IBM DATABASE MAGAZINE » Powering business information

Quarter 1, 2009 Vol. 14, Issue 1

http://www.dbmag.intelligententerprise.com/story/showArticle.jhtml?articleID=216300333

# IDUG User Views: Three Tips for DB2 Java Apps

By David Beulke

## Keep DB2 performance high by avoiding these common Java oversights.

Even though COBOL application development continues to grow, Java is the dominant development language used in applications that access DB2 for Linux, Unix, and Windows as well as for z/OS today. Most veteran DBAs already know how to debug COBOL applications. These performance tips will help DBAs and application developers continue to ensure good DB2 performance with Java applications, too.

**Always check DB2 SQL return codes.** This advice may seem basic, but you'd be surprised how often SQL return checking is overlooked. At a recent performance review I worked on, developers executed a transaction while the database was offline (we were refreshing and reloading test data). They were amazed by the performance — until I pointed out that the Java methods didn't check any `SQLCODE` or `SQLSTATE` return codes for the SQL statements. Because the Java methods were executing the SQL through a poorly designed home-grown persistence framework, data was returned even though the SQL didn't execute. Underlying the non-existent SQL return code checking is another problem: The developers didn't understand how to correctly connect to and access data from the seven-year-old Unix legacy system. And that leads me to my next tip.

**Go directly to the database to avoid persistence frameworks.** Use SQL statements to go directly to the database instead of relying on the Java execution environment. Why? At run time, some Java persistence frameworks, like Hibernate, will change your SQL into poorly performing dynamic SQL statements. You end up thinking your Java application is executing one SQL statement when it is really doing something completely different. As a result, you could spend endless hours trying to find these poorly

performing rewritten SQL statements, because they're found only in a DB2 monitor — they don't exist in any written application code. If you must use any of these old methods, verify and *capture* the SQL and the access path during your testing. Make sure that the SQL is programmed efficiently, that it is retrieving the minimum amount of data necessary, and that its scope is only for the current transaction.

**Understand your transaction service scope.** Many IT departments are now using service-oriented architecture (SOA) to enable integration and to reduce costs. Web services rely on many Java methods or classes to complete a transaction. Within each of the various services, dynamic SQL may be executed against the database. Each method could be connecting to the database and inserting or updating data, then moving on to execute the next service. Database transaction commit and rollback integrity can become an issue if there's no understanding of what transaction calls which service. If you understand the transaction scope, you can consolidate the database connection operations. Executing the connection once for all database SQL services avoids expensive repeat connections and saves tremendous amounts of CPU for more efficient transaction completion.

Java programming continues to evolve. Keep up with state-of-the-art design and application techniques by attending an International DB2 Users Group conference (find one at idug.org), where DB2 users go to tell what really happens in their environments. Stop by my presentation or just say hello if you see me there.

---

***David Beulke*** *is former president of IDUG and has more than 22 years experience in architecture, design, and development of high performance DB2 systems across all platforms.*

Return to Article