

The Performance and Cost Optimization Ripple Effect *and Hidden Costs that are damaging your profitability*

Your database IT Infrastructure is bloated. You have overpaid at least twice as much as you needed to for hardware and software licenses, and your ongoing energy and data center costs are at least 30% too high as a consequence. Further, you are at risk of perpetuating these costly mistakes - unless you understand and follow the advice offered by this article. The good news is that performance and cost optimization solutions are available with payback periods as short as three months.

For over 20 years, I have been writing, teaching, speaking, and consulting on topics of DB2 performance, security, auditing, and best practices. As we say in Texas, "I'm going to shoot you straight." We're not here to make friends with hardware or software vendors that sell CPUs or PVU licenses; in fact, they loathe us for we've helped cancel over \$50M of unnecessary upgrades during recent years.

Database vendors are making increasing claims about automated tuning, but proper physical design is still critically important. By physical design, we ordinarily consider index definitions first and then give consideration to more advanced techniques.

Let's assume you have a bowl of blueberries in your refrigerator and you want to use them in a recipe. You get your favorite 592 page cookbook out to find recipes that use blueberries. You can flip through every page of the cookbook and scan ingredients lists looking for blueberries. If you are a speed reader and only spend two seconds on every page, it will take you almost twenty minutes to search your cookbook for blueberry recipes. In contrast, if the cookbook has a good index in the back of the book, you could look up blueberries and discover recipes on pages 88, 92, 400, and 502. The index lookup takes five seconds, and jumping to the four different pages to review the recipes might take another 5 seconds for each lookup. Which method is more efficient and a better use of your valuable time?

If the cookbook does not provide an index, or an index with sufficiently detailed entries to help you find blueberries, then you have only one choice - scan the entire cookbook. Scans are costly. They take time, energy, and might even make your eyes hurt.

So let's now imagine that we put your 592 page cookbook into a database table. It is about 20MB in size - so small that it would fit on a 32MB USB memory stick. If you have a nifty cookbook application on your computer, it might give you the ability to do ingredients searches. You type in [Blueberries] and hit the SEARCH button. Under the covers, the application runs the query `SELECT RECIPE_NAME, RECIPE_TYPE from RECIPES where INGREDIENT = 'Blueberries'`.

In relative terms, scanning the cookbook to find recipes with blueberries as an ingredient might take 592 CPU instructions if the entire cookbook is loaded into database memory. If the cookbook is not already in database memory, additional CPU and I/O instructions will be needed

to complete the scan, and the scan might take five seconds to complete. Now, if a useful index on INGREDIENT is available, the same search could be completed using only six CPU instructions and would be finished in a fraction of a second. Five seconds might be a reasonable response time, but the properly indexed access uses 99% fewer CPU instructions and delivers sub-second response time.

We readily admit that your business system applications probably don't do frequent searches for blueberries, but the same principles apply. Just about every business system database and data warehouse has a large number of frequently accessed small tables that would fit on USB jump drives - and easily fit into database main memory. Ask your DBAs how many tables exist in your databases that are smaller than 64MB. And, I repeat, scans are costly. They take time, lots of CPU processing and energy, and they make your IT budgets hurt through inflated hardware and licensing costs.

Here is the first core problem: DBAs use a utility provided by the database vendor called EXPLAIN to review the database's access strategy to satisfy queries. The EXPLAIN utility reports an anticipated cost for executing the query. DBAs normally review the anticipated costs before putting queries into production. If the anticipated cost is high, they work on tuning the statement and physical design to reduce its anticipated cost. If the anticipated cost is relatively low, they pay little or no attention to it.

Please understand that a query that accesses a small table will have a relatively small anticipated execution cost according to EXPLAIN. You also need to understand that EXPLAIN provides estimates for only a single execution of a given query. So now what happens when your business application frequently accesses smaller tables to look up codes, items, or recipes? The database will load the smaller tables into memory and scan them, in the absence of proper indexes, at a CPU instruction cost that is **99% higher** than needed. Processing waste like this is prolific in at least nine out of ten production databases and it isn't the DBAs fault - EXPLAIN does not consider frequency of execution and it will report low anticipated costs for queries against small tables.

At a large bank having a multi-terabyte OLTP banking application, 34% of all CPU time, on a machine with four CPUs, was attributable to queries running against a table having only 32 rows. At a large retailer, adding a missing index to a small 2,000 row table in an 8TB warehouse reduced the elapsed time of a critical decision support query from three hours down to two minutes. At another online retailer, 97% of the CPU cost, on a machine with eight CPUs, was attributable to a single query accessing a relatively small table. These examples are not uncommon. Sometimes physical design defects and their cures will provide outstanding performance results to the business - as in the case of the improved query response time. Unfortunately, **most often these physical design defects and their response time consequences are hidden from the business - you won't know how much this waste hurts your business until you get your bill for electricity, hardware upgrades, and related license costs.** Let's look at another way: If your neighbor ran extension cords from your house to his to power his home, your lights would still be on but your electric bill might be doubled --- and you'd likely want to put an end to this leakage immediately. Why are you letting your database rob you?

Here is the second core problem: Despite all of the marketing claims about databases with self tuning memory capabilities, the best that any commercially available database can do is *REACT*

If databases tune themselves, why do the database vendors and other ISVs sell performance tools?

to observed performance characteristics and adjust memory settings to *COMPENSATE* for physical design flaws. Simply stated, automatic memory tuning will detect costly queries and attempt to reconfigure database memory such that the small tables

remain relatively resident in database memory. This mitigates I/O costs and provides some response time protection, but these queries are still running at a CPU cost that is 99% higher than they should be. Further exacerbating the problem, continuous adjustments to memory to compensate for physical design flaws also carries an unnecessary CPU cost. Automatic tuning works best, if at all, when proper indexes are in place and the physical design is optimized.

If your Database Vendor sells CPUs and PVU licenses, and claims to automatically tune your database, is it prudent to have the fox watch the hen house?

Adding insult to injury, so to speak, it is also well documented by IBM Austin Research, multiple hardware vendors, and this YouTube video (<http://www.youtube.com/watch?v=qWfgYotYfOk>)

How many people can you get into an elevator? More if you use smaller people.

that higher CPU consumption consumes more electricity and throws off more heat. By ridding your organization's databases of unnecessary and wasteful CPU consumption, you can obtain a ripple effect of several benefits:

Lower CPU consumption, lower energy costs for hardware and cooling, improved response times, improved server consolidation and virtualization, lower hardware costs, lower software licensing costs, and improved organization productivity through better, more reliable, and more predictable response times.

*How many virtual database servers can you get into a frame? **More if you put your database transaction processing costs on a diet!***

In this current economic climate, the immediate need and value of database performance cost optimization and tuning cannot be overstated. Unless your organization is flush with cash and has no regard for energy and the environment, it is past time to give your organization's databases a proper tune up. Ignoring this call to action makes no more sense than driving your car around with under inflated tires and a trunk full of rocks. For independent professional opinions and expert guidance, and the industry's best automated performance analysis and tuning tools, please contact DBI to help tune your databases and optimize your IT costs.

Scott Hayes, President & CEO, DBI, IBM Data Champion & GOLD Consultant
Scott.Hayes@DBISoftware.com

Biography:

Scott Hayes is President & CEO, DBI Software, an IBM Data Champion and GOLD Consultant, a frequent speaker at IDUG, IBM, ISACA, and ISSA conferences, published author, and regular blogger on DB2 LUW Performance.