Executive Summary

During development of a new mobile application integrating with a CMMS platform, initial stored procedures for paginated data loading presented critical performance and maintainability issues:

Problem #1 – Performance Crisis

Inconsistent and unacceptable execution times averaging 27 seconds per page load, creating a poor user experience and system bottlenecks.

Problem #2 – Maintenance Nightmare

The original approach required building individual stored procedures for each data list, resulting in 350+ procedures across modules (Assets, Work Orders, Parts, Repair Centers, Departments, Classifications, etc.).

As Database Administrator and Lead SQL Developer, I was tasked with architecting a comprehensive solution.

Solution Architecture

Strategic Approach

Instead of managing hundreds of individual procedures, I designed a single, universal stored procedure with parameterized inputs to handle all CMMS modules. This approach prioritized:

- Development efficiency and consistency
- Simplified maintenance with schema changes
- Centralized optimization and performance tuning
- Reduced database storage footprint

Technical Evolution

My optimization progressed through two iterations

1. **Initial Optimization**: Recursive CTE implementation reduced execution time to ~250ms—acceptable, but not optimal.
2. **Final Architecture**: Three-layered CTE design with zero recursion achieved consistent 17ms average execution times across all table sizes.

Performance Results

- **Before**: 27,000ms average execution time
- **After**: 17ms average execution time
- **Improvement**: 99.94% performance gain
- **Consistency**: Execution time independent of table size
- **Maintainability:** 350+ procedures reduced to 1 universal solution

Technical Implementation Notes

Dynamic SQL Approach

The parameterized design necessitated dynamically generated SQL. Two common concerns addressed:

Query Plan Recompilation

While dynamic SQL requires new query plans per execution rather than cached plans, the compilation overhead proved negligible compared to the massive performance gains from tailored query execution.

SQL Injection Security

This implementation maintains security through controlled parameter handling and exclusive use by internal applications with no direct user input exposure.

The solution demonstrates how strategic architectural decisions can transform both performance and maintainability in enterprise database systems.

Full SQL is available in text file format at: GetPagedData