A BPTrends Column Updated Feb. 2013

MDA Journal

David S. Frankel
David Frankel Consulting

david@dFrankelConsulting.com
http://www.linkedin.com/in/davidsfrankel

Author:
Model Driven Architecture:
Applying MDA to Enterprise Computing

Radical Simplification

In-Memory Databases Challenge Assumptions in Enterprise IT

Medium-sized and large enterprises strictly separate their operational data stores from data warehouses used for post-transactional analysis. Now in-memory database technology is poised to shuffle these cards. This new technology also has the potential to change enterprise business processes and the overall architecture of enterprise hardware and software systems.

Reconsidering Data Warehousing

In myriad IT data centers, specialized ETL (extract-transform-load) software extracts data from operational enterprise systems that serve as systems of record for online transaction processing (OLTP). The ETL software transforms the extracted data into formats amenable to online analytical processing (OLAP). The OLAP-optimized data is a copy – albeit a differently formatted copy -- of the OLTP-optimized data. The ETL software loads the copy into a physical data store that we usually refer to as a data warehouse, which is separate from the operational data stores.

The reason that the data warehouse's OLAP-optimized data is maintained in a separate, additional data store is that the extraction and transformation of operational data into OLAP form takes a significant amount of time. The computer industry has invested in software and hardware that stores the results of this data processing, so that extraction and transformation do not have to be repeated when an analyst requests an analysis from the OLAP system. The benefit of not having to repeat the extraction and transformation has – up to now – outweighed the cost of the additional software and hardware required to maintain a separate data store.

In-memory database technology fundamentally changes the cost-benefit tradeoff. The new database technology takes advantage of the fact that massive amounts of solid-state memory are now so inexpensive that companies can afford to run their enterprise databases in such memory. The OLTP-OLAP scenario of the future has the operational database living in solid state memory, while extraction and transformation into OLAP-optimized form takes place on-the-fly when an analysis is requested.

This change obviates the need to maintain a separate data store to support data analysis. The extraction from and transformation of the operational system of record data occurs dynamically upon demand. There is still a need to design the OLAP-oriented data formats ahead of time, but

Copyright © 2013 David S. Frankel. All Rights Reserved.

the formatting of the actual data in accordance with the design occurs in real time.

Consequently, ETL software is morphing into simpler extract-transform software. Furthermore, the need for hardware to store the extracted and transformed data is going away, which portends substantial simplification of hardware topologies.

Reconsidering Business Processes

Producing an analysis of data requires two sets of computations. The first set is the extraction and transformation of data into OLAP-optimized format; these are the computations we have been discussing. The second set of computations performs the required analysis on the transformed data; in-memory databases dramatically speed up this set of computations too.

The speed-up of both of these sets of computations does not mean simply that enterprises will do the same old things more rapidly. The speed increase is so great as to impel qualitative changes in business processes. When analysis reports that would have taken twelve hours to produce previously now take just a few minutes, enterprises can seize new opportunities to use OLAP to optimize their operations.

For example, a bank will be able to obtain far-reaching liquidity and risk analysis so much more rapidly as to allow quick operational adjustments in scenarios where the analysis would previously have taken so long to perform that by the time the results were available the reality that the results portray would have changed, making it too late to execute the adjustments.

Reconsidering Design of Operational Applications

A substantial percentage of the database tables and fields used by today's operational enterprise applications are not logically necessary, but are there to store computations that some application functionality needs. When a demand for the functionality occurs during the course of enterprise operations, the functionality executes more quickly because the results of the computations are already available. (The computations of the operational systems whose results are stored this way are separate from the computations needed to support analysis described earlier.)

In-memory databases obviate the need for database tables that store pre-run computations, and eliminate the need for many of the fields in the tables that remain. Applications will gradually change to adapt to the new technological reality, significantly simplifying their design and maintenance.

Reconsidering Multi-Tiered Systems

Since the new in-memory databases are so fast and can hold so much on small server boxes, IT architects are starting to question the architecture developed in the 1990s that separates enterprise systems into three or even four "tiers."[1] The best IT architects today are intimately familiar with the multi-tiered systems approach, but with the emergence of in-memory database technology we are hearing calls to move away from it.

This questioning of conventional wisdom is fair, but here I wish to interject a note of caution. Multi-tiered architecture has two key dimensions. The first dimension is the logical organization of information and code. The second dimension is the physical location of the information and code. As the IT industry considers the implications of in-memory databases for multi-tier architecture, we must not conflate these distinct dimensions.

The logical organization of information and code is about separation of concerns, which architects know is vital for scalable design. By the 1970s, the IT industry had figured out that, since multiple applications use the same data, it makes sense to factor the definition and management of the data out of the applications into what the pioneers decided to refer to as a database. This logical

[1] The Enterprise Java Blueprints were a key contribution to the body of knowledge about the multi-tiered approach. See http://java.sun.com/blueprints/enterprise/index.html, and John Crupi's patterns books.

Copyright © 2013 David S. Frankel. All Rights Reserved.

separation of databases from applications is distinct from the physical organization; the applications and the database can be on the same machine or they can be on separate machines. The logical organization makes development and maintenance of the software more efficient, while the physical organization takes other factors into account. It is true that logical distribution may in part drive decisions about the physical distribution, but these are still separate dimensions.

I have been a bit troubled that some of the discussions I've seen regarding the idea of "pushing code to the database" do not expressly point out the difference between logical vs. physical distribution. If as a result of these conversations developers move away from hard-won knowledge about logical separation of concerns, the IT industry will do itself, its users, and the macro-economy a disservice.

My cautionary note should in no way deter architects from considering changes in the physical distribution. Collapsing physical distribution tiers can make an important contribution to simplifying our systems. However, collapsing logical distribution tiers could actually make our systems more complex and difficult to manage.

This discussion of multi-tiered architecture has focused on separation between application and database on the back end, because that is where in-memory databases have the potential to change the distribution. As an aside, I also have been uneasy that in the rush to build mobile capabilities into enterprise systems there may be a tendency to disregard accumulated knowledge about logical distribution tiers closer to the front ends of systems. There is a reason that many systems architects have engineered a separation of concerns, on the client side, between presentation and interaction. In this approach, code running on multiple client devices reuses a logical pattern of user interaction that has been captured separately, much as, on the back end, multiple business applications reuse a database design that is logically outside the application. The separation of concerns on the front end helps tremendously with scalability; and it does not preclude an optimization where some kind of automated or semi-automated process takes the logical interaction design as input and produces a device-dependent implementation of the interaction that deploys physically on the device – again, logical and physical distribution are not the same.

Conclusion

Sometimes conventional wisdom is a barrier to progress. Sometimes refusal to consider conventional wisdom is a barrier, as progress achieved with great effort can be lost. I urge the IT industry to have the courage to break out of the chains of old thought when necessary in order to simplify enterprise systems as radically as possible, and to have the humility to realize that people before us may have figured out some things that we would be well advised to learn up front, as opposed to learning them the hard way down the road.

David Frankel has over 30 years of experience in the software industry as a technical strategist, architect, and programmer. He is recognized as a pioneer and international authority on the subject of model-driven systems and semantic data modeling. He has published two books and dozens of trade press articles, and has co-authored a number of industry standards including XBRL, ISO 20022, BIAN, and UML®.

Copyright © 2013 David S. Frankel. All Rights Reserved.