PowerBuilder Performance Review

Client: Recruitment Software House
Business Size: SME
Industry: Recruitment
Country: Netherlands
Technology: PowerBuilder 2022, SQL Server 2019

Objective: Review slow performance of current Application limiting change to Architecture

Background

The Client, a software house, has developed a PowerBuilder application over the past 25 years. It is currently utilising PowerBuilder 2022 and SQL Server 2019. They've been experiencing significant performance issues, particularly when using the application's search functionality. The application follows a layered architecture, including Presentation, Business, Persistence and Database Layers. Due to inefficiencies in design and execution, performance bottlenecks were observed in response times, CPU utilisation, memory consumption and database queries.

Methodology

To assess and address the client's performance issues, a comprehensive analysis was conducted on their PowerBuilder 2022 and SQL Server 2019 applications, identifying key bottlenecks and areas for improvement. These findings were used to develop a set of actionable recommendations, which served as a roadmap for performance optimisation, ensuring better application stability, efficiency and user experience. Future implementation of these optimisations could lead to significant improvements in the system responsiveness and operational effectiveness.

Consultant Contribution

Following the performance review, the consultant delivered a comprehensive set of technical recommendations to address the key issues identified in applications:

Optimisation of Layered Architecture

- Recommended the consolidation of the Business and Persistent Layers to streamline processing.
- Proposed the migration of SQL queries to stored procedures and functions to enhance execution efficiency.

Reduction of Front-End Overhead

- Conducted code mining to remove redundant PowerScript logic and unnecessary visual controls.
- Used PowerBuilder's native event handling to optimise function execution.

Database Optimisation

- Suggested converting frequently used views into indexed tables where applicable.
- Identified and reduced unnecessary joins and subqueries in critical queries.

SQL Query Optimisation

- Recommended replacing redundant SELECT DISTINCT statements with well-structured queries.
- Introduced the use of NOLOCK directives to reduce unnecessary table locking.
- Proposed indexing enhancements to improve query execution times.

Enhancement of Presentation Layer Performance

- Suggested restricting the number of concurrent search windows.
- Recommended implementing pagination for large result sets to improve efficiency.
- Introduced the use of UI skeletons to provide visual feedback during data retrieval.

Challenges

The analysis highlighted several key performance concerns:

Layered Architecture Overhead (Critical)

- The application's layered architecture increased the number of interactions between layers, leading to slow processing times.
- Shared resource environments exacerbated these issues due to inefficient handling of system dependencies.

Heavy Front-End (Critical)

- The PowerBuilder front-end consisted of over 650,000 lines of code, leading to excessive execution time.
- Large numbers of visual and non-visual controls contributed to performance degradation.

Use of Views in Database Queries (Severe)

- The application relied heavily on database views instead of direct table queries, causing additional processing overhead.
- Analysis found 294 views were in use, with many capable of being optimised into table structures.

Inefficient SQL Queries (Severe)

- Widespread use of SELECT DISTINCT statements slowed down the query execution, especially in large datasets.
- The lack of NOLOCK directives resulted in unnecessary table locking and reduced efficiency.

Presentation Layer Design Flaws (Medium)

- Unlimited instances of search windows lead to excessive memory consumption and orphaned database connections.
- Retrieving large result sets without pagination overloaded the database and network, reducing responsiveness.
- No user feedback during data retrieval created a poor user experience.