The Explosion in DBMS Choice

Database options in a cost-conscious world

A Monash Information Services Bulletin

by

Curt A. Monash, PhD.

August, 2008

Sponsored by:

The end of a database era?

Leading DBMS do a good job at most tasks.

General-purpose relational database management systems -- such as Oracle – are great products. They form the underpinnings to most of the world's transactional applications, most of the world's data warehouses, and many more specialized systems as well. Their versatility is truly amazing.

More targeted DBMS do even better.

But the same maturity and versatility that make the leaders so impressive also cause them to be complex, cumbersome, and locked into design choices that aren't ideal today. And so, for ever more use cases, there are superior alternatives. Specialty data warehouse DBMS, running on cost-effective MPP (Massively Parallel Processing) systems, far outperform the SMP-bound (Symmetric MultiProcessing) market leaders. While the leaders support native text search and XML processing, specialist products far outdo them in relevance and speed. And if you want a stripped-down, embeddable database running on a single-purpose electronic device, Oracle and SQL Server are far from the best alternatives.

Mid-range DBMS meet most needs, at low TCO.

Even where general-purpose DBMS do a great job, market-share leading products may not be the best choice. For 20 years, there have been simpler and less expensive alternatives, including reseller-oriented products (such as Progress or Informix SE), Microsoft SQL Server (before it grew up), or open source DBMS such as MySQL and PostgreSQL. These *mid-range* products' initial appeal is often just price – they cost less to buy than the alternatives, or perhaps even can be had for free. An even greater cost advantage, however, can come in ongoing administration – because they're newer and simpler, mid-range DBMS are often much easier to administer than their high-end brethren.

They meet ever more application needs.

For more and more transactional applications, mid-range DBMS do everything that is needed. Hot standby/failover? 24/7 operation? Triggers, stored procedures, and declarative referential integrity? Datatype extensibility? Draw up your OLTP feature checklist, and one or more mid-range DBMS are apt to meet it. Leading commercial products still are ahead in super-high-end scenarios, whether the need is extreme throughput, many-9s bullet-proofing, or advanced security certifications. But except for those edge cases, mid-range DBMS can meet just about any need.

Plug-compatibility extends their reach.

Actually, there's one big exception – portability from existing DBMS. Startups aside, almost every enterprise has a rich portfolio of database applications, and porting them to a new DBMS is a daunting task. But even here there's been major progress. EnterpriseDB offers robust Oracle compatibility. Other vendors – especially but not only in data warehousing -- have portability/transparency offerings as well.

A disruption scenario is unfolding.

These developments add up to a classic "disruption" scenario, as the term is used in Clayton Christensen's *The Innovator's Dilemma.* Startup DBMS vendors are exploiting new markets (e.g., Web 2.0), technical strategies (e.g., data warehouse appliances), and business models (e.g., open source). Traditional leaders continue to innovate magnificently, but in ways that matter to fewer and fewer users (mainly, the largest enterprises running the biggest transactional apps). And the cheaper, simpler upstarts are maturing until they're suitable for ever-larger fractions of the overall market.

Consider moving away from the market leaders.

So is it time to move away from your current market-leading DBMS products? Yes and no. Reasons not to move include in-house skills, quantity-discount contracts, applications that would be difficult to port, or third-party applications that don't run on upstarts' products at all. But for many new applications, the TCO advantages of newer DBMS are so compelling that a switch is called for. And in increasingly many cases, even existing applications are better off moved.

Differentiation among relational DBMS

Until recently, few RDBMS could do the job at all.

For years, choosing a relational DBMS boiled down to one question: Does it work? Only in the past ten years have there been multiple market-leading DBMS that could run substantially all OLTP applications on a variety of large multiprocessor systems. Before then, DBMS choices were sharply constrained by such factors as row-level locking, subquery execution, online backup performance, multiprocessor support, or basic functionality in triggers, stored procedures, and declarative referential integrity.

Choices have exploded.

But now the choices are broader. No matter what your OLTP application design or (almost) transaction volume, you can probably get the job done with any of Oracle, DB2, or Informix. Postgres Plus isn't far behind. If you're only using standard alphanumeric datatypes, the list expands further, to include products from Microsoft, Sybase, Progress, and perhaps MySQL as well. And if you want to do analytics, the list of viable data warehouse DBMS is probably even longer.

It's time to look beyond the "it works" feature, and focus on TCO.

If many DBMS *can* do the job for you, which should you choose? How about the least costly one? That's almost always the right answer, provide you account for all the various elements of TCO (Total Cost of Ownership), including:

- License fee
- Maintenance fee
- Programming cost
- Administrative cost
- Hardware (and power, etc.) cost

True, time-to-deployment can be a separate consideration than cost – but generally, the system you can get into production the fastest is also the one that's easiest to program and administer.

TCO depends on four major factors.

So how do you determine which DBMS actually provides lowest TCO? Well, let's start by rephrasing our list of TCO components. It really breaks up into four buckets:

1. Performance (throughput and/or latency) given a certain configuration of hardware.
2. Features that save you from what otherwise would be painful coding effort.
3. Inherent ease or difficulty of administration, plus tools that save administrative effort.
4. Money paid to the vendor.

Sorting out the details

The top performance issue is platform support.

The biggest performance issue separating DBMS is: What platform(s) do they run (well) on? If you have a 20 (or 200) terabyte data warehouse, it probably belongs on a shared-nothing MPP set-up. Shared-everything SMP-oriented Oracle is not a good alternative. But for your 5 terabyte OLTP database, cluster- and cache-friendly Oracle may be an outstanding choice.

Performance, programming, and administration are intertwined.

Beyond that, performance, programmability, and ease of administration are all intertwined. A feature – even an important one – isn't valuable just because it exists; performance must be decent as well. That can be an issue for anything from replication to referential integrity to, most particularly, extended datatype support. Second, a large amount of programming and administrative effort goes into assuring performance – and the amount of such effort needed may vary greatly from system to system.

Different DBMS support different ways to code.

One might think that DBMS don't differ much in their programmability. After all, DBMS are basically big SQL interpreters, and SQL is a fairly standardized language. Even so, different DBMS can, for the same application, require vastly different amounts of coding. Many of the differences lie in the answers to two questions:

1. Where does the code go?
2. What do you have to code at all?

Extended datatype support can be crucial.

Most database applications get by with 3-5 kinds of datatype, including

1. Numbers (denoting quantity)
2. Character strings (as identifiers)
3. Dates

and maybe also

4. Free text/memo fields (a lot like character strings)
5. BLOBs, CLOBs (Binary or Character Large OBjects), or externally-managed files.

But as the world gets ever more wired, increasingly many kinds of information are managed. So DBMS need to handle datatypes well beyond the five core kinds, for example in the areas of:

6. XML
7. Text (with much better search than is common on character or memo fields)
8. Geospatial

Implementations vary greatly.

In theory, one can get by without built-in datatype support. But when it's present, it can have two huge advantages:

- Reduced programming effort
- Much superior performance

Actually, there are two major ways of implementing extended datatype support, and they have very different performance implications. First, one can do almost anything via User Defined Functions (UDFs). These run in the memory controlled by the DBMS, and confer a programming advantage but usually not a performance one. (Indeed, they can be a source of regrettable overhead.) Second, there can be true native datatype support, complete with tailored access methods. This is found mainly in DBMS that have a full datatype extensibility framework – Oracle, DB2, Informix, and the Postgres family. Absent such a framework, adding native datatype support is hard, although Microsoft has invested in doing so in a few important cases.

Database code can run on multiple tiers.

Every commercial DBMS knows how to respond to a SQL call, whether it comes directly from a client or from some kind of middle tier. But there actually are a number of other possibilities. While terminology varies slightly depending on who you ask, the main ones are:

- *Triggers,* which can be the best way to enforce database integrity, security, and so on.
- *Declarative referential integrity,* the most important special case of triggers.
- *Stored procedures,* a generalization of the trigger idea that also can be used to boost performance by minimizing network traffic.
- *Middle-tier caching,* which can happen independent of the DBMS, but increasingly is being done via DBMS companion products.

Every serious OLTP DBMS vendor has capabilities in the first three of these areas, if not all all four, although maturity and performance may be another matter. However, the same is not quite true in data warehousing. For example, referential integrity may not be required, if one can assume the data is coming from source OLTP systems where integrity has already been checked.

Advanced DBMS automate "industrial-strengthness."

Ideally, an application runs on a single DBMS instance. If the application is mission-critical, this instance is mirrored on another, perhaps geographically remote machine, with hot failover should the first machine crash. Practice deviates from this ideal in a number of ways. Sometimes, the only way you get good performance is to break up data among multiple instances. Replication and failover may need to be hard-coded as well. In general, the market leaders have an advantage over the mid-range vendors in these areas, the Postgres family possibly excepted.*

**Also, MySQL has famously been used for some large "sharded" web databases with* very *simple database schemas.*

Index maintenance is a huge component of database TCO.

One area in which mid-range or specialty data warehouse DBMS actually outdo the market leaders is index maintenance. The market leaders have old designs, and all-things-to-all-people products. The resulting heap of spaghetti leads to a tremendous amount of manual work getting indexes right (and similarly the organization of the database itself). Sophisticated administration tools, whether third-party or vendor-supplied, certainly help. What's more, most enterprises have in-house expertise in at least one market-leading DBMS, so the incremental cost of running another database isn't always high. But on the whole, mid-range or data warehouse specialty DBMS are fundamentally easier and cheaper to administer than traditional market-leading relational products.

How to move forward

Few new databases belong on market-leading DBMS.

Many new applications are built on existing databases, adding new features to already-operating systems. But others are built in connection with truly new databases. And in the latter cases, it's rare that a market-leading product is the best choice. Mid-range DBMS (for OLTP) or specialty data warehousing systems (for analytics) are usually just as capable, and much more cost-effective. Exceptions arise mainly in three kinds of cases:

- Small enterprises with very limited staff.
- Large enterprises that have negotiated heavily-discounted deals for a market-leading product.
- Super-high-end OLTP apps that need absolute top throughput (or security certifications, etc.)

Otherwise, the less costly products are typically the wiser choice.

Many legacy analytic systems should be ported.

In the analytics area, appliances and other specialty data warehousing products offer huge price/performance advantages over general-purpose systems. What's more, their superior performance allows them to get by with much simpler indexing structures, greatly reducing administrative burdens. If you have a data warehouse -- or just a collection of data marts – running on Oracle or Sybase Adaptive Server or Microsoft SQL Server, it's likely you could do yourself a huge favor by moving it to a specialty system.

So should third-party applications.

If you're an ISV, selling copies of the same software to many different customers, you should not be locked into expensive market-leading DBMS. Whatever the remaining deficiencies of mid-range systems, at least one of them will surely be good enough to support your software with an acceptably low level of one-time porting effort. (In many cases, EnterpriseDB's Postgres Plus Advanced Server will have the edge, due to its Oracle compatibility as well as its generally rich feature set.) What's more, besides saving license and maintenance fees, a mid-range DBMS may be easier for your customers to operate than a complex market leader is.

In-house OLTP apps may be fine where they are.

The one area where it may be premature to port away from market-leading DBMS is in-house OLTP applications. The first rule for OTLP apps is that they Must Not Break And so if they're not broken, it is often advisable to be cautious about fixing them. In some cases prompt porting is a good idea anyway, but often there will be lower-hanging fruit elsewhere in the enterprise.

Links are provided to further analysis.

As you may imagine, this paper contains only a small fraction of our analysis of DBMS alternatives. Indeed, that's the main topic of our blog DBMS2. Specific recommended links include:

- The first of an extensive series of blog posts about database diversity, containing links to many of the rest (most of which are by me, but a couple of which are by database pioneer Michael Stonebraker).
- our coverage of data warehousing
- our coverage of mid-range database management systems

About the Author

For more than a quarter-century, Curt Monash has been a leading analyst of and strategic advisor to the software industry. Praised by Lawrence J. Ellison for his "unmatched insight into technology and marketplace trends," Curt was the software/services industry's #1 ranked stock analyst while at PaineWebber, Inc., where he served as a First Vice President until 1987. Since 1990 he has owned and operated Monash Research, a highly acclaimed technology analysis firm focused on enterprise software. He has been extensively published and quoted in the technology and general business press, and has been a regular columnist for Application Development Trends, Software Magazine, and Computerworld. To get Curt's latest research, please see www.monash.com/blogs.html or the database-centric site www.dbms2.com.

Prior to his business career, Curt earned a PhD. in Mathematics (Game Theory) from Harvard University at the age of 19. He has held faculty positions in mathematics, economics and public policy at Harvard, Yale, and Suffolk Universities. For more information please see www.monash.com/curtbio.html.

About the Sponsor

EnterpriseDB provides enterprise-class products and services based on PostgreSQL, the world's most advanced open source database management system. Its flagship product lines are Postgres Plus and Postgres Plus Advanced Server, which we regard as top-tier products in the category we call *mid-range database management systems*. For more information please see www.enterprisedb.com.