precisely

Case Study:

Automaker Discovers Not All DB2 Optimizers the Same

Precisely Optimize DB2 Delivers After Extensive Trials

Critical Issue

First came the emergency. This company, a prestige auto manufacturer, had implemented a new distributed web system for its dealers. Unexpectedly, the system generated huge workloads for the company's Db2 database, and soon the Db2 CPU usage was enough to cause the entire CPU to grossly exceed capacity, forcing a quick and costly CPU upgrade.

Given that experience, the company's IT management went looking for a better way to track and tune its database processing. They found solutions for tracking and tuning batch and CICS jobs, but managing the web processes and Db2 enclaves presented bigger challenges. In particular, none of the SQL tuning products they tried was able adequately to manage the very large number of identical SQL statements.

Solution

After a thorough vendor and solution search they found Optimize DB2, from Precisely.

In testing with Optimize DB2, it was discovered that the SQL statements that were causing the performance problems were not, as suspected, the most complex ones. They were instead some of the very frequently executed SQL even though each one individually used relatively small amounts of CPU.

Optimize DB2 enabled them to consolidate identical SQL statements, so that instead of a million separate lines of the same SQL statement, they had one line showing a single SQL statement being executed a million times.

Optimize DB2 also showed the total of CPU time used by the SQL statement, as well as the average time. Optimize DB2 was an easy choice for them just based on their immediate need, and they indicated interest in other components of the Optimize DB2 suite for solving longer-term problems as well.

Results

Following Optimize DB2 implementation, the company counted the following as the most notable benefits:

- A "very large amount of money" saved in continuing operations, plus avoidance of another CPU upgrade.
- Reporting by program and by AuthID (each web app uses a common AuthID).
- Tracking of changes over time, so they can look back two weeks or six months to see how a particular SQL performed then versus now; this surfaces problems caused by system changes.
- A stats copy feature that populates the Db2 catalog with stats from production, ensuring that test access paths accurately reflect production.
- A system change feature (Impact Analyzer) allows them to see what access path changes occur when they make changes to an environment, such as Db2 code fixes. Eliminates "those nasty surprises" when implementing new Db2 maintenance or from rebinds.