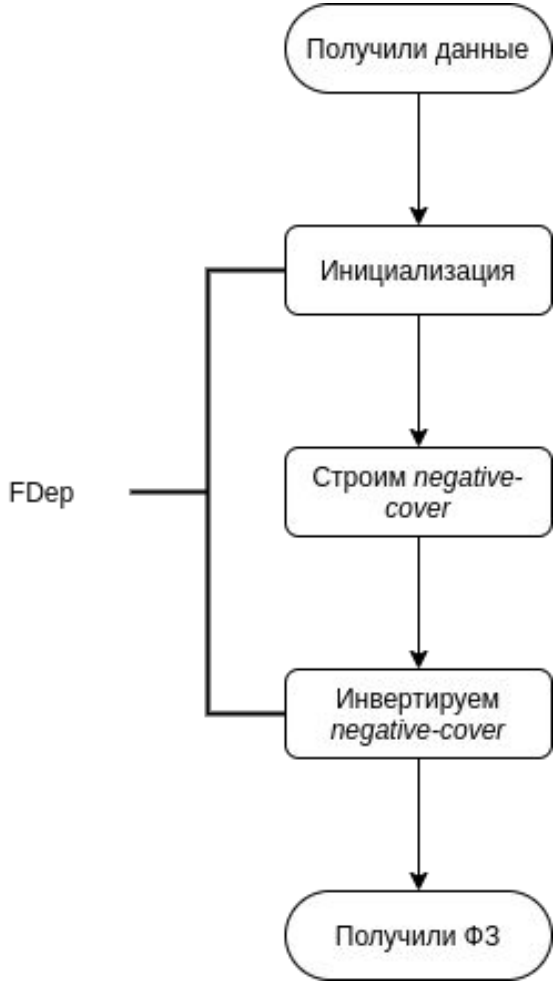
Задачи:

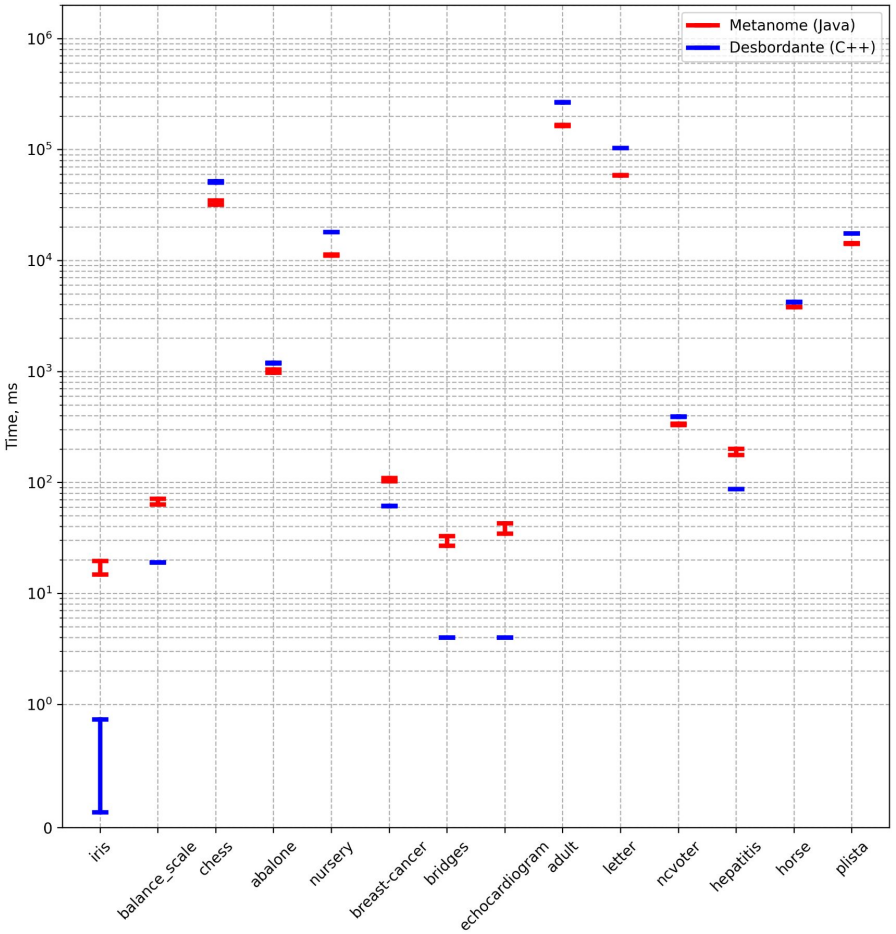
- обзор предметной области;
- идейный обзор алгоритма FDep;
- реализация алгоритма FDep на C++;
- сравнительный анализ производительности алгоритма.

Алгоритм FDep

- Индуктивный алгоритм “от общего к частному”;
- *bottom-up* подход показывает высокую эффективность;
- основная структура — дерево зависимостей.

Блок-схема работы FDep





Сравнение с Metanome

3 датасета - выигрыш (в 1.5 раза)

4 датасета - приблизительно одинаково

5 датасетов - проигрыш (в 1.5 раза)

Заключение

Проведен обзор предметной области:

- обзор области применения функциональных зависимостей;
- обзор алгоритма FDep.

А так же

- реализован алгоритм FDep на языке C++;
- проведен анализ производительности алгоритма.

Дополнительно (пояснение работы FDep)

A	B	C	D
0	0	0	0
1	1	0	0
0	2	0	2
1	2	3	4

Получение NCOVER:
t1, t2: CD→A, CD→B
t1, t3: AC→B, AC→D

Если продолжить, получим множество ВСЕХ нарушенных ФЗ. Оставим с минимальной LHS.
NCOVER: {B→A, CD→A, AC→B, CD→B, A→C, B→C, AC→D, B→D}

Преобразование к positive-cover

На предыдущем шаге получено NCOVER = {B→A, CD→A, AC→B, CD→B, A→C, B→C, AC→D, B→D}
DEPS = {Ø→A, Ø→B, Ø→C, Ø→D} - начальное множество наиболее общих зависимостей.

Во внешнем цикле рассматриваем ND: B→A.
Из каких зависимостей в DEPS следует ND? Ответ - {Ø→A}
Для каждой зависимости D в этом множестве делаем следующее:
Строим spec(D, ND) = {C→A, D→A} - специализации, из которых не следует ND.
Далее DEPS = (DEPS / **D**) ∪ spec(**D**, ND) = {C→A, D→A, Ø→B, Ø→C, Ø→D}

Теперь рассматриваем CD→A. Эта ФЗ следует из {C→A, D→A}
spec(C→A, CD→A) = {BC→A}
spec(D→A, CD→A) = {BD→A}
DEPS = {BC→A, BD→A, Ø→B, Ø→C, Ø→D} И так далее… Получим ответ:

POS_COV = {BC→A, BD→A, AD→B, AB→C, D→C, AB→D, BC→D}