The Negative Impact of Anti-SQL Patterns on Query Performance

By Bert Scalzo, CTO & Chief Architect, QIKR
February 10, 2026

SQL anti-patterns are common bad or inefficient coding habits. While they *may* produce correct results, they lead to severe performance degradation, high resource consumption, and system instability**.** Some common patterns limiting query performance are:

Retrieval & Selection Anti-Patterns

- **Using SELECT *****: This is the most common anti-pattern. It retrieves every column, including unneeded data, which increases network traffic, consumes excess memory, and prevents the use of "covering indexes" that could otherwise satisfy the query without reading from the table heap.
- **Missing LIMIT Clauses**: On large datasets, fetching thousands of rows when only a few are needed wastes CPU and memory.
- **Leading Wildcards (%abc)**: Using a wildcard at the start of a LIKE string prevents the database from using an index. This forces a slow full table scan because the engine cannot determine a starting point in the index
- **Trailing Wildcards (abc%)**

Logical & Functional Anti-Patterns (Non-SARGable)

- **Functions on Indexed Columns**: Applying functions like UPPER(column) or DATEDIFF(column) in a WHERE clause makes a query "non-SARGable" (not Search ARGument Able). The database must evaluate the function for every row, disabling index seeks.
- **Implicit Type Conversions**: Comparing different data types (e.g., a string to an integer) forces the database to convert every row before comparison, negating index benefits.
- **Mishandling NULLs**: Using standard operators like = or != with NULL (e.g., WHERE col = NULL) always returns false or unknown. Use IS NULL or IS NOT NULL instead.

Structural & Complexity Anti-Patterns

The N+1 Query Problem: Making one query for a parent record and then additional queries for its children instead of using a single JOIN. This creates massive overhead from repeated round-trips to the database.

- **Overusing DISTINCT**: This is often used as a "quick fix" for duplicates caused by bad join logic. DISTINCT is resource-heavy because it requires the database to sort or hash the entire result set.
- **Correlated Subqueries**: These run once for every row in the outer query, which is extremely slow on large datasets. Replacing them with JOIN operations or Common Table Expressions (CTEs) is usually more efficient.
- **Deeply Nested Views**: Layering views upon views creates complex dependency chains that are difficult for the query optimizer to evaluate, often leading to poor execution plans.

Design Anti-Patterns

- **Storing Comma-Separated Values**: Storing multiple values in one column (e.g., hobbies: "reading, swimming") breaks normalization and makes indexing impossible.
- **Tables without Primary Keys**: This prevents efficient row identification and can lead to duplicate data, significantly slowing down joins and updates

Summary

Avoiding anti-SQL patterns is vitally important and the path to better query performance. QIKR is 100% focused on query performance optimization and created HyperQuery™ to fix – and eliminate – the common problems, the "garbage in/garbage out" root cause problems, of poor query performance, namely, badly coded SQL and inefficiently written, poorly executing queries.

HyperQuery is powered by ASTRO™ (**A**utonomic **S**QL **T**ransformation and Query **R**ewrite **O**ptimization), our 100% self-contained (no outside LLM calls) Agentic AI engine comprised of 100s of analysis, inferencing, deduction, pattern matching, forward chaining, logic reduction algorithms that performs all the heavy lifting to automate away the complexities of manual tuning for performance optimization. With simple 1-click optimization, HyperQuery delivers instant, accurate, and trusted query rewrites (anti-SQL pattern conversions) that are guaranteed returned in the exact same order/# of rows and verifiable via mathematical proofs with:

1. syntactically correct PerfectSQL
2. best cost-optimized execution plan decisioning efficiency
3. 100s to 1,000s X better performance

Best of all, HyperQuery delivers this friction-free/risk-free autonomically – no touch, no schema needed, no database changes, no manual tuning, no hardware required.

Simpler…better…faster…no other solution optimizes query performance more accurately, more efficiently, or more cost-effectively than HyperQuery. Learn more at www.hyperquery.net

About the Author

Bert Scalzo is a recognized database expert. An Oracle ACE and author of a dozen books on database technology, Bert possesses decades-long expertise in SQL, database internals, tuning & performance optimization, and automation. Prior to QIKR, at Quest Software, Bert was the co-architect/co-developer of TOAD and earlier, at other database and tools vendors, helped architect and develop a dozen of the leading database tools used daily by enterprise IT teams globally.