# DEDALUS: A Quantum-Enhanced End-to-End Framework for Cost-Aware Join Order Optimization with Search Space Pruning

Emmanouil Limnaios
Markos Stergiopoulos
George T. Stamatiou
Efthymios Papageorgiou
elimnaios@csd.uoc.gr
stamatiou@ics.forth.gr
papageorgiou@csd.uoc.gr
csd4043@csd.uoc.gr
University of Crete
Heraklion, Greece

Vasilis Efthymiou
vefthym@hua.gr
Harokopio University of Athens
Athens, Greece

Dimitris Loupas
Dimitrios Tsourounis
Kostas Blekos
Aggelos Tsikas
dcl@quantumtech.ai
dct@quantumtech.ai
cab@quantumtech.ai
ait@quantumtech.ai
Quantum Neural Technologies SA
Athens, Greece

Dimitris Plexousakis
Kostas Magoutis
Yannis Tzitzikas
Haridimos Kondylakis
dp@ics.forth.gr
magoutis@ics.forth.gr
tzitzik@ics.forth.gr
kondylak@csd.uoc.gr
University of Crete, FORTH-ICS
Heraklion, Greece

## Abstract

Join order optimization remains one of the most critical and difficult tasks in relational database systems. Even modern cost-based optimizers struggle with exponentially large search spaces and error-prone cardinality estimates, often leading to suboptimal plans that can degrade performance by orders of magnitude. As analytical workloads grow in complexity and scale, there is increasing interest in exploring quantum and quantum-inspired optimization methods to tackle these inherently combinatorial problems. In this demonstration, we present Dedalus, a quantum-enhanced framework that formulates join ordering as a Quadratic Unconstrained Binary Optimization (QUBO) problem enriched with database statistics and structural constraints. Dedalus introduces a multi-factor cost model that yields more robust subset weights and a join-graph–guided pruning strategy that eliminates structurally invalid subsets, substantially reducing the dimensionality of the QUBO instance. The system integrates classical simulated annealing, gate-based quantum simulators, exact solvers, and support for various quantum backends, within a single end-to-end pipeline. The demonstration allows users to load SQL queries, inspect the pruned search space, analyze solver-produced join trees, and compare them with PostgreSQL's native plans, showing how quantum-inspired optimization can already offer practical benefits for join ordering.

## 1 Introduction

Join Order Optimization (JOO) remains a central challenge in relational database systems. Despite decades of improvements, optimizers still suffer from cardinality estimation error propagation, limiting plan quality [1]. Simultaneously, the structure of bushy or alternative (sub-optimal) join trees yields an exponentially growing search space [8]. Emerging quantum and quantum-inspired algorithms offer novel combinatorial optimization strategies, yet naïve encodings suffer from scalability issues [6].

Nayak et al. [8] encode bushy join ordering as a QUBO and solve it on quantum annealers/simulators, demonstrating feasibility for small instances. Their work exposes both the potential of quantum-inspired JOO and limitations from variable blow-up and noisy cardinalities, but evaluates only tree validity and optimality—not execution on a real DBMS. Our demo extends this line with a richer, statistics-driven cost model and a join-graph pruning layer that reduces dimensionality before solving, improving scalability and interpretability. A related hybrid strategy is proposed in [6], combining structural encodings with staged decomposition and adaptive solver selection. However, it omits multi-factor weight synthesis from live catalog statistics and lacks connectivity-based subset elimination before binary formulation. In contrast, we pair semantic statistics enrichment with join-graph pruning prior to QUBO construction, producing a leaner variable set and a cost surface closer to classical optimization dynamics. Variational Quantum Circuit (VQC) methods [12] instead learn join orders from historical workloads via hybrid training. Although compact and generalizable, they abstract away multi-factor cost semantics and skip explicit pruning of disconnected subsets. Our formulation is complementary: it preserves structural transparency (subset connectivity, explicit penalties) and a tunable cost model while remaining solver-agnostic—enabling future integration of learned proposal distributions to guide search efficiently.

In this paper we demonstrate Dedalus, a practical bridge between database internals and quantum-enhanced JOO. Our key idea is to transform multi-way join enumeration into a QUBO formulation enriched by: (a) a multi-factor cost model sourced from live database statistics, and (b) an aggressive pruning strategy based on the query's inferred join graph. The result is a reduced QUBO instance, fed into various (real or simulated) quantum solvers. Compared to combinatorial variable generation, pruning shrinks problem dimensionality, improving solver tractability, stability, and accuracy, leading to better query execution plans, identified faster. Crucially, our work provides an end-to-end implementation that bridges the gap between abstract problem formulations and practical application, demonstrating a complete pipeline from SQL query to solver execution against a live database. To the best of our knowledge, Dedalus is the first end-to-end system that enables the execution and optimization of arbitrary SQL queries over a live database through a fully integrated QUBO-based quantum–classical pipeline.

*EDBT '26, Tampere (Finland)*

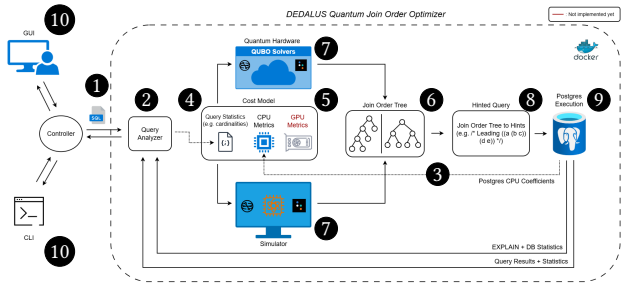



**Figure 1: High-level workflow.**

## 2 System Overview

DEDALUS follows a modular, end-to-end architecture that transforms query answering of an SQL query into a quantum-amenable optimization problem and resulting in an optimized plan executed over a live database. The system integrates classical database statistics extraction, QUBO formulation with structural pruning, and hybrid quantum–classical solvers within a unified pipeline. Its components operate as a staged workflow—analysis, encoding, optimization, hint generation, and execution—enabling transparent exploration of quantum-inspired join order optimization. Figure 1 illustrates the end-to-end workflow, where the numbered components are described in detail in the sequel.

❶ **SQL Intake:** The user provides an SQL query. The system parses it to extract aliases, predicates, and join conditions, forming an initial join graph.

❷ **Subset Enumeration:** The power-set of relations $\mathcal{S}$ (excluding singletons) is generated as the initial set of candidate variables for the QUBO formulation.

❸ **Statistics Extraction:** DEDALUS gathers cardinality information and statistics from the underlying PostgreSQL catalog. This includes per-column distinct value counts, histogram-derived distribution summaries, correlation statistics, predicate selectivity indicators, and index metadata. These statistics collectively equip the framework with the information needed to compute stable, multi-factor cost weights, while also enabling conservative adjustments when metadata is missing or noisy.

❹ **Stats-Aware Cost Weighting:** Each subset $S \in \mathcal{S}$, receives a raw weight $w(S)$ from a cost model that integrates filtered cardinalities, per-column distinct value counts, predicate complexity, skew, and variance, into a single, stable score. In particular, for a relation $r$, we shall use $cost(r)$ to proxy predicate evaluation cost for $r$, $skew_r$ to encode its distribution skew, and $var_r$ to encode its estimation uncertainty. For a subset $S$, $bonus(S)$ is a bonus applied to highly selective, index-supported filters, $penalty(S)$ raises cost for structural conditions that potentially increase cost (such as those mentioned bellow in step ❺), while $\bar{N}_S$ is a heuristic conservative approximation of the filtered cardinality. Finally, we use a set of constants $c_{size}, c_{pred}, c_{skew}, c_{var}$ which are cost coefficients gathered from PostgreSQL catalogs. Based on all the above, $w(S)$ is eventually defined as:

$$w(S) = c_{size}\,(1+\bar{N}_S) + c_{pred}\sum_{r\in S} cost(r) + c_{skew}\,\mathrm{avg}_{r\in S}(skew_r) + c_{var}\,\mathrm{avg}_{r\in S}(var_r) - bonus(S) + penalty(S)\,. \quad (1)$$

❺ **Join-Graph Pruning & Normalization:** The system removes disconnected or cycle-inducing subsets based on the join graph. This critical step preserves the potential for bushy join trees while controlling variable blow-up. Remaining weights are then normalized to a stable range for the solvers. Optional log-space scaling (e.g., $\log(1+\bar{N}_S)$) dampens multiplicative explosion and improves stability before normalization [10].

❻ **QUBO Construction:** A binary variable $x_S$ is mapped to each pruned subset $S$, and quadratic penalty terms are added to enforce the constraint that overlapping subsets cannot co-exist in a valid join tree. In particular, let $\mathcal{R}$ be the set of base relations in the query and let the join graph $G = (\mathcal{R}, E)$ contain an edge for every join predicate. We derive a pruned variable set

$$V = \{x_S \mid S \subseteq \mathcal{R},\ |S| \geq 2\}\,, \quad (2)$$

where $S$ induces a connected subgraph of $G$. Each binary decision variable $x_S$ indicates that subset $S$ participates as an intermediate join result along a plan construction path. Pruning subsets eliminates structurally impossible components early and shrinks $|V|$ versus the full power set, enabling denser, semantically faithful QUBO instances that remain solver-agnostic across multiple quantum backends.

For every pruned subset $S$ we compute a multi-factor, statistics-aware cost weight $w_S$ as shown in (1). Optionally, log-scaling and post-pruning normalization produce stable relative magnitudes for quantum solvers. To enforce mutual exclusivity of overlapping subsets that cannot co-exist in a single valid step, we construct the *conflict set*:

$$C = \{(A,B) \mid A \cap B \neq \emptyset,\ A \not\subseteq B,\ B \not\subseteq A\}\,. \quad (3)$$

A QUBO objective function is typically written as $H(\mathbf{x}) = \mathbf{x}^{\mathrm{T}}Q\mathbf{x} + c^{\mathrm{T}}\mathbf{x} + const.$, where $\mathbf{x} \in \{0,1\}^n$, $Q$ is the matrix of pairwise interaction coefficients, $c$ is the vector of linear biases and $n$ is the number of binary variables. This function is computationally equivalent, via a change of variables, to the Ising Hamiltonian used as the cost (total energy) function [7] in quantum annealing and variational quantum algorithms such as the Quantum Approximate Optimization Algorithm (QAOA) [2]. For a constrained optimization problem translated to a QUBO, it is required to add quadratic penalty terms (absorbed into the QUBO coefficients) to the objective function to punish constraint violations. Following a penalty design adapted from [8], but applied after connectivity pruning and enriched weighting, the opened QUBO objective function reads:

$$H(x) = \sum_{S\in V}(w_S - \lambda)\,x_S \;+\; \lambda \sum_{(A,B)\in C} x_A x_B\,. \quad (4)$$

Eq. (4) introduces one binary variable $x_S \in \{0,1\}$ for each pruned subset $S \in V$. Variables $x_A$ and $x_B$ are defined similarly to $x_S$, for the subsets $(A,B) \in V$, drawn from the conflict set $C$. Its linear coefficients $(w_S - \lambda)$ are the statistics-aware weights from Eq. (1) shifted by $\lambda$ for penalty balancing, and its quadratic term penalizes conflicts $(A,B) \in C$. Thus, Eq. (4) is the expanded QUBO objective whose linear part derives from Eq. (1) and whose quadratic part encodes feasibility constraints. We set $\lambda = 2\,\max_{S\in V}(w_S)$ so a conflicting activation is dominated by its penalty. Minimizing $H$ yields a bitstring selecting a feasible set of connected subsets interpretable as a join sequence. Each variable $x_S$ maps to a logical qubit with bias $(w_S - \lambda)$ and couplings from conflict penalties. Connectivity-aware pruning lowers $|V|$ and coupling density, aiding minor embedding on annealers and reducing circuit width on gate-based backends.

❼ **Hybrid Solver Execution:** The QUBO instance is dispatched to a selected backend. The framework supports both annealing-based and gate-based simulators, as well as real quantum backends.

More specifically, it includes Simulated Annealing and other classical algorithms, the gate-based Aer quantum simulator capable of running the Quantum Approximate Optimization Algorithm (QAOA) [2] and the Variational Quantum Eigensolver (VQE) [9], as well as exact enumeration for small instances. Our framework also supports currently available IBM quantum backends (e.g. *ibm_torino*) and D-Wave's quantum annealers (e.g. *Advantage_system6.4*) as well as hybrid solvers (e.g. *Hybrid BQM solver*). Thus, providing the appropriate token/API key, the user may connect to available quantum hardware via the quantum cloud service of each provider, allowing for experimentation, performance metrics, and cross-validation.

In the case of annealing, to balance solution quality with robustness, Dedalus employs a lightweight iterative annealing procedure rather than a single static schedule [4]. Each iteration $k$ executes a short annealing run with a random seed while gradually expanding the inverse-temperature interval to $[0.1,\ 10 + 2k]$. Early iterations, therefore, explore broadly, whereas later ones freeze more aggressively as the schedule deepens. The number of sweeps is also adapted in a simple deterministic way: when a new best energy is observed, the sweep count of the next iteration is increased; when progress stalls, it is reduced. This compact "intensify-on-success, skip-on-stagnation" mechanism avoids spending computation in unproductive regions and leads to more reliable convergence than fixed-budget annealing, consistent with prior observations on schedule shaping for Ising systems [3]. The procedure is deliberately lightweight so that total latency remains compatible with live demonstrations.

Hyperparameters for all solvers are handled using predefined defaults chosen for stability rather than problem-specific tuning. For simulated or quantum annealing, the parameter *num_reads* controls the number of sampled bitstrings; increasing it raises solution reliability at the predictable cost of additional runtime. For QAOA, the depth $p$ corresponds to the number of cost–mixer layers [2] and therefore directly determines circuit depth and noise sensitivity on NISQ hardware. Dedalus does not attempt automated hyperparameter tuning and instead exposes a small, fixed set of stable options (e.g., COBYLA or SPSA for classical optimization). This avoids introducing optimizer-dependent variability while ensuring reproducible behavior across backends.

In each case, the selected solver will return a bitstring which is subsequently decoded to obtain the solution of our initial JOO problem, i.e., a proposed join tree.

❽ **Hints:** After decoding the solver's bitstring into a join tree, Dedalus translates the resulting join order into a PostgreSQL-compatible join-order hint. Hints provide the optimizer with an explicit ordering of relations, enforcing PostgreSQL to follow the solver-derived plan rather than exploring its own search space.

❾ **Execution:** The hinted SQL query is executed against the live PostgreSQL instance, enabling direct measurement of planning latency, execution time, and output cardinalities. This step provides end-to-end validation of whether the quantum-inspired plan yields practical runtime improvements over the native optimizer.

❿ **Visualization:** The framework allows through a user-friendly interface the visualization of the pruned variable counts, the comparison of the resulting tree against the native PostgreSQL plan, and the inspection of performance differences. It also supports custom databases and differing workloads, as well as a benchmark harness for systematic evaluation (see Fig. 2).

## 3 Demonstration

We showcase the full Dedalus pipeline in action, allowing attendees to observe how an arbitrary SQL query is transformed into a pruned QUBO instance, optimized by various solvers. Although we have results from classical, quantum, and hybrid solvers, due to monetary restrictions in the demo, we will show classical results. The pipeline will be executed on a live PostgreSQL database. Through an interactive GUI, users can inspect every stage—from statistics extraction and subset pruning to solver output, join-tree reconstruction, and final query execution—gaining hands-on insight into how quantum-inspired methods can influence real-world query optimization. The demo highlights both the transparency of the system's internals and the practical performance implications of solver-derived join plans.

**Datasets and Setup.** Dedalus runs on a containerized environment comprising Python 3.11, PostgreSQL 15, Qiskit v2.1.1, and D-Wave's Ocean SDK v9.1.0. The demonstration platform is a ThinkPad T14s Gen 1 with an AMD Ryzen 7 PRO 4750U CPU and 16GB RAM. Three benchmark datasets will be pre-loaded for interactive experimentation: TPC-H (SF-1) [11], the Join Order Benchmark (JOB) [5], and a synthetic SampleDB.

**Solvers.** In our demonstration, we will use classical simulated annealing using D-Wave's Ocean SDK and related libraries, allowing users to efficiently explore post-pruning energy landscapes. The latter is the preferred method, as it better serves our problem formulation. Further, we will show execution using the IBM's Qiskit and the Aer quantum simulator to run QAOA/VQE algorithms for gate-based experimentation and performance assessment. We also include exact enumeration, i.e. exhaustive search, when the problem size is sufficiently small and able to run on our demo machine. The live demonstration will unfold through the following interactive steps, illustrated in Figure 2.

① **Query Selection:** The attendee chooses one of the bundled multi-join SQL workloads (TPC-H, JOB, or SampleDB) or provides a custom SQL query. Dedalus parses the query and constructs the initial join graph.

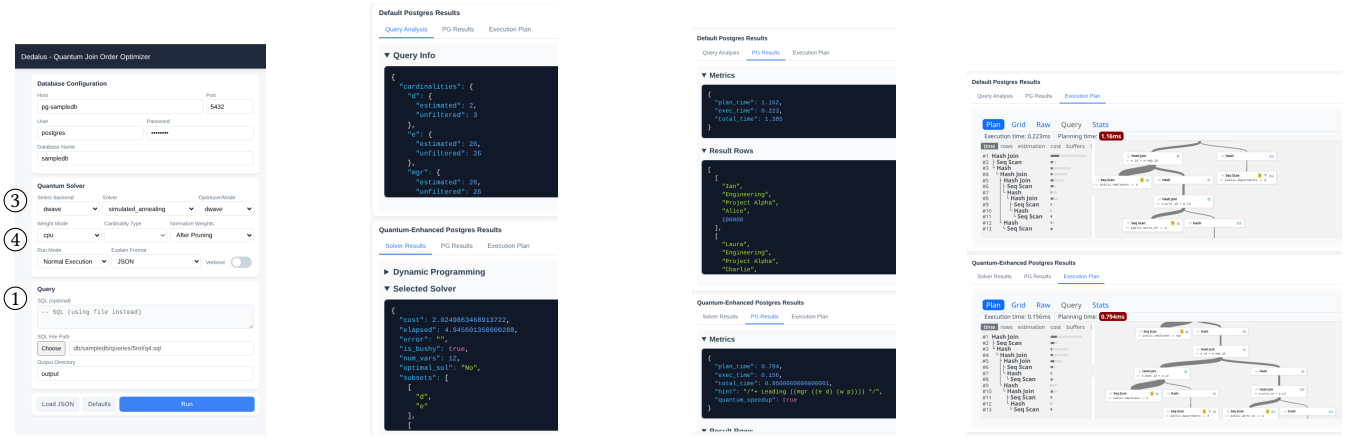
② **Stats View:** The GUI displays base table cardinalities, column-level statistics (distinct value counts, histograms, correlations), and index metadata drawn from the PostgreSQL catalog. These feed into the multi-factor cost model.

③ **Solver Comparison:** Participants select among simulated annealing, QAOA/VQE simulators, or exact enumeration. Dedalus logs solver run times, energies, convergence behavior, and validity checks for each backend.

④ **Exploration Controls:** Users can toggle cost-model variants, normalization/pruning strategies, and solver backends in real time to study their effect on the resulting plans.

⑤ **Inspection & Benchmarking:** The system decodes the solver's bitstring into a join tree for comparison against PostgreSQL's native plan, then translates it into join-order hints to execute the query and report performance metrics.

⑥ **Benchmark Results:** The system presents side-by-side planning and execution metrics, along with plots for deeper analysis, enabling participants to assess when quantum-inspired search offers competitive plans. In practice, the derived join orders often achieve lower estimated cost, avoid unnecessary cross-products, and, through adaptive annealing, converge to high-quality solutions in fewer iterations on medium-size queries (up to seven relations). Pruning and faster convergence commonly reduce




(a) Configuration Pane (b) ② Analysis Tab (c) ⑤ Results Trees (d) ⑥ Plan Visualization Tab

**Figure 2: Graphical user interface: configuration, analysis, join trees, and benchmarking.**

planning time and frequently yield faster—or at least comparable—execution times, although some highly selective or filter-heavy queries show no consistent gains. In many cases, the resulting execution plan is also faster than PostgreSQL's native choice.

## 4 Conclusion and Positioning

We present Dedalus, the first cost-aware, end-to-end framework for join order optimization, that enables further plan search space exploration via a pruned QUBO formulation, solvable by hybrid quantum-classical methods. Dedalus is able to identify optimal query plans using quantum simulators as well as currently available annealing-based and gate-based quantum hardware, leading in many cases to better query execution plans with tangible benefits on query efficiency.

A key insight underlying Dedalus is that its statistics-enriched and connectivity-aware QUBO formulation significantly reduces the logical problem size, eliminating large portions of the variable space that naïve encodings must still represent. Since each retained subset corresponds to a logical qubit, the pruning phase directly lowers the number of qubits required, consequently improving embedding, reducing chain lengths, and enabling deeper optimization schedules even on near-term quantum hardware. This more compact and semantically faithful formulation yields higher-quality energy landscapes, allowing solvers to converge to plans that are consistently closer to the best enumerated solution than those produced by earlier QUBO-based approaches. Moreover, across evaluated workloads on actual quantum hardware, the solver derives join orders that often dominate PostgreSQL's native plans, reducing estimated cost and matching or outperforming the classical optimizer's best alternatives–while achieving these improvements with markedly faster exploration of the search space.

## References

[1] Alberto Belussi, Sara Migliorini, and Ahmed Eldawy. 2024. A Generic Machine Learning Model for Spatial Query Optimization based on Spatial Embeddings. *ACM TSAS* (12 2024). doi:10.1145/3657633

[2] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. 2014. A Quantum Approximate Optimization Algorithm. arXiv:1411.4028 https://arxiv.org/abs/1411.4028

[3] Sergei V Isakov, Ilia N Zintchenko, Troels F Rønnow, and Matthias Troyer. 2015. Optimised simulated annealing for Ising spin glasses. *Computer Physics Communications* 192 (2015), 265–271. doi:10.1016/j.cpc.2015.02.015

[4] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. 1983. Optimization by Simulated Annealing. *Science* 220, 4598 (1983), 671–680. doi:10.1126/science.220.4598.671

[5] 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. doi:10.14778/2850583.2850594

[6] Hanwen Liu, Abhishek Kumar, Federico Spedalieri, and Ibrahim Sabek. 2025. Hybrid Quantum-Classical Optimization for Bushy Join Trees. In *Q-Data Workshop*. 20–24. doi:10.1145/3736393.3736695

[7] Andrew Lucas. 2014. Ising formulations of many NP problems. *Frontiers in Physics* Volume 2 - 2014 (2014). doi:10.3389/fphy.2014.00005

[8] 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 *BiDEDE Workshop*. Article 7, 7 pages. doi:10.1145/3579142.3594298

[9] Alberto Peruzzo, Jarrod McClean, Peter Shadbolt, Man-Hong Yung, Xiao-Qi Zhou, Peter J. Love, Alán Aspuru-Guzik, and Jeremy L. O'Brien. 2014. A variational eigenvalue solver on a photonic quantum processor. *Nature Communications* 5, 1 (July 2014). doi:10.1038/ncomms5213

[10] Pranshi Saxena, Ibrahim Sabek, and Federico Spedalieri. 2024. Constrained Quadratic Model for Optimizing Join Orders. In *Q-Data Workshop*. 38–44. doi:10.1145/3665225.3665447

[11] Transaction Processing Performance Council (TPC). 2025. TPC Benchmark H Standard Specification. http://www.tpc.org/tpch/.

[12] Tobias Winker, Umut Çalikyilmaz, Le Gruenwald, and Sven Groppe. 2023. Quantum Machine Learning for Join Order Optimization using Variational Quantum Circuits. In *BiDEDE Workshop*. Article 5, 7 pages. doi:10.1145/3579142.3594299