Algorithmic Complexity Attacks on All Learned Cardinality Estimators: A Data-centric Approach

Yingze Li, Xianglong Liu, Dong Wang, Zixuan Wang, Hongzhi Wang*, Kaixin Zhang
Harbin Institute of Technology
China
{23B903046,23S003029,202111049,2023113027}@stu.hit.edu.cn,wangzh@hit.edu.cn,21B903037@stu.hit.edu.cn

ABSTRACT

Learned cardinality estimators have great potential in predicting query cardinalities. However, these learned methods, without exception, demonstrate fragility to drifts within the training data, thereby posing significant risks for real-world deployment. In this work, we are the first to investigate how minimal data-level drifts can maximally compromise the accuracy of learned estimators. To this end, we introduce the data-centric algorithmic complexity attack against all learned estimators in a black-box setting. We establish that finding the optimal data-centric attack strategy is NP-Hard. Thus, we designed a specialized approximation algorithm that achieves near-optimal solutions within polynomial-time, offering a $(1-\kappa)$ approximation ratio. Comprehensive experiments validate the effectiveness of our approach. On renowned open-sourced cardinality estimation workloads such as STATS-CEB and IMDB-JOB, our attack algorithm modifies only 1% of the tuples within the entire database, which results in three orders of magnitude increase in the 90-th percentile Qerror for all estimators and an increase in end-to-end processing time by up to 20 times.

PVLDB Reference Format:
Yingze Li, Xianglong Liu, Dong Wang, Zixuan Wang, Hongzhi Wang*, Kaixin Zhang. Algorithmic Complexity Attacks on All Learned Cardinality Estimators: A Data-centric Approach. PVLDB, 14(1): XXX-XXX, 2020.
doi:XX.XX/XXX.XX

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/LiYingZe/DACA.

1 INTRODUCTION

Cardinality Estimation (CE) plays a crucial role in query optimization within database systems, as it predicts the number of tuples that satisfy a query without execution and aids the database optimizer select the best query execution plan with the lowest algorithmic time complexity [20], thus improving system efficiency[22, 43]. Compared to traditional estimators [33, 41], learned cardinality estimators offer more accurate estimations and have therefore garnered extensive attention in recent years [24, 29, 36, 52]. These learned approaches are primarily categorized into three types: (1) Data-driven approaches [24, 52], which learn joint data distributions to enable accurate and generalized estimations that are robust to changes among query distributions. (2) Query-driven approaches [21, 29], which employ regression models to directly map query representations to cardinalities. (3) Hybrid approaches [36, 39, 44], which combine the information from both data and queries to provide a comprehensive prediction.

Despite the proven superior performance of learned estimators and their potential to replace traditional methods, the robustness and security concerns with these learned approaches have garnered widespread attention [22, 36, 39, 55]. In fact, similar concerns arise not only in the CE domain but also across numerous learned database topics [30, 50, 57]. When the training data are poisoned [30, 50, 55, 57] or the testing data are inconsistent with the training data [36, 39], the performance of these learned models can degrade catastrophically. Therefore, studying and understanding how to bring the most extreme performance degradation to these learned DB components with adversary input, known as Algorithm Complexity Attack (ACA) on these learned database components, can help us develop stronger preventive measures within learned databases and avoid the worst-case scenario [50, 57].

Query-centric ACA have been devised to compromise the performance of query-driven CE models. PACE [55], proposed by Zhang et al., involves adding noise to query workloads and poisoning the query-driven estimators. This degrades the accuracy of query-driven models and slows down the end-to-end performance. Although query-centric attacks can significantly impair the performance of query-driven estimators, we find that they have limited impact on widely used data-driven learned estimators [24, 28, 37, 42, 45, 46, 51, 52, 60]. This is primarily because data-driven methods utilize data-level information that is independent of historical workloads to achieve robust cardinality estimation across diverse testing query distributions. Consequently, these methods can effectively withstand poisoning attacks that target historical workloads. Therefore, query-centric ACA methodology is not suitable for decreasing the performance of these prevalent data-driven models.

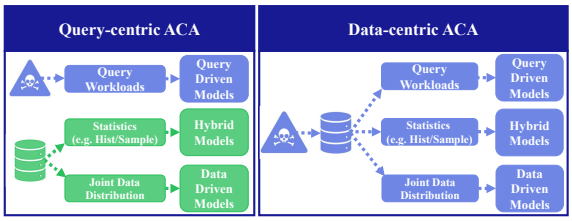


Figure 1: Query-centric ACA (Left) vs. Data-centric ACA (Right): Green Indicates Clean, Blue Indicates Poisoned

Proceedings of the VLDB Endowment, Vol. 14, No. 1 ISSN 2150-8097.
doi:XX.XX/XXX.XX

* Corresponding author.

Additionally, changes in training data can influence the performance of nearly all cardinality estimators [31, 32, 35, 36, 42]. Fundamentally, all existing learned estimators are learning from statistical information based on their training data sources [27], such as cardinality labels from historical workloads [21, 29], or joint data distributions within the training dataset [24, 45, 51]. When the training data is perturbed, the resulting statistical information naturally changes, causing all models trained on these altered statistics to incorporate erroneous information [32, 36]. This observation has prompted us to explore the feasibility of designing a **data-centric black-box ACA that slightly changes training data and compromises the performance of all cardinality estimators.**

However, initiating a successful and stealthy black-box ACA via a data-centric approach involves overcoming the following key challenges: (1)**Black-box Limitations.** The attacker lacks information about the currently deployed cardinality estimator and does not know whether it is MLP [21, 52], CNN [29, 39], Transformer [36, 51], or Bayes Network [45, 46]. This uncertainty restricts the attacker's options and prohibits the attacker from exploiting the existing white-box attack strategies designed for compromising specific machine learning models [30, 50, 55, 57]. (2)**Tractability Awareness.** The attacker has to foresee whether the current attack can inflict maximal damage within polynomial-time. This information is crucial for deciding whether to launch an attack. (3) **Limited Budget.** How can a limited number of attack operations on the training data maximally degrade the accuracy of models trained on this poisoned dataset? On the one hand, excessive data modification operations are likely to reduce the stealthiness of the attack and reveal the attacker's presence. On the other hand, intuitively, small-scale changes at the data level appear to have an insignificant impact on cardinality estimators. For instance, many studies have demonstrated that stale learned models can tolerate certain data-level changes exceeding 30% while still maintaining excellent estimation performance [24, 31, 43].

According above discussions, the toughest challenge lies in that the attacker has no information about the estimator deployed internally within the black box, which hinders the attacker from effectively quantifying and boosting the efficiency of the attack. To address this issue, we discovered a crucial and often overlooked fact: learned estimators are approaching oracle-level accuracy on training datasets, obtaining accurate cardinality. For example, many learned estimators [24, 36, 42, 46, 52, 60] achieve more than 90% of the Qerrors close to 1. However, this primary advantage also renders these models with a common weakness. Despite their varying model types, the predictive output behavior of each model tends to be similar to the oracle's behavior within the training dataset.

Therefore, we can leverage this exposed vulnerability to address the aforementioned challenge. Instead of directly tackling the black-box attack head-on and lacking the necessary information to solve the problem, we can indirectly construct a finite set of data updates designed to poison the oracle in the training database, and thereby impact the performance of all learned estimators. Thus, to overcome the first black-box challenge, we utilize the oracle based on the training data as our surrogate model and convert the difficult black-box attack problem into a tuple selection problem with much more information. To tackle the second tractability challenge, we formulate this data-centric attack as a combinatorial optimization problem and establish that this problem is NP-Hard. To achieve the most effective attack within a limited budget to address the third challenge, we analyze the supermodular properties of the optimization objective under specific conditions and design a $(1-\kappa)$-approximation algorithm for efficient resolution.

Contributions: We make the following contributions.

C1: **Black-box Attack on All Learned Estimators**: We investigate how to launch a successful Data-centric Algorithm Complexity Attack (DACA) that compromises the accuracy of all learned CE models in a black-box setting (§ 3.2).

C2: **Intractability Insights**: We demonstrate that, under a limited budget, finding the optimal attack strategy is NP-Hard, even when only considering deletion operations (§ 3.3).

C3: **Near-Optimal Attack strategy**: We design a polynomial-time approximation algorithm (§ 4.2) that guarantees a $(1-\kappa)$ approximation for the optimal attack strategy (§ 4.3).

C4: **Common Security Vulnerabilities**: We validate the effectiveness of our attack through extensive experiments (§ 5). Our experimental findings reveal that altering the data distribution in a single database by merely 1% is sufficient to severely compromise various learned cardinality estimators trained on this poisoned database. It causes a three-orders-of-magnitude increase in all estimators' 90-th percentile Qerror. The compromised learned estimator can mislead the query optimizer to select the execution operator with the highest algorithmic time complexity. Consequently, this leads to up to a 20× increase in end-to-end processing time, exposing a common vulnerability in nearly all learned cardinality estimators.

2 PRELIMINARIES

We introduce the learned estimators in § 2.1, the algorithmic complexity attack of learned DBMS in § 2.2 and establish the threat model in § 2.3.

2.1 Learned Cardinality Estimation

Suppose a database D consists ℓ relations, i.e. $D = \{R_1, R_2, \dots R_\ell\}$. Given a query $\mathbf{q}$, when $\mathbf{q}$ is executed on D, we obtain $\mathbf{q}$'s results in result set $\mathbb{S}(\mathbf{q}|D)$. We denote the cardinality of $\mathbb{S}(\mathbf{q}|D)$ as $C_D(\mathbf{q})$.

Tuple's Joint Weight: For relation R_x with N tuples, a tuple $t_i \in R_x$ and a query $\mathbf{q}_j$, the joint weight w_{ij} denotes tuple t_i's contribution to query $\mathbf{q}_j$'s result's cardinality, that is $w_{ij} = \#|\mathbb{S}(\mathbf{q}_j|D - R_x + \{t_i\})|$. $\mathbf{q}_j$'s cardinality can be seen as a linear summation of R_x's tuple's the joint weights: $C_D(\mathbf{q}_j) = \sum_{i=1}^{N} w_{ij}$.

Learned Cardinality Estimation: Learned cardinality estimation needs to make predictions on $C_D(\mathbf{q})$ using learned estimator E without executing $\mathbf{q}$ on D. E is trained via the information of D. Based on the training information, these learned cardinality estimators can be categorized into these main paradigms: data-driven, query-driven and hybrid.

Learned Data-driven Estimator: Learned data-driven cardinality estimator E_{Data} learns the joint distribution of database D. Based on the learned statistical information, E_{Data} provides the query independent estimation of the query $\mathbf{q}$'s cardinality $est(\mathbf{q}|E_{Data})$.

Learned Query-driven Estimator: Learned query-driven cardinality estimator E_{Query} learns the mapping from historical workloads Q to cardinalities C, i.e., $Q \rightarrow C$. Based on the learned mapping knowledge, E_{Query} provides the estimation of the query $\mathbf{q}$'s cardinality $est(\mathbf{q}|E_{Query})$.

Hybrid estimator: Learned hybrid estimator E_{Hybrid} combines query Q with statistics (e.g. histograms [36], samples [29]) $\mathcal{S}$ as the input features, and learns the mapping $\{Q \times \mathcal{S}\} \rightarrow C$. Based on the enhanced features, E_{Hybrid} provides a comprehensive estimation of query $\mathbf{q}$'s cardinality $est(\mathbf{q}|E_{Hybrid})$ compared to $est(\mathbf{q}|E_{Query})$.

For query $\mathbf{q}_j$, we follow the assumptions made in the majority of CE literature [22, 28, 29, 36, 51], considering $\mathbf{q}_j$ to be the most prevalent selection-projection-join query, represented in Eq. 1:

$$\texttt{SELECT * FROM } \mathbf{q}_j.Tabs \texttt{ WHERE } \mathbf{q}_j.Joins \texttt{ AND } \mathbf{q}_j.Filters; \quad (1)$$

where: $\mathbf{q}_j.Tabs$ denotes the set of tables involved in the query $\mathbf{q}_j$, such that $\mathbf{q}_j.Tabs \subseteq D$, and $\mathbf{q}_j.Joins$ represents the join predicates, $\mathbf{q}_j.Filters$ signifies the filter predicates.

2.2 Algorithmic Complexity Attack

Algorithmic Complexity Attack (ACA) is a variant of the Denial-of-Service Attack [9, 12, 19, 40]. The attacker introduces a small amount of adversarial perturbations into the target system and changes its computational complexity, forcing it to utilize all available computational resources. Such an attack paradigm is widely used to evaluate the robustness of learned DB components under worst-case scenarios [30, 50, 55]. An example is the ACA on the dynamic learned index [50], Kornaropoulos et al. designed a specific adversary index insertion strategy that increased the memory overhead and query time complexity of the learned index and caused an out-of-memory error with only a few hundred adversarial insertions.

Fundamentally, the database optimizer ***decreases*** the computational complexity of physical plans by utilizing cost estimates provided by estimators and applying relevant rewrite rules [20]. However, if the estimator is compromised by misleading training information, it may mislead the optimizer to select the worst execution plan, thereby ***increasing*** the time complexity of the physical plan. Targeting this vulnerability, PACE [55] achieves a query-centric ACA by contaminating the historical query workload, thereby affecting query-driven estimators. However, PACE is unable to influence data-driven estimators, as these models rely on data features that are independent of the historical workload for their predictions, rendering them unaffected by the poisoned historical workload. Surprisingly, we find that data-centric ACA simultaneously impacts nearly all cardinality estimators and greatly increases the computational complexity of physical plans, as demonstrated in the example below:

Example 2.1. *Figure 2 shows one possible Data-centric ACA on learned estimators by **deleting a single tuple from the training dataset.** The attacker first removes a single tuple from R_x in the training data(Step 1). After the training procedure is accomplished(Step 2), the attacker inserts the deleted tuple back to restore the database(Step 3), which leaves the actual data in the database unchanged and keeps the attacker undiscovered. However, this data-centric attack compromised all the training information involving $R_x.ID = 2$, whether the cardinality labels in historical queries, sampling points within a sampling pool, or joint data distributions in databases, will all be aligned with a cardinality label of 0. Thus, almost all estimators fall into the attacker's trap and predict a cardinality of 0 for queries containing predicates on $R_x.ID = 2$(Step 4). Incorrect cardinality estimates distort the estimated costs, causing the optimizer to erroneously assume that the cost of a Nested Loop join is lower than that of a Hash Join(Step 5). Consequently, the optimizer opts for the Nested Loop operator(Step 6), which severely amplifies the query's theoretical time complexity and physical execution time. Figure 2 (right) shows one possible consequence on JOB-Q60, the complexity of the query operator degrades from Hash Join's $O(M + N)$ to Nested Loop's $O(M \times N)$, and the execution time increases 9.2×10^4 seconds.*

2.3 Threat Model

In this section, we will establish the threat model, define the adversary's goals, their knowledge, and introduce the metrics that will be used to assess the effectiveness of the attack on learned estimators.

Adversary's goal: The attacker's goal is to craft limited data updates ΔD into the original database D, making it $D' = D + \Delta D$, so that the model trained on the poisoned database D' shall provide estimations with extremely low precision, misleading the optimizer to choose the physical plan with the highest time complexity.

Adversary's knowledge: In this work, we study the black-box attack on learned cardinality estimators. The attacker has no information about the estimator used in the database, except that it is a learned estimator trained on database D'. The attacker is unaware of the specific parameter details of the model and does not know whether it is data-driven or query-driven. Meanwhile, apart from the testing workloads W_{Test} and their results $\mathcal{R}es = \{\$(\mathbf{q}_j|D), \mathbf{q}_j \in W_{Test}\}$, the attacker does not know any other data distribution within this database.

Adversary's Capacity: We assume that an attacker can access the testing workloads W_{Test} and their results $\mathcal{R}es$. Furthermore, the attacker is capable of performing projection and group-by operations on $\mathcal{R}es$ to obtain the joint weights w_{ij} of each tuple t_i in table R_x that participates in the testing query $\mathbf{q}_j$. Additionally, we assume that the attacker can execute at most K update operations ΔD (insertions/deletions) on the specific relation R_x within the training database, poisoning the training database into $D' = D + \Delta D$. After the model has finished training on D', the attacker can perform inverse operations $-\Delta D$ on R_x to restore the poisoned database D' into the original clean state D, thereby poisoning the learned models and remaining stealthy.

Attack Evaluation Metric. The attacker has to devise specific strategies to significantly impair these commonly used metrics: (1) **Qerror metric**: Defined as $Q(est(\mathbf{q}|E), C_D(\mathbf{q})) = max(\frac{est(\mathbf{q}|E)}{C_D(\mathbf{q})}, \frac{C_D(\mathbf{q})}{est(\mathbf{q}|E)})$, where $est(\mathbf{q}|E)$ is the estimators E's prediction on given query $\mathbf{q}$. It measures the distance between the estimated cardinality $est(\mathbf{q}|E)$ and the true cardinality $C_D(\mathbf{q})$ of a query. (2) **End-to-end latency**: Measures the total latency taken for generating a plan with estimated cardinalities and executing the physical plan. This metric is essential for demonstrating how a CE method can improve query optimization performance. It serves as a gold standard for assessing the effectiveness of CE approaches [22, 28, 36, 45].

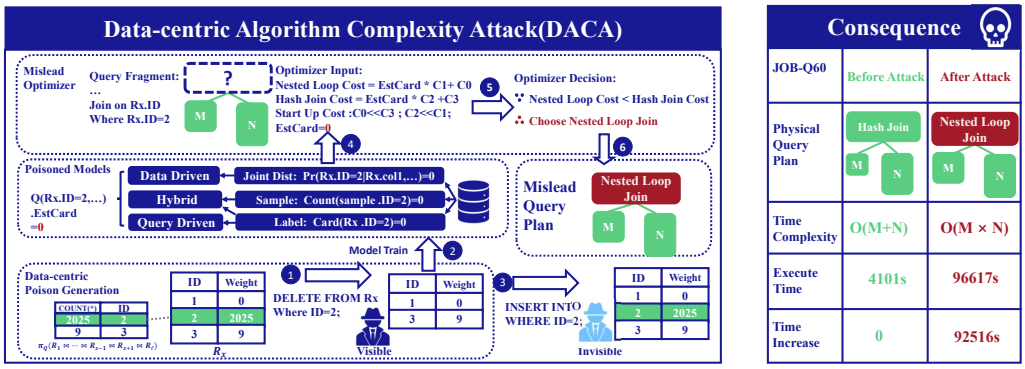


Figure 2: DACA workflow(Left); The consequence of DACA(Right);

3 BLACK-BOX ATTACK AGAINST ALL LEARNED ESTIMATORS

In this section, we will apply the concepts of algorithmic complexity attack and threat model established in § 2 to define the black-box data-centric algorithmic complexity attacks in § 3.1. We will appropriately transform this problem into a constrained integer nonlinear programming problem in § 3.2 and analyze its computational intractability in § 3.3.

3.1 Black-box Data-centric ACA

In this section, we provide the original definition of black-box data-centric ACA and introduce the challenge associated with directly solving this black-box problem. We assume that an attacker can execute at most K tuple-level data modifications ΔD on the training database instance D. Thereby poisoning the training data into $D' = D + \Delta D$. The attacker aims to ensure that, when evaluated on the testing workload $W_{Test} = \{\mathbf{q}_j | 1 \le j \le M\}$ with M queries, the estimator E, trained on the poisoned database D', has the worst accuracy on the original database D, i.e., to maximize Eq. 2:

$$\text{Max}: \sum_{j=1}^{M} Q(est(\mathbf{q}_j|E), C_D(\mathbf{q}_j)) = \sum_{j=1}^{M} \max\left(\frac{est(\mathbf{q}_j|E)}{C_D(\mathbf{q}_j)}, \frac{C_D(\mathbf{q}_j)}{est(\mathbf{q}_j|E)}\right) \tag{2}$$

$$\text{s.t. } (1)D' = D + \Delta D \quad (2)E \text{ is trained on } D' \quad (3)\#|\Delta D| \le K$$

While maximizing Eq. 2, we consider the complete black-box attack scenario, meaning that the attacker does not know the type of estimator used internally in the database. In other words, besides knowing that E is trained on D', the attacker does not know whether E is data-driven, query-driven, or a hybrid estimator. Meanwhile, the black-box setting also blocks the attacker from knowing the deployed model's details and introduces numerous challenges for attackers, significantly restricting the attacker's visibility. The internally adopted models could be MLP [21, 52], CNN [29, 39], Transformer [36, 51], or Bayes Network [45, 46]. Therefore, attackers cannot utilize existing attack methodology targeting white-box learned models in ML security [11, 15, 16, 26, 48, 49, 56]. This makes it seem impossible to launch the black-box data-centric ACA directly, we need to find some other angles to transform the problem.

3.2 Attack the surrogate oracle

In this section, we will take a different perspective to transform the seemingly impossible black-box ACA problem from the previous section into a constrained integer nonlinear programming problem that can be tackled. Specifically, we are inspired by the experimental results of learned CE studies [22, 36, 43, 51, 52], which show that existing learned estimators possess extremely high estimation accuracy, with their estimates on the training database approaching oracle-level. Therefore, we utilize an oracle estimator E_O on the dataset D' as our surrogate model. We assume that the E_O trained on D' can precisely report the actual cardinality of queries in D'. For E_O, we have $est(\mathbf{q}|E_O) = C_{D'}(\mathbf{q})$. Thus, our task now is to maximize the Qerror of the oracle trained on D':

$$\text{Max}: \sum_{j=1}^{M} \max\left(\frac{est(\mathbf{q}_j|E_O)}{C_D(\mathbf{q}_j)}, \frac{C_D(\mathbf{q}_j)}{est(\mathbf{q}_j|E_O)}\right) = \sum_{j=1}^{M} \max\left(\frac{C_{D'}(\mathbf{q}_j)}{C_D(\mathbf{q}_j)}, \frac{C_D(\mathbf{q}_j)}{C_{D'}(\mathbf{q}_j)}\right) \tag{3}$$

$$\text{s.t. } (1)D' = D + \Delta D \quad (2)\#|\Delta D| \le K$$

Given the test workload W_{Test} and a relation R_x of N tuples, denoted as $R_x = \{t_1, t_2, \ldots, t_N\}$. We consider using $t_i.\beta$ to represent the attacker's behavior on tuple t_i, where $t_i.\beta$ takes -1 to indicate delete tuple t_i from the R_x, $t_i.\beta$ takes 0 to indicate no operation on tuple t_i, and $t_i.\beta$ takes k to indicate duplicating tuple t_i by k times and re-inserting them into the database. Therefore, for the poisoned database D', the cardinality of query $\mathbf{q}_j$ on D' equeals to $C_{D'}(\mathbf{q}_j) = C_D(\mathbf{q}_j) + \sum_{i=1}^{N} t_i.\beta \times w_{ij}$. Based on the above, we reorganize the attack problem in Eq.2 into:

Optimal Data-centric ACA (Optimal DACA) problem: Given testing workloads W_{Test}, the attacker needs to provide an attack strategy and maximize the following Eq. 4:

$$\text{Max}: \sum_{j=1}^{M} \max\left(\frac{C_D(\mathbf{q}_j) + 1 + \sum_{i=1}^{N} t_i.\beta \times w_{ij}}{C_D(\mathbf{q}_j) + 1}, \frac{C_D(\mathbf{q}_j) + 1}{C_D(\mathbf{q}_j) + 1 + \sum_{i=1}^{N} t_i.\beta \times w_{ij}}\right) \tag{4}$$

$$s.t. \sum_{i=1}^{N} |t_i.\beta| \le K \quad t_i.\beta \in \{-1, 0, 1, \ldots K\}$$

For the sake of brevity, we employ the abbreviation 'optimal DACA' to represent 'optimal Data-centric Algorithmic Complexity Attack' throughout the remainder of this paper. Meanwhile, we add 1 to both the numerator and denominator when calculating Qerror to avoid division by zero errors. This transformation is a commonly seen evaluation technique in many cardinality estimation methods and can be found in many open-sourced cardinality estimators' GitHub repositories e.g. line 4 in [5], line 21-22 in [6].

In summary, we converted the seemingly unsolvable black-box attack problem stated in Eq. 2 into a constrained integer nonlinear programming problem, as detailed in Eq. 4. However, it remains uncertain whether the attacker can effectively solve Eq. 4 in polynomial time. Therefore, we will provide an analysis of intractability in the next section.

3.3 Intractability

In this section, we give an intractability analysis of the optimal DACA problem defined in Eq.4. We conclude that finding the optimal attack strategy is NP-Hard even when considering deletions alone. Additionally, when considering both insertions and deletions, finding an optimal attack strategy still remains NP-Hard.

When only considering data deletions, the problem is converted into the following optimization problem in Eq. 5:

$$\text{Max:} \sum_{j=1}^{M} \frac{C_D(\mathbf{q}_j)+1}{C_D(\mathbf{q}_j)+1+\sum_{i=1}^{N} t_i.\beta \times w_{ij}} \tag{5}$$
$$s.t. \sum_{i=1}^{N} |t_i.\beta| \le K \quad t_i.\beta \in \{-1, 0\}$$

We now present a proof sketch that finds the optimal attack strategy in Eq. 5, is NP-Hard. We start our polynomial-time reduction from the known Densest K-Subgraph problem. Defined as:

Densest-K-Subgraph (DKS) problem: Given a simple, undirected graph $\mathbb{G} = (\mathbb{V}, \mathbb{E})$ and an integer K, find a subset of vertices $S \subseteq \mathbb{V}$ such that: $\#|S| \le K$ and the subgraph $\mathbb{G}[S]$ induced by S has the maximum number of edges.

The left subfigure in Figure 3 shows a DKS example when $K = 4$. The DKS problem is a well studied NP-Hard problem in graph theory [8, 17, 18] and closely related to many applications in graph database community such as community search [58, 59]. We will next construct a polynomial-time reduction based on this problem.

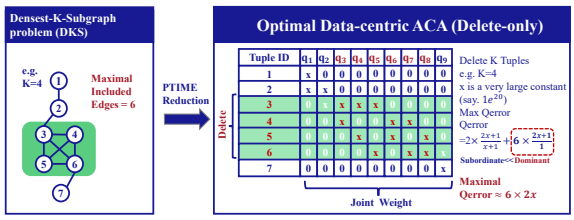


Figure 3: Polynomial-time reduction example starting from Densest-K- Subgraph(DKS) problem.

Theorem 3.1. *The optimal DACA problem defined in Eq. 5 when considering only deletions, is NP-Hard.*

Proof. (Sketch)We establish the NP-Hardness proof by providing a polynomial-time reduction from the DKS problem to the constrained maximization in Eq 5 that find K deleting tuples to maximize the Qerror.

For any DKS problem P_1, consisting of a graph $\mathbb{G} = (\mathbb{V}, \mathbb{E})$ and a number K, we can always construct a corresponding database instance and a group of testing queries to form the delete-only optimal DACA problem P_2 in Eq 5 in polynomial-time. It is constructed as follows:

(1) Each vertex $v_i \in \mathbb{V}$ corresponds to a unique tuple t_i in the relation R_x.
(2) Each edge $e_j = \{v_{i1}, v_{i2}\} \in \mathbb{E}$ corresponds to a query $\mathbf{q}_j$ involving only two tuples in R_x, t_{i1} and t_{i2}.
(3) We assign each tuple contributes a common join weight $\mathbf{x}$, where $\mathbf{x}$ is a sufficiently large positive integer.

For the constructed delete-only DACA problem P_2, and a given query $\mathbf{q}_j$ consisting of two tuples. If one of its tuples is deleted, the contribution to the objective function is $\Delta Qerr_1 = \frac{2\mathbf{x}+1}{\mathbf{x}+1}$, if two tuples are both deleted, the contribution is $\Delta Qerr_2 = 2\mathbf{x}+1$. When $\mathbf{x}$ is sufficiently large, $\Delta Qerr_1$ approaches 2 and $\Delta Qerr_2$ approaches $2\mathbf{x}$ which is significantly larger: ($\Delta Qerr_2 \gg \Delta Qerr_1$). Consequently, the maximization objective is predominantly influenced by $\Delta Qerr_2$. This implies that maximizing the objective function is effectively equivalent to maximizing $2\mathbf{x}$ times the number of edges in the K-densest subgraph of $\mathbb{G}$. To better illustrate the above ideas, we use example. 3.1 to show the reduction process.

Example 3.1. *Figure 3 gives a reduction example. The left side of the figure illustrates an instance of the DKS problem with $K = 4$. Correspondingly, on the right side, we have designed an optimal delete-only DACA problem for $K = 4$. For a graph comprising 7 vertices and 9 edges, we can construct a database instance containing a relation R_x with 7 tuples and 9 queries. For example, queries $\boldsymbol{q}_3$, $\boldsymbol{q}_4$, and $\boldsymbol{q}_5$ were designed to represent the connections between vertex 3 and vertices {4, 5, 6}. Each tuple contributes $\mathbf{x}$ to the involved query's cardinality. When $\mathbf{x}$ is sufficiently large (e.g., 10^{20}), the contribution to the Qerror from deleting one tuple for a single query is approximately 2, whereas deleting two tuples results in a contribution of approximately $2\mathbf{x}$. Consequently, the maximum Qerror is determined by scenarios where tuples from two queries are deleted. Furthermore, the maximum possible contribution to the Qerror is proportional to the number of internal edges within the K-vertex subgraph selected by the DKS. In our instance, the optimal solution to the DKS is a clique composed of four vertices, which contain six edges. Therefore, the maximum Qerror resulting from the delete-only DACA on the right side is determined by $6 \times 2\mathbf{x}$.*

Since the DKS problem is NP-Hard, and we have reduced it to the optimal delete-only DACA problem in polynomial-time, it follows that finding the optimal DACA problem when only considering deletions is NP-Hard. □

Based on the Theorem 3.1, we will demonstrate that considering both insertion and deletion simultaneously, the complete DACA problem is also NP-Hard.

Theorem 3.2. *The optimal DACA problem defined in Eq. 4 when considering both insertions and deletions, is NP-Hard.*

Proof. (Sketch): Our overall proof sketch is as follows: We extend the polynomial-time reduction constructed in Theorem 3.1. When the joint weight $\mathbf{x}$ is larger than $K \times M$, where K is the attack budget, and M is the number of queries, the benefits from I insertion operations($2 \leq I \leq K$) become negligible compared to delete two tuples within a given query $\mathbf{q}_j$. This leads the attacker to abandon insertion operations and transform the case into a delete-only environment, and we can use the remainder of Theorem 3.1's proof. Therefore, we establish a polynomial-time reduction from the DKS problem to the optimal DACA problem that considers both insertions and deletions, thereby proving that the optimal update strategy involving both insertions and deletions is NP-Hard. We provide rigorous proof in our appendix [7].

□

In summary, for an attacker, unless P=NP, it is not feasible to find an optimal solution for DACA problem in polynomial-time. This theoretical result may offer some reassurance to database administrators as offering an optimal DACA solution is not an easy task. However, we note that this does not imply that database systems employing learned cardinality estimators are inherently secure. As we will demonstrate in the next section, the attacker can devise polynomial-time approximation algorithms with tight approximation ratio guarantees to achieve near-optimal attack effectiveness within polynomial-time.

4 NEAR OPTIMAL ATTACK DEPLOYMENT

In the previous section, we obtained some bad news for the attacker that finding the optimal DACA strategy is NP-Hard even when the operations are limited to deletions. It means that no polynomial-time algorithm can efficiently find the optimal DACA strategy unless P=NP. Fortunately, the optimal DACA problem we are studying possesses certain favorable properties, which allow for the existence of polynomial-time algorithms capable of providing exact or approximately optimal attack strategies under specific conditions. In § 4.1, we will analyze these favorable properties. In § 4.2, we will utilize these special properties to design an efficient approximate algorithm. In § 4.3, we will analyze the algorithm in § 4.2.

4.1 Analysis of the DACA's objective function

In this section, we will analyze the nature of DACA's objective function under specific conditions to derive certain insights.

To facilitate the presentation of symbols in the following discussion, we denote the Qerror induced by the j-th query, considering only deletions, as: $\mathbb{Q}_j(X) = \frac{C_D(\mathbf{q}_j)+1}{C_D(\mathbf{q}_j)+1+\sum_{t_i \in X} t_i.\beta \times w_{ij}}$. X is the deleted tuples within the relation R_x, having $X \subseteq R_x, \forall t_i \in X, t_i.\beta = -1$. Therefore, the total Qerror when only deletion is considered is $\mathbb{Q}(X) = \sum_{j=1}^{M} \mathbb{Q}_j(X)$. Based on these definitions, we have Theorem. 4.1

Theorem 4.1. *When only deletion operations are considered, total Qerror $\mathbb{Q}(X)$ of the optimal DACA problem satisfies the supermodular property, that is:* $\forall A, B \subseteq R_x, \mathbb{Q}(A \cup B) + \mathbb{Q}(A \cap B) \geq \mathbb{Q}(A) + \mathbb{Q}(B)$.

Proof. (Sketch)We aim to demonstrate that the objective

$$\mathbb{Q}(X) = \sum_{j=1}^{M} \mathbb{Q}_j(X) = \sum_{j=1}^{M} \frac{C_D(\mathbf{q}_j) + 1}{C_D(\mathbf{q}_j) + 1 + \sum_{t_i \in X} t_i.\beta \times w_{ij}}$$

satisfies the supermodular property. Specifically, for any two sets A and B within R_x, we need to prove that $\forall A, B \subseteq R_x, \mathbb{Q}(A \cup B) + \mathbb{Q}(A \cap B) \geq \mathbb{Q}(A) + \mathbb{Q}(B)$.

First, consider each component function $\mathbb{Q}_j(X)$. We aim to show that $\mathbb{Q}_j(X)$ is supermodular. We denote

$$w'_{ij} = \frac{w_{ij}}{1 + C_D(\mathbf{q}_j)}, \quad S_j(X) = \sum_{t_i \in X} t_i.\beta \times w'_{ij}$$

Then, the expression

$$\mathbb{Q}_j(A \cup B) + \mathbb{Q}_j(A \cap B) - \mathbb{Q}_j(A) - \mathbb{Q}_j(B)$$

can be reorganized into the following form, we provide the detailed derivation in the appendix [7].

$$\frac{(2 + S_j(A) + S_j(B))(S_j(A \setminus B))(S_j(B \setminus A))}{(1 + S_j(A \cup B))(1 + S_j(A \cap B))(1 + S_j(A))(1 + S_j(B))}$$

Observe that given $\forall t_i \in X, t_i.\beta = -1, w'_{ij} \geq 0$, we have $S_j(X)$ is negative. We deduced that the two factors of the numerator, $S_j(A \setminus B)$ and $S_j(B \setminus A)$, are both less than or equal to 0. Thus, their product is greater than or equal to 0. Meanwhile, $\forall X \subseteq R_x, (1 + S_j(X)) \geq (1 + S_j(R_x)) = \frac{1}{1+C_D(\mathbf{q}_j)} > 0$. Therefore, each term in the denominator is strictly positive, and $(2 + S_j(A) + S_j(B))$ is strictly positive. In conclusion, within the above fractional, the numerator is greater than or equal to 0, while the denominator is strictly greater than 0. We have:

$$\mathbb{Q}_j(A \cup B) + \mathbb{Q}_j(A \cap B) - \mathbb{Q}_j(A) - \mathbb{Q}_j(B) \geq 0,$$

which confirms that $\mathbb{Q}_j(X)$ is supermodular.

Since $\mathbb{Q}(X)$ is the sum of supermodular functions $\mathbb{Q}_j(X)$, it inherits the supermodular property. Formally,

$$\begin{aligned} &\mathbb{Q}(A \cup B) + \mathbb{Q}(A \cap B) - \mathbb{Q}(A) - \mathbb{Q}(B) \\ &= \sum_{j=1}^{M} \left[\mathbb{Q}_j(A \cup B) + \mathbb{Q}_j(A \cap B) - \mathbb{Q}_j(A) - \mathbb{Q}_j(B)\right] \geq 0. \end{aligned}$$

Hence, the objective $\mathbb{Q}(X)$ satisfies the supermodular property. □

Next, we consider the scenario where only insertions are considered, the attack problem discussed in Eq. 4 can be much simpler. Similar to Theorem. 4.1, we define the Qerror induced by the j-th query, considering only insertions, as: $\mathbb{Q}'_j(X) = \frac{C_D(\mathbf{q}_j)+1+\sum_{t_i \in X} t_i.\beta \times w_{ij}}{C_D(\mathbf{q}_j)+1}$. And X is the set of tuples that are repeatedly inserted into R_x, having $X \subseteq R_x, \forall t_i \in X, t_i.\beta = 1$. Therefore, the total Qerror when only insertion is considered is $\mathbb{Q}'(X) = \sum_{j=1}^{M} \mathbb{Q}'_j(X)$. Based on these definitions, we have Theorem. 4.2

Theorem 4.2. *When only considering insertions, the optimal data-level attack problem satisfies the modular property, i.e.,* $\forall A, B \subseteq R_x, \mathbb{Q}'(A \cup B) + \mathbb{Q}'(A \cap B) = \mathbb{Q}'(A) + \mathbb{Q}'(B)$.

Proof. We leave the detailed proof in our appendix [7]. □

The above theorems indicate that although the optimal DACA problem in Eq. 4 is NP-Hard, the objective function possesses special properties when the attack operation types are limited. We will utilize this characteristic to design an effective approximate attack algorithm and analyze the attack's effectiveness.

4.2 Attack strategy generation

In this section, we will utilize the characteristics analyzed in the previous section to design a corresponding algorithm to maximize Eq. 4. We first develop the attack strategy as outlined in Algorithm 1:

Algorithm 1 Approximate solution to optimal DACA.

Require: Table R_x; Testing queries W_{Test}; Testing queries' results $\mathcal{R}es$; Budget K ;
Ensure: Modification set ΔD with at most K modifications;

1: $w = getJointWeight(\mathcal{R}es, R_x)$ ▷ Obtaining joint weight
2: $\Delta D = \phi; Mask = \mathbf{0}_N$
3: **while** $\#|\Delta D| \leq K$ **do**
4: $bestOp = \phi; bestVal = 0;$
5: **for** $(t_i \in R_x) \wedge (Mask[i] = 0)$ **do** ▷ Get R_x's non-deleted tuples
6: $gainD = \sum_{\mathbf{q}_j \in W_{Test}} Q(C_{D+\Delta D}(\mathbf{q}_j), C_{D+\Delta D}(\mathbf{q}_j) - w_{ij});$
7: **if** $gainD \geq bestVal$ **then**
8: $bestVal = gainD$
9: $bestOp = -t_i$
10: $gainI = \sum_{\mathbf{q}_j \in W_{Test}} Q(C_{D+\Delta D}(\mathbf{q}_j), C_{D+\Delta D}(\mathbf{q}_j) + w_{ij});$
11: **if** $gainI \geq bestVal$ **then**
12: $bestVal = gainI$
13: $bestOp = t_i$
14: **if** $bestOp = \phi$ **then**
15: $break$
16: **if** $bestOp = -t_\delta$ **then** ▷ Mask the deleted tuple
17: $Mask[\delta] = 1$
18: $\Delta D.append(bestOp);$
19: **return** $\Delta D;$

The general idea of the Algorithm 1 is as follows. In line 1, the attacker performs a group-by operation on the result $\mathcal{R}es$ of the testing queries W_{Test} over the relation R_x and calculates the joint weight of each tuple in R_x with respect to each query join. For a given query $\mathbf{q}_j$ in Eq. 1 and its materialized result in $\mathbb{S}(\mathbf{q}_j|D)$, this operation can be executed via the following SQL:

```
SELECT Rx.PK, COUNT(*) FROM S(qj|D) GROUP BY Rx.PK;
```

where $R_x.PK$ is the primary key of the relation R_x.

Lines 2 and 4 of the algorithm handle the initialization of variables. Lines 3 through 17 employ a greedy approach to sequentially select operations that maximize the current Qerror. Specifically, lines 5 to 13 iterate through the modified relation table one by one, computing the weights for both deletion and duplicate insertion for each tuple. In each iteration, the algorithm greedily selects the operation that maximizes the local Qerror. Lines 16-17 temporally saved the deleted tuples to prevent duplicate computation on already deleted tuples.

The aforementioned algorithm guarantees termination within polynomial-time. Specifically, the time complexity of the algorithm is $O(\#(\mathcal{R}es) + N \times K \times M)$, where: $\#(\mathcal{R}es)$ is the overhead of the group by operation in line 1 and is linear to the size of the result set $\mathcal{R}es$, N is the cardinality of the relation R_x, K is the budget allocated to the attacker for data insertion, and M is the size of the test query.

4.3 Analysis on the attack strategy

In this section, we evaluate the effectiveness of our attack strategy outlined in Algorithm 1. We demonstrate that the strategy can deliver optimal or near-optimal results in scenarios limited to either insertions or deletions. Additionally, we present two corollaries to showcase its performance in more general situations. Initially, we establish that when operations are confined to deletions, Algorithm 1 achieves a $1-\kappa$ approximation ratio.

Theorem 4.3. *In the scenario where only deletions are considered, let $\mathbb{Q}(O)$ denote the optimal attack result. Algorithm 1 can achieve an approximation of $(1-\kappa)$. This is, the output ΔD of the Algorithm 1 possesses the property that $\mathbb{Q}(\Delta D) \geq (1-\kappa)\mathbb{Q}(O)$, where $\kappa = 1 - \min_{t \in R_x} \min_{A,B \subseteq R_x - t} \frac{\mathbb{Q}(A+t)-\mathbb{Q}(A)}{\mathbb{Q}(B+t)-\mathbb{Q}(B)}$.*

Proof. (Sketch).

Based on the definition of κ, we obtain $\forall A, B \subseteq R_x, 1-\kappa \leq \min_{t \in R_x} \frac{\mathbb{Q}(A+t)-\mathbb{Q}(A)}{\mathbb{Q}(B+t)-\mathbb{Q}(B)}$. Furthermore, we let the set ΔD_i denote the set chosen by Algorithm 1 at the i-th step, and we let ΔD_i to substitute A. Similarly, we can arbitrarily choose a subset permutation of the optimal solution O and let a subset O_i with size i from the permutation to substitute B. We also define $t_{i1} = O_{i+1} - O_i$ and $t_{i2} = \Delta D_{i+1} - \Delta D_i$. By definition, we can derive that:

$$1 - \kappa \leq \frac{\mathbb{Q}(t_{i1} + \Delta D_i) - \mathbb{Q}(\Delta D_i)}{\mathbb{Q}(t_{i1} + O_i) - \mathbb{Q}(O_i)}$$

Thus, we obtain:

$$(\mathbb{Q}(t_{i1} + O_i) - \mathbb{Q}(O_i)) \times (1-\kappa) \leq \mathbb{Q}(t_{i1} + \Delta D_i) - \mathbb{Q}(\Delta D_i) \quad (6)$$

Given that the outer loop of the Algorithm 1 runs for K steps, we can apply the aforementioned operation at each step of the process. Consequently, we have:

$$(1-\kappa)\mathbb{Q}(O) = (1-\kappa)\mathbb{Q}(\emptyset) + (1-\kappa)\sum_{i=1}^{K} \mathbb{Q}(O_i + t_{i1}) - \mathbb{Q}(O_i).$$

Since $\mathbb{Q}(\emptyset) \geq 0$ and combining Eq. 6, we can conclude:

$$(1-\kappa)\mathbb{Q}(O) \leq \mathbb{Q}(\emptyset) + \sum_{i=1}^{K} \mathbb{Q}(t_{i1} + \Delta D_i) - \mathbb{Q}(\Delta D_i)$$

Finally, based on lines 6-9 of Algorithm 1 at each step, we have $(\mathbb{Q}(t_{i2} + \Delta D_i) - \mathbb{Q}(\Delta D_i)) \geq (\mathbb{Q}(t_{i1} + \Delta D_i) - \mathbb{Q}(\Delta D_i))$. Therefore:

$$\begin{aligned}(1-\kappa)\mathbb{Q}(O) &\leq \mathbb{Q}(\emptyset) + \sum_{i=1}^{K} (\mathbb{Q}(t_{i1} + \Delta D_i) - \mathbb{Q}(\Delta D_i)) \\ &\leq \mathbb{Q}(\emptyset) + \sum_{i=1}^{K} (\mathbb{Q}(t_{i2} + \Delta D_i) - \mathbb{Q}(\Delta D_i)) \\ &= \mathbb{Q}(\Delta D_K).\end{aligned}$$

This means that:

$$(1-\kappa)\times\mathbb{Q}(O)\leq\mathbb{Q}(\Delta D_K).$$

Thus, we prove that the approximation ratio of algorithm is $1-\kappa$. □

After considered the scenario where only deletions are made, we will continue to analyze our algorithm's performance in the context of insertion-only scenarios. Fortunately, we conclude that, when considering only insertions, our algorithm is capable of achieving the optimal solution constituted solely by insertions.

Theorem 4.4. *In the case of considering only insertions, the above greedy approach can reach an optimal insertion result.*

Proof. (Sketch). **Base Case ($K=1$):** When $K=1$, the algorithm iterates through all possible values and identifies the tuple that maximizes the gain function G for insertion. Therefore, the algorithm is optimal at this initial step. **Inductive Step ($K\geq 2$):** Assume that for $K-1$, the algorithm achieves optimality. We need to show that under this assumption, the algorithm also achieves optimality for K. Suppose that ΔD_{K-1} achieves the maximal Qerror and inserts $K-1$ tuples. Let the optimal insertion attack strategy select tuple t_a at the K-th step, while our Algorithm 1 selects t_b. According to lines 10-13 of Algorithm 1, following inequality holds:

$$\sum_{j=1}^{M}\frac{w_{bj}}{C_D(\mathbf{q}_j)+1}\geq\sum_{j=1}^{M}\frac{w_{aj}}{C_D(\mathbf{q}_j)+1}$$

This implies that $\mathbb{Q}'\left(t_b+\Delta D_{K-1}\right)\geq\mathbb{Q}'\left(t_a+\Delta D_{K-1}\right)$. Thus, if that optimality is achieved at step $K-1$, using lines 10-13 of the Algorithm 1 ensures that the algorithm remains optimal for step K. □

Although the two aforementioned theorems only preliminary considering insert-only or delete-only optimal DACA scenarios. We will now demonstrate, through the following corollaries, that in certain special cases, even when considering both insertions and deletions simultaneously, our algorithm can still achieve good approximation performance. The proof of these corollaries can be found in the appendix [7].

Corollary1: If an attacker can determine in advance for each query in the testing workloads whether it is primarily insertion or deletion based, this would mathematically eliminate the in the inner max layer of the Qerror summation in Eq. 4. Under this condition, ΔD of Algorithm 1 still possesses the following property: $\mathbb{Q}(\Delta D)\geq(1-\kappa)\mathbb{Q}(O)$.

Corollary 2: If the queries meet the following conditions: $M>K$, and for all queries $\mathbf{q}_j$, we have $\text{count}(w_{i,j}\neq 0)\leq 2$ and $C_D(\mathbf{q}_j)>K\times M$. Then, Algorithm 1 still possesses the following property: $\mathbb{Q}(\Delta D)\geq(1-\kappa)\mathbb{Q}(O)$.

The above theorem and corollary indicate that although the optimal DACA is NP-Hard, in certain cases, Algorithm 1 is still capable of providing a good attack strategy that approximates an optimal solution within polynomial time.

5 EXPERIMENT

In this section, we provide an experimental analysis of the effectiveness, scalability of the data-centric algorithmic complexity attack on learned estimators and its potential consequences. We will use our experiment results to answer the following questions:

1. Effectiveness: Can the proposed DACA techniques effectively compromise the performance of data-driven, query-driven, and hybrid estimators? (§ 5.2)

2. Scalability: How do variations in data size, query scale, and the selection of attacked tables influence the efficiency and effectiveness of the proposed DACA technique? (§ 5.3)

3. Consequence: How would the compromised estimator mislead the query optimizer into selecting suboptimal query plans? What are the characteristics of such plans? (§ 5.4)

5.1 Experimental Setup

Datasets: To evaluate the performance of attack methods in multi-table scenarios, we selected the widely adopted benchmarks IMDB-JOB and STATS-CEB for assessment, as they are extensively used in the evaluation of multi-table cardinality estimators [22, 28, 36, 39, 45, 54]. The IMDB-JOB benchmark [34] is based on the IMDB movie database [1] and comprises six relations (`cast_info, movie_info, movie_companies, movie_keyword, movie_info_idx, title`) and fourteen attributes, totaling 62,118,470 records. For the test workload, we selected the open-source JOB-light workload [2], which consists of 70 real queries and 696 subqueries. Additionally, STATS-CEB [22] is a collection of user-generated anonymized content from stack exchange network [3] includes eight relations (`users, posts, postLinks, postHistory, comments, votes, badges, tags`) and 43 attributes, comprising a total of 1,029,842 records. For the test workload, we selected the open-source workload verison [4] consisting of 146 queries and 2,603 subqueries. To train the query-driven model on these databases, we followed the approach of Li et al. [36], generating 2,000 corresponding training queries and their respective subqueries for training.

CE models: Based on various studies [36, 39, 45], we select the following most competitive and representative learned estimators covering the paradigms of data-driven, query-driven, and hybrid:

(1) *LWNN (Query-driven)* [21]: formulates the CE mapping function as a regression problem and uses MLPs to map query representations to cardinalities. We follow the instructions in [22] to extend LWNN to support joins.

(2) *MSCN (Query-driven)* [29]: the most famous learned query-driven method based on a multiset convolutional network model.

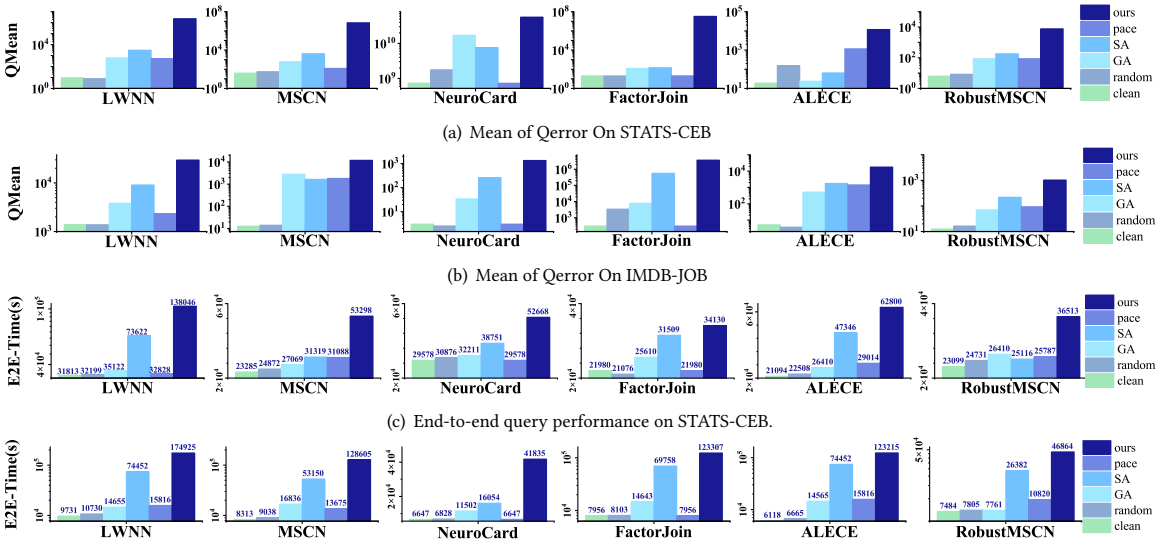
(3) *NeuroCard (Data-driven)* [51]: learns a single deep auto-regressive model for the joint distribution of all tables in the database and estimates the cardinalities using the learned distribution. Following the settings in [51], we set the sampling size of the NeuroCard to 8,000.

(4) *Factorjoin (Data-driven)* [45]: integrates classical join-histogram methods and learned Bayes Network into a factor graph.

(5) *ALECE (Hybrid)* [36]: uses a transformer to learn the mapping from query representation and histogram features to cardinality.

(6) *RobustMSCN (Hybrid)* [39]: is an improvement to the basic MSCN method [29], aimed at improving the robustness of the naive MSCN by employing masked training and incorporating data-leveled features such as PG estimates and join bitmaps.

(7) *Oracle*: is the oracle E_O which is capable of utilizing all information from the poisoned database D' to obtain query $\mathbf{q}$'s true cardinality $C_{D'}(\mathbf{q})$ within D'.



(a) Mean of Qerror On STATS-CEB

(b) Mean of Qerror On IMDB-JOB

(c) End-to-end query performance on STATS-CEB.

(d) End-to-end query performance on IMDB-JOB.

Figure 4: Mean of Qerrors and end-to-end(E2E) latency degradation per query on STATS/IMDB database

Baselines: We compare DACA with the estimators before any attacks (Clean). Meanwhile, we compare the four attack baselines as follows, we selected these baselines because they have either been used to reduce the performance demands on learned cardinality estimators or have been employed to solve similar NP-Hard database optimization problems [13, 25, 55].

(1) *PACE* [55]: is the State-Of-The-Art query-centric attack method on learned query-driven estimators. PACE compromises query-driven estimators by poisoning historical workloads.

(2) *Random Generation (Random)*: randomly selects tuples from the training database to delete or duplicate.

(3) *Simulated Annealing (SA)* [14]: constructs attack strategies via a probabilistic hill-climbing algorithm and maximizes Eq. 4.

(4) *Genetic Algorithm (GA)* [38]: treats a set of potential attack strategies as a gene, by randomly selecting several sets of possible attack strategies and using them as a population for genetic crossover and mutation, therefore maximizing Eq. 4.

Experimental Settings: *Estimators Settings*: For the cardinality estimators to be attacked, we train them on the poisoned database D' and test them on the clean database D. That is to say, the cardinality labels for the Qerror evaluation and the data used to evaluate the actual latency are both derived from the clean dataset D. For hybrid methods, we provide the data features of the clean database during testing. Specifically, we supply ALECE with histogram representations from the clean database D, and provide RobustMSCN with PG estimates and joint sampling features from the clean database. *Attack Settings*: In § 5.2, we target the `title` table of the IMDB-JOB database and the `posts` table of the STATS-CEB database. We set the data-centric attack budget to 20% of the attacked table. This means that in the IMDB-JOB database, the attacker performs 5×10^5 tuple leveled updates, altering a data scale of $\frac{5\times10^5}{6.2\times10^7} = 0.008$. Similarly, in the STATS-CEB database, the attacker conducts 1.8×10^4 update operations, changing a data scale of $\frac{1.8\times10^4}{1\times10^6} = 0.012$.

Metrics: We use the metrics defined in § 2.3 to evaluate the attack's effectiveness. We use Qerror to measure the degradation in estimation accuracy to evaluate the quality of the generated query plans, and end-to-end (E2E) time to assess the impact of estimation on the end-to-end execution of queries. We utilize the framework developed in [22] to inject the cardinalities predicted by the estimator into Postgres and evaluate the end-to-end execution time.

Environments: Our end-to-end experiments were conducted on an individual Huawei Cloud server with 4 Intel(R) Xeon(R) Platinum 8378A CPUs and 32GB RAM and 1T SSD. Apart from that, all remaining experiments were run on a server with 4 RTX A6000 GPUs, 40 Intel(R) Xeon(R) Silver 4210R CPUs, and 504GB RAM.

5.2 Decline of the Estimator's Performance

Average Decline of the Estimator's Qerror: The decline of the estimators' average Qerror can reflect the overall attack ability of the given attack strategy [55]. Meanwhile, it is also the main maximizing objective of PACE [55] and ours Eq 2, reflecting on the overall attack effectiveness. Figure 4(a) and 4(b) demonstrates the average Qerror of each estimator before (Clean) and after the attack on IMDB-JOB and STATS-CEB datasets. We can find that for models like LWNN, MSCN, and ALECE, the order of the attack effectiveness is Ours > SA > PACE > GA > random. On average, our approach outperforms the four baselines 6.85×, 10.47×, 15.27×, 1876× respectively. For the rest of the data-driven and hybrid models, the overall order of the attack effectiveness is Ours > SA > GA > PACE> random. On average, our approach outperforms the four baselines 4.92×, 26.9×, 225×, 293× respectively. We have discovered that PACE's effectiveness in launching attacks is limited to LWNN, MSCN, and ALECE. This is because PACE poisons historical queries, effectively corrupting the mapping that maps query representations to their cardinalities, or the knowledge of using these representations to look up values in a histogram-based lookup table. Other estimation methods utilize data-level knowledge that is independent of historical loads, such as attribute correlations (NeuroCard)

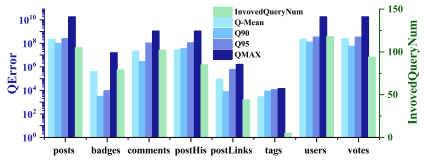
and PG estimation (RobustMSCN). These forms of knowledge enhance the robustness of the methods on poisoned queries, thereby rendering them immune to PACE's attacks. In contrast to PACE, our method, which originates from data-level attacks, can contaminate these data-level pieces of information, thereby impacting estimation methods that are data-driven, query-driven, and hybrid models. Meanwhile, we have found that our proposed attack strategy achieves better attack efficacy compared to SA/GA/Random. This corroborates the conclusions presented in § 4.3, that is, although optimizing Eq. 4 is generally NP-Hard, Algorithm 1 is capable of effectively identifying an approximate optimal solution.

Percentile Decline of the Estimator's Qerror: The percentage Qerrors can reflect the distribution of the estimator's errors in many perspectives, where the 50th percentile provides an alternative perspective on the overall estimation quality [52], and the high percentiles can easily affect the performance of database query optimization [53, 55]. Table 1 demonstrates the percentile Qerror of each CE model before and after being attacked on different datasets. We find that for the 50-th Qerror, our approach outperforms PACE, SA, GA, Random 8.28×, 2.43×, 6.47×, 47.1×, respectively. Meanwhile, on >90-th percentile Qerror, our approach also significantly outperforms PACE, SA, GA, Random 3-4 orders of magnitude, respectively. Additionally, we have included in Table 1 the attack performance of different methods against the oracle E_O in the training database. Our method has the most severe misleading effect on the oracle, theoretically capable of increasing the Qerror by 5 orders of magnitude for more than 50% of queries. Meanwhile, estimators with high accuracy are often more vulnerable to DACA attacks. For example, Neurocard, the best-performing model in IMDB, has suffered from a Qerror increase by 2 orders of magnitude for more than 50% of queries after being attacked by our method. This is because such models' behavior is closer to the oracle within the training data, making them more easily affected by DACA. This confirms the correctness of the transformation discussed in § 3.2.

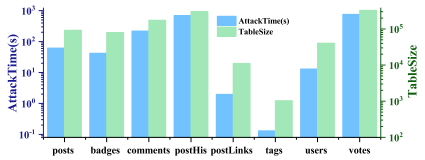
Decline of the E2E-Time. To thoroughly understand how different attack methods affect the query optimization pipeline integrated within a database, we measured the end-to-end execution time of various cardinality estimators subjected to different attack strategies. Figure 4(c) and Figure 4(d) illustrate the end-to-end performance degradation analysis for each query within STATS-CEB and IMDB-JOB. We observe that for each dataset and cardinality estimation model, our method effectively compromises the estimators, misleading the optimizer pipeline and resulting in the longest end-to-end execution times. In terms of execution time increments, our approach outperforms PACE, SA, GA, and Random on the IMDB-JOB and STATS-CEB datasets by factors of 24.2×, 2.21×, 17.5×, 202× as well as 11.6×, 2.34×, 10.3×, 41.8×, respectively. Additionally, we note that different cardinality estimators are variably affected by data-centric attacks concerning E2E-Time. Specifically, the order of susceptibility is as follows: LWNN > MSCN > NeuroCard > ALECE > FactorJoin > RobustMSCN. RobustMSCN is the least affected by attacks because this hybrid method leverages clean and robust features, which partially mitigate the impact of attacks. However, it is important to note that hybrid estimators can still be compromised. For example, after our attack, RobustMSCN's end-to-end time increases by 1.58×. The aforementioned end-to-end results demonstrate that our attack method can effectively contaminate the knowledge acquired by almost all estimators, thereby misleading the corresponding optimizers into selecting suboptimal execution plans and causing significant slowdowns in end-to-end performance.

Take away: Our DACA methodology effectively compromises the performance of all kinds of learned estimators. It successfully corrupted the knowledge acquired by different learned models, whether they are data-driven, query-driven, or hybrid models. Although our DACA methodology was originally designed to reduce the average Qerror. We are surprised to find out that it can also significantly diminish the model's worst-case prediction accuracy, misguiding the optimizer to generate suboptimal physical execution plans and resulting in multiple-fold increases in end-to-end latency. Furthermore, we observe that the data-level robustness features employed by the latest hybrid models can mitigate some of the attack's effects; however, the effectiveness of this mitigation remains to be enhanced.

5.3 Scalability Study



(a) Attack Effectiveness on different tables



(b) Attack Efficiency on different tables

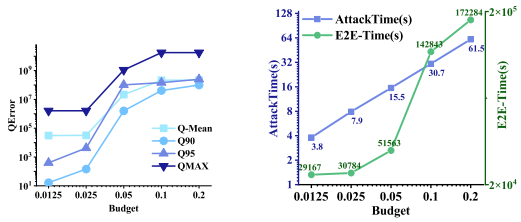
Figure 5: Scalability on different tables

Scalability On Attacked Table. To validate the theoretical attack effectiveness on different tables, in Figure 5(a) and Figure 5(b), we conducted experiments by switching the attacked tables within the STATS-CEB database. We selected the oracle E_O as the victim. The attack budget was set at 20% of the rows in the targeted table. Our findings indicate that, despite the varying number of queries associated with different tables in the STATS-CEB workload, our method can significantly compromise E_O for any table, resulting in a maximum Qerror ranging from 10^4 to 10^{10}. Additionally, we observed that the more queries a table encompasses, the greater the impact of the attack. For instance, when the number of queries increased from 10 (in `tags`) to 76 (in the `votes`), the maximum Qerror escalated from 10^4 to 10^{10}.

Scalability On Attacked Budget: To test the theoretical attack effectiveness on different attack budgets, we conducted experiments by switching the attack budget in the `posts` table of the STATS-CEB database from 1.25% to %20. We choose the E_O as the victim estimator. Figure 6(a) and 6(b) illustrate the effectiveness and duration of

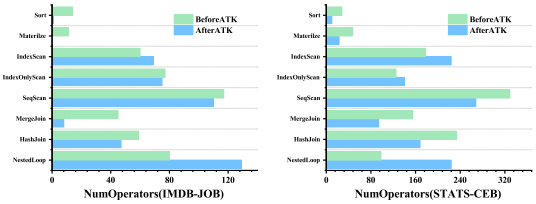
Table 1: Percentile Qerrors on STATS-CEB and IMDB-JOB

Paradigm	CE_Method	Attack_Method	IMDB					STATS				
			Q50	Q90	Q95	Q99	QMAX	Q50	Q90	Q95	Q99	QMAX
Query-Driven	LWNN	Clean	2.661	65.04	117.6	1567	$5.67 \cdot 10^4$	4.762	25.98	47.18	1182	$2.66 \cdot 10^4$
		Random	2.271	64.52	118.2	1595	$3.81 \cdot 10^4$	4.489	20.37	32.10	1505	$3.51 \cdot 10^4$
		GA	48.61	1105	5570	$8.41 \cdot 10^4$	$9.60 \cdot 10^4$	21.84	377.7	1167	9173	$5.52 \cdot 10^4$
		SA	184.1	$2.05 \cdot 10^4$	$4.21 \cdot 10^4$	$1.28 \cdot 10^5$	$1.91 \cdot 10^5$	39.87	1065	3847	$4.65 \cdot 10^4$	$3.67 \cdot 10^6$
		PACE	27.54	1359	6542	$4.79 \cdot 10^4$	$9.59 \cdot 10^4$	26.44	654.8	2486	$1.36 \cdot 10^4$	$7.23 \cdot 10^7$
		Ours	**350.3**	$\mathbf{3.29 \cdot 10^4}$	$\mathbf{9.12 \cdot 10^4}$	$\mathbf{1.51 \cdot 10^5}$	$\mathbf{9.35 \cdot 10^8}$	**532.8**	$\mathbf{2.43 \cdot 10^4}$	$\mathbf{8.35 \cdot 10^4}$	$\mathbf{1.14 \cdot 10^6}$	$\mathbf{1.14 \cdot 10^8}$
	MSCN	Clean	2.076	13.92	53.68	199.3	1653	2.235	99.10	179.1	505.1	2689
		Random	2.085	20.14	59.14	178.1	2116	2.724	161.8	298.8	732.2	4342
		GA	18.58	443.2	2111	$2.13 \cdot 10^4$	$1.12 \cdot 10^6$	19.71	397.2	1257	$1.09 \cdot 10^4$	$5.03 \cdot 10^4$
		SA	30.07	2198	6911	$2.51 \cdot 10^4$	$1.71 \cdot 10^5$	38.31	3731	$1.58 \cdot 10^4$	$1.05 \cdot 10^5$	$1.24 \cdot 10^5$
		PACE	54.60	569.1	1437	$4.46 \cdot 10^4$	$2.11 \cdot 10^5$	10.01	117.6	209.2	1488	9724
		Ours	**60.23**	**6201**	$\mathbf{3.04 \cdot 10^4}$	$\mathbf{1.56 \cdot 10^5}$	$\mathbf{1.46 \cdot 10^6}$	**184.3**	$\mathbf{4.63 \cdot 10^6}$	$\mathbf{1.16 \cdot 10^7}$	$\mathbf{4.74 \cdot 10^7}$	$\mathbf{8.08 \cdot 10^8}$
Data-Driven	NeuroCard	Clean	1.416	3.312	8.32	18.39	21.51	2.621	1089	6643	$4.95 \cdot 10^4$	$1.71 \cdot 10^6$
		Random	1.869	4.632	8.533	25.97	27.03	2.986	1092	6426	$4.95 \cdot 10^4$	$4.85 \cdot 10^6$
		GA	1.884	11.72	56.56	830.9	971.6	4.09	1073	6690	$6.6 \cdot 10^5$	$1.04 \cdot 10^6$
		SA	39.58	484	1031	3626	6618	3.967	1181	7211	$4.95 \cdot 10^5$	$1.04 \cdot 10^7$
		PACE	1.416	3.312	8.32	18.39	21.51	2.621	1089	6643	$4.95 \cdot 10^4$	$1.71 \cdot 10^6$
		Ours	**108.4**	**2972**	**3888**	$\mathbf{1.53 \cdot 10^4}$	$\mathbf{3.47 \cdot 10^4}$	**15.91**	$\mathbf{1.8 \cdot 10^5}$	$\mathbf{1.12 \cdot 10^6}$	$\mathbf{4.09 \cdot 10^7}$	$\mathbf{2.39 \cdot 10^8}$
	FactorJoin	Clean	3.015	6.82	39.11	$1.36 \cdot 10^4$	$1.98 \cdot 10^5$	2.018	40.61	78.91	302.7	760.1
		Random	3.176	5.481	45.82	$1.63 \cdot 10^4$	$8.01 \cdot 10^5$	2.072	45.98	64.56	312.98	774.3
		GA	4.088	156.7	4087	$1.19 \cdot 10^5$	$1.54 \cdot 10^6$	5.03	115.8	296.2	921.8	$1.13 \cdot 10^4$
		SA	3.999	889.6	$1.76 \cdot 10^4$	$8.76 \cdot 10^5$	$3.68 \cdot 10^8$	6.18	784.6	$1.67 \cdot 10^4$	$2.22 \cdot 10^7$	$1.41 \cdot 10^9$
		PACE	3.015	6.82	39.11	$1.36 \cdot 10^4$	$1.98 \cdot 10^5$	2.208	49.10	69.83	341.9	873.2
		Ours	**6.706**	$\mathbf{1.56 \cdot 10^5}$	$\mathbf{2.33 \cdot 10^6}$	$\mathbf{4.523 \cdot 10^7}$	$\mathbf{1.04 \cdot 10^9}$	**6.486**	**2099**	$\mathbf{8.05 \cdot 10^4}$	$\mathbf{1.72 \cdot 10^8}$	$\mathbf{4.52 \cdot 10^9}$
Hybrid	ALECE	Clean	1.374	2.611	3.702	16.03	649.4	1.532	3.007	4.321	22.24	$1.52 \cdot 10^4$
		Random	1.566	2.741	3.499	9.782	439.7	1.659	4.008	6.235	27.04	$1.96 \cdot 10^5$
		GA	13.86	159.2	212.9	5547	$1.28 \cdot 10^5$	2.996	37.63	82.93	296.1	6323
		SA	27.01	696.1	2723	$5.06 \cdot 10^4$	$1.49 \cdot 10^5$	3.825	93.88	317.5	1141	5144
		PACE	7.229	134.4	265.6	4284	$4.62 \cdot 10^5$	7.712	173.5	605.1	4870	$7.98 \cdot 10^5$
		Ours	**29.33**	**9323**	$\mathbf{2.73 \cdot 10^4}$	$\mathbf{2.45 \cdot 10^5}$	$\mathbf{2.02 \cdot 10^6}$	**5.568**	**6910**	$\mathbf{3.47 \cdot 10^4}$	$\mathbf{3.31 \cdot 10^5}$	$\mathbf{9.46 \cdot 10^5}$
	RobustMSCN	Clean	1.811	4.415	23.67	355.5	1102	2.288	5.196	7.43	28.62	4470
		Random	2.145	5.409	23.81	381.5	2547	2.723	5.903	7.743	23.55	4656
		GA	15.30	109.1	173.8	627.8	$1.83 \cdot 10^4$	6.834	131.1	234.1	753.2	$2.91 \cdot 10^4$
		SA	12.69	186.3	805.5	5705	$2.16 \cdot 10^4$	8.567	223.7	747.3	3282	$2.85 \cdot 10^4$
		PACE	5.019	77.26	278.7	2352	7155	8.56	238.9	504.6	3468	6638
		Ours	**15.87**	**814.7**	**2185**	$\mathbf{1.52 \cdot 10^4}$	$\mathbf{2.64 \cdot 10^5}$	**13.20**	**3433**	$\mathbf{1.09 \cdot 10^4}$	$\mathbf{1.76 \cdot 10^5}$	$\mathbf{1.18 \cdot 10^6}$
-	Oracle	Clean	1	1	1	1	1	1	1	1	1	1
		Random	1.255	1.294	1.663	1.766	3.338	1.212	1.298	1.319	1.515	2.328
		GA	83.39	230.3	2005	$5.15 \cdot 10^5$	$1.674 \cdot 10^6$	25.97	258.8	1566	8741	$1.44 \cdot 10^4$
		SA	301.1	$2.72 \cdot 10^5$	$5.12 \cdot 10^5$	$1.54 \cdot 10^7$	$3.23 \cdot 10^7$	35.98	2133	4835	$1.43 \cdot 10^4$	$1.56 \cdot 10^6$
		Ours	$\mathbf{1.49 \cdot 10^5}$	$\mathbf{1.23 \cdot 10^8}$	$\mathbf{5.03 \cdot 10^8}$	$\mathbf{3.89 \cdot 10^9}$	$\mathbf{9.53 \cdot 10^9}$	$\mathbf{1.66 \cdot 10^4}$	$\mathbf{1.02 \cdot 10^8}$	$\mathbf{2.50 \cdot 10^8}$	$\mathbf{2.21 \cdot 10^9}$	$\mathbf{1.78 \cdot 10^{10}}$



(a) Attack budget vs Qerror (b) Scalability and E2E Time

Figure 6: Scalability on different Budget



(a) Distribution on: IMDB-JOB (b) Distribution on: STATS-CEB

Figure 7: Operator Distribution

our attack. It can be observed that the attack time is proportional to the attack budget, demonstrating good scalability and performance. Additionally, a larger attack budget, while requiring more time, also yields better attack effectiveness. For instance, increasing the budget from 0.05 to 0.1 results in an end-to-end time increase by 2.7×.

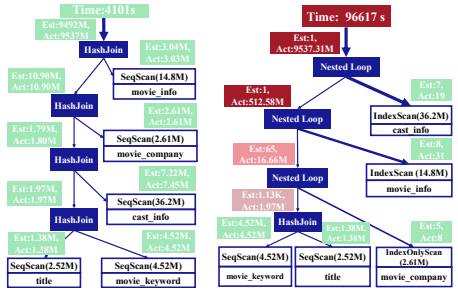
Take away: The overhead of our proposed DACA attack strategy is linear to the attack budget and correlated with table size and query number. Moreover, the more queries that involve the attacked table, the larger the attack budget, and consequently, the greater the benefit derived from the attack.

5.4 Consequence study

In this section, we have selected LWNN as the target model. We conducted tests on IMDB-JOB and STATS-CEB, and we analyzed the distribution of physical plans. We visualized the operators distribution in Figure 7.

Join pattern: We found that, compared to the execution plans before the attack, the execution plans after the attack exhibited the greatest increases in the Nested Loop and Index Scan operators. Specifically, the Nested Loop operator performance improved by 128% and 61% in the IMDB-JOB and STATS-CEB databases, respectively, while the Index Scan operator performance increased by 12% and 15% in these two databases, respectively. The operators that experienced the most significant reductions in quantity were the Hash Join and Merge Join operators. The Hash Join operator decreased by 27% and 20% in the IMDB-JOB and STATS-CEB databases, respectively, whereas the Merge Join operator decreased by 18% and 32% in these databases, respectively. These operators indicate that the erroneous cardinalities provided by the cardinality estimator after the attack severely mislead the query optimizer into selecting Nested Loop Joins or Index-Based Nested Loop Joins, which are more suitable for small-scale queries, instead of the more optimal

Hash Joins or Merge Joins for complex analytical queries. A quintessential example is the degradation of IMDB-JOB Q60, as illustrated in Figure 8(a) and Figure 8(b). The cardinality estimator, when misled, opts for Index Nested Loop joins rather than Hash Joins for all three nodes at the top of the query plan tree. The incorrect selection of operators increases the computational complexity of the physical query plan, thereby extending end-to-end processing time.



(a) Before Attack (b) After Attack

Figure 8: Physical plan of IMDB-JOB Q60

Cache pattern: Additionally, we observed that, aside from the common joint operators, the proportion of materialized operators also significantly decreased. In the IMDB-JOB database, the use of materialize operators was reduced by 90%, and in the STATS-CEB database, their proportion decreased by 51%. This suggests that the compromised cardinality estimator misleads the optimizer to reduce access to materialized caches, thereby increasing the rate of redundant computations and decreasing cache hit rates.

Take away: After being subjected to our data-centric algorithm complexity attack, the learned estimator supplies inaccurate estimates to the database optimizer, which has two major consequences: (1) **Surge in computational complexity:** The mislead optimizer tends to select the suboptimal Nested Loop operators for hard analytical queries, thereby increasing the theoretical time complexity and prolonging execution times. (2) **Decline in memory cache hit rates:** The misguided optimizer tends to expand computations from scratch rather than reuse existing materialized computational results, which subsequently lowers the system's cache hit rate and diminishes overall system efficiency.

6 RELATED WORK

Learned Cardinality Estimators: Modern cardinality estimators can be categorized into three paradigms: query-driven [21, 29, 47], data-driven[24, 37, 37, 45, 51, 52] and hybrid [36, 39, 54]. **(1) Query-driven estimators** employ neural networks [21, 29, 47] to learn mappings from query representations to true cardinalities. Although query-driven methods can achieve quick and lightweight estimations, is confirmed by many studies that query-driven estimators suffer from changes in testing query distribution, which may be caused by data updates or malicious poisoning within query workloads. [36, 42, 43, 52]. **(2)Data-driven estimators** learn joint data distributions and sample on deep model [28, 51, 52] to estimate cardinalities. They are robust against workload drifts [43, 52]. **(3) Hybrid estimators:** [23, 29, 36, 39, 39, 44] use statistical information from the data and query representations for unified learning. Commonly used data features include histograms [36, 54], sample collections [29, 39], and estimations from PostgreSQL [23, 39, 54].

Attacks on learned database components Nowadays, attacks on learned database components have garnered widespread attention [30, 50, 55, 57]. Kornaropoulos et al. [30, 50] introduced poisoning attacks and algorithm complexity attacks targeting learned indexes by constructing malicious inputs that cause learned indexes to fail. Additionally, Zheng et al. [57] proposed poisoning attacks aimed at learning-based index advisors. The work most closely related to ours is PACE proposed by Zhang et al. [55], which affected query-driven estimators by adding noise to their training workloads. However, PACE has a clear limitation: it can only target query-driven models. In contrast, the attack method proposed in this paper can influence nearly all types of cardinality estimators.

Data Poisoning Attacks. In the domain of machine learning, poisoning attacks are primarily analyzed under the white-box assumption and attacker has knowledge of the internal model [11, 15, 16, 26, 48, 49, 56]. For example: (1) Assuming the use of linear regression, Jagielski et al. [26] proposed an optimization framework for poisoning attacks targeting linear regression. (2) Assuming the internal model uses SVM [16, 48, 56]. By injecting maliciously crafted training data, the decision boundary of the SVMs can be manipulated, thereby elevating the error rate. (3) Assuming the use of neural networks, Yang et al. [49] introduced a gradient-based technique to generate poisoning input for neural networks. In contrast, we consider a more realistic scenario where the attacker has no knowledge of the details of the model used within the database; the attacker is unaware whether the internal model is based on neural networks [36, 51] or decision trees [21, 24, 60] or Bayes Networks [45, 46] or any other new models [42, 54].

7 CONCLUSION

In this study, we have unveiled the significant vulnerabilities of all learned cardinality estimators to minimal data-level drifts. Through a black-box data-centric algorithmic complexity attack, we showed that even slight modifications to the training data can severely degrade all estimators' accuracy. We proved the NP-Hardness of finding an optimal attack strategy and developed an efficient approximation algorithm that achieves near-optimal solutions in polynomial time with a $(1-\kappa)$ approximation ratio. Our experiments on benchmarks like STATS-CEB and IMDB-JOB demonstrated the effectiveness of our attack. Notably, altering just 1% of the database tuples led to a thousandfold increase in the 90-th percentile Qerror and the end-to-end processing time by up to twentyfold. These results highlight the fragility of current estimators and the critical risks they pose in real-world applications. Moving forward, it is crucial to develop more robust estimation techniques that can effectively handle data-level disruptions, thereby ensuring reliable performance in practical deployments.

REFERENCES

[1] [n.d.]. IMDB. http://homepages.cwi.nl/~boncz/job/imdb.tgz.,.
[2] [n.d.]. JobLightSubqueries. https://github.com/Nathaniel-Han/End-to-End-CardEst-Benchmark/tree/master/workloads/job-light.
[3] [n.d.]. StackExchange. https://stats.stackexchange.com/,.
[4] [n.d.]. STATSSubqueries. https://github.com/Nathaniel-Han/End-to-End-CardEst-Benchmark/tree/master/workloads/stats_CEB.

[5] 2023. ALECE: Evaluation Utilities. https://github.com/pfl-cs/ALECE/blob/main/src/utils/eval_utils.py

[6] 2023. PRICE: Evaluation Utilities. https://github.com/StCarmen/PRICE/blob/master/utils/model/qerror.py

[7] 2025. DACA. https://github.com/LiYingZe/DACA.

[8] Yuichi Asahiro, Refael Hassin, and Kazuo Iwama. 2002. Complexity of finding dense subgraphs. Discrete Appl. Math. 121, 1–3 (Sept. 2002), 15–26. https://doi.org/10.1016/S0166-218X(01)00243-8

[9] Nirav Atre, Hugo Sadok, Erica Chiang, Weina Wang, and Justine Sherry. 2022. SurgeProtector: mitigating temporal algorithmic complexity attacks using adversarial scheduling. In Proceedings of the ACM SIGCOMM 2022 Conference (Amsterdam, Netherlands) (SIGCOMM '22). Association for Computing Machinery, New York, NY, USA, 723–738. https://doi.org/10.1145/3544216.3544250

[10] Wenruo Bai and Jeffrey A. Bilmes. 2018. Greed is Still Good: Maximizing Monotone Submodular+Supermodular (BP) Functions. In Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018 (Proceedings of Machine Learning Research), Jennifer G. Dy and Andreas Krause (Eds.), Vol. 80. PMLR, 314–323. http://proceedings.mlr.press/v80/bai18a.html

[11] Marco Barreno, Blaine Nelson, Anthony Joseph, and J. Tygar. 2010. The security of machine learning. Machine Learning 81 (11 2010), 121–148. https://doi.org/10.1007/s10994-010-5188-5

[12] Udi Ben-Porat, Anat Bremler-Barr, and Hanoch Levy. 2013. Vulnerability of Network Mechanisms to Sophisticated DDoS Attacks. IEEE Trans. Comput. 62, 5 (2013), 1031–1043. https://doi.org/10.1109/TC.2012.49

[13] Kristin P Bennett, Michael C Ferris, and Yannis E Ioannidis. 1991. A genetic algorithm for database query optimization. Technical Report. University of Wisconsin-Madison Department of Computer Sciences.

[14] Dimitris Bertsimas and John Tsitsiklis. 1993. Simulated annealing. Statistical science 8, 1 (1993), 10–15.

[15] Battista Biggio, Blaine Nelson, and Pavel Laskov. 2012. Poisoning attacks against support vector machines. In Proceedings of the 29th International Coference on International Conference on Machine Learning (Edinburgh, Scotland) (ICML'12). Omnipress, Madison, WI, USA, 1467–1474.

[16] Battista Biggio, Ignazio Pillai, Samuel Rota Bulò, Davide Ariu, Marcello Pelillo, and Fabio Roli. 2013. Is data clustering in adversarial settings secure?. In Proceedings of the 2013 ACM Workshop on Artificial Intelligence and Security (Berlin, Germany) (AISec '13). Association for Computing Machinery, New York, NY, USA, 87–98. https://doi.org/10.1145/2517312.2517321

[17] Mark Braverman, Young Kun Ko, Aviad Rubinstein, and Omri Weinstein. 2017. ETH hardness for densest-k-subgraph with perfect completeness. In Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms(SODA). SIAM, 1326–1341.

[18] Chandra Chekuri, Kent Quanrud, and Manuel R. Torres. [n.d.]. Densest Subgraph: Supermodularity, Iterative Peeling, and Flow. In Proceedings of the 2022 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA). 1531–1555. https://doi.org/10.1137/1.9781611977073.64 arXiv:https://epubs.siam.org/doi/pdf/10.1137/1.9781611977073.64

[19] Scott A. Crosby and Dan S. Wallach. 2003. Denial of service via algorithmic complexity attacks. In Proceedings of the 12th Conference on USENIX Security Symposium - Volume 12 (Washington, DC) (SSYM'03). USENIX Association, USA, 3.

[20] Jialin Ding, Ryan Marcus, Andreas Kipf, Vikram Nathan, Aniruddha Nrusimha, Kapil Vaidya, Alexander van Renen, and Tim Kraska. 2022. SageDB: An Instance-Optimized Data Analytics System. Proc. VLDB Endow. 15, 13 (Sept. 2022), 4062–4078. https://doi.org/10.14778/3565838.3565857

[21] Anshuman Dutt, Chi Wang, Azade Nazi, Srikanth Kandula, Vivek Narasayya, and Surajit Chaudhuri. 2019. Selectivity estimation for range predicates using lightweight models. Proceedings of the VLDB Endowment (May 2019), 1044–1057. https://doi.org/10.14778/3329772.3329780

[22] Yuxing Han, Ziniu Wu, Peizhi Wu, Rong Zhu, Jingyi Yang, Liang Wei Tan, Kai Zeng, Gao Cong, Yanzhao Qin, Andreas Pfadler, Zhengping Qian, Jingren Zhou, Jiangneng Li, and Bin Cui. 2021. Cardinality estimation in DBMS: a comprehensive benchmark evaluation. Proc. VLDB Endow. 15, 4 (dec 2021), 752–765. https://doi.org/10.14778/3503585.3503586

[23] Benjamin Hilprecht and Carsten Binnig. 2022. Zero-Shot Cost Models for Out-of-the-box Learned Cost Prediction. Proc. VLDB Endow. 15, 11 (2022), 2361–2374. https://doi.org/10.14778/3551793.3551799

[24] Benjamin Hilprecht, Andreas Schmidt, Moritz Kulessa, Alejandro Molina, Kristian Kersting, and Carsten Binnig. 2020. DeepDB: Learn from Data, Not from Queries! Proc. VLDB Endow. 13, 7 (mar 2020), 992–1005. https://doi.org/10.14778/3384345.3384349

[25] Yannis E Ioannidis and Eugene Wong. 1987. Query optimization by simulated annealing. In Proceedings of the 1987 ACM SIGMOD international conference on Management of data. 9–22.

[26] Matthew Jagielski, Alina Oprea, Battista Biggio, Chang Liu, Cristina Nita-Rotaru, and Bo Li. 2021. Manipulating Machine Learning: Poisoning Attacks and Countermeasures for Regression Learning. arXiv:1804.00308 [cs.CR] https://arxiv.org/abs/1804.00308

[27] Kyoungmin Kim, Jisung Jung, In Seo, Wook-Shin Han, Kangwoo Choi, and Jaehyok Chong. 2022. Learned Cardinality Estimation: An In-depth Study. In Proceedings of the 2022 International Conference on Management of Data (Philadelphia, PA, USA) (SIGMOD '22). Association for Computing Machinery, New York, NY, USA, 1214–1227. https://doi.org/10.1145/3514221.3526154

[28] Kyoungmin Kim, Sangoh Lee, Injung Kim, and Wook-Shin Han. 2024. ASM: Harmonizing Autoregressive Model, Sampling, and Multi-dimensional Statistics Merging for Cardinality Estimation. Proc. ACM Manag. Data 2, 1, Article 45 (mar 2024), 27 pages. https://doi.org/10.1145/3639300

[29] Andreas Kipf, Thomas Kipf, Bernhard Radke, Viktor Leis, Peter A. Boncz, and Alfons Kemper. 2018. Learned Cardinalities: Estimating Correlated Joins with Deep Learning. ArXiv abs/1809.00677 (2018). https://api.semanticscholar.org/CorpusID:52154172

[30] Evgenios M. Kornaropoulos, Silei Ren, and Roberto Tamassia. 2022. The Price of Tailoring the Index to Your Data: Poisoning Attacks on Learned Index Structures. In Proceedings of the 2022 International Conference on Management of Data (Philadelphia, PA, USA) (SIGMOD '22). Association for Computing Machinery, New York, NY, USA, 1331–1344. https://doi.org/10.1145/3514221.3517867

[31] Meghdad Kurmanji, Eleni Triantafillou, and Peter Triantafillou. 2024. Machine Unlearning in Learned Databases: An Experimental Analysis. Proc. ACM Manag. Data 2, 1, Article 49 (mar 2024), 26 pages. https://doi.org/10.1145/3639304

[32] Meghdad Kurmanji and Peter Triantafillou. 2023. Detect, Distill and Update: Learned DB Systems Facing Out of Distribution Data. Proceedings of the ACM on Management of Data 1, 1 (2023), 1–27.

[33] Per-Ake Larson, Wolfgang Lehner, Jingren Zhou, and Peter Zabback. 2007. Cardinality estimation using sample views with quality assurance. In Proceedings of the 2007 ACM SIGMOD international conference on Management of data. 175–186.

[34] Viktor Leis, Andrey Gubichev, Atanas Mirchev, Peter Boncz, Alfons Kemper, and Thomas Neumann. 2015. How good are query optimizers, really? Proceedings of the VLDB Endowment (Nov 2015), 204–215. https://doi.org/10.14778/2850583.2850594

[35] Beibin Li, Yao Lu, and Srikanth Kandula. 2022. Warper: Efficiently Adapting Learned Cardinality Estimators to Data and Workload Drifts. In Proceedings of the 2022 International Conference on Management of Data (Philadelphia, PA, USA) (SIGMOD '22). Association for Computing Machinery, New York, NY, USA, 1920–1933. https://doi.org/10.1145/3514221.3526179

[36] Pengfei Li, Wenqing Wei, Rong Zhu, Bolin Ding, Jingren Zhou, and Hua Lu. 2023. ALECE: An Attention-based Learned Cardinality Estimator for SPJ Queries on Dynamic Workloads. Proc. VLDB Endow. 17, 2 (oct 2023), 197–210. https://doi.org/10.14778/3626292.3626302

[37] Yingze Li, Hongzhi Wang, and Xianglong Liu. 2024. One Seed, Two Birds: A Unified Learned Structure for Exact and Approximate Counting. Proc. ACM Manag. Data 2, 1, Article 15 (mar 2024), 26 pages. https://doi.org/10.1145/3639270

[38] Melanie Mitchell. 1998. An introduction to genetic algorithms. MIT press.

[39] Parimarjan Negi, Ziniu Wu, Andreas Kipf, Nesime Tatbul, Ryan Marcus, Sam Madden, Tim Kraska, and Mohammad Alizadeh. 2023. Robust Query Driven Cardinality Estimation under Changing Workloads. Proc. VLDB Endow. 16, 6 (Feb. 2023), 1520–1533. https://doi.org/10.14778/3583140.3583164

[40] Theofilos Petsios, Jason Zhao, Angelos D. Keromytis, and Suman Jana. 2017. SlowFuzz: Automated Domain-Independent Detection of Algorithmic Complexity Vulnerabilities. In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (Dallas, Texas, USA) (CCS '17). Association for Computing Machinery, New York, NY, USA, 2155–2168. https://doi.org/10.1145/3133956.3134073

[41] Viswanath Poosala, Peter J. Haas, Yannis Ioannidis, and Eugene J. Shekita. 1996. Improved histograms for selectivity estimation of range predicates. International Conference on Management of Data (1996).

[42] Jiayi Wang, Chengliang Chai, Jiabin Liu, and Guoliang Li. 2021. FACE: A normalizing flow based cardinality estimator. Proceedings of the VLDB Endowment 15, 1 (2021), 72–84.

[43] Xiaoying Wang, Changbo Qu, Weiyuan Wu, Jiannan Wang, and Qingqing Zhou. 2021. Are We Ready for Learned Cardinality Estimation? Proc. VLDB Endow. 14, 9 (may 2021), 1640–1654. https://doi.org/10.14778/3461535.3461552

[44] Peizhi Wu and Gao Cong. 2021. A Unified Deep Model of Learning from both Data and Queries for Cardinality Estimation. In Proceedings of the 2021 International Conference on Management of Data. 2009–2022.

[45] Ziniu Wu, Parimarjan Negi, Mohammad Alizadeh, Tim Kraska, and Samuel Madden. 2023. FactorJoin: A New Cardinality Estimation Framework for Join Queries. Proc. ACM Manag. Data 1, 1, Article 41 (may 2023), 27 pages. https://doi.org/10.1145/3588721

[46] Ziniu Wu, Amir Shaikhha, Rong Zhu, Kai Zeng, Yuxing Han, and Jingren Zhou. 2021. BayesCard: Revitilizing Bayesian Frameworks for Cardinality Estimation. arXiv:2012.14743 [cs.DB] https://arxiv.org/abs/2012.14743

[47] Ziniu Wu, Peilun Yang, Pei Yu, Rong Zhu, Yuxing Han, Yaliang Li, Defu Lian, Kai Zeng, and Jingren Zhou. 2022. A Unified Transferable Model for ML-Enhanced

DBMS. Conference on Innovative Data Systems Research (2022).
[48] Han Xiao, Huang Xiao, and Claudia Eckert. 2012. Adversarial label flips attack on support vector machines. In Proceedings of the 20th European Conference on Artificial Intelligence (Montpellier, France) (ECAI'12). IOS Press, NLD, 870–875.
[49] Chaofei Yang, Qing Wu, Hai Li, and Yiran Chen. 2017. Generative Poisoning Attack Method Against Neural Networks. arXiv:1703.01340 [cs.CR] https://arxiv.org/abs/1703.01340
[50] Rui Yang, Evgenios M. Kornaropoulos, and Yue Cheng. 2024. Algorithmic Complexity Attacks on Dynamic Learned Indexes. Proc. VLDB Endow. 17, 4 (March 2024), 780–793. https://doi.org/10.14778/3636218.3636232
[51] Zongheng Yang, Amog Kamsetty, Sifei Luan, Eric Liang, Yan Duan, Xi Chen, and Ion Stoica. 2020. NeuroCard: One Cardinality Estimator for All Tables. Proc. VLDB Endow. 14, 1 (sep 2020), 61–73. https://doi.org/10.14778/3421424.3421432
[52] Zongheng Yang, Eric Liang, Amog Kamsetty, Chenggang Wu, Yan Duan, Xi Chen, Pieter Abbeel, Joseph M. Hellerstein, Sanjay Krishnan, and Ion Stoica. 2019. Deep Unsupervised Cardinality Estimation. Proc. VLDB Endow. 13, 3 (nov 2019), 279–292. https://doi.org/10.14778/3368289.3368294
[53] Xiang Yu, Chengliang Chai, Guoliang Li, and Jiabin Liu. 2022. Cost-Based or Learning-Based? A Hybrid Query Optimizer for Query Plan Selection. Proc. VLDB Endow. 15, 13 (Sept. 2022), 3924–3936. https://doi.org/10.14778/3565838.3565846
[54] Tianjing Zeng, Junwei Lan, Jiahong Ma, Wenqing Wei, Rong Zhu, Pengfei Li, Bolin Ding, Defu Lian, Zhewei Wei, and Jingren Zhou. 2024. PRICE: A Pretrained Model for Cross-Database Cardinality Estimation. https://doi.org/10.48550/arXiv.2406.01027
[55] Jintao Zhang, Chao Zhang, Guoliang Li, and Chengliang Chai. 2024. PACE: Poisoning Attacks on Learned Cardinality Estimation. Proc. ACM Manag. Data 2, 1, Article 37 (mar 2024), 27 pages. https://doi.org/10.1145/3639292
[56] Rui Zhang and Quanyan Zhu. 2017. A game-theoretic analysis of label flipping attacks on distributed support vector machines. In 2017 51st Annual Conference on Information Sciences and Systems (CISS). 1–6. https://doi.org/10.1109/CISS.2017.7926118
[57] Yihang Zheng, Chen Lin, Xian Lyu, Xuanhe Zhou, Guoliang Li, and Tianqing Wang. 2024. Robustness of Updatable Learning-based Index Advisors against Poisoning Attack. Proc. ACM Manag. Data 2, 1, Article 10 (March 2024), 26 pages. https://doi.org/10.1145/3639265
[58] Yingli Zhou, Yixiang Fang, Wensheng Luo, and Yunming Ye. 2023. Influential Community Search over Large Heterogeneous Information Networks. Proc. VLDB Endow. 16, 8 (April 2023), 2047–2060. https://doi.org/10.14778/3594512.3594532
[59] Yingli Zhou, Qingshuo Guo, Yixiang Fang, and Chenhao Ma. 2024. A Counting-based Approach for Efficient k-Clique Densest Subgraph Discovery. Proc. ACM Manag. Data 2, 3, Article 119 (May 2024), 27 pages. https://doi.org/10.1145/3654922
[60] Rong Zhu, Ziniu Wu, Yuxing Han, Kai Zeng, Andreas Pfadler, Zhengping Qian, Jingren Zhou, and Bin Cui. 2021. FLAT: fast, lightweight and accurate method for cardinality estimation. Proc. VLDB Endow. 14, 9 (May 2021), 1489–1502. https://doi.org/10.14778/3461535.3461539

Due to space limitations, we present some proof details in this appendix.

A PROOF OF THEOREM3.2

We present the proof of Theorem 3.2 here:

Proof. We aim to prove that when $x > K \times M$, the benefit in Qerror resulting from inserting I tuples ($2 \le I \le K$) into the database constructed from Theorem 3.1 is smaller than the benefit obtained by not deleting any two tuples corresponding to any q_x. Consequently, the plan of inserting I tuples can be disregarded and replaced with I delete operations. And we can therefore reuse the PTIME reduction in theorem3.1, construct a polynomial-time reduction from the DKS problem to the DACA problem when simultaneously considering insertions and deletions. Therefore the DACA problem considering both insertions and deletions remains NP-hard.

For a fixed query q_x, deleting its two tuples in R_x yields a benefit $g_{\text{Del}} = 2x + 1$. Given that $x > K \times M$, we have

$$g_{\text{Del}} > 2 \times K \times M + 1.$$

Assume that all M queries share one common test tuple in R_x. Therefore, repeatedly inserting into this special tuple would be the optimal solution for inserting I tuples, which provides an upper bound on the Qerror when inserting I tuples. That is,

$$g_{\text{Ins}} \le M \times \frac{I+2}{2}.$$

Given that $g_{\text{Del}} > 2 \times K \times M + 1$ and $K \ge I$, we have

$$g_{\text{Del}} > 2 \times K \times M > 2 \times I \times M = \frac{IM}{2} + \frac{3IM}{2}.$$

Since $I \ge 2$, we have $\frac{3IM}{2} > M$, and thus

$$2 \times I \times M = \frac{IM}{2} + \frac{3IM}{2} > \frac{IM}{2} + M \ge g_{\text{Ins}}.$$

In conclusion, we deduce that the benefit g_{Del} obtained by deleting any two tuples corresponding to query q_x in R_x is greater than the benefit g_{Ins} of inserting I tuples ($2 \le I \le K$). □

B PROOF OF THE DERIVATION IN THEOREM4.1

Given that $\mathbb{Q}_j(X) = \frac{C_D(\mathbf{q}_j)+1}{C_D(\mathbf{q}_j)+1+\sum_{t_i \in X} t_i.\beta \times w_{ij}}$ and $w'_{ij} = \frac{w_{ij}}{1+C_D(\mathbf{q}_j)}$, $\quad S_j(X) = \sum_{t_i \in X} t_i.\beta w'_{ij}$

We prove that the expression

$$\mathbb{Q}_j(A \cup B) + \mathbb{Q}_j(A \cap B) - \mathbb{Q}_j(A) - \mathbb{Q}_j(B)$$

can be reorganized in the following form:

$$\frac{(2 + S_j(A) + S_j(B))(S_j(A \setminus B))(S_j(B \setminus A))}{(1 + S_j(A \cup B))(1 + S_j(A \cap B))(1 + S_j(A))(1 + S_j(B))}$$

Proof. Based on the weights w'_{ij} and the function $S_j(X)$, we define the function $\mathbb{Q}_j(X)$ as follows:

$$\mathbb{Q}_j(X) = \frac{C_D(\mathbf{q}_j) + 1}{C_D(\mathbf{q}_j) + 1 + \sum_{t_i \in X} t_i.\beta \times w_{ij}} = \frac{1}{1 + S_j(X)}.$$

Therefore, for any sets A and B, the following relationship holds:

$$\begin{aligned}&\mathbb{Q}_j(A \cup B) + \mathbb{Q}_j(A \cap B) - \mathbb{Q}_j(A) - \mathbb{Q}_j(B)\\ =&\frac{1}{1 + S_j(A \cup B)} + \frac{1}{1 + S_j(A \cap B)} - \frac{1}{1 + S_j(A)} - \frac{1}{1 + S_j(B)}\end{aligned}$$

We examine the first two terms of the above summation:

$$\frac{1}{1 + S_j(A \cup B)} + \frac{1}{1 + S_j(A \cap B)}$$

This can be combined as follows:

$$\begin{aligned}&\frac{1}{1 + S_j(A \cup B)} + \frac{1}{1 + S_j(A \cap B)}\\ &= \frac{(2 + S_j(A \cup B) + S_j(A \cap B)) \times (1 + S_j(A)) \times (1 + S_j(B))}{(1 + S_j(A \cup B))(1 + S_j(A \cap B))(1 + S_j(A))(1 + S_j(B))}\\ &= \frac{2 + S_j(A \cup B) + S_j(A \cap B)}{(1 + S_j(A \cup B))(1 + S_j(A \cap B))} \times \frac{(1 + S_j(A))(1 + S_j(B))}{(1 + S_j(A))(1 + S_j(B))}\end{aligned}$$

Similarly, the last two terms of the summation are:

$$\frac{1}{1 + S_j(A)} + \frac{1}{1 + S_j(B)}$$

These terms can be combined as:

$$\begin{aligned}&\frac{1}{1 + S_j(A)} + \frac{1}{1 + S_j(B)}\\ &= \frac{(2 + S_j(A) + S_j(B)) \times (1 + S_j(A \cup B)) \times (1 + S_j(A \cap B))}{(1 + S_j(A))(1 + S_j(B))(1 + S_j(A \cup B))(1 + S_j(A \cap B))}\\ &= \frac{2 + S_j(A) + S_j(B)}{(1 + S_j(A))(1 + S_j(B))} \times \frac{(1 + S_j(A \cup B))(1 + S_j(A \cap B))}{(1 + S_j(A \cup B))(1 + S_j(A \cap B))}\end{aligned}$$

Notably, we observe that:

$$2+S_j(A)+S_j(B) = 2+S_j(A\setminus B)+2S_j(A\cap B)+S_j(B\setminus A) = 2+S_j(A\cup B)+S_j(A\cap B)$$

Therefore, by combining the above results, we obtain:

$$\frac{1}{1 + S_j(A \cup B)} + \frac{1}{1 + S_j(A \cap B)} - \frac{1}{1 + S_j(A)} - \frac{1}{1 + S_j(B)} = \frac{(2 + S_j(A) + S_j(B)) \left((1 + S_j(A))(1 + S_j(B)) - (1 + S_j(A \cup B))(1 + S_j(A \cap B))\right)}{(1 + S_j(A \cup B))(1 + S_j(A \cap B))(1 + S_j(A))(1 + S_j(B))}$$

Additionally, we observe that:

$$\begin{aligned}&(1 + S_j(A))(1 + S_j(B)) - (1 + S_j(A \cup B))(1 + S_j(A \cap B))\\ &= \left[1 + S_j(A \cap B) + S_j(A \setminus B)\right]\left[1 + S_j(A \cap B) + S_j(B \setminus A)\right]\\ &\quad - \left[1 + S_j(A \cap B) + S_j(A \setminus B) + S_j(B - A)\right](1 + S_j(A \cap B))\\ &= S_j(A \setminus B) \times S_j(B \setminus A).\end{aligned}$$

Therefore, we can substitute the equation in and have:

$$\begin{aligned}&\frac{1}{1 + S_j(A \cup B)} + \frac{1}{1 + S_j(A \cap B)} - \frac{1}{1 + S_j(A)} - \frac{1}{1 + S_j(B)}\\ &= \frac{(2 + S_j(A) + S_j(B))(S_j(A \setminus B))(S_j(B \setminus A))}{(1 + S_j(A \cup B))(1 + S_j(A \cap B))(1 + S_j(A))(1 + S_j(B))}.\end{aligned}$$

Therefore we proofed that:

$$\mathbb{Q}_j(A \cup B) + \mathbb{Q}_j(A \cap B) - \mathbb{Q}_j(A) - \mathbb{Q}_j(B) = \frac{(2 + S_j(A) + S_j(B))(S_j(A \setminus B))(S_j(B \setminus A))}{(1 + S_j(A \cup B))(1 + S_j(A \cap B))(1 + S_j(A))(1 + S_j(B))}.$$

□

C PROOF OF THEOREM4.2

We present the proof of Theorem 4.2 here:

Proof. We aim to demonstrate that the objective

$$\mathbb{Q}'(X) = \sum_{j=1}^{M} \mathbb{Q}_j(X) = \sum_{j=1}^{M} \frac{C_D(\mathbf{q}_j) + 1 + \sum_{t_i \in X} t_i.\beta \times w_{ij}}{C_D(\mathbf{q}_j) + 1}$$

Due to the linear nature of the target $\mathbb{Q}'(X)$, we can move the summation over $t_i.\beta$ from the inner layer to the outer layer, rearrange $t_i.\beta$, and combine elements based on the sets $(A \setminus B)$, $(B \setminus A)$, and $(A \cap B)$. Then, we prove that $\mathbb{Q}'(X)$ satisfies the modular property.

We can rewrite $\mathbb{Q}'(X)$ as:

$$\mathbb{Q}'(X) = M + \sum_{j=1}^{M} \frac{\sum_{t_i \in X} t_i.\beta \times w_{ij}}{C_D(\mathbf{q}_j) + 1}.$$

Let us define $H_i = \sum_{j=1}^{M} \frac{w_{ij}}{C_D(\mathbf{q}_j)+1}$. Consequently, we have

$$\mathbb{Q}'(X) = M + \sum_{i=1}^{N} t_i.\beta \times H_i.$$

Considering the left-hand side, we obtain:

$$\mathbb{Q}'(A \cup B) + \mathbb{Q}'(A \cap B) = \sum_{t_k \in A \cup B} t_k.\beta \times H_k + \sum_{t_k \in A \cap B} t_k.\beta \times H_k.$$

This can be expanded as:

$$\sum_{t_k \in A \setminus B} t_k.\beta \times H_k + 2 \sum_{t_k \in A \cap B} t_k.\beta \times H_k + \sum_{t_k \in B \setminus A} t_k.\beta \times H_k.$$

On the other hand, the right-hand side is:

$$\mathbb{Q}'(A) + \mathbb{Q}'(B) = \sum_{t_k \in A} t_k.\beta \times H_k + \sum_{t_k \in B} t_k.\beta \times H_k.$$

This also expands to:

$$\sum_{t_k \in A \setminus B} t_k.\beta \times H_k + 2 \sum_{t_k \in A \cap B} t_k.\beta \times H_k + \sum_{t_k \in B \setminus A} t_k.\beta \times H_k.$$

Therefore, we have

$$\mathbb{Q}'(A) + \mathbb{Q}'(B) = \mathbb{Q}'(A \cup B) + \mathbb{Q}'(A \cap B).$$

This equality demonstrates the modular property. □

D PROOF OF COROLLARY 1

Here, we will prove our Corollary 1. When the attacker specifies which queries to insert and which to delete, our optimization goal becomes

$$\text{Max}: \sum_{\mathbf{q}_j \in \text{Insert}} \frac{C_D(\mathbf{q}_j) + 1 + \sum_{i=1}^{N} t_i \cdot \beta \times w_{ij}}{C_D(\mathbf{q}_j) + 1} + \sum_{\mathbf{q}_j \in \text{Delete}} \frac{C_D(\mathbf{q}_j) + 1}{C_D(\mathbf{q}_j) + 1 + \sum_{i=1}^{N} t_i \cdot \beta \times w_{ij}}$$

$$\text{s.t.} \quad \sum_{i=1}^{N} |t_i \cdot \beta| \leq K, \quad t_i \cdot \beta \in \{-1, 0, 1, \ldots, K\}$$

According to Theorem 4.1 and Theorem 4.2, the first term of the optimization objective

$$\sum_{\mathbf{q}_j \in \text{Insert}} \frac{C_D(\mathbf{q}_j) + 1 + \sum_{i=1}^{N} t_i \cdot \beta w_{ij}}{C_D(\mathbf{q}_j) + 1}, \tag{7}$$

exhibits modular properties. In contrast, the second term of the optimization objective

$$\sum_{\mathbf{q}_j \in \text{Delete}} \frac{C_D(\mathbf{q}_j) + 1}{C_D(\mathbf{q}_j) + 1 + \sum_{i=1}^{N} t_i \cdot \beta w_{ij}}, \tag{8}$$

demonstrates supermodular characteristics. The combination exhibits supermodular characteristics. This aligns with the conditions outlined in [10], where the objective is defined as the sum of a supermodular function and a submodular function with zero curvature. Furthermore, Bai et al. [10] have demonstrated that this linear combination does not impact the approximation ratio of the greedy algorithm when applied to such linear summation functions. As a result, the approximation ratio remains:

$$\mathbb{Q}(\Delta D) \geq (1 - \kappa)\mathbb{Q}(O). \tag{9}$$

E PROOF OF COROLLARY 2

We aim to demonstrate the following:

1. Under the given conditions, the optimal solution does not include insertions.

Consider the scenario where insertions are contemplated. The upper bound of the potential gain from insertions arises from the repeated insertion across M queries, specifically denoted as $K \times M$. This gain is inferior compared to the cost incurred by deleting one or two tuples corresponding to a query q_x, represented as $C_D(q_j)$. Mathematically, this relationship can be expressed as:

$$K \times M < C_D(q_j)$$

Consequently, the optimal solution favors deletion over insertion, implying that the optimal solution does not incorporate insertions.

2. Under the given conditions, Algorithm 1's will not choose to insert

Algorithm 1 evaluates the remaining undeleted tuples at each step. Consider the L-th step of the algorithm, where we examine an undeleted tuple t_i. We use D_L to represent the current database state and let S_1 denote the set of queries q_j involving t_i with $count(w_{i,j} \neq 0) = 1$, S_2 denote the set of queries q_j involving t_i with $count(w_{i,j} \neq 0) = 2$. The potential gain from repeatedly inserting tuples associated with queries in S is given by:

$$gainI = \sum_{q_j \in S_1} \frac{2 \cdot C_{D_L}(q_j) + 1}{C_{D_L}(q_j) + 1} + \sum_{q_j \in S_2} \frac{C_{D_L}(q_j) + 1 + w_{i,j}}{C_{D_L}(q_j) + 1}$$

On the other hand, if tuple t_i is deleted, the corresponding gain is:

$$gainD = \sum_{q_j \in S_1} \frac{C_{D_L}(q_j) + 1}{1} + \sum_{q_j \in S_2} \frac{C_{D_L}(q_j) + 1}{C_{D_L}(q_j) + 1 - w_{i,j}}$$

Note that:
$\frac{C_{D_L}(q_j)+1}{C_{D_L}(q_j)+1-w_{i,j}} - \frac{C_{D_L}(q_j)+1+w_{i,j}}{C_{D_L}(q_j)+1} = \frac{w_{i,j}}{C_{D_L}(q_j)+1-w_{i,j}} - \frac{w_{i,j}}{C_{D_L}(q_j)+1} > 0$
and:

$$\frac{C_{D_L}(q_j) + 1}{1} - \frac{2 \cdot C_{D_L}(q_j) + 1}{C_{D_L}(q_j) + 1} = C_{D_L}(q_j) - \frac{C_{D_L}(q_j)}{C_{D_L}(q_j) + 1} \geq 0$$

We can compute the difference between the inner summations of the two groups separately, yielding:

$$\sum_{q_j \in S_1} \frac{C_{D_L}(q_j) + 1}{1} - \sum_{q_j \in S_1} \frac{2 \cdot C_{D_L}(q_j) + 1}{C_{D_L}(q_j) + 1}$$
$$= \sum_{q_j \in S_1} \left(\frac{C_{D_L}(q_j) + 1}{1} - \frac{2 \cdot C_{D_L}(q_j) + 1}{C_{D_L}(q_j) + 1} \right) > 0$$

Additionally, for the second group, we have:

$$\sum_{q_j \in S_2} \frac{C_{D_L}(q_j) + 1}{C_{D_L}(q_j) + 1 - w_{i,j}} - \sum_{q_j \in S_2} \frac{C_{D_L}(q_j) + 1 + w_{i,j}}{C_{D_L}(q_j) + 1}$$
$$= \sum_{q_j \in S_2} \left(\frac{C_{D_L}(q_j) + 1}{C_{D_L}(q_j) + 1 - w_{i,j}} - - \frac{C_{D_L}(q_j) + 1 + w_{i,j}}{C_{D_L}(q_j) + 1} \right) > 0$$

Combining the results from both groups, we obtain:

$$gainI < gainD$$

This inequality indicates that the gain from insertion (*gainI*) is less than the gain from deletion (*gainD*). Therefore, Algorithm 1 will opt for deletion over insertion at each step. As a result, the algorithm inherently avoids insertions, effectively reducing the problem to an insert-only scenario under the given conditions.

$$\mathbb{Q}(\Delta D) \geq (1 - \kappa)\mathbb{Q}(O). \tag{10}$$