Санкт-Петербургский государственный университет

Кафедра Информационно Аналитических Систем

Группа 21.Б08-мм

Приближённые зависимости в базах данных

Фофанов Максим Александрович

Отчёт по учебной практике
в форме «Теоретическое исследование»

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

Санкт-Петербург
2024

Оглавление

Введение

В современном мире, где данные стали одним из ключевых активов во всех сферах жизни, понимание зависимостей содержащихся в данных приобретает особую значимость. Зависимости в базах данных не просто отражают связи между различными атрибутами, но и являются фундаментальными инструментами для обеспечения целостности данных, оптимизации запросов и эффективного проектирования схем баз данных.

Однако в реальном мире данные могут содержать пропуски, ошибки и неточности. Поэтому научным сообществом были предложены приближённые зависимости — версии обычных зависимостей, таких как например функциональные зависимости или зависимости включения, но выполняющиеся только для какой-то доли записей. Для разных типов зависимостей используются разные метрики, а для разных задач уместно выбрать разные их пороговые значения.

Таким образом приближенные зависимости позволяют аналитикам и разработчикам баз данных гибко работать с неполными или неточными данными без их предварительной очистки или обработки, что позволяет сохранить большее количество точной информации. В условиях, когда абсолютная точность данных недостижима, приближенные зависимости предлагают практическое решение для выявления скрытых закономерностей и взаимосвязей в данных, которые было бы невозможно обнаружить с помощью «точных» версий зависимостей.

В данной работе мы делаем попытку собрать и структурировать имеющуюся в открытом доступе информацию о наиболее популярных метриках приближённости в базах данных, связанных с ними приближённых зависимостей и алгоритмах их майнинга.

1. Постановка задачи

Цель данной работы — предоставить обзор современного состояния научных исследований в области приближенных зависимостей. Для её выполнения были поставлены следующие задачи:

1. Составить обзор существующих метрик приближённости зависимостей.
2. Для каждой метрики привести список основных зависимостей, в контексте которых она используется.
3. Для каждой зависимости привести её определение и описать основные алгоритмы.
4. Для всех метрик и зависимостей составить примеры наглядно демонстрирующие их работу.

2. Обзор

В мире приближённых зависимостей существуют три основные метрики приближённости: g_1, g_2, g_3 [6]. Не смотря на то, что в оригинальной статье приводится авторы приводят в качестве примера приближённые функциональные зависимости, концепция легко обобщается благодаря гибкости приведённых определений.

Пусть дано некоторое отношение r со схемой R и функциональная зависимость $X \to Y$ над R. Тогда мы утверждаем, что пара (u, v) кортежей из r **нарушает зависимость** или, что тоже самое, является **нарушающей парой**, если $u[X] = v[X]$, но $u[Y] \neq v[Y]$. Отсюда можно сделать вывод, что зависимость выполняется, на отношении, если отношение не содержит нарушающих пар. Кортеж u называется **нарушающим**, если если он входит в нарушающую пару.

2.1. Метрики g_1 и g_2

Определим G_1 как количество нарушающих пар для зависимости $X \to Y$ на отношении r. Тогда метрика g_1 представляет из себя нормированную версию G_1.

$$G_1(X \to Y, r) = |\{(u, v) | u, v \in r, u[X] = v[X] \wedge u[Y] \neq v[Y]\}|$$

$$g_1(X \to Y, r) = G_1(X \to Y, r)/|r|^2$$

G_2 — это количество нарушающих кортежей для зависимости $X \to Y$ на отношении r. Метрика g_2, в свою очередь представляет из себя нормированную версию G_2.

$$G_2(X \to Y, r) = |\{u | u \in r, \exists v \in R : u[X] = v[X] \wedge u[Y] \neq v[Y]\}|$$

$$g_2(X \to Y, r) = G_2(X \to Y, r)/|r|$$

Метрики g_1 и g_2, как уже было показано выше, применяются при

Таблица 1: Таблица для примера расчёта g_1, g_2, g_3

X	Y
a	1
b	2
a	3
c	3
d	4

поиске приближённых функциональных зависимостей. Однако в силу худшей обобщаемости и большей сложности вычисления они не так распространены как g_3 [4].

Для примера приведённого в Таблице 1 значением метрики g_1 рассчитывается следующим образом: нарушающий кортеж всего один — это $((a, 1), (a, 3))$, поэтому:

$$g_1(X \to Y, r) = \frac{1}{5^2} = 0.04,$$

а для g_2 есть два значения с разными правыми частями: $(a, 1)$ и $(a, 3)$, поэтому значением метрики g_2 будет:

$$g_2(X \to Y, r) = \frac{2}{5} = 0.4.$$

2.1.1. Приближённые функциональные зависимости

Функциональная зависимость над отношением r со схемой R — это выражение $X \to A$, где $X \subseteq R$ и $A \in R$. Зависимость **выполняется**, если для всех пар кортежей $t, u \in r$ выполняется: если $\forall B \in X\ t[B] = u[B]$, то $t[A] = u[A]$ или, что тоже самое, t и u **согласуются** по X и A [10].

Приближённая функциональная зависимость — это функциональная зависимость, которая почти выполняется. Пример: связь между именем и полом. Существует множество способов определить степень приближённости зависимости, один из них — это метрика g_3. Она используется в одном из двух основных алгоритмах для поиска приближённых функциональных зависимостей: PYRO [7].

Таблица 2: Пример AFD: Должность → Зарплата

ID	Должность	Запрлата
1	Программист	3000
2	Дизайнер	2800
3	Программист	3200
4	Менеджер	3000
5	Дизайнер	2800
6	Программист	3000

В алгоритме PYRO используется адаптация метрики g_1, которую авторы статьи называют e:

$$e(X \to A, r) = \frac{|\{(t_1, t_2) \in r^2 | t_1[X] = t_2[X] \wedge t_1[A] \neq t_2[A]\}|}{|r|^2 - |r|}.$$

Алгоритм опережает TANE (описан в разделе 2.2.1) по скорости благодаря использованию PLI Cache (в дальнейшем просто PLI). Пускай r — это отношение со схемой R и $X \subseteq R$ — это набор атрибутов. Кластер — это множество всех индексов кортежей из r, у которых одинаковые значения для X или $c(t) = \{i | t_i[X] = t[X]\}$. PLI для X — это все такие множества, кроме кластеров-синглтонов:

$$\bar{\pi}(X) = \{c(t) | t \in r \wedge |c(t)| > 1\}.$$

Размером получившегося индекса назовём $||\bar{\pi}(X)|| \sum_{c \in \bar{\pi}(X)} |c|$.

Тогда расчёт метрики приближённости e выглядит следующим образом: пары кортежей которые согласуются по X и не согласуются по A (для кандидата $X \to A$) являются нарушающими парами, которые необходимо подсчитать. Но вместо того, чтобы считать их напрямую, в PYRO используется более оптимальный метод. Для каждого кластера $\bar{\pi}(X)$ рассчитывается количество пар кортежей, которые также согласуются по A, а затем получившийся результат вычитается из общего количества пар кортежей в кластере. Это делается с помощью v_A — вектора, в котором в формате one-hot-encoding записана информация о содержимом кластера. Суммируя ошибки на каждом кластере мы

получаем финальную ошибку.

Листинг 1: Расчёт e для AFD с помощью PYRO

```
e ← 0
foreach c ∈ π̄(X) do
    counter ← dictionary with default value 0
    foreach i ∈ c do
        if v_A[i] ≠ 0 then
            increase counter[v_A[i]]
        end if
    end foreach
    e ← e + |c|² − |c| − Σ_{A∈counter} c_A² − c_A
end foreach
return e
```

2.1.2. Приближённые уникальные зависимости колонок

AUCC — ещё один тип приближённой зависимости который можно находить с помощью PYRO. Уникальная зависимость колонок — это [7], для отношения r со схемой R с наборами атрибутов $X, Y \subseteq R$, мы говорим, что X — это UCC, если для всех пар кортежей $t_1, t_2 \in r$ из того, что $t_1[X] \neq t_2[X]$ следует $t_1[Y] = t_2[Y]$.

Определим метрику приближённости для AUCC:

$$e(X \to A, r) = \frac{|\{(t_1, t_2) \in r^2 | t_1[X] \neq t_2[X] \wedge t_1[A] = t_2[A]\}|}{|r|^2 - |r|}.$$

Для приближённой UCC, в отличие от AFD, расчёт ошибки для $\bar{\pi}(X)$ тривиален. Всё дело в том, что все пары кортежей хранящиеся внутри каждого кластера — это и есть нарушающие кортежи.

Листинг 2: Расчёт e для AUCC с помощью PYRO

```
return Σ_{c∈π̄(X)} (|c|²−|c|)/(|r|²−|r|)
```

Приближённый уникальные колоночные зависимости используются

для таких задач, как очистка данных, нормализация баз данных и оптимизация запросов.

2.2. Метрика g_3

G_3, в свою очередь — это количество количество кортежей для зависимости $X \to Y$ на отношении r, которые нужно удалить, чтобы получить точную зависимость.

$$G_3(X \to Y, r) = |r| - max\{|s| s \subset r, s \models X \to Y\}$$
$$g_3(X \to Y, r) = G_3(X \to Y, r)/|r|$$

Метрика g_3 является стандартом индустрии и используется в контексте множества разных приближённых зависимостей: приближённых функциональных зависимостей, приближённых зависимостей включения, вероятностных функциональных зависимостей, численных зависимостей.

Для примера определённого выше:

$$g_3(X \to Y, r) = 5 - \frac{4}{5},$$

так как достаточно выкинуть один кортеж и «точная» зависимость будет выполняться.

2.2.1. Приближённые функциональные зависимости

Не смотря на то, что в более современном алгоритме PYRO для поиска приближённых функциональных зависимостей используется метрика g_1, первый алгоритм для расчёта AFD использует g_3. Алгоритм TANE [10], предназначенный для майнинга точных функциональных зависимостей может также быть модифицирован для майнинга приближённых функциональных зависимостей. Для этого в процедуре отбрасывания неподходящих кандидатов-зависимостей применяется метрика аналогичная метрике g_3 метрика, которую авторы статьи называют e:

$$e(X \to A) = min\{\frac{|s|}{|r|} : s \subset r \; and \; X \to A \; holds \; in \; r \setminus s\}.$$

Ещё один алгоритм [1] использующий метрику g_3, называется $DiM\varepsilon$. Это высоко-оптимизированный алгоритм использует уровневый подход в генерации кандидатов, начиная с множеств-синглтонов на нулевом уровне. Также в статье авторы утверждают, что алгоритм может быть адаптирован для работы с некоторыми другими метриками и даже зависимостями.

Функциональные и приближённые функциональные зависимости используются для нормализации баз данных, очистки данных, а также помогают аналитикам обнаружить скрытые тренды в данных.

2.2.2. Приближённые зависимости включения

Зависимость включения [7] над схемой R — это утверждение вида $R_i[X] \subseteq R_j[Y]$, где $R_i, R_j \in R$, $X \subseteq R_i$, $Y \subseteq R_j$. Размером (арностью) такой зависимости называют $i = R[X] \subseteq R[Y]$, где $|i| = |X| = |Y|$. Зависимости включения размера один принято называть унарными.

Зависимость включения выполняется, если все значения из левой части присутствуют в правой части. Чтобы оценить степень приближённости используется аналог метрики g_3 — g_3', версия адаптированная под зависимости включения и несущая абсолютно тот же смысл.

Несмотря на то, что не существует отдельных алгоритмов для выявления приближённых зависимостей включения, многие популярные алгоритмы для поиска «точных» были адаптированы для использования в этой задаче, например MIND [8].

В MIND используется уровневый подход, где из уже обнаруженных зависимостей размера i генерируются кандидаты размера $i + 1$. В случае с приближёнными зависимостями на стадии валидации кандидатов в расчёт идёт также и неточные зависимости которые соответствуют заданному пользователем порогу для g_3'.

Основная область применения как «точных», так и приближённых зависимостей включения — поиск внешних ключей в базах данных [2].

Таблица 3: Пример AIND: Email пользователя → Зарегистрированный email ($\varepsilon = 0.34$)

TID	Email пользователя	ID	Зарегистрированный email
T001	example@email.com	C123	example@email.com
T002	sample@email.com	C124	sample@email.com
T003	missing@email.com	C125	-

2.2.3. Вероятностные функциональные зависимости

Пусть R — это отношение [5], $\bar{X}$ набор атрибутов в R и A — атрибут из R. Вероятностная функциональная зависимость имеет нотацию $\bar{X} \to^p A$, где p — это вероятность того, что $\bar{X} \to A$ выполняется.

Существует два способа расчёта вероятности $\bar{X} \to A$, если данные поступают из единственного источника R. В идеальном мире, если $\bar{X} \to A$ выполняется, то все кортежи с одинаковыми значениями в $\bar{X}$ должны иметь одинаковые значения в A. Но так как реальные данные скорее всего будут зашумлены, то в них могут присутствовать кортежи, значение в A у которых отличается от самого популярного значения для этого набора значений из $\bar{X}$.

Первый способ посчитать вероятность того, что $\bar{X} \to A$ выполняется — это сперва посчитать долю таких кортежей для всех уникальных значений из $\bar{X}$, а затем на основе этих данных посчитать вероятность. Этот подход авторы статьи называют $PerValue$.

$$P_r(\bar{X} \to A, R)_{PerValue} = frac \sum_{V_x \in D_{\bar{X}}} P_r(\bar{X} \to A, V_X) D_{\bar{X}}$$

Второй подход — это адаптированная версия g_3, которую авторы называют $PerTuple$:

$$P_r(\bar{X} \to A, R)_{PerTuple} = \frac{\sum_{V_x \in D_{\bar{X}}} |V_A, V_X|}{\sum_{V_x \in D_{\bar{X}}} |V_X|}.$$

Здесь $D_{\bar{X}}$ — это все различные значения из $\bar{X}$ в R и $|D_{\bar{X}}|$ — это

Таблица 4: Пример pFD: ID → Имя, Департамент

ID	Имя	Департамент
1	Алиса	HR
2	Боб	IT
3	Чарли	Финансы
1	Алиса	Финансы
4	Дана	Маркетинг

размер $D_{\bar{X}}$, а $P_r(\bar{X} \to A, V_X)$ для обоих случаев определяется как:

$$P_r(\bar{X} \to A, V_X) = \frac{|V_A, V_X|}{|V_X|}$$

где $|V_A, V_X|$ — это количество кортежей со значениями V_X для $\bar{X}$ и V_A для A, а $|V_X|$ — количество кортежей со значением V_X в $\bar{X}$ и не нулевым (non-null) значением для A.

Вероятностные функциональные зависимости, как и обычные функциональные зависимости используются для очистки данных и для нормализации баз данных.

2.3. Другое

2.3.1. Запрещающие ограничения

Запрещающие ограничения (Denial constraints, DC) — это тип ограничений целостности, используемых в базах данных для обеспечения качества данных. Это правила, которые указывают условия, которые не должны возникать в базе данных. Например, DC может указывать, что две строки в таблице не должны иметь определенных комбинаций значений. Если вставка или обновление строки нарушает DC, действие обычно отклоняется.

Приблизительное отрицательное ограничение в базах данных — это тип ограничения, который допускает некоторую гибкость или исключения. В отличие от точных отрицательных ограничений, которые строго запрещают определенные комбинации значений данных, приблизитель-

ные отрицательные ограничения терпят ограниченное количество нарушений.

Для вычисления меры ошибки используется [9] стандартная метрика g_1 описанная выше.

DCs и их приближённая версия критически важны для поддержания согласованности и надежности данных в базе данных, так как они предотвращают ввод недопустимой или противоречивой информации.

2.3.2. Графовые сущностные зависимости

Графовая сущностная зависимость (GED) - это ограничение в графе свойств G, выраженное как пара $\varphi = (Q[\bar{u}], X \to Y)$. Она указывает, что для любого экземпляра паттерна в графе $Q[\bar{u}]$ в G должна соблюдаться зависимость $X \to Y$. Это означает, что если в экземпляре паттерна выполняются определенные условия, заданные X, то должны выполняться и другие условия, заданные Y. В качестве метрики приближённости для GED используется [3] g_3 в оригинальном виде.

Графовые сущностные зависимости (GED) используются для нескольких ключевых целей в области графовых баз данных и управления данными. Они обеспечивают целостность и согласованность данных и помогают оптимизировать сложные запросы.

3. Итоги обзора

Метрики из семейства g_1, g_2, g_3 активно применяются и адаптируются научным сообществом под новые типы приближённых зависимостей. Метрика g_3, благодаря своей хорошей адаптируемости и простоте расчёта нашла себе применение в таких доменах как AIND и PFD, в то время как вариации g_1 были разработаны для ускорения расчёта приближённых функциональных зависимостей и AUCC. Общее состояние дел в области приближенных зависимостей и используемых метрик приведено в Таблице 5.

Кроме того, метрики нашли применение и для менее «классических» объектов, таких как запрещающие ограничения и графовые сущностные зависимости.

Таблица 5: Использование метрик из семейства g

	AFD	**AUCC**	**AINDS**	**PFD**	**DC**	**GED**
g_1	✓	✓	✗	✗	✓	✗
g_2	✓	✗	✗	✗	✗	✗
g_3	✓	✗	✓	✓	✗	✓

Заключение

Результаты:

- Составлен обзор метрик приближённости из семейства g.
- Для каждой метрики обозначен список основных зависимостей, в контексте которых она используется.
- Для каждой зависимости было приведено определение и описаны основные алгоритмы.
- Обзор был снабжён примерами расчёта метрик и зависимостей.

Список литературы

[1] Caruccio Loredana, Deufemia Vincenzo, Polese Giuseppe. Mining relaxed functional dependencies from data // Data Mining and Knowledge Discovery. — 2020. — Vol. 34, no. 2. — P. 443–477.

[2] De Marchi Fabien, Lopes Stéphane, Petit Jean-Marc. Efficient Algorithms for Mining Inclusion Dependencies // Advances in Database Technology — EDBT 2002 / Ed. by Christian S. Jensen, Simonas Šaltenis, Keith G. Jeffery et al. — Berlin, Heidelberg : Springer Berlin Heidelberg, 2002. — P. 464–476.

[3] FASTAGEDS: fast approximate graph entity dependency discovery / Guangtong Zhou, Selasi Kwashie, Yidi Zhang et al. // International Conference on Web Information Systems Engineering / Springer. — 2023. — P. 451–465.

[4] Vilmin Simon, Faure-Giovagnoli Pierre, Petit Jean-Marc, Scuturici Vasile-Marian. Functional Dependencies with Predicates: What Makes the g_3-error Easy to Compute? — 2023. — 2306.09006.

[5] Functional Dependency Generation and Applications in Pay-As-You-Go Data Integration Systems / Daisy Zhe Wang, Xin Luna Dong, Anish Das Sarma et al. // 12th International Workshop on the Web and Databases, WebDB 2009, Providence, Rhode Island, USA, June 28, 2009. — 2009. — URL: http://webdb09.cse.buffalo.edu/papers/Paper18/webdb09.pdf.

[6] Kivinen Jyrki, Mannila Heikki. Approximate inference of functional dependencies from relations // Theoretical Computer Science. — 1995. — Vol. 149, no. 1. — P. 129–149. — Fourth International Conference on Database Theory (ICDT '92). URL: https://www.sciencedirect.com/science/article/pii/030439759500028U.

[7] 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.

[8] Marchi Fabien De, Lopes Stéphane, Petit Jean-Marc. Unary and n-ary inclusion dependency discovery in relational databases // Journal of Intelligent Information Systems. — 2009. — Vol. 32. — P. 53–73.

[9] Pena Eduardo HM, De Almeida Eduardo C, Naumann Felix. Discovery of approximate (and exact) denial constraints // Proceedings of the VLDB Endowment. — 2019. — Vol. 13, no. 3. — P. 266–278.

[10] Tane: An Efficient Algorithm for Discovering Functional and Approximate Dependencies / Yká Huhtala, Juha Kärkkäinen, Pasi Porkka, Hannu Toivonen // The Computer Journal. — 1999. — Vol. 42, no. 2. — P. 100–111.