Privacidade Diferencial em Sistemas *Polystore*: uma Abordagem Prática*

Lucas Bertelli[1], Victor Ströele[2], Javam Machado[3], Daniel de Oliveira[1]

[1]Instituto de Computação – Universidade Federal Fluminense (UFF)

lucasbm@id.uff.br, danielcmo@ic.uff.br

[2]Universidade Federal de Juiz de Fora (UFJF)

victor.stroele@ice.ufjf.br

[3]Universidade Federal do Ceará (UFC)

javam.machado@dc.ufc.br

Resumo. *Diversas técnicas são capazes de garantir a privacidade de dados, em especial em Sistemas de Gerência de Banco de Dados (SGBDs). Entretanto, nos dias atuais muitas organizações armazenam os dados em seu formato bruto em data lakes. Como os dados podem ser encontrados em múltiplos formatos, os sistemas Polystore são utilizados para conseguir consultá-los de forma integrada. Porém, os mesmos não consideram questões de privacidade, delegando essa responsabilidade para os SGBDs subjacentes. Nesse artigo, propomos uma abordagem chamada* `DIMPLY` *para integrar mecanismos de privacidade em sistemas Polystore. Os usuários do* `DIMPLY` *submetem consultas na sintaxe do sistema Polystore e recebem os resultados anonimizados. Como técnica de privacidade, escolhemos a privacidade diferencial. Para avaliar o* `DIMPLY`*, utilizamos um dataset de exames de casos suspeitos de Zika no Brasil.*

Abstract. *Several techniques guarantee data privacy, especially in Database Management Systems (DBMSs). However, nowadays many organizations store data in its raw format in data lakes. As the data can be found in multiple formats, Polystore systems are used to query data in an integrated way. However, Polystore systems do not consider privacy issues, delegating this responsibility to the underlying DBMSs. In this paper, we propose an approach called* `DIMPLY` *to couple privacy mechanisms into Polystore systems.* `DIMPLY` *users submit queries in the Polystore system syntax and receive anonymized results. As a privacy technique, we chose differential privacy. To evaluate* `DIMPLY`*, we used a dataset of exams of suspected cases of Zika in Brazil.*

1. Introdução

A privacidade de dados é uma responsabilidade legal das organizações. No Brasil, a Lei Geral de Proteção de Dados Pessoais (LGPD)[1] estabelece que os processos que lidam com

*O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001. A pesquisa foi também apoiada parcialmente por CNPq e FAPERJ.

[1]www.planalto.gov.br/ccivil_03/_ato2015-2018/2018/lei/l13709.htm

dados pessoais são obrigados a seguir medidas que respeitem princípios como o armazenamento e a consulta aos dados utilizando técnicas de *pseudonimização* ou *anonimização* [Ramos and Silva 2019], de forma que não seja possível identificar os indivíduos em um *dataset*. Em especial, a anonimização objetiva preservar a privacidade por meio da alteração dos valores originais dos atributos do *dataset*. Contudo, essa modificação implica em perdas e pode reduzir a utilidade dos dados. O desafio é anonimizar os dados ao mesmo tempo em que se procura garantir a utilidade dos mesmos, o que pode não ser trivial.

Ao longo das últimas décadas, diversas técnicas de anonimização de dados foram desenvolvidas, com os modelos sintáticos sendo os primeiros a serem propostos, *e.g.*, o k-anonimato [Sweeney 2002] e a l-diversidade [Machanavajjhala et al. 2007]. Porém, tais modelos são vulneráveis à ataques maliciosos que utilizam conhecimento prévio por parte do atacante. Muito esforço foi despendido na busca de técnicas que garantissem um nível de privacidade forte, como alcançado pelas técnicas RAPPOR (*Randomized Aggregatable Privacy-Preserving Ordinal Response*) [Erlingsson et al. 2014] e a Privacidade Diferencial (PD) [Dwork et al. 2006].

Em especial, a PD é uma técnica de anonimização de dados com um forte rigor matemático que permite análises estatísticas sobre *datasets* enquanto preserva a privacidade dos indivíduos. A PD foi proposta inicialmente no contexto de consultas interativas, nas quais o usuário submete uma consulta e recebe a resposta de forma anonimizada. Existe também o uso da PD para a publicação de *datasets* anonimizados, quando a PD é aplicada sobre o *dataset* antes de publicá-lo e as consultas são realizadas sobre o *dataset* já anonimizado [de Oliveira et al. 2019]. Para sanitizar os dados, a PD utiliza um algoritmo de anonimização ou mecanismo responsável por inserir um ruído aleatório nos dados do *dataset* ou no retorno da consulta, *e.g.*, o mecanismo de Laplace [Dwork et al. 2014], o Gaussiano [Dwork et al. 2014] e o de Resposta Randômica (RR) [Warner 1965]. Cada um dos mecanismos possui vantagens e desvantagens, e pode ser mais adequado para um determinado tipo de consulta. Avaliar a utilidade de cada um deles para uma determinada consulta se torna uma importante tarefa a ser realizada.

Ao mesmo tempo em que a técnica de PD foi proposta, o modo como os dados são armazenados e consultados também tem mudado. Nos últimos anos, percebeu-se uma tendência de se armazenar os dados em seus formatos brutos, em um ambiente de *Data Lake* [Nargesian et al. 2019]. Uma vez que os dados se encontram armazenados em múltiplos formatos, foram propostos sistemas que permitem que os mesmos sejam consultados por meio de uma única linguagem de consulta, e sem exigir um *schema* único, chamados de sistemas *Polystore* [Duggan et al. 2015, Mendes et al. 2020]. Tais sistemas atuam como uma camada sobre fontes de dados existentes, submetendo consultas para múltiplas fontes de dados e integrando o resultado. Entretanto, garantir a privacidade de dados em tais sistemas ainda é um problema a ser atacado.

Atualmente, os sistemas *Polystore* delegam a responsabilidade de anonimização para as fontes de dados que se encontram na camada subjacente. Como cada fonte de dados pode utilizar técnicas de privacidade diferentes (ou não utilizar), se torna difícil mensurar a perda de privacidade total de uma consulta submetida. Tomemos como exemplo o cenário apresentado por [de Lourdes Maia Silva et al. 2021] onde ao combinar dois *datasets*, um não anonimizado e o outro pseudonimizado, os autores foram capazes de identificar indivíduos no *dataset*. Apesar do trabalho de [de Lourdes Maia Silva et al. 2021] não se encontrar no

contexto de sistemas *Polystore*, ele deixa claro um problema que pode ocorrer ao se integrar dados de múltiplas fontes. Mesmo que sejam disponibilizados apenas os resultados agregados ainda é possível um ataque, especialmente se o atacante possui conhecimento prévio obtido de outras fontes de informações [Backstrom et al. 2007].

Diante dessa lacuna, propomos nesse artigo um *middleware* de PD para sistemas *Polystore* chamado `DIMPLY`, que tem como objetivo prover respostas anonimizadas para consultas submetidas, aplicando a técnica de PD sobre os dados. De forma a maximizar a utilidade dos dados, o `DIMPLY` escolhe o mecanismo de PD que apresenta o menor erro relativo para a consulta submetida. A medida do erro relativo avalia o quão distantes as respostas anonimizadas se encontram das originais. Para avaliar o `DIMPLY`, foi utilizado um *dataset* com dados reais contendo casos suspeitos do Vírus da Zika (ZIKV) no Brasil. A área da saúde tem sido alvo de frequentes ataques maliciosos, tornando-se de extrema importância a anonimização dos dados.

Este artigo se encontra organizado em quatro seções além da Introdução. A Seção 2 apresenta o referencial teórico e discute trabalhos relacionados. A Seção 3 apresenta os detalhes do `DIMPLY`. A Seção 4 apresenta a avaliação experimental, e, finalmente, a Seção 5 conclui o presente artigo.

2. Referencial Teórico e Trabalhos Relacionados

Nesta seção são apresentados os conceitos necessários para o entendimento deste artigo, como PD e sistemas *Polystore*. Além disso, são discutidos os trabalhos relacionados.

2.1. Privacidade Diferencial

A PD [Dwork et al. 2006] é um modelo matemático que permite análises estatísticas sobre um *dataset*, sem comprometer a privacidade dos indivíduos. Com fortes garantias de privacidade, ela assegura que qualquer resposta a uma determinada consulta tem valor igualmente possível e independe da presença, ou ausência, de um indivíduo no *dataset*. Tal característica é obtida por meio de um mecanismo que insere um ruído aleatório na resposta das consultas. A PD foi projetada para um ambiente interativo, onde os usuários submetem as consultas a um *dataset*, que por sua vez responde a consulta anonimizada por um mecanismo de aleatoriedade. Com a PD é possível mensurar matematicamente o *tradeoff* entre utilidade e privacidade, e calibrar, por meio do parâmetro ϵ, de acordo com a legislação vigente. A PD também é imune a pós-processamento, mesmo considerando poder computacional, conhecimento prévio e ataques de cruzamentos de dados. Em particular, nesse artigo utilizamos os mecanismos de RR, o de Laplace e o Gaussiano. A seguir discutimos com mais detalhes cada um destes mecanismos.

O mecanismo de Resposta Randômica (RR) é baseado em uma técnica originalmente aplicada para obter respostas privadas em entrevistas. O objetivo é garantir que os entrevistados sejam capazes de responder questões sensíveis, *e.g.,* sexualidade ou crença religiosa, mantendo a confidencialidade das respostas [Warner 1965]. Na RR, é adicionado um processo de aleatoriedade na resposta do entrevistado, mascarando se a resposta é verdadeira ou não. Para cada consulta, o participante joga mentalmente uma moeda, caso o lançamento dessa moeda seja "Coroa" ele sempre responderá a "Verdade", seja ela "Sim" ou "Não". Agora, se no lançamento der "Cara", o participante deve lançar uma segunda moeda e responder "Sim" se der "Cara" e "Não" se der "Coroa". A intuição por trás da RR

é que o mecanismo sempre forneça uma "negação plausível", de forma que mesmo que o indivíduo venha a ser identificado como participante da pesquisa, não será possível saber se a sua resposta dada foi verdadeira. Assim, um mecanismo de RR $\mathcal{M}$ com domínio A e intervalo discreto B está associado a um mapeamento $M : A \rightarrow \Delta(B)$. Para uma entrada $a \in A$, o algoritmo $\mathcal{M}$ produz uma saída $\mathcal{M}(a) = b$ com probabilidade $(M(a))_b$ para cada $b \in B$.

Antes de descrever os mecanismos de Laplace e Gaussiano, precisamos explicar o conceito fundamental de sensibilidade global da consulta, chamada de Δf. Dado um conjunto de *datasets* $\mathbb{D}$, todos os *datasets* $D_i \in \mathbb{D}$ derivados a partir da remoção de um indivíduo i do *dataset* original D são definidos como *datasets* vizinhos. Seja f uma função de consulta que mapeia o *dataset* em vetores de números reais. A sensibilidade global da função f é dada por $\Delta f = \max_{x,y \in \mathbb{D}} ||f(x) - f(y)||$, para todo x, y diferindo de no máximo um elemento. A definição do Δf permite encontrar a menor quantidade de ruído necessária para transformar a resposta de uma consulta em diferencialmente privada. A definição do Δf está diretamente ligada à consulta, e assim, cada consulta possui seu próprio Δf. Quanto maior for o valor de Δf, mais ruído é necessário ser adicionado para garantir a privacidade.

O mecanismo de Laplace [Dwork et al. 2006] oferece PD para consultas que retornam valores numéricos, sendo o mais aplicado a consultas estatísticas como as utilizadas no estudo de caso desse artigo. Este mecanismo introduz um ruído aleatório na resposta original com base na distribuição de Laplace. O ruído se baseia nos parâmetros ϵ e Δf. A distribuição de Laplace possui média μ e escala b, sendo que na geração das nossas variáveis aleatórias utilizamos uma distribuição de Laplace centrada em 0, *i.e.*, $\mu = 0$. A função densidade de probabilidade da distribuição de Laplace é dada por $Lap(z|b) = \frac{1}{2b} exp(\frac{|z|}{b})$. $Lap(b)$ denota uma distribuição de Laplace com escala b, e $X \sim Lap(b)$ denota uma variável aleatória gerada a partir da distribuição de Laplace. O mecanismo de Laplace calcula o valor da função f e adiciona um ruído aleatório que segue a distribuição de Laplace com escala $b = \frac{\Delta f}{\epsilon}$. Formalmente, para uma dada função $f : \mathbb{N}^{|x|} \rightarrow \mathbb{R}^k$, o mecanismo de Laplace pode ser definido como $M_L(x, f, \epsilon) = f(x) + (Y_1...Y_k)$, onde $Y_i \sim Lap(\frac{\Delta f}{\epsilon})$, i.i.d. Além disso, é válido destacar que o mecanismo de Laplace utiliza a distância de Manhattan no cálculo do Δf, para consultas que retornam vetores de valores.

O mecanismo Gaussiano [Dwork et al. 2014] consiste em adicionar um ruído aleatório que segue a distribuição Gaussiana. Porém, diferentemente do mecanismo de Laplace, o mecanismo gaussiano não satisfaz a privacidade ϵ-diferencial, mas a privacidade (ϵ, δ)-diferencial, onde δ é um fator de relaxamento aplicado. Apesar de utilizar o fator de relaxamento δ, [Dwork et al. 2014] provam que usando ruído gaussiano com variância calibrada para $\Delta f \ ln(1/\delta)/\epsilon$, pode-se obter privacidade (ϵ, δ)-diferencial. O mecanismo Gaussiano utiliza, para calcular o Δf da consulta, a distância Euclidiana que é menor do que distância de Manhattan. Isso significa que, para altas dimensões, a diferença de vetores de múltiplas dimensões terá valores menores em comparação com os do Laplace. Por consequência o Δf será menor e o mecanismo Gaussiano inserirá menos ruído nesses dados. Porém, no contexto deste artigo temos apenas uma dimensão, e com isso o mecanismo Gaussiano tende a inserir mais ruído nos dados do que o mecanismo de Laplace. [Dwork et al. 2014] discutem que no cenário de uma única dimensão, o Δf calculado com distância de Manhattan é igual ao calculado com distância Euclidiana, *i.e.*, Δf = $\Delta_2(f)$. O que é bom para diminuir drasticamente o *overhead* do `DIMPLY`, visto que só precisamos calcular um Δf de forma a avaliar os dois mecanismos. Formalmente, a distribuição Gaussiana é dada

por $f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$. onde, μ é a média, σ o desvio padrão e σ^2 a variância. Assim, para qualquer função d-dimensional arbitrária, seja $f : \mathbb{N}^{|x|} \rightarrow \mathbb{R}^d$, o mecanismo de Gaussiano [Dwork et al. 2014] é definido como $M_G(x, f, \epsilon) = f(x) + (Y_1...Y_k)$, onde $Y_i \sim Gau(c_2(f)/\epsilon)$, i.i.d.

2.2. Sistemas *Polystore*

Um sistema *Polystore* [Duggan et al. 2015, Mendes et al. 2020] pode ser definido como qualquer sistema capaz de manter dados de múltiplos Sistemas de Gerência de Banco de Dados (SGBDs) que seguem modelos heterogêneos. Ao mesmo tempo, os sistemas *Polystore* são capazes de consultar esses dados de forma integrada por meio de uma interface única e com alto desempenho. É importante distinguir sistemas *Polystore* de SGBDs federados, que também integram diferentes SGBDs e possibilitam consultas com uma interface única, porém estes integram bases de dados com modelos homogêneos. Nos sistemas *Polystore* não há a necessidade de definição de um *schema* padrão para os dados, o que é ótimo para ser usado em conjunto com a abordagem de *Data Lake* [Nargesian et al. 2019], uma vez que os *Data Lakes* são sistemas baseados no conceito *schema-on-read*, de forma que os arquivos são carregados sem associação com nenhum *schema*, e só no momento da consulta é definida a sua estrutura a partir dos seus metadados armazenados.

Existem diversos sistemas *Polystore* disponíveis para uso, *e.g.*, o BIGDAWG [Duggan et al. 2015] e o Apache Drill (`drill.apache.org`), entretanto apresentaremos unicamente detalhes do Drill, que foi o sistema *Polystore* utilizado neste artigo. O Drill é capaz de processar os dados (*in-situ*) sem exigir que os usuários definam *schemas* ou transformem os dados previamente. Com o Drill, é possível analisar dados estruturados ou semi-estruturados em larga escala, armazenados de forma distribuída. Apesar de compatível com SGBDs relacionais, o foco do Drill é possibilitar acesso e consultas integradas a SGBDs NoSQL, além do *stack* Hadoop. Para garantir um bom desempenho no processamento das consultas, o Apache Drill faz uso de processamento em memória principal. O serviço *Drillbit* é o responsável por aceitar requisições, processá-las e retornar os resultados. Quando um serviço *Drillbit* é executado, o Drill maximiza a localidade dos dados durante a execução, e assim evita a necessidade de transportar dados na rede ou movê-los entre os nós. Atualmente, o Drill oferece apoio para SGBDs relacionais e para os seguintes SGBDs NoSQL: MongoDB, HBase e MonetDB. Além disso, o Drill oferece acesso a soluções de armazenamento em nuvem, como por exemplo o *Amazon S3*, o *Google Cloud Storage* e o *Azure Blob Storage*.

2.3. Trabalhos Relacionados

Até o momento não foram encontrados na literatura trabalhos que abordem de forma prática o uso de PD em conjunto com sistemas *Polystore*. Entretanto, [Kraska et al. 2019] discutem uma abordagem teórica para garantir privacidade em sistemas *Polystore* que considera PD. Dessa forma, analisamos trabalhos que resolvem questões similares as do `DIMPLY`, ainda que não aplicados no mesmo contexto. O PINQ [McSherry 2009] é uma plataforma de consultas integradas com preservação da privacidade nas tarefas de análise de dados. Ele fornece aos analistas uma interface de acesso a dados não criptografados, ao mesmo tempo em que garante a privacidade de dados. No PINQ, é possível estabelecer definições de privacidade para cada usuário. Antes da execução de cada consulta de agregação, o PINQ verifica se o usuário pode realizar a consulta com aquele valor de ϵ. Se for permitido, a

resposta da consulta é anonimizada com o mecanismo de Laplace. O wPINQ (*Weighted PINQ*) [Proserpio et al. 2014], uma extensão do PINQ, apoia consultas com junções do tipo *equijoin*, que o PINQ não apoia.

O FLEX [Johnson et al. 2018] é um sistema de PD para consultas SQL com base na *sensibilidade elástica*. O FLEX anonimiza os dados com um mecanismo próprio que estende o de Laplace. Os autores provam que o mecanismo FLEX garante privacidade (ϵ, δ)-diferencial. O FLEX utiliza uma aproximação da sensibilidade local da consulta, chamada de sensibilidade elástica, para dimensionar a magnitude do ruído do mecanismo de Laplace para uma dada consulta. O APEx [Ge et al. 2019] é um sistema que permite aos analistas de dados submeterem consultas em SQL e retorna o resultado com um ruído associado. O APEx realiza esse processo de forma adaptativa. *i.e.*, o usuário não necessita informar os parâmetros para a anonimização. O APEx se concentra apenas em consultas que envolvem a função COUNT.

Se compararmos as abordagens supracitadas com o DIMPLY, podemos perceber algumas diferenças importantes. A primeira é que tais abordagens não se encontram preparadas para serem acopladas aos sistemas *Polystore*. Outra diferença é que as abordagens, com exceção do PINQ, oferecem apoio a anonimização de consultas de contagem, enquanto o DIMPLY apoia também consultas com outras funções de agregação. As abordagens citadas também anonimizam apenas com o mecanismo de Laplace (ou uma variante do mesmo), enquanto o DIMPLY oferece atualmente três opções de mecanismo diferentes. Em especial, o PINQ e wPINQ assumem que a sensibilidade global $\Delta f = 1$ para todas as consultas, diferente do DIMPLY. No caso do FLEX, ele não lida tão bem com *datasets* pequenos, uma vez que ele insere muito ruído nos dados.

3. Abordagem Proposta: DIMPLY

O DIMPLY tem como objetivo atuar como uma camada entre o sistema *Polystore* e o usuário, de forma a prover um resultado anonimizado para uma consulta. Um diferencial do DIMPLY é que ele auxilia na escolha do mecanismo adequado a ser usado. Essa escolha é uma tarefa importante e tem relação direta com a qualidade do resultado. A arquitetura do DIMPLY é apresentada na Figura 1. A utilização do DIMPLY segue um fluxo bem definido. No passo ❶, o usuário submete uma consulta para o DIMPLY seguindo a linguagem apoiada pelo sistema *Polystore*. No passo ❷, o *Query Broker* verifica se a consulta é uma consulta estatística utilizando as funções de agregação suportadas. Em sua versão atual, o DIMPLY permite que sejam realizadas somente consultas estatísticas, e apoia as seguintes funções de agregação: SUM, AVG, VARIANCE, STDEV, MIN e MAX. Atualmente, o DIMPLY não oferece apoio a função COUNT, uma vez que a mesma é sobrecarregada na sintaxe de consulta do Drill. Como o DIMPLY obtém os atributos e funções envolvidos na consulta a partir do plano de execução da consulta retornado pelo Drill, não é possível diferenciar os dois tipos de COUNT. Caso a consulta esteja dentro do formato esperado, o *Query Broker* envia a consulta para o sistema *Polystore* e atualiza o banco de metadados com dados da consulta submetida e os operadores associados. Tais informações são utilizadas pelo *Modelo de Custo* para encontrar o Δf previamente calculado para consultas que possuam as mesmas funções de agregação que a consulta submetida.

O sistema *Polystore* processa a consulta e envia a resposta sem anonimização para o DIMPLY junto com o plano de consulta que foi executado. No passo ❸, o *Query Broker* recebe a resposta não anonimizada, identifica no plano de consulta enviado quais fun-

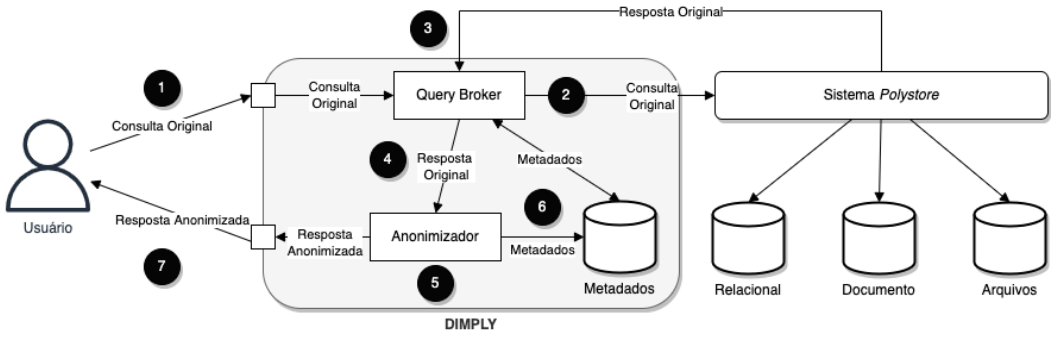

Figura 1. Arquitetura do `DIMPLY`.

ções de agregação foram usadas e verifica por meio do *Modelo de Custos* (Passo ❹) se já existe um mecanismo indicado para ser utilizado para aquela consulta, *i.e.*, o mecanismo que apresenta o menor erro relativo. Caso exista, o *Query Broker* envia a resposta original para o *Anonimizador* junto com o mecanismo a ser utilizado. Caso ainda não existam metadados que apoiem essa escolha, o *Anonimizador* anonimizará a resposta original com todos os mecanismos disponíveis (Passo ❺). Após a anonimização (com um ou três mecanismos), o *Anonimizador* atualiza o banco de metadados (Passo ❻) e envia o resultado anonimizado para o usuário (Passo ❼). Em sua versão atual, o `DIMPLY` foi construído sobre o sistema *Polystore* Apache Drill, e está limitado aos tipos de armazenamentos compatíveis com o mesmo, que podem ser vistos na Subseção 2.2. O `DIMPLY` foi desenvolvido em Python 3, e como o Drill não possui apoio nativo para Python, utilizamos um *wrapper* para consultar via API REST o Drill. Portanto, em sua versão atual, o `DIMPLY` se encontra limitado a requisições com a máxima latência do protocolo HTTP. O *wrapper* utilizado no desenvolvimento foi o Pydrill. O código-fonte do `DIMPLY` pode ser obtido em `https://github.com/UFFeScience/dimply`.

3.1. Modelo de Custo

Nesta subseção apresentamos o modelo de custo que auxilia na escolha do melhor mecanismo de PD para uma determinada consulta. Em um momento inicial, o `DIMPLY` precisa anonimizar um resultado de consulta com todas as opções de mecanismos de PD disponíveis, para obter informações do desempenho de cada mecanismo. Tal desempenho é calculado a partir do tempo de execução do mecanismo e do erro relativo. O modelo de custo entra em funcionamento em um segundo momento, que é quando o `DIMPLY` já possui informações suficientes em seu banco de metadados. O modelo de custo proposto leva em consideração duas dimensões simultaneamente para minimização (biobjetivo), atribuindo pesos a cada um dos critérios em sua escolha: (i) tempo de execução e (ii) erro relativo. Antes de realizar a escolha baseada nos critérios supracitados, o modelo de custo filtra quais consultas anteriormente executadas possuem o mesmo padrão da nova consulta submetida. Os critérios de semelhança considerados foram: (i) valor igual do parâmetro ϵ, e caso não encontre um valor igual, considere ϵ dentro do intervalo $(\epsilon_j - \theta \leq \epsilon_{q_i} \leq \epsilon_j + \theta) \mid \theta \geq 0$, onde θ é um *threshold* definido, e (ii) mesmo identificador único de conjunto de funções de agregação. O identificador único é um identificador numérico atribuído pelo `DIMPLY` para um conjunto de funções de agregação identificadas naquela consulta. Por exemplo, caso uma determinada consulta utilize as funções de agregação `SUM` e `AVG`, é atribuído um ID interno no `DIMPLY` para representar essa combinação. Considere as seguintes variáveis definidas na Tabela 1.

Tabela 1. Variáveis do Modelo de Custo

Variáveis	Descrição
M	Mecanismo de Privacidade Diferencial analisado
D_M	Conjunto com todos os mecanismos de PD existentes no DIMPLY
q_j	Consulta a ser anonimizada
q_i	Consulta semelhante a q_j encontrada no banco de metadados
Q	Conjunto de todas as consultas semelhantes a q_j, *i.e.*, $q_i \in Q$
$\overline{t_{exec}}(M, q_j)$	Média do tempo de execução das consultas semelhantes a q_j para um dado mecanismo M
$\overline{ERel}(M, q_j)$	Média do Erro Relativo de um mecanismo M e consultas semelhantes a consulta q_j
α	Peso definido para um determinado critério
ϵ_j	Parâmetro de entrada ϵ para a consulta q_j
ϵ_{q_i}	Valor de ϵ utilizado na anonimização das consultas semelhantes a q_j
θ	*Threshold* do ϵ_i
$id_{agreg}(q_j)$	ID atribuído pelo DIMPLY para um conjunto de funções de q_j
$id_{agreg}(Q)$	ID atribuído pelo DIMPLY para um conjunto de funções de agregação das consultas semelhantes a q_j
$M_{ideal}(q_j)$	Mecanismo de PD ideal para a consulta submetida

Foi utilizado um modelo de custo ponderado em que o usuário informa o peso de cada critério. A vantagem de oferecer um modelo de custo ponderado é que o usuário pode realizar uma sintonia fina de critérios de forma simples, sem que tenha que eleger somente um critério principal. Desta forma, para cada consulta q_j submetida, o DIMPLY identifica um conjunto Q de consultas semelhantes a q_j. Tal semelhança depende de quais funções de agregação são utilizadas nas consultas. Uma vez identificada uma consulta $q_i \equiv q_j$ é, então, realizada uma pesquisa para descobrir o melhor mecanismo M na lista de mecanismos disponíveis D_M para anonimizar q_j seguindo o modelo de custo. Assim, dada uma consulta q_j, temos de encontrar $M \in D_M$ que minimize a função de custo $f(M, q_j) = \alpha\, \overline{t_{exec}}(M, q_j) + (1 - \alpha)\overline{ERel}(M, q_j)$.

3.2. Limitações do DIMPLY

Além da limitação dos tipos de função de agregação apoiados pelo DIMPLY, é válido ressaltar outras limitações da abordagem: (i) os atributos e funções de agregação devem respeitar a mesma ordem de aparição na cláusula SELECT e na cláusula WHERE, (ii) consultas aninhadas não são suportadas, (iii) atributos não associados à funções de agregações no SELECT devem ser declaradas na cláusula especial WHERE_FREE_COLUMNS e devem ter sido registrados previamente pelo administrador como liberados por não oferecerem risco à privacidade.

4. Avaliação Experimental

De forma a avaliar o DIMPLY, foi selecionado um *dataset* real extraído do sistema GAL (Gerenciador de Ambiente Laboratorial) do SUS, e um conjunto de consultas significativas. Nesta seção apresentamos os resultados obtidos ao submeter as consultas no DIMPLY.

4.1. Estudo de Caso

Foi utilizado um *dataset* contendo originalmente 1.846.602 tuplas exportadas do sistema GAL do SUS. Esse *dataset* possui 104 atributos com informações relativas a exames realizados por pacientes suspeitos e confirmados com o vírus da Zika, *e.g.*, Nome do Paciente, Sexo, Raça/Cor, CPF, Idade, Data de Nascimento, Data de Atendimento, UF, Agravo (Diagnóstico), Sintomas e Exames Solicitados. Foi realizado um pré-processamento no *dataset*,

e as tuplas onde o campo UF e DATASOL (Data da Solicitação do Exame) eram vazios foram removidas. Foram removidos os atributos identificadores e mantidos os atributos semi-identificadores (atributos que não são identificadores, mas podem identificar um indivíduo). Após essa etapa, restaram 1.840.198 tuplas. O *dataset* foi exportado para o formato CSV, fragmentando o mesmo horizontalmente de acordo UF e Mês/Ano (no padrão zikadb-UF-mês-ano.csv). Ao final, obtivemos um total de 795 arquivos no formato CSV.

4.2. Ambiente de Execução

A avaliação experimental foi realizada em uma máquina com 16 GB de RAM DDR4, processador Intel Core i5-7200U 2.50 GHz x 4, com placa de vídeo Geforce 940MX, com HDD Sata (sem SSD). O sistema operacional utilizado foi Ubuntu 18.04.5 LTS 64 bits. Foram instalados os *softwares* Python 3.6.9, Apache Drill 1.17.0, Apache HDFS e PostgreSQL, além das bibliotecas do Python *cmath*, *numpy*, *random*, *time*, *re* e *json*. É importante ressaltar que o Apache Drill foi configurado para utilizar no máximo 8 GB de RAM. A comunicação do DIMPLY com o Drill é feita via PyDrill 0.3.4.

4.3. Configuração do Experimento

Optamos por explorar empiricamente os seguintes valores para $\epsilon = \{0,01; 0,05; 0,1; 0,25; 0,5; 1\}$ no caso dos mecanismos de Laplace e Gaussiano. Os mecanismos de Laplace e Gaussiano foram centralizados com média $\mu = 0$. É importante observar que o ϵ não é uma medida absoluta de privacidade, mas sim uma medida relativa. Ou seja, um mesmo valor de ϵ oferece garantias de privacidade diferentes com base no domínio do atributo em questão e nas consultas. Para o mecanismo RR não é necessário configurar nenhum parâmetro. Em relação aos parâmetros configuráveis na função objetivo do modelo de custo, utilizamos $\alpha = 0,6$. Além disso, definimos $\theta = 0,4$. Os valores de α e θ foram escolhidos arbitrariamente para testar o DIMPLY e são configuráveis a critério do usuário. Definimos também as consultas significativas de forma a contemplar todas as funções de agregações apoiadas pelo DIMPLY. Dessa forma, as consultas significativas utilizadas nesse experimento são apresentadas na Tabela 2.

Tabela 2. Consultas executadas no DIMPLY

Q1	Qual a média de idade dos pacientes com Zika no RJ em 2016?
Q2	Qual a média de idade das grávidas com Zika no RJ em 2016?
Q3	Qual a média de idade das crianças com Zika no RJ em 2016?
Q4	Qual a média de idade dos idosos com Zika no RJ em 2016?
Q5	Qual o desvio padrão da idade dos pacientes com Zika no RJ em 2016?
Q6	Qual a média de idade dos pacientes com Zika em municípios com mais de 100 mil habitantes no RJ em 2016?
Q7	Qual a média de idade das grávidas com Zika em municípios com menos de 50 mil habitantes no RJ em 2016?

4.4. Resultados

A Figura 2 apresenta a média do tempo de execução, em segundos, de sete consultas sem a anonimização e com anonimização, para cada mecanismo disponível atualmente no DIMPLY. Dessa forma, é possível analisar o *overhead* da etapa de anonimização. Esses valores consideram que o Δf já foi calculado *a priori*. É possível concluir que com o cálculo do Δf prévio, o *overhead* para os mecanismos de Laplace e Gaussiano foi aceitável, mesmo com um *overhead* de 480%, em média. Isso se dá uma vez que o tempo de execução das consultas sem anonimização para esse *dataset* é sempre menor que 500ms. O mecanismo

de Laplace apresentou um tempo de execução ligeiramente menor que o Gaussiano para todas as consultas. O mecanismo de RR não depende do Δf calculado *a priori*, e consequentemente não se beneficia da otimização do DIMPLY em utilizar um Δf previamente calculado. Com isso, o tempo decorrido na sua anonimização foi bastante superior aos demais. Entretanto, o tempo necessário para se calcular o Δf para o mecanismo de Laplace e Gaussiano nas primeiras execuções deve ser considerado na análise (Tabela 3).

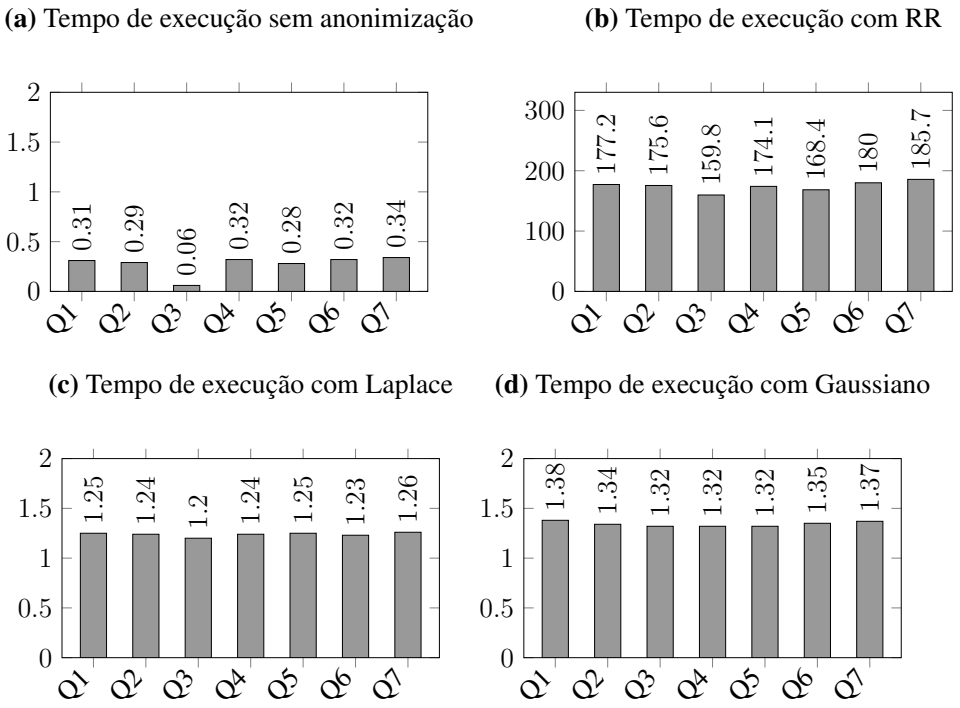

Figura 2. Tempos de execução (em segundos) da anonimização pelos mecanismos.

Tabela 3. Tempo para cálculo do Δf de cada consulta (em segundos).

Consulta	Tempo de Cálculo Δf (em segundos)
Q1	270,51
Q2	268,38
Q3	269,68
Q4	269,81
Q5	259,27
Q6	272,87
Q7	272,24

Ao analisarmos a Figura 2 e a Tabela 3, podemos observar que se o cálculo do Δf for realizado em tempo de execução, o mecanismo de RR ficaria em torno de 100 segundos mais rápido do que os mecanismos de Laplace e Gaussiano para todas as consultas. Porém, o cenário de uso geral do DIMPLY assume que as consultas significativas serão previamente executadas (e o Δf calculado *a priori*) em um horário oportuno, *e.g.*, fora do pico de uso do sistema. Apesar disso, quando há uma modificação na estrutura dos arquivos envolvidos na consulta submetida, o DIMPLY precisa anonimizar novamente com todos mecanismos disponíveis e recalcular o Δf.

Analisemos agora a utilidade dos dados. Utilizaremos a métrica de Erro Relativo, que serve para mensurar a utilidade dos dados. Esse valor está diretamente ligado à distri-

buição dos dados no *dataset*, *i.e.*, o quão distantes os dados se encontram distribuídos e o quanto a sua ausência impacta no resultado da consulta. Essa informação é medida a partir do valor do Δf. Analisando as Tabelas 4 e 5 podemos perceber como o erro relativo se comporta com a variação dos valores de ϵ no intervalo $[0,01; 0,05; 0,1; 0,25; 0,5; 1,0]$ para cada consulta. Nas Tabelas 4 e 5 podemos verificar que valores baixos de ϵ, apesar de garantirem mais privacidade aos indivíduos presentes no *dataset*, implicam em menor utilidade dos dados. Observe que a medida que o valor de ϵ aumenta, o erro relativo diminui para todas as consultas, e isso independe do mecanismo utilizado. Assim, um especialista de privacidade de dados se faz necessário no período de configuração do `DIMPLY` para definir qual valor de ϵ deve ser utilizado para anonimizar as consultas, de forma a gerenciar esse *tradeoff* entre utilidade e privacidade de acordo com o contexto e exigências da sua organização.

Analisando especificamente a Tabela 4, podemos perceber que em todas as consultas o Erro Relativo para todos os valores de ϵ apresentou valores aceitáveis, que mantém a utilidade dos dados. É importante destacar a consulta Q5 para o ϵ = 0,01, onde o erro relativo diminui quase duas ordens de grandeza em relação ao próximo valor de ϵ = 0,05. A Tabela 5, referente ao mecanisno Gaussiano, também apresenta valores de erro relativo altos para a Q5. Neste caso específico, podemos perceber que valores a partir de ϵ = 0,1 já apresentam resultados com um certo nível de utilidade. É importante ressaltar que o modelo de custo apresentado anteriormente já é capaz de fazer essa escolha automaticamente.

A Tabela 6 apresenta o erro relativo da anonimização do resultado de cada uma das consultas com o mecanismo RR para vários conjuntos de execuções, onde em cada consulta o mecanismo RR foi executado dez vezes e o valor apresentado é a média do erro relativo dessas execuções. Observe que os valores do erro relativo do RR são mais aleatórios se comparados com os dos mecanismos de Laplace e Gaussiano. De forma que é complicado saber se anonimização vai manter a utilidade dos dados no longo prazo. Esse comportamento pode ser explicado pela própria implementação do estado da arte do RR que não dimensiona um limiar para o ruído inserido, visto que ele não utiliza, por exemplo, uma sensibilidade global Δf da consulta para dimensionar a quantidade de ruído necessária para se garantir privacidade de dados. Por outro lado, o RR pode ser aplicado em qualquer tipo de consulta, inclusive as que possuem atributos categóricos.

Tabela 4. Erro relativo do mecanismo Laplace para valores de ϵ.

		ϵ					
Consulta	Δf	0,01	0,05	0,1	0,25	0,5	1,0
Q1	0,016141	0,042162	0,005337	0,002861	0,001512	0,000912	0,000218
Q2	0,016141	0,052829	0,007538	0,007098	0,001717	0,001189	0,000413
Q3	0,016141	1,323828	0,413744	0,319265	0,249796	0,296934	0,370228
Q4	0,016141	0,495453	0,312381	0,395607	0,340911	0,344729	0,336834
Q5	0,042119	1,438709	0,062976	0,010226	0,003479	0,001853	0,000788
Q6	0,016141	0,030888	0,006012	0,003016	0,001185	0,000536	0,000270
Q7	0,016141	0,186181	0,039830	0,026318	0,008565	0,003803	0,002516

5. Conclusão

Nos últimos anos, muitas organizações passaram a ter dados armazenados em diferentes SGBDs com modelos heterogêneos. De forma a oferecer maior flexibilidade na análise dos dados, existe uma tendência que as organizações armazenem seus dados em seus formatos

Tabela 5. Erro relativo do mecanismo Gaussiano para valores de ϵ.

		ϵ					
Consulta	Δf	0,01	0,05	0,1	0,25	0,5	1,0
Q1	0,016141	0,130563	0,031255	0,012133	0,005881	0,001899	0,000826
Q2	0,016141	0,151091	0,041702	0,007605	0,006364	0,004964	0,001808
Q3	0,016141	1,357299	1,098258	0,468009	0,307247	0,235683	0,336283
Q4	0,016141	2,353529	0,600366	0,583416	0,311039	0,357326	0,342671
Q5	0,042119	17,967001	1,192552	0,283338	0,047204	0,014423	0,004168
Q6	0,016141	0,182831	0,038602	0,017111	0,006699	0,002287	0,001020
Q7	0,016141	1,069495	0,287755	0,135901	0,046959	0,031026	0,005755

Tabela 6. Erro relativo do mecanismo RR para vários conjuntos de execuções.

Consulta	Conjunto 1	Conjunto 2	Conjunto 3	Conjunto 4	Conjunto 5	Conjunto 6
Q1	0,127888	0,143100	0,194020	0,234873	0,119120	0,082374
Q2	0,093054	0,176988	0,084601	0,097060	0,148610	0,060889
Q3	0,068254	0,252901	0,376025	0,255469	0,057781	0,216585
Q4	0,488495	0,408080	0,814734	0,529738	0,339901	0,108987
Q5	0,155230	0,078142	0,165868	0,272680	0,122857	0,158153
Q6	0,078021	0,290413	0,135161	0,210663	0,119621	0,205092
Q7	0,959084	1,348687	0,763824	0,676354	0,367760	0,270064

originais, sem necessidade de definição de um *schema*. A ideia é se beneficiar de arquiteturas de *Data Lakes*. Porém, se faz necessário consultar esses dados de forma integrada, eficiente e com uma interface de consulta unificada. Para resolver este problema surgiram os Sistemas *Polystore* que são capazes de consultar os dados armazenados sem *schemas*, em diferentes formatos, por meio de uma sintaxe única de consulta. Apesar de representarem um avanço, os sistemas *Polystore* existentes não oferecem mecanismos de anonimização de forma nativa. Ao contrário, eles delegam para os SGBDs subjacentes essa responsabilidade.

Diante disso, propomos nesse artigo o `DIMPLY`, um *middleware* de PD para sistemas *Polystore*. O `DIMPLY` atua entre o usuário e o sistema *Polystore* e provê respostas anonimizadas aplicando o modelo de PD sobre o retorno das consultas. Além disso, o `DIMPLY` escolhe em tempo real qual o melhor mecanismo para cada tipo de consulta estatística submetida. Essa não é uma tarefa trivial, pois existe um *trade-off* entre utilidade e privacidade. A avaliação experimental do `DIMPLY` foi realizada com um *dataset* de dados de exames de pacientes com suspeita de Zika vírus extraído do GAL do SUS, e mostrou a viabilidade do `DIMPLY`. Em geral, o mecanismo de RR foi o que inseriu o menor relativo, garantindo assim mais utilidade para os resultados das consultas Q3, Q4 e Q5. Porém, considerando o Δf já calculado previamente, o tempo de execução do RR foi ordens de grandeza maior que os outros mecanismos. O mecanismo de Laplace foi o que obteve os menores valores de erro relativo para as consultas Q1, Q2, Q6 e Q7. O mecanismo Gaussiano foi o que teve os maiores valores de erro relativo para todas as consultas significativas. Trabalhos futuros incluem avaliar o `DIMPLY` com outros mecanismos de PD e acoplar o `DIMPLY` com outros sistemas *Polystore* como o BigDAWG.

Referências

Backstrom, L., Dwork, C., and Kleinberg, J. (2007). Wherefore art thou r3579x? anonymized social networks, hidden patterns, and structural steganography. In *WWW'07*, pages 181–190.

de Lourdes Maia Silva, M., Chaves, I. C., and Machado, J. C. (2021). Private reverse top-k algorithms applied on public data of COVID-19 in the state of ceará. *J. Inf. Data Manag.*, 12(5).

de Oliveira, D., Neto, E. R. D., et al. (2019). Um estudo comparativo de mecanismos de privacidade diferencial sobre um dataset de ocorrências do ZIKV no brasil. In *Proc. of the 34th SBBD*, pages 253–258. SBC.

Duggan, J., Elmore, A. J., Stonebraker, M., Balazinska, M., Howe, B., Kepner, J., Madden, S., Maier, D., Mattson, T., and Zdonik, S. (2015). The bigdawg polystore system. *ACM Sigmod Record*, 44(2):11–16.

Dwork, C., McSherry, F., Nissim, K., and Smith, A. (2006). Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*, pages 265–284. Springer.

Dwork, C., Roth, A., et al. (2014). The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3–4):211–407.

Erlingsson, Ú., Pihur, V., and Korolova, A. (2014). Rappor: Randomized aggregatable privacy-preserving ordinal response. In *SIGSAC'14*, pages 1054–1067.

Ge, C., He, X., Ilyas, I. F., and Machanavajjhala, A. (2019). Apex: Accuracy-aware differentially private data exploration. In *SIGMOD '19*, pages 177–194.

Johnson, N., Near, J. P., and Song, D. (2018). Towards practical differential privacy for sql queries. *Proceedings of the VLDB Endowment*, 11(5):526–539.

Kraska, T., Stonebraker, M., Brodie, M. L., Servan-Schreiber, S., and Weitzner, D. J. (2019). Schengendb: A data protection database proposal. In *Poly'19*, volume 11721, pages 24–38. Springer.

Machanavajjhala, A., Kifer, D., Gehrke, J., and Venkitasubramaniam, M. (2007). l-diversity: Privacy beyond k-anonymity. *ACM TKDD*, 1(1):3–es.

McSherry, F. D. (2009). Privacy integrated queries: an extensible platform for privacy-preserving data analysis. In *SIGMOD'09*, pages 19–30.

Mendes, Y., de Oliveira, D., and Ströele, V. (2020). Polyflow: a polystore-compliant mechanism to provide interoperability to heterogeneous provenance graphs. *J. Inf. Data Manag.*, 11(3).

Nargesian, F., Zhu, E., Miller, R. J., Pu, K. Q., and Arocena, P. C. (2019). Data lake management: Challenges and opportunities. *Proc. VLDB Endow.*, 12(12):1986–1989.

Proserpio, D., Goldberg, S., and McSherry, F. (2014). Calibrating data to sensitivity in private data analysis: A platform for differentially-private analysis of weighted datasets. *PVLDB*, 7(8):637–648.

Ramos, L. F. M. and Silva, J. a. M. C. (2019). Privacy and data protection concerns regarding the use of blockchains in smart cities. In *ICEGOV'2019*, page 342–347, Melbourne, Australia. ACM.

Sweeney, L. (2002). k-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(05):557–570.

Warner, S. L. (1965). Randomized response: A survey technique for eliminating evasive answer bias. *Journal of the American Statistical Association*, 60(309):63–69.