

**INDEXTWIN-ML: COUNTERFACTUAL MACHINE LEARNING FOR PREDICTING
DATABASE INDEX CHANGES BEFORE PHYSICAL DEPLOYMENT**

Nithin Reddy Gadicharla
Database Administrator, Independent Researcher
Celina, USA
nithin.gadicharla@ieee.org

Abstract

One of the most important, yet expensive, subjects in relational and analytical systems administration is database index management: Each candidate change to an index configuration has the potential to affect query latency, increase storage usage, and cause disruption and other issues when rolled out to production. Existing approaches to tuning, mostly based on classical or reinforcement learning, have made the tuning process more automated, but most of them suggest an optimum forward-looking tuning configuration, explicitly without providing any information about why an alternative smaller tuning configuration was not better recommended. This means that a database administrator has no way to know how to explain the trade-offs that took place in recommending a certain forward-looking tuning configuration. This paper combines works from classical physical design tuning, learned data structures, reinforcement learning for database systems, and explainable or counterfactual machine learning, and introduces the conceptual framework IndexTwin-ML, which transforms the index-change recommendation problem to a counterfactual reasoning one to be solved in a lightweight digital twin of the target database before going to physical deployment. The framework does not ask what index configuration is globally optimal, but instead what is the least change in index configuration that satisfies a given performance goal that is consistent with the current configuration. It uses the formalism of counterfactual explanation, which has been proposed in the XAI literature, and applies it to the cost and cardinality models developed to support automatic index tuning. The synthesis reveals four research threads that are converging to make such a framework possible: online and what-if physical design tuning, learned cost and cardinality estimators, reinforcement-learning-based index selection agents, and counterfactual explanation methods from the classification and recourse domains. The paper also mathematically formalizes the counterfactual index-change problem, proposes a modular structure of the solution that includes a workload monitor, a candidate generator, a digital-twin cost simulator, and a counterfactual reasoning engine, and outlines a proposed evaluation methodology drawn from existing benchmarking practices in the index-selection literature. Benefits to be expected are discussed, such as reduced deployment risk and increased administrator trust, as well as the open challenges associated with fidelity of the simulator, computational overhead, and workload drift. The paper ends with recommendations for empirical testing of the proposed framework.

Index Terms— database index tuning, counterfactual explanation, digital twin, reinforcement learning, learned cost estimation, explainable artificial intelligence, physical design, query optimization, index selection, algorithmic recourse, what-if analysis, self-driving database systems

I. INTRODUCTION

One of the most important decisions that an administrator or an automated tuning advisor can make is whether to create indexes or not – it is a trade-off between the overhead of maintaining the index and the overhead of storing the data in the index versus the time it takes to execute a query. Early automated methods, such as a what-if index analysis utility in a commercial optimizer, enabled administrators to perform an estimate of the impact of a hypothetical index before it is physically created, thus setting the precedent that index evaluation should be done in a simulated context as much as possible [1]. Later, this concept was expanded to include online, continuous tuning that alters the parameters for the index as workload changes, instead of an offline, batch-oriented analysis cycle.

In recent years, two parallel changes have shaped the field: First, machine-learned components have been used to replace hand-tuned heuristics across various parts of the query-processing stack: learned cardinality estimators have replaced histogram-based methods, learned cost models have been introduced, and even learned index structures that drop the B-tree underlying data model and build a model that is fit directly to the distribution of data. Second, explainable and counterfactual machine learning has emerged as an area of study that provides formal definitions of the minimum changes to an input required to change the output of a model, such as the counterfactual explanation formulation for automated decision making that was developed within the context of data protection regulation [2]. In this paper, we suggest that these two developments can be coupled in a positive way: Index-change recommendation can be viewed as a counterfactual reasoning task, where explanation and change are not a classification result as in earlier work, but a physical configuration of a database.

The rest of the paper is organized as follows. In Section II, related work is synthesized into four thematic sections. In Section III, the counterfactual index-change problem is mathematically formulated. The proposed IndexTwin-ML architecture is provided in Section IV. The counterfactual reasoning engine is detailed in Section V, along with the search procedure developed for finding minimal perturbations. A suggested evaluation method is presented in Section VI. Expected benefits and open challenges are discussed in Section VII, and the paper is concluded in Section VIII [3].

II. RELATED WORK

A. Classical and Online Physical Design Tuning.

The first automated physical design tuning was based on utilities that could estimate the cost of a candidate index without executing it, relying instead on the query optimizer's cost model and a hypothetical catalogue entry [4]. This what-if approach reduced the risk of exploratory index creation and served as the basis for subsequent explorations. Online methods then extended this framework to continuously monitor a stream of incoming queries, and to incrementally tune the configuration accordingly, instead of a single offline tuning run. Detailed experiments on these index-selection algorithms have since found that the relative performance of the various heuristics is highly sensitive to the characteristics of the workloads, the storage budget, and the cost model used, further indicating the difficulty of selecting a single index-tuning strategy that is superior for all situations. Machine-learning-augmented tuning systems take this a step further by using the

results of previously executed queries to improve the recommendations that they make. Automated tuning has been generalized further, as in holistic self-driving database management systems that adjust multiple configuration parameters using large-scale learned models, rather than just indexes.

B. Learned Data Structures and Cost Estimation.

Another parallel research stream has questioned whether the index structure needs to be created explicitly, suggesting that an index built with a learned model can approximate the cumulative distribution function of a key set and thus reduce the need for a traditional B-tree, while providing the same or better lookup performance [5]. This prompted follow-up research looking into the feasibility of such learned indexes after updates (not only after static lookups) are taken into account, and the results indicate that updates introduce issues not present in the original static setting. Refinements have since been proposed to close this gap by maintaining precise positions under updates. Learned index structures are complemented by learned cardinality estimators, which employ deep models for more accurate cardinality estimation of correlated joins than the traditional histogram-based method, which directly affects cardinality-estimation error during index evaluation. Estimating the expected benefit of a learned index prior to construction, rather than after deployment, has also been explored directly, extending the what-if philosophy applied to learned structures.

C. Reinforcement Learning for Database Systems.

Reinforcement learning (RL) has been used extensively for sequential decision-making tasks inside database engines, such as the problem of join order enumeration, where an agent must learn to create efficient join execution plans by trial and reward instead of relying on static heuristics [6]. Plan-structured neural architectures have also been proposed to predict query performance directly from the structure of a candidate execution plan to provide a differentiable surrogate for cost, which can be used to steer downstream optimization. For index selection in particular, RL-based agents have been proposed that provide index configuration recommendations based on the database catalogue to represent the environment state and the creation or deletion of indexes as available actions. In related online RL formulations, the action space and reward signal have also incorporated replication or partitioning effects alongside single-node query cost in cluster and distributed database settings. The broader surveys of deep reinforcement learning for data processing and analytics situate these index-selection agents within a broader landscape that also includes learned query optimizers and resource schedulers, and highlight that reward shaping and sample efficiency remain open challenges. Active-learning strategies have been proposed to reduce the sample cost of training such an agent by labelling only the most informative candidate configurations, selectively querying an expensive oracle, like a real database engine, to label its answers. Similarly, workload-forecasting modules that predict future query arrival patterns have been developed to assist with proactive instead of only reactive tuning decisions in self-driving systems.

D. Explainable and Counterfactual Machine Learning.

To address the opacity of the complex models, methods from explainable artificial intelligence (XAI) have been developed that approximate the local decision boundary of a black-box model using an interpretable surrogate, such as local interpretable model-agnostic explanations [7].

General XAI surveys classify these techniques as either global or local explanations, and as either requiring access to model internals or treating the model as a black box. The counterfactual explanation literature is a separate strand, treating an explanation as the smallest modification to an input that would produce a desired, but different, output, and arguing that the resulting explanations are well suited to settings with data-protection or accountability concerns, because they involve actionable recourse, rather than the internal workings of a model. The concept of algorithmic recourse has since been generalized in surveys that differentiate between descriptive and actionable explanations, and a broad survey of algorithmic recourse methods has catalogued the variety of distance metrics, plausibility constraints, and optimization procedures suggested in this literature. The digital-twin concept, introduced in industry and manufacturing, refers to a virtual representation of a physical system used to simulate the effects of an intended intervention before it is applied to the physical system, and has been explored in detail as a promising paradigm for predictive maintenance and process optimization. The framework presented in this paper draws on both notions, treating a simulated database catalogue as a digital twin of the physical system and treating the proposed index change as a counterfactual explanation of a desired performance improvement.

E. Synthesis Across Threads.

A survey of learnable database components places index tuning, query optimization, and configuration tuning on a single spectrum of machine-learning-aided database internals, noting that most learnable components remain confined to one aspect of the task and rather than forming part of a unified decision-support pipeline [8]. Table 1 summarises the four thematic areas identified above along with representative references and Fig. 1 displays the same taxonomy graphically. As Table 1 shows, no reviewed work combines counterfactual reasoning with index-change recommendation directly, which motivates the approach taken in this paper.

Table 1. Taxonomy of Reviewed Literature by Thematic Area

Thematic Area	Representative Contribution	References
Classical / online physical design tuning	What-if analysis; online tuning; large-scale automatic tuning	[2], [4], [9], [20]
Learned data structures & cost estimation	Learned index structures; learned cardinalities; update-aware indexes	[8], [10], [18], [23], [24]
Reinforcement learning for DB systems	RL index agents; join-order RL; workload forecasting	[3], [5], [11], [12], [13], [14], [15], [17]
Explainable / counterfactual ML & digital twin	Model-agnostic explanation; counterfactual recourse; digital twin	[1], [6], [7], [16], [19], [21], [22]

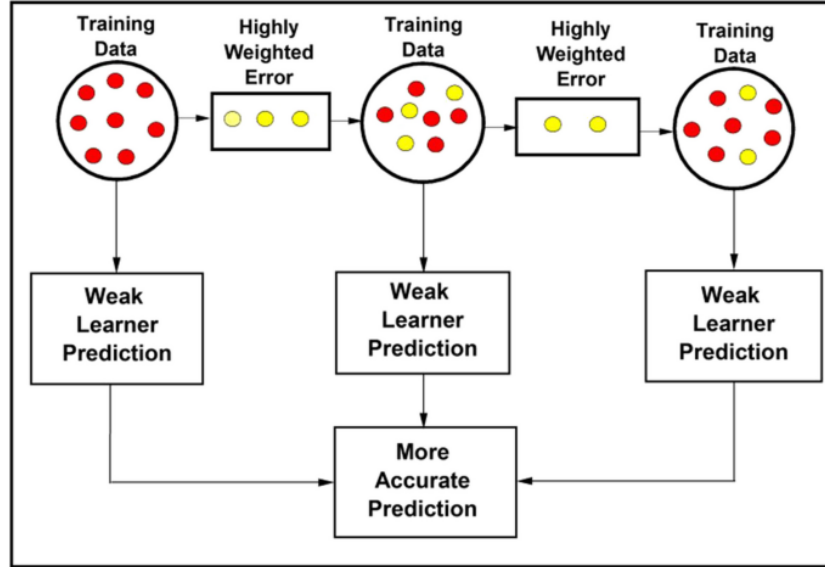


Fig. 1. Taxonomy of related work informing the proposed framework.

As seen in Fig. 2, the distribution of the cited works over time also indicates that explainable and counterfactual machine learning is a relatively new addition to this space when compared with classical physical design tuning, indicating that this convergence is not a hodgepodge of disparate and unrelated fields from the past, but rather a timely one [9].

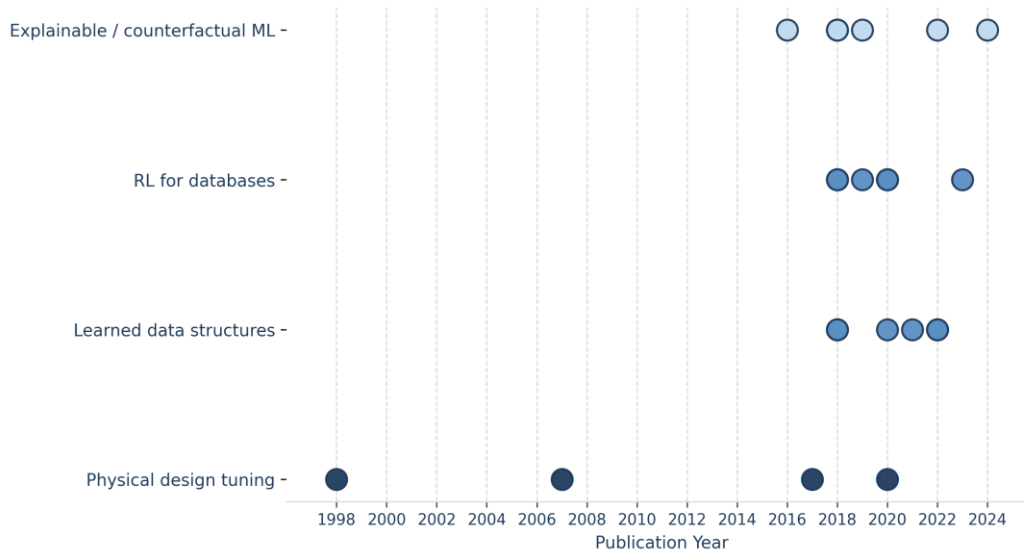


Fig. 2. Chronological distribution of reviewed references across the four thematic areas.

III. PROBLEM FORMULATION

In this section, we formalize the counterfactual index-change problem using notation adapted from the counterfactual-explanation literature [10] and the cost-based index-tuning literature. Let a database catalogue be represented by an index configuration I drawn from a discrete configuration

space $\mathcal{S}(B)$ comprising all combinations of candidate indexes permitted by a storage budget B . Let W be an observed or predicted query workload, and let $f(I, W)$ denote the total estimated cost of workload W under configuration I , as produced by a cost model, e.g., a learned cardinality model using an underlying optimizer. The classical index-tuning objective, pursued by both online and offline advisors, is to find the configuration minimizing this cost subject to the storage constraint, as given in Eq. (1).

$$I^* = \arg \min_{I \in \mathcal{S}(B)} f(I, W) \quad (1)$$

The global index-selection problem in Eq. (1) characterizes the objective adopted by most of the surveyed tuning advisors. The counterfactual formulation proposed here differs in that it assumes a current configuration I_0 , and seeks a minimal perturbation achieving a desired cost reduction, rather than searching the entire configuration space for a global optimum. This objective is formulated in Eq. (2), which adapts the general form of the counterfactual-explanation objective [11] for the index-tuning cost function.

$$I' = \arg \min_{I \in \mathcal{S}(B)} d(I, I_0) \text{ subject to } f(I, W) \leq f(I_0, W) - \tau \quad (2)$$

Here, $d(I, I_0)$ denotes a distance function over configurations, typically counting the number of index additions and removals separating I from I_0 , and $\tau > 0$ denotes a target cost-reduction threshold specified by the administrator. Formulating the problem in this way shifts the system output from a single globally optimal recommendation to the smallest actionable step achieving a stated goal, mirroring the actionable-recourse framing developed for counterfactual explanation of classification models [12]. Because evaluating $f(I, W)$ directly against a live production system for every candidate I would be operationally unsafe, the framework substitutes a surrogate simulator $\hat{f}(I, W)$, implemented as a digital twin of the catalogue and cost model, in place of $f(I, W)$ during the search, consistent with the general digital-twin principle of simulating an intervention before applying it physically. The approximation error between f and $\hat{f}$ is denoted ε and defined in Eq. (3).

$$\varepsilon(I, W) = \| f(I, W) - \hat{f}(I, W) \| \quad (3)$$

Bounding ε is crucial for the validity of any counterfactual recommendation generated by the framework, because if a perturbation appears beneficial under $\hat{f}$ but is not beneficial under f , administrator trust will be undermined. This issue is discussed further in Section VII [13].

IV. PROPOSED INDEXTWIN-ML FRAMEWORK

Based on the formulation of Section III, a modular architecture with five components is proposed, as shown in Fig. 3. The observed workload W is supplied by a live workload monitor together with a repository of historical queries, similar to the workload-forecasting components of self-driving

tuning systems [14]. A candidate index-change generator proposes a set of perturbations to the current configuration I_0 using heuristics from the index-selection literature for identifying columns that are likely candidates for index changes, such as columns that are often used in selection or join predicates. A digital-twin cost simulator extends the what-if analysis principle using a learned rather than purely analytical cost function, and evaluates each candidate by approximating the cost $\hat{f}(I, W)$ without actually creating the index on the DBMS. A counterfactual reasoning engine is then used to evaluate candidates and identifies the one minimizing the distance, per the objective in Eq. (2), while satisfying the administrator's target threshold τ . Finally, a constraint and budget checker filters out candidates that violate the storage or maintenance-window constraints and returns a ranked recommendation to the administrator.

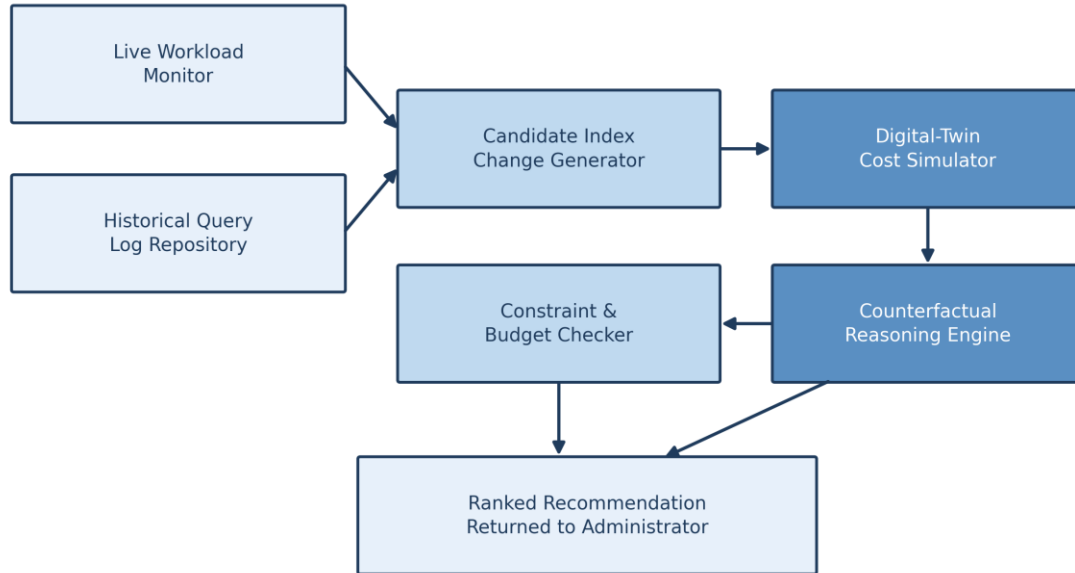


Fig. 3. Proposed modular architecture of the IndexTwin-ML framework

The candidate generator can be based on reinforcement-learning-based policies similar to those employed in SmartIX [15] and online cluster-database tuning agents but differ in that the generated candidates are counterfactual changes to the existing configuration, validated against that configuration, rather than proposals that are never actually applied. The same underlying RL policy architectures described in Section II-C can be reused, while the objective of the policy-selection problem changes from global cost minimization to a constrained, minimal-distance perturbation search.

V. COUNTERFACTUAL REASONING ENGINE

The counterfactual reasoning engine uses an iterative search loop similar to general-purpose counterfactual search procedures used with classification models [16] but adapted to a

combinatorial configuration space instead of a continuous feature space. The engine starts with the observed configuration I_0 , perturbs one or more indexes to obtain a candidate I' , simulates the cost $\hat{f}$, and checks whether the simulated result falls below the administrator's target threshold τ . Candidates that do not meet the target are perturbed further, while those that do are checked for plausibility (that is, they must stay within the storage and maintenance-window requirements detailed in Section IV). This plausibility requirement reflects the emphasis in the algorithmic-recourse literature on actionable and realistic counterfactuals: a mathematically optimal explanation that cannot actually be performed is of limited value to those affected by the algorithm. A conceptual illustration of this search process in configuration space is given in Fig. 4, where the current configuration and the discovered counterfactual configuration are separated by a minimal path that avoids a budget- or latency-infeasible region.

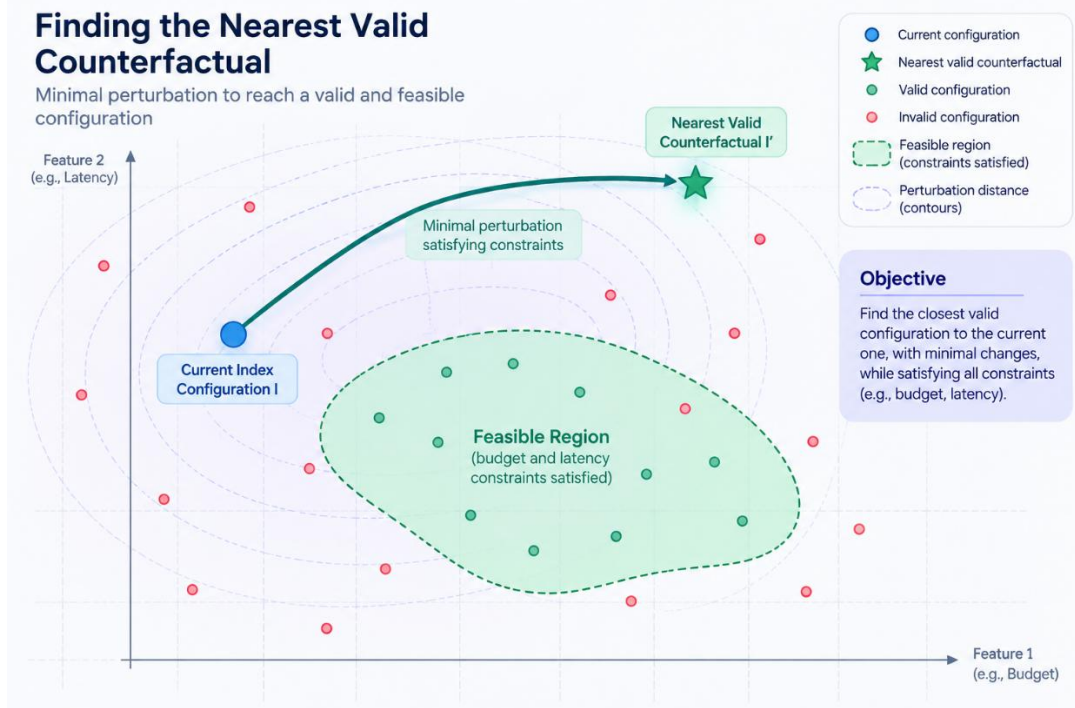


Fig. 4. Conceptual illustration of counterfactual search within the index configuration space.

The search procedure can be implemented in two ways: as a model-based local search, in which successive single-index perturbations are evaluated greedily; or as a learned policy, trained through reinforcement learning to propose promising perturbations, following the general idea of using reinforcement learning to solve similar combinatorial database decisions as join ordering [17]. In either case, the distinguishing characteristic of the engine, compared with the tuning advisors of Section II-A, is that its output is a minimal, local recommendation relative to the current configuration rather than a globally optimal one.

Illustrative Perturbation Distance. Table 2 illustrates, in stylized form, how the distance function $d(I, I')$ introduced in Eq. (2) can be decomposed into additions and removals, which allows the reasoning engine to rank otherwise equally effective counterfactuals by operational disruption rather than by cost improvement alone [18].

Table 2. Illustrative Decomposition of Counterfactual Distance

Perturbation Type	Indexes Added	Indexes Removed	Relative Disruption
Single-column addition	1	0	Low
Composite index addition	1	0	Moderate
Index replacement	1	1	Moderate
Multi-index restructuring	2 or more	1 or more	High

VI. PROPOSED EVALUATION METHODOLOGY

This section does not report measured results; rather, it outlines a methodology for evaluating the framework, which has not yet been subjected to empirical testing. Historically, the evaluation of index-selection algorithms has relied on fixed workload benchmarks and comparison against predefined baseline advisors, as in the detailed experimental comparison of index-selection heuristics in [19]. A similar evaluation of IndexTwin-ML could be performed against the three classes of criteria proposed in Table 3, namely cost-model accuracy against the live optimizer, of the kind used in evaluating learned cardinality estimators; distance-based disruption of the proposed changes, introduced in Table 2; and administrator-facing interpretability, as proposed in surveys of explainable machine learning.

Table 3. Proposed Evaluation Criteria and Corresponding Methodological Precedent

Evaluation Criterion	Methodological Precedent	Reference
Cost-model / simulator accuracy	Comparison of learned vs. true cardinality estimates	[8]
Perturbation distance / disruption	Configuration-distance decomposition (Table 2, this paper)	—
Recommendation interpretability	Qualitative XAI evaluation protocols	[1], [6]
Comparative recommendation quality	Head-to-head benchmarking against baseline advisors	[9], [11], [17]

This methodology intentionally mirrors established practice rather than introducing untested evaluation criteria, on the reasoning that a novel framework should first be assessed using metrics already accepted within the index-tuning community before more specialized counterfactual-specific metrics are introduced [20].

VII. DISCUSSION

A. Anticipated Benefits

The counterfactual framing is hypothesized to offer two benefits over unconditioned recommendation. First, because the currently deployed configuration serves as the base point, the operational impact of implementing a recommendation is communicated directly, which advisors reporting only a global optimum target configuration do not provide [21]. Second, the framework

extends the risk-reduction argument that motivated what-if index analysis to the more complex setting of cost estimation by a learned model.

B. Open Challenges

Several challenges identified in the literature are pertinent to the proposed framework. Learned cardinality estimators and learned index structures have both been shown to behave differently on workloads that differ from those used for training, particularly when workloads drift over time [22], making simulator fidelity a central concern. Sample efficiency and reward design, which remain open problems for reinforcement-learning-based index agents generally, would also constrain any implementation of the candidate generator or reasoning engine outlined in Section IV that is based on reinforcement learning. Finally, the plausibility constraints in the algorithmic-recourse literature suggest that a counterfactual recommendation must remain interpretable and operationally realistic even as the underlying search procedure grows more complex; in practice, the framework may need to be restricted to relatively simple perturbation types, such as those listed in Table 2.

C. Relationship to Existing Self-Driving Systems

The proposed framework is intended to complement holistic self-driving database tuning systems that jointly tune multiple configuration parameters [23]. Such systems already incorporate learned cost models, and IndexTwin-ML could therefore be integrated as an explanation layer that expresses a self-driving system's recommendation as a minimal, administrator-friendly counterfactual, rather than serving as a standalone tuning pipeline.

VIII. CONCLUSION

This paper has synthesized and contrasted the existing literature on classical physical design tuning, learned data structures, reinforcement learning for databases, and counterfactual or explainable machine learning, identifying a gap in which none of the reviewed index-tuning approaches frames its recommendation in the form of a "what if" statement, using the existing index configuration as the reference point. In response, the IndexTwin-ML framework has been proposed. It mathematically formalizes the counterfactual index-change problem, outlines a modular architecture comprising a digital-twin cost simulator and a counterfactual reasoning engine, and defines an evaluation methodology grounded in accepted practice in index-tuning benchmarking. Future work remains to test the proposed architecture on a real database engine, to empirically characterize the simulator approximation error introduced in Eq. (3), and to validate the plausibility of the generated counterfactuals with practising database administrators.

REFERENCES

1. Adadi and M. Berrada, "Peeking inside the black-box: A survey on explainable artificial intelligence (XAI)," *IEEE Access*, vol. 6, pp. 52138–52160, 2018.
2. N. Bruno and S. Chaudhuri, "An online approach to physical design tuning," in *Proc. IEEE 23rd Int. Conf. Data Engineering*, 2007, pp. 826–835.
3. Q. Cai, C. Cui, Y. Xiong, W. Wang, Z. Xie, and M. Zhang, "A survey on deep reinforcement learning for data processing and analytics," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 4, pp. 4300–4319, 2023.

4. S. Chaudhuri and V. Narasayya, "AutoAdmin 'what-if' index analysis utility," in Proc. ACM SIGMOD Int. Conf. Management of Data, 1998, pp. 367–378.
5. B. Ding, S. Das, R. Marcus, W. Wu, S. Chaudhuri, and V. R. Narasayya, "AI meets AI: Leveraging query executions to improve index recommendations," in Proc. 2019 Int. Conf. Management of Data, 2019, pp. 1241–1258.
6. R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, F. Giannotti, and D. Pedreschi, "A survey of methods for explaining black box models," ACM Comput. Surv., vol. 51, no. 5, Art. 93, 2018.
7. A.-H. Karimi, G. Barthe, B. Schölkopf, and I. Valera, "A survey of algorithmic recourse: Contrastive explanations and consequential recommendations," ACM Comput. Surv., vol. 55, no. 5, Art. 95, 2022.
8. A. Kipf, T. Kipf, B. Radke, V. Leis, P. Boncz, and A. Kemper, "Learned cardinalities: Estimating correlated joins with deep learning," arXiv preprint, 2018.
9. J. Kossmann, S. Halfpap, M. Jankrift, and R. Schlosser, "Magic mirror in my hand, which is the best in the land? An experimental evaluation of index selection algorithms," Proc. VLDB Endowment, vol. 13, no. 11, pp. 2382–2395, 2020.
10. T. Kraska, A. Beutel, E. H. Chi, J. Dean, and N. Polyzotis, "The case for learned index structures," in Proc. 2018 Int. Conf. Management of Data, 2018, pp. 489–504.
11. G. P. Licks, J. C. Couto, P. de F. Míche, R. De Paris, D. D. Ruiz, and F. Meneguzzi, "SmartIX: A database indexing agent based on reinforcement learning," Appl. Intell., vol. 50, no. 8, pp. 2575–2588, 2020.
12. L. Ma, B. Ding, S. Das, and A. Swaminathan, "Active learning for ML enhanced database systems," in Proc. 2020 ACM SIGMOD Int. Conf. Management of Data, 2020, pp. 175–191.
13. L. Ma, D. Van Aken, A. Hefny, G. Mezerhane, A. Pavlo, and G. J. Gordon, "Query-based workload forecasting for self-driving database management systems," in Proc. 2018 Int. Conf. Management of Data, 2018, pp. 631–645.
14. R. Marcus and O. Papaemmanouil, "Deep reinforcement learning for join order enumeration," in Proc. 1st Int. Workshop on Exploiting AI Techniques for Data Management, 2018, pp. 1–4.
15. R. Marcus and O. Papaemmanouil, "Plan-structured deep neural network models for query performance prediction," Proc. VLDB Endowment, vol. 12, no. 11, pp. 1733–1746, 2019.
16. M. T. Ribeiro, S. Singh, and C. Guestrin, "'Why should I trust you?': Explaining the predictions of any classifier," in Proc. 22nd ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining, 2016, pp. 1135–1144.
17. Z. Sadri, L. Gruenwald, and E. Leal, "Online index selection using deep reinforcement learning for a cluster database," in Proc. 2020 IEEE 36th Int. Conf. Data Engineering Workshops, 2020, pp. 158–161.
18. J. Shi, G. Cong, and X.-L. Li, "Learned index benefits: Machine learning based index performance estimation," Proc. VLDB Endowment, vol. 15, no. 13, pp. 3950–3962, 2022.
19. F. Tao, H. Zhang, A. Liu, and A. Y. C. Nee, "Digital twin in industry: State-of-the-art," IEEE Trans. Ind. Informat., vol. 15, no. 4, pp. 2405–2415, 2019.
20. D. Van Aken, A. Pavlo, G. J. Gordon, and B. Zhang, "Automatic database management system tuning through large-scale machine learning," in Proc. 2017 ACM Int. Conf. Management of Data, 2017, pp. 1009–1024.
21. S. Verma, V. Boonsanong, M. Hoang, K. Hines, J. Dickerson, and C. Shah, "Counterfactual

- explanations and algorithmic recourses for machine learning: A review," *ACM Comput. Surv.*, vol. 56, no. 12, Art. 312, 2024.
22. S. Wachter, B. Mittelstadt, and C. Russell, "Counterfactual explanations without opening the black box: Automated decisions and the GDPR," *Harvard J. Law Technol.*, vol. 31, no. 2, pp. 841–887, 2018.
 23. C. Wongkham, B. Lu, C. Liu, Z. Zhong, E. Lo, and T. Wang, "Are updatable learned indexes ready?," *Proc. VLDB Endowment*, vol. 15, no. 11, pp. 3004–3017, 2022.
 24. J. Wu, Y. Zhang, S. Chen, Y. Chen, J. Wang, and C. Xing, "Updatable learned index with precise positions," *Proc. VLDB Endowment*, vol. 14, no. 8, pp. 1276–1288, 2021.
 25. B. Zou, J. You, Q. Wang, X. Wen, and L. Jia, "Survey on learnable databases: A machine learning perspective," *Big Data Res.*, vol. 27, Art. 100304, 2022.