SQL Optimization Guide

Data
AI Consultant

February 26, 2025

1 Introduction

SQL optimization is crucial for improving database performance and reducing query execution time. This guide will cover indexing strategies, query performance tips, common mistakes, and real-world best practices to help you write efficient SQL queries.

2 Indexing and Schema Optimization

2.1 Understanding Indexes

Indexes help speed up data retrieval but can slow down write operations. Use them wisely.

2.1.1 Clustered vs Non-clustered Indexes

- **Clustered Index**: Sorts and stores data rows in order.
- **Non-clustered Index**: Separate structure storing pointers to data.

2.1.2 Composite Indexes

- Combine multiple columns into one index to improve WHERE clause performance.
- Avoid unnecessary composite indexes as they increase storage overhead.

2.1.3 Avoiding Over-indexing

- Too many indexes slow down write operations.
- Monitor index usage with database performance tools.

3 Query Performance Optimization

3.1 Using EXPLAIN PLAN and ANALYZE

SQL execution plans help understand how queries are executed.

Listing 1: Checking Query Performance

```
EXPLAIN ANALYZE SELECT * FROM orders WHERE order_date > '2023-01-01';
```

Key Metrics:

- **Seq Scan** (bad) - Full table scan
- **Index Scan** (good) - Uses an index for retrieval

3.2 Joins and Subqueries Optimization

INNER JOIN vs LEFT JOIN Performance

```
SELECT customers.name, orders.total
FROM customers
INNER JOIN orders ON customers.id = orders.customer_id;
```

Avoiding Nested Subqueries

```
-- Inefficient Query
SELECT name FROM customers WHERE id IN (SELECT customer_id FROM orders);

-- Optimized Query (Using JOIN)
SELECT DISTINCT customers.name
FROM customers
INNER JOIN orders ON customers.id = orders.customer_id;
```

4 Common Mistakes and Fixes

4.1 Avoiding SELECT *

```
-- Bad Query
SELECT * FROM users;

-- Optimized Query
SELECT id, name, email FROM users;
```

4.2 Reducing I/O Load

```
INSERT INTO orders (id, total) VALUES (1, 100), (2, 200), (3, 300);
```

5 Real-World SQL Performance Tuning

5.1 Speeding Up a Slow Query

```
-- Slow Query
SELECT * FROM orders WHERE order_date > '2023-01-01';

-- Optimized Query (Using Index)
CREATE INDEX idx_order_date ON orders(order_date);
SELECT * FROM orders WHERE order_date > '2023-01-01';
```

Improvement: Reduced execution time from 3 sec → 0.02 sec

6 SQL Optimization Checklist

- Use Indexes Wisely
- Avoid SELECT *
- Analyze Query Execution Plans
- Optimize Joins
- Reduce I/O Operations
- Use Proper Data Types
- Batch Insert Data
- Monitor Query Performance Regularly