Learning-based Index Advisor Poisoning Attack

Written by Xian Lu 23020231154214, Hanghai Hong 30920231154348(AI), Linghang Shi 24520230157436, Jiahao Wu 30920231154363(AI), FeiYu 23020231154252

Xiamen University

Abstract

Recently, many learning-based Index Advisors (IA) have been proposed to optimize index selection. Most of them model the index selection process as a Reinforcement Learning (RL) problem, which learns from interactions with databases through the workload. Learning-based IAs have shown more promise in index selection than heuristic-based IAs. However, they also introduce new security challenges, such as the risk of training data poisoning during model updates. In this paper, we make the first attempt of poisoning attacks on learning-based IA. We propose **PIDPA** (`Probing-Injecting Data Poisoning Attack`), a two-step poisoning framework that can launch an effective attack by only being a user of the victim IA, without knowing its internal structure or interfering with its training process. **PIDPA** probes the IA's index preferences to guide the attack in the probing stage and determines an optimal attack strategy by combining index preferences with the characteristics of the victim IA in the attack stage. We also introduce **IABART** (`Index Aware BART`), an index-aware workload generator that can produce tailored workloads based on specific index requirements for **PIDPA** . Our extensive experiments on various learning-based IAs under popular benchmarks demonstrate the effectiveness of our attacks on different IAs, datasets, and attack workload proportions. Finally, we analyze the principles of our attack based on different characteristics of victim IAs such as agent structure and state representation to guide more secure design of learning-based IAs in the future.

Introduction

Index selection is crucial for relational database systems. Traditional index selection relies on expert database administrators, which is costly. To automate this process, many automated Index Advisors (IA) have been proposed. Recently, learning-based IAs using Reinforcement Learning (RL) to select indexes have gained attention due to their ability to learn from data such as workloads and underlying data structures. However, learning-based IAs are vulnerable to poisoning attacks as a machine learning model. These attacks can be launched by users of IAs such as other tenants of a cloud database with low data isolation or other users of the same database or even ourselves who submit a burst of database transactions that are unrelated to the daily database transactions. Figure 1 illustrates a case of IA poisoning attack in an opaque-box scenario.

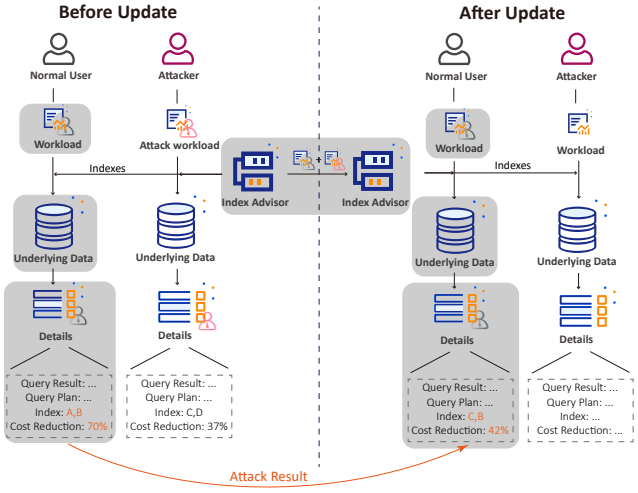

Figure 1: An example of IA poisoning attack

In this paper, we make the first attempt of poisoning attacks on learning-based IAs and evaluate their vulnerability. we propose **PIDPA** (`Probing-Injecting Data Poisoning Attack`), which consists of two stages. As shown in Figure 2, the first stage is a probing stage. Given a probing budget (i.e., number of iterations and the number of probing workloads allowed in each iteration), **PIDPA** perceives the victim IA's action preferences by repeatedly generating a small batch of *probing workload* and observing the victim IA's output index configurations. The generation of probing workload is achieved by calling a workload generator **IABART**(Section 5) that can tailor the workload given the desired index performance. The second stage is the attacking stage. **PIDPA** generates the *attacking workload* based on the perceived action preferences and injects the attacking workload to retrain the victim IA. Again the generation of the attacking workload is achieved by calling **IABART**.

Related Work

Index Advisor

Conventionally, most IAs are based on heuristic algorithms (Whang 1987; Bruno and Chaudhuri 2005; Chaudhuri and Narasayya 1997; Schlosser, Kossmann, and Boissier 2019). However, the latest learning-based index advisors show new promise in accuracy and efficiency. Most learning-based IAs learn from workload and model the index selection process as a Markov Decision Process (MDP) (Sharma, Schuhknecht, and Dittrich 2018; Lan, Bao, and Peng 2020; Sadri, Gruenwald, and Lead 2020; Sadri, Gruenwald, and Leal 2020; Perera et al. 2021; Kossmann, Kastius, and Schlosser 2022). This technology works by training an *agent* that takes the current *state* as input, outputs a predicted *action* to maximize the *reward* function, and updates parameters using the true *reward* from the environment. In this process, the model captures valuable information in the data through the *state* and *reward* during index selection and stores it in the form of *agent* parameters. In other words, the *state*, *reward*, and *agent* reflect the model's dependence on data, which is also key to the effectiveness of poisoning attacks.

For the *state* representation, different works have differences in index candidate composition and workload-aware capabilities. For instance, SmartIX (Paludo Licks et al. 2020) considers only the current single-attribute index configuration. NoDBA (Sharma, Schuhknecht, and Dittrich 2018) represents the *state* as a combination of the current single-attribute index configuration and a matrix containing each attribute's selectivity. DQN (Lan, Bao, and Peng 2020) represents it as a combination of current multi-attribute index configuration selected by predefined rules and queries frequency in the workload. DRLinda (Sadri, Gruenwald, and Lead 2020; Sadri, Gruenwald, and Leal 2020), which considers multiple instances in a database cluster, uses a binary query-attribute matrix, an access vector, and a density vector to count each attribute's frequency and selectivity. SWIRL (Kossmann, Kastius, and Schlosser 2022) takes into account not only the multi-attribute indexes configuration representation of the current state but also the queries' representation, frequencies and estimated costs.

For the *reward* function, SmartIX (Paludo Licks et al. 2020) uses TPCH benchmark indices to measure reward; NoDBA (Sharma, Schuhknecht, and Dittrich 2018) and DQN (Lan, Bao, and Peng 2020) use absolute cost reduction; DBAbandits (Perera et al. 2021) and SWIRL (Kossmann, Kastius, and Schlosser 2022) use relative cost reduction, which additionally considers storage and build costs.

For the *agent* network, NoDBA (Sharma, Schuhknecht, and Dittrich 2018) and SmartIX (Paludo Licks et al. 2020) use a simple MLP (LeCun, Bengio, and Hinton 2015); DQN (Lan, Bao, and Peng 2020) and DRLinda (Sadri, Gruenwald, and Lead 2020; Sadri, Gruenwald, and Leal 2020) use a traditional value-based DRL model DQN (Mnih et al. 2013); SWIRL (Kossmann, Kastius, and Schlosser 2022) uses a policy-based DRL model PPO (Schulman et al. 2017). DBAbandit (Perera et al. 2021), on the other hand, uses a non-DRL-based agent, a multi-armed slot machine based algorithm called C^2UCB.

In addition, a small number of learning-based algorithms work by Monte Carlo Tree Search (MCTS) (Browne et al. 2012; Zhou et al. 2022). They refine heuristics for index benefit estimation by learning from rollout which differ from learning from workload and data. This means that such models cannot be updated online: a change in workload means that they need to recalculate the rewards from rollout for all nodes of the tree.

Poisoning Attack

A poisoning attack occurs when an attacker influences the training process such as injecting the false training data into a machine learning model to corrupt it (Liu and Lai 2021). There are poisoning attacks for RL (Rakhsha et al. 2020; Ma et al. 2019; Behzadan and Munir 2017; Zhang et al. 2020; Sun, Huo, and Huang 2020; Rakhsha et al. 2021; Liu and Lai 2021), supervised learning (Biggio et al. 2013; Jagielski et al. 2018; Biggio, Nelson, and Laskov 2012), and unsupervised learning (Yang et al. 2017). From a capability perspective, RL poisoning attacks can be further classified into clear-box attacks (Ma et al. 2019; Behzadan and Munir 2017; Rakhsha et al. 2020; Zhang et al. 2020) and opaque-box attacks (Sun, Huo, and Huang 2020; Rakhsha et al. 2021; Liu and Lai 2021).

In terms of attack implementation details, existing research on poisoning attacks against RL has identified three types of adversarial manipulations. In the *observation poisoning attack* setting, the attacker can manipulate the agent's observation and alter its reward. Some studies assume that the victim model is known (Ma et al. 2019; Behzadan and Munir 2017; Zhang et al. 2020), while others assume that it is unknown (Sun, Huo, and Huang 2020; Rakhsha et al. 2021). In the *environment poisoning attack* setting, the attacker can alter the underlying environment (Rakhsha et al. 2020). In the *action poisoning attack* setting, the attacker can change the agent's action(Liu and Lai 2021).

Our poisoning attack differs significantly from all three types as it does not require any ability to interfere with the agent's reward, environment or action. Our approach also differs from another line of research on poisoning attacks against learned index structures (Kornaropoulos, Ren, and Tamassia 2020), which assumes a clear-box setting.

Proposed Solution

Our work considers four issues that are important to be addressed

I1: Opaque-Box Poisoning Attack. To make our attack more realistic, we assume that attackers are only users of the IA and have limited knowledge of its internal structure. Thus, attackers have only two capabilities: (1) to submit their workload as part of the training data used in IA updates and (2) to observe the index selection results given by the IA for a given workload. However, existing studies on RL poisoning attacks usually assume that attackers can either know the internal structure (Ma et al. 2019; Rakhsha et al. 2020; Behzadan and Munir 2017; Zhang et al. 2020; Liu and Lai 2021) or interfere with the internal training process such as agent's reward, environment, or action (Sun,

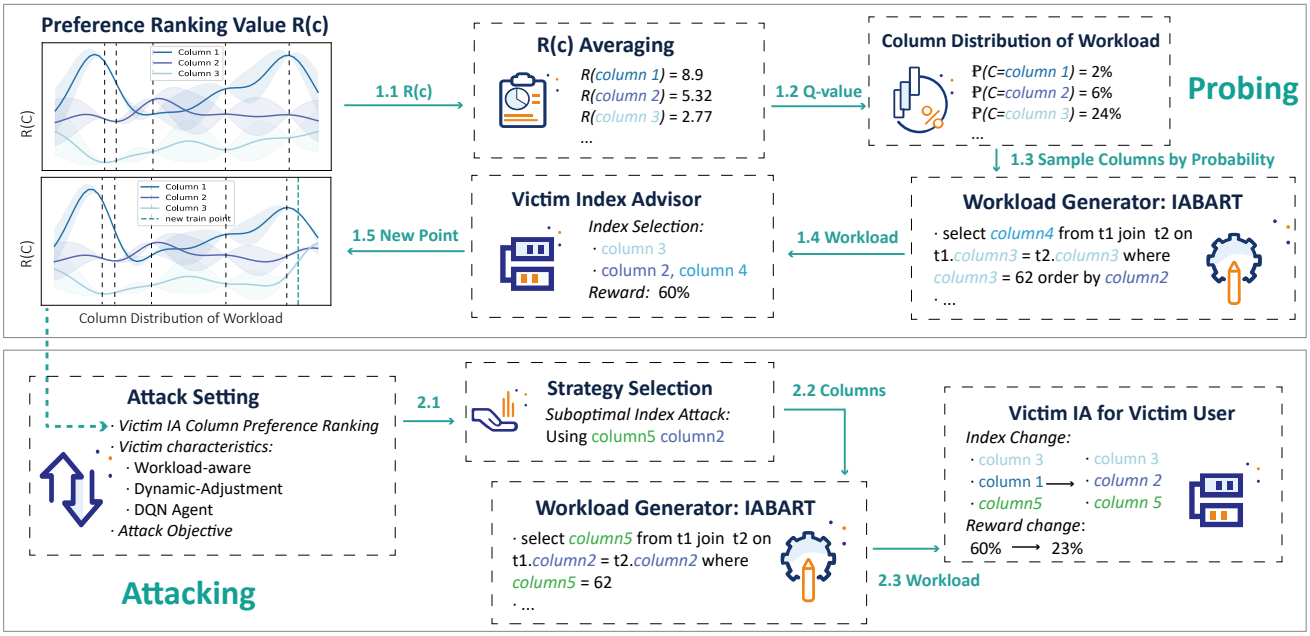

Figure 2: Overview Framework of P^{2}LIA.

Huo, and Huang 2020; Rakhsha et al. 2021; Liu and Lai 2021).

I2: Implicit Constraint Aware SQL Generator. The workload is the only way for the attacker to interact with the victim IA. Therefore, the attacker needs a powerful workload generator that can produce queries that: (1) follow the SQL syntax specifications; (2) can be optimized by indexes, ensuring the attacker can communicate with IAs; and (3) satisfy specific index requirements, enabling the crafting of workloads for particular indexes. However, existing studies on SQL generation either only generate SQL with explicit constraints, such as including column A in "where" clauses (Slutz 1998; Seltenreich, Tang, and Mullender 2020; Bruno, Chaudhuri, and Thomas 2006; Mishra, Koudas, and Zuzarte 2008), or only generate SQL with a single implicit constraint, such as generating SQL with a cardinality of 2000 (which does not generalize to other cardinalities) (Zhang et al. 2022).

I3: Poisoning Attack Strategy. Existing learning-based IAs have various characteristics. They can be categorized based on their workload awareness (Perera et al. 2021; Sadri, Gruenwald, and Lead 2020; Sadri, Gruenwald, and Leal 2020; Kossmann, Kastius, and Schlosser 2022), dynamic adaptability (Perera et al. 2021; Kossmann, Kastius, and Schlosser 2022), and neural network usage (Lan, Bao, and Peng 2020; Sadri, Gruenwald, and Leal 2020; Kossmann, Kastius, and Schlosser 2022), etc. It is still unclear whether and what different attack strategies are needed to cope with these different characteristics of victim IAs.

I4: Assessment of Existing Learning-based IAs. Most existing learning-based IAs are designed mainly to improve index selection performance, with little attention to performance security. There is a lack of research on the performance security of these models and an absence of evaluation of the impact of their design details on performance security.

For the above four issues, we have made the following four-fold contributions:

To address **I1**, we propose **PIDPA** (`Probing-Injecting Data Poisoning Attack`), a two-step poisoning attack framework. To degrade the victim IAs' performance, our framework aims to: (1) obtain the preference ranking for each index in the victim IAs in an opaque-box setting, and (2) use the attack workload to change the preference ranking from the optimal index to worse indexes, i.e., induce models to favor worse indexes. Our framework obtains the preference ranking in the probing stage using a designed probing workload and uses the preference ranking to attack in the attack stage using a designed attack workload. This knowledge acquisition-knowledge utilization paradigm provides a solution to limited knowledge and capabilities in an opaque-box setting.

To address **I2**, we propose **IABART** (`Index Aware BART`). We formulate the SQL generation task with implicit index constraints as a Masked Span Prediction (MSP) task (Song et al. 2019), treating "SQL + Index + Cost Reduction" as a sequence and performing a mask prediction task on "<MASK> + Index + Cost Reduction". **IABART**is built on the backbone of BART (Lewis et al. 2019), a large-scale pre-trained language model (Devlin et al. 2018). Additionally, we propose a progressive training procedure and use a Finite State Machine (FSM) (Zhang et al. 2022) to enforce syntactic constraints. Our **IABART**is well-suited for index constraint-aware SQL generation tasks and this paradigm is general enough to be applied to other constraints.

To address **I3**, we propose several attack strategies to cope

with different learning-based IAs. Specifically, we categorize the preference-changing goals into three types: suboptimal indexes, bad indexes, and a hybrid of suboptimal and bad indexes. We measure the attack effects of these three goals on each type of victim IA and interpret them through the bias-variance influence analysis. Moreover, we measure the attack effect of randomly generated attack workloads, representing unintentional attack behaviors such as burst transactions from ourselves.

To address **I4**, we analyze the training process and performance changes of four existing learning-based IAs during the attack process. We evaluate the strengths and weaknesses of each work's internal design from a security perspective to guide future learning-based IA designs towards more security and stability. The effectiveness of poison attacks in the opaque-box setting is a challenging issue. Since the attacker cannot directly access the parameters of the victim IA, if the injected workload $\tilde{\mathbb{W}}$ is randomly generated, it is difficult to manipulate the IA's training purposely. However, the attacker can interact with the IA by submitting a workload and obtaining the IA's recommended index configurations. Intuitively, the victim IA's responses to different workloads expose the IA's *action preferences*, i.e., which column is more likely to be chosen to build indexes. According to the training paradigm in Section 3, the action preference depends on the IA's parameters ϕ, θ, which is effective on the original workloads $\mathbb{W}^1, \cdots, \mathbb{W}^M$ and reflects the workload characteristics (e.g., frequency of columns in workloads). We can use the action preference to supervise the attack (e.g., inject workloads that can not be optimized using the preferred column indexes), and eventually down-weigh the preferred columns after re-training. As a result, the updated parameters $\tilde{\phi}, \tilde{\theta}$ of the victim IA will be less effective on the original workloads.

EXPERIMENT

In this section, we first report the attack performance of our proposed two strategies on different datasets, data volumes and different victim IAs (Section 6.2). We then investigate the effect of different attacking workload proportion (Section 6.3) and the effect of different optimal, sub-optimal, and bad indexes division hyper-parameters (Section 6.4). Finally, we report some additional experiments, including a performance report on IABART (Section 6.5), a case study on extreme victim workload (Section 6.6), and a possible case of index interaction influence (Section 6.7).

Experimental Setup

Victim Index Advisors. We select four different typical works in learning-based index advisor, DQN (Lan, Bao, and Peng 2020), DRLinda (Sadri, Gruenwald, and Lead 2020; Sadri, Gruenwald, and Leal 2020), SWIRL (Kossmann, Kastius, and Schlosser 2022), DBAbandit (Perera et al. 2021), as our victim models, the design details of which are briefly described in Section 2.1 and the attack part of Section 3.2. We summarize them as shown in Table 1.

We use the open source code of DQN[1], SWIRL[2], DBAbandit[3] and fully retain all the original hyperparameters. We reproduced DRLinda, which has no open source, to the best of our effort according to the design details described in the DRLinda paper (Sadri, Gruenwald, and Lead 2020; Sadri, Gruenwald, and Leal 2020).

Victim Datasets and Workloads. We choose two widely-used open benchmarking datasets, TPC-H[4] which contains 8 tables, 61 columns and 22 query templates, and TPC-DS[5] which contains 25 tables, 429 columns and 99 query templates. We generate workloads using the query templates, treating them as workloads of the victims. In the main experiment in Section 6.2 and 6.3 we fixed the frequency of each query template the same to SWIRL (Kossmann, Kastius, and Schlosser 2022) and in Section 6.6 we will simply argue the effect of extreme queries frequency distribution.

Evaluation Metrics. We use **RR** (Reward Reduction) before and after the attack as follows:

$$RR \triangleq \frac{\mathcal{R}_{\text{before attack}} - \mathcal{R}_{\text{after attack}}}{\mathcal{R}_{\text{before attack}}} \in (-\infty, 1], \qquad (1)$$

where the $\mathcal{R}$ is calculated by cost reduction and mapped to the interval $[0, 100)$. The closer the RR is to 1 means the stronger the effect of the attack. And when the RR value is less than 0, it means the attack has the opposite effect to improve the performance of victim IA.

RR metrics are categorized into Best-RR and Mean-RR based on different training modes of learning-based IAs. Experiments were conducted on a database server and workstation. The program takes approximately 6 hours and is open source.

Main Results

In this section we will measure the effectiveness of our two attack strategies: **S** (*sub-optimal indexes attack*) and **B** (*bad indexes attack*) on different victim IA. For this purpose, we design three control groups for the two attack strategies as follows:

- **O**: *Original.* We used the victim workload as the attacking workload to conduct a poisoning attack. This strategy serves as a control group, allowing us to more scientifically incorporate factors such as updating epochs and model volatility into our analysis.

[1]https://github.com/rmitbggroup/IndexAdvisor
[2]https://github.com/hyrise/index_selection_evaluation
[3]https://github.com/malingaperera/DBABandits
[4]http://www.tpc.org/tpch/
[5]http://www.tpc.org/tpcds/

Table 1: Victim Index Advisor

Victim	Workload-Aware	Dynamic	Agent
DQN	NO	NO	DQN
DRLinda	YES	NO	DQN
DBAbandit	NO	YES	C^2UCB
SWIRL	YES	YES	PPO

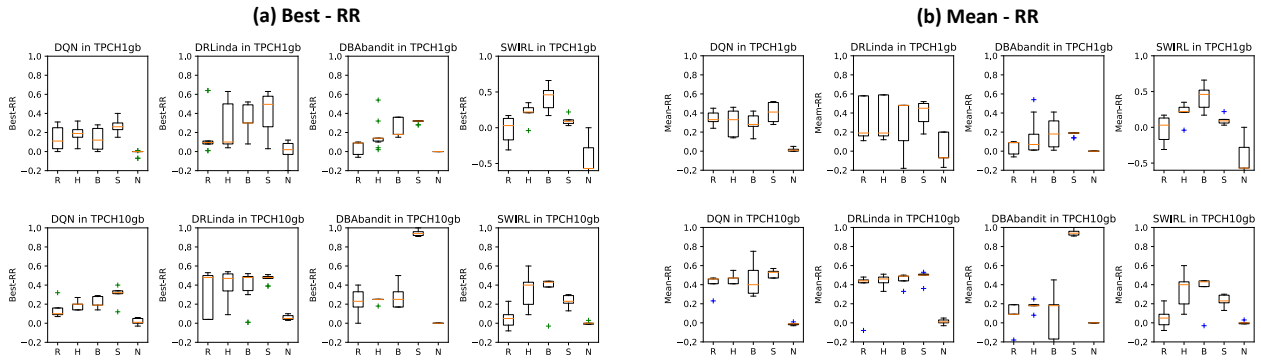

Figure 3: Strategy Selection Experiment Results

- **R**: *Random Indexes Attack.* Using ***IA-BART***, we generated workloads by selecting random index columns. This strategy simulates an attack without prior knowledge and can reflect the impact of unconscious poisoning attacks in real scenarios or serve as an ablation study of our probing method.
- **H**: *Hybrid Indexes Attack.* Using ***IA-BART***, we generated workloads by selecting both suboptimal and bad indexes. This strategy exploits knowledge from the probing section by only ignoring the optimal index column during the index column sampling process.

As shown in Figure 3, we performed poisoning attacks on the four victim models mentioned in section 6.1 using the five attack strategies described above on the *TPCH* and *TPCDS* benchmarks. We selected data volumes of *1GB* and *10GB* for each benchmark, setting the attack workload proportion of the total training workload to 0.5 and then calculated the **Best-RR** and **Mean-RR**.

The poisoning attack works differently for different victims. For SWIRL, the attack directly provides bad parameters to the victim IA, while for DQN, DRLinda, and DBAbandit, the attack essentially provides a bad initial value to the victim IA's parameters.

Conclusion

In this paper, we made the first attempt at poisoning attacks against learning-based IAs. First, we proposed to probe the index preference under the opaque-box setting. Second, we designed attacking workloads to trap IAs within local optimum. Besides, we proposed a query generation method for probing and attacking using large language models (BART). Extensive experiments on different benchmarks against four typical learning-based IAs demonstrated the effectiveness of **PIDPA**.

References

Behzadan, V.; and Munir, A. 2017. Vulnerability of deep reinforcement learning to policy induction attacks. In *Machine Learning and Data Mining in Pattern Recognition: 13th International Conference, MLDM 2017, New York, NY, USA, July 15-20, 2017, Proceedings 13*, 262–275. Springer.

Biggio, B.; Nelson, B.; and Laskov, P. 2012. Poisoning attacks against support vector machines. *arXiv preprint arXiv:1206.6389*.

Biggio, B.; Pillai, I.; Rota Bulò, S.; Ariu, D.; Pelillo, M.; and Roli, F. 2013. Is data clustering in adversarial settings secure? In *Proceedings of the 2013 ACM workshop on Artificial intelligence and security*, 87–98.

Browne, C. B.; Powley, E.; Whitehouse, D.; Lucas, S. M.; Cowling, P. I.; Rohlfshagen, P.; Tavener, S.; Perez, D.; Samothrakis, S.; and Colton, S. 2012. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games*, 4(1): 1–43.

Bruno, N.; and Chaudhuri, S. 2005. Automatic physical database tuning: A relaxation-based approach. In *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, 227–238.

Bruno, N.; Chaudhuri, S.; and Thomas, D. 2006. Generating queries with cardinality constraints for dbms testing. *IEEE Transactions on Knowledge and Data Engineering*, 18(12): 1721–1725.

Chaudhuri, S.; and Narasayya, V. R. 1997. An efficient, cost-driven index selection tool for Microsoft SQL server. In *VLDB*, volume 97, 146–155. San Francisco.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Jagielski, M.; Oprea, A.; Biggio, B.; Liu, C.; Nita-Rotaru, C.; and Li, B. 2018. Manipulating machine learning: Poisoning attacks and countermeasures for regression learning. In *2018 IEEE symposium on security and privacy (SP)*, 19–35. IEEE.

Kornaropoulos, E. M.; Ren, S.; and Tamassia, R. 2020. The Price of Tailoring the Index to Your Data: Poisoning Attacks on Learned Index Structures. *arXiv preprint arXiv:2008.00297*.

Kossmann, J.; Kastius, A.; and Schlosser, R. 2022. SWIRL: Selection of Workload-aware Indexes using Reinforcement Learning. In *EDBT*, 2–155.

Lan, H.; Bao, Z.; and Peng, Y. 2020. An index advisor using deep reinforcement learning. In *Proceedings of the 29th*

ACM International Conference on Information & Knowledge Management, 2105–2108.

LeCun, Y.; Bengio, Y.; and Hinton, G. E. 2015. Deep learning. Nat. 521, 7553 (2015), 436–444. *Crossref, ISI*.

Lewis, M.; Liu, Y.; Goyal, N.; Ghazvininejad, M.; Mohamed, A.; Levy, O.; Stoyanov, V.; and Zettlemoyer, L. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*.

Liu, G.; and Lai, L. 2021. Provably efficient black-box action poisoning attacks against reinforcement learning. *Advances in Neural Information Processing Systems*, 34: 12400–12410.

Ma, Y.; Zhang, X.; Sun, W.; and Zhu, J. 2019. Policy poisoning in batch reinforcement learning and control. *Advances in Neural Information Processing Systems*, 32.

Mishra, C.; Koudas, N.; and Zuzarte, C. 2008. Generating targeted queries for database testing. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, 499–510.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; and Riedmiller, M. 2013. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Paludo Licks, G.; Colleoni Couto, J.; de Fátima Miehe, P.; De Paris, R.; Dubugras Ruiz, D.; and Meneguzzi, F. 2020. SMARTIX: A database indexing agent based on reinforcement learning. *Applied Intelligence*, 50: 2575–2588.

Perera, R. M.; Oetomo, B.; Rubinstein, B. I.; and Borovica-Gajic, R. 2021. DBA bandits: Self-driving index tuning under ad-hoc, analytical workloads with safety guarantees. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*, 600–611. IEEE.

Rakhsha, A.; Radanovic, G.; Devidze, R.; Zhu, X.; and Singla, A. 2020. Policy teaching via environment poisoning: Training-time adversarial attacks against reinforcement learning. In *International Conference on Machine Learning*, 7974–7984. PMLR.

Rakhsha, A.; Zhang, X.; Zhu, X.; and Singla, A. 2021. Reward poisoning in reinforcement learning: Attacks against unknown learners in unknown environments. *arXiv preprint arXiv:2102.08492*.

Sadri, Z.; Gruenwald, L.; and Lead, E. 2020. DRLindex: deep reinforcement learning index advisor for a cluster database. In *Proceedings of the 24th Symposium on International Database Engineering & Applications*, 1–8.

Sadri, Z.; Gruenwald, L.; and Leal, E. 2020. Online index selection using deep reinforcement learning for a cluster database. In *2020 IEEE 36th International Conference on Data Engineering Workshops (ICDEW)*, 158–161. IEEE.

Schlosser, R.; Kossmann, J.; and Boissier, M. 2019. Efficient scalable multi-attribute index selection using recursive strategies. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*, 1238–1249. IEEE.

Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Seltenreich, A.; Tang, B.; and Mullender, S. 2020. SQLsmith: Score list.

Sharma, A.; Schuhknecht, F. M.; and Dittrich, J. 2018. The case for automatic database administration using deep reinforcement learning. *arXiv preprint arXiv:1801.05643*.

Slutz, D. R. 1998. Massive stochastic testing of SQL. In *VLDB*, volume 98, 618–622. Citeseer.

Song, K.; Tan, X.; Qin, T.; Lu, J.; and Liu, T.-Y. 2019. Mass: Masked sequence to sequence pre-training for language generation. *arXiv preprint arXiv:1905.02450*.

Sun, Y.; Huo, D.; and Huang, F. 2020. Vulnerability-aware poisoning mechanism for online rl with unknown dynamics. *arXiv preprint arXiv:2009.00774*.

Whang, K.-Y. 1987. Index selection in relational databases. *Foundations of Data Organization*, 487–500.

Yang, C.; Wu, Q.; Li, H.; and Chen, Y. 2017. Generative poisoning attack method against neural networks. *arXiv preprint arXiv:1703.01340*.

Zhang, L.; Chai, C.; Zhou, X.; and Li, G. 2022. Learnedsqlgen: Constraint-aware sql generation using reinforcement learning. In *Proceedings of the 2022 International Conference on Management of Data*, 945–958.

Zhang, X.; Ma, Y.; Singla, A.; and Zhu, X. 2020. Adaptive reward-poisoning attacks against reinforcement learning. In *International Conference on Machine Learning*, 11225–11234. PMLR.

Zhou, X.; Liu, L.; Li, W.; Jin, L.; Li, S.; Wang, T.; and Feng, J. 2022. Autoindex: An incremental index management system for dynamic workloads. In *2022 IEEE 38th International Conference on Data Engineering (ICDE)*, 2196–2208. IEEE.