Expeditus: Database Optimization Based on Workload Analysis with Structural Query Clustering

Marlon Gonçalves Duarte[1], **Júlio César de Lira Marinho**[1], **Ângelo Rocalli Alencar Brayner**[1], **Wellington Franco**[2]

[1] Departamento de Computação – Universidade Federal do Ceará
[2] Kunumi Lab – Universidade Federal do Ceará (UFC)
Fortaleza – Ceará – Brazil

{marlongduarte,juliocesarliramarinho}@alu.ufc.br, brayner@dc.ufc.br

wellington@crateus.ufc.br

Abstract. *Workload analysis is fundamental to database performance management, as heterogeneous and complex workloads require automated identification of similar query patterns. In this context, this work investigates whether SQL query clustering based on structural embeddings extracted from the workload can support more efficient database tuning strategies. As an experimental foundation, we employ the TPC-H benchmark, whose queries were transformed into vector representations using the Qwen3-Embedding-0.6B model. After dimensionality reduction with UMAP, we applied the HDBSCAN algorithm to identify structurally coherent query clusters. We then evaluated whether indexes derived from a representative query within each cluster could improve the performance of the remaining queries in the same group. The experiments resulted in seven structurally cohesive clusters, demonstrating the model's ability to capture structural similarities across distinct queries. The results indicate that optimizing a single query can benefit other queries within the same cluster, supporting the use of structure-driven tuning strategies for complex database workloads.*

1. Introduction

Database applications have become very complex, dealing with a huge volume of data and database objects (e.g., tables, indexes, materialized views). Concurrently, low query response time and high transaction throughput have emerged as mandatory requirements to be ensured by database management systems (DBMSs). In such a scenario, systematic database fine-tuning is a critical requirement for achieving and maintaining optimal performance in DBMSs.

Workload is central to database fine-tuning process due to fact that a DBMS does not have a single "optimal" configuration. In other words, a DBMS performance entirely depends on the characteristics of the workload it executes. Tuning without understanding workload is essentially blind optimization. In scenarios with high SQL query submission rates and substantial heterogeneity of access patterns, workload analysis becomes particularly challenging [Guabtni et al. 2013]. In such environments, even small changes in request profiles can intensify contention, increase latency, and induce significant variations in the execution plans selected by the optimizer.

In modern systems with self-tuning capabilities, the workload ceases to be a predominantly static artifact and instead becomes a central signal for automated operational decision-making [Oloruntoba 2025]. In this sense, autonomous systems can employ artificial intelligence and machine learning techniques to continuously monitor workload variations in (near) real time.

According to [Oloruntoba 2025], the incorporation of predictive models enables the anticipation of performance trends and the execution of dynamic adjustments in configuration, index selection, and proactive resource allocation (predictive scaling) before bottlenecks materialize. In this way, workload management evolves into an adaptive paradigm, yielding gains in scalability, efficiency, and reduced operational costs, while simultaneously decreasing the need for constant human intervention.

Nonetheless, the coexistence of short, recurrent queries with high-cost analytical queries, combined with multiple concurrent application profiles, makes it difficult to distinguish stable behavioral patterns from anomalous events, thereby requiring continuous instrumentation and correlation of metrics (response time, cache hit ratio, I/O, locks, and lock wait times). Thus, optimization should consider not only individual queries but also the aggregated effect of the workload on CPU, memory, and I/O, reinforcing the need for sampling strategies, query signature–based clustering, and representative observation windows to reduce analytical bias [do Amaral et al. 2025].

Text embedding models map SQL queries into vector representations that preserve semantic similarity while capturing structural differences [Cer et al. 2018, Almeida and Moura 2024]. Such representations enable the clustering of heterogeneous workloads, supporting the identification of similar query patterns and more effective workload-driven optimization.

This work presents a mechanism, denoted Expeditus, for constructing vector representations of SQL queries across two reference workloads and, based on these representations, performs clustering and analysis of SQL queries grouped by similarity in the vector space. The objective is to investigate the feasibility of using embeddings for query clustering and, consequently, to evaluate the impact of optimizations based on the creation of specific indexes.

In particular, we analyze whether, after cluster formation, the optimization of an individual query translates into performance improvements for the cluster as a whole, assuming that the queries were grouped according to structural similarity. In doing so, the goal is to reduce the number of optimization interventions required in the system, since certain actions may be applicable to multiple queries, even if they are not identical.

The remainder of this paper is organized as follows. Section 2 reviews related work on workload analysis, database optimization, and LLM-based Text-to-SQL systems. Section 3 presents the proposed architecture for workload analysis and SQL query clustering. Section 4 reports the experimental evaluation using the TPC-H benchmark, assessing the effectiveness of semantic embeddings and cluster-based index optimization. Finally, Sections 5 and 6 present the conclusions, future research directions, and the statement on the use of generative AI.

2. Related Work

In this section, we present some works that addressed research similar to this and that served as the basis for the formulation of our methodology.

2.1. A CBR Model for Workload Characterization in Autonomic Database Management System

[Shaheen et al. 2018] present a Case-Based Reasoning (CBR) model for workload characterization and adaptation in autonomous database management systems, addressing limitations of existing techniques that exhibit low accuracy in distinguishing between Online Transaction Processing (OLTP) and Decision Support Systems (DSS) profiles, which can lead to performance degradation. The proposed methodology consists of a framework based on MySQL feature vectors that support the retrieval, reuse, revision, and retention phases of the CBR cycle, enabling adaptive workload management. For validation, the approach was compared against traditional machine learning models, including Support Vector Machines (SVM) and Neural Networks (NN), using the standard TPC-C and TPC-H benchmarks. The results indicate that the CBR model achieves superior accuracy and effectiveness, with a 15% improvement in adaptation capability after incorporating new characterization metrics, and statistical validation through Friedman tests confirms its ranking as the best-performing classifier among the evaluated methods. Additionally, the model demonstrates the ability to autonomously adapt to evolving workload patterns, reducing the need for manual intervention in database tuning.

2.2. Real-time Workload Pattern Analysis for Large-scale Cloud Databases

In the work of [Wang et al. 2023], the authors present a real-time system for discovering and analyzing workload patterns in large-scale cloud database environments, addressing the limitations of existing approaches in distinguishing multiple business logics embedded within complex query logs, which affects pattern identification and optimization accuracy. The proposed methodology is based on a modular architecture that combines foundation models for generating SQL semantic embeddings with Markov chain-based pattern mining, integrating both offline training and online inference components. Additionally, the system employs the Minimum Description Length (MDL) principle for model selection and constructs dependency graphs to support parallel execution recommendations. Experimental evaluation on both synthetic datasets and real-world workloads from Alibaba Cloud demonstrates improvements of up to 70% in pattern discovery accuracy compared to state-of-the-art methods, along with a 25% reduction in online inference latency. The system is designed for production-scale deployment and provides autonomous pattern mining capabilities while preserving user data privacy, enabling effective optimization guidance in high-demand environments.

2.3. Heuristic-Guided Text-to-SQL Translation with LLMs: Optimizing Natural Language Interfaces for Relational Databases

The work of [Petrola et al. 2025] employs large language models for Text-to-SQL translation with integrated query optimization through prompt engineering and embedding-based retrieval of semantically similar queries. Evaluated on TPC-H, Mondial, and a real-world database, the approach achieved substantial execution time reductions while improving semantic correctness. However, it remains focused on query-level

optimization, whereas incorporating workload analysis and semantic clustering could enable broader workload-aware tuning strategies.

3. Expeditus

This section summarizes the experimental design used to evaluate structural clustering of SQL queries and its impact on index-based optimization. The method has four stages: workload definition, query embedding generation, dimensionality reduction, and density-based clustering. As shown in the Figure 1, then assess how optimizations applied within each cluster affect overall query performance.

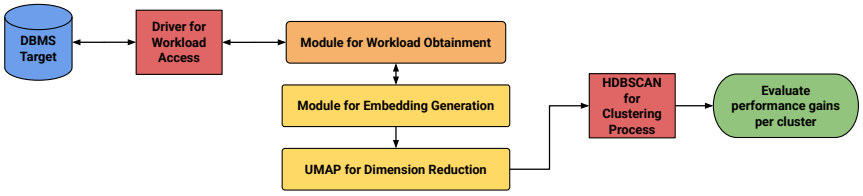

Figure 1. Flowchart representing a macro-level representation of the Expeditus framework.

3.1. Workload Capture

In the first stage, the query workload is captured directly from the DBMS by extracting a time-bounded slice of the workload log, providing a representative sample of the system's query activity. This workload snapshot preserves the diversity of query patterns, enabling the analysis of query structures, access paths, predicate usage, and recurring execution behaviors, while serving as the basis for subsequent workload characterization and optimization.

3.2. Vector Representation of Queries

The next stage of the approach consists of transforming SQL queries into dense vector representations (embeddings). This step aims to map SQL textual structures into a continuous vector space in which geometric proximity reflects structural similarity.

To this end, the model "Qwen3-Embedding-0.6B,"[1] available on the Hugging Face platform and part of the Alibaba Cloud Qwen family, was selected. The choice of this model was based on its reported performance in text vectorization, classification, and clustering tasks, as well as its ability to capture structural relationships in technical texts. These qualities can be verified on the Hugging Face platform itself, particularly through its embedding model leaderboard.

Each SQL query was provided to the model as a complete textual input, generating high-dimensional embedding vectors. These vectors constitute the foundational representation for the subsequent stages of dimensionality reduction and clustering.

It is important to note that no prior syntactic normalization procedures were applied, such as automatic clause reordering or identifier abstraction, nor were any SQL

[1]https://huggingface.co/Qwen/Qwen3-Embedding-0.6B

statement hierarchization procedures performed. This methodological decision aimed to evaluate the model's ability to capture similarities even in the presence of structural and lexical variations.

3.3. Dimensionality Reduction Module

In the third stage, Uniform Manifold Approximation and Projection (UMAP) is applied to reduce the dimensionality of the query embeddings. By projecting high-dimensional vectors into a lower-dimensional latent space while preserving local and global relationships, UMAP facilitates visualization and improves the effectiveness of subsequent clustering.

3.4. Density-Based Clustering Module

After obtaining the reduced vector representations, the HDBSCAN (Hierarchical Density-Based Spatial Clustering of Applications with Noise) algorithm was employed to cluster the queries. HDBSCAN is a density-based clustering method that builds a hierarchy of clusters and automatically extracts the most stable ones [McInnes et al. 2017]. It was selected because it does not require a predefined number of clusters, is robust to noise, and is capable of identifying clusters with varying densities and arbitrary shapes.

Unlike partitioning algorithms such as K-Means, which assume approximately spherical cluster structures, HDBSCAN constructs a hierarchy of clusters based on density and automatically extracts a stable partition of the space. This property is particularly relevant when the structural distribution of queries is not known in advance [McInnes et al. 2017].

3.5. Selecting the Best Index for the Cluster.

The final stage of the methodology focuses on selecting a representative index configuration for each structural cluster based on its performance across the cluster workload. For each cluster, candidate indexes (originally derived from individual query optimization) are evaluated against all queries within the cluster. Performance is assessed using metrics such as query execution time, I/O cost, and overall throughput under controlled execution conditions. This evaluation is conducted in a comparative manner, considering both baseline (non-indexed or default configuration) and indexed scenarios. The objective is to identify index configurations that yield consistent performance improvements across multiple queries, thereby maximizing cluster-level efficiency rather than isolated query optimization.

This approach is grounded in the assumption that proximity in the embedding space reflects structural similarity at the SQL level. Queries grouped within the same cluster tend to exhibit overlapping access patterns, including shared join graphs, similar predicate structures, and recurring attribute usage in filtering, grouping, or ordering operations. Under these conditions, an index designed to optimize a specific query may also reduce the cost of evaluating other queries within the same cluster. This perspective is consistent with workload-driven index selection strategies, where optimization decisions are guided by recurring patterns observed in the workload rather than single-query heuristics [Koßmann 2023].

To operationalize this evaluation, the following procedure is adopted:

1. Select a query (q_i) from a given cluster (C).
2. Extract candidate indexing attributes based on its execution plan (e.g., attributes involved in join predicates, selection filters, and grouping clauses).
3. Materialize the corresponding index configuration. For the TPC-H benchmark, predefined indexes from the official repository are also considered.
4. Measure the performance of (q_i) before and after index creation, ensuring consistent execution conditions.
5. Apply the same index configuration to all other queries $(q_j \in C)$, recording their performance under identical conditions.

The analysis distinguishes between localized improvements (restricted to the query used to derive the index) and cluster-wide effects, where the same index contributes to performance gains across multiple queries. This distinction supports the selection of index configurations that are more robust and representative of the cluster's workload characteristics.

4. Experimental Results

This section presents the results of the experiment, including the distribution of queries across clusters and the analysis of the impact of optimizations, both at the individual query level and at the aggregated group level.

4.1. Experimental Pipeline

The proposed pipeline integrates natural language processing techniques with classical data mining methods applied to the database context, establishing a bridge between structural textual representation and the structural analysis of SQL queries. The complete experimental workflow is illustrated in Figure 2, encompassing the stages of vectorization, dimensionality reduction, and clustering.

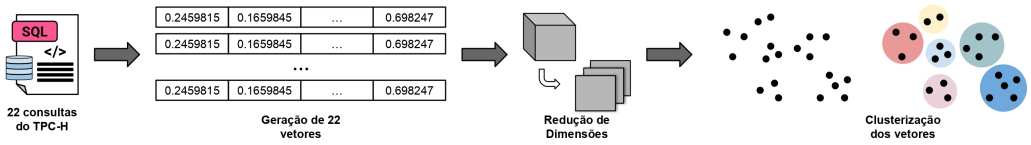

Figure 2. Experimental pipeline with the stages of vectorization, dimensionality reduction, and clustering.

The process was conducted as follows: embeddings for the 22 queries of the TPC-H benchmark were generated using the Qwen-Embedding-0.6B model in the Google Colab environment, a cloud-based platform developed by Google that enables the execution of Python code with support for GPUs and machine learning libraries. The query vectorization stage had an average duration of approximately 40 minutes.

After generating the embeddings, the subsequent stages of the experiment were executed on a local machine. The tests were conducted on a notebook equipped with an AMD Ryzen 5 4600H processor and 8 GB of DDR4 RAM. The TPC-H benchmark was configured and executed in a containerized environment using Docker, aiming to ensure isolation, reproducibility, and standardization of query execution. A total of 20 experimental runs were performed, resulting in an overall execution time of approximately 1 hour and 20 minutes to complete the entire experimental process.

4.2. Workload Experimental

As an experimental basis, the 22 SQL queries from the Transaction Processing Performance Council (TPC) benchmark were used, specifically from TPC-H barata2015overview. TPC-H is widely adopted for evaluating the performance of decision support systems (OLAP), consisting of a standardized relational schema and a fixed set of queries that simulate complex analytical scenarios. The choice of this benchmark was motivated by the large number of studies that employ it in their experiments.

The 22 TPC-H queries encompass different access patterns, including multiple join operations, aggregations, correlated subqueries, selective filters, and sorting operations. This structural diversity makes the benchmark suitable for evaluating query clustering techniques, as it enables the observation of both evident similarities and subtle variations in query logic.

Each query was considered in its canonical form, as specified in the official benchmark, without initial structural modifications. The focus of the investigation is not on parametric variability, but rather on the structural similarity among complete SQL statements.

4.3. Dimensionality Reduction

The generated embedding vectors have high dimensionality, which may introduce challenges associated with the "curse of dimensionality" and increased computational cost in distance-based algorithms [BRASIL et al. 2025].

To mitigate these effects, the UMAP (Uniform Manifold Approximation and Projection) algorithm was applied. UMAP is a nonlinear dimensionality reduction technique that constructs a neighborhood graph in the original space and projects the data into a lower-dimensional space while preserving, as much as possible, the proximity relationships among samples. Figure 3 presents the two-dimensional visualization of the clusters obtained after projection using UMAP.

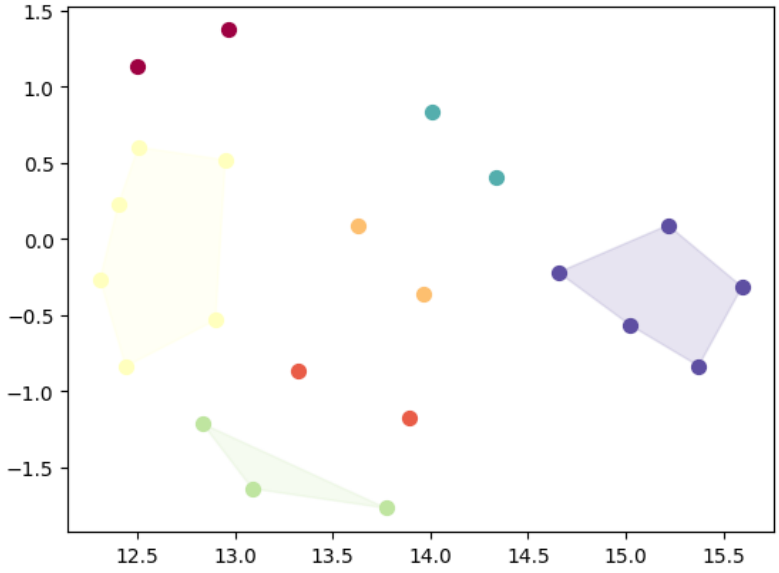

Figure 3. Two-dimensional visualization of clusters after projection via UMAP.

Unlike strictly linear approaches, UMAP tends to better represent complex structures and nonlinear separations, which is particularly important for SQL queries with heterogeneous structural patterns [McInnes et al. 2018]. In addition, its behavior can be adjusted through hyperparameters such as `n_neighbors` and `min_dist`, which control the balance between global structure preservation and local cluster compactness. UMAP was employed for two distinct purposes:

1. Dimensionality reduction to support the clustering stage with greater geometric stability.
2. Two-dimensional projection for visualization and exploratory analysis of the clusters.

In comparison to other well-known methods for this task, such as PCA (which reduces dimensionality through linear projections that maximize variance [Abdi and Williams 2010]) and t-SNE (which projects data by prioritizing the preservation of local neighborhoods [Van der Maaten and Hinton 2008]), UMAP provides a better balance between global structure preservation, neighborhood coherence, and computational cost, especially in datasets with nonlinear separations [McInnes et al. 2018].

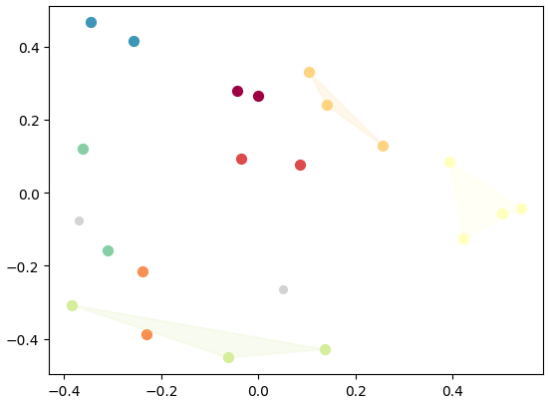

Figure 4. Projection of embeddings with PCA and visualization of clusters.

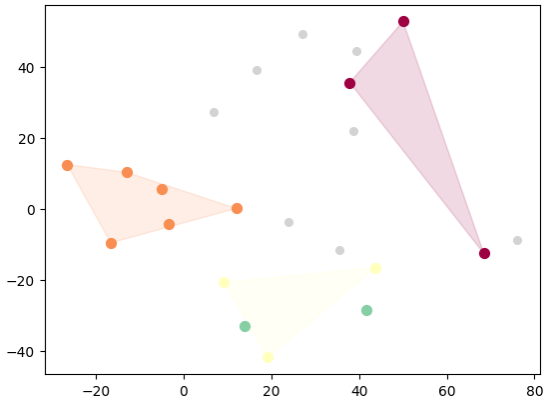

Figure 5. Projection of embeddings with t-SNE and visualization of clusters.

4.4. Clustering of TPC-H queries

The application of HDBSCAN resulted in the identification of seven distinct clusters among the 22 TPC-H queries. Queries classified as noise were analyzed separately to determine whether they represented isolated patterns or structurally atypical behaviors. Table 1 presents the distribution of queries across the clusters generated by the model. In total, seven distinct clusters were identified. It can be observed that Cluster 3 stands out as the largest group, containing six queries (q1, q3, q4, q6, q12, and q14), indicating greater structural similarity among these queries.

Clusters 0, 1, 2, and 5 each contained two queries, indicating more specific and potentially more homogeneous groupings. Cluster 4 grouped three queries (q13, q18, and q21), while Cluster 6 contained five queries (q2, q9, q11, q16, and q20), representing the second largest identified group.

Table 1. Distribution of queries by cluster.

Cluster	Queries
0	q17, q19
1	q10, q22
2	q5, q15
3	q1, q3, q4, q6, q12, q14
4	q13, q18, q21
5	q7, q8
6	q2, q9, q11, q16, q20

A relevant observation is the absence of the "-1" label, which is typically associated with noise or outliers in HDBSCAN. This indicates that all queries were considered sufficiently similar to at least one group, with no elements classified as diffuse or isolated in the analyzed vector space, even though the TPC-H queries are considerably distinct, as they are designed to evaluate database systems from different perspectives.

In Figure 6, a structural analysis of the query structure within the cluster formed by queries q17 and q19 reveals alignment in both analytical objectives and structural patterns. Both queries perform revenue aggregations based on the `lineitem` table, using expressions involving `l_extendedprice` and `l_discount`, as well as applying filters on product and shipping attributes.

The join with the `part` table, combined with conditions on attributes such as `p_brand`, `p_container`, `l_quantity`, `l_shipmode`, and `l_shipinstruct`, further reinforces that both queries follow the same commercial analysis pattern based on specific product characteristics.

4.5. General Analysis per SQL Query

Table 2 presents a summary of the results obtained after applying the strategy proposed in this study to assess the impact of optimizations on the clusters. When analyzing the queries individually, it can be observed that the proposed approach yielded positive results for the majority of queries, although performance degradation was observed in a subset of them.

The analysis of the 22 queries highlights several important points, such as:

- Expressive reduction in execution time for query Q17 (**-99.97%**), which showed the greatest savings.
- Significant improvements in queries Q20 (**-73.73%**), Q2 (**-70.58%**), Q19 (**-69.19%**), Q7 (**-66.73%**), Q16 (**-63.59%**), and Q6 (**-61.41%**). All demonstrated time reductions greater than 60%.
- However, some queries experienced performance degradation, such as Q8 (**+83.46%**), Q5 (**+49.06%**), and Q4 (**+41.57%**).

4.6. Cluster-Based Analysis

Table 3 presents the results by cluster in the proposed experiment. We can evaluate each group as follows:

```
select
  sum(l_extendedprice * (1 - l_discount)) as revenue
from
  lineitem,
  part
where (
    p_partkey = l_partkey
    and p_brand = ':1'
    and p_container in ('SM CASE', 'SM BOX', 'SM PACK', 'SM PKG')
    and l_quantity >= :4 and l_quantity <= :4 + 10
    and p_size between 1 and 5
    and l_shipmode in ('AIR', 'AIR REG')
    and l_shipinstruct = 'DELIVER IN PERSON')
  or (
    p_partkey = l_partkey
    and p_brand = ':2'
    and p_container in ('MED BAG', 'MED BOX', 'MED PKG', 'MED PACK')
    and l_quantity >= :5 and l_quantity <= :5 + 10
    and p_size between 1 and 10
    and l_shipmode in ('AIR', 'AIR REG')
    and l_shipinstruct = 'DELIVER IN PERSON' )
  or (
    p_partkey = l_partkey
    and p_brand = ':3'
    and p_container in ('LG CASE', 'LG BOX', 'LG PACK', 'LG PKG')
    and l_quantity >= :6 and l_quantity <= :6 + 10
    and p_size between 1 and 15
    and l_shipmode in ('AIR', 'AIR REG')
    and l_shipinstruct = 'DELIVER IN PERSON' );

```

Query q19.

```
select
  sum(l_extendedprice) / 7.0 as avg_yearly
from
  lineitem,
  part
where
  p_partkey = l_partkey
  and p_brand = ':1'
  and p_container = ':2'
  and l_quantity < (
    select
      0.2 * avg(l_quantity)
    from
      lineitem
    where
      l_partkey = p_partkey
  );

```

Query q17.

Figure 6. Structural analysis of cluster 0.

- **Cluster 0**: average gain of 87.64%, representing the highest mean performance among all clusters. Query Q17 achieved the greatest absolute improvement; however, it also exhibited high percentage variation, indicating that it is sensitive to the applied index.
- **Cluster 1**: average gain of 20.17%, showing moderate improvement across the queries within the group.
- **Cluster 2**: negative average gain, indicating that the applied indexes were not suitable for the queries in this cluster.
- **Clusters 3, 4, 5, and 6**: presented positive averages ranging from 5.90% to

Table 2. Query-level analysis: average execution time and variation after optimization.

Query	Raw Query Average	Optimized Query Average	Variation (%)
q17	94.75	0.03	-99.97
q20	1.37	0.36	-73.73
q2	0.19	0.05	-70.58
q19	0.15	0.04	-69.19
q7	1.12	0.37	-66.73
q16	1.02	0.37	-63.59
q6	0.54	0.21	-61.41
q3	0.56	0.31	-43.94
q9	1.26	0.87	-31.24
q15	1.03	0.77	-25.09
q18	2.93	2.25	-23.08
q22	0.13	0.10	-18.53
q1	2.21	1.90	-13.89
q13	0.81	0.78	-3.72
q12	0.54	0.54	0.98
q14	0.26	0.26	1.43
q10	0.55	0.57	4.01
q21	1.80	1.87	4.03
q11	0.13	0.15	15.73
q4	0.15	0.22	41.57
q5	0.40	0.60	49.06
q8	0.26	0.47	83.46

23.58%, demonstrating a favorable impact of the applied strategy.

Table 3. Cluster-level results: best index (source) and average gain. .

Cluster ID	Best Index (Source)	Average Gain (%)
0	19	84.64
1	22	20.17
2	5	-14.78
3	14	23.58
4	18	16.40
5	8	5.90
6	20	21.20

4.7. Discussion of Results

The results largely confirm the central premise of this study: structurally similar queries tend to be grouped in such a way that an optimization applied to one query may positively impact the remaining queries within the same cluster. However, the findings also indicate that not all groupings lead to collective benefits, as observed in the case of Cluster 2, where the applied indexing strategies did not consistently improve performance

across all queries. In some cases, the creation of an index for a particular query negatively affected the execution performance of other queries within the same cluster. Furthermore, certain queries may not benefit from index-based optimization at all, thus requiring alternative fine-tuning strategies, such as query rewriting, materialized views, or execution plan adjustments.

In particular, query q5 exhibited performance degradation after the creation of its corresponding index structure. Nevertheless, from a cluster-level optimization perspective, the index derived from q5 still produced better aggregate performance results for Cluster 2 than the index configuration derived from q15. These findings suggest that vector-based similarity representations are capable of capturing relevant structural relationships among SQL queries; however, they do not guarantee uniform behavior with respect to indexing strategies and physical execution characteristics.

5. Conclusions and Future Work

This study investigated how the creation of an index can impact other structurally similar queries in the vector space, based on the premise that clustered queries exhibit potential structural similarity in their SQL formulation. By employing embedding models, dimensionality reduction techniques, and clustering algorithms, it was possible to conduct experiments to evaluate this hypothesis.

Overall, most queries exhibited improved or stable execution times after index creation, although some regressions indicate that structural similarity alone does not guarantee uniform indexing behavior. Nevertheless, the positive average gains observed across most clusters suggest that similarity-driven optimization is a viable strategy for workload-aware SQL tuning.

As future work, we intend to extend the empirical validation to more heterogeneous workloads, such as those from the TPC-DS benchmark, as well as to investigate in greater depth the structural characteristics of queries in order to define more adaptive, cluster-specific indexing policies. Additionally, we plan to incorporate parsing techniques and structural enrichment of query representations during the vectorization stage, with the aim of improving clustering quality and enhancing the relevance of the attributes used to measure similarity.

6. Acknowledgements

We gratefully acknowledge the Kunumi Institute for funding this research and for providing the computational infrastructure and support for the presentation of this work.

7. Statement on the Use of Generative AI

We do not anticipate any immediate social or ethical concerns arising from this work. Additionally, we acknowledge the use of LLMs to assist with grammar checking in parts of the manuscript.

References

Abdi, H. and Williams, L. J. (2010). Principal component analysis. *Wiley interdisciplinary reviews: computational statistics*, 2(4):433–459.

Almeida, J. A. O. d. S. and Moura, R. S. (2024). Investigação de métodos de similaridade textual no contexto da avaliação automática de questões discursivas. In *Escola Regional de Computação do Ceará, Maranhão e Piauí (ERCEMAPI)*, pages 110–118. SBC.

BRASIL, R. M., de Freitas BRITO, D., Homero, J., et al. (2025). Fundamentos e aplicação de redes neurais artificiais à saúde e humanidades: Tutorial–parte 1. *Revista Presença*, 11(26):405–419.

Cer, D., Yang, Y., Kong, S.-y., Hua, N., Limtiaco, N., John, R. S., Constant, N., Guajardo-Cespedes, M., Yuan, S., Tar, C., et al. (2018). Universal sentence encoder. *arXiv preprint arXiv:1803.11175*.

do Amaral, K. P. N., de Andrade, L. P., and de Campos, E. A. V. (2025). O uso da inteligência artificial na otimização de consultas em bancos de dados. *Revista Camalotes*, 4(1).

Guabtni, A., Ranjan, R., and Rabhi, F. A. (2013). A workload-driven approach to database query processing in the cloud. *The Journal of Supercomputing*, 63(3):722–736.

Koßmann, J. (2023). *Unsupervised database optimization: efficient index selection & data dependency-driven query optimization*. PhD thesis, Universität Potsdam.

McInnes, L., Healy, J., Astels, S., et al. (2017). hdbscan: Hierarchical density based clustering. *J. Open Source Softw.*, 2(11):205.

McInnes, L., Healy, J., and Melville, J. (2018). Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*.

Oloruntoba, N. (2025). Ai-driven autonomous database management: Self-tuning, predictive query optimization, and intelligent indexing in enterprise it environments. *World Journal of Advanced Research and Reviews*, 25(2):1558–1580.

Petrola, L., Brayner, A., and Franco, W. (2025). Heuristic-guided text-to-sql translation with llms: Optimizing natural language interfaces for relational databases. In *Simpósio Brasileiro de Banco de Dados (SBBD)*, pages 126–139. SBC.

Shaheen, N., Raza, B., and Malik, A. K. (2018). A cbr model for workload characterization in autonomic database management system. In *2018 14th international conference on emerging technologies (ICET)*, pages 1–6. IEEE.

Van der Maaten, L. and Hinton, G. (2008). Visualizing data using t-sne. *Journal of machine learning research*, 9(11).

Wang, J., Li, T., Wang, A., Liu, X., Chen, L., Chen, J., Liu, J., Wu, J., Li, F., and Gao, Y. (2023). Real-time workload pattern analysis for large-scale cloud databases. *arXiv preprint arXiv:2307.02626*.