MASTERING SQL PERFORMANCE OPTIMIZATION

FROM QUERY FUNDAMENTALS TO PRODUCTION-GRADE TUNING ACROSS DATABASE PLATFORMS

POSTGRESQL MYSQL SQL SERVER ORACLE SQLITE

FASTER QUERIES.
LOWER COSTS.
BETTER SCALE.

ABBY TAYLOR

Mastering SQL Performance Optimization

From Query Fundamentals to Production-Grade Tuning Across Database Platforms

Steve T. Publications

This book is available at
https://leanpub.com/masteringsqlperformanceoptimization

This version was published on 2026-07-17

Contents

From Query Fundamentals to Production-Grade Tuning Across Database Platforms

About this book: Slow SQL queries cost real money. Every millisecond of latency compounds across thousands of concurrent users, driving up infrastructure bills and degrading user experience. This book teaches you how databases execute queries internally, so you can systematically diagnose and resolve performance problems across PostgreSQL, MySQL, SQL Server, Oracle, and SQLite. Through execution plan analysis, indexing strategies, join optimization, schema design principles, concurrency management, and real-world case studies, you will build the skills to transform sluggish queries into high-performance operations that scale under production load.

Introduction: The Performance Imperative

The Hidden Cost of Slow Queries

On January 31, 2017, GitLab.com suffered an 18-hour outage that began with a database maintenance incident. An engineer accidentally wiped the primary PostgreSQL data directory while troubleshooting replication lag. Compounding the problem, daily `pg_dump` backups had been failing silently for months due to a version mismatch between the backup tool and the database server, and error notification emails were being rejected by DMARC settings. Approximately six hours of production data was lost, affecting roughly 5,000 projects, 5,000 comments, and 700 user accounts [1]. The incident became one of the most transparently documented database disasters in tech history, with the team publishing a detailed public postmortem that has been studied by engineering organizations worldwide.

The GitLab incident was caused by operational procedures, not a slow query, but it illustrates a broader truth: database systems are unforgiving environments where small mistakes cascade into major failures. Slow queries create their own class of disaster, one that accumulates more gradually. A query that takes 200 milliseconds instead of 20 milliseconds may seem like a trivial difference. But when that query runs 50,000 times per hour across your application, you have just added 16.7 hours of cumulative database CPU time every single day. Multiply that by the cost of cloud compute, the infrastructure needed to absorb the load, and the degraded user experience, and what seemed like a minor inefficiency becomes a significant operational expense.

The everyday reality is that most SQL queries in production systems are not optimally written. Developers focus on correctness first, which is absolutely right, but performance often takes a back seat until it becomes visible to end users. By that point, the data has grown, the query patterns have multiplied, and the accumulated technical debt makes optimization more painful than it would have been had performance been considered from the start.

The reality is that most SQL queries in production systems are not optimally written. Developers focus on correctness first, which is absolutely right, but performance often takes a back seat until it becomes visible to end users. By that point, the data has grown, the query patterns have multiplied, and the accumulated technical debt makes optimization more painful than it would have been had performance been considered from the start.

What This Book Will Teach You

This book is built on a single premise: SQL performance optimization is not about memorizing tricks or applying silver-bullet solutions. It is about understanding how databases work internally. When you grasp the mechanics of query execution, indexing, and resource management, you gain the ability to systematically diagnose and resolve performance problems across any relational database system.

You will learn to read execution plans like a roadmap, identifying exactly where time and resources are consumed. You will understand when indexes help, when they hurt, and how to design composite indexes that serve multiple query patterns. You will master the trade-offs between different join algorithms, understand how aggregation and sorting consume memory, and know when to use window functions versus self-joins.

The book covers five major database platforms: PostgreSQL, MySQL, SQL Server, Oracle, and SQLite. While each has its own quirks and features, the fundamental principles of query optimization are shared across all of them. Where platform-specific differences matter, the book highlights them explicitly so you can apply the right technique for your environment.

The Book's Structure

The chapters build progressively from foundation to mastery:

Chapters 1 through 3 establish the core foundations. You will learn how a database processes SQL from text to results, how to read execution plans across all major platforms, and how indexing strategies fundamentally shape query performance.

Chapters 4 through 7 dive into the specific query patterns that dominate production workloads. You will examine common anti-patterns and their

optimized alternatives, master join algorithms and their cost characteristics, understand the performance implications of aggregation and sorting, and learn when to use subqueries, CTEs, or derived tables.

Chapters 8 through 10 expand the scope beyond individual queries to system-level concerns. Schema design decisions, transaction isolation levels, locking strategies, memory management, caching layers, and I/O patterns all affect performance in ways that no amount of query rewriting can fix alone.

Chapter 11 covers advanced techniques including query rewriting transformations, materialized views, adaptive query processing, and database-specific tuning features that push the boundaries of what modern optimizers can achieve.

Chapter 12 provides practical methodologies for monitoring, troubleshooting, and benchmarking in production environments, because optimization is not a one-time activity but an ongoing discipline.

Chapters 13 and 14 synthesize everything into real-world case studies and production-ready checklists that you can apply immediately to your own systems.

How to Use This Book

This book works at multiple levels. If you are new to SQL performance, read it sequentially from start to finish. Each chapter builds on the previous one, and the progressive complexity will take you from foundational concepts to advanced techniques without overwhelming you.

If you already have experience with query optimization, use the chapters as a reference for specific topics. The execution plan analysis chapters will help you interpret plans across different database platforms. The indexing chapters provide decision frameworks for choosing the right index type and structure. The case studies offer concrete examples of real problems and their solutions.

Every chapter contains SQL examples that you can run against your own database system. Where platform-specific syntax differs, multiple versions are provided. Execution plan outputs are shown alongside the queries that produce them, so you can connect theory to practice.

Setting Up Your Environment

To get the most from this book, set up at least one of the following database systems for hands-on experimentation:

PostgreSQL 16+ is recommended as the primary platform. It has excellent EXPLAIN output, supports all major SQL features, and its documentation is among the best in the industry. Install it locally or use Docker:

```
docker run -d --name postgres-lab \
  -e POSTGRES_PASSWORD=labpassword \
  -p 5432:5432 \
  postgres:16
```

MySQL 8.0+ provides a second perspective on query optimization, with different optimizer behaviors and indexing defaults. The InnoDB storage engine is the focus of MySQL coverage in this book.

```
docker run -d --name mysql-lab \
  -e MYSQL_ROOT_PASSWORD=labpassword \
  -p 3306:3306 \
  mysql:8.0
```

SQLite requires no installation and is useful for quick experiments with smaller datasets. Its simplicity makes it an excellent platform for understanding fundamental concepts before moving to server-based systems.

For SQL Server and Oracle, use Docker containers or cloud trial instances. The conceptual coverage in this book applies even if you cannot run every example locally.

The sample databases used throughout the book are based on a simplified e-commerce schema with customers, orders, products, and order items tables. The schema evolves progressively: the base schema is introduced in Chapter 1, indexing layers are added in Chapter 3, partitioning is applied in Chapter 8, materialized views and dimensional models are layered on in Chapters 6 and 11, and the full evolved schema with data generation scripts is documented in Appendix A. Each chapter provides the DDL needed to create the tables and populate them with test data, so you can follow along regardless of which platform you choose.

A Note on Benchmarks and Numbers

Throughout this book, you will encounter performance numbers: execution times, I/O counts, memory usage figures, and throughput measurements. These numbers come from real tests run against the sample databases described in each chapter, but they are illustrative rather than prescriptive. Your actual results will vary based on hardware, data volume, database version, and configuration settings.

The important takeaway is not the specific number but the relative difference between an optimized and unoptimized approach. A query that improves from 5 seconds to 200 milliseconds in my test environment might improve from 30 seconds to 1.5 seconds in yours. The magnitude of improvement and the techniques that produce it are what matter.

When comparing approaches, always measure on your own data with your own workload. Database optimizers are sophisticated but context-sensitive, and a technique that helps one query may hurt another under different conditions.

Chapter 1: How Databases Execute Queries

Chapter Objectives. By the end of this chapter, you will be able to:

- Describe the five phases of query processing (parsing, semantic analysis, rewriting, optimization, execution) and explain what happens in each phase.
- Identify why the cost-based optimizer depends on accurate statistics and how stale statistics lead to poor plans.
- Distinguish between blocking and non-blocking operators in the iterator model.
- Explain the roles of the buffer pool, sort area, and hash table area in database memory architecture.
- Calculate an approximate buffer pool size for a given workload and hardware configuration.
- Read basic PostgreSQL, MySQL, SQL Server, and Oracle monitoring queries to assess buffer pool health.

From Text to Results: The Query Pipeline

Every SQL query begins as a string of text. When you type `SELECT name, email FROM customers WHERE state = 'CA'` and press Enter, that text travels through a complex pipeline before any data is returned. Understanding this pipeline is the single most important step toward becoming an effective query optimizer, because every optimization technique either changes how the query enters the pipeline or influences a decision made somewhere along the way.

The query pipeline consists of five major phases, each with a distinct responsibility:

Phase 1: Parsing. The database reads your SQL text and breaks it into tokens: keywords like `SELECT` and `FROM`, identifiers like `customers` and `state`,

operators like =, and literals like `'CA'`. It builds a parse tree that represents the syntactic structure of your query. If there is a syntax error, the process stops here with an error message.

Phase 2: Semantic Analysis. The database checks that every table, column, and function referenced in your query actually exists. It resolves ambiguous column names when multiple tables are involved. It verifies that data types are compatible in comparisons and expressions. This phase catches errors like referencing a non-existent column or comparing a date to a string without proper casting.

Phase 3: Query Rewriting. Before optimization begins, the database may transform your query into an equivalent form that is easier to optimize. Common rewrites include converting `IN (subquery)` to a semi-join, flattening views into their underlying table references, and pushing filter conditions as deep into the query tree as possible. This phase is largely invisible to you but can dramatically affect what optimization choices are available.

Phase 4: Optimization. This is where the cost-based optimizer does its work. It considers multiple ways to execute your query, estimates the cost of each approach, and selects the cheapest one. The optimizer decides which indexes to use, in what order to join tables, which join algorithm to apply, and whether to sort or hash for aggregation. This phase produces the execution plan.

Phase 5: Execution. The execution engine walks the plan tree from leaf to root, performing each operation and passing rows upward. Data flows through the pipeline like water through a series of filters and pipes, with each operator consuming input rows, transforming them, and producing output rows for the next operator.

Consider what happens when you run this query against an e-commerce database:

```
SELECT c.name, o.order_date, o.total
FROM customers c
JOIN orders o ON c.customer_id = o.customer_id
WHERE o.total > 100
ORDER BY o.order_date DESC
LIMIT 20;
```

The parser breaks this into tokens and builds a tree showing a SELECT with columns from two tables, a JOIN condition, a WHERE filter, an ORDER

BY clause, and a LIMIT. The semantic analyzer confirms that `customers` and `orders` exist, that `customer_id` is a valid column in both, and that the comparison `o.total > 100` uses compatible types.

The rewriter might push the `WHERE o.total > 100` filter down to apply before the join, reducing the number of rows that need to be joined. The optimizer then considers several strategies: should it scan the `orders` table first and look up matching customers, or scan `customers` first and find matching orders? Should it use an index on `o.total` to filter orders, or scan the entire table? Once it selects a plan, the execution engine carries it out.

Parsing and Validation

Parsing is a mechanical process that converts text into structure. Most database systems use parser generators like Yacc or Bison to define the grammar rules for SQL. The output is an abstract syntax tree (AST) that captures the logical meaning of your query without committing to any particular execution strategy.

Consider these two queries:

```
-- Query A
SELECT * FROM customers WHERE id = 42;

-- Query B
SELECT name, email, created_at FROM customers WHERE id = 42;
```

Both produce nearly identical parse trees. The optimizer will likely generate the same execution plan for both if `id` is the primary key. But Query A fetches every column from the table, while Query B fetches only three. If the `customers` table has a wide text column like `bio` that stores thousands of characters, Query A may read significantly more data from disk than necessary, even though the execution plan looks the same.

This is why `SELECT *` is a performance anti-pattern in production systems. The parser and optimizer treat it as "select all columns," and while a covering index might satisfy a specific column list, it cannot cover "all columns" unless every column in the table is included in the index.

During semantic analysis, the database also resolves implicit type conversions. Consider this query:

```
SELECT * FROM orders WHERE order_id = '12345';
```

If `order_id` is defined as an INTEGER, the database must convert the string `'12345'` to an integer before comparison. In most cases, this conversion happens at parse time and has no performance impact. But if `order_id` were a VARCHAR column with an index, the implicit conversion of the literal would still allow the index to be used. The problem arises when the conversion applies to the column itself:

```
SELECT * FROM orders WHERE CAST(order_id AS TEXT) = '12345';
```

Now the database must convert every `order_id` value to text before comparing, which prevents the use of an index on `order_id`. This is one of the most common causes of unexpected full table scans.

The Query Optimizer: Cost-Based Decisions

The query optimizer is the brain of the database system. Its job is to find the cheapest way to execute your query, where "cheapest" means the lowest estimated cost in terms of CPU time, disk I/O, and memory usage. Modern optimizers are cost-based, meaning they use statistical information about your data to estimate how expensive each possible execution strategy will be.

The optimizer faces a combinatorial explosion. For a query joining N tables, there are N! possible join orders. For each join order, there are multiple join algorithms (nested loop, hash, merge). For each table access, there are potentially multiple indexes or a sequential scan. The search space grows exponentially with query complexity.

To manage this complexity, optimizers use several strategies:

Dynamic programming. PostgreSQL and MySQL use dynamic programming to explore join orders efficiently. They build up the cheapest plans for joining one table, then two tables, and so on, reusing previously computed results. This reduces the search space from exponential to polynomial in practice.

Greedy heuristics. SQL Server uses a combination of dynamic programming and greedy left-deep tree construction. It builds join trees by repeatedly

adding the cheapest next table, which limits the search space but can miss globally optimal plans for complex queries.

Rule-based shortcuts. All optimizers include rules that bypass cost analysis for obviously good choices. A primary key lookup is always chosen over a sequential scan for an equality condition on a small number of rows. These rules prevent the optimizer from wasting time considering absurd alternatives.

The quality of the optimizer's decisions depends entirely on the quality of its statistics. If the statistics say a column has 10,000 distinct values when it actually has only 3, the optimizer will make poor choices about index usage and join order. This is why keeping statistics up to date through regular ANALYZE (PostgreSQL), UPDATE STATISTICS (SQL Server), or ANALYZE TABLE (MySQL) operations is critical for consistent query performance. See Chapter 12 for monitoring approaches that detect stale statistics through row estimate discrepancies.

Consider this example with stale statistics. Suppose an `orders` table has a `status` column with values 'pending', 'shipped', and 'cancelled'. If 95% of rows have status 'shipped' and the statistics are current, the optimizer knows that `WHERE status = 'pending'` will return roughly 2.5% of rows. It will likely choose an index scan if one exists on `status`. But if the data distribution has changed dramatically since the last statistics update, and now 90% of orders are 'pending', the optimizer still thinks only 2.5% match and may choose a plan that works well for small result sets but performs terribly when returning most of the table.

Execution Engines and Memory Architecture

Once the optimizer produces a plan, the execution engine brings it to life. Modern databases use a volcano iterator model (also called the iterator model or pipelined execution) in which each operator in the plan is implemented as an object with three methods: `open()`, `next()`, and `close()`.

When `open()` is called on a root operator, it recursively calls `open()` on its children. When `next()` is called, the root operator requests a row from its child, which requests a row from its grandchild, and so on down to the leaf operators that read data from disk or indexes. Each operator processes one row at a time and passes it up the tree. This pipelined approach means that

data starts flowing through the plan as soon as the first rows are available, without waiting for the entire input to be materialized.

This architecture has important performance implications. Consider a query with a LIMIT clause:

```
SELECT * FROM orders ORDER BY created_at DESC LIMIT 10;
```

The Sort operator at the top of the plan needs to see all rows before it can produce the first sorted output. It is a blocking operator. But once sorting is complete, `next()` returns the first row, then the second, and after 10 rows have been returned to the client, execution stops. The Sort operator does not need to finish processing any remaining input because the LIMIT operator signals that no more rows are needed. This early termination saves significant work when the full result set is large but only a small portion is needed.

Not all operators are blocking. A Filter operator passes rows through as soon as they arrive, applying the WHERE condition to each one. A Nested Loop Join can produce output rows as soon as it finds the first matching pair. The distinction between blocking and non-blocking operators affects how memory is used and when results become available to the client.

Memory architecture. Database systems divide their memory into several regions, each serving a different purpose:

The **buffer pool** (called `shared_buffers` in PostgreSQL, `innodb_buffer_pool_size` in MySQL, and the buffer pool in SQL Server) caches data pages and index pages read from disk. When a query needs a page, the database first checks the buffer pool. If the page is already cached (a "hit"), it avoids a disk read entirely. Buffer pool hit ratios above 99% are typical for well-tuned systems. The size of the buffer pool is often the single most impactful configuration setting for query performance.

The **sort area** provides memory for Sort operators. When a query requires sorting more data than fits in the sort area, the database spills to temporary disk files, which dramatically increases execution time. In PostgreSQL, `work_mem` controls the memory available per sort operation. In MySQL, `sort_buffer_size` serves a similar purpose.

The **hash table area** provides memory for Hash Join and Hash Aggregate operators. Like sorting, hash operations spill to disk when they exceed their

memory allocation. PostgreSQL's `work_mem` also governs hash table size, while SQL Server uses memory grants that are calculated during compilation.

Understanding these memory regions is essential because many performance problems stem from insufficient memory allocation. A query that runs in 50 milliseconds with adequate `work_mem` may take 30 seconds when it spills to disk, and the difference shows clearly in the execution plan. Chapter 10 covers buffer pool sizing and memory tuning in depth, while Chapter 6 demonstrates how to detect sort spills in execution plans.

SQLite: A Different Architecture for Edge and Embedded Workloads

SQLite diverges significantly from the client-server architecture shared by PostgreSQL, MySQL, SQL Server, and Oracle. Understanding these differences is essential for applying the right optimization techniques to the right platform.

The Virtual Database Engine (VDBE). Instead of a volcano iterator model, SQLite compiles SQL queries into bytecode instructions executed by an internal virtual machine called the VDBE (Virtual Database Engine). The query planner generates a sequence of opcodes (like `OpenRead`, `IdxGT`, `Column`, `ResultRow`) that the VDBE interprets. This approach is analogous to how Java uses the JVM or Python uses bytecode: the SQL is compiled once into platform-independent instructions, then executed by a portable interpreter.

The VDBE model has performance implications. Because SQLite compiles to bytecode rather than generating native machine code, there is interpretation overhead per row processed. For simple queries on small datasets, this overhead is negligible. For complex analytical queries processing millions of rows, the bytecode interpretation can be measurably slower than the native-code execution of server-based databases. PostgreSQL 14+ partially addresses this gap with JIT (Just-In-Time) compilation, which compiles hot expressions to native machine code at runtime.

Single-process, file-based architecture. SQLite has no server process. The database engine is a C library linked directly into the application process. All data resides in a single file on disk (plus optional WAL and shared-memory files). This eliminates network latency, connection overhead, and the complexity of managing a separate database server. It also means there is no shared

buffer pool: each process connection maintains its own private page cache, sized via the `cache_size` PRAGMA.

```
-- SQLite: configure per-connection page cache (value in pages, default 2000)
PRAGMA cache_size = -64000;  -- Negative value means kilobytes: 64 MB cache

-- Compare with PostgreSQL shared_buffers (global, all connections share):
-- shared_buffers = 8GB  -- in postgresql.conf
```

The Pager and VFS layers. SQLite's storage architecture has two key layers beneath the B-tree engine:

- **Pager**: Manages page caching, transaction journals, and coordinates with the Virtual File System. The pager ensures ACID compliance by maintaining rollback journals or WAL files. It handles page-level locking and dirty-page writeback.
- **VFS (Virtual File System)**: An abstraction layer that isolates SQLite from operating system differences. Custom VFS implementations enable in-memory databases, encrypted storage, and even database files stored in RAM or on network shares. The default Unix VFS (`unix`) and Windows VFS (`win32`) implement OS-specific file operations.

Query planner characteristics. SQLite's query planner is simpler than those of server-based databases. It uses a cost model based on row estimates and index selectivity but lacks some advanced features:

- No dynamic programming for join ordering in older versions (SQLite 3.37+ improved this)
- Limited statistics collection compared to PostgreSQL's histogram-based approach
- No native support for parallel query execution
- Hash joins were added in SQLite 3.39.0 (2022), expanding available join strategies

When to choose SQLite. SQLite excels in scenarios where its architectural trade-offs are advantages:

- **Edge computing and IoT devices**: No server to manage, minimal memory footprint

- **Embedded applications**: Mobile apps, desktop software, browser storage (via SQL.js)
- **Read-heavy analytics on moderate datasets**: WAL mode with memory-mapped I/O delivers excellent read throughput
- **Development and testing**: Zero configuration, single-file portability

When SQLite is not suitable:

- High-concurrency write workloads: SQLite allows unlimited concurrent readers but only one writer at a time (even in WAL mode, the commit phase requires an exclusive lock)
- Multi-tenant OLTP systems requiring row-level locking and MVCC
- Datasets exceeding available RAM without effective caching strategies
- Applications requiring sophisticated query optimization for complex analytical workloads

SQLite performance tuning pragmas. SQLite's performance is tuned at the connection level via PRAGMA statements rather than server configuration files:

```
-- Essential pragmas for production read/write workloads
PRAGMA journal_mode = WAL;          -- Enable WAL for concurrent readers + writer
PRAGMA synchronous = NORMAL;        -- Safe durability with reduced fsync
↪  overhead
PRAGMA cache_size = -64000;         -- 64 MB page cache per connection
PRAGMA mmap_size = 268435456;       -- Allow 256 MB memory-mapped I/O
PRAGMA temp_store = MEMORY;         -- Store temporary tables in RAM
PRAGMA foreign_keys = ON;           -- Enforce referential integrity

-- For read-only workloads (e.g., analytics on a static copy)
PRAGMA query_only = ON;             -- Disables write locking entirely
PRAGMA synchronous = OFF;           -- No fsync calls at all
PRAGMA journal_mode = OFF;          -- No journal overhead
```

The `synchronous` PRAGMA controls durability versus performance. `FULL` (default in rollback-journal mode) ensures every commit waits for data to reach persistent storage. `NORMAL` is safe in WAL mode and significantly faster because it reduces fsync frequency. `OFF` risks data loss on crash but maximizes throughput for disposable or reproducible data.

SQLite vs server databases: architectural comparison:

Feature	SQLite	PostgreSQL	MySQL/InnoDB	SQL Server
Deployment model	Embedded library	Client-server	Client-server	Client-server
Concurrency model	File-level locking (1 writer)	MVCC with row-level locks	MVCC with row-level locks	MVCC/RCSI with row-level locks
Buffer pool	Per-connection page cache	Global shared_buffers	Global InnoDB buffer pool	Global buffer pool
Query compilation	VDBE bytecode	Native code execution	Native code execution	Native code + natively compiled procs
Statistics collection	Basic (table/index row counts)	Histograms, MCV, correlation	Histograms, cardinality	Histograms, density, CE feedback
Parallel query	Not supported	Yes (parallel workers)	Limited (InnoDB parallel reads)	Yes (parallel execution)
WAL mode	Supported (PRAGMA)	Built-in	InnoDB redo log	Write-Ahead Log
Maximum database size	140 TB (theoretical)	Unlimited (16 EB per table)	64 TB per table	524 PB

SQLite's embedded architecture makes it the most widely deployed database engine in the world, powering billions of devices from smartphones to spacecraft. But its single-writer limitation and simpler optimizer mean that the performance optimization techniques covered throughout this book apply differently. Index design and query writing matter just as much, but server-specific features like connection pooling, parallel query, and MVCC tuning are irrelevant.

Understanding Buffer Pools and Working Sets

The buffer pool is the database's primary defense against slow disk I/O. Every page read from disk is cached in the buffer pool, so subsequent reads of the same page are served from memory at microsecond latency instead of millisecond (or worse) disk latency. The effectiveness of this caching depends on the relationship between your working set and your buffer pool size.

The **working set** is the collection of pages that your typical query workload touches. If your working set fits entirely within the buffer pool, nearly every page read is a cache hit, and queries run at memory speed. If your working set exceeds the buffer pool, pages are constantly evicted and reloaded, causing a phenomenon called "buffer pool thrashing" that degrades performance toward disk speed.

Consider an e-commerce database with these tables:

Table	Rows	Page Size	Approximate Pages
customers	50,000	8 KB	625
orders	2,000,000	8 KB	25,000
order_items	8,000,000	8 KB	100,000
products	10,000	8 KB	125
Indexes	–	–	~30,000

The total database size is approximately 156,000 pages at 8 KB each, or about 1.2 GB. If the buffer pool is 4 GB, the entire working set fits with room to spare. Queries will enjoy high hit ratios and fast response times. But if the buffer pool is only 256 MB (about 32,000 pages), only a fraction of the data can be cached, and many queries will trigger disk reads.

The rule of thumb for sizing the buffer pool is to allocate 60-80% of available RAM on a dedicated database server. On a shared server, you must account for the operating system, other processes, and the database's own memory needs beyond the buffer pool. PostgreSQL recommends at least 25% of RAM for `shared_buffers` on systems where the OS page cache handles the rest. MySQL InnoDB should have `innodb_buffer_pool_size` set to 70-80% of total RAM on a dedicated server.

Buffer pool monitoring. All major databases provide metrics for tracking buffer pool effectiveness:

PostgreSQL exposes cache hit ratios through `pg_stat_database`:

```
SELECT datname,
       blks_hit,
       blks_read,
       round(100.0 * blks_hit / NULLIF(blks_hit + blks_read, 0), 2) AS hit_ratio
FROM pg_stat_database
WHERE datname = current_database();
```

MySQL provides InnoDB buffer pool statistics:

```
SHOW STATUS LIKE 'Innodb_buffer_pool_read%';
-- Hit ratio = read_requests / (read_requests + reads)
```

SQL Server exposes Buffer Manager counters through Performance Monitor or DMV queries:

```
SELECT cntr_value AS page_life_expectancy
FROM sys.dm_os_performance_counters
WHERE counter_name = 'Page life expectancy'
AND object_name LIKE '%Buffer Manager%';
```

A healthy buffer pool shows a hit ratio above 99% and a stable page life expectancy (the average time a page stays in the buffer pool before eviction). If the hit ratio drops below 95%, or if page life expectancy falls below 300 seconds in SQL Server, it is time to investigate whether the buffer pool needs to be enlarged or whether specific queries are polling the buffer with excessive sequential scans.

Chapter Summary

This chapter established the foundation for understanding query performance by examining how databases process SQL from text to results. The five-phase pipeline (parsing, semantic analysis, rewriting, optimization, execution) determines every aspect of query behavior. The cost-based optimizer makes decisions based on statistics about your data, and stale statistics lead to poor

plans. The iterator model pipelines data through operators, with blocking and non-blocking operators having different memory and timing characteristics. Memory architecture divides available RAM into specialized regions for caching, sorting, and hashing, with the buffer pool being the most impactful component.

Understanding these fundamentals enables you to reason about why a query performs the way it does. In the next chapter, you will learn to read execution plans across all major database platforms, turning abstract concepts into concrete diagnostic skills.

Hands-On Exercise

Create a simple `customers` table with 100,000 rows and run these queries:

```
-- Query 1: Primary key lookup
SELECT * FROM customers WHERE customer_id = 50000;

-- Query 2: Filter on non-indexed column
SELECT * FROM customers WHERE state = 'CA';

-- Query 3: Aggregation
SELECT state, COUNT(*) AS cnt, AVG(total_spent) AS avg_spent
FROM customers
GROUP BY state;
```

Run each query with EXPLAIN (or EXPLAIN ANALYZE in PostgreSQL) and observe the different plan structures. Note which operators appear, whether indexes are used, and how costs compare. We will revisit these queries in Chapter 2 with detailed plan analysis.

Common Pitfalls

- Assuming that a query that works correctly is also performant. Correctness and performance are independent dimensions.
- Ignoring the difference between estimated and actual execution statistics. Estimates guide the optimizer; actuals reveal what really happened.
- Forgetting that the first execution of a query may be slower due to cold cache. Always warm up the buffer pool before measuring.
- Overlooking the impact of `SELECT *` on covering index usage and network transfer overhead.

Interview Questions

1. Describe the five phases of query processing in a relational database.
2. What is the difference between a blocking and non-blocking operator? Give examples of each.
3. Why does stale statistics cause bad query plans, and how do you fix it?
4. How would you size the buffer pool for a 16 GB dedicated PostgreSQL server?
5. What is the volcano iterator model, and why is it used for query execution?

Chapter 2: Reading Execution Plans

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

What an Execution Plan Tells You

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

PostgreSQL EXPLAIN and EXPLAIN ANALYZE

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

MySQL EXPLAIN Output Decoded

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

SQL Server Execution Plans

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Oracle Explain Plan Fundamentals

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

SQLite EXPLAIN and Query Plan Reading

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Common Patterns and Red Flags

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Hands-On Exercise

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Common Pitfalls

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Interview Questions

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Chapter Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Chapter 3: Indexing Strategies

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

How B-Tree Indexes Work Internally

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

When Indexes Help (and When They Hurt)

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Composite and Covering Indexes

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Specialized Index Types: Hash, GiST, GIN, BRIN

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Index Maintenance and Fragmentation

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Cross-Platform Index Comparisons

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Hands-On Exercise

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Common Pitfalls

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Interview Questions

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Chapter Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Chapter 4: Query Patterns and Anti-Patterns

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

SELECT * and Column Selection Discipline

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

The LEFT JOIN Trap

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Correlated Subqueries vs. EXISTS

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

OR Conditions and Index Inefficiency

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Functions on Indexed Columns

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Implicit Type Conversion Pitfalls

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

The N+1 Query Pattern

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Pagination Anti-Patterns

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Side-by-Side Comparison Table

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Hands-On Exercise

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Common Pitfalls

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Interview Questions

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Chapter Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Chapter 5: Joins and Their Performance

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Nested Loop Joins: When Small Meets Big

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Hash Joins for Large Data Sets

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Merge Joins and Sorted Inputs

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Join Order Optimization

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Self-Joins and Multi-Table Complexities

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Cross Joins and Cartesian Product Dangers

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Advanced Join Topics: Batch Index Probe and Vectorized Execution

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Join Algorithm Selection Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Hands-On Exercise

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Common Pitfalls

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Interview Questions

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Chapter Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Chapter 6: Aggregation, Sorting, and Filtering

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

GROUP BY Implementation Strategies

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Window Functions vs. Self-Joins

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

ORDER BY and Sort Spill

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Filtering Early: Pushdown Optimization

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

DISTINCT and Deduplication Costs

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Partial Aggregations and Precomputation

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Hands-On Exercise

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Common Pitfalls

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Interview Questions

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Chapter Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Chapter 7: Subqueries, CTEs, and Derived Tables

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Correlated vs. Non-Correlated Subqueries

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Common Table Expressions: Readability vs. Performance

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Materialized vs. Inlined CTEs

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Lateral Joins and Cross Apply

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Temporal Tables and Time-Series Queries

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Query Refactoring Patterns

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Hands-On Exercise

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Common Pitfalls

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Interview Questions

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Chapter Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Chapter 8: Schema Design for Performance

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Normalization vs. Denormalization Trade-offs

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Data Modeling Fundamentals: ER Design and Dimensional Models

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Data Types and Their Storage Cost

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Partitioning Strategies and Query Routing

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Table Inheritance and Sharding

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Columnar vs. Row-Oriented Storage

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Schema Evolution and Migration Strategies

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Hands-On Exercise

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Common Pitfalls

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Interview Questions

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Chapter Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Chapter 9: Transactions, Locking, and Concurrency

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Isolation Levels and Their Performance Cost

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Row-Level vs. Table-Level Locking

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Deadlock Detection and Prevention

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

MVCC and Snapshot Isolation

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Connection Pooling and Resource Limits

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Write Contention and Hot Rows

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

SQLite Locking Model: File-Level Locks and WAL Mode

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Hands-On Exercise

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Common Pitfalls

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Interview Questions

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Chapter Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Chapter 10: Memory, Caching, and I/O Optimization

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Buffer Pool Sizing and Hit Ratios

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Query Result Caching Strategies

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Sequential vs. Random I/O Patterns

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

SSD Impact on Database Performance

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Temp Tables and Disk Spill

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

OS-Level I/O Interactions: Linux Readahead, fsync, and NUMA

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Memory-Optimized Tables

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Hands-On Exercise

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Common Pitfalls

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Interview Questions

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

SQLite I/O Architecture and Performance Tuning

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Chapter Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Hands-On Exercise

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Chapter 11: Advanced Optimization Techniques

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Query Rewriting and Transformation

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Materialized Views and Refresh Strategies

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Partition Pruning and Predicate Pushdown

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Adaptive Query Processing

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Cardinality Estimation: The Math Behind Plan Quality

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Database-Specific Tuning Features

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Custom Aggregates and Extensions

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Hands-On Exercise

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Common Pitfalls

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Interview Questions

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Chapter Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Chapter 12: Monitoring, Troubleshooting, and Benchmarking

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Building a Performance Baseline

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Slow Query Log Analysis

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Real-Time Monitoring Dashboards

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Load Testing and Benchmark Design

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Production Incident Response

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Performance Regression Prevention

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Hands-On Exercise

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Common Pitfalls

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Interview Questions

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Chapter Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Chapter 13: Case Studies in Performance Optimization

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Case Study 1: E-Commerce Search and Cart Performance

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Analytics Dashboard Timeout Crisis

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Case Study 3: Low-Latency Time-Series Data Access

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Case Study 4: Multi-Tenant SaaS Database Isolation

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Case Study 5: Database Migration Performance Parity

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Lessons Learned Across Industries

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Hands-On Exercise

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Common Pitfalls

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Interview Questions

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Chapter Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Chapter 14: Production Best Practices and Checklists

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

The Performance Review Checklist

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

CI/CD Integration for Query Validation

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Capacity Planning and Growth Patterns

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Database Migration Performance Considerations

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Team Training and Knowledge Sharing

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Staying Current: What to Watch For

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Performance Optimization Cheat Sheet

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Hands-On Exercise

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Common Pitfalls

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Interview Questions

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Chapter Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Conclusion: The Optimizer's Mindset

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

References

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Glossary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Appendix A: Sample Database Schema Evolution

Phase 1: Base Schema (Chapters 1-5)

Phase 2: Indexing Layer (Chapter 3)

Phase 3: Partitioning (Chapter 8)

Phase 4: Materialized Views and Aggregates (Chapters 6, 11)

Phase 5: Dimensional Model (Chapter 8 data modeling)

Data Generation Script (PostgreSQL)

Appendix B: Platform-Specific Configuration Templates

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

PostgreSQL 16 Performance Configuration

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

MySQL 8.0 InnoDB Configuration

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

SQL Server Configuration Guidelines

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

PgBouncer Configuration

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Appendix C: Exercise Solutions

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Chapter 1 Exercise Solution

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Chapter 2 Exercise Solution

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Chapter 3 Exercise Solution

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Chapter 4 Exercise Solution

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Chapter 5 Exercise Solution

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Chapter 6 Exercise Solution

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Chapter 7 Exercise Solution

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Chapter 8 Exercise Solution

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Chapter 9 Exercise Solution

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Chapter 10 Exercise Solution

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Chapter 11 Exercise Solution

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Chapter 12 Exercise Solution

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Chapter 13 Exercise Solution

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Chapter 14 Exercise Solution

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.

Index

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/masteringsqlperformanceoptimization.