QFusion: A Demonstration of Boundary-Aware Fusion Planning and Execution for Large-Scale QUBO Optimization

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

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

ABSTRACT

Large-scale database optimization problems can be encoded as Quadratic Unconstrained Binary Optimization (QUBO) instances for execution on quantum hardware. As the problem size grows, these QUBOs quickly exceed the capacity of current devices and must be decomposed into multiple subproblems. However, decomposition does not eliminate global dependencies: subproblems remain coupled through boundary interfaces, and overall solution quality depends critically on how partial solutions are fused.

In this demonstration, we present *QFusion*, the first boundary-aware fusion planning and execution framework for large-scale QUBO solving that runs on an actual quantum annealer. We propose a DBMS-inspired, cost-based fusion tree that explicitly selects the fusion order and merge strategy. *QFusion* supports three database optimization tasks: join order optimization, multiple query optimization, and index selection. Conference attendees can interactively control the complete workflow through three designed scenarios: inspecting the construction of QUBOs with their database semantics, adjusting the decomposition granularity to observe the resulting boundary interfaces, and configuring fusion planning to examine the execution results on an actual quantum annealer. Overall, the demonstration highlights that scalable QUBO solving is not only a matter of solving local subproblems, but also a systems problem that requires boundary-aware fusion planning. The video corresponding to this demonstration is available at this link.

PVLDB Reference Format:
Hanwen Liu and Ibrahim Sabek. QFusion: A Demonstration of Boundary-Aware Fusion Planning and Execution for Large-Scale QUBO Optimization. PVLDB, 19(12): 4826 - 4829, 2026.
doi:10.14778/3827998.3828132

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/ihanwen99/vldb26-demo.

1 INTRODUCTION

Quantum computing has become a promising computational paradigm, driven by rapid advances in quantum hardware [3, 4]. Recently, quantum computing has been explored for database optimization tasks [6], including join order optimization [7–9, 11, 12, 15, 17, 18], multiple query optimization [19, 21], and index selection [5, 22]. Quantum annealing [2] is a representative quantum computing paradigm that outperforms classical approaches in certain settings, particularly for optimization problems, by enabling efficient exploration of exponentially large search spaces [1]. A common pattern is to encode a database optimization problem as a Quadratic Unconstrained Binary Optimization (QUBO) [13] and then solve it on a quantum annealer. However, QUBOs may require tens of thousands of binary variables to represent complex database semantics with large relations (e.g., a 50-table join order problem may generate over 10,000 binary variables), whereas current quantum annealers provide fewer physical qubits than required (e.g., D-Wave systems [3] offer on the order of 5,000 qubits).

To bridge the mismatch between problem size and hardware capability, prior work [16] suggests applying advanced solvers (e.g., mixed-integer linear programming) to the most challenging parts while relying on classical algorithms for the remainder. However, directly transferring this strategy to quantum annealing does not fully exploit quantum advantages and does not scale efficiently to larger QUBO instances. As noted in our prior work [8, 10], a practical and generic approach is to decompose an oversized QUBO into smaller subproblems, solve them separately, and strategically merge partial solutions. Yet QUBOs introduce a unique difficulty: they contain a large number of dense pairwise correlations between variables (i.e., couplings) that encode the semantics of the given database problem and the penalties for violating constraints. After decomposition, many couplings cross partition boundaries, so the final solution quality depends not only on local subproblem solutions but also on how cross-boundary couplings are handled during composition. Consequently, the composition stage requires careful fusion planning that accounts for boundary interfaces (i.e., boundary variables and cut couplings) between partitions and decides both the fusion order and the merge strategy.

In this demonstration, we present *QFusion*, the *first* boundary-aware Fusion planning and execution framework for large-scale QUBO solving that runs on an actual quantum annealer. *QFusion* focuses on how large-scale QUBOs are fused after decomposition and serves as a monitoring tool for studying the quality–efficiency trade-off. We first formalize the optimization target by accounting for both the contributions from partitioned subproblems and the boundary interfaces between them. We then formulate fusion planning by drawing inspiration from join order optimization in DBMSs and propose a cost-based strategy for constructing a fusion tree. Moreover, we present three merge strategies that pairwise compose partial solutions with different levels of boundary awareness. Finally, *QFusion* integrates with a D-Wave quantum annealer [3] to execute and evaluate fusion plans and return final solutions.

Proceedings of the VLDB Endowment, Vol. 19, No. 12 ISSN 2150-8097.
doi:10.14778/3827998.3828132

This demonstration, featuring three designed scenarios, allows conference attendees to interact with *QFusion* directly and control the entire lifecycle, from QUBO construction and decomposition to fusion planning and execution. It provides the first visualization that bridges QUBO construction with database semantics across three classical database optimization tasks: *join order optimization* [18], *multiple query optimization* [21], and *index selection* [22]. In Scenario 1, attendees can examine how QUBOs are constructed across different problem sizes and how their variables and couplings reflect the underlying database semantics. In Scenario 2, they can adjust the number of partitions to explore decomposition behavior and the resulting boundary interfaces, highlighting the practical challenges of recombining components under cross-boundary interactions. In Scenario 3, attendees can experiment with alternative fusion tree structures (e.g., linear or bushy) and merge strategies, and observe their impact on end-to-end solution quality and timing metrics during execution on the quantum annealer. This demonstration makes fusion planning and execution explicit, observable, tunable, and comparable, highlighting fusion planning as a first-class decision for solving large-scale QUBOs. A recorded video of the demonstration is available at this link.

2 PRELIMINARIES

We first present the foundations of quantum annealing, and then introduce the three QUBO formulations used in our demonstration.

2.1 Quantum Annealing

Quantum annealing (QA) relies on quantum fluctuations to explore regions of the configuration space that are hard to reach under classical evolution. A key feature is *quantum tunneling*, which lets the system pass through energy barriers rather than overcome them thermally. Quadratic Unconstrained Binary Optimization (QUBO) [13] is a standard formulation for expressing optimization problems for QA. The objective is to determine a binary vector $\mathbf{x} \in \{0,1\}^n$ that minimizes the following quadratic function [12]:

$$E(\mathbf{x}) = \sum_{i,j} J_{ij}\, x_i x_j + \sum_i h_i\, x_i,$$

where h_i specifies the linear bias associated with variable i, and J_{ij} captures the pairwise coupling between variables i and j.

2.2 QUBO Construction for Database Optimization Problems

Our system currently supports three representative database optimization tasks: join order optimization [18], multiple query optimization [21], and index selection [22]. We briefly summarize the corresponding QUBO constructions used in our demo.

Join Order Optimization. We follow prior work on QUBO encoding for left-deep join trees [18]. The construction combines four construction blocks: (1) H_{Va} enforces that each join step contains the correct number of relations; (2) H_{Vb} ensures that each join step introduces exactly one new relation, which then remains active in all later steps; (3) H_p activates a predicate only after both endpoint relations appear in the current step; and (4) H_{Cost} approximates logarithmic intermediate-result growth using relation cardinalities and predicate selectivities. The resulting join order objective is

$$E_{JO} = A\big(H_{Va} + H_{Vb} + H_p\big) + H_{Cost}, \tag{1}$$

where A amplifies violations of constraints as a penalty weight.

Multiple Query Optimization. We follow the QUBO construction framework proposed in prior work [21]. The objective consists of three construction blocks: (1) H_{Sel} enforces that exactly one execution plan is selected for each query; (2) H_{Share} captures consistency between selected plans and materialized intermediates; and (3) H_{Cost} models total execution cost, including plan execution costs and materialization costs, while subtracting reuse savings enabled by shared intermediates. The resulting objective is given by

$$E_{MQO} = A\,H_{Sel} + B\,H_{Share} + H_{Cost}, \tag{2}$$

where A and B are penalty weights that enforce constraints.

Index Selection. We follow prior formulations [22] to encode index selection under storage and workload constraints. The construction consists of three construction blocks: (1) $H_{Storage}$ enforces a storage budget constraint; (2) $H_{Conflict}$ captures mutual exclusion constraints among incompatible indexes (e.g., overlapping or redundant indexes), modeled as pairwise penalties that prevent simultaneous selection; and (3) H_{Cost} models workload execution cost under selected indexes. The resulting index selection objective is

$$E_{IS} = A\,H_{Storage} + B\,H_{Conflict} + H_{Cost}, \tag{3}$$

where A and B are penalty weights enforcing feasibility constraints.

These three classical database optimization tasks generate increasingly large-scale QUBOs as the problem size grows. We use them in the following scenarios to demonstrate QUBO construction, decomposition, fusion planning, and execution.

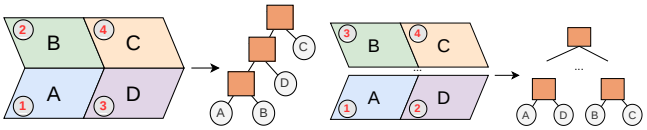

Figure 1: Fusion Tree Structure.

3 FUSION PLANNING

The main technical focus of this demonstration is to show how large-scale QUBOs are fused after decomposition, illustrating a practical approach to solving instances that exceed current quantum hardware limits. Once an oversized QUBO is partitioned into smaller components, it is not sufficient to solve them in isolation. The system must also decide how to recombine partial solutions while accounting for boundary interfaces. Inspired by join ordering in DBMSs, we formulate this procedure with proposing a cost function and merge strategies for boundary-aware fusion.

3.1 From Decomposition to Boundary Interfaces

Let a QUBO be defined over binary variables $\mathbf{x}$ with energy $E(\mathbf{x})$. After decomposition, the variables are divided into disjoint components $\{C_1, \ldots, C_m\}$. The resulting subproblems are not independent because some quadratic terms (i.e., couplings) connect variables

across different components. These cross-component couplings induce boundary interfaces that must be handled during composition.

Therefore, the decomposition produces two outputs. The first output is the set of sub-QUBOs that can be optimized separately. The second output is the boundary interface that contains the cut couplings. If K denotes the joint assignment of boundary variables, then the optimization target can be rewritten as a combination of intra-component energies and cross-component contributions:

$$\min_{\mathbf{x}} E(\mathbf{x}) \longrightarrow \sum_{i=1}^{m} \min_{\mathbf{x}_{C_i}} E_i(\mathbf{x}_{C_i}) + \min_{K} E_{cut}(K). \tag{4}$$

This expression highlights the system challenge: even if each component is solved well, the final solution quality is governed by how cross-component interactions are handled during merging.

3.2 Fusion Planning as a Tree

We plan the fusion as a binary tree T whose m leaves are the decomposed sub-QUBOs, and in which each internal node v merges the partial solutions of its two children $L(v)$ and $R(v)$. As a join tree fixes the order in which relational inputs are combined, a fusion tree fixes the order in which components are composed. As shown in Figure 1, the fusion tree can also take different shapes. A *linear* tree repeatedly merges a new component into an existing partial result. A *bushy* tree may merge multiple-component subtree in each merge. As a query optimizer costs a join tree [20], we score a fusion tree by summing a cost over its merges:

$$C(T) = \sum_{v \in \text{Internal}(T)} C_{merge}\big(L(v), R(v)\big). \tag{5}$$

Each merge resolves the couplings on the boundary. Resolving a boundary late is more expensive, because the boundary assignment must then accommodate a larger scope of merged components. Each merge is therefore priced by its boundary coupling, weighted by the number of components in the merged result:

$$C_{merge}(L, R) = |L \cup R| \cdot \text{cut}(L, R). \tag{6}$$

Here, $\text{cut}(L, R) = \sum_{u \in L,\, v \in R} |J_{uv}|$ adds up the strengths of couplings across the two sides being merged, and $|L \cup R|$ is the number of decomposed components in the merged result. Both quantities directly come from the cut structure alone, so *QFusion* can plan whole trees before solving any component on the quantum annealer. We use dynamic programming [14] to find the lowest-cost tree.

3.3 Merge Strategies with Boundary Awareness

Solving each component C_i on the quantum annealer returns a *candidate set* $\mathcal{V}_i = \{v_{i1}, \ldots, v_{ik}\}$ of k different assignments with low QUBO energies. At each internal node of the fusion tree, a merge strategy is applied to combine its two children. We expose three strategies with different levels of boundary awareness.

Direct Fusion. This is the simplest strategy: it selects the best candidate from each side independently and concatenates the two into a merged assignment. It incurs essentially no additional cost for interface coordination and is therefore fast, but it ignores cross-boundary interactions during composition and may yield poor global quality when cut couplings are strong.

Top-k Merge. This strategy evaluates pairwise combinations of the two children's candidate sets and retains the k best combinations as the merged node's candidate set. If $\mathcal{V}_A$ and $\mathcal{V}_B$ denote the retained candidate sets from the two children, the chosen pair is

$$(\hat{v}_A, \hat{v}_B) = \arg \min_{v_A \in \mathcal{V}_A,\, v_B \in \mathcal{V}_B} \big(E_A(v_A) + E_B(v_B) + E_{cut}(v_A, v_B)\big). \tag{7}$$

In our demo, we focus on $k = 2$, which offers a practical trade-off between merge overhead and improved boundary handling.

Conditioned Fusion. This strategy makes the composition procedure asymmetric. It first fixes an upstream candidate and then solves or re-ranks the downstream side under the boundary conditions induced by that upstream assignment. If v_A is the selected upstream candidate, the downstream optimization is performed over a conditioned objective:

$$\hat{v}_B = \arg \min_{v_B} \big(E_B(v_B) + E_{cut}(v_A, v_B)\big). \tag{8}$$

This strategy can reduce boundary conflicts early, especially in deep fusion trees where a poor upstream decision would otherwise propagate instability to many later steps.

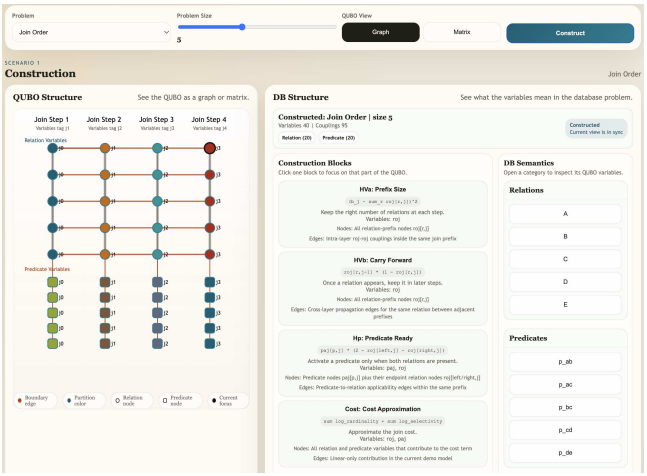

Figure 2: QUBO Construction with DB Semantics (S1).

4 DEMONSTRATION SCENARIOS

This demonstration presents three scenarios that together form an end-to-end workflow for solving large-scale QUBOs: problem-specific construction with database semantics, decomposition with boundary inspection, and boundary-aware fusion and execution. We evaluate three database optimization workloads: join order optimization, multiple query optimization, and index selection. Scenarios are presented in sequence, and each stage consumes artifacts produced by the previous stage. Note that for visualization clarity, we keep the number of variables in each component moderate.

Scenario 1: QUBO Construction and Database Semantics. The user first selects one of three workloads (join order optimization, multiple query optimization, or index selection), specifies the problem size, and examines how the resulting database problem is encoded as a QUBO. As shown in Figure 2, the left panel visualizes the QUBO as either a graph (more intuitive for inspection) or a matrix (aligned with the underlying quadratic form), while the right panel summarizes key statistics and the corresponding database semantics. For join order optimization, the interface highlights relation

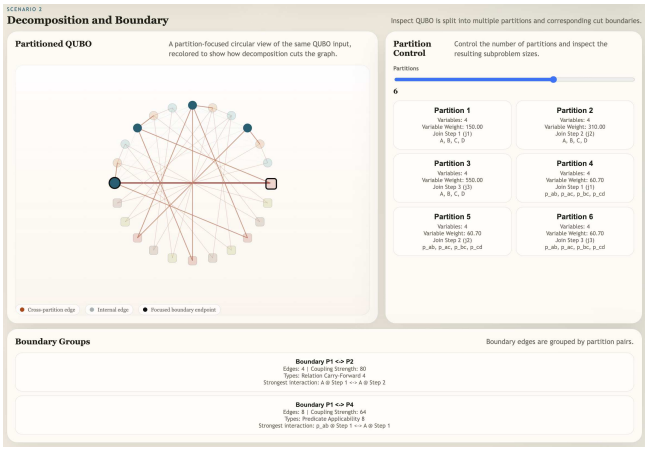

Figure 3: Decomposition and Boundary Inspection (S2).

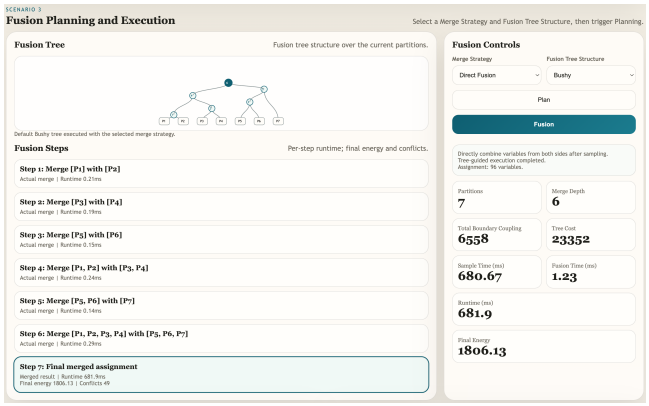

Figure 4: Boundary-Aware Fusion and Execution (S3).

variables, predicate variables, and the main construction blocks of the QUBO. For multiple query optimization, it groups variables by queries and candidate plans. For index selection, it presents tables, candidate indexes, and storage-budget auxiliary variables. Users can click semantic objects or construction blocks to locate the associated QUBO variables and couplings. This scenario establishes the input for subsequent decomposition and fusion.

Scenario 2: Decomposition and Boundary Inspection. This scenario shows how the same QUBO is partitioned. Users adjust the partition count with a slider and immediately observe the partition-aware structure. As shown in Figure 3, node colors indicate partition membership, and cross-partition couplings reveal boundary interfaces. A boundary-group panel aggregates boundary groups by partition pairs and reports summary statistics for each interface. This view makes the decomposition trade-off explicit: smaller subproblems improve hardware feasibility, while larger and denser boundaries increase coordination complexity during fusion.

Scenario 3: Boundary-Aware Fusion Planning and Execution on a Quantum Annealer. This scenario demonstrates fusion planning and execution on an actual quantum annealer. Users first choose a merge strategy and a fusion tree structure, which define the default fusion tree shown in the interface. They optionally trigger a planning step that rewrites this tree using a cost-based planner, yielding an alternative fusion order. As shown in Figure 4, the interface visualizes both the resulting fusion tree and the step-by-step fusion process, allowing users to compare linear and bushy structures as well as default and planned orders. This demonstration scenario supports three merge strategies: *Direct Fusion*, *Top-k Merge*, and *Conditioned Fusion*. After the tree is finalized, users launch actual execution on the quantum annealer, and the system reports runtime statistics, including (quantum annealing) sampling time, fusion time, and end-to-end runtime, along with the boundary coupling, the planned tree cost, and final energy, which indicates the solution quality of the given QUBO. Overall, this scenario highlights that, after decomposition, solution quality depends not only on the chosen merge strategy, but also on the fusion order. Users can directly inspect and compare both effects in the interface.

ACKNOWLEDGMENTS

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

REFERENCES

[1] Abhishek Awasthi, Nico Kraus, et al. 2024. Real World Application of Quantum-Classical Optimization for Production Scheduling. In *IEEE QCE*.
[2] Umut Çalikyilmaz et al. 2023. Opportunities for Quantum Acceleration of Databases: Optimization of Queries and Transaction Schedules. *Proc. VLDB Endow.* (2023).
[3] D-Wave Systems. 2022. The D-Wave Advantage2 System (White Paper). https://www.dwavequantum.com/media/wakjcpsf/adv2_4400q_whitepaper-1.pdf.
[4] IBM Quantum. 2022. Cloud access to quantum computers provided by IBM. https://quantum-computing.ibm.com.
[5] Manish Kesarwani and Jayant R. Haritsa. 2024. Index Advisors on Quantum Platforms. *Proc. VLDB Endow.* (2024).
[6] Hanwen Liu et al. 2026. How Can Quantum Computing and Databases Meet "in Practice"? From Algorithms to Systems. *Proc. VLDB Endow.* 19, 12 (2026).
[7] Hanwen Liu, Abhishek Kumar, et al. 2025. Hybrid Quantum-Classical Optimization for Bushy Join Trees. In *Q-Data '25@SIGMOD.*
[8] Hanwen Liu, Abhishek Kumar, et al. 2026. QDBO: A Real-time Quantum-augmented Database System Optimizer. *Proc. VLDB Endow.* 19, 11 (2026), 3606–3620. https://doi.org/10.14778/3836663.3836712
[9] Hanwen Liu and Ibrahim Sabek. 2026. Is Quantum Computing Ready for Real-Time Database Optimization?. In *ICDE '26 Lightning Talk.*
[10] Hanwen Liu and Ibrahim Sabek. 2026. Towards a Hybrid Quantum-Classical Computing Framework for Database Optimization Problems in Real Time Setup. In *ICDE '26 DEFT.*
[11] Hanwen Liu, Pranshi Saxena, et al. 2025. Optimizing Join Orders via Constrained Quadratic Models (Abstract). In *QDML '25@ICDE.*
[12] Hanwen Liu, Federico Spedalieri, et al. 2025. A Demonstration of Q2O: Quantum-augmented Query Optimizer. In *VLDB '25.*
[13] Andrew Lucas. 2014. Ising formulations of many NP problems. *Frontiers in Physics* 2 (2014). https://doi.org/10.3389/fphy.2014.00005
[14] Guido Moerkotte et al. 2008. Dynamic Programming Strikes Back. In *SIGMOD '08.*
[15] Nitin Nayak et al. 2023. Constructing Optimal Bushy Join Trees by Solving QUBO Problems on Quantum Hardware and Simulators. In *BiDEDE '23@SIGMOD.*
[16] Manuel Schönberger et al. 2025. Hybrid Mixed Integer Linear Programming for Large-Scale Join Order Optimisation. *arXiv:2510.20308* (2025).
[17] Manuel Schönberger, Stefanie Scherzinger, et al. 2023. Ready to Leap (by Co-Design)? Join Order Optimisation on Quantum Hardware. *PACMMOD* (2023).
[18] Manuel Schönberger, Immanuel Trummer, et al. 2023. Quantum-Inspired Digital Annealing for Join Ordering. In *VLDB '23.*
[19] Manuel Schönberger, Immanuel Trummer, et al. 2026. Large-Scale Multiple Query Optimisation with Incremental Quantum(-Inspired) Annealing. In *SIGMOD '26.*
[20] P. Griffiths Selinger, M. M. Astrahan, et al. 1979. Access Path Selection in a Relational Database Management System. In *SIGMOD '79.*
[21] Immanuel Trummer and Christoph Koch. 2016. Multiple Query Optimization on the D-Wave 2X Adiabatic Quantum Computer. In *VLDB '16.*
[22] Immanuel Trummer and Davide Venturelli. 2024. Leveraging Quantum Computing for Database Index Selection. In *Q-Data '24@SIGMOD.*