Bol. Soc. Paran. Mat.
©SPM – E-ISSN-2175-1188
SPM: www.spm.uem.br/bspm

(3s.) **v. 2026 (44)** : 1–9.
ISSN-0037-8712
doi:10.5269/bspm.81609

SQL Optimization for Fast Retrieval of Transportation Route Data in TMS

A. Sandhya and G.Y. Mythili

ABSTRACT: Transportation Management Systems (TMS) need regular and effective access to transportation data such as route cost, travel distance and CO_2 emissions to support route planning, freight pricing, dispatch scheduling and real time logistics operations. As transportation databases expand in size and query frequency increases, SQL query performance becomes a critical factor affecting overall system responsiveness. This study presents a standardized relational schema intended for query intensive TMS workloads and examines how SQL optimization techniques affect the efficiency of origin destination (OD) based transportation data retrieval. The proposed schema is assessed using an extensive synthetic transportation dataset through typical TMS style queries, including OD cost lookup, least cost route determination from a specified origin and emission restricted route filtering. Query performance is evaluated under baseline and indexed configurations using execution time measurements and analysis of query execution plans. The findings provide insights into how schema design and indexing strategies influence SQL execution efficiency for high frequency OD queries, providing pragmatic guidelines for the development of scalable and performance centric transportation databases.

Keywords: Transport management systems, logistics, data retrieval, SQL, relational schema, transportation.

Contents

1. Introduction

Transportation Management Systems (TMS) constitute a foundational component of modern logistics and supply chain infrastructures, enabling organizations to plan routes, determine freight costs, schedule vehicle dispatches, and monitor transportation performance. With the rapid expansion of e-commerce, globalized supply chains, and intelligent transportation technologies, TMS platforms are increasingly required to handle large volumes of heterogeneous transportation data while supporting operational decisions under strict time constraints. In practical deployments, even small delays in data retrieval can

2020 *Mathematics Subject Classification:* 68P15, 90B06, 68P20, 90C27.
Submitted January 28, 2026. Published April 17, 2026.

directly affect pricing accuracy, dispatch efficiency, and overall service quality, making system responsiveness a critical requirement for contemporary transportation platforms [1, 2].

At the core of most TMS implementations lies a relational database management system (RDBMS), which is used to store and manage structured transportation data such as locations, routes, distances, costs, travel times, and environmental indicators. Relational databases remain popular in this domain due to their maturity, robustness, transactional reliability, and expressive SQL based query capabilities [3]. The relational model naturally supports origin destination (OD) representations, where each transportation route is defined by a specific origin, destination, and a set of associated attributes. However, while the logical representation of transportation data is straightforward, achieving efficient query execution at scale remains a significant challenge.

In operational TMS environments, OD based queries dominate system workloads. Typical examples include retrieving the transportation cost between a given origin and destination, identifying the least cost route from a particular source location, or filtering routes that satisfy emission or distance constraints. These queries are executed repeatedly during pricing, routing, and sustainability aware decision making processes. As transportation datasets grow in size and query frequency increases, inefficient database designs can lead to full table scans, suboptimal execution plans, and excessive response times. Such performance issues directly impact the ability of TMS platforms to support real time operations [4, 5].

Although transportation research has made substantial progress in routing algorithms, vehicle scheduling, traffic prediction and data driven logistics optimization, much of this literature implicitly assumes that route related data can be accessed efficiently from the underlying storage layer [1, 6]. In real world systems, however, algorithmic efficiency alone is insufficient. The performance of optimization and analytics components is tightly coupled with the efficiency of database operations that supply the required data. Poor SQL execution performance can negate algorithmic improvements and become a practical bottleneck in large scale TMS deployments [7].

Database systems research has repeatedly demonstrated that generic, one size fits all relational schemas are rarely optimal for query intensive applications. Instead, workload aware schema design and index selection play a decisive role in reducing query execution time by aligning physical data structures with dominant access patterns [8, 9]. Despite this understanding, many TMS platforms continue to employ non specialized schemas that are not explicitly designed for OD centric query workloads, often relying on default indexing strategies without systematic performance evaluation.

Despite parallel advances in transportation analytics and database optimization, limited attention has been devoted to database level performance in Transportation Management Systems. In particular, there is a lack of focused studies that systematically evaluate how relational schema design and SQL optimization strategies influence the execution efficiency of high frequency OD queries under realistic TMS workloads. Motivated by this gap, the present study investigates SQL performance optimization for OD based transportation data. A standardized OD oriented relational schema is proposed and evaluated using large scale synthetic transportation datasets that reflect practical TMS scenarios. The study examines the impact of indexing strategies and SQL execution plans through detailed execution time measurements and query plan analysis, with the aim of providing practical and transferable guidelines for designing efficient transportation databases.

2. Related Work

Research related to Transportation Management Systems has traditionally emphasized algorithmic and operational aspects of transportation planning. Numerous studies have addressed routing, scheduling and optimization problems with the objective of minimizing cost, travel time or environmental impact. Recent surveys highlight the growing reliance on large scale transportation data and advanced analytics to support decision making in logistics and intelligent transportation systems [1, 2]. While these contributions are essential for improving transportation efficiency, they primarily focus on modeling and algorithmic layers, often abstracting away the data management mechanisms that support frequent route queries.

With the increasing scale of transportation networks and the adoption of real time decision support systems, several studies have recognized data management as a critical component of intelligent transportation systems. Research in this area emphasizes challenges related to data volume, velocity,

and latency, particularly in applications that require continuous access to updated transportation information [3, 10]. However, in much of this literature, databases are treated as passive repositories, and limited attention is paid to how relational schema design or SQL execution behavior affects system level performance.

From a database systems perspective, query optimization and indexing have long been recognized as key determinants of performance in relational systems. Recent surveys provide comprehensive overviews of modern query optimization techniques, including cost based optimization, adaptive query processing, and workload aware tuning mechanisms [8, 11]. These studies consistently demonstrate that performance improvements are achieved when database design decisions are informed by actual query workloads rather than generic assumptions.

Workload aware schema design has gained particular attention in recent years. Empirical studies show that schemas tailored to dominant access patterns significantly outperform generic designs, especially in environments characterized by high query repetition and selective predicates [9, 12]. Indexing strategies such as composite and covering indexes have also been shown to reduce I/O overhead and eliminate expensive sorting operations when index definitions closely match query predicates and ordering requirements [13]. These findings are directly relevant to TMS workloads, where OD-based queries frequently involve equality and range conditions on multiple attributes.

In transportation applications, query workloads have become increasingly complex due to the integration of sustainability considerations. Emission aware routing and freight optimization models incorporate environmental indicators such as CO_2 emissions and energy consumption, increasing both the dimensionality of transportation data and the complexity of query predicates [14]. Efficiently supporting such multi-attribute OD queries places additional demands on relational schema design and index configuration.

More recently, a limited number of studies have begun to examine database performance optimization in transportation and mobility analytics. These works explore techniques such as composite indexing, data partitioning, and query plan tuning to improve response times in data intensive transportation systems [15, 16]. While these contributions provide valuable insights, comprehensive and reproducible evaluations focused specifically on OD oriented relational schemas for high-frequency TMS query workloads remain scarce.

In summary, existing literature offers strong foundations in transportation analytics and general purpose database optimization. However, the intersection of these domains particularly the systematic evaluation of SQL performance on OD based relational schemas under realistic TMS workloads remains underexplored. The present study addresses this gap by focusing on schema design and indexing strategies tailored to the operational realities of Transportation Management Systems.

3. Proposed OD Based Relational Schema

Transportation Management Systems require efficient and scalable data structures to support frequent access to route level information for operational and analytical tasks. In practice, many existing systems store transportation data using heterogeneous or loosely structured schemas, which can hinder SQL query performance as data volume and access frequency increase. To address this issue, this study introduces a standardized origin destination (OD) oriented relational schema tailored for query intensive TMS environments.

The proposed schema consists of two fundamental relational entities: *Locations* and *Routes*, each serving a clearly defined role within the database design. This separation enables a clean representation of transportation nodes and route specific attributes while maintaining structural clarity and extensibility. The *Locations* table captures static information associated with transportation nodes, such as cities, warehouses, or distribution centers. Each node is uniquely identified by a primary key (`location_id`) and is linked to basic geographic descriptors, including latitude and longitude. By isolating node level information in a dedicated table, the schema supports normalization and reduces redundancy, particularly in large scale transportation networks where the same locations participate in multiple routes.

The *Routes* table represents directed transportation connections between origin and destination nodes. Each record corresponds to a single OD pair and includes attributes required for operational decision making, such as route distance, transportation cost, estimated travel time, and CO_2 emissions. The

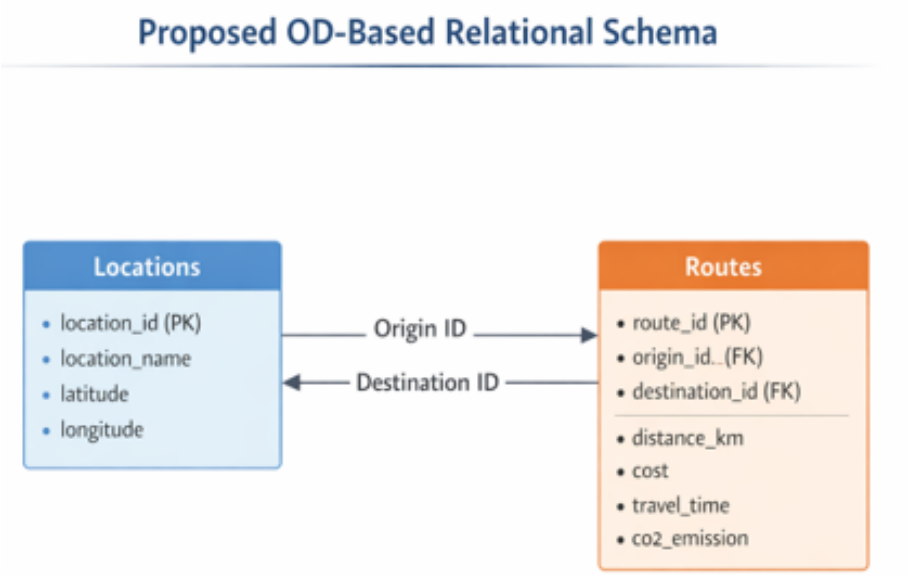

Figure 1: Proposed origin destination (OD) based relational schema for Transportation Management Systems

origin and destination identifiers are implemented as foreign keys referencing the *Locations* table, ensuring referential consistency across the schema.

A deliberate design choice of the proposed schema is the aggregation of all route related attributes within a single table. This design minimizes the need for join operations during query execution, which is advantageous for high frequency SQL queries commonly executed in TMS platforms. Queries such as OD based cost retrieval, least cost route identification from a given origin, and emission constrained route selection can therefore be evaluated directly on the *Routes* table with predictable execution behavior.

From an optimization perspective, the schema is explicitly designed to accommodate common indexing techniques. Composite indexes on origin destination attributes support efficient equality based lookups, while multi column indexes involving cost or distance facilitate index assisted ordering for route selection queries. In addition, single attribute indexes on emission values enable efficient evaluation of range predicates. These indexing strategies can be applied without altering the logical schema, providing flexibility for performance tuning. Overall, the proposed OD based relational schema offers a compact, scalable, and index friendly foundation for transportation databases. Its simplicity supports efficient SQL query execution while remaining adaptable to both real time operational workloads and offline analytical processing in modern TMS deployments.

4. Experimental Setup

To assess the efficacy of SQL optimization strategies for transportation data retrieval, a series of controlled experiments were carried out using a synthetic dataset that represents typical Transportation Management System workloads. Because of the experimental design's emphasis on realism and repeatability, query performance under various optimization configurations can be systematically assessed.

4.1. Database Environment

MySQL 8 was used as the relational database management system for all experiments. The database was set up on a standard desktop computing environment with adequate processing power and memory to manage large scale query execution. MySQL Workbench was used to execute queries and manage database objects. Query execution behaviour was examined using the `EXPLAIN ANALYZE` command, which provides detailed information on execution plans, runtime statistics and operator level costs.

4.2. Dataset Description

The experimental dataset was created synthetically based on the proposed OD based schema to ensure control over data properties and scalability. The dataset comprises two tables: *Locations* and *Routes*. The *Locations* table contains 100 distinct transportation nodes representing cities or logistical centres. The *Routes* table contains 100,000 directed OD records, each corresponding to a transportation link between two locations.

Realistic attribute values were generated for each route to emulate real world transportation scenarios. Travel distances were sampled within a plausible range, and corresponding cost, travel time and CO_2 emission values were calculated proportionally from distance. SQL scripts were used to generate the dataset automatically, ensuring consistency and eliminating bias associated with manual data entry.

4.3. Query Workload Design

To evaluate SQL performance under representative TMS scenarios, a query workload was designed based on typical operational requirements observed in logistics platforms. The selected queries reflect high frequency access patterns related to pricing, routing, and sustainability analysis.

The first query retrieves the transportation cost for a given origin destination pair using an OD based cost lookup. As this query relies on equality predicates, it is frequently executed during pricing and dispatch processes and is highly sensitive to index availability.

The second query focuses on least cost route selection from a specified origin.It retrieves candidate routes associated with the origin, orders them by transportation cost, and returns the minimum cost option. From a database perspective, this query combines filtering and ordering operations, making it well suited for evaluating the benefits of composite indexing and index assisted sorting.

The third query addresses emission constrained route selection by filtering routes that satisfy a predefined CO_2 emission threshold and ordering them by travel distance. This query combines range based filtering and sorting operations to assess the effectiveness of single column indexing and represents sustainability driven decision making in modern TMS platforms.

Together, these queries form a balanced workload that captures a range of SQL access patterns, including equality conditions, range predicates, and sorting intensive operations, enabling a comprehensive assessment of query optimization techniques.

4.4. SQL Optimization Strategies Evaluated

Query performance was evaluated under both baseline and optimized database configurations. To establish a baseline representative of unoptimized database deployments, all queries were initially executed without secondary indexes. This configuration serves as a benchmark for measuring performance improvements. Subsequently, targeted indexing strategies were implemented based on query access characteristics. A composite index on origin and destination attributes was introduced to optimize OD based cost lookups. A multi column index on origin and cost was created for least cost route selection queries to facilitate efficient filtering and index supported ordering. In addition, a single column index on CO_2 emission values was applied to accelerate range based filtering in emission constrained queries.

To ensure that observed performance improvements could be attributed solely to the implemented optimizations, each indexing strategy was evaluated independently by re-executing the same query workload after index creation.

4.5. Performance Evaluation Metrics

Multiple complementary metrics were used to assess query performance. Query execution time, measured in seconds, was the primary metric, as it directly reflects database responsiveness in practical TMS environments. In addition, the number of rows examined during query execution was analysed using `EXPLAIN ANALYZE` output. This metric provides insight into query selectivity and the effectiveness of index utilization, with reductions in examined rows indicating improved access efficiency and reduced unnecessary data processing.

Finally, query execution plans were analysed to identify changes in execution strategies, including transitions from full table scans to index lookups or index range scans, as well as the elimination of costly operations such as `filesort`. By combining runtime measurements with execution plan analysis,

this evaluation approach provides a comprehensive understanding of how SQL optimization techniques influence performance at both logical and physical execution levels.

5. Results and Discussion

This section discusses the results obtained from a series of experiments carried out to evaluate the effectiveness of SQL optimization strategies on the proposed OD based relational schema. The focus is on how query execution behaviour changes when moving from a baseline database configuration to an indexed setup for typical Transportation Management System (TMS) queries. All experiments were performed using MySQL 8, and performance was assessed primarily through measured execution times. In addition, query execution plans were analysed to understand how indexing influenced the internal behaviour of the database engine.

5.1. Baseline Query Performance (Without Specialized Indexing)

The initial set of experiments examined query performance under a baseline configuration in which only the primary key and required foreign key related indexes were present. Under this configuration, queries relied predominantly on full table scans or weakly selective access paths, leading to inefficient execution behaviour.

In the case of the least cost route selection query, which identifies the minimum transportation cost for a given origin, the execution plan showed that MySQL scanned all routes associated with that origin and then performed an explicit sorting operation to satisfy the `ORDER BY cost_usd` clause. Although the query returned only a single row due to the `LIMIT 1` constraint, the database engine still processed a large number of rows internally. This behaviour reflects a common inefficiency in unoptimized TMS databases, where repeated sorting operations introduce avoidable computational overhead.

A similar pattern was observed for origin destination (OD) cost lookup queries. In the absence of a dedicated composite index, the database engine was required to evaluate multiple rows before identifying the matching record. While this behaviour may be acceptable for small datasets, it becomes increasingly inefficient as the number of routes grows, particularly when queries are executed concurrently.

Emission constrained route selection queries exhibited the highest baseline costs. Filtering routes based on a CO_2 emission threshold combined with sorting operations resulted in large intermediate result sets, leading to increased execution times and higher estimated query costs. These observations confirm that baseline relational schemas are insufficient for supporting high frequency, multi attribute TMS queries efficiently.

Table 1: Impact of Indexing on Query Execution Performance

Query Type	Index Used	Execution Time	Rows Examined	Query Plan
Cost based OD lookup	No	Higher	Large	Table Scan
Cost based OD lookup	Yes	Lower	Few	Index Scan
OD pair lookup	No	Higher	Large	Table Scan
OD pair lookup	Yes	Lower	Minimal	Index Lookup
CO_2 based filtering	No	Higher	Large	File sort
CO_2 based filtering	Yes	Lower	Reduced	Index Range Scan

5.2. Impact of Composite Indexing on OD Based Queries

The introduction of a composite index on `(origin, cost_usd)` significantly improved the performance of least cost route selection queries. After index creation, execution plans revealed a clear transition from table scans with explicit sorting to index assisted access paths. The database engine was able to retrieve the minimum cost route directly from the index structure, thereby eliminating the need for a separate sorting phase.

This optimization resulted in a noticeable reduction in execution time and internal row processing. From a TMS perspective, this improvement is particularly important, as least cost route selection is a core operation in pricing, dispatching, and routing workflows. The results demonstrate that aligning composite indexes with both filtering and ordering attributes is an effective strategy for optimizing such queries.

It is noteworthy that attempts to recreate the same composite index were correctly rejected by the database system, indicating proper index reuse. This confirms that the observed performance gains are attributable to the intended index design rather than redundant or conflicting indexing.

5.3. OD Pair Lookup Performance with Origin Destination Indexing

For exact OD pair queries retrieving cost, distance, and emission attributes, the use of a composite index on `(origin, destination)` yielded highly efficient execution behaviour. Query execution plans consistently showed direct index lookups, with the database engine accessing only the required row.

Execution times for these queries were minimal and largely insensitive to table size, highlighting the scalability benefits of OD oriented indexing. Such performance characteristics are essential for real time TMS operations, where OD lookups are repeatedly executed during route planning and dispatch decision making.

The experiments also confirmed the importance of respecting foreign key constraints when managing indexes. Attempts to drop system required indexes associated with referential integrity were correctly rejected, ensuring data consistency while preserving optimized access paths.

5.4. Emission Constrained Query Optimization

Emission-based route selection queries represent an increasingly important class of Transportation Management System (TMS) workloads due to growing sustainability and regulatory requirements. Under the baseline configuration, these queries exhibited relatively high execution costs, as the database engine evaluated a large number of rows to satisfy range predicates on CO_2 emission values.

After introducing a single column index on CO_2, the execution plan shifted from full table scans to an index range scan, substantially reducing the number of rows examined. This change resulted in a marked improvement in execution time and eliminated unnecessary data processing. The results demonstrate that even simple indexing strategies can provide significant performance benefits for sustainability driven queries when appropriately aligned with query predicates.

These findings are particularly relevant for modern TMS platforms that incorporate environmental metrics into routing and pricing decisions, where efficient emission based filtering is essential for real time and regulation compliant operations.

5.5. Index Management and Experimental Robustness

Throughout the experiments, index creation, reuse, and removal were carefully controlled and verified using the `SHOW INDEX` command. Duplicate index creation attempts and foreign key related index constraints behaved as expected, reinforcing the correctness and reproducibility of the experimental setup.

The final index configuration retained only those indexes directly relevant to the evaluated queries, ensuring that performance results were not influenced by unintended or redundant index structures. This systematic approach to index management strengthens the validity of the observed performance improvements and supports fair comparison between baseline and optimized database configurations.

5.6. Discussion and Practical Implications

The experimental results clearly demonstrate that SQL performance in Transportation Management Systems is strongly influenced by relational schema design and index selection. Queries that dominate TMS workloads, including OD lookups, least cost route selection, and emission constrained filtering, benefited substantially from workload aware indexing strategies.

From a practical perspective, the findings indicate that many performance bottlenecks in real world TMS deployments can be mitigated without modifying application logic or routing algorithms. Instead, targeted database level optimizations can significantly enhance system responsiveness and scalability. This is particularly valuable in operational environments where changes to application code may be

costly, risky, or impractical. More broadly, the results underscore the importance of treating database optimization as a first class concern in transportation systems research. Algorithmic advancements alone are insufficient if underlying data access mechanisms remain inefficient. By explicitly addressing SQL execution behaviour, this study bridges a critical gap between transportation analytics and database systems engineering.

6. Conclusion

This study examined the role of SQL optimization in improving the efficiency of transportation route data retrieval in Transportation Management Systems. A standardized origin destination based relational schema was proposed to support query intensive TMS workloads, and its performance was evaluated using representative operational queries on a large synthetic dataset. The experimental results demonstrate that baseline database configurations are insufficient for handling high frequency OD queries efficiently as data volume increases.

The findings show that workload aware indexing strategies, particularly composite indexes aligned with filtering and ordering predicates, significantly reduce query execution time and internal data processing. Least cost route selection and OD pair lookups benefited most from composite indexing, while emission constrained queries showed substantial improvement through single column range indexing. Execution plan analysis confirmed that these gains were achieved by eliminating full table scans and computationally expensive sorting operations.

Overall, the results highlight that effective database schema design and index selection are critical enablers of real time decision making in modern TMS platforms. By focusing on SQL level optimization rather than application level changes, the proposed approach offers a practical and scalable solution for improving system responsiveness. Future work may extend this study by evaluating additional optimization techniques such as data partitioning and query rewriting, as well as validating the proposed schema on real world transportation datasets.

References

1. J. Zhang, F. Wang and K. Wang, *Big data analytics for intelligent transportation systems: A survey*, IEEE Transactions on Intelligent Transportation Systems (2021).
2. A. Gunasekaran, N. Subramanian and E. Ngai, *Information systems in supply chain and logistics management: A review*, International Journal of Production Economics (2020).
3. J. Wang, Y. Chen and Y. Zhou, *Data management for intelligent transportation systems: A survey*, IEEE Access (2022).
4. Y. Gul, R. Iqbal and F. K. Hussain, *A survey of database performance research focusing on indexing and query optimization*, Information Systems (2021).
5. Z. Li *et al.*, *Real-time freight analytics enabled by data-driven transportation systems*, IEEE Transactions on Intelligent Transportation Systems (2023).
6. K. Zhou, S. Yang and Z. Shao, *Edge intelligence for real-time transportation systems*, IEEE Communications Magazine (2021).
7. C. Chen *et al.*, *The promises of big data for travel behavior analysis*, Transportation Research Part C (2020).
8. S. Chaudhuri and R. Kaushik, *Query optimization in modern database systems*, ACM Computing Surveys (2020).
9. S. Suraj, S. Suresh and P. Jayanthi, *Workload-aware schema design for high-performance relational databases*, Journal of Systems and Software (2023).
10. J. Zhang *et al.*, *Scalable data infrastructures for intelligent transportation systems*, IEEE Access (2021).
11. T. Neumann and A. Kemper, *Database systems: Query processing and optimization revisited*, ACM Computing Surveys (2020).
12. V. Kadambi and H. Garcia-Molina, *Joint optimization of index design and query processing*, The VLDB Journal (2020).
13. L. Liu, Y. Zhang and M. Zhao, *Benchmarking SQL performance on composite index strategies*, Journal of Systems Architecture (2024).
14. X. Wang *et al.*, *Emission-aware routing optimization: A survey*, Transportation Research Part D (2021).
15. F. Xiao, L. Wang and H. Li, *Efficient processing of multi-attribute queries in spatial databases*, GeoInformatica (2022).
16. S. Zhu, H. Wang and L. Gao, *Multi-criteria route selection in freight transportation systems*, Computers & Industrial Engineering (2023).

Sandhya Abimannan,
Research Scholar,
Department of Mathematics,
Vellore Institute of Technology,
Chennai,
Tamil Nadu,
India
First Author.
E-mail address: sandhya.a2023@vitstudent.ac.in

and

Mythili G. Y.,
Assistant Professor Senior Grade 1,
Department of Mathematics,
Vellore Institute of Technology,
Chennai,
Tamil Nadu,
India
Corresponding Author.
E-mail address: mythili.gy@vit.ac.in