DATA BASED DOWNSIZING: A Case Study that Exposes the Pitfalls!

DATA BASED ADVISOR®

For Developing Desktop Database Applications

March 1992 $3.95 Canada $4.95

SPEED UP THAT APP!

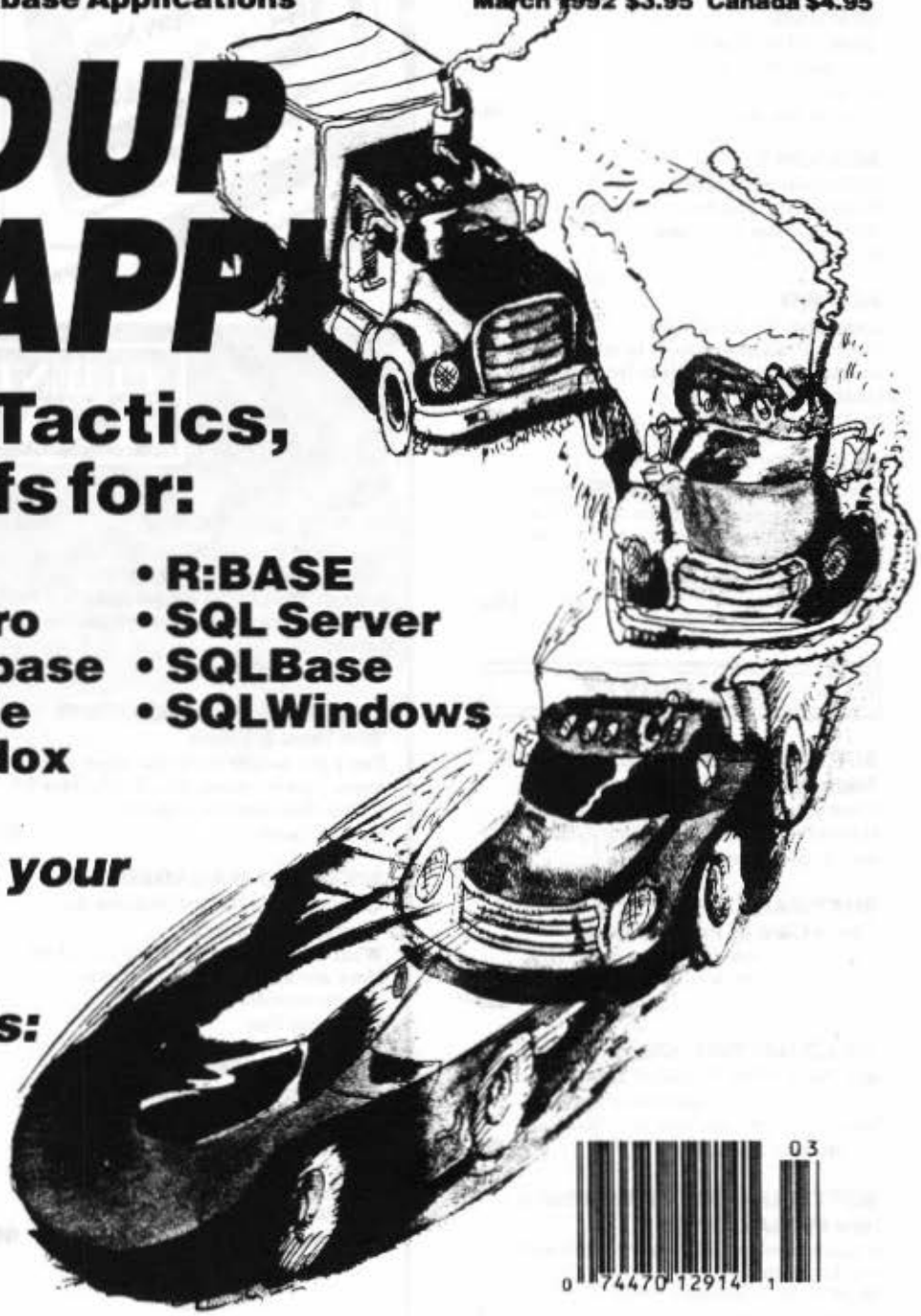
Tuning Tips, Tactics, and Trade-offs for:

- AREV
- Clarion
- Clipper
- DataEase
- dBASE IV
- EB
- FoxPro
- Interbase
- Oracle
- Paradox
- R:BASE
- SQL Server
- SQLBase
- SQLWindows

Plans for growing your VAR business

Nantucket's RDDs: why they're so late...

(see page 134)



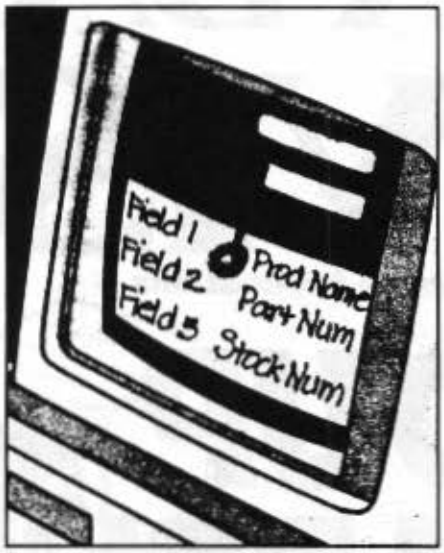
03

0 74470 12914 1

Vol. 10, No. 3 March 1992

THE EXPERT SERIES

TEST DRIVES



Page 58

DEPARTMENTS

CLIENT/SERVER ADVISOR

TUNING TIPS

Oracle Server

By David McGoveran

Relational database application tuning has a number of dimensions, each of which needs to be understood and exploited to achieve the desired effect. There are many possible goals in tuning a database server: performance, concurrency, memory usage, storage allocation, etc. Each of these involve trade-offs. For example, it may be possible to achieve some levels of performance only at the expense of concurrency or some levels of concurrency only at the expense of database integrity guarantees.

Oracle Server for OS/2 offers a wide variety of tuning opportunities and a number of facilities for the purpose. Tuning can be performed at various levels of "granularity" for a particular SQL statement, a critical module, a particular application, some portion of the database, or globally (as for throughput, storage demands, etc.). As a result, tuning Oracle Server requires planning and experience. Here are a few tips for doing that planning and gaining that experience, most of which could be applied to any database server with some minor changes.

➤ **1.** Throw as much memory and as many disk drives at the problem as you can afford. Oracle Server likes resources, and essentially uses all that are available. On the other hand, don't expect too much. Oracle Server for OS/2 will support about 10 concurrent users at high performance in practice (despite TPC benchmark numbers) before something gives.

➤ **2.** Try to reduce the granularity of the tuning problem. Profilers can be used to determine critical code segments and point to high cost database operations. Oracle's tuning facilities can help identify specific statements that are performing poorly or determine if additional memory would help.

➤ **3.** Plan for and develop tuning metrics, both within the application and independent of it. The first can be done with special routines within the application that are conditionally compiled or that are "enabled" via a command line argument or a special command file opened when the application is invoked. The second can generally be done with SQL*Plus scripts.

➤ **4.** Prioritize the goals, and reduce the granularity of the problem as much as possible. Don't get overly involved in the details if what you need is system throughput. And don't spend time optimizing every SQL statement if one statement is consuming all the resources. Think globally, act locally.

➤ **5.** Study the *Database Administrator's Guide* and the *Advanced Performance Tuning Guide* to become familiar with the various tuning facilities and techniques. Read between the lines. Like all vendors, Oracle isn't eager to state that certain functionality doesn't perform well. Think through each suggestion, and question its implications. The more you understand about how Oracle works internally, the better you can tune Oracle Server. Also, remember that even the best Oracle expert may not understand your specific application requirements. You're probably best qualified to determine if advice is appropriate or how it needs to be interpreted. Among the facilities you should be familiar with are the following tables: V$ROWCACHE, V$WAITSTAT, X$KCBCBH, X$KCBRBH, and PLAN_TABLE. Also be familiar with SQL*DBA, the SQL trace facility, TKPROF, and EXPLAIN PLAN. In addition, familiarity with the system catalog and INIT.ORA parameters is essential.

➤ **6.** When tuning SQL statements, remember that phrasing is critical to Oracle Server. Unlike most RDBMS optimizers, Oracle's optimizer is syntax-sensitive. It examines table order in the FROM clause to determine the order of processing, and doesn't use a beneficial index on a column or columns unless they're explicitly referenced in the WHERE clause. The column must not be used inside a function. Because the optimizer doesn't decide between alternative methods of processing a SQL statement, it may use an index that it shouldn't. In this case, the index can be disabled by preventing the optimizer from recognizing a column of the index by concatenating a NULL or a space to a character column or adding zero to a numerical column.

➤ **7.** If a critical request needs access to one or two more columns than are in a useful index, try adding these to the index. During reads, only the index need be accessed. This is especially beneficial if indexed access results in a large amount of random data block I/O.

➤ **8.** Avoid the use of NULL wherever possible. NULLs introduce anomalies in SQL, and make it hard to interpret results. They usually disable the use of a otherwise beneficial index.

➤ **9.** Denormalization may be necessary to reduce the number of tables in common joins. Oracle doesn't respond well to more than five tables in a single SQL statement. However, you might be able to avoid denormalization by creating intermediate results in temporary tables and subsequently joining these intermediate results.

➤ **10.** Distribute data. Place rollback segments and redo log files on separate disk drives to reduce drive contention. Proper sizing can reduce storage sage and dynamic extension of storage space. The number of rollback segments is important to concurrency.

➤ **11.** Tune the INIT.ORA parameters (DB_BLOCK_WRITE_BATCH and DB_BLOCK_MAX_SCAN_CNT) to improve multi-block writes. You can also tune DB_FILE_MULTIBLOCK_READ_COUNT to improve multi-block reads. Some of the other important parameters to consider include the integrity/concurrency trade-off (with ROW_LOCKING and SERIALIZABLE), the number of ROLLBACK_SEGMENTS, and statistics gathering parameters (avoid these if performance is important).

➤ **12.** Look out for "deadlocks," especially if you aren't using the transaction processing option. Oracle treats any process waiting for a resource as though it is possibly involved in a deadly embrace. Resource waits can be used to identify high contention resources. Requests for high contention resources should receive special tuning attention. If this doesn't work, try staggering requests for these resources, further nor-

malization, or even duplicating the resource (assuming you can do so without endangering database integrity).

➤ **13.** Use SEQUENCE generators to eliminate application processing and therefore unnecessary requests to the database server. Note that you must trade-off between speed due to caching of sequence numbers and the fact that entries in cache are lost in an instance failure.

➤ **14.** Tune the System Global Area (SGA). The SGA should fit into real memory, but the larger the buffer cache and data dictionary cache, the less disk I/O.

Editor's Note: The material in this article was adapted from a chapter on performance tuning in the author's book, *An Advanced Guide to Client/Server Applications*, in preparation.

David McGoveran is president of Alternative Technologies in Santa Cruz, CA. The firm specializes in relational databases. David can be reached at (408) 425-1859. ❐