Published in AJSE, Vol: 25, Issue:1
Received on 5th April 2025
Revised on 16th August 2026
Published on 31st August 2026

AIUB JOURNAL OF SCIENCE AND ENGINEERING
ISSN: 1608 - 3679 (print) 2520 - 4890 (Online)
https://doi.org/10.53799/zb6b1375

A Controlled Evaluation of Relational Database Models for Query Performance across DAS, RAID-0, and RAID-1 Storage Configurations

Juanda Hakim Lubis, Ivan Jaya, Fajrul Malik Aminullah Napitupulu, and Herman Mawengkang

Abstract— **Relational database performance is shaped not only by query formulation and storage configuration, but also by the underlying database design. This study evaluates the impact of alternative relational database models and RAID storage configurations on query performance using a Cost-Based Optimizer and measured access time. Three relational database designs were examined across different data volumes and query workloads, including simple selection, joins, aggregation, correlated subqueries, and ordering operations. The analysis considers how schema decomposition, record count, attribute size, query type, primary and unique key usage, index sequence, SQL functions, and ORDER BY clauses influence optimizer-estimated cost and execution behavior. The findings show that relational designs that separate entities according to specialized usage can reduce query complexity and improve performance under several workloads. In addition, storage configuration contributes to performance variation, although its effects depend on the query structure and database model applied. Overall, the study demonstrates that query performance optimization requires an integrated view of database schema design, indexing considerations, optimizer behavior, and storage architecture. These results provide practical guidance for database designers and administrators seeking to improve relational database efficiency in transaction-oriented systems.**

Keywords— **Database Design, Query Optimization, Relational Database Model, Cost-Based Optimizer, Query Performance**

I. INTRODUCTION

Relational database systems remain fundamental to enterprise applications because they provide structured data management, declarative querying, transactional reliability, and mature optimization mechanisms. In such systems, performance is often discussed in terms of indexing, join algorithms, query plans, and cost-based optimization [1-5].

However, performance is also strongly influenced by the logical structure of the database itself. Different relational designs may represent the same application domain while producing different table decompositions, relationship paths, join requirements, and index-access opportunities. These design-level differences can directly affect query complexity, optimizer-estimated cost, and execution time [6-12].

Classical relational design principles emphasize normalization as a means of reducing redundancy, preventing update anomalies, and preserving data consistency [6]. Nevertheless, normalization is not always performance-neutral. Highly normalized schemas may require additional joins during retrieval, while selective decomposition or denormalization can simplify access paths for particular workloads. Prior studies have shown that relational design choices influence retrieval efficiency, query throughput, database size, and the computational complexity of query execution [7-12]. This indicates that database design should be evaluated not only from a conceptual and integrity-oriented perspective, but also from an empirical performance perspective.

The performance consequences of relational design become especially important in the context of query optimization. Cost-based optimizers compare alternative access paths, join orders, and execution strategies using estimated resource cost and data-distribution information [1-5]. Schema characteristics such as primary keys, foreign keys, table decomposition, and relation dependencies influence the optimizer’s ability to select efficient plans [1], [2], [4], [10]. Recent studies have further shown that additional dependency information and cardinality-estimation quality can significantly affect query-plan efficiency and execution behavior [10], [13]. Therefore, two schemas implementing the same business domain may lead to substantially different optimizer decisions and query runtimes, even when the logical meaning of the workload is similar.

Although prior literature has examined normalization, denormalization, and query optimization extensively, fewer studies directly compare alternative ERD-level relational designs for the same application domain and evaluate their performance across diverse query categories using both response time and optimizer-estimated query cost. Existing studies tend to focus either on general normalization strategies [6-12] or on optimizer behavior and cost estimation [10], [13],

Juanda Hakim Lubis is an Assistant Proffesor at Universitas Pembangunan Masyarakat Indonesia, Medan, Indonesia (e-mail: juandahakim@gmail.com).

Ivan Jaya is an Assistance Professor at the Universitas Sumatera Utara, Medan, Indonesia (e-mail: ivanjaya@usu.ac.id).

Fajrul Malik Aminullah Napitupulu is an Assistant Proffesor at Institut Modern Arsitektur dan Teknologi, Deli Serdang, Indonesia (e-mail: fajrul.napitupulu@gmail.com).

Herman Mawengkang is a Distinguished Professor at Universitas Sumatera Utara, Medan, Indonesia (e-mail: mawengkang@usu.ac.id).

leaving a gap in understanding how alternative relational design choices influence practical DBMS performance under a shared workload.

This study addresses that gap by empirically comparing three relational database designs: ERD-1, ERD-2, and ERD-3, constructed for the same inventory-oriented application domain. The models are evaluated across multiple data volumes and seven categories of queries: simple queries, aggregate queries, inner joins, outer joins, subqueries, correlated subqueries, and complex queries. Performance is assessed using two complementary indicators: observed query response time and cost-based optimizer output.

The contribution of this study is threefold. First, it provides an empirical comparison of alternative relational design choices under a common domain and workload. Second, it demonstrates how schema structure affects query execution behavior, especially for join-intensive, aggregation-heavy, and complex retrieval operations. Third, it offers practical insight for database designers and administrators seeking to align ERD decisions with query-performance objectives.

II. Theory and Related works

A. Relational Schema Design and Query Performance

Relational schema design has traditionally been guided by normalization theory, which aims to reduce data redundancy, preserve consistency, and prevent update anomalies [6]. While these goals remain central to logical database design, the performance implications of schema structure are more complex. A highly normalized design can increase the number of joins required to reconstruct business entities during retrieval, whereas a selectively denormalized or differently decomposed schema may reduce retrieval complexity for certain workloads.

Sanders and Shin examined the effects of denormalization on relational database performance and argued that denormalization can improve query efficiency when it is applied systematically and in accordance with application requirements [7]. Shin and Sanders later extended this line of work by analyzing denormalization strategies in data warehouses, showing that alternative schema organizations influence retrieval cost, join complexity, and the efficiency of analytical queries [8]. These studies established that relational design decisions can have measurable performance consequences beyond data integrity considerations.

More recent work has reinforced this conclusion. Milićev proposed a formal model of denormalized transactional relational databases, illustrating how alternative schema organizations can reshape retrieval paths and optimization opportunities [9]. Taipalus empirically studied the effects of logical database design on database size, query complexity, query performance, and energy consumption, finding that normalization levels can produce distinct trade-offs across these dimensions [11]. Fotache et al. similarly reported that varying degrees of normalization affect decision-support query performance, confirming that schema structure remains a relevant empirical factor in modern database research [12].

Taken together, these works show that logical database design affects both the structural complexity of queries and the efficiency of retrieval operations. However, most of the existing literature evaluates normalization or denormalization at a general level. Less attention has been given to comparing multiple ERD-level alternatives representing the same application domain across a broad range of operational query types. The present study addresses this limitation by examining how three relational designs differ in performance under shared workloads and data scales.

B. Query Optimization, Cost Estimation, and Schema Effects

Modern DBMSs rely on cost-based optimization to identify efficient execution plans. The foundational work of Selinger et al. introduced cost-based access-path selection, establishing the principle that optimizers evaluate competing alternatives such as index access, table scans, and join orders based on estimated execution cost [1]. Chaudhuri later summarized the major components of relational query optimization, including query rewriting, access-path selection, and join optimization [2]. Graefe further reviewed the physical execution techniques used for large database workloads [3].

The quality of optimizer decisions depends heavily on the accuracy of statistics and selectivity estimation. Poosala et al. demonstrated the importance of improved histograms for estimating predicate selectivity [4]. Leis et al. later showed that query optimizers can suffer significant performance degradation when cardinality estimates are inaccurate [5]. Ba and Rigger further confirmed that cardinality-estimation errors remain a meaningful source of DBMS performance issues across mature database systems [13].

Schema design is directly connected to these optimization concerns. Primary-key and foreign-key structures, entity decomposition, join relationships, and available dependencies influence the optimizer's ability to select efficient plans. Lindner et al. showed that additional data dependencies can improve query optimization and reduce execution time beyond what is possible using only conventional key constraints [10]. This finding supports the broader argument that relational structure is not merely a modeling concern, but also a factor that shapes optimizer behavior and query efficiency.

Despite extensive research on query optimization, most studies focus on optimizer algorithms, statistics, or estimation quality rather than on how alternative ERD-level designs alter optimizer-reported query cost under equivalent application semantics. The present study contributes to this area by examining how different relational design models affect both observed response time and optimizer-estimated cost across multiple query categories.

C. Research Gap

Prior studies have established that relational schema design influences query performance [7-12] and that optimizer effectiveness depends on access paths, cardinality estimation, and structural dependencies [1-5], [10], [13]. However, these two research strands are often investigated separately. There remains a need for empirical studies that connect database design alternatives with query execution behavior in a unified experimental setting.

Specifically, relatively few works compare several ERD-level relational designs for the same domain while simultaneously evaluating:

- response time across multiple query categories;
- cost-based optimizer outputs;
- and the relative behavior of alternative relational structures under increasing data volume.

This study addresses that gap by comparing three relational database models under a common application domain and workload configuration. Its novelty lies in empirically demonstrating how alternative relational design choices influence query performance and optimizer-estimated cost, thereby providing practical evidence for database designers seeking to make performance-aware schema decisions.

D. Storage Architecture, Database Tuning, and Foundational DBMS Concepts

Database performance is also affected by storage architecture, memory hierarchy, and system-level tuning. Recent database-tuning studies have explored automated tuning support using large-language-model-guided optimization, showing that performance improvement is increasingly treated as a combined problem of configuration, workload behavior, and optimizer decisions [14]. In storage-oriented database research, SSD and flash-memory studies have shown that flash-based storage can reduce random I/O latency and improve enterprise database responsiveness compared with magnetic disks [15-17]. NVMe SSDs further change database bottlenecks by reducing storage latency and increasing I/O parallelism, although query operators, indexing strategies, and access-path selection remain important for performance [18-20].

The HDD-based experimental setting in the present study should therefore be interpreted carefully. Prior HDD-versus-SSD evaluation indicates that storage media can significantly affect query response and data-loading behavior [21]. At the same time, RAID remains relevant for understanding performance–reliability trade-offs in disk-based systems. Foundational RAID studies introduced disk striping and mirroring as mechanisms for improving throughput and reliability, while later RAID evaluations showed that storage architecture influences database workload behavior [22-24]. In addition, database performance has long been shaped by memory-access bottlenecks and by core DBMS principles such as indexing, query processing, relational design, and transaction-oriented data management [25-27].

III. Methodology

This study applies a controlled experimental method to evaluate how different relational database designs affect query performance in an inventory database environment. The experiment compares three alternative entity-relationship designs that represent the same business domain but differ in the way item, supply, employee, material-document, and transaction data are structured. The purpose of the experiment is to determine whether changes in logical database design influence query response time and optimizer-estimated query cost under comparable data, query, and storage conditions.

The inventory domain was selected because inventory systems commonly involve master data, supplier records, employee records, material receipt, material issue, stock control, and transaction-document processing. These operations require frequent selection, aggregation, join, subquery, and complex query execution. Therefore, the domain provides a suitable basis for evaluating the effect of relational schema structure on database performance.

The methodological design follows the general principle of controlled database performance evaluation, where competing schema designs are implemented under the same hardware, software, data-volume, query-workload, and storage conditions. Existing literature on relational database design and query optimization emphasizes that schema structure, keys, relationships, indexes, and access paths can influence query execution behavior. Therefore, the present experiment focuses on the relationship between conceptual database design and practical query performance.

A. Experimental Design

The experiment was conducted in six main stages. First, three alternative entity-relationship models were developed for the same inventory-system domain. Second, each ERD was converted into a relational database schema and implemented in Oracle 11gR2. Third, the same inventory data scenario was applied to each model. Fourth, the three schemas were tested using the same query workload. Fifth, the tests were repeated under three storage configurations: Direct Attached Storage, RAID-0, and RAID-1. Finally, performance was evaluated using response time and optimizer-estimated query cost.

To ensure a fair comparison, the database management system, hardware platform, storage configuration procedure, data scale, query categories, and measurement process were kept consistent across all tests. The main experimental variables were the database design model and the storage configuration. The dependent performance indicators were query response time and optimizer-estimated cost.

B. Hardware, Software, and Storage Environment

The experiment was performed using a computer equipped with an Intel Core i3-3220 processor, 8 GB DDR3 RAM, and two 500 GB hard disk drives. The operating system used was Windows 7 Ultimate, and the database management system was Oracle 11gR2.

Three storage configurations were tested:

1. Direct Attached Storage (DAS), representing a conventional single-storage configuration.
2. RAID-0, representing a striped storage configuration intended to improve read and write throughput by distributing data across disks.
3. RAID-1, representing a mirrored storage configuration intended to improve redundancy by duplicating data across disks.

Each database model was implemented and tested under the same storage conditions. This allowed the experiment to observe not only the performance difference among ERD models, but also whether the relative performance pattern changed when the storage configuration was modified.

C. Relational Database Models

Three relational database models were designed for the same inventory domain. All three models include the core business objects required by the inventory system, namely items, suppliers, supply records, employees, material documents, and transaction records. However, the models differ in how these objects are grouped, generalized, or specialized. The three ERDs are presented in Fig. 1–3.

Fig. 1: ERD-1 separated receive and issue transaction model

1) ERD-1: Separated Receive and Issue Transaction Model

ERD-1 uses a separated transaction structure. The main entities in this model are Items, Supplier, Supply, Employee, Material_doc_receive, Material_doc_issue, Transaction_receive, and Transaction_issue.

In this model, material receipt and material issue activities are represented using separate document and transaction tables. The Material_doc_receive table records receipt-related material documents, while the Material_doc_issue table records issue-related material documents. Similarly, Transaction_receive and Transaction_issue store the transaction details associated with received and issued materials.

The Items table stores item information such as item identifier, material name, item category, minimum order, maximum order, and stock. The Supplier table stores supplier data, while the Supply table connects suppliers and items. The Employee table is connected to both material-document tables, indicating that employees are responsible for material receipt and issue documentation.

The main characteristic of ERD-1 is the physical separation of receipt and issue transactions. This structure can reduce ambiguity between transaction types because each activity has its own table. However, it may require additional query logic when a query needs to retrieve both receipt and issue activities together.

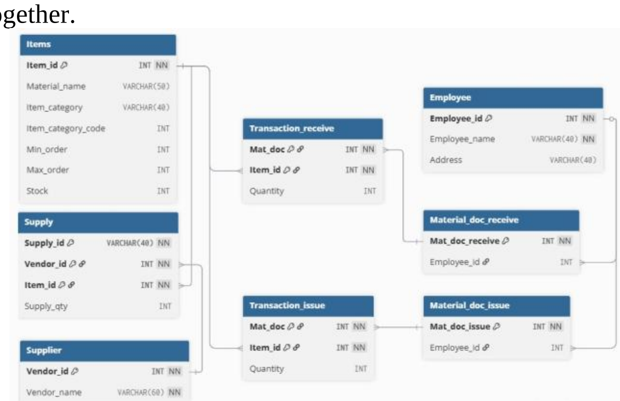


Fig. 2: ERD-2 generalized transaction-document model

2) ERD-2: Generalized Transaction-Document Model

ERD-2 uses a more generalized structure than ERD-1. The main entities are Items, Supplier, Supply, Employee, Material_doc, and Transaction_doc.

In this model, material receipt and material issue activities are not stored in separate transaction-document tables. Instead, the Material_doc table stores general document information and uses document category attributes, such as Doc_category and Doc_category_code, to distinguish document types. The Transaction_doc table stores transaction details and is connected to both Material_doc and Items.

The main advantage of ERD-2 is structural simplicity. Compared with ERD-1, it reduces the number of separate transaction-document tables by generalizing document and transaction data into common tables. This design may simplify schema maintenance because similar transaction types are stored in a unified structure. However, the use of category attributes may require additional filtering conditions during query execution, especially when the query needs to distinguish receipt and issue transactions.

Therefore, ERD-2 represents a generalized schema design that prioritizes consolidation of similar business processes into common relations.

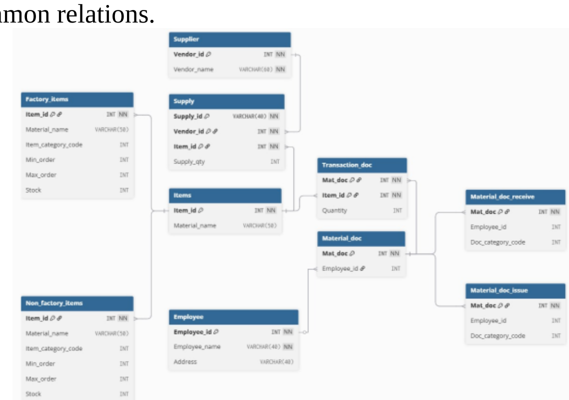


Fig. 3: ERD-3 specialized item and material-document model

3) ERD-3: Specialized Item and Material-Document Model

ERD-3 introduces specialization in both item and material-document structures. The main entities include Factory_items, Non_factory_items, Items, Supplier, Supply, Employee, Material_doc, Material_doc_receive, Material_doc_issue, and Transaction_doc.

Unlike ERD-1 and ERD-2, ERD-3 separates item data into Factory_items and Non_factory_items. These specialized item entities are connected to the general Items entity. This structure represents a specialization approach in which different item types are modeled explicitly while still maintaining a shared item reference for transaction processing.

ERD-3 also uses a generalized Material_doc table that is further associated with Material_doc_receive and Material_doc_issue. This allows the schema to maintain a common material-document reference while still distinguishing receipt and issue documents through specialized tables. The Transaction_doc table records transaction quantities and connects material documents with item records.

The main characteristic of ERD-3 is its hybrid structure. It combines generalization and specialization by using common parent entities while separating important business subtypes. This design may improve query organization when queries target specific item types or document types. However, it may also introduce additional joins when information must be retrieved across generalized and specialized tables.

D. Comparison of the Three ERD Models

The three ERD models represent different database design strategies for the same inventory domain. ERD-1 applies a separated transaction design, ERD-2 applies a generalized document design, and ERD-3 applies a specialized item and material-document design.

ERD-1 separates material receipt and material issue activities into different document and transaction tables. This design makes the distinction between receiving and issuing activities explicit at the table level. As a result, queries that focus only on receipt transactions or only on issue transactions can access the relevant table directly. However, queries that need to analyze both transaction types may require additional query operations, such as combining results from separate tables.

ERD-2 reduces the number of transaction-related tables by storing material-document data in a generalized Material_doc table and transaction data in a generalized Transaction_doc table. Receipt and issue activities are distinguished through document category attributes instead of separate document tables. This approach simplifies the schema structure and may reduce redundancy. However, because different transaction types are stored in common tables, queries may require additional filtering conditions to separate receipt and issue documents.

ERD-3 combines generalization and specialization. It maintains common reference tables such as Items, Material_doc, and Transaction_doc, but also separates specific item and document subtypes into specialized tables. Factory and non-factory items are represented separately, while receipt and issue documents are also represented through specialized material-document tables. This structure provides more explicit representation of business subtypes. It may support more targeted access for specific item or document categories, although it may also increase the number of joins when queries must retrieve data across both generalized and specialized entities.

This comparison is important because the experiment does not compare arbitrary schemas. Instead, it compares three alternative modeling strategies: separation, generalization, and specialization. These strategies can affect the number of joins, filtering conditions, access paths, index usage, and optimizer-estimated query cost during execution.

E. Schema Implementation and Key Constraints

Each ERD was transformed into a relational schema and implemented in Oracle 11gR2. Primary keys were defined for all main entities, including item identifiers, supplier identifiers, employee identifiers, supply identifiers, material-document identifiers, and transaction-document identifiers. Foreign-key relationships were defined according to the relationships shown in the ERDs.

In ERD-1, the Items, Supplier, Supply, and Employee tables are connected to separate material-document and transaction tables for receipt and issue operations. The receipt-related tables are represented by Material_doc_receive and Transaction_receive, while the issue-related tables are represented by Material_doc_issue and Transaction_issue.

In ERD-2, the schema uses a generalized structure. The Material_doc table stores common document information and is connected to the Employee table. The Transaction_doc table connects material documents with item records and stores transaction quantity information. Supplier and item relationships are represented through the Supply table.

In ERD-3, the schema includes specialized item and document structures. Factory_items and Non_factory_items represent item subtypes, while Material_doc_receive and Material_doc_issue represent material-document subtypes. These subtype tables are connected to general reference tables such as Items, Material_doc, and Transaction_doc.

All three models were implemented using comparable business rules. Although their table structures differ, the underlying inventory information remains equivalent across the models. This equivalence is necessary to ensure that performance differences are caused by schema design rather than by differences in business content.

F. Indexing Configuration

Indexing was controlled to ensure a fair comparison among the three ERD models. Primary-key indexes were created for all primary-key attributes. Foreign-key attributes used in join operations were also indexed consistently across equivalent relationships. No additional query-specific secondary indexes were introduced unless required by the schema structure.

In ERD-1, indexing was applied to the primary-key attributes of the item, supplier, employee, supply, material-document, and transaction tables. Foreign-key attributes connecting items, suppliers, employees, material documents, and transaction records were also indexed where they were used in join operations.

In ERD-2, indexing was applied to the primary keys of Items, Supplier, Employee, Supply, Material_doc, and Transaction_doc. Foreign-key attributes connecting supplier-to-supply, item-to-supply, employee-to-material-document, material-document-to-transaction, and item-to-transaction relationships were indexed consistently.

In ERD-3, indexing was applied to primary keys in the generalized and specialized entities, including item, factory item, non-factory item, supplier, employee, material-document, material-document subtype, and transaction-document entities. Foreign-key attributes connecting item specialization, material-document specialization, supply records, and transaction records were indexed where required for join operations.

This indexing strategy was used to avoid giving one model an artificial performance advantage. Because the aim of the study is to evaluate the effect of database design, the indexing configuration was kept as consistent as possible across ERD-1, ERD-2, and ERD-3. As a result, the observed differences in response time and optimizer cost can be interpreted as being primarily associated with differences in relational structure, relationship complexity, and storage configuration.

G. Data Population

Each database model was populated using equivalent inventory-system data. The data covered item records, supplier records, employee records, supply records, material documents, and transaction records. The same data scale was applied across the three models as far as the schema structure allowed.

The experiment used large-scale data scenarios ranging from 100,000 to 100,000,000 records or entities, depending on the test condition. The purpose of using multiple data volumes was to evaluate whether the performance behavior of each ERD model remained stable as the database size increased.

Using comparable data across ERD-1, ERD-2, and ERD-3 was necessary to maintain experimental validity. If one model contained substantially different data from another model, the performance results could be influenced by data distribution rather than by schema design. Therefore, the data population process was controlled to preserve equivalence across the three schema alternatives.

H. Query Workload

Seven query categories were used to evaluate database performance: simple selection query, aggregate query, inner join query, outer join query, subquery, correlated subquery, and complex query.

These query categories were selected because they represent common operations in relational database systems and inventory applications. Simple selection queries evaluate basic retrieval performance. Aggregate queries evaluate summarization performance. Join queries evaluate the effect of relationships among tables. Subqueries and correlated subqueries evaluate nested query behavior. Complex queries evaluate the combined effect of filtering, joining, aggregation, and ordering operations.

The same logical query workload was executed on each ERD model. Because the table structures differ across ERD-1, ERD-2, and ERD-3, the SQL syntax was adjusted where necessary to match each schema. However, the business meaning of each query was kept equivalent. This ensured that the comparison measured the effect of database design rather than differences in query purpose.

I. Query Execution Procedure

Each query was executed under every combination of ERD model and storage configuration. Before measurement, the database environment was stabilized to reduce irregular cache effects. Each query was executed three times, and the mean response time was used as the reported execution time.

The response time represents the elapsed time required to execute a query and return the result. The same measurement procedure was applied to ERD-1, ERD-2, and ERD-3 under DAS, RAID-0, and RAID-1. Repeating each query helped reduce the influence of temporary system variation during execution.

Each query was executed three times, and the mean response time was used as the primary performance measure. Because the available experimental records contain summarized mean values rather than complete run-level logs, this study does not report standard deviation or confidence intervals. This limitation is explicitly addressed in the threats-to-validity discussion and should be considered when interpreting the statistical strength of the findings.

J. Optimizer-Cost Measurement

In addition to response time, the study measured optimizer-estimated query cost using the Oracle 11gR2 cost-based optimizer. Optimizer cost is a relative estimate of the computational work required to execute a query. It may be influenced by table access methods, index usage, join operations, filtering conditions, estimated cardinality, and disk I/O.

For each query category, optimizer cost was collected using equivalent query logic across the three ERD models. This allowed the study to compare whether the model with lower response time also produced lower estimated execution cost. Because optimizer cost is not the same as actual elapsed time, it was interpreted together with response time rather than as a direct substitute for runtime measurement.

Representative query execution plans were considered to explain why one ERD model performed better than another. In simple selection queries, the analysis considered whether the query accessed a direct item or document relation or required filtering through category attributes. In aggregate queries, the analysis considered whether aggregation was performed over separated transaction tables, generalized transaction tables, or specialized item/document structures. In join queries, the analysis considered the number of relationships traversed and whether the schema required additional joins to retrieve equivalent business information. In complex queries, the analysis considered the combined effect of filtering, joining, grouping, and ordering operations.

This execution-plan interpretation was used to support the response-time and optimizer-cost results. A schema requiring fewer unnecessary joins, fewer filtering conditions, or more direct access paths was expected to produce lower optimizer cost and faster response time.

K. Data Analysis

The analysis compared the three ERD models using two performance indicators: response time and optimizer-estimated query cost. The results were analyzed by query category and storage configuration.

The first stage of analysis compared ERD-1, ERD-2, and ERD-3 under the same storage configuration. This showed how schema structure affected query performance. The second stage compared DAS, RAID-0, and RAID-1 for each ERD model. This showed how storage configuration affected the performance of each schema. The third stage examined whether the same ERD model remained superior across different query types and storage configurations.

The interpretation focused on the relationship between schema structure and query execution behavior. ERD-1 was interpreted as a separated transaction design, ERD-2 as a generalized document design, and ERD-3 as a specialized item and document design. Differences in response time and optimizer cost were explained based on the number of relations accessed, join complexity, filtering requirements, and possible index usage.

L. Validity Controls

Several validity controls were applied. First, all three ERD models were tested using the same hardware, operating system, and DBMS. Second, the same storage configurations were used across all models. Third, equivalent data scenarios were applied to each model. Fourth, equivalent query categories were executed across the three schemas. Fifth, primary-key and foreign-key constraints were consistently defined. Sixth, indexing was limited to primary-key and foreign-key attributes to reduce uncontrolled optimization effects.

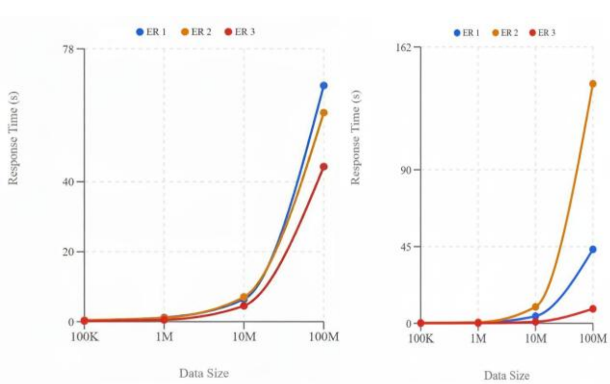
Despite these controls, the experiment has several limitations. The hardware platform used HDD-based storage and an older Oracle 11gR2 environment. Therefore, the results may differ in modern SSD, NVMe, cloud, or distributed database environments. The experiment also focused on a single inventory-system domain, so the results should not be generalized to all application domains without further testing. In addition, if only mean response times are available, the absence of standard deviation and confidence intervals limits the statistical strength of the findings.

IV. RESULTS AND DISCUSSION

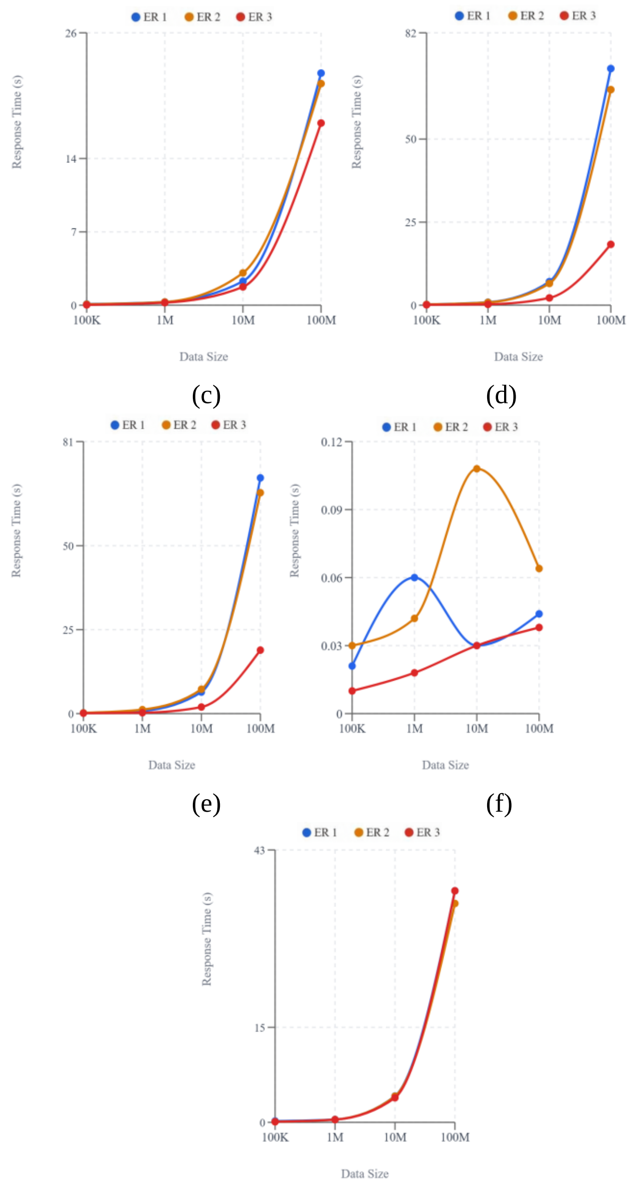
This section presents the analysis of query performance across the three relational database models and three storage configurations: DAS, RAID-1, and RAID-0. The evaluation is conducted by measuring response time and query cost across seven types of queries.

A. Response Time Analysis under DAS Storage

This subsection presents the response-time comparison of ERD-1, ERD-2, and ERD-3 under the Direct-Attached Storage (DAS) configuration. The evaluation covers seven query categories and four data-size levels, ranging from 100 thousand to 100 million records. As shown in Fig. 4, the response time of all ERD models remains relatively low at smaller data sizes, but the performance gap becomes more visible as the data volume increases.



(a) (b)



(c) (d)

(e) (f)

(g)

Fig. 4: Response time under DAS storage: seven subfigures, namely (a) Simple, (b) Aggregate, (c) Inner Join, (d) Outer Join, (e) Subquery, (f) Correlated Subquery, (g) Complex Query

For simple queries, ERD-3 records the lowest response time at the largest data size, followed by ERD-2 and ERD-1. This indicates that even simple retrieval operations become sensitive to schema structure when the data volume reaches 100 million records. A more pronounced difference appears in aggregate queries, where ERD-2 shows the highest response time, while ERD-3 remains substantially lower. This suggests that aggregation workloads are strongly affected by table structure, access paths, and the amount of intermediate data that must be processed.

For inner join and outer join queries, ERD-3 again shows better scalability than ERD-1 and ERD-2. The gap becomes especially apparent at 100 million records, indicating that the more modular structure of ERD-3 reduces the execution burden of join-based retrieval. The same tendency appears in subquery workloads, where ERD-3 maintains a lower response time as the data size increases.

The correlated subquery results show a different pattern because the response times remain very small across all data sizes. Nevertheless, ERD-3 still provides the most stable behavior compared with the other models. In complex queries, ERD-3 remains competitive, although the performance difference among the ERD models is not as large as in aggregate and join-based workloads. Overall, the DAS results indicate that ERD-3 provides the most consistent response-time advantage across most query categories, particularly when the workload reaches large-scale data volumes.

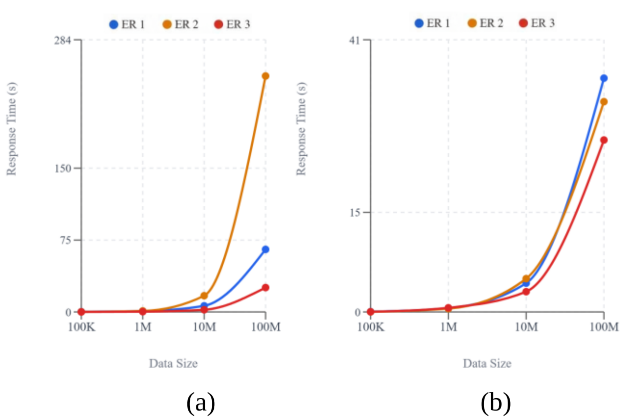
B. Response Time Analysis under RAID-1 Storage

Fig. 5 presents the response-time results under RAID-1 storage. RAID-1 provides mirroring for redundancy, but this configuration may introduce additional overhead depending on the query type and storage access pattern. Similar to the DAS scenario, the response times remain low at smaller data sizes and increase sharply at 100 million records for several query categories.

In aggregate queries, ERD-2 records the highest response time at 100 million records, while ERD-3 remains the fastest. This shows that ERD-2 is less efficient for aggregation workloads under RAID-1, likely because its relational structure requires more expensive data access and processing. In simple queries, ERD-3 also produces the lowest response time, while ERD-1 and ERD-2 require longer execution times at the largest data size.

For inner join, outer join, and subquery workloads, ERD-3 consistently performs better than the other two ERD models. The difference is particularly clear in subquery and inner join operations, where ERD-1 and ERD-2 increase substantially at 100 million records. This result suggests that RAID-1 does not eliminate the effect of schema design; instead, the efficiency of the relational model remains important even when the storage configuration provides redundancy.

For correlated subqueries, the response times are very small, but ERD-2 shows higher values than ERD-1 and ERD-3. In complex queries, the three models show closer results than in aggregate and subquery workloads, although ERD-3 remains among the most efficient. Overall, the RAID-1 results support the finding that ERD-3 is generally more scalable and less sensitive to increasing data volume.



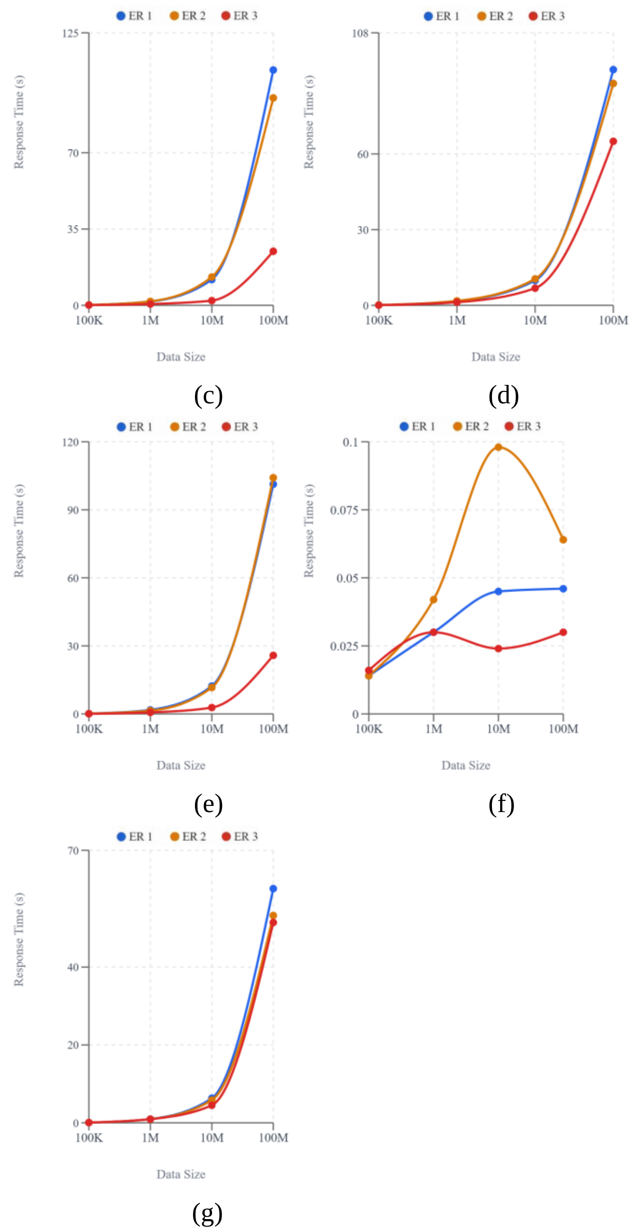


Fig. 5: Response time comparison of ERD-1, ERD-2, and ERD-3 under RAID-1 storage across seven query types: (a) simple query, (b) aggregate query, (c) inner join query, (d) outer join query, (e) subquery, (f) correlated subquery, and (g) complex query

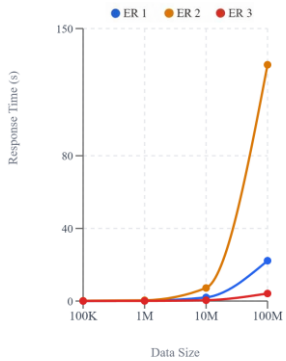
C. Response Time Analysis under RAID-0 Storage

Fig. 6 shows the response-time comparison under RAID-0 storage. RAID-0 is designed to improve performance by distributing data across disks, but it does not provide redundancy. The results show that storage configuration changes the absolute response time, but the influence of ERD structure remains visible across query categories.

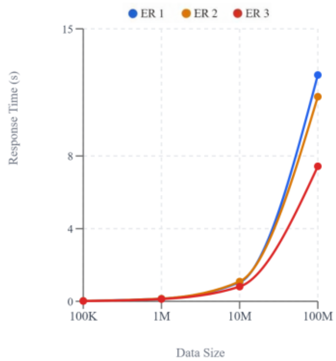
For aggregate queries, ERD-2 again records the highest response time at 100 million records, while ERD-3 produces the lowest response time. This confirms that aggregation workloads are highly sensitive to schema design, regardless of whether the storage configuration is DAS, RAID-1, or RAID-

0. In simple queries, ERD-3 also performs better than ERD-1 and ERD-2 at the largest data size.

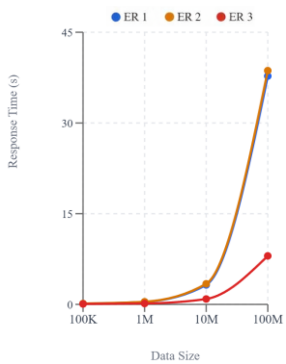
For inner join, outer join, and subquery workloads, ERD-3 consistently shows lower response time than ERD-1 and ERD-2. The difference is especially strong in subquery workloads, where ERD-1 and ERD-2 rise sharply at 100 million records, while ERD-3 remains much lower. This indicates that ERD-3 is more effective in controlling query execution growth as data volume increases.



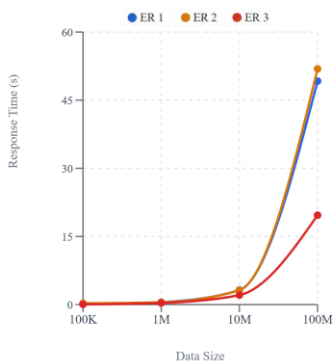
(a)



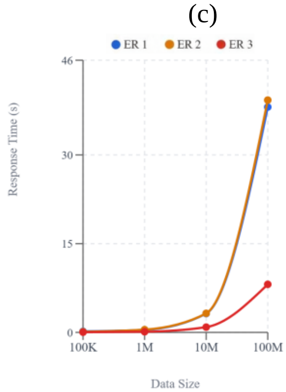
(b)



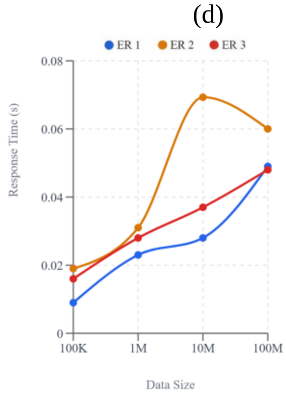
(c)



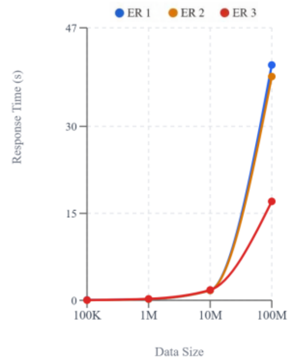
(d)



(e)



(f)



(g)

Fig. 6: Response time comparison of ERD-1, ERD-2, and ERD-3 under RAID-0 storage across seven query types: (a) simple query, (b) aggregate query, (c) inner join query, (d) outer join query, (e) subquery, (f) correlated subquery, and (g) complex query

For correlated subqueries, the response times are again very small compared with other query types. However, ERD-2 still shows the highest response time, while ERD-1 and ERD-3 remain lower. In complex queries, ERD-3 records the lowest response time at the largest data size, whereas ERD-1 and ERD-2 show higher values. Overall, the RAID-0 results further strengthen the conclusion that ERD-3 provides the most favorable response-time behavior across most workloads.

D. Cost-Based Optimization Analysis across Storage Configurations

Fig. 7 compares the Oracle optimizer query cost for ERD-1, ERD-2, and ERD-3 at 100 million records under DAS, RAID-1, and RAID-0 configurations. The cost-based optimizer results provide a complementary perspective to the response-time analysis by showing the estimated computational cost of each query type.

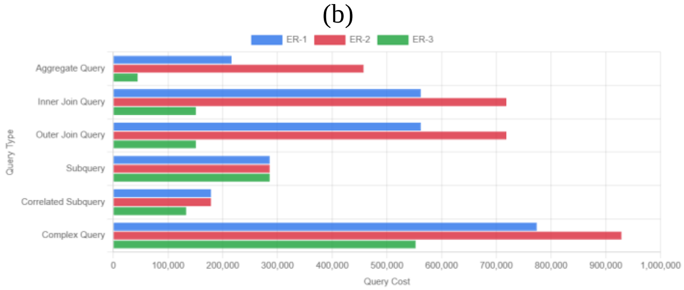
Across the three storage configurations, ERD-3 generally produces the lowest query cost for aggregate queries, inner join queries, outer join queries, correlated subqueries, and complex queries. ERD-2 frequently records the highest optimizer cost, especially in aggregate, join-based, and complex workloads. This indicates that ERD-2 tends to generate more expensive access paths and query execution plans than the other two models.

For subqueries, the query cost values of the three ERD models appear relatively similar compared with other query types. This suggests that the optimizer estimates a comparable execution burden for subquery operations across the three database designs. However, for aggregate, join, correlated, and complex queries, ERD-3 shows a clear advantage in terms of lower optimizer-estimated cost.

The similarity of the cost patterns across DAS, RAID-1, and RAID-0 suggests that the optimizer cost is more strongly influenced by schema structure and query formulation than by the physical storage configuration alone. Therefore, the cost-based optimization results support the response-time findings: Within the tested inventory-domain workload, ERD-3 showed the most consistent performance advantage, while ERD-2 tends to be the most expensive model in several query categories.

(a)

(b)



(c)

Fig. 7: Oracle optimizer query cost comparison at 100 million records under different storage configurations: (a) DAS, (b) RAID-1, and (c) RAID-0

The optimizer-cost results were further interpreted using representative Oracle execution plans. Across the tested workloads, ERD-3 generally required fewer unnecessary filtering steps and more direct access paths for document- and item-related retrieval, while ERD-2 often required additional category-based filtering through generalized document structures. This supports the observed pattern in which ERD-3 produced lower optimizer-estimated cost for aggregate, join-intensive, correlated, and complex queries.

E. Comparative Discussion and Practical Implications

The overall results demonstrate that database schema design has a substantial impact on query performance. Across DAS, RAID-1, and RAID-0 configurations, ERD-3 generally provides the most consistent performance advantage, especially for aggregate queries, inner joins, outer joins, subqueries, and complex queries. This suggests that a more modular relational structure can reduce query complexity and improve access-path efficiency.

However, the results also show that performance superiority is not determined by ERD structure alone. Query type, data size, and storage configuration jointly influence the final response time. At smaller data sizes, the differences among ERD models are relatively small. At 100 million records, however, the performance gap becomes much more apparent. This indicates that schema design decisions become increasingly important as database scale grows.

From a practical perspective, ERD-3 is the most recommended design for large-scale workloads involving aggregation, joins, and subqueries. ERD-1 may still be acceptable for simpler workloads, but it becomes less efficient as data volume and query complexity increase. ERD-2 should be used carefully because it frequently produces higher response time and optimizer cost, particularly in aggregate and join-intensive workloads.

The storage results also show that RAID configuration should be selected based on system priority. RAID-0 may be suitable when performance is the main concern, while RAID-1 is more appropriate when data redundancy is required. Nevertheless, the findings indicate that storage improvement alone cannot fully compensate for inefficient schema design. A well-structured ERD remains essential for achieving scalable query performance.

V. Conclussion

This study examined how alternative relational database designs influence query performance under a common inventory-oriented application domain. Three ERD models were evaluated across multiple data volumes and seven query categories using two complementary indicators: observed response time and cost-based optimizer output. The results show that relational design choices materially affect database performance, particularly for aggregate, join-intensive, correlated, and complex queries.

Among the evaluated models, ERD-3 demonstrated the most consistent overall performance advantage. Its more modular relational structure reduced unnecessary query complexity, supported more efficient access paths, and produced lower optimizer-estimated costs in several workload categories. These findings indicate that schema design should not be treated only as a data-modeling activity, but also as a performance-sensitive engineering decision. A relational model that aligns entity decomposition, relationship structure, and index utilization with expected query patterns can improve execution efficiency as data volume increases.

The study also highlights the practical importance of interpreting performance in relation to storage architecture. Within the tested HDD-based environment, the results suggest that storage configuration and relational design jointly affect query behavior. However, the contribution of this work lies primarily in demonstrating the design-level impact of ERD alternatives on query execution, rather than making universal claims about one storage architecture across all modern systems.

Several limitations should be acknowledged. First, the experiments were conducted on a legacy hardware and software platform using HDD storage, Windows 7, and Oracle 11gR2, which may not fully represent current production environments. Second, the study focused on a single DBMS platform and one application domain, limiting direct generalization to other database engines and workloads. Third, the reported results would be strengthened by a more explicit statistical treatment of repeated runs, including variance measures and confidence intervals. Finally, although cost-based optimizer outputs were useful for comparative analysis, such cost values remain DBMS-specific estimates rather than hardware-independent performance measures.

Future work should extend this study in several directions. First, the same ERD comparison should be reproduced on modern SSD and NVMe-based systems to determine whether the observed ranking among ERD models persists under lower-latency storage. Second, evaluation across additional DBMS platforms, such as PostgreSQL or MySQL, would improve external validity. Third, future experiments should include repeated-trial statistics, execution-plan inspection, and more detailed indexing analysis to provide a stronger causal explanation of performance differences. Finally, the study could be expanded to other application domains and mixed transactional–analytical workloads to assess whether the design principles observed here remain robust under broader database scenarios.

Overall, this research provides empirical evidence that relational design alternatives can produce meaningful differences in query response time and optimizer-estimated cost. The findings support a practical recommendation for database designers and administrators: schema design decisions should be evaluated not only for correctness and maintainability, but also for their measurable impact on query performance.

REFERENCES

[1] P. G. Selinger, M. M. Astrahan, D. D. Chamberlin, R. A. Lorie, and T. G. Price, "Access path selection in a relational database management system," in Proc. ACM SIGMOD Int. Conf. Manage. Data, 1979, pp. 23–34, https://doi.org/10.1145/582095.582099.

[2] S. Chaudhuri, "An overview of query optimization in relational systems," in Proc. ACM SIGACT-SIGMOD-SIGART Symp. Principles Database Syst. (PODS), 1998, pp. 34–43, https://doi.org/10.1145/275487.275492.

[3] G. Graefe, "Query evaluation techniques for large databases," ACM Comput. Surv., vol. 25, no. 2, pp. 73–169, 1993, https://doi.org/10.1145/152610.152611.

[4] V. Poosala, Y. E. Ioannidis, P. J. Haas, and E. J. Shekita, "Improved histograms for selectivity estimation of range predicates," in Proc. ACM SIGMOD Int. Conf. Manage. Data, 1996, pp. 294–305, https://doi.org/10.1145/233269.233342.

[5] V. Leis, A. Gubichev, A. Mirchev, P. Boncz, A. Kemper, and T. Neumann, "How good are query optimizers, really?" Proc. VLDB Endow., vol. 9, no. 3, pp. 204–215, 2015, https://doi.org/10.14778/2850583.2850594.

[6] H. Köhler and S. Link, "SQL schema design: Foundations, normal forms, and normalization," in Proc. ACM SIGMOD Int. Conf. Manage. Data, 2016, pp. 267–282, https://doi.org/10.1145/2882903.2915239.

[7] G. L. Sanders and S. K. Shin, "Denormalization effects on performance of RDBMS," in Proc. 34th Hawaii Int. Conf. Syst. Sci., 2001, https://doi.org/10.1109/HICSS.2001.926306.

[8] S. K. Shin and G. L. Sanders, "Denormalization strategies for data retrieval from data warehouses," Decis. Support Syst., vol. 42, no. 1, pp. 267–282, 2006, https://doi.org/10.1016/j.dss.2004.12.004.

[9] D. Milićev, "Hyper-relations: A model for denormalization of transactional relational databases," IEEE Trans. Knowl. Data Eng., 2021. Available: https://ieeexplore.ieee.org/document/9599377.

[10] D. Lindner, D. Ritter, and F. Naumann, "Enabling data dependency-based query optimization," arXiv preprint arXiv:2406.06886, 2024. Available: https://arxiv.org/abs/2406.06886.

[11] T. Taipalus, "On the effects of logical database design on database size, query complexity, query performance, and energy consumption," arXiv preprint arXiv:2501.07449, 2025. Available: https://arxiv.org/abs/2501.07449.

[12] M. Fotache, M. I. Cluci, T. Taipalus, and G. Talaba, "The effects of database normalization on decision support system performance," Inf. Syst., 2025. Available: https://www.sciencedirect.com/science/article/pii/S030643792500122X.

[13] J. Ba and M. Rigger, "CERT: Finding performance issues in database systems through the lens of cardinality estimation," in Proc. IEEE/ACM 46th Int. Conf. Softw. Eng. (ICSE), 2024, https://doi.org/10.1145/3597503.3639076.

[14] J. Lao, Y. Wang, Y. Li, J. Wang, Y. Zhang, Z. Cheng, W. Chen, M. Tang, and J. Wang, "GPTuner: A manual-reading database tuning system via GPT-guided Bayesian optimization," arXiv preprint arXiv:2311.03157, 2023. Available: https://arxiv.org/abs/2311.03157.

[15] S.-W. Lee, B. Moon, C. Park, J.-M. Kim, and S.-W. Kim, "A case for flash memory SSD in enterprise database applications," in Proc. ACM SIGMOD Int. Conf. Manage. Data, 2008, pp. 1075–1086, https://doi.org/10.1145/1376616.1376723.

[16] S.-W. Lee, B. Moon, and C. Park, "Advances in flash memory SSD technology for enterprise database applications," in Proc. ACM SIGMOD Int. Conf. Manage. Data, 2009, pp. 863–870, https://doi.org/10.1145/1559845.1559937.

[17] D. Bausch, I. Petrov, and A. Buchmann, "On the performance of database query processing algorithms on flash solid state disks," in Proc. Int. Workshop Database and Expert Syst. Appl., 2011. Available: https://ieeexplore.ieee.org/document/6059807.

[18] Q. Xu, H. Siyamwala, M. Ghosh, T. Suri, M. Awasthi, Z. Guz, A. Shayesteh, and V. Balakrishnan, "Performance analysis of NVMe SSDs and their implication on real world databases," in Proc. 8th ACM Int. Syst. Storage Conf., 2015, https://doi.org/10.1145/2757667.2757684.

[19] A. Fevgas, L. Akritidis, P. Bozanis, and Y. Manolopoulos, "Indexing in flash storage devices: A survey on challenges, current approaches, and future trends," VLDB J., 2020, https://doi.org/10.1007/s00778-019-00559-8.

[20] G. Graefe, S. Harizopoulos, H. A. Kuno, and J. Shapiro, "Designing database operators for flash-enabled memory hierarchies," IEEE Data Eng. Bull., vol. 33, no. 4, pp. 21–27, 2010. Available: http://sites.computer.org/debull/A10dec/A10DEC-CD.pdf.

[21] S. A. M. Tipan and R. B. Ricafort, "A performance analysis: Evaluating relational database system on HDD vs. SSD storage," Int. J. Latest Technol. Eng. Manag. Appl. Sci., vol. 15, no. 4, 2026, https://doi.org/10.51583/IJLTEMAS.2026.150400049.

[22] D. A. Patterson, G. Gibson, and R. H. Katz, "A case for redundant arrays of inexpensive disks (RAID)," in Proc. ACM SIGMOD Int. Conf. Manage. Data, 1988, pp. 109–116, https://doi.org/10.1145/50202.50214.

[23] P. M. Chen, E. K. Lee, G. A. Gibson, R. H. Katz, and D. A. Patterson, "RAID: High-performance, reliable secondary storage," ACM Comput. Surv., vol. 26, no. 2, pp. 145–185, 1994, https://doi.org/10.1145/176979.176981.

[24] P. M. Chen, E. K. Lee, G. A. Gibson, R. H. Katz, and D. A. Patterson, "Performance and design evaluation of the RAID-II storage server," Distrib. Parallel Databases, vol. 2, no. 3, pp. 243–260, 1994, https://doi.org/10.1007/BF01266330.

[25] S. Manegold, P. Boncz, and M. Kersten, "Optimizing database architecture for the new bottleneck: Memory access," VLDB J., vol. 9, no. 3, pp. 231–246, 2000, https://doi.org/10.1007/s007780000031.

[26] R. Ramakrishnan and J. Gehrke, Database Management Systems, 3rd ed. New York, NY, USA: McGraw-Hill, 2003. ISBN: 978-0-07-246563-1.
[27] H. Garcia-Molina, J. D. Ullman, and J. Widom, Database Systems: The Complete Book, 2nd ed. Upper Saddle River, NJ, USA: Pearson, 2008. ISBN: 978-0-13-187325-4.

Juanda Hakim Lubis is currently pursuing a doctoral program in Computer Science at the University of North Sumatra. He completed his bachelor's degree in informatics engineering at Institut Teknologi Telkom Bandung in 2010 and his master's degree in Informatics Engineering at Universitas Sumatera Utara in 2013. His current research focuses on Software Engineering, database management systems, data science.

Ivan Jaya (Member, IEEE) received the B.Sc. degree in Math and M.Sc. degree in Informatics from Universitas Sumatera Utara, Indonesia, in 2010 and 2014, respectively, where he is currently pursuing the Ph.D. degree in computer vision. He is currently an Assitance Professor with the Department of Information Technology, Universitas Sumatera Utara, Indonesia. His research interests include Artificial Inteligence.

Fajrul Malik Aminullah Napitupulu is a Lecturer in Computer Science at Institut Modern Arsitektur dan Teknologi, Indonesia. He earned his Master’s degree in Information Technology from Universitas Sumatera Utara in 2024. His research interests include Web Technologies, Mobile Applications, and Artificial Intelligence.

Herman Mawengkang is a distinguished Professor at the Department of Mathematics, Universitas Sumatera Utara in Medan, Indonesia. He completed his bachelor’s degree in mathematics at Universitas Sumatera Utara in 1974 and earned his PhD in Operations Research from the University of New South Wales, Sydney, Australia. Professor Mawengkang has extensive expertise in operations research and optimization, contributing significantly to these fields through his research and teaching. His scholarly work is widely recognized, as evidenced by his h-index of 11 and his role as a reviewer for several prestigious academic journals. He is the author or co-author of over 200 research publications in computer science and applied mathematics.