Estudo Experimental da Estimativa de Cardinalidade para Autojunções

Alexandre T. Enokida[1], **Eduardo H. M. Pena**[1]

[1]Universidade Tecnológica Federal do Paraná (UTFPR) – Campo Mourão, PR – Brazil

alexandretolomeotti@alunos.utfpr.edu.br, eduardopena@utfpr.edu.br

Resumo. *A estimativa de cardinalidade é um dos principais fatores que contribuem para a geração de planos de consulta subótimos. A literatura recente tem se concentrado na estimativa para consultas do tipo seleção-projeção-junção, enquanto a estimativa de cardinalidade para consultas com autojunção permanece uma área ainda pouco explorada. Consultas com autojunção são amplamente utilizadas em tarefas de limpeza de dados, como a validação de restrições de negação. Nesse contexto, investigamos empiricamente as principais técnicas de estimativa de cardinalidade para consultas com autojunção sob diferentes distribuições de dados. Nossos resultados mostram que estruturas de dados probabilísticas compactas (*sketches*) produzem estimativas consistentemente melhores do que modelos de custo tradicionais baseados em suposições.*

Abstract. *Cardinality estimation is a major contributor to suboptimal query plans. The recent literature has been concentrating on estimation for select-project-join queries, while cardinality estimation for self-join queries remains a largely unexplored area. Self-join queries are widely used in data cleaning tasks, such as validating denial constraints. In this context, we empirically investigate the main cardinality estimation techniques for self-join queries across various data distributions. Our results show that compact probabilistic data structures (sketches) consistently yield better estimates than traditional cost models based on assumptions.*

1. Introdução

Otimização de consulta é uma tarefa inerente realizada por Sistemas de Gerenciamento de Banco de Dados (SGBDs) durante o processamento de uma consulta, cujo objetivo é encontrar o plano de execução ótimo (ou, ao menos, próximo do ótimo) para a consulta em questão. Encontrar um bom plano é crucial para a execução eficiente de uma consulta, tendo em vista que planos subótimos reduzem o desempenho (i.e., tempo de resposta) em algumas ordens de magnitude [Leis et al. 2018].

O *System R* foi o primeiro SGBD a utilizar do modelo de custo [Leis et al. 2015], introduzido por [Selinger et al. 1979]. Em alto nível, uma consulta é tipicamente decomposta em blocos de consulta lógica e existem inúmeras sequências de execução desses blocos. Uma das ideias-chave apresentadas no *System R* foi a utilização de um modelo de custo que utiliza estatísticas das relações, armazenadas no catálogo do sistema, para estimar o custo de cada sequência. Um dos principais desafios do modelo de custo ocorre quando essas estatísticas não estão disponíveis. Nesse caso, SGBDs recorrem à

suposições simplificadoras sobre a distribuição dos dados. Essas suposições representam limitações do modelo devido à ausência de alternativas mais precisas [Moerkotte 2024].

A literatura apresenta uma variedade de trabalhos que propõem soluções capazes de manter essas estatísticas de forma escalável. Por exemplo, [Flajolet et al. 2007] apresentam uma estrutura probabilística compacta (isto é, utiliza pouco espaço) para estimar o número de valores distintos de um conjunto de dados de grande volume. [Cormode and Muthukrishnan 2005] introduzem uma estrutura compacta para estimar a frequência de valores distintos (contagem). Além disso, observamos uma tendência que grande parte dos estudos recentes têm se concentrado em consultas do tipo SPJ (select-project-join) (e.g., [Leis et al. 2018]) e em consultas de agrupamento (e.g., [Moerkotte 2024]). Apesar da relevância dessas consultas, autojunções permanecem pouco exploradas, criando uma lacuna na compreensão dos limites dos algoritmos do estado da arte aplicados a esse tipo de consulta.

Consultas de autojunção têm ampla aplicação prática, especialmente em tarefas de limpeza de dados [Pena et al. 2021]. Um exemplo atual e relevante é o uso de *Denial Constraints* (DCs), que modelam regras de qualidade de dados. Cada DC define a negação de uma condição que, quando satisfeita por um par de tuplas em uma base de dados, indica uma violação da integridade semântica dos dados, revelando erros ou inconsistências. SGBDs já implementam algumas restrições, como `UNIQUE` e `NOT NULL`. No entanto, o modelo relacional tradicional não abrange todas as formas possíveis de restrições semânticas. Por exemplo, em uma relação fictícia `funcionario`, uma regra implícita pode ser: "entre dois funcionários do mesmo departamento, o funcionário com data de admissão anterior deve possuir salário superior ao daquele com data de admissão posterior". Esse tipo de regra pode ser expressa por uma DC, representada por predicados sobre atributos da relação e implementada com uma consulta SQL com autojunção para identificar possíveis violações.

Um dos principais custos nos algoritmos do estado-da-arte para a verificação de DCs é, em geral, correlacionada ao número de valores distintos do atributo que participa do predicado (e.g., [Pena et al. 2021, Liu et al. 2024]). No trabalho de [Alon et al. 1999], os autores apresentaram um algoritmo para estimar o tamanho de uma junção combinando os tamanhos de suas respectivas autojunções. Em específico, a solução estima o tamanho da autojunção por meio do k-ésimo momento da frequência sobre uma distribuição. Dado que o artigo foi publicado há mais de duas décadas, levantamos a seguinte questão: não seria o momento de revisitar esse problema?

O restante deste trabalho está organizado da seguinte forma: na Seção 2 realizamos uma breve revisão sobre otimização de consultas e estimativa de cardinalidade, e descrevemos aplicações de estruturas de dados probabilísticas em SGBDs para armazenamento de estatísticas. Na Seção 3 detalhamos os materiais e métodos. Apresentamos os resultados obtidos na Seção 4. Por fim, realizamos nossas conclusões na Seção 5.

2. Otimização de Consultas, Estimativa de Cardinalidade e Estatísticas

Tradicionalmente, existem duas estratégias[1] de otimização de consultas: (i) heurísticas; e (ii) busca baseada em custo. O uso de heurísticas é uma otimização lógica

[1]A literatura recente de otimização de consultas apresenta abordagens baseadas em aprendizagem de máquina. No entanto, neste trabalho, consideramos apenas abordagens tradicionais baseadas no *System R*.

[Moerkotte 2025] que consiste na reescrita de expressões do plano lógico por meio das regras de equivalência da álgebra relacional. A segunda estratégia é uma otimização física que consiste em utilizar um modelo de custo para estimar o custo (tempo) de diferentes planos de execução de uma consulta, escolhendo ao final aquele que apresenta o menor custo estimado. Método de acesso, operador físico (algoritmo) e estatísticas são alguns dos aspectos utilizados para estimar custo. Resumidamente, o modelo de custo utiliza estimativas de cardinalidade dos operadores relacionais (isto é, o número de tuplas emitidas) como entrada para gerar planos de execução semanticamente equivalentes. A estimativa de cardinalidade é obtida através da seletividade do predicado — a fração de tuplas da relação que qualificam o predicado. Para isso, estatísticas dos atributos das tabelas (por exemplo, cardinalidade e frequência de valor) são armazenadas no catálogo do sistema.

Quando estatísticas não estão disponíveis, SGBDs recorrem às tradicionais suposições introduzidas pelo *System R* [Selinger et al. 1979]: uniformidade, independência e princípio da inclusão. A suposição de uniformidade assume que os valores de um atributo são uniformemente distribuídos. A suposição de independência supõe a ausência de correlação entre atributos. O princípio da inclusão estabelece que, em uma junção, a relação com menor domínio da operação terá uma tupla correspondente no outro domínio para cada valor distinto. No restante deste trabalho, referimos *Simple Profile* aos procedimentos de estimativa seminais introduzidos no *System R*.

Estimativa de Junção. Sejam $R(\text{A}, ...)$ e $S(\text{B}, ...)$ duas relações, a estimativa da junção $R \bowtie_{A=B} S$ pode ser generalizada pela Equação 1.

$$\frac{n_R \times n_S}{max(V(\text{B},\ S),\ V(\text{A},\ R))} \tag{1}$$

, onde n_R e n_S são o tamanho das relações R e S, respectivamente; e $V(\text{B},\ S)$ e $V(\text{A},\ R)$ o número de valores distintos dos atributos B em S e A em R, respectivamente.

Por limitações de escopo, omitimos a descrição das demais estimativas. O leitor interessado pode consultar qualquer livro-texto sobre sistemas de banco de dados (e.g., [Silberschatz et al. 2020]) para uma descrição detalhada sobre outras estimativas.

Histogramas. SGBDs utilizam diversas técnicas para armazenar estatísticas sobre as relações. Devido à sua simplicidade, histogramas são uma das técnicas mais utilizadas para representar a distribuição da frequência de valores distintos [Ioannidis 2003]. Entretanto, manter um histograma para cada atributo pode ser custoso, pois a quantidade de colunas armazenadas cresce linearmente com a cardinalidade do atributo. Para mitigar esse custo, SGBDs utilizam diversas alternativas baseadas em *binning*, como histogramas de classe de mesma profundidade.

***Sketches*.** *Sketches* são estruturas de dados probabilísticas que armazenam propriedades sobre um grande volume de dados com custo de espaço substancialmente menor ao conjunto de dados original. Pelo fato de serem compactas, não fornecem respostas exatas. *Sketches* são uma das abordagens mais recentes para armazenamento de estatísticas que são amplamente adequadas para SGBDs [Cormode et al. 2012].

***HyperLogLog*.** É uma estrutura de dados probabilística quase ótima — alcança uma estimativa precisa enquanto utiliza assintoticamente a menor quantidade de memória necessária para o problema—, voltada à estimativa do número de elementos distintos (car-

dinalidade) para conjuntos de dados de grande volume. Foi inicialmente proposta para lidar com problemas relacionados a fluxos de dados [Flajolet et al. 2007], mas também é altamente adequada para aplicações em SGBDs.

Count-min Sketch. Diferentemente do *HyperLogLog*, o *Count-Min Sketch* é utilizado para estimar a frequência de um determinado elemento. A estrutura é implementada através de uma matriz A de dimensão $d \times w$, onde cada linha d possui uma função *hash* independente que mapeia um valor de entrada para $v \in \{1, ..., w\}$. As células da matriz são contadores, inicializados em 0, e são incrementados quando elementos mapeados pela função resultam em sua posição. Computa-se, para qualquer elemento inserido, suas respectivas posições sobre todas as linhas da matriz, i.e., um elemento passa por d funções *hash*. A estimativa de frequência de um valor x é dada por $min\ A_{j,h_j(x)},\ j \in [1, d]$. O valor mínimo é escolhido pois a célula com o menor valor representa a estimativa com o menor "ruído" — isto é, onde houveram menos colisões.

Primordialmente, o *Count-min Sketch* foi projetado para estimar a frequência de um determinado valor distinto. Entretanto, no trabalho seminal de [Cormode and Muthukrishnan 2005], os autores descrevem um procedimento para estimativa de produto interno entre duas estruturas *Count-min Sketch*. O produto interno entre dois fluxos de dados é equivalente à estimativa de tamanho de junção.

JoinSketch. O *JoinSketch* é a *sketch* mais recente estudada neste trabalho. Introduzida por [Wang et al. 2023], os autores descrevem que *sketches* como *Count-min Sketch* possuem baixo desempenho em dados reais que possuem alta assimetria. A partir disso, o *JoinSketch* é projetado com base na suposição de que dados do mundo real apresentam poucos valores frequentes e muitos valores infrequentes. Trabalhos anteriores apresentaram soluções com base nessa suposição, entretanto, nenhum obteve uma solução que exigisse apenas uma varredura sobre o conjunto de dados. Quando a distinção entre valores frequentes e infrequentes é realizada corretamente, a variância da estimativa do produto interno é reduzida substancialmente.

Resumidamente, a estrutura do *JoinSketch* é composta por três componentes: (i) uma tabela *hash* aumentada para contagem de valores frequentes; (ii) uma tabela *hash* aumentada para contagem de valores "intermediários"; e (iii) um *Fast-AGMS Sketch* para contagem de valores infrequentes. Durante a varredura, o segundo componente mantém a contagem dos valores que ainda não se enquadraram como frequentes ou infrequentes. Quando a contagem de um valor nessa parte ultrapassa um limiar T, o valor é adicionado no componente de valores frequentes e removido do componente intermediário. Uma descrição detalhada da estrutura pode ser encontrada em [Wang et al. 2023].

3. Materias e Métodos

3.1. Conjuntos de Dados

Para a realização do experimento, selecionamos conjuntos de dados de diferentes origens: *TPC-H*; *Internet Movie Database* (IMDb); e dados sintéticos gerados a partir do artefato deste trabalho. Essa diversidade é importante em nosso contexto, pois *benchmarks* frequentemente utilizados em experimentos com SGBDs (e.g., *TPC-H*) apresentam a inconveniência de terem seus dados uniformemente distribuídos [Freitag and Neumann 2019].

Trabalhos anteriores [Leis et al. 2015, Leis et al. 2018] apresentam que a estima-

tiva de cardinalidade é, possivelmente, o fator primordial do modelo de custo. Consequentemente, se as estimativas são substancialmentes distantes do valor real, o otimizador pode selecionar uma solução não ótima (ou distante da ótima). Nessa perspectiva, utilizar um conjunto de dados uniformemente distribuídos pode mascarar o desempenho relativo sobre estimadores em conjuntos de dados do mundo real. Isso ocorre devido às suposições de uniformidade, independência, e princípio da inclusão. Adicionalmente, conjuntos de dados do mundo real frequentemente apresentam distribuições assimétricas e correlações entre os atributos [Leis et al. 2018], propriedades que os dados sintéticos do *benchmark* não conseguem replicar. Inspirado em [Leis et al. 2018], selecionamos o conjunto de dados IMDb para ilustrar uma instância de dados do mundo real em nossos experimentos.

Empiricamente, [Lee et al. 2023] demonstrou que existe uma alta variabilidade no impacto das estimativas em diferentes cargas de trabalho. Isso decorre de uma escassez de consultas que sejam capazes de evidenciar a distribuição enviesada e a correlação dos atributos. Com base nisso, levantamos a hipótese de que um conjunto de dados com parâmetros configuráveis (e.g., família de distribuições, média, desvio padrão, etc.) pode facilitar a elaboração de consultas que enfatizem a distribuição assimétrica ou a correlação dos atributos. A necessidade de controlar parâmetros sobre a distribuição dos atributos motivou o desenvolvimento de um gerador de dados sintéticos, disponível como artefato[2] deste estudo. Utilizamos três conjuntos de dados gerados pela ferramenta, todos ilustrando um mesmo cenário de uma tabela `funcionarios`. A tabela possui três atributos: (i) `age`; (ii) `salary`; e (iii) `department`. Os dois primeiros atributos seguem, para cada conjunto de dados, distribuições uniforme, normal e Zipf. O último atributo segue uma distribuição uniforme nos três conjuntos de dados. A Tabela 1 apresenta os predicados utilizados. Esse estudo se concentra em *equi-joins*.

Tabela 1. Predicados utilizados nos experimentos

Conjunto de dados	Predicado	Símbolo
Distribuição Normal	$t.age = t'.age$	φ_1
	$t.age = t'.age$ AND $t.department = t'.department$	φ_2
	$t.department = t'.department$	φ_3
Distribuição Uniforme	$t.age = t'.age$	φ_4
	$t.age = t'.age$ AND $t.department = t'.department$	φ_5
	$t.department = t'.department$	φ_6
IMDb (Movies)	$t.company\_id = t'.company\_id$ AND $t.company\_type\_id = t'.company\_type\_id$	φ_7
IMDb (Person)	$t.person\_id = t'.person\_id$ AND $t.info\_type\_id = t'.info\_type\_id$	φ_8
TPC-H	$t.l\_quantity = t'.l\_quantity$ AND $t.l\_partkey = t'.l\_partkey$ AND $t.l\_shipmode = t'.l\_shipmode$	φ_9
	$t.l\_quantity = t'.l\_quantity$	φ_{10}
Distribuição Zipf	$t.age = t'.age$	φ_{11}
	$t.age = t'.age$ AND $t.salary = t'.salary$	φ_{12}

[2] `https://anonymous.4open.science/r/a59abd-28D8`

3.2. Métricas

Utilizamos o *q-error* para avaliar a distância entre as estimativas e os valores reais. O *q-error* é uma métrica de erro amplamente utilizada (e.g., [Leis et al. 2015, Leis et al. 2018, Lee et al. 2023, Moerkotte 2024]) para avaliar técnicas de estimativa de cardinalidade.

Uma das principais características do *q-error*, quando comparado com outras métricas, é a garantia de correlação entre seu resultado e a qualidade do plano de execução gerado pelo otimizador. Em [Moerkotte et al. 2009], os autores demonstram que o custo do melhor plano de execução é obtido desde que o *q-error* não ultrapasse um limiar. Consequentemente, é crucial que as técnicas de estimativa busquem minimizar o *q-error*. A métrica é obtida por meio da Equação 2.

$$\text{q-error}(x, \hat{x}) = \begin{cases} 1 & \text{se } x = 0 \wedge \hat{x} = 0 \\ max(\frac{x}{\hat{x}}, \frac{\hat{x}}{x}) & \text{se } x \neq 0 \wedge \hat{x} \neq 0 \\ \infty & \text{caso contrário} \end{cases} \tag{2}$$

3.3. Estimadores

Na Seção 2, apresentamos aplicações de *sketches* para estimativas de diferentes operadores (cardinalidade, seleção e junção). Nesta seção, introduzimos o *Tug-of-War*, *sketch* seminal de [Alon et al. 1999] projetada para estimativa de cardinalidade de junções baseada na estimativa de autojunções.

Para estimar o tamanho de uma junção, os autores propuseram uma solução que utiliza a segunda frequência dos momentos. Suponha um fluxo de dados $\sigma = a_1, a_2, a_3, ..., a_m$ onde $a_i \in \{1, ..., n\}$. Também assuma um vetor de frequência $f = (f_1, f_2, ..., f_n)$ definido sobre o fluxo σ. Cada f_i representa o número de ocorrências de a_i em σ. Com base nisso, o k-ésimo momento da frequência é definido pela Equação 3.

$$F_k = \sum_{i=1}^{n} f_i^k \tag{3}$$

Observe que: (i) o momento zero F_0 representa a cardinalidade do fluxo σ; (ii) o primeiro momento F_1 representa o tamanho do fluxo σ; e (iii) o segundo momento F_2 representa o tamanho da autojunção do fluxo σ sobre um operador de comparação equitativo ($=$).

Construir vetores de frequência sobre grandes volumes de dados é uma tarefa computacionalmente custosa. Para isso, [Alon et al. 1999] propuseram uma solução baseada em *sketches*. O pseudoalgoritmo 1 sumariza a ideia por trás da solução.

Algoritmo 1: *Tug-of-War Sketch*

input : Conjunto de Dados: $x_1, x_2, ..., x_n$.
output: Estimativa de F_2.
1 seja $h : [n] \rightarrow \{\pm 1\}$ pertencer a uma família de funções *hash* universal 4-independente $\mathcal{H}$;
2 $z \leftarrow 0$;
3 **for** i *to* n **do**
4 $\quad z \leftarrow z + h(i)$;
5 **return** z^2;

Uma das limitações sobre o *Tug-of-War* é sua alta variância, que pode resultar em estimativas de baixa acurácia. Para mitigar o problema, os autores realizaram a execução de múltiplas instâncias paralelas, combinada com a técnica de estimativa "*mediana das médias*". A técnica consiste em: (i) particionar as variáveis aleatórias (i.e., cada z^2) em grupos de mesmo tamanho; (ii) computar a média de cada grupo; por fim, (iii) obter a mediana desses valores.

4. Experimentos

Os experimentos deste trabalho foram executados em um *desktop* equipado com um processador Intel Core i5-10400F (6 núcleos, 12 *threads*, 2.9 GHz, 12 MB cache), 16 GB de memória RAM DDR4-2666 e um dispositivo de armazenamento NVMe SSD de 1 TB. A sequência de execução dos experimentos segue a enumeração dos predicados utilizados (veja Tabela 1). Cada algoritmo foi executado cinco vezes sobre cada predicado; a média aritmética das execuções foi utilizada como resultado.

Simple Profile. Na Seção 3 descrevemos as famílias de distribuições adotadas para os atributos dos conjuntos de dados sintéticos. Especificamente, a coluna `age` foi avaliada sob distribuições normal e uniforme (predicados φ_1 e φ_4). Adicionalmente, a coluna `l_quantity` do *TPC-H* (predicado φ_{10}) também apresenta uma distribuição uniforme.

Os resultados confirmam o comportamento esperado: em distribuições uniformes, a suposição tradicional de uniformidade produz estimativas precisas. Em contraste, conforme a distribuição se afasta da uniformidade e o número de predicados aumenta, o *Simple Profile* tende a subestimar a cardinalidade em algumas ordens de magnitude, como observado nos predicados $\varphi_7 - \varphi_9$, φ_{11} e φ_{12}.

Tug-of-War. Para o *Tug-of-War*, utilizamos cinco grupos com 512 contadores, totalizando aproximadamente 50 KB de memória alocada para a estrutura. Alguns resultados obtidos foram atípicos: o *Simple Profile* produziu estimativas substancialmente mais precisas — em alguns casos, por várias ordens de magnitude — do que o *Tug-of-War* em todos os predicados. Esse comportamento é inesperado, pois a *sketch* deveria apresentar menor sensibilidade à assimetria da distribuição do que o *Simple Profile*, especialmente para distribuições não uniformes. Por fim, os valores dos *q-errors* obtidos (Tabela 2) evidenciam a elevada variância apresentada pelo *Tug-of-War*.

Count-min Sketch. Para o *Count-min Sketch*, selecionamos os valores 24 e 5000, correspondentes a d e w, respectivamente, resultando em aproximadamente 468 KB de memória alocada.

O *Count-min Sketch* teve estimativas significativamente melhores que o *Tug-of-*

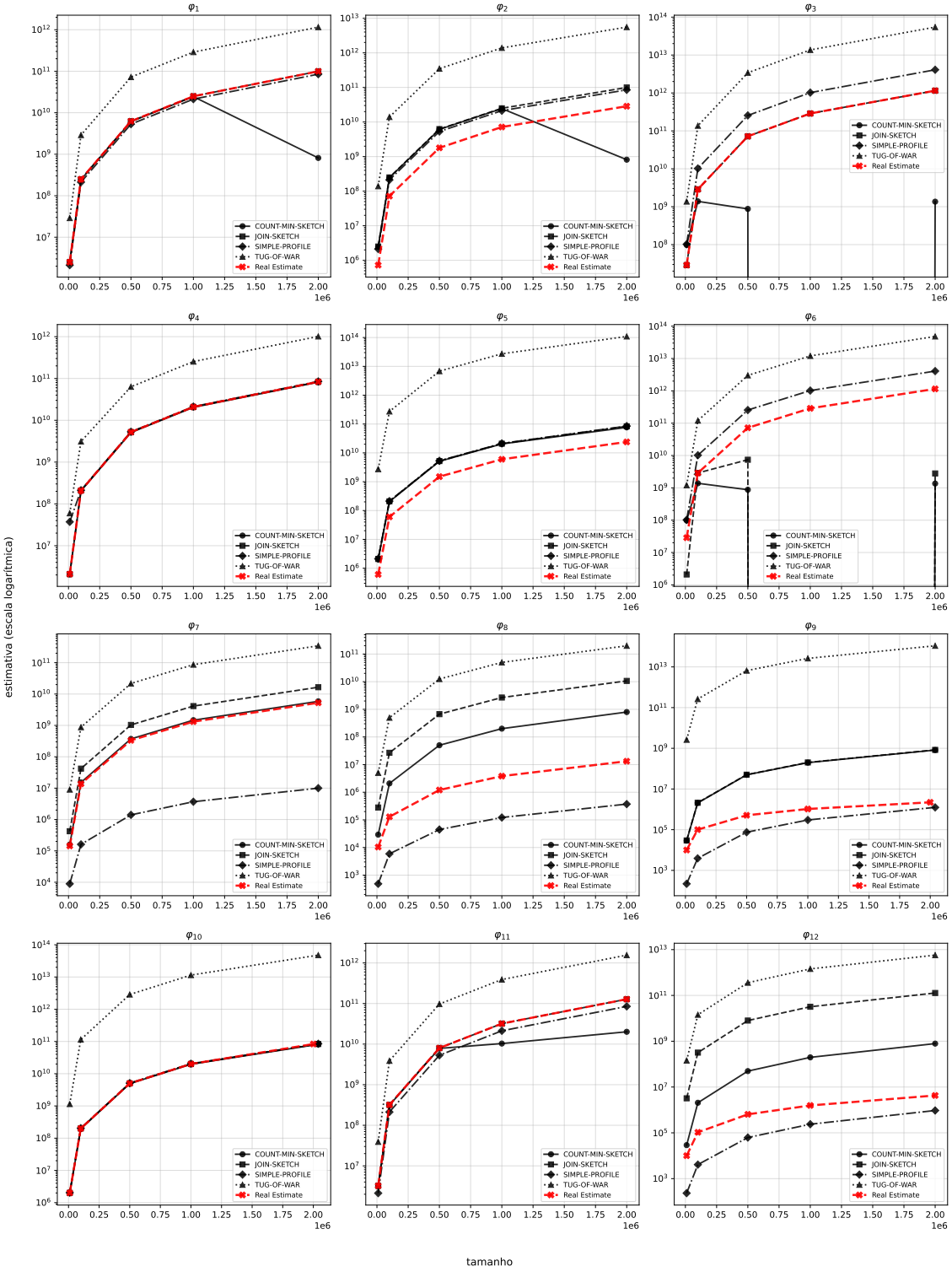

Figura 1. Estimativas dos algoritmos sobre os predicados $\varphi_1 - \varphi_{12}$.

War em diversos predicados. Além disso, para dados reais (IMDb), o *Count-min Sketch* apresentou o melhor desempenho na consulta φ_7. A Figura 1 apresenta os resultados. No entanto, o *Count-min Sketch* foi a estrutura que mais apresentou problemas. Diversos resultados inesperados foram observados nos predicados executadas nos conjuntos de dados

sintéticos. Os predicados φ_{1-3}, φ_6 e φ_{11}, por exemplo, apresentaram uma redução abundante na estimativa no aumento de tamanho. Esse comportamento é inesperado, dado que o *Count-min Sketch* é um estimador enviesado que sempre superestima os valores reais. O *JoinSketch* apresentou comportamento similar apenas na consulta φ_6.

***JoinSketch*.** Seguindo a mesma abordagem de [Wang et al. 2023], utilizamos 80 KB para a estrutura em nossos experimentos. O *JoinSketch* apresentou o melhor desempenho geral, pois obteve as estimativas mais precisas (com exceção de φ_8 e φ_{12}). Os resultados de estimativa são consistentes com os achados de [Wang et al. 2023]. Especificamente, o *Count–min Sketch* requer substancialmente mais memória do que o *JoinSketch* para atingir uma precisão comparável na estimativa do produto interno. Em nossos experimentos, essa tendência se manteve em quase todos os predicados: o *Count–min Sketch* utilizou quase seis vezes mais memória que o *JoinSketch*, mesmo produzindo estimativas menos precisas.

	Simple Profile	*Tug-of-War*	*Count-min Sketch*	*JoinSketch*
φ_1	1.19	11.15	24.80	**1.0**
φ_2	**2.90**	186.22	9.67	3.47
φ_3	3.48	45.64	183.09	**1.00**
φ_4	4.28	15.36	**1.00**	**1.00**
φ_5	3.46	4360.27	**3.42**	**3.42**
φ_6	**3.47**	40.21	183.15	87.34
φ_7	249.41	62.01	**1.13**	3.07
φ_8	**27.98**	8216.62	35.26	458.40
φ_9	1.70	1.73	**1.38**	**1.38**
φ_{10}	**1.00**	549.43	**1.00**	**1.00**
φ_{11}	1.53	11.69	2.45	**1.00**
φ_{12}	18.45	571276.72	83.65	13224.21

Tabela 2. Média do *q-error* sobre os tamanhos: 10K, 100K, 500K, 1M e 2M.

4.1. Limitações

Este estudo apresenta limitações. As consultas φ_1, φ_2, φ_3, φ_6 e φ_{11} exibem uma queda acentuada nas estimativas do *Count-min Sketch* entre 500 mil e 2 milhões de registros. O *JoinSketch* apresentou comportamento similar apenas na consulta φ_6. Outro resultado inesperado foram as estimativas obtidas da consulta φ_2: o *Simple Profile* teve o melhor desempenho. Esse resultado é atípico pois o *Simple Profile* realiza a suposição de uniformidade; e os valores dos atributos do predicado φ_2 formam uma distribuição normal. De forma análoga, consideramos os resultados de φ_{11} e φ_{12} inesperados.

Atribuímos os problemas dessas consultas sobre às seguintes hipóteses: (i) a quantidade de memória alocada para as estruturas não foram suficientes para conjuntos de dados maiores, resultando em estouro aritmético. Por exemplo, para predicados homogêneos (i.e., comparação sobre o mesmo atributo entre tuplas distintas, como φ_1), o *Count-min Sketch* realiza a estimativa selecionando o menor valor da soma dos quadrados de cada linha — uma célula com valor 46340 é suficiente para causar um estouro de inteiro de 32 bits; (ii) as funções *hash* utilizadas provocaram muitas colisões, saturando

contadores (i.e., alcançando o valor máximo para o tipo de dado do contador) e causando estouro aritmético no cálculo da estimativa; e (iii) erros na geração dos conjuntos de dados sintéticos, uma vez que as consultas φ_7 e φ_8, obtidas a partir do mundo real, produziram os resultados mais precisos.

5. Conclusões

Este trabalho teve como objetivo investigar experimentalmente o desempenho de diferentes técnicas de estimativa de cardinalidade aplicadas a consultas de autojunção, uma área ainda pouco explorada na literatura de otimização de consultas. Demonstramos que, em conjuntos de dados do mundo real que apresentam não uniformidade e assimetria, métodos baseados em *sketches*, como o *JoinSketch*, superam consideravelmente o modelo tradicional baseado nas suposições. Por fim, uma direção para trabalhos futuros é investigar técnicas alternativas para estimar predicados de intervalo.

Agradecimentos

Este estudo foi realizado com apoio da Fundação Araucária de Apoio ao Desenvolvimento Científico e Tecnológico do Estado do Paraná e do Instituto Serrapilheira, concedido por meio do Programa de Apoio a Jovens Cientistas (Protocolo n° SER2023371000001 / PI 28/2023).

Referências

Alon, N., Gibbons, P. B., Matias, Y., and Szegedy, M. (1999). Tracking join and self-join sizes in limited storage. In *Proceedings of the Eighteenth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '99, page 10–20, New York, NY, USA. Association for Computing Machinery.

Cormode, G., Garofalakis, M., Haas, P. J., and Jermaine, C. (2012). Synopses for massive data: Samples, histograms, wavelets, sketches. *Found. Trends Databases*, 4(1–3):1–294.

Cormode, G. and Muthukrishnan, S. (2005). An improved data stream summary: the count-min sketch and its applications. *Journal of Algorithms*, 55(1):58–75.

Flajolet, P., Fusy, É., Gandouet, O., and Meunier, F. (2007). Hyperloglog: the analysis of a near-optimal cardinality estimation algorithm. *Discrete mathematics & theoretical computer science*, (Proceedings).

Freitag, M. and Neumann, T. (2019). Every row counts: Combining sketches and sampling for accurate group-by result estimates. *ratio*, 1:1–39.

Ioannidis, Y. (2003). The history of histograms (abridged). In *Proceedings of the 29th International Conference on Very Large Data Bases - Volume 29*, VLDB '03, page 19–30. VLDB Endowment.

Lee, K., Dutt, A., Narasayya, V., and Chaudhuri, S. (2023). Analyzing the impact of cardinality estimation on execution plans in microsoft sql server. *Proc. VLDB Endow.*, 16(11):2871–2883.

Leis, V., Gubichev, A., Mirchev, A., Boncz, P., Kemper, A., and Neumann, T. (2015). How good are query optimizers, really? *Proc. VLDB Endow.*, 9(3):204–215.

Leis, V., Radke, B., Gubichev, A., Mirchev, A., Boncz, P., Kemper, A., and Neumann, T. (2018). Query optimization through the looking glass, and what we found running the join order benchmark. *The VLDB Journal*, 27(5):643–668.

Liu, Z., Deep, S., Fariha, A., Psallidas, F., Tiwari, A., and Floratou, A. (2024). Rapidash: Efficient detection of constraint violations. *Proc. VLDB Endow.*, 17(8):2009–2021.

Moerkotte, G. (2024). Cardinality estimation for having-clauses. *Proc. VLDB Endow.*, 18(1):28–41.

Moerkotte, G. (2025). Building query compilers. University of Mannheim. `https://pi3.informatik.uni-mannheim.de/~moer/querycompiler.pdf`.

Moerkotte, G., Neumann, T., and Steidl, G. (2009). Preventing bad plans by bounding the impact of cardinality estimation errors. *Proc. VLDB Endow.*, 2(1):982–993.

Pena, E. H. M., de Almeida, E. C., and Naumann, F. (2021). Fast detection of denial constraint violations. *Proc. VLDB Endow.*, 15(4):859–871.

Selinger, P. G., Astrahan, M. M., Chamberlin, D. D., Lorie, R. A., and Price, T. G. (1979). Access path selection in a relational database management system. In *Proceedings of the 1979 ACM SIGMOD International Conference on Management of Data*, SIGMOD '79, page 23–34, New York, NY, USA. Association for Computing Machinery.

Silberschatz, A., Korth, H. F., and Sudarshan, S. (2020). *Database System Concepts, Seventh Edition*. McGraw-Hill Book Company.

Wang, F., Chen, Q., Li, Y., Yang, T., Tu, Y., Yu, L., and Cui, B. (2023). Joinsketch: A sketch algorithm for accurate and unbiased inner-product estimation. *Proc. ACM Manag. Data*, 1(1).