QDBO: A Real-time Quantum-augmented Database System Optimizer

Hanwen Liu
University of Southern California
Los Angeles, CA, USA
hanwen_liu@usc.edu

Abhishek Kumar
University of Southern California
Los Angeles, CA, USA
akumar17@usc.edu

Federico Spedalieri
Capital One
Los Angeles, CA, USA
federico.spedalieri@capitalone.com

Ibrahim Sabek
University of Southern California
Los Angeles, CA, USA
sabek@usc.edu

ABSTRACT

Quantum annealers offer a promising sampling-based computing paradigm for large-scale combinatorial optimization. However, directly using them to solve database optimization problems in real-time scenarios remains challenging due to strict hardware constraints and the opacity of commercial hybrid solvers. To address these challenges, we present *QDBO*, a white-box quantum-augmented database optimizer that directly leverages low-level quantum sampling on a quantum annealer. It first performs an initial reduction to produce embeddable QUBO instances via multiple mapping strategies that trade off efficiency and quality. It then applies an online iterative relaxation correction procedure that uses sampling feedback to correct the reduced QUBO across iterations, enabling controllable quality improvements under varying time budgets. We integrate *QDBO* with the PostgreSQL query optimizer to construct *Q^2O-QDBO* for join order optimization. Evaluations on the Join Order Benchmark (JOB) and the Cardinality Estimation Benchmark (CEB) show that *Q^2O-QDBO* improves 62 out of 113 JOB queries, and achieves up to 49.36× execution-time speedup and 36.88× end-to-end speedup on the CEB workload. Evaluations on synthetic queries with up to 50 relations show that *Q^2O-QDBO* achieves significantly less end-to-end latency while maintaining similar solution quality compared with black-box quantum solvers. To demonstrate generality, we further integrate *QDBO* with PostgreSQL for index selection on TPC-H, where *QDBO* achieves up to 2.64× workload speedup over the empty-index baseline and outperforms classical and black-box solver baselines. These results demonstrate that using *QDBO* to solve large-scale database optimization problems in real-time settings is promising.

PVLDB Reference Format:
Hanwen Liu, Abhishek Kumar, Federico Spedalieri, and Ibrahim Sabek. QDBO: A Real-time Quantum-augmented Database System Optimizer. PVLDB, 19(11): 3606 - 3620, 2026.
doi:10.14778/3836663.3836712

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/ihanwen99/VLDB26-QDBO.

Proceedings of the VLDB Endowment, Vol. 19, No. 11 ISSN 2150-8097.
doi:10.14778/3836663.3836712

1 INTRODUCTION

The performance of database systems highly depends on the efficiency and quality of solving multiple optimization problems, including join ordering (e.g., [35, 56, 58, 95, 98]), transaction scheduling (e.g., [13, 15]), multi-query optimization (e.g., [41, 83]), and index tuning [1, 12, 93]. For a small-scale problem, the exhaustive search (e.g., a DP-based algorithm [82] for the join order problem) can be applied to obtain an optimal solution. However, most database optimization problems are inherently combinatorial and NP-hard [16]. As data volumes increase rapidly and the query workload becomes more complex, the difficulty of solving these large-scale optimization tasks grows exponentially. To balance the solution quality and optimization overhead, many heuristic methods [33, 59, 64] have been proposed to navigate this trade-off. While such methods are effective in practice, they shift the focus to producing a high-quality solution quickly, where accepting a suboptimal result is justified if it substantially reduces optimization time. For example, for larger queries, PostgreSQL [69] switches to evolutionary genetic query optimization (GEQO) [70], which enables fast convergence at the cost of weaker optimality. Learning-based techniques, including machine and reinforcement learning [35, 56–58, 95, 98] and more recent large language model-based methods [53, 54, 84], have been proposed as promising alternatives. However, they continue to face significant challenges in real database settings, including high pre-training cost [94], dependence on large training datasets [57], and the need for frequent retraining under workload drift [75]. These limitations motivate alternative compute paradigms that can deliver high-quality solutions without such overhead.

Fortunately, recent advances in quantum hardware (e.g., [34, 73, 76]) have positioned quantum computing as a powerful and promising computational paradigm. Under widely accepted complexity-theoretic assumptions [3, 5, 29], quantum systems are believed to surpass classical systems in computational capability, with the potential to explore exponentially large search spaces more efficiently for certain optimization problems [4, 17]. Quantum cloud platforms such as D-Wave Leap [20] further improve accessibility. The recent quantum hardware, namely quantum annealers [86] based on quantum annealing, supports more than 5000 qubits, suggesting the potential to host large-scale database optimization problems. However, quantum computing systems remain at an early stage of development and still face substantial hardware-related challenges [71], especially the sparse hardware topologies (e.g., Pegasus [23] and Zephyr [9]). Such limited qubit connectivity hinders the direct solving of complex database optimization problems on current quantum hardware. For example, recent approaches faithfully encode the original database optimization problems into

Quadratic Unconstrained Binary Optimization (QUBO) formulations [26, 28, 60, 80, 88], which preserve the semantics of the original problem. However, this faithful encoding often results in highly dense interaction graphs, producing over-dense QUBOs that are difficult to embed and run on sparsely connected quantum hardware (**C1. QUBO overcomplexity**). To overcome this bottleneck, several hybrid quantum-classical solvers [19, 66] have been proposed to leverage the complementary strengths of both paradigms: `classical` processors handle orchestration, while `quantum` annealers focus on sampling and exploring complex energy landscapes that are otherwise computationally prohibitive. This progress opens the door to integrating quantum computing into operational database systems, as demonstrated by early prototypes such as Q^2O [52] and Q^2O-NL [47]. However, applying these commercial black-box solvers to database optimization introduces several unique challenges, especially in a *real-time* setup:

C2. Black-box Workflow and Service-level Constraints. In current deployments, commercial quantum solvers are accessed as remote black-box services. The typical workflow is to formulate the database optimization problem in a solver-specific input format (e.g., QUBO for the BQM solver [19], or a Constrained Quadratic Model for the CQM solver [19]), submit the instance, and wait for the solver to return the final result (e.g., [47, 51, 72, 80]). These solvers often impose service-level latency floors (e.g., the CQM solver enforces a minimum 5 s response time), making it difficult to integrate them flexibly into a latency-sensitive query optimizer.

C3. Transparency and Fine-grained Control. The current black-box workflow restricts fine-grained control over the hybrid quantum-classical optimization process (e.g., qubit-variable mapping) and limits flexibility for tuning. In particular, the solver cannot easily adjust its strategy using problem-specific semantic information (e.g., customizing the variable-qubit mapping based on the join order problem and formulation). Consequently, this impedes the design of more efficient algorithms for large-scale database optimization.

C4. Budget-Quality Controllability and Anytime Planning. In practical database optimization scenarios, an optimizer may face variable time budgets across different optimization tasks, or even for the same task under different quality requirements. Anytime planning aims to effectively utilize the available budget, return a usable plan quickly under tight budgets, and improve plan quality as more budget becomes available. However, existing black-box solvers are mostly designed to solve optimization problems in a *one-shot* manner and do not explicitly support iterative improvement as the budget increases. For example, NL-Solver [66] allows users to adjust the time budget for end-to-end optimization latency, yet users may encounter scenarios where the assigned budget is either too short (i.e., it returns a substantially worse solution) or too long (i.e., the solution has already stopped improving early). This shortcoming makes these solvers ill-suited to meet the demands and expectations of real-world planning scenarios.

To address Challenges **C1-C4** and enable practical quantum computing in *real-time* database environments, we propose *QDBO*, a white-box, sampling-centric quantum-augmented optimization framework for large-scale database optimization problems. Unlike existing black-box hybrid solvers that encapsulate all quantum-classical interactions behind opaque orchestration layers, *QDBO* exposes and directly controls the lowest-level operations of the hybrid workflow, namely quantum sampling on a quantum annealer and the associated parameterization of the underlying Ising model, while solving database optimization problems.

To cope with the sparse connectivity of current quantum hardware (Challenge C1), *QDBO* incorporates database problem-aware QUBO reduction and embedding techniques that explicitly encode problem characteristics, such as decision variable roles and semantic constraints, into the mapping between logical variables and physical qubits. Rather than relying on solver-internal chaining or decomposition strategies [19], *QDBO* embeds problem structure (e.g., join tree structure enforcement in the join order problem) directly into low-level operations, including variable-qubit mapping and correlation selection, enabling scalable execution while preserving semantic correctness. By interacting with the quantum annealer at the sampling level (the same primitive used internally by commercial hybrid solvers), *QDBO* harnesses quantum exploration while avoiding the additional orchestration overhead and service-level constraints imposed by black-box solver interfaces (addressing Challenge C2). This white-box access also enables *QDBO* to observe intermediate sampling results and react immediately, facilitating tight integration with latency-sensitive database optimizers. Moreover, *QDBO* provides fine-grained control over how problem characteristics influence optimization behavior, including customized variable-qubit mappings and selective correlation relaxation. This allows database problem semantics to guide low-level hybrid operations that are typically inaccessible in existing solvers, addressing the lack of transparency and tunability highlighted in Challenge C3. Finally, *QDBO* embeds an iterative, feedback-driven refinement mechanism directly into the sampling loop, using lightweight *gradient-based* updates to progressively bias future samples toward higher-quality solutions. This design enables explicit budget-quality tradeoffs and anytime optimization behavior (addressing Challenge C4). Together, these design choices move QUBO reduction and sample-and-correct optimization from the *offline* regime where they are typically studied [7, 27, 32, 87, 99] into the *online, real-time* regime required for database query optimization. The framework is heuristic-driven and does not aim at a general theoretical guarantee; we provide empirical validation through end-to-end experiments on an actual database system (Section 7).

To evaluate *QDBO* in end-to-end database optimization scenarios, we integrate it with an actual database system (PostgreSQL's query optimizer [69]) to re-implement an earlier quantum-augmented query optimizer prototype, namely *Q^2O* [52], for join order optimization [82]. We evaluate the resulting system, *Q^2O-QDBO*, on join-intensive workloads, including the Join Order Benchmark (JOB) [46] and the Cardinality Estimation Benchmark (CEB) [63]. On JOB, *Q^2O-QDBO* improves 62 out of 113 queries. On CEB, it achieves up to 49.36× execution-time speedup and 36.88× end-to-end speedup. Additional experiments on synthetic queries with up to 50 relations show that *Q^2O-QDBO* achieves significantly less end-to-end optimization latency while maintaining comparable solution quality to black-box quantum solvers. To further demonstrate the generality of *QDBO* beyond join ordering, we additionally apply it to index selection on TPC-H, where it achieves up to a 2.64× workload speedup over the empty-index baseline. While join order optimization and index selection serve as representative case studies, *QDBO* is designed as a general and extensible framework that

can support a wide range of database optimization tasks beyond these prototypes (e.g., transaction scheduling), demonstrating its broader applicability to real-time database systems.

We summarize our contributions as follows:

(1) We introduce *QDBO*, the first white-box, hybrid quantum-augmented database optimization framework that avoids black-box solver constraints and exposes low-level hybrid operations for large-scale database optimization.
(2) We propose three variable-qubit mapping methods that enable scalable execution of over-dense QUBOs on quantum annealers, including a problem-specific semantic mapping that leverages database characteristics during embedding.
(3) We design an iterative correlation relaxation algorithm that uses gradient descent to correct the reduced QUBO, achieving a better trade-off between latency and solution quality.
(4) We integrate *QDBO* with an actual query optimizer and demonstrate significant improvements on two case studies, join order optimization and index selection.

2 PRELIMINARIES

In this section, we first review the background of quantum annealing and existing quantum solvers, then discuss recent advances in correlation relaxation for QUBO formulations, and finally present the QUBO representations for the two case studies considered in this paper: join order optimization and index selection.

Quantum Annealing (QA). QA is a computational paradigm that leverages quantum hardware to solve complex optimization problems [39]. Unlike classical simulated annealing (SA) [90], which relies on thermal fluctuations to escape local minima, QA exploits quantum tunneling to traverse energy barriers, increasing the chance of reaching near-global optima [2, 21]. QA targets discrete optimization problems that can be formulated as a Quadratic Unconstrained Binary Optimization (QUBO) problem [55] or, equivalently, an Ising model [36]. A QUBO over binary variables $x \in \{0,1\}^n$ maps to an Ising form over spin variables $s_i = 2x_i - 1 \in \{-1,+1\}$ with energy:

$$E(x) \;=\; E_{\text{Ising}}(\vec{s}) \;=\; \sum_i h_i\, s_i + \sum_{i<j} J_{ij}\, s_i\, s_j \tag{1}$$

where $h = \{h_i\}$ are local biases and $J = \{J_{ij}\}$ are pairwise couplings. In QA, qubits encode the spins; starting from a quantum superposition induced by a transverse field, the system slowly turns off the quantum fluctuations while turning on the biases and couplings [77], settling into low-energy configurations that encode solutions to the original problem.

Quantum Solvers. Quantum solvers fall into two categories: native quantum annealers [18, 86] and hybrid quantum-classical solvers [19, 66]. Native annealers, e.g., D-Wave Advantage2 [18], host thousands of superconducting qubits with pairwise connectivity of maximum degree 20, and expose low-level quantum sampling. They can only execute `embeddable` QUBOs whose variables and couplings fit within the qubit budget and the device's native connectivity. A QUBO is overly dense if any variable interacts with more variables than the device's maximum coupler degree (e.g., 20 on Advantage2); such instances cannot be embedded directly. Hybrid solvers, including D-Wave's BQM, CQM [19], and Nonlinear [66] solvers, sidestep this limit by classically decomposing the problem and delegating subproblems to the annealer, but they operate as black boxes, providing neither transparency into their internal mechanisms nor sufficient flexibility for performance tuning.

Correlation Relaxation for QUBO. To make overly dense QUBO instances runnable on current annealers, correlation relaxation [20, 22, 40] prunes selected pairwise couplings J_{ij} in Equation 1 while retaining the full set of decision variables. A common strategy ranks pairwise terms in J by their contribution to the Hamiltonian energy [6]. However, correlations with relatively small energy weights may encode essential semantic constraints. For instance, in join order optimization [79, 80], low-energy correlations enforce validity constraints that prevent duplicate joins or infeasible join structures; removing them may lead to invalid query plans. Energy-based criteria alone are therefore insufficient for database optimization tasks, as they risk violating essential semantic constraints.

QUBO Formulation for Join Order Optimization. As shown in Section 6, we use the join order optimization problem as the first case study to evaluate our proposed quantum-based database optimization framework. We adopt the same QUBO formulation for join order optimization as the widely used approach in [80], which minimizes an energy function composed of three groups of variables: V, PRED, and SLACK. Specifically, let R be the number of relations and $L = R - 1$ be the number of join steps in a left-deep join tree. We use binary variables $x_{rj} \in \{0,1\}$ to indicate whether relation r participates in join step j, and $y_{pj} \in \{0,1\}$ to indicate whether predicate p is applicable at join step j. With a sufficiently large penalty weight A, the overall QUBO objective is:

$$E(x,y) \;=\; A\big(E_{\mathsf{V}} + E_{\mathsf{PRED}}\big) \;+\; E_{\mathsf{SLACK}} \tag{2}$$

Here, V enforces structural validity of a left-deep join tree via constraints on x_{rj} in two parts $E_{\mathsf{V}} = E_{\mathsf{V1}} + E_{\mathsf{V2}}$: (1) each join step j must involve exactly j+1 relations, captured by $E_{\mathsf{V1}} = \sum_{j=1}^{L}\big((j{+}1) - \sum_{r=1}^{R} x_{rj}\big)^2$, and (2) relation participation must be monotone across steps (once a relation appears, it remains present), captured by $E_{\mathsf{V2}} = \sum_{r=1}^{R}\sum_{j=2}^{L} x_{r,j-1}(1 - x_{rj})$. PRED ties predicate activation to the current join step by allowing a predicate variable to be active only when both endpoint relations of predicate p are present at step j, i.e., $E_{\mathsf{PRED}} = \sum_p \sum_j y_{pj}\big(2 - x_{\mathrm{Rel}_1(p),j} - x_{\mathrm{Rel}_2(p),j}\big)$, where $\mathrm{Rel}_1(p)$ and $\mathrm{Rel}_2(p)$ denote the two relations connected by p. SLACK encodes the cost objective using auxiliary binary *slack* variables that make the cost approximation QUBO-compatible. Concretely, we approximate logarithmic intermediate cardinalities and penalize threshold violations via a quadratic form $E_{\mathsf{SLACK}} = \sum_{t,j} \theta_t \big(\mathrm{Buffer}_{tj} - \mathrm{LogIntCard}(j)\big)^2$, where $\mathrm{LogIntCard}(j)$ is computed from previously defined (x,y), and the buffer is represented with binary slack variables $s_{tij} \in \{0,1\}$ as $\mathrm{Buffer}_{tj} = \sum_{i=1}^{N} 2^{i-1}\, s_{tij}$. Intuitively, V guarantees a valid join tree, PRED ties predicate activation to that tree, and SLACK encodes the cost logic for join order optimization.

QUBO Formulation for Index Selection. As a second case study, we consider the index selection problem and adopt the QUBO formulation from [42]. Given a database instance, a query workload $\mathcal{Q}$, a candidate index set $\mathcal{I} = \{I_1, \ldots, I_n\}$, and a storage budget W, the selector chooses one index combination to be installed for the entire workload. Each candidate index I_i is associated with a binary decision variable $x_i \in \{0,1\}$ indicating whether I_i is selected, a storage weight w_i, and a workload-level profit p_i. The resulting

budget-constrained index selection problem is

$$\max_{x\in\{0,1\}^n} \sum_{i=1}^{n} p_i x_i \quad \text{s.t.} \quad \sum_{i=1}^{n} w_i x_i \le W. \tag{3}$$

Following [42], we compile Equation 3 into a QUBO using binary slack variables that encode the unused budget. Let $s_0, \ldots, s_{m-1} \in \{0,1\}$ with $m = \lceil \log_2(W+1) \rceil$. The overall QUBO objective is

$$E(x,s) = -\sum_{i=1}^{n} p_i x_i + P\left(\sum_{i=1}^{n} w_i x_i + \sum_{k=0}^{m-1} 2^k s_k - W\right)^2, \tag{4}$$

where P is a sufficiently large penalty coefficient. The linear term rewards high-profit indexes, while the quadratic penalty enforces the storage budget. Intuitively, the index decision variables x encode which indexes to install, and the slack variables s ensure that any feasible configuration can drive the penalty term to zero.

3 SYSTEM OVERVIEW

In this section, we first formalize database optimization in the context of *QDBO* (Section 3.1), and then present an overview of *QDBO*'s workflow and key components (Section 3.2).

3.1 Problem Formulation

Our overarching goal is to solve database optimization problems using a quantum annealer and decode its output into concrete database configurations (e.g., join orders or index combinations) that improve query execution performance. However, directly solving the original QUBO parameterized by h and J (i.e., the direct QUBO formulation for the given database optimization problem) on a quantum annealer is challenging due to hardware constraints such as limited qubit connectivity and budget. To address these challenges, we decompose the optimization into two complementary phases: *Initial QUBO Reduction*, which renders the problem executable on existing quantum hardware, and *Online Relaxation Correction*, which iteratively refines the reduced formulation.

Phase 1: Initial QUBO Reduction. In this phase, we transform the original (dense) QUBO $\mathbf{Q}$ into a reduced (simplified) QUBO $\widetilde{\mathbf{Q}}$ compatible with the target quantum annealer. To satisfy hardware constraints, we sparsify the quadratic terms by pruning couplings (i.e., simplifying J) so that the reduced QUBO instance $\widetilde{\mathbf{Q}}$ is embeddable on the hardware connectivity graph within the available qubit budget. In this process, we minimize the distortion introduced by the reduction by prioritizing the most influential couplings for maintaining the database problem semantics and objective accuracy, ensuring that low-energy solutions of the reduced QUBO still decode to high-quality database configurations under the original formulation. These objectives can be formally defined as:

$$\min_{\widetilde{\mathbf{Q}}} \quad \mathcal{D}\left(\mathbf{Q}, \widetilde{\mathbf{Q}}\right) \qquad \text{(Distortion Minimization)} \tag{5}$$

$$\text{s.t.} \quad \widetilde{\mathbf{Q}} \text{ is embeddable on } G_{\text{hw}} \qquad \text{(Hardware Feasibility)} \tag{6}$$

where $\mathcal{D}(\cdot,\cdot)$ is a high-level measure of the distortion introduced by the reduction, and G_{hw} denotes the hardware connectivity graph.

Phase 2: Online Relaxation Correction. Given a reduced and embeddable QUBO $\widetilde{\mathbf{Q}}$ produced by Phase 1, we iteratively refine the QUBO using feedback from quantum annealing runs. The key idea is to compensate for the approximation errors introduced by coupling pruning in Phase 1 by adjusting the coefficients (equivalently, the Ising parameters (h, J)) with respect to the original QUBO $\mathbf{Q}$. Let $(h^{(t)}, J^{(t)})$ denote the model parameters at iteration t, and let $\mathcal{X}^{(t)}$ be the set of samples returned by the annealer. We update the parameters using discrepancy signals computed from $\mathbf{Q}$ and $\mathcal{X}^{(t)}$:

$$(h^{(t+1)}, J^{(t+1)}) = (h^{(t)}, J^{(t)}) + \eta\,\Delta\left(h^{(t)}, J^{(t)};\, \mathbf{Q},\, \mathcal{X}^{(t)}\right) \tag{7}$$

where $\eta > 0$ is a correction step size and $\Delta(\cdot)$ produces parameter adjustments computed based on $\mathcal{X}^{(t)}$ and guided by $\mathbf{Q}$. The objective of online relaxation correction is to minimize the expected query execution latency induced by the decoded database configuration from the annealer:

$$\min \quad \mathbb{E}_{\mathbf{s}\sim p_T(\mathbf{s})}\left[\text{Latency}\left(q, \pi(\mathbf{s})\right)\right] \qquad \text{(Latency Minimization)} \tag{8}$$

where $p_T(\mathbf{s})$ denotes the sampling distribution induced by quantum annealing under the final parameters $(h^{(T)}, J^{(T)})$, and $\pi(\mathbf{s})$ maps a sampled solution $\mathbf{s}$ to a database configuration for query q. This online relaxation correction phase continuously improves feasibility and solution quality through iterative refinement.

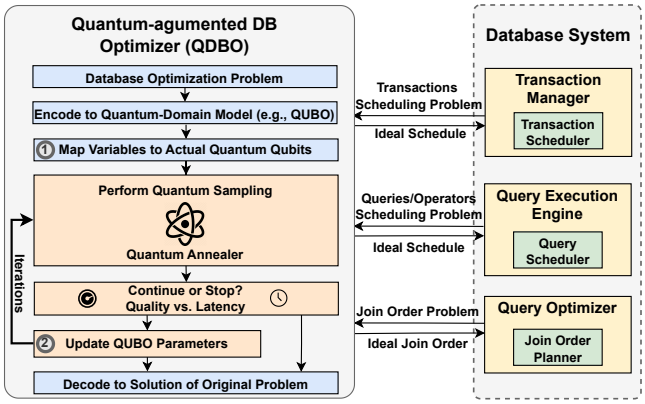

Figure 1: *QDBO* System Architecture.

3.2 System Architecture

Figure 1 illustrates the overall architecture of our proposed quantum-augmented database optimizer, referred to as *QDBO*, which tightly integrates with a conventional database system in a *real-time* setup. The left part of Figure 1 demonstrates the workflow of *QDBO*. The system first receives a large-scale database optimization problem, such as join order optimization, which is then encoded into a quantum-domain model (e.g., a QUBO). To solve such problems under the limited connectivity constraints of current quantum hardware in a transparent and controllable manner, we depart from black-box optimization approaches that directly invoke commercial quantum solvers to search for a single energy-minimizing configuration. Instead, *QDBO* adopts a sampling-based workflow that iteratively generates high-quality (low-energy) solutions for the original problem without connectivity restrictions. To enable execution on quantum hardware, *QDBO* constructs a reduced QUBO that preserves all decision variables while pruning a subset of correlations, and embeds this reduced formulation onto physical qubits

(Step ① in Figure 1; details are in Section 4). The resulting QUBO is then executed on the quantum annealer via quantum sampling. Based on the observed solution quality and user-defined latency constraints, the system may perform additional iterations to further improve the solution quality, thereby explicitly balancing optimization latency against solution quality. When additional iterations are executed, the sampled configurations and their energies are used to update the QUBO parameters (Step ② in Figure 1; details are in Section 5), and the process repeats. Otherwise, the current solution is decoded and returned as a solution to the original optimization problem. The iterative loop terminates when either a user-specified iteration budget or time budget is reached, allowing *QDBO* to operate under tight real-time latency constraints.

4 QUBO REDUCTION AND EMBEDDING

QUBOs derived from database optimization problems typically encode complex cost functions alongside constraints that enforce semantic correctness (e.g., ensuring a valid join-tree structure in query optimization). However, due to the limited physical connectivity of current quantum hardware, such QUBOs are often not natively embeddable. To approximately solve these problems on a quantum annealer, it is therefore necessary to reduce the QUBO while retaining all decision variables and preserving as much of the original connectivity as possible. This consists of two steps. Step 1 (Mapping) efficiently finds a high-quality mapping from logical QUBO variables to physical qubits. This step balances mapping quality and mapping latency: higher-quality mappings better preserve the original problem semantics, while low latency is crucial because large instances can involve thousands of variables and qubits. Step 2 (Embedding) relabels variables into qubit IDs using the mapping from Step 1 and filters quadratic couplings (i.e., simplifies J) to retain only the couplers supported by the device.

In this section, we first present three variable-qubit mapping strategies used in *QDBO*: *random* mapping, *greedy* mapping, and *semantic* mapping.[1] These strategies reflect different quality-efficiency trade-offs by incorporating quantum hardware geometry and database problem-specific semantics. Figure 2 illustrates how these strategies map logical variables ($\{v_1, \ldots, v_7\}$) to physical qubits ($\{q_1, \ldots, q_7\}$) on a quantum annealer. The blue nodes denote the original QUBO variables (e.g., V, PRED, and SLACK variables of the join order problem, introduced at the end of Section 2). For ease of illustration, the running example partitions the seven nodes into $\mathcal{V}_V = \{v_1, \ldots, v_4\}$ (validity) and $\mathcal{V}_{PRED} = \{v_5, v_6, v_7\}$ (predicate activation), abstracting away the SLACK group; edges among v_1–v_4 are intra-group validity couplings, and edges between the two groups represent cross-group couplings tying predicate activation to the join tree. The orange nodes represent the qubits selected in the first step of each mapping (i.e., red arrow), while the green nodes indicate the qubits selected in the second step. Edges between blue nodes indicate logical couplings, and edges between qubits represent physical couplers supported by the quantum annealer.

Random Mapping. The most straightforward approach to reduce an overly dense QUBO onto quantum hardware is random mapping, which assigns logical variables to physical qubits without considering either the geometry of the target hardware or the structure of the QUBO. For example, in Figure 2a, random mapping may first assign v_3 to q_5 and then assign v_4 to q_3, without enforcing any prioritization or structural constraints. This random mapping may end up with originally connected variables mapped to distant qubits, discarding their cross-group couplings. Here, we assume that the number of qubits is no smaller than the number of variables, ensuring an injective mapping in which each logical variable is assigned to a unique qubit. A random seed ρ is used to shuffle the variable list and assign variables to available qubits, yielding a mapping M. In general, random mapping is computationally inexpensive and easy to implement, making it attractive as a baseline or a fast preprocessing step. However, it often discards a large fraction of important correlations, leading to significant distortion of the original QUBO and degraded solution quality.

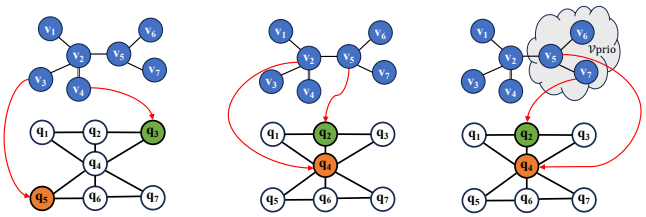

Figure 2: Visualization of Proposed Variable-Qubit Mappings.

Greedy Mapping. To better preserve information from the original QUBO while respecting hardware constraints, we introduce a greedy mapping strategy that explicitly accounts for both the logical structure of the QUBO and the physical connectivity of the quantum annealer. This method, referred to as *seeded-neighbor greedy mapping*, first identifies and maps a seed variable and a seed qubit based on structural properties, and then incrementally constructs the mapping by *greedily maximizing the strength of logical couplings that can be realized on physical couplers with already-mapped variables during the subsequent QUBO embedding procedure.* Figure 2b illustrates an example in which the greedy mapping first selects v_2 as the seed variable due to its large aggregate coupling strength (indicated by the double-line edges), and maps it to q_4, the most connected qubit in the hardware graph. Subsequent variables such as $v_5 \in \mathcal{V}_{PRED}$ are then mapped to neighboring qubits (e.g., q_2) to preserve the strong cross-group couplings.

Algorithm 1 illustrates the main routine of our greedy approach for constructing an injective mapping $M : \mathcal{V} \to \mathcal{N}$ from QUBO logical variables to physical qubits. First, we build the hardware neighborhood structure Neighbors from the physical qubits $\mathcal{N}$ and couplers $\mathcal{E}$ (line 1). Here, Neighbors$[q]$ denotes the set of qubits that are directly coupled to qubit q on the annealer. We then compute two logical statistics from the QUBO couplings $\mathcal{J}$: (i) AbsJ, which stores the absolute coupling magnitude $|J_{u,v}|$ for pairs of logical variables, and (ii) AbsJStrength, which aggregates the total absolute coupling strength for each variable. Concretely, AbsJStrength$[v] = \sum_u |J_{u,v}|$ measures how strongly variable v interacts with the remaining variables in the QUBO. Next, we select the seed variable v_0 as the

[1]We map each variable to a single qubit, instead of mapping a variable to multiple qubits (i.e., qubit chains), which is commonly done in black-box solvers. Chains may break easily and reduce scalability. For example, D-Wave Advantage [86] can handle only hundreds of variables even though it contains around 5000 quantum qubits.

Algorithm 1 Seeded-Neighbor Greedy Mapping

Input: QUBO logical variables $\mathcal{V}$, logical couplings $\mathcal{J}$ (with weights $J_{u,v}$), physical qubits $\mathcal{N}$, physical couplers $\mathcal{E}$
Output: Injective mapping $M : \mathcal{V} \to \mathcal{N}$

1: Neighbors $\leftarrow$ BuildHardwareNeighbor($\mathcal{N}, \mathcal{E}$)
▷ Neighbors$[q]$: *directly coupled quantum qubits of* q
2: (AbsJ, AbsJStrength) $\leftarrow$ ComputeAbsCouplingInfo($\mathcal{V}, \mathcal{J}$)
▷ AbsJ$[u][v] = |J_{u,v}|$, AbsJStrength$[v] = \sum_u |J_{u,v}|$ *for variables*
3: $v_0 \leftarrow \arg\max_{v \in \mathcal{V}}$ AbsJStrength$[v]$ ▷ *Seed variable*
4: $q_0 \leftarrow \arg\max_{q \in \mathcal{N}} |\text{Neighbors}[q]|$ ▷ *Seed qubit*
5: $M \leftarrow \{v_0 \mapsto q_0\}$, $q_{\text{curr}} \leftarrow q_0$ ▷ *Start greedy mapping*
6: $V_{\text{left}} \leftarrow \mathcal{V} \setminus \{v_0\}$, $Q_{\text{used}} \leftarrow \{q_0\}$
7: **while** $V_{\text{left}} \neq \emptyset$ **do**
8: $\mathcal{P} \leftarrow$ GetCandidateQubits(Q_{used}, Neighbors, ...)
▷ *Unused neighbors of already-mapped qubits*
9: $(v^\star, q^\star) \leftarrow$ SelectBestPlacement($q_{\text{curr}}, V_{\text{left}}, \mathcal{P}, \ldots$)
10: $M[v^\star] \leftarrow q^\star$
11: $V_{\text{left}} \leftarrow V_{\text{left}} \setminus \{v^\star\}$, $Q_{\text{used}} \leftarrow Q_{\text{used}} \cup \{q^\star\}$, $q_{\text{curr}} \leftarrow q^\star$
12: **Return** M

variable with the largest total interaction strength (line 3). Similarly, we select the seed qubit q_0 as the most connected qubit in the hardware graph (line 4). Placing the most "influential" logical variable on a highest-degree qubit enables the subsequent greedy expansion to preserve as many strong couplings as possible (lines 5–6). We initialize two auxiliary sets: $V_{\text{left}} = \mathcal{V} \setminus \{v_0\}$ stores the remaining unassigned logical variables, and $Q_{\text{used}} = \{q_0\}$ tracks the already-occupied qubits to guarantee the correctness of injectivity (i.e., no two variables map to the same qubit).

The main greedy loop continues until all logical variables are mapped. In each iteration, we first compute the candidate qubit pool $\mathcal{P}$, which contains the unused neighbors of already-mapped qubits, falling back to any unused qubit if the neighborhood is exhausted (line 8). This design encourages spatially coherent placements on the quantum annealer. We then select the next placement $(v^\star, q^\star)$ by invoking SelectBestPlacement$(\cdot)$, which continues the seeded-neighbor greedy logic (line 9). For each q in the current candidate qubit pool $\mathcal{P}$, it selects an unmapped variable $v \in V_{\text{left}}$ that maximizes the preserved coupling strength to *already-mapped* variables. Ties are broken by preferring the qubit closest to the most recently assigned qubit q_{curr} to encourage locality. The placement (v, q) with the largest score is returned. If no eligible placement is found, the procedure falls back to an arbitrary $(v, q) \in V_{\text{left}} \times \mathcal{P}$. Next, we commit the chosen placement by setting $M[v^\star] \leftarrow q^\star$ and update the corresponding loop state (lines 10–11). The algorithm repeats this loop until V_{left} becomes empty. Finally, it returns the completed injective greedy mapping M (line 12).

Compared to random mapping, greedy mapping incurs higher computational overhead but substantially improves embedding quality by preserving high-weight couplings.

Problem-specific Semantic Mapping. While greedy mapping is problem-agnostic, many database optimization QUBOs exhibit rich semantic structure that can be explicitly leveraged during reduction. Therefore, we also propose a semantic mapping that extends the greedy approach by incorporating domain knowledge about the roles of different variables in the QUBO formulation. For example, in the join order optimization problem, the logical

Algorithm 2 Problem-Specific Semantic Mapping

Input: logical QUBO $G_L = (\mathcal{V}, \mathcal{J})$, hardware graph $G_{\text{hw}} = (\mathcal{N}, \mathcal{E})$, priority set $\mathcal{V}_{\text{prio}} \subseteq \mathcal{V}$ (e.g., $\mathcal{V}_{\text{prio}} \in \{\mathcal{V}_V, \mathcal{V}_{PRED}, \mathcal{V}_{SLACK}\}$)
Output: injective mapping $M : \mathcal{V} \to \mathcal{N}$

1: Sort $q \in \mathcal{N}$ by hardware degree $d(q) = |\{q' \mid (q, q') \in \mathcal{E}\}|$ (desc); let $\mathcal{N}_{\text{top}}$ be the top $|\mathcal{V}_{\text{prio}}|$ qubits.
2: Greedily map $\mathcal{V}_{\text{prio}}$ onto $\mathcal{N}_{\text{top}}$ to prioritize preserving couplers.
3: Map the remaining variables $\mathcal{V} \setminus \mathcal{V}_{\text{prio}}$ onto $\mathcal{N} \setminus M(\mathcal{V}_{\text{prio}})$.
4: **return** M

variables can be categorized into semantic groups corresponding to validity constraints, predicate activation, and cost modeling.

Algorithm 2 shows our procedure for problem-specific semantic mapping. We select one group of variables (e.g., join order validity variables) and prioritize mapping them to the most highly connected qubits. This prioritization preserves correlations that are critical for semantic correctness. The remaining variables are then mapped using the remaining qubits. Figure 2c shows an example in which variables from the prioritized group (gray region, corresponding to $\mathcal{V}_{\text{prio}} = \mathcal{V}_{PRED} = \{v_5, v_6, v_7\}$ in the running example, where the priority group can be any semantic subset of $\mathcal{V}$) are mapped first, with v_5 placed on q_4 and v_7 on q_2, thereby preserving dense intra-group interactions as well as cross-group couplings that tie predicate activation to join tree validity.

Semantic mapping offers the highest potential solution quality among the three strategies by explicitly protecting semantically critical correlations while also considering hardware geometry. Moreover, semantic prioritization leaves more room to trade off mapping quality and efficiency: for selecting the next placement $(v^\star, q^\star)$, one can adopt simpler and faster heuristics, such as restricting the search space to neighbors of already-mapped qubits rather than evaluating thousands of qubits globally, or using the number of preserved couplings as a proxy instead of aggregating coupling strengths. However, it requires problem-specific knowledge and careful variable grouping, which increases design complexity.

Generate Embeddable QUBO. After obtaining the variable-to-qubit mapping M, the next key step is to generate a *quantum-hardware-embeddable* QUBO that can be executed on the quantum annealer. The key idea is to (i) relabel each logical variable using its mapped physical qubit ID (i.e., making the embedded QUBO directly runnable on the quantum annealer) and then (ii) remove any logical couplings that cannot be realized by physical couplers, and scale coupling strengths to fall within the hardware-supported range. The resulting embeddable QUBO is reduced and exhibits a sparser interaction structure than the original formulation, which naturally accelerates quantum sampling and improves scalability for large, complex database optimization problems.

5 ITERATIVE CORRELATION RELAXATION

While the QUBO reduction and embedding procedure enforces hardware feasibility and prioritizes semantic structure, it may still discard couplings that contain information critical to solution quality. This simplification introduces a gap between the reduced QUBO and the original QUBO with full database semantics.

To address this challenge, we propose an iterative correlation relaxation framework that progressively corrects the reduced QUBO

using feedback from quantum annealing runs. Rather than relying on a one-shot, black-box quantum solver that produces an opaque solution with limited user control, our approach treats the quantum annealer as an interactive optimization component. This design enables online refinement of the QUBO, thereby improving solution quality and providing explicit latency control, which are essential for deployment in `real-time` database systems. Our goal is not to recover every lost coupling, but to use the original QUBO objective to improve the final solution under hardware constraints.

In this section, we first describe our core idea of using the quantum annealer as a *programmable sampler* to generate sample solutions. We then show how these sampled solutions and their energy distributions are used to adjust the QUBO parameters, biasing future samples toward lower-energy configurations that better optimize the original database objective. Finally, we present our end-to-end approach, which iteratively corrects the QUBO relaxation during refinement and produces high-quality final solutions.

Core Idea: Quantum Annealer as a Programmable Sampler. We formulate a general database optimization problem as an objective function $G(x)$ over binary strings x, where each string contains a set of decision variables (e.g., Equation 2 instantiates $G(x)$ for join order optimization). Lower values of $G(x)$ indicate better solutions. We explicitly distinguish this defined objective $G(x)$ from the `sampled` Ising energy E_{Ising}, which arises from executing the QUBO on the quantum annealer. We rely on the "probability collectives" [7] approach, which seeks the minima by sampling from a Gibbs distribution [37]:

$$P(x) = \frac{\exp\big(-\beta \cdot G(x)\big)}{Z} \tag{9}$$

where β is the inverse temperature[2] and Z is the partition function, defined as $Z = \sum_{x'} \exp\big(-\beta \cdot G(x')\big)$, where the sum runs over all binary strings x'. This distribution strongly biases sampling toward configurations with lower objective values due to the exponential factor. For example, consider two configurations x_a and x_b which represent two solutions for a database optimization problem with $G(x_a) = 0$ and $G(x_b) = 5$. With $\beta = 1$, their probability ratio is $P(x_a)/P(x_b) = \exp\big(-\beta \cdot (G(x_a) - G(x_b))\big) = \exp(5) \approx 148$, meaning that the better solution x_a is about 148× more likely to be sampled than x_b. Consequently, if the quantum annealer can generate samples following the distribution $P(x)$, then high-quality solutions to the original database optimization problem will be observed with high probability.

Owing to the sampling nature of quantum annealing, each programmed problem is independently executed multiple times (e.g., 100 runs), producing a distribution of configurations that is peaked at the lowest Ising energies (i.e., E_{Ising}). From this perspective, we utilize the quantum annealer [18] as a programmable sampler [45], whose behavior can be controlled by adjusting the Ising parameters h and J, to generate candidate solutions (i.e., sampled configurations and their energies). The output can thus be viewed as a distribution over binary strings. Then, our goal is to iteratively adjust h and J such that the induced output distribution $Q(x; h, J)$ increasingly aligns with the target distribution $P(x)$ defined by the database objective $G(x)$. This alignment enables principled, feedback-driven optimization that is both transparent and adaptable (key properties for real-time database workloads).

QUBO Relaxation Correction. To reduce the distance between distributions $Q(x; h, J)$ and $P(x)$, we use gradient descent to update (h, J) based on quantum annealer samples and their energies.

Distribution closeness measurement. We use KL divergence [44] to quantify the closeness of these distributions, i.e., how well $Q(x; h, J)$ matches the target distribution $P(x)$:

$$D_{\text{KL}}(Q \,\|\, P) = \sum_x Q(x; h, J)\, \log\left(\frac{Q(x; h, J)}{P(x)}\right) \tag{10}$$

The goal is then to optimize the parameters (h, J) (i.e., correct the QUBO relaxation) to minimize $D_{\text{KL}}(Q\|P)$, thereby steering the quantum annealer toward generating low-$G(x)$ configurations (i.e., better solutions to the original problem) with higher probability.

Parameter update using gradient descent. The core challenge in optimizing Equation 10 is that the functional form of $Q(x; h, J)$ is not directly available. Instead, we draw samples from this distribution, i.e., perform parameterized sampling on the quantum annealer as a practical alternative. We utilize the Gibbs distribution of the Ising energy to express $Q(x; h, J)$.

$$Q(x; h, J) = \frac{\exp\big(-\beta\, E_{\text{Ising}}(x; h, J)\big)}{\sum_{x'} \exp\big(-\beta\, E_{\text{Ising}}(x'; h, J)\big)} \tag{11}$$

where $E_{\text{Ising}}(x; h, J)$ is the Ising energy obtained while sampling with the programmed biases $h = \{h_i\}$ and couplings $J = \{J_{ij}\}$ on the quantum annealer. To minimize $D_{\text{KL}}(Q\|P)$, we perform gradient descent as in machine learning approaches [74] for updating the parameters h and J. To simplify the mathematical expressions in the gradients, we introduce the spin representation $s_i = 2x_i - 1 \in \{-1, +1\}$; then the gradients of $D_{\text{KL}}(Q\|P)$ with respect to the couplings and biases can be expressed as covariance-style terms under Q:

$$\nabla_{J_{ij}} D_{\text{KL}}(Q\|P) = -\beta\Big(\big\langle s_i s_j \cdot \Lambda(x)\big\rangle_Q - \big\langle s_i s_j\big\rangle_Q \big\langle \Lambda(x)\big\rangle_Q\Big) \tag{12}$$

$$\nabla_{h_i} D_{\text{KL}}(Q\|P) = -\beta\Big(\big\langle s_i \cdot \Lambda(x)\big\rangle_Q - \big\langle s_i\big\rangle_Q \big\langle \Lambda(x)\big\rangle_Q\Big) \tag{13}$$

where $\Lambda(x)$ denotes $\log(Q(x; h, J)/\exp(-\beta\, G(x)))$, and $\langle\cdot\rangle_Q$ denotes expectation with respect to $Q(x; h, J)$, i.e., the Gibbs distribution associated with the Ising model executed on the quantum annealer. In practice, these expectations over all binary strings can be approximated using the finite samples returned from the quantum annealer. As a result, we can compute $\Lambda(x)$ across each sampled configuration using $G(x)$ (i.e., the original objective) and $E_{\text{Ising}}(x; h, J)$.

To be more specific, in iteration t, we draw K samples $\{x^{(k)}\}_{k=1}^{K} \sim Q(\cdot; h^{(t)}, J^{(t)})$ from the quantum annealer, compute $G(x^{(k)})$ for each sample, and approximate expectations in (12)–(13) with sample averages (i.e., $\langle f(x)\rangle_Q \approx \frac{1}{K}\sum_{k=1}^{K} f(x^{(k)})$). We then update the QUBO parameters using a gradient-descent step:

$$J_{ij}^{(t+1)} = J_{ij}^{(t)} - \eta\, \widehat{\nabla}_{J_{ij}} D_{\text{KL}}(Q\|P) \tag{14}$$

$$h_i^{(t+1)} = h_i^{(t)} - \eta\, \widehat{\nabla}_{h_i} D_{\text{KL}}(Q\|P) \tag{15}$$

where η is the step size and $\widehat{\nabla}$ denotes the sample-based gradient estimate. This procedure repeats until a termination criterion is

[2] Intuitively, β controls how sharply the distribution concentrates on low-$G(x)$ configurations: as $\beta \to 0$, $P(x)$ approaches a nearly uniform distribution over all binary strings, whereas as $\beta \to \infty$, $P(x)$ concentrates on the global minimizers of $G(x)$.

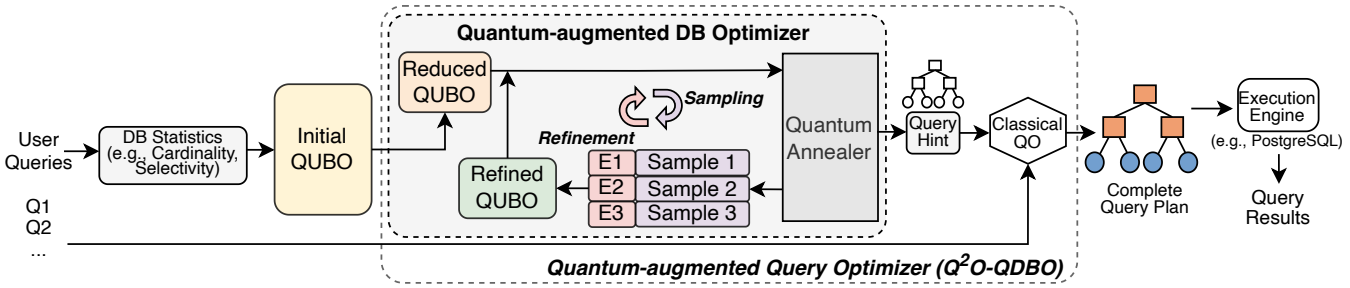

Figure 3: Framework of Quantum-augmented Query Optimizer (*Q²O-QDBO*).

met (e.g., a maximum number of iterations). Its lightweight, sample-driven nature enables efficient online execution.

Iterative Sampling and Correction. Combining parameterized sampling on the quantum annealer with gradient-descent-based parameter updates as described before, we define a *closed-loop* iterative sampling and correction process. The procedure runs for T iterations, where T denotes the total number of sampling rounds allowed under a user-specified budget (e.g., latency constraints or desired solution quality). When $T = 1$, the procedure reduces to a single sampling round without correction. Specifically, $t = 0$ uses the initial reduced-and-embedded parameters $(h^{(0)}, J^{(0)})$. For each round $t \geq 1$, we draw a batch of samples $\mathcal{X}^{(t)}$ from the quantum annealer programmed with $(h^{(t)}, J^{(t)})$, and then use these samples and their energies to obtain $(h^{(t+1)}, J^{(t+1)})$. Algorithm 3 outlines the overall procedure. Upon completion of T iterations, the algorithm outputs a final configuration $x^\star$, chosen from the quantum annealer samples, along with its objective value $G(x^\star)$. Sampling fidelity is controlled by the per-iteration sample size K.

6 CASE STUDIES

To demonstrate *QDBO*'s generality, we present two use cases of *QDBO* on representative database optimization problems: join ordering with *Q²O-QDBO* and index selection with *IS-QDBO*.

Algorithm 3 Closed-Loop Iterative Sampling and Correction

Input: Objective $G(x)$; inverse temperature β; learning rate η; samples per iteration K; max iterations T based on time constraint
Output: Best solution $x^\star$ across all rounds and its objective $G(x^\star)$

Initialize programmable parameters $(h^{(0)}, J^{(0)})$; $x^\star \leftarrow \perp$; $G^\star \leftarrow +\infty$
for iteration $t = 0, 1, \ldots, T-1$ **do**
  $\mathcal{X}^{(t)} \leftarrow \textsc{QSampling}(h^{(t)}, J^{(t)}, K)$ ▷ *Draw K samples*
  $(x^\star, G^\star) \leftarrow \textsc{GetBest}(x^\star, G^\star, \mathcal{X}^{(t)}, G)$
  if $T > 1$ **then** ▷ *No correction when $T = 1$*
    $\widehat{Q}^{(t)} \leftarrow \textsc{EmpiricalDistribution}\big(\{x^{(k)}, E_{\text{Ising}}(x^{(k)})\}_{k=1}^{K}\big)$
    $D_{\text{KL}}^{(t)} \leftarrow \textsc{KLDivergence}(\widehat{Q}^{(t)}, G(x), \beta)$
    $(\widehat{\nabla}_h^{(t)}, \widehat{\nabla}_J^{(t)}) \leftarrow \textsc{EstimateGradient}\big(\mathcal{X}^{(t)}, h^{(t)}, J^{(t)}, G, \beta\big)$
    $h^{(t+1)} \leftarrow h^{(t)} - \eta \cdot \widehat{\nabla}_h^{(t)}$
    $J^{(t+1)} \leftarrow J^{(t)} - \eta \cdot \widehat{\nabla}_J^{(t)}$
Return $x^\star$, $G(x^\star)$

6.1 Quantum-augmented Query Optimization

To demonstrate the practicality and system-level feasibility of *QDBO*, we first present *Q²O-QDBO*, a Quantum-augmented Query Optimizer, as a representative case study. *Q²O-QDBO* performs join order optimization and illustrates how *QDBO* can be integrated into a production database engine. It is a re-implementation of our earlier real-time, black-box-hybrid-solver-based optimizer, *Q²O* [52], redesigned to leverage the transparent and controllable optimization pipeline provided by *QDBO*. *Q²O-QDBO* tightly couples the QUBO reduction techniques (Section 4) and iterative correlation relaxation framework (Section 5) with a classical query optimizer, PostgreSQL [69]. Figure 3 shows the architecture of our system. Upon receiving a user query, *Q²O-QDBO* first retrieves cardinality and selectivity statistics from the DBMS engine. These statistics are used to instantiate a QUBO following the formulation in [80], parameterized by (h, J) in the equivalent Ising formulation, that encodes both the optimization objective (e.g., estimated execution cost) and semantic constraints (e.g., validity of join-tree structures).

Once constructed, the resulting `Initial` QUBO is typically too dense to be executed directly on current quantum annealers [86]. *QDBO* therefore applies the QUBO reduction and embedding methods described in Section 4 to generate a hardware-compatible formulation. For problem-specific semantic mapping, we separately prioritize three variable groups (i.e., V, PRED, and SLACK, introduced at the end of Section 2), enabling fine-grained control over the tradeoff between feasibility and semantic preservation.

Following reduction, *QDBO* invokes the iterative correlation relaxation process. In each iteration, the system samples from the current QUBO using the quantum annealer and refines the QUBO parameters based on the observed sample distribution and energies (Section 5). Each sampling-refinement cycle constitutes one iteration (Iteration 1 samples the initial reduced QUBO; subsequent iterations sample the refined QUBO). The number of iterations is governed by an explicit time budget, specified either by the user or automatically determined by the system, allowing *QDBO* to operate under real-time latency constraints. After completing the prescribed number of iterations, *QDBO* returns the best join order sequence discovered within the time budget. This sequence is translated into a query plan hint.[3] PostgreSQL then selects appropriate

[3]PostgreSQL [69] uses *pg_hint_plan* [67] to influence plan selection by incorporating "hints" in SQL comments. The hint can be expressed as a clause beginning with "`Leading`" to specify the join order of relations. For example, the hint `Leading (a (b c))` indicates that $b \bowtie c$ is performed first, followed by $a \bowtie (b \bowtie c)$.

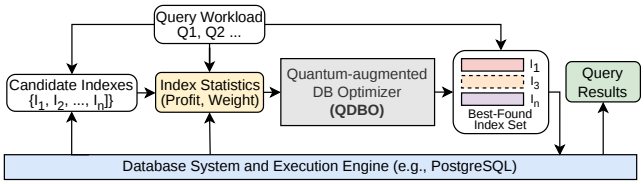

Figure 4: Framework of *QDBO*-based Index Selection.

join operators (e.g., hash join or nested-loop join) and produces a complete execution plan, which is executed by the DBMS engine.

6.2 Quantum-augmented Index Selection

As a second case study, we apply *QDBO* to the index selection problem. Figure 4 shows the end-to-end *IS-QDBO* pipeline. Given a query workload and a storage budget, *IS-QDBO* first extracts a candidate index set from the queries and computes a per-index profit (the workload-level execution-time saving) and storage weight by probing the DBMS engine. These statistics are used to instantiate the budget-constrained QUBO in Equation 4.

Following the same workflow as in Q^2O-*QDBO*, the resulting QUBO is reduced and embedded onto the quantum annealer using the methods of Section 4. For semantic mapping, we prioritize either the index decision variables or the slack variables, providing fine-grained control over which couplings are protected during reduction. The iterative correlation relaxation procedure (Section 5) then refines the QUBO across iterations. Finally, *IS-QDBO* returns an index combination that is materialized in PostgreSQL.

7 EVALUATION

We evaluate Q^2O-*QDBO* on real workloads and synthetic benchmarks to answer the following key questions: (1) How effective is Q^2O-*QDBO* for join order optimization, as measured by query execution latency? What is the end-to-end latency of Q^2O-*QDBO*? (Section 7.2) (2) How does Q^2O-*QDBO* handle extremely large join order optimization problems? How does Q^2O-*QDBO* compare with black-box quantum solvers in terms of efficiency and solution quality? (Section 7.3) (3) Does *QDBO* generalize to other database optimization problems beyond join ordering? Specifically, how does *IS-QDBO* perform on the index selection problem? (Section 7.4)

7.1 Experimental Setup

We run all experiments on a server with an Intel i9-9980XE CPU and 128 GB of RAM. We use Advantage2_system1.7 as the quantum sampler on the D-Wave Leap quantum service platform [20] and PostgreSQL 16.4 with the corresponding *pg_hint_plan*. For query execution, we use Balsa's PostgreSQL setup [94] and a hot-cache, three-run protocol, discarding the first warm-up run and averaging the following two. For the index selection use case, we additionally set the *maximum parallel worker* to 1 to ensure a fair comparison.

Baselines. As black-box baseline approaches, we adopt D-Wave's BQM-Solver [19], CQM-Solver [19], and NL-Solver [66], which are representative quantum solvers. In this paper, similar to *QDBO*, we integrate these solvers into our quantum-augmented query optimizer to construct Q^2O-*BQM*, Q^2O-*CQM* [52], and Q^2O-*NL* [52]. These solvers take formulated database optimization problems, such as join order optimization, as input and return corresponding solutions, which we translate back into join orders for evaluation. We additionally compare against two leading learned query optimizers, Balsa [94] and LEON+ [14], as representative reinforcement-learning-based and machine-learning-based optimizers. We train them using their official implementations. For each real workload, both learned optimizers require multi-hour offline training on a subset of queries and are then evaluated on the remainder. For JOB, we take 33 queries as the training set and the remaining 80 queries as the test set; for CEB, 32 and 48 queries, respectively. For the index selection use case, we similarly integrate D-Wave's BQM-Solver into our pipeline to construct *IS-BQM*, analogous to Q^2O-*BQM* in the join order use case. We additionally compare *IS-QDBO* against a classical dynamic-programming (DP) algorithm that solves the budget-constrained selection as a 0-1 knapsack problem.

Workloads. To comprehensively evaluate the effectiveness and scalability of our system, we use both real workloads and synthetic workloads. For real workloads, we use the *Join Order Benchmark (JOB)* [46] and the *Cardinality Estimation Benchmark (CEB)* [63]. Both JOB (113 queries) and CEB (80 queries) are built on the IMDB movie database schema and data [46]. Compared with JOB, CEB contains more complex query patterns and poses a broader range of cardinality estimation challenges. This makes both benchmarks suitable and widely adopted for evaluating join order optimization. For synthetic workloads, we use the same queries as in [65, 80], which include 270 queries with up to 50 relations for the classical chain, star, and cycle query graphs [80], and 400 queries with up to 50 relations for tree queries [65]. For the index selection use case, we use the standard TPC-H benchmark at scale factor 1. Following the prior methodology [43], we remove all primary keys and existing indexes so that the relative effect of each candidate index is measured against an empty-index baseline. We evaluate 19 TPC-H queries, excluding Q2, Q17, and Q20 as in [43] because their subquery costs are orders of magnitude higher than the rest and would otherwise dominate the workload. We construct B-tree candidate indexes covering single-column, two-column, and three-column combinations from `WHERE`, `JOIN`, and `ORDER BY` columns, plus the original primary-key constraints, yielding 92 candidates in total. The workload-level profit p_i is the execution time saved relative to the empty-index baseline, and the weight w_i is the actual storage size of the materialized index. The total weight is $\sum w_i = 7{,}634$ MB. We evaluate four storage budgets $B \in \{10\%, 20\%, 40\%, 50\%\}$ of $\sum w_i$ and set the per-query timeout to 60 s.

Evaluation Metrics. We focus on the following metrics. (1) *Query Execution Time (Exec) and Execution Speedup.* To evaluate join-order quality on real queries, we measure the actual query execution time under PostgreSQL. For each query, we compare the execution time of the default optimizer plan ($T^{\text{Exec}}_{\text{base}}$) with that of the plan enforced by the join-order hint derived from a quantum-based solution (either *QDBO* or a black-box solver), denoted $T^{\text{Exec}}_{\text{hint}}$. We report the execution speedup as $T^{\text{Exec}}_{\text{base}}/T^{\text{Exec}}_{\text{hint}}$. (2) *End-to-End Latency (E2E) and Speedup.* End-to-end latency measures the complete lifecycle of solving a database optimization problem, including optimization overhead (e.g., QUBO construction, reduction, sampling, and refinement) as well as query execution time. Let $T^{\text{E2E}}_{\text{base}}$ denote the

Table 1: Query Execution and End-to-End Latency Evaluation on JOB and CEB Benchmarks, where Iter 1 and Iter 3 Refer to Q^2O-QDBO under the 1-Iteration and 3-Iteration Settings, Respectively.

Type	Workload	Metric	LQOs		Q^2O with Black-box Solvers			Q^2O-QDBO (ours)					
								Iter 1			Iter 3		
			Balsa	LEON$^+$	BQM	CQM	NL	Random	Greedy	Semantic	Random	Greedy	Semantic
Exec	JOB	Improved queries	20	35	13	4	**18**	23	18	**38**	39	30	**62**
		Best speedup	5.37×	2.06×	13.77×	3.17×	**13.91×**	16.51×	16.30×	**16.95×**	16.63×	**22.53×**	19.14×
	CEB	Improved queries	9	2	12	8	**13**	14	**19**	13	23	**26**	25
		Best speedup	30.00×	1.05×	8.21×	15.89×	**25.29×**	25.22×	31.72×	**43.52×**	45.93×	27.19×	**49.36×**
E2E	JOB	Improved queries	0	1	0	0	**3**	9	4	**12**	**7**	2	**7**
		Best speedup	-	1.04×	-	-	**1.27×**	**5.95×**	5.60×	5.91×	2.77×	2.68×	**3.16×**
		Average speedup		1.04×			**1.20×**	1.96×	2.29×	**2.37×**	1.77×	1.88×	**2.06×**
	CEB	Improved queries	2	0	4	4	**6**	**8**	5	7	11	6	**12**
		Best speedup	6.08×	-	2.77×	6.28×	**6.48×**	6.18×	5.19×	**36.88×**	32.92×	7.98×	**34.65×**
		Average speedup	4.30×		1.83×	**2.84×**	2.32×	2.64×	2.30×	**7.60×**	5.19×	3.37×	**5.39×**

total latency using the default optimizer and $T^{\text{E2E}}_{\text{hint}}$ denote the total latency when using a quantum-based solution. We report the end-to-end speedup as $T^{\text{E2E}}_{\text{base}}/T^{\text{E2E}}_{\text{hint}}$. (3) *DB Cost:* Synthetic workloads only contain cardinalities and selectivities, and thus the queries cannot be executed to obtain query execution latency. Instead, we evaluate join order quality using the classical cost function [82]:

$$C(\mathbf{s}) \;:=\; \sum_{i=2}^{n} C_{\text{out}}(|s_1 \cdots s_{i-1}|,\, |s_i|)\,, \tag{16}$$

where $\mathbf{s} = (s_1, \ldots, s_n)$ denotes a join order, $|s_1 \cdots s_{i-1}|$ is the result size after joining $s_1, \ldots, s_{i-1}$, and $C_{\text{out}}(\cdot,\cdot)$ is the per-step cost computed from the given cardinalities and selectivities.

Default Setups. We perform evaluation under an end-to-end real-time execution scenario. The number of iterations in *QDBO* refers to the number of quantum sampling rounds performed. Specifically, in an n-iteration configuration (denoted as *Iter n*), the first iteration consists of: (i) generating the optimization problem (i.e., constructing the original QUBO) from problem-specific inputs (e.g., query statistics for join ordering, or workload-level index profits and storage weights for index selection); (ii) submitting the QUBO to the quantum annealer and retrieving the sampled results; and (iii) performing local QUBO refinement when $n > 1$. Each of the remaining $n-1$ iterations repeats the refinement step followed by resubmission to obtain new sampling results. Unless otherwise specified, we use Iter 1 and Iter 3 as the default settings for *QDBO*. For the iterative correlation relaxation in *QDBO*, we set $\beta = 1.0$, the gradient-descent step size $\eta = 0.01$, and the per-iteration sample size `num_reads`=100, selected empirically to balance solution quality and the real-time latency budget. For BQM-Solver and CQM-Solver, we use their default configurations, where the solver automatically determines the processing time for each optimization instance. For NL-Solver, we set its *maximum allowed run time* to 1 s for real workloads and to 5 s for larger synthetic queries to ensure sufficient optimization time for solution quality while avoiding excessive latency.

7.2 Query Execution and End-to-End Latency

In this section, we focus on query execution latency (Exec) and end-to-end latency (E2E) for both the JOB and CEB workloads.

Execution Latency. As expected, the plan quality (i.e., only the running time of the selected plan) of most long-running queries is improved by Q^2O-QDBO. The upper part of Table 1 (Exec) shows that Q^2O-QDBO consistently outperforms black-box solvers in the following aspects. *(1) Coverage:* A primary strength of Q^2O-QDBO is its broad improvement coverage: it improves substantially more queries than black-box solvers. For example, on JOB under execution-time evaluation (JOB-Exec), Iter 3 with semantic mapping improves 62 queries, compared with 13 (BQM), 4 (CQM), and 18 (NL). A similar trend holds on CEB, where 25 queries are improved in Iter 3, exceeding 12 (BQM), 8 (CQM), and 13 (NL). *(2) Iterative refinement:* Our closed-loop QUBO refinement yields systematic gains as correction rounds are applied. From Iter 1 to Iter 3, the number of improved queries increases markedly (e.g., JOB-Exec under semantic mapping: 38 → 62), and the best-case speedups also increase (e.g., JOB-Exec under greedy mapping: 16.30× → 22.53×; CEB-Exec under semantic mapping: 43.52× → 49.36×). These results indicate that the sampling-correction loop effectively pushes Q^2O-QDBO toward higher-quality database solutions. *(3) Semantic mapping advantage:* Among our three mapping strategies, semantic mapping delivers the strongest overall performance. It achieves the highest improvement counts and the largest speedups in most settings (e.g., Iter 3 semantic mapping reaches an execution speedup of 49.36× on CEB). This supports our design choice to explicitly prioritize preserving semantically connected variables.

End-to-End Latency (E2E). The lower part of Table 1 reports the end-to-end latency comparison. On CEB, Q^2O-QDBO achieves an average end-to-end speedup of 7.60× in Iter 1 and 5.39× in Iter 3, substantially exceeding 1.83× (BQM), 2.84× (CQM), and 2.32× (NL). The trend is similar on JOB; however, Q^2O-BQM and Q^2O-CQM do not achieve end-to-end latency improvements due to their relatively longer overhead. In general, the slight drop in average speedup from Iter 1 to Iter 3 is due to additional iterations introducing more quantum sampling calls. This also affects the number of improved queries on the JOB workload. Surprisingly, we improve more queries in Iter 3 than in Iter 1 on the more complex CEB workload. This highlights the effect of correlation correction, which yields better query plans to reduce query execution latency in Iter 3 and compensates for the additional overhead introduced by the iterative procedure. Across the three mapping strategies in Q^2O-QDBO, semantic mapping achieves the highest average speedup. Notably, this higher average is achieved on a comparable or larger fraction of queries, indicating that the gains are both strong

Table 2: End-to-End Latency Breakdown and Comparison Between Q^2O-QDBO and PostgreSQL (PG).

	Workload Time (ms)	JOB		CEB	
Solver	Query / #Joins	26c / 17		q54 / 11	
Baseline (PG)	Planning	93		89	
	Execution	5604		196,176	
	End-to-End	5697		196,265	
Q^2O-QDBO (ours)	Iteration	1	3	1	3
	Planning with hints	66	67	67	69
	Execution	4623	**689**	4508	**3974**
	QDBO overhead	852	1971	747	1621
	End-to-End	5541	2727	5322	5664
Our Gain (Over PG)	**Absolute Time Saving**	156	2970	190,943	190,601
	Exec Time ↑	1.21×	**8.13×**	43.52×	**49.36×**
	E2E Latency ↑	1.03×	**2.09×**	**36.88×**	34.65×

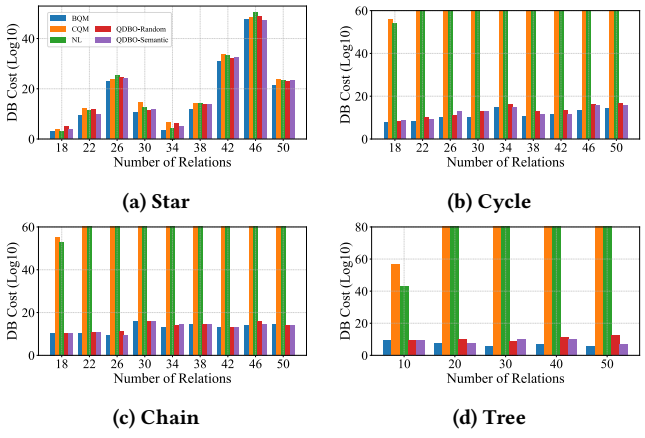

Figure 5: DB Cost for Synthetic Queries.

and broadly applicable. Overall, these results confirm that Q^2O-QDBO delivers both solution-quality improvements and system-level efficiency gains.

Table 2 shows the end-to-end latency breakdown for two selected JOB and CEB queries. The high quality of Q^2O-QDBO's generated join order sequences significantly reduces execution time and can also slightly reduce planning time in the classical query optimizer (i.e., PostgreSQL produces a complete plan based on the given join order as a hint). Even after accounting for quantum solver overhead (e.g., 747 ms in Iter 1 and 1621 ms in Iter 3), we still achieve multi-second absolute time savings (i.e., up to 190,943 ms). The reported overhead is averaged over multiple runs to better reflect the cloud-service latency of the D-Wave quantum platform. This verifies the effectiveness and efficiency of Q^2O-QDBO for `real-time` database optimization on real workloads.

Comparison with Learned Query Optimizers. Table 1 compares *QDBO* against two leading learned query optimizers, Balsa [94] and LEON+ [14], on the JOB and CEB workloads. In terms of query execution time compared with PostgreSQL, Balsa improves 20 JOB queries and 9 CEB queries, while LEON+ improves 35 JOB queries and 2 CEB queries. In contrast, *QDBO*-Iter 3 with semantic mapping improves 62 and 25 queries on JOB and CEB, respectively, with higher best speedups of 19.14× and 49.36×. Beyond the broader execution time speedup and the elimination of multi-hour offline model training, Q^2O-QDBO also achieves better results in the end-to-end evaluation, where learned query optimizers suffer from model-inference and system overhead. This indicates *QDBO*'s practical competitiveness in real-time database optimization.

7.3 Scalability Analysis on Synthetic Queries

To evaluate scalability, we compare Q^2O-QDBO with other black-box-solver-based baselines on synthetic queries with up to 50 relations under four query graph schemas: *chain*, *cycle*, *star*, and *tree*. Among these baselines, we consider Q^2O-BQM our main competitor as BQM is a QUBO-native hybrid solver that employs multiple workers to solve large-scale optimization instances in parallel. For Q^2O-QDBO, we adopt a 2-iteration configuration (Iter 2). This configuration captures both solution quality and system overhead, reflecting the practical trade-off between refinement and latency. We characterize scalability from two perspectives: *Quality scalability* (whether the *DB Cost* metric degrades catastrophically as query size grows) and *Efficiency scalability* (whether the quantum-based optimization overhead remains within a reasonable range).

DB Cost Analysis. Figure 5 reports solution quality across schemas and query sizes. Overall, Q^2O-BQM delivers strong and stable performance. Our Q^2O-QDBO (both random and semantic mappings) consistently maintains low *DB Cost* without catastrophic degradation as the number of relations increases, demonstrating strong quality scalability. In most cases, Q^2O-QDBO achieves performance comparable to Q^2O-BQM, and outperforms it on several large instances (e.g., Relation 50 in chain and Relation 46 in star). In contrast, Q^2O-NL and Q^2O-CQM exhibit severe quality degradation on chain, cycle, and tree schemas, where global join-tree constraints induce strong variable dependencies. Their *DB Cost* increases rapidly as problem size grows. They perform relatively better on the star schema, where structural dependencies are weaker. Within Q^2O-QDBO, semantic mapping consistently yields more stable and often lower *DB Cost* than random mapping, particularly on large and structurally complex instances (e.g., Relation 46 in chain, Relation 50 in cycle and star, and the tree schema). By prioritizing semantically critical variables and couplings during reduction, semantic mapping better preserves key correlations, improving robustness at scale.

Quantum Optimization Overhead Analysis. Figure 6 reports the end-to-end quantum solver time as the overhead. Q^2O-CQM shows a clear upward trend in latency as problem size increases (orange bars). Q^2O-NL maintains relatively stable overhead due to its enforced runtime cap (5s in our setting). Our Q^2O-QDBO achieves consistently lower end-to-end overhead than Q^2O-BQM, even though Iter 2 requires two quantum sampling rounds, incurring an additional cloud round-trip on the order of seconds. This indicates that Q^2O-QDBO offsets the extra invocation cost: QUBO reduction decreases the difficulty of embedding and sampling, and the correction step improves solution quality without introducing substantial additional overhead. This advantage becomes more pronounced for large and dense QUBOs, where embedding and sampling dominate runtime. More broadly, the white-box control

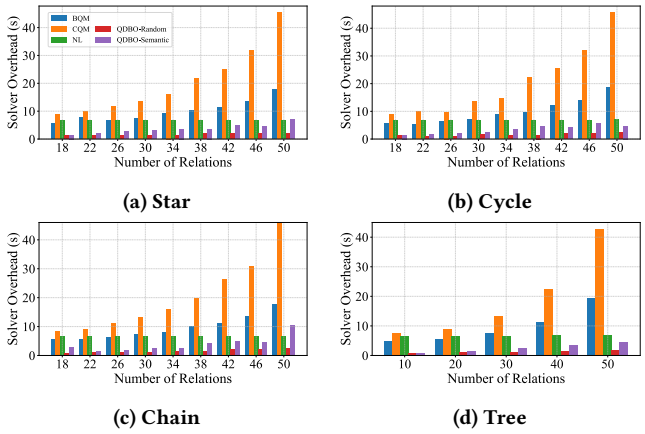

Figure 6: Overhead for Synthetic Queries.

in *QDBO* over reduction, mapping, and refinement allows it to avoid unnecessary solver effort when sufficient solution quality has already been achieved. Within Q^2O-*QDBO*, semantic mapping incurs slightly higher overhead than random mapping, reflecting the natural quality-overhead trade-off: better semantic preservation comes at a modest increase in computational cost.

7.4 Index Selection Evaluation

To validate the generality of *QDBO* beyond join ordering, we evaluate *IS-QDBO* on the index selection problem. For *IS-QDBO*, we evaluate the three variable-qubit mapping strategies (random, greedy, and semantic) introduced in Section 4 and report both `iter=1` and `iter=3` to assess the effect of iterative correlation relaxation.

Table 3 reports the workload speedup over the empty-index baseline. *IS-QDBO* matches or exceeds both DP and *IS-BQM* in most budget settings. The best result, **2.639×**, is achieved by *IS-QDBO* with semantic mapping at $B = 50\%$. The differences across solvers are sometimes small because a few dominant indexes capture most of the available speedup; once these are selected, the marginal benefit of additional indexes is limited.

An interesting case occurs at $B = 40\%$, where *IS-QDBO* with greedy mapping at `iter=1` drops to 1.087× because it misses one of the `lineitem(l_orderkey)`-prefix indexes that is critical for Q21. By `iter=3`, the iterative correlation relaxation reselects the missing index, reducing Q21's execution time and lifting the result to 2.531×. This illustrates how iterative correction can repair the sub-optimal solution generated from the first-iteration reduced embeddings. The semantic mapping consistently delivers stable and competitive results because it preserves the QUBO structure most faithfully when the candidate set contains many borderline items.

Table 3: Workload Speedup over the Baseline on TPC-H.

Budget	Classical	Black-box Solver	*IS-QDBO*					
	DP	*IS-BQM*	Random		Greedy		Semantic	
			`iter=1`	`iter=3`	`iter=1`	`iter=3`	`iter=1`	`iter=3`
$B = 10\%$	2.422×	2.425×	2.431×	2.432×	**2.637×**	**2.637×**	2.453×	2.453×
$B = 20\%$	2.447×	2.410×	2.365×	2.447×	2.455×	2.450×	2.423×	**2.539×**
$B = 40\%$	2.531×	**2.636×**	2.543×	2.543×	1.087×	2.531×	2.529×	2.533×
$B = 50\%$	2.530×	2.628×	2.522×	2.522×	2.541×	2.541×	**2.639×**	**2.639×**

8 RELATED WORK

Quantum Solvers. Quantum optimization is broadly split into quantum annealing and gate-based models. Quantum annealing drives the system from a superposition toward a low-energy state corresponding to a near-optimal solution [11, 88, 96]. Limited native qubit connectivity has motivated hybrid quantum-classical solvers [19, 25] that decompose the problem, embed subproblems onto the hardware, sample, and compose results. Examples include D-Wave Advantage [86], the CQM solver [19], and NL-Solver [66]. While scalable, these solvers incur high service-level overhead and expose little low-level control, hindering integration with latency-sensitive engines. Gate-based models [34, 97] instead use variational algorithms such as VQE [87] and QAOA [27, 32, 99]. We focus on quantum annealing, which is currently more mature for real-system integration, and leave gate-based approaches to future work.

Quantum Computing for Database Optimization. Early work used Grover's algorithm and its variants for database search [10, 24, 29, 30, 38, 68, 85]. Recent efforts target broader tasks: join ordering [50, 61, 62, 78–80, 91, 92], multiple query optimization [26, 81, 88], index selection and tuning [31, 42, 89], and transaction scheduling [8, 28, 60]. Recent work also examines the readiness of quantum hardware for real-time database optimization [48] and fusion-based execution of large-scale QUBOs [49]. Existing systems [47, 51, 52, 78] treat commercial hybrid solvers as black boxes, leaving QUBO reduction, variable mapping, and low-level refinement outside user control. *QDBO* is the first end-to-end sampling-centric pipeline that integrates with database systems in *real-time* settings, applies hardware-aware and problem-specific QUBO reduction and mapping, and closes the loop with lightweight online refinement. We target annealers because current devices [18, 86] host enough qubits for our problem sizes within a real-time budget, while gate-based circuits remain limited in qubit count and coherence. The design of *QDBO* is not tied to a specific quantum annealer and can be carried over to future architectures.

9 CONCLUSION

We presented *QDBO*, a white-box quantum-augmented database optimizer that brings quantum annealing into real-time large-scale database optimization. It addresses key obstacles in current deployments of black-box quantum solvers. Using join order optimization and index selection as two case studies, extensive experiments with Q^2O-*QDBO* and *IS-QDBO* show that *QDBO* delivers high-quality solutions with low end-to-end latency. Looking ahead, we expect the deployment cost of quantum annealers to decrease substantially, following the trajectory of other emerging technologies such as GPUs. Beyond these case studies, *QDBO* is designed as a general-purpose framework that opens the door to quantum-augmented solutions for a broad class of database optimization problems.

ACKNOWLEDGMENTS

This work is supported by the National Science Foundation, USA, under grant IIS-2544715.

REFERENCES

[1] Sanjay Agrawal, Surajit Chaudhuri, and Vivek R. Narasayya. 2000. Automated Selection of Materialized Views and Indexes in SQL Databases. In *Proceedings of the 26th International Conference on Very Large Data Bases (VLDB '00)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 496–505.

[2] Tameem Albash and Daniel A. Lidar. 2018. Demonstration of a Scaling Advantage for a Quantum Annealer over Simulated Annealing. *Phys. Rev. X* 8 (Jul 2018), 031016. Issue 3. https://doi.org/10.1103/PhysRevX.8.031016

[3] Sanjeev Arora and Boaz Barak. 2009. *Computational Complexity: A Modern Approach* (1st ed.). Cambridge University Press, USA.

[4] Abhishek Awasthi, Nico Kraus, Florian Krellner, and David Zambrano. 2024. Real World Application of Quantum-Classical Optimization for Production Scheduling. In *2024 IEEE International Conference on Quantum Computing and Engineering (QCE)*, Vol. 02. 239–244. https://doi.org/10.1109/QCE60285.2024.10285

[5] Ethan Bernstein and Umesh Vazirani. 1997. Quantum Complexity Theory. *SIAM J. Comput.* 26, 5 (1997), 1411–1473. https://doi.org/10.1137/S0097539796300921

[6] Zhengbing Bian, Fabian Chudak, Robert Israel, Brad Lackey, William G. Macready, and Aidan Roy. 2014. Discrete optimization using quantum annealing on sparse Ising models. *Frontiers in Physics* 2 (Sept. 2014). https://doi.org/10.3389/fphy.2014.00056

[7] Stefan Bieniawski, Ilan Kroo, and David Wolpert. [n.d.]. *Discrete, Continuous, and Constrained Optimization Using Collectives.* https://doi.org/10.2514/6.2004-4580 arXiv:https://arc.aiaa.org/doi/pdf/10.2514/6.2004-4580

[8] Tim Bittner and Sven Groppe. 2020. Avoiding blocking by scheduling transactions using quantum annealing. In *Proceedings of the 24th Symposium on International Database Engineering & Applications* (Seoul, Republic of Korea) *(IDEAS '20)*. Association for Computing Machinery, New York, NY, USA, Article 21, 10 pages. https://doi.org/10.1145/3410566.3410593

[9] Kelly Boothby, Andrew D. King, and Jack Raymond. 2021. Zephyr Topology of D-Wave Quantum Processors. In *D-Wave Technical Report*. 1–18.

[10] Michel Boyer, Gilles Brassard, Peter H?yer, and Alain Tapp. 1996. *Tight bounds on quantum searching*. Technical Report.

[11] Umut Çalikyilmaz, Sven Groppe, Jinghua Groppe, Tobias Winker, Stefan Prestel, Farida Shagieva, Daanish Arya, Florian Preis, and Le Gruenwald. 2023. Opportunities for Quantum Acceleration of Databases: Optimization of Queries and Transaction Schedules. *Proc. VLDB Endow.* 16, 9 (May 2023), 2344–2353. https://doi.org/10.14778/3598581.3598603

[12] Surajit Chaudhuri and Vivek R. Narasayya. 1997. An Efficient, Cost-Driven Index Selection Tool for Microsoft SQL Server. In *Proceedings of the 23rd International Conference on Very Large Data Bases (VLDB '97)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 146–155.

[13] Shuhan Chen, Congqi Shen, and Chunming Wu. 2025. Intelligent Transaction Scheduling to Enhance Concurrency in High-Contention Workloads. *Applied Sciences* 15, 11, Article 6341 (2025). https://doi.org/10.3390/app15116341

[14] Xu Chen, Ximu Zeng, Yuze Wang, Zibo Liang, Kai Zeng, Han Su, and Kai Zheng. 2026. LEON+: Towards Robust ML-Aided Query Optimization. *The VLDB Journal* 35, 3 (2026), 21.

[15] Audrey Cheng, Aaron Kabcenell, Jason Chan, Xiao Shi, Peter Bailis, Natacha Crooks, and Ion Stoica. 2024. Towards Optimal Transaction Scheduling. *Proc. VLDB Endow.* 17, 11 (July 2024), 2694–2707. https://doi.org/10.14778/3681954.3681956

[16] Sophie Cluet and Guido Moerkotte. 1995. On the Complexity of Generating Optimal Left-Deep Processing Trees with Cross Products. In *Database Theory - ICDT '95, 5th International Conference, Prague, Czech Republic, January 11-13, 1995, Proceedings (Lecture Notes in Computer Science)*, Georg Gottlob and Moshe Y. Vardi (Eds.), Vol. 893. Springer, 54–67. https://doi.org/10.1007/3-540-58907-4_6

[17] Giuseppe Colucci, Stan van der Linde, and Frank Phillipson. 2023. Power Network Optimization: A Quantum Approach. *IEEE Access* 11 (2023), 98926–98938. https://doi.org/10.1109/ACCESS.2023.3312997

[18] D-Wave Quantum Inc. 2025. Performance gains in the D-Wave Advantage2 system at the 4,400-qubit scale. https://www.dwavequantum.com/media/wakjcpsf/adv2_4400q_whitepaper-1.pdf. D-Wave Whitepaper Series, No. 14-1083A-A.

[19] D-Wave Systems. 2021. *Hybrid Solver for Constrained Quadratic Models.* White Paper 14-1055A-A. D-Wave Systems Inc. https://www.dwavesys.com/media/rldh2ghw/14-1055a-a_hybrid_solver_for_constrained_quadratic_models.pdf.

[20] D-Wave Systems Inc. 2024. D-Wave Leap's Hybrid Solvers. https://docs.dwavequantum.com/en/latest/industrial_optimization/index.html. Online documentation.

[21] Vasil S. Denchev, Sergio Boixo, Sergei V. Isakov, Nan Ding, Ryan Babbush, Vadim Smelyanskiy, John Martinis, and Hartmut Neven. 2016. What is the Computational Value of Finite-Range Tunneling? *Phys. Rev. X* 6 (Aug 2016), 031015. Issue 3. https://doi.org/10.1103/PhysRevX.6.031015

[22] dimod [n.d.]. dimod Documentation. https://docs.ocean.dwavesys.com/en/stable/docsdimod/.

[23] D-Wave documentation. 2024. https://docs.dwavesys.com/docs/latest/c_gs_4.html.

[24] Christoph Durr and Peter Hoyer. 1996. A Quantum algorithm for finding the minimum. (7 1996). arXiv:quant-ph/9607014

[25] Suguru Endo, Zhenyu Cai, Simon C. Benjamin, and Xiao Yuan. 2021. Hybrid Quantum-Classical Algorithms and Quantum Error Mitigation. *Journal of the Physical Society of Japan* 90, 3 (2021), 032001. https://doi.org/10.7566/JPSJ.90.032001

[26] Tobias Fankhauser, Marc E. Solèr, Rudolf M. Füchslin, and Kurt Stockinger. 2023. Multiple Query Optimization Using a Gate-Based Quantum Computer. *IEEE Access* 11 (2023), 114031–114043. https://doi.org/10.1109/ACCESS.2023.3324253

[27] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. 2014. A Quantum Approximate Optimization Algorithm. arXiv:1411.4028 [quant-ph]

[28] Sven Groppe and Jinghua Groppe. 2021. Optimizing Transaction Schedules on Universal Quantum Computers via Code Generation for Grover's Search Algorithm. In *Proceedings of the 25th International Database Engineering & Applications Symposium (IDEAS)* (Montreal, QC, Canada) *(IDEAS '21)*. Association for Computing Machinery, New York, NY, USA, 149–156. https://doi.org/10.1145/3472163.3472164

[29] Lov K. Grover. 1996. A fast quantum mechanical algorithm for database search. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing* (Philadelphia, Pennsylvania, USA) *(STOC '96)*. Association for Computing Machinery, New York, NY, USA, 212–219. https://doi.org/10.1145/237814.237866

[30] Lov K. Grover. 1997. Quantum Computers Can Search Arbitrarily Large Databases by a Single Query. *Phys. Rev. Lett.* 79 (Dec 1997), 4709–4712. Issue 23. https://doi.org/10.1103/PhysRevLett.79.4709

[31] Le Gruenwald, Tobias Winker, Umut Çalikyilmaz, Jinghua Groppe, and Sven Groppe. 2023. Index Tuning with Machine Learning on Quantum Computers for Large-Scale Database Applications. In *Joint Proceedings of Workshops at the 49th International Conference on Very Large Data Bases (VLDB 2023), Vancouver, Canada, August 28 - September 1, 2023 (CEUR Workshop Proceedings)*, Vol. 3462. CEUR-WS.org. https://ceur-ws.org/Vol-3462/QDSM5.pdf

[32] Stuart Hadfield, Zhihui Wang, Bryan O'Gorman, Eleanor G. Rieffel, Davide Venturelli, and Rupak Biswas. 2019. From the Quantum Approximate Optimization Algorithm to a Quantum Alternating Operator Ansatz. *Algorithms* 12, 2 (2019). https://doi.org/10.3390/a12020034

[33] Toshihide Ibaraki and Tiko Kameda. 1984. On the optimal nesting order for computing N-relational joins. *ACM Trans. Database Syst.* 9, 3 (Sept. 1984), 482–502. https://doi.org/10.1145/1270.1498

[34] IBM Quantum. 2022. Cloud access to quantum computers provided by IBM. https://quantum.ibm.com. Accessed: 2022.

[35] Yannis E. Ioannidis. 1996. Query optimization. *ACM Comput. Surv.* 28, 1 (March 1996), 121–123. https://doi.org/10.1145/234313.234367

[36] Ernst Ising. 1925. Beitrag zur Theorie des Ferromagnetismus. *Zeitschrift fur Physik* 31, 1 (Feb. 1925), 253–258. https://doi.org/10.1007/BF02980577

[37] E. T. Jaynes. 1957. Information Theory and Statistical Mechanics. *Physical Review* 106 (1957), 620–630. https://doi.org/10.1103/PhysRev.106.620

[38] Yi-Lin Ju, I-Ming Tsai, and Sy-Yen Kuo. 2007. Quantum Circuit Design and Analysis for Database Search Applications. *IEEE Transactions on Circuits and Systems I: Regular Papers* 54, 11 (2007), 2552–2563. https://doi.org/10.1109/TCSI.2007.907845

[39] Tadashi Kadowaki and Hidetoshi Nishimori. 1998. Quantum annealing in the transverse Ising model. *Phys. Rev. E* 58, 5 (Nov 1998), 5355–5363. https://doi.org/10.1103/PhysRevE.58.5355

[40] Hamed Karimi and Gili Rosenberg. 2017. Boosting quantum annealer performance via sample persistence. *Quantum Information Processing* 16, 7 (July 2017), 1–24. https://doi.org/10.1007/s11128-017-1615-x

[41] Tarun Kathuria and S. Sudarshan. 2017. Efficient and Provable Multi-Query Optimization. In *Proceedings of the 36th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems* (Chicago, Illinois, USA) *(PODS '17)*. Association for Computing Machinery, New York, NY, USA, 53–67. https://doi.org/10.1145/3034786.3034792

[42] Manish Kesarwani and Jayant R. Haritsa. 2024. Index Advisors on Quantum Platforms. *Proc. VLDB Endow.* 17, 11 (July 2024), 3615–3628. https://doi.org/10.14778/3681954.3682025

[43] Jan Kossmann, Stefan Halfpap, Marcel Jankrift, and Rainer Schlosser. 2020. Magic Mirror in My Hand, Which Is the Best in the Land? An Experimental Evaluation of Index Selection Algorithms. *Proceedings of the VLDB Endowment* 13, 11 (2020), 2382–2395. https://doi.org/10.14778/3407790.3407832

[44] Solomon Kullback and Richard A Leibler. 1951. On information and sufficiency. *The annals of mathematical statistics* 22, 1 (1951), 79–86.

[45] Brad Lackey. 2018. A belief propagation algorithm based on domain decomposition. arXiv:1810.10005 [cs.DS] https://arxiv.org/abs/1810.10005

[46] Viktor Leis, Andrey Gubichev, Atanas Mirchev, Peter Boncz, Alfons Kemper, and Thomas Neumann. 2015. How Good Are Query Optimizers, Really? *Proc. VLDB Endow.* 9, 3 (nov 2015), 204–215. https://doi.org/10.14778/2850583.2850594

[47] Hanwen Liu, Abhishek Kumar, Federico Spedalieri, and Ibrahim Sabek. 2025. Hybrid Quantum-Classical Optimization for Bushy Join Trees. In *Proceedings of the 2nd Workshop on Quantum Computing and Quantum-Inspired Technology for Data-Intensive Systems and Applications (Q-Data '25)*. Association for Computing

Machinery, New York, NY, USA, 20–24. https://doi.org/10.1145/3736393.3736695
[48] Hanwen Liu and Ibrahim Sabek. 2026. Is Quantum Computing Ready for Real-Time Database Optimization?. In *Proceedings of the 42nd IEEE International Conference on Data Engineering (ICDE), Lightning Talks*. arXiv:2601.12123.
[49] Hanwen Liu and Ibrahim Sabek. 2026. QFusion: A Demonstration of Boundary-Aware Fusion Planning and Execution for Large-Scale QUBO Optimization. In *Proceedings of the VLDB Endowment (Demonstrations)*.
[50] Hanwen Liu and Ibrahim Sabek. 2026. Towards a Hybrid Quantum-Classical Computing Framework for Database Optimization Problems in Real Time Setup. In *Proceedings of the 42nd IEEE International Conference on Data Engineering (ICDE), Data Engineering Frontier Technologies Track*. arXiv:2602.14263.
[51] Hanwen Liu, Pranshi Saxena, Federico Spedalieri, and Ibrahim Sabek. 2025. Optimizing Join Orders via Constrained Quadratic Models (Abstract). In *Proc. 1st Workshop Quantum Data and Machine Learning* (HongKong, China) *(QDML '25)*.
[52] Hanwen Liu, Federico Spedalieri, and Ibrahim Sabek. 2025. A Demonstration of Q2O: Quantum-Augmented Query Optimizer. *Proc. VLDB Endow.* 18, 12 (2025), 5439–5443. https://doi.org/10.14778/3750601.3750691
[53] Hanwen Liu, Qihan Zhang, Ryan Marcus, and Ibrahim Sabek. 2025. SEFRQO: A Self-Evolving Fine-Tuned RAG-Based Query Optimizer. arXiv:2508.17556 [cs.DB] https://arxiv.org/abs/2508.17556
[54] Hanwen Liu, Qihan Zhang, Ryan Marcus, and Ibrahim Sabek. 2025. SERAG: Self-Evolving RAG System for Query Optimization. https://viterbi-web.usc.edu/~sabek/pdf/25_workshop_serag.pdf
[55] Andrew Lucas. 2014. Ising formulations of many NP problems. *Frontiers in Physics* 2 (2014). https://doi.org/10.3389/fphy.2014.00005
[56] Ryan Marcus, Parimarjan Negi, Hongzi Mao, Nesime Tatbul, Mohammad Alizadeh, and Tim Kraska. 2021. Bao: Making Learned Query Optimization Practical. In *SIGMOD*. 1275–1288. https://doi.org/10.1145/3448016.3452838
[57] Ryan Marcus, Parimarjan Negi, Hongzi Mao, Chi Zhang, Mohammad Alizadeh, Tim Kraska, Olga Papaemmanouil, and Nesime Tatbul. 2019. Neo: A Learned Query Optimizer. *Proc. VLDB Endow.* 12, 11 (2019), 1705–1718. https://doi.org/10.14778/3342263.3342644
[58] Ryan Marcus and Olga Papaemmanouil. 2018. Deep Reinforcement Learning for Join Order Enumeration. In *Proceedings of the First International Workshop on Exploiting Artificial Intelligence Techniques for Data Management* (Houston, TX, USA) *(aiDM'18)*. Association for Computing Machinery, New York, NY, USA, Article 3, 4 pages. https://doi.org/10.1145/3211954.3211957
[59] Guido Moerkotte and Thomas Neumann. 2008. Dynamic Programming Strikes Back. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*. 539–552. https://doi.org/10.1145/1376616.1376672
[60] Nitin Nayak, Alexandru Prisacaru, Umut Çalıkyılmaz, Jinghua Groppe, and Sven Groppe. 2025. Quantum-Enhanced Transaction Scheduling with Reduced Complexity via Solving QUBO Iteratively using a Locking Mechanism. In *Proceedings of the 2nd Workshop on Quantum Computing and Quantum-Inspired Technology for Data-Intensive Systems and Applications* (Berlin, Germany) *(Q-Data '25)*. Association for Computing Machinery, New York, NY, USA, 26–35. https://doi.org/10.1145/3736393.3736701
[61] Nitin Nayak, Jan Rehfeld, Tobias Winker, Benjamin Warnke, Umut Çalikyilmaz, and Sven Groppe. 2023. Constructing Optimal Bushy Join Trees by Solving QUBO Problems on Quantum Hardware and Simulators. In *Proceedings of the International Workshop on Big Data in Emergent Distributed Environments (BiDEDE '23)*. Association for Computing Machinery, New York, NY, USA, Article 7, 7 pages. https://doi.org/10.1145/3579142.3594298
[62] Nitin Nayak, Tobias Winker, Umut Çalıkyılmaz, Sven Groppe, and Jinghua Groppe. 2024. Quantum Join Ordering by Splitting the Search Space of QUBO Problems. *Datenbank-Spektrum* 24, 1 (2024), 21–32. https://doi.org/10.1007/s13222-024-00468-3
[63] Parimarjan Negi, Ryan Marcus, Andreas Kipf, Hongzi Mao, Nesime Tatbul, Tim Kraska, and Mohammad Alizadeh. 2021. Flow-Loss: Learning Cardinality Estimates That Matter. *Proc. VLDB Endow.* 14, 11 (jul 2021), 2019–2032. https://doi.org/10.14778/3476249.3476259
[64] Thomas Neumann and Bernhard Radke. 2018. Adaptive Optimization of Very Large Join Queries. In *Proceedings of the 2018 International Conference on Management of Data*. 677–692. https://doi.org/10.1145/3183713.3183733
[65] Thomas Neumann and Bernhard Radke. 2018. Adaptive Optimization of Very Large Join Queries. In *Proceedings of the 2018 International Conference on Management of Data* (Houston, TX, USA) *(SIGMOD '18)*. Association for Computing Machinery, New York, NY, USA, 677–692. https://doi.org/10.1145/3183713.3183733
[66] Eneko Osaba and Pablo Miranda-Rodriguez. 2025. D-Wave's Nonlinear-Program Hybrid Solver: Description and Performance Analysis. *IEEE Access* 13 (2025), 4724–4736. https://doi.org/10.1109/ACCESS.2025.3525620
[67] OSSC-DB. [n.d.]. pg_hint_plan. https://github.com/ossc-db/pg_hint_plan.
[68] Apoorva Patel. 2001. Quantum database search can do without sorting. *Phys. Rev. A* 64 (Aug 2001), 034303. Issue 3. https://doi.org/10.1103/PhysRevA.64.034303
[69] PostgreSQL. [n.d.]. https://www.postgresql.org/.
[70] PostgreSQL Global Development Group. 2026. *Genetic Query Optimization*. https://www.postgresql.org/docs/current/geqo-pg-intro.html Accessed: 2026-01-30.
[71] John Preskill. 2018. Quantum Computing in the NISQ era and beyond. *Quantum* 2 (Aug. 2018), 79. https://doi.org/10.22331/q-2018-08-06-79
[72] Q2ODemo [n.d.]. Video Recording of VLDB 2025 Demonstration: Q2O: Quantum-augmented Query Optimizer. https://drive.google.com/drive/folders/1XpCJnP_21kukHDOi7pmQ_R-dB7tKyXuX.
[73] Rigetti Computing. 2022. Rigetti quantum processors. https://qcs.rigetti.com/qpus.
[74] Herbert Robbins and Sutton Monro. 1951. A Stochastic Approximation Method. *The Annals of Mathematical Statistics* 22, 3 (1951), 400–407.
[75] Ibrahim Sabek, Tenzin Samten Ukyab, and Tim Kraska. 2022. LSched: A Workload-Aware Learned Query Scheduler for Analytical Database Systems. In *SIGMOD*. 1228–1242. https://doi.org/10.1145/3514221.3526158
[76] James Sanders. 2019. D-Wave Announces 5,000-Qubit Fifth Generation Quantum Annealer. TechRepublic. https://www.dwavequantum.com/company/newsroom/media-coverage/techrepublic-d-wave-announces-5-000-qubit-fifth-generation-quantum-annealer/.
[77] Giuseppe E. Santoro, Roman Martoňák, Erio Tosatti, and Roberto Car. 2002. Theory of Quantum Annealing of an Ising Spin Glass. *Science* 295, 5564 (2002), 2427–2430. https://doi.org/10.1126/science.1068774 arXiv:https://www.science.org/doi/pdf/10.1126/science.1068774
[78] Pranshi Saxena, Ibrahim Sabek, and Federico Spedalieri. 2024. Constrained Quadratic Model for Optimizing Join Orders. In *Proceedings of the 1st Workshop on Quantum Computing and Quantum-Inspired Technology for Data-Intensive Systems and Applications* (Santiago, Chile) *(Q-Data '24)*. ACM, New York, NY, USA, 38–44. https://doi.org/10.1145/3665225.3665447
[79] Manuel Schönberger, Stefanie Scherzinger, and Wolfgang Mauerer. 2023. Ready to Leap (by Co-Design)? Join Order Optimisation on Quantum Hardware. *Proc. ACM Manag. Data* 1, 1, Article 92 (may 2023), 27 pages. https://doi.org/10.1145/3588946
[80] Manuel Schönberger, Immanuel Trummer, and Wolfgang Mauerer. 2023. Quantum-Inspired Digital Annealing for Join Ordering. *Proc. VLDB Endow.* 17, 3 (2023), 511–524. https://doi.org/10.14778/3632093.3632112
[81] Manuel Schönberger, Immanuel Trummer, and Wolfgang Mauerer. 2025. Large-Scale Multiple Query Optimisation with Incremental Quantum(-Inspired) Annealing. *Proc. ACM Manag. Data* 3, 4 (2025), 253:1–253:25. https://doi.org/10.1145/3749171
[82] P. Griffiths Selinger, M. M. Astrahan, D. D. Chamberlin, R. A. Lorie, and T. G. Price. 1979. Access Path Selection in a Relational Database Management System. In *Proc. ACM SIGMOD Int. Conf. Manage. Data (SIGMOD '79)*. 23–34. https://doi.org/10.1145/582095.582099
[83] Timos K. Sellis. 1988. Multiple-query optimization. *ACM Trans. Database Syst.* 13, 1 (March 1988), 23–52. https://doi.org/10.1145/42201.42203
[84] Jie Tan, Kangfei Zhao, Rui Li, Jeffrey Xu Yu, Chengzhi Piao, Hong Cheng, Helen Meng, Deli Zhao, and Yu Rong. 2025. Can Large Language Models Be Query Optimizer for Relational Databases? *Proc. ACM Manag. Data* 3, 6, Article 306 (Dec. 2025), 28 pages. https://doi.org/10.1145/3769771
[85] Barbara M. Terhal and John A. Smolin. 1998. Single quantum querying of a database. *Phys. Rev. A* 58 (Sep 1998), 1822–1826. Issue 3. https://doi.org/10.1103/PhysRevA.58.1822
[86] The D-Wave Advantage System [WhitePaper]. 2022. https://www.dwavesys.com/media/3xvdipcn/14-1058a-a_advantage_processor_overview.pdf.
[87] Jules Tilly, Hongxiang Chen, Shuxiang Cao, Dario Picozzi, Kanav Setia, Ying Li, Edward Grant, Leonard Wossnig, Ivan Rungger, George H. Booth, and Jonathan Tennyson. 2022. The Variational Quantum Eigensolver: A review of methods and best practices. *Physics Reports* 986 (Nov. 2022), 1–128. https://doi.org/10.1016/j.physrep.2022.08.003
[88] Immanuel Trummer and Christoph Koch. 2016. Multiple Query Optimization on the D-Wave 2X Adiabatic Quantum Computer. *Proc. VLDB Endow.* 9, 9 (May 2016), 648–659. https://doi.org/10.14778/2947618.2947621
[89] Immanuel Trummer and Davide Venturelli. 2024. Leveraging Quantum Computing for Database Index Selection. In *Proceedings of the 1st Workshop on Quantum Computing and Quantum-Inspired Technology for Data-Intensive Systems and Applications* (Santiago, AA, Chile) *(Q-Data '24)*. Association for Computing Machinery, New York, NY, USA, 14–26. https://doi.org/10.1145/3665225.3665445
[90] Peter J. M. van Laarhoven and Emile H. L. Aarts. 1987. Simulated Annealing. In *Simulated Annealing: Theory and Applications*. Springer Netherlands, Dordrecht, 7–15. https://doi.org/10.1007/978-94-015-7744-1_2
[91] Tobias Winker, Umut Çalikyilmaz, Le Gruenwald, and Sven Groppe. 2023. Quantum Machine Learning for Join Order Optimization using Variational Quantum Circuits. In *Proceedings of the International Workshop on Big Data in Emergent Distributed Environments (BiDEDE '23) (DoiDEDE '23)*. Association for Computing Machinery, Article 5, 7 pages. https://doi.org/10.1145/3579142.3594299
[92] Tobias Winker, Umut Çalikyilmaz, Le Gruenwald, and Sven Groppe. 2023. Quantum Machine Learning for Join Order Optimization using Variational Quantum Circuits. In *Proceedings of the International Workshop on Big Data in Emergent Distributed Environments* (<conf-loc>, <city>Seattle</city>, <state>WA</state>,

<country>USA</country>, </conf-loc>) *(BiDEDE '23)*. Association for Computing Machinery, New York, NY, USA, Article 5, 7 pages. https://doi.org/10.1145/3579142.3594299

[93] Wentao Wu, Chi Wang, Tarique Siddiqui, Junxiong Wang, Vivek Narasayya, Surajit Chaudhuri, and Philip A. Bernstein. 2022. Budget-aware Index Tuning with Reinforcement Learning. In *Proceedings of the 2022 International Conference on Management of Data* (Philadelphia, PA, USA) *(SIGMOD '22)*. Association for Computing Machinery, New York, NY, USA, 1528–1541. https://doi.org/10.1145/3514221.3526128

[94] Zongheng Yang, Wei-Lin Chiang, Sifei Luan, Gautam Mittal, Michael Luo, and Ion Stoica. 2022. Balsa: Learning a Query Optimizer Without Expert Demonstrations. In *Proceedings of the 2022 International Conference on Management of Data*. 931–944. https://doi.org/10.1145/3514221.3517885

[95] Xiang Yu, Guoliang Li, Chengliang Chai, and Nan Tang. 2020. Reinforcement Learning with Tree-LSTM for Join Order Selection. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*. IEEE, 1297–1308. https://doi.org/10.1109/ICDE48307.2020.00116

[96] Gongsheng Yuan, Yuxing Chen, Jiaheng Lu, Sai Wu, Zhiwei Ye, Ling Qian, and Gang Chen. 2024. Quantum Computing for Databases: Overview and Challenges. *CoRR* abs/2405.12511 (2024). https://doi.org/10.48550/arXiv.2405.12511

[97] Ehsan Zahedinejad and Arman Zaribafiyan. 2017. Combinatorial Optimization on Gate Model Quantum Computers: A Survey. arXiv:1708.05294 [quant-ph] https://arxiv.org/abs/1708.05294

[98] Junyi Zhao, Kai Su, Yifei Yang, Xiangyao Yu, Paraschos Koutris, and Huanchen Zhang. 2025. Debunking the Myth of Join Ordering: Toward Robust SQL Analytics. *Proc. ACM Manag. Data* 3, 3, Article 146 (June 2025), 28 pages. https://doi.org/10.1145/3725283

[99] Leo Zhou, Sheng-Tao Wang, Soonwon Choi, Hannes Pichler, and Mikhail D. Lukin. 2020. Quantum Approximate Optimization Algorithm: Performance, Mechanism, and Implementation on Near-Term Devices. *Phys. Rev. X* 10 (Jun 2020), 021067. Issue 2. https://doi.org/10.1103/PhysRevX.10.021067