Федеральное государственное автономное образовательное учреждение высшего образования Национальный исследовательский университет
«Высшая школа экономики»

Факультет компьютерных наук
Основная образовательная программа
Прикладная математика и информатика

Курсовая работа на тему

Дифференциальная приватность в ClickHouse

Выполнил студент группы БПМИ175, 3 курса,
Вишняков Артемий Юрьевич
Научный руководитель:
Доцент, кандидат технических наук,
Сухорослов Олег Викторович
Консультант:
Руководитель группы разработки ClickHouse,
Миловидов Алексей Николаевич Москва, 2019

Москва, 2020

Содержание

Аннотация

Дифференциальная приватность позволяет выполнять статистический анализ, не разглашая исходную информацию о каждом индивиде. Однако на данный момент СУБД ClickHouse, оптимизированная под аналитику, лишена этой функциональности. В работе рассматриваются добавленные агрегатные функции, вычисляющающиеся дифференциально приватно с использованием шума Лапласа, отсечение выбросов, и другие методики ограничения информации, рекомендации по составлению дифференциально приватных запросов, а также специальный инструмент для перевода запроса в дифференциально приватную форму.

Annotation

Differential privacy allows statistical analysis, without disclosing private information. At the moment, ClickHouse, optimized for analytical use, lacks this feature. In this project I introduce 6 new aggregate functions utilizing Laplace noise, deviations clipping and other methods of information constrain. Also I include recommendations on creating differential private queries and tool for converting queries to differentially private form.

Ключевые слова: Базы данных, Дифференциальная приватность, SQL

1. Введение

С первого взгляда может показаться что агрегатные функции и так достаточно скрывают исходные данные. Однако, к примеру, если нам доступна средняя оценка фильма (или результат любого другого тайного голосования), а также количество голосов в реальном времени, то, зная точное время выставления кем-то оценки, мы можем узнать его мнение. А если аналитик (здесь и далее аналитиками я называю тех, кто составляет запросы к базе

данных, а те объекты базы данных, информацию о которых мы защищаем, называю пользователями) может задавать произвольные запросы, то при помощи частичных сумм он способен получить базу данных целиком. Таким образом, целью работы является внедрение в ClickHouse механизмов расчёта агрегатных функций, которые бы как можно меньше разглашали информацию об индивидуальном пользователе. Рассмотрим один из способов формализации этого концепта.

Запрос F назовём ε-Дифференциально Приватным *(ε-ДП)*[1], если для любых 2 баз данных *D1* и *D2* , различающихся только наличием в одной из них дополнительного пользователя, и подмножества *S* области значений

$$\mathbf{P[F(D_1) \in S] < P[F(D_2) \in S] \cdot e^{\varepsilon}}$$

Что это означает? Допустим, аналитик желает проверить некоторое предположение *A* про конкретного пользователя. Для этого он составляет запрос F и подмножество возможных результатов выполнения запроса *S*, таких, что он считает предположение верным тогда, и только тогда, когда результат попадает в *S*. Условие приватности можно представить как $P[A|S] \cdot e^{-\varepsilon} < P[!A|S] < P[A|S] \cdot e^{\varepsilon}$, что ограничивает информацию об истинности A, которую можно получить из одного запроса. При этом N применений *ε-ДП* запросов являются *(N·ε)-ДП* запросом. (Можно получить, возведя приведёное неравенство в степень, что соответствует наиболее раскрывающему информацию случаю получения всех одинаковых ответов)[9]

Для соблюдения условий дифференциальной приватности к точному вычисленному результату агрегатной функции прибавляется шум. Для создания шума используется механизм Лапласа (случайная величина с плотностью распределения $exp(-|x|/\varepsilon)\cdot C$), как создающий минимальную ошибку при фиксированном желаемом значении ε. Как и все подобные механизмы,

механизм Лапласа использует параметр *C* - “чувствительность”, ограничивающий максимальное значение, на которое может измениться ответ при добавлении ещё одного пользователя в базу. Ключевая задача - вычисление правильного коэффициента шума.

Задачами проекта являются внесение в ClickHouse функций, удовлетворяющих условиям дифференциальной приватности, а также разработка вспомогательных инструментов, так, чтобы покрыть все распространённые сценарии агрегации данных.

Можно выделить 2 “уровня” функциональности в базе данных:

1) База позволяет благонадёжному квалифицированному аналитику создать ДП запрос. Под благонадёжностью подразумевается, что у аналитика будет возможность создать любой запрос, но аналитик не намерен получать и разглашать приватные данные, разглашаться будет только результат выполнения запроса. Квалифицированность означает, что аналитик способен сделать запрос дифференциально приватным. Такой уровень применим для демонстрации метрик, но данные, получаемые другими наблюдателями, ограничены уже созданными запросами. Решение этой задачи в виде добавления дополнительных аггрегатных функций в СУБД ClickHouse будет рассмотрено в 1 главе.

2) База позволяет квалифицированному аналитику составить запрос, который будет проверен на дифференциальную приватность, и, в случае соответствия, выполнен. Такой уровень существенно ограничивает множество выполнимых запросов, однако не требует доверия аналитику. Эта задача решена посредством введения преобразователя запросов и будет рассмотрена во 2 главе.

2. Обзор альтернативных решений

2.1. Альтернативный способ формализации

Для решения задачи разглашения статистических данных без разглашения приватной информации также может быть использованна k-анонимность. Это означает, что любая разглашённая информация, относящаяся к пользователю, не может быть однозначно ему сопоставлена, а должна потенциально быть сопоставима к ещё не менее k - 1 другим пользователям.[8]
Однако данный подход имеет несколько серьёзных недостатков:

INSERT

В результате использовались подходы, базирующиеся на дифференциальной приватности.

2.2. Теоретические алгоритмы

Предоставляются алгоритмы, обрабатывающие поступающие SQL запросы. При этом в некоторых случаях алгоритм не может дифференциально приватно исполнить запрос (например, если запрос включает слияние, приводящее к соединению строк таблиц, принадлежащих разным пользователям). На данный момент опубликовано 2 алгоритма: работники Google описали систему, основывающуюся на классической дифференциальной приватности, а компания Uber предложила использовать эластичную дифференциальную приватность.

2.2.1. Предложение Google

Составитель запроса помечает некоторый агрегационный оператор как дифференциально приватный. Общий цикл обработки запроса можно разделить на три части:

Обработка данных до помеченного оператора:

На этом этапе необходимо, получить список строк, у каждой из которых известен ровно один пользователь владелец. Других ограничений нет, поэтому допустимы все операции, не нарушающие единоличное владение каждой строкой.

Агрегационный оператор:

Проблему принадлежности множества записей одному пользователю предлагается решать выбором (с повторениями) для пользователя K случайных записей, где K - конфигурируемый параметр. После этого все K записей агрегируются функцией. Получается таблица, в которой существует не более одной записи на каждого пользователя.[2]

Для решения проблемы чувствительности предлагается использовать *ограничение чувствительности*. Необходимо выбрать границы для значений. Значения, выходящие за выбранные границы, приравниваются к границам. Теперь можно по границам определить необходимый коэффициент шума. Но если выбрать границы слишком широкими, то амплитуда шума будет больше, чем необходимо, что уменьшает точность.

Если выбрать границы слишком узкими, то данные будут потеряны во время ограничения значений. Поэтому, если границы не указаны самим составителем запроса, они вычисляются по имеющимся данным, однако тогда значения границ тоже содержат приватные данные.

В результате половина доступной приватности тратится на определение границ, а другая на финальную агрегацию (это означает, что все шумы удваиваются, что эквивалентно совершению запросов, но в *ε/2-ДП* систему). Для определения границ строится логарифмическая гистрограма, ко всем столбцам, содержащим ровно 1 элемент добавляется шум, максимальный оставшийся положительным с учётом шума столбец определяет границы.

После агрегации:

Данные можно обрабатывать произвольным образом (но не смешивать с данными, не прошедшими агрегацию) и вернуть составителю запроса.

Преимущества такого подхода:

- Границы значений определяются автоматически
- Требования к запросам просты и понятны
- Шум распределяется по пользователям пропорционально их вкладу

Недостатки:

- Необходима модификация программного обеспечения базы данных
- Составитель запроса должен правильно указать, какая из агрегаций будет дифференциально приватной

2.2.2. Предложение Uber

Главной целью исследователей было составить алгоритм, который можно было бы максимально просто встроить в любую инфраструктуру, поэтому на вход принимается тот же самый запрос, который аналитик бы послал в стандартную базу данных, после преобразования алгоритм должен создать запрос, который выполняется над любой SQL совместимой базой данных. При этом считается, что для всех значений известны границы, и эти границы являются публичной информацией.[3]

Для каждого пользователя учитывается частота использования соответствующих ему строк при обработке запросов, максимум этой частоты включается множителем в конечный коэффициент шума. К сожалению, такой алгоритм уязвим к фабрикованным запросам, так как частоты проникают в амплитуду шума.

Алгоритм рекурсивно преобразует запрос (обходит тем же способом, что и при выполнении), собирая данные о требуемом “бюджете приватности”, и строит коэффициент шума, прибавляемого к финальному результату. Алгоритм может обрабатывать все операторы, кроме некоторых вариантов слияния. Такой подход авторы назвали эластичной приватностью. Авторы заявили, что только 6% используемых аналитиками Uber запросов были отклонены, при этом никаких усилий по подстройке запросов под нужды алгоритма не производилось.

Преимущества такого подхода:

- Возможность встроить в любую инфраструктуру без изменения СУБД и обучения аналитиков

Недостатки:

- Добавляемый шум увеличен за счёт отсутствия балансировки между пользователями с большим и маленьким количеством записей
- Необходимо зафиксировать границы величин
- Если запрос отвергается алгоритмом, для его подстройки необходимо разбираться в сложной системе

2.3. Практические реализации

Как и с теоретическими подходами, представленные решения обозначаются именами компаний, которые их опубликовали. На данный момент нет реализации данного функционала для ClickHouse.

2.3.1. Uber

Реализация алгоритма преобразования на Scala выложена в общий доступ. На момент написания статьи репозиторий помечен устаревшим и не поддерживается. Реализованы только базовые операторы. Поддержки пользователей, владеющих несколькими строками, нет. [4] Это решение можно адаптировать под ClickHouse в виде преобразователя запросов. Однако недостатки использованного подхода и отказ самих разработчиков от использования этого решения привели к решению не использовать его.

2.3.2. Google

Доступна реализация алгоритма определения границ и наложения шума, однако API значительно ограничен. Доступны функции sum, min, max, avg, stddev. Добавить поддержку пользователей не представляется возможным. Для определения границ используется ограниченная чувствительность. [5] Поскольку это решение использует тот же язык программирования, что и ClickHouse, планировалось встроить эту библиотеку. Однако из-за различий структуры кода, используемого в 2 проектах и ограниченности применимости встраивание не целесообразно.

2.3.3. IBM

Реализация на Python, использует некоторые наработки эластичной дифференциальной приватности. Не имеет поддержки пользователей, доступны 12 агрегационных функций. Скорость работы на 30% ниже, чем у реализации от Google. [6]

3. Глава 1. Устройство агрегатных функций

Функции имеют те же названия, что и их не дифференциально приватные аналоги, но с префиксом *anon_*, а также имеют дополнительный параметр ε.

Агрегатные функции не учитывают пользователей (они действуют из предположения, что каждому пользователю соответствует не более 1 строки).

Каждая функция подсчитывает данные, необходимые для вычисления истинного значения, а также логарифмическую гистограмму, необходимую для оценки величин значений.

Исключением является функция *anon_count*. Поскольку каждая запись может увеличить её только на 1, шум всегда добавляется с коэффициентом 1, после чего происходит округление, в случае отрицательного ответа возвращается 0.

Необходимость гистограммы обусловлена тем, что сами границы данных тоже являются приватной информацией. К примеру, если у одного пользователя значение суммируемой характеристики миллиард, а у остальных не больше 1. Тогда большие по модулю значения на радикально вероятнее при условии прохождения пользователем фильтра, заданного аналитиком (потому что коэффициент шума будет огромен), а при непрохождении скорее всего ответ будет маленьким, и это поведение не зависит от ε. В случае вероятностного характера выбора границ, есть шанс получить большой шум при маленьких входных данных. С одной стороны, это уничтожит информацию о сумме, с другой стороны получение большого числа не будет указывать на присутствие пользователя, нарушая ε-дифференциальную приватность.

Теперь рассмотрим принцип работы гистограммы.

Гистограммы содержат 64 интервала .

Введём основание, как корень 64 степени из максимального представимого типом столбца значения. Тогда интервал i имеет границы **[основание^{i}; основание$^{i+1}$)**, для i от 1 до 63, 0 интервал имеет границы [0, 1), число учитывается в интервале, включающем его модуль.

Для вычисления границ ко всем интервалам добавляется шум с коэффициентом 1 и в 2 раза уменьшенным ε, после этого берётся максимум из верхних границ интервалов, значение гистограммы в которых превосходит порог.

Порог вычисляется как

$$\textbf{порог} = \log(1 - P^{1/63}) / \varepsilon$$

Где *P* - вероятность не ложноположительного срабатывания, то есть вероятность, что будет выбран тот интервал, в которое попало хотя бы одно число из исходных. При увеличении *P* повышается возможный шум, при уменьшении возникает вероятность, что многие значения выйдут за границы. Рекомендуемое значение *P = 1 - 1E-9*

Функции суммы, среднего и стандартного отклонения используют эти диаграммы. Также можно явно указать границы, в этом случае значения, выходящие за границы будут учитываться как значения соответствующих границ (clip), а шум будет меньше, так как не требуется тратить приватность на вычисление границ. Важное следствие использования гистограмм состоит в том, что необходимо 2 прохода по данным. Поскольку ClickHouse использует конвейерную обработку блоков, сохранять весь столбец для второго прохода приходится явно внутри функции. Таким образом, все данные столбца должны помещаться в оперативной памяти одного узла, иначе вычисление будет невозможно.

Функции минимума и максимума реализованы как частные случаи функции перцентиля.

Перцентиль требует получения всех исходных данных на одном сервере базы данных. Все значения сортируются, после чего выполняется нечёткий бинарный поиск по ответу: на каждой итерации вычисляется, сколько чисел больше проверяемого ответа, сколько меньше (к каждой стороне добавляется шум), и отношение сравнивается с прецентилем.

Вывод:

Предложенные реализации anon_count, anon_stddev, anon_avg и anon_sum имеют накладные расходы по сравнению с прототипами, так как требуют хранения всех данных в памяти одного сервера. anon_quantile также требует отсортировать данные, что значительно сказывается на быстродействии и выполняется за $O(nlog(n) + log(n)^2)$. И все агрегатные функции не учитывают пользователей. В случае, когда могут попадаться строки, принадлежащие одному пользователю, рекомендуется воспользоваться преобразователем запросов.

4. Глава 2. Преобразователь запросов.

Удобство вынесения преобразователя в отдельную программу в том, что можно явно выполнять часть запросов напрямую, а часть пропускать через преобразование.

Все таблицы, содержащие приватные данные, должны содержать столбец UserID, служащий для идентификации пользователей.

Преобразователь переводит входящий запрос в абстрактное синтаксическое дерево, а потом собирает обратно, накладывая некоторые модификации. А именно:

Сохранение UserID

Если UserID явно не включен в оператор запроса, но присутствует в таблице, этот столбец добавляется. При слиянии таблиц, если ровно у одной есть UserID, он переходит в UserID результата слияния. В случае, если у обеих сливаемых таблиц есть этот столбец, равенство значений этого столбца добавляется в условия слияния.

Объединение строк, принадлежащих одному пользователю

Для каждого пользователя выбирается не более M строк. После этого агрегатные функции меняются на дифференциально приватные аналоги, при этом шум увеличивается в M раз. Параметр M выбирается аналитиком для конкретного запроса.

Предотвращение утечки ключей GROUP BY

Сам факт вхождения значения в список ключей однозначно говорит о его присутствии. К примеру, если сгруппировать данные по пользователям, то можно получить истинный список пользователей. Для борьбы с этим в отчёт включаются только те ключи, в формировании которых участвовало хотя бы K пользователей. K - конфигурируемый параметр, рекомендуемое значение 2. Таким образом, добавление или удаление пользователя из базы данных может привести к гарантированному появлению или исчезновению ключа.

Произвольные отношения владения

Предложенная модель не рассматривает ситуации, когда одной строкой владеют несколько пользователей. Рассмотрение такого сценария позволит снять все ограничения на слияния, останется только поддерживать множество владельцев для каждой строки. Однако для сохранения приватности необходимо и достаточно увеличить шум в M_u раз, где M_u - максимальное количество пользователей, владеющих одной строкой, так как каждая строка вносит вклад во всех владеющих пользователей. Однако такой метод не упоминается в существующих работах, предположительно по причине значительного роста шума. Внедрение в ClickHouse также потребовало бы работы с множествами. В итоге, подобный сценарий в работе не рассматривается. Если аналитик обладает информацией о максимальном совместном владении, он может самостоятельно уменьшить параметр ε в необходимое количество раз.

Вывод:

Механизм преобразования запросов позволяет с минимальными модификациями синтаксиса выполнять дифференциально приватные запросы, однако спектр доступных запросов на данный момент сильно ограничен.

5. Тестирование

Использовался компьютер с процессором Intel Core i7-8750H и 16 гигабайтами памяти. Параметр max_threads имел значение 4, база состояла из 1 млн записей, агрегировался столбец типа Int64, числа в котором были случайно выбраны из равномерного распределения на [-1E12, 1E12]. Результаты таковы:

Функция	count	sum	avg	stddev	quantile(0.7)	max
Падение производительности, по сравнению с неанонимной	1%	4%	3%	2%	32%	96%

Примечательно, что для count, sum, avg и stddev накладные расходы незначительны, в то время как для quantile ощутимо замедлилось время выполнения. В случае распределенной системы замедление будет ещё значительнее, а если столбец не поместится в памяти, то выполнение приведёт к ошибке. max выполнялся не за O(1) времени и памяти, а ресурсоёмким методом quantile, что привело к падению производительности в 25 раз.

6. Заключение.

Представленное решение реализует механизм дифференциальной приватности. Большинство простых запросов могут быть выполнены. Однако присутствуют существенные ограничения: каждая запись может принадлежать только одному пользователю. Реализованный алгоритм определения коэффициента шума хоть и гарантирует ε-дифференциальную приватность, но добавляемый шум очень быстро разрастается в нетривиальных случаях, например при работе с вещественными числами. Задачи работы выполнены настолько, насколько это позволяет дифференциальная приватность.

Реальная применимость дифференциальной приватности крайне ограниченна. Даже небольшое (около 100) количество одинаковых запросов позволяет получить приватные данные с достаточной для большинства случаев уверенностью.

В будущем стоит применить эластичную дифференциальную приватность для запросов, которые составлены доверенными аналитиками, а также провести более глубокий анализ возможных атак на систему и внести соответствующие шаблоны в преобразователь запросов. Также возможно применение распределённых методов вычисления quantile.

7. Использованная литература

[1] Dwork C., Lei J. Differential privacy and robust statistics //Proceedings of the forty-first annual ACM symposium on Theory of computing. – 2009. – С. 371-380

[2] Wilson R. J. et al. Differentially Private SQL with Bounded User Contribution //arXiv preprint arXiv:1909.01917. – 2019.

[3] Johnson N., Near J. P., Song D. Towards practical differential privacy for SQL queries // Proceedings of the VLDB Endowment. – 2018. – Т. 11. – №. 5. – С. 526-539.

[4] Celia Yuxin Zhang, differential-privacy (Google), 2019 // https://github.com/google/differential-privacy

[5] Noah Johnson, Joe Near, sql-differential-privacy (Uber), 2017 // https://github.com/uber-archive/sql-differential-privacy

[6] Naoise Holohan, differential-privacy-library (IBM), 2019 // https://github.com/IBM/differential-privacy-library

[7] Документация ClickHouse // https://ClickHouse.tech/docs/ru/

[8] Samarati, Pierangela; Sweeney, Latanya. Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression (1998) // Harvard Data Privacy Lab. Retrieved April 12, 2017.

[9] Cynthia Dwork and Aaron Roth. The Algorithmic Foundations of Differential Privacy. (2014) // Found. Trends Theor. Comput. Sci. 9, 3–4 (August 2014), 211–407.

[10] Dwork, Cynthia and Moni Naor. On the Difficulties of Disclosure Prevention in Statistical Databases or The Case for Differential Privacy (2014) // Journal of Privacy and Confidentiality: Vol. 2: Iss. 1, Article 8