Executive Summary

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

Problem #1 – Performance Crisis

Execution times averaging an unacceptable 27 seconds per page load, which threatened the product release timeline.

Problem #2 – Maintenance Nightmare

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

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

1. **Initial Optimization**: Recursive CTE implementation reduced execution time to ~250ms—acceptable, but not optimal.
2. **Final Architecture**: Three-layered CTE design without recursion achieved consistent 17ms average execution times across all table sizes.

Performance Results

- **Before**: 27 second average execution time
- **After**: 17 millisecond average execution time
- **Improvement**: 99.94% performance gain
- **Consistency**: Execution time independent of table size
- **Maintainability**: Approximately 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 query plan compilation on each execution rather than cached plans, the compilation overhead proved negligible compared to the massive performance gains.
- **SQL Injection Security** – This implementation maintains security through controlled parameter handling and exclusive use by internal applications with no direct user input exposure.

Why Not Fetch/Offset?

Using offset and fetch would have been simpler, but compatibility with SQL Server 2008 had to be maintained.

Full SQL is available in text file format at: GetPagedData