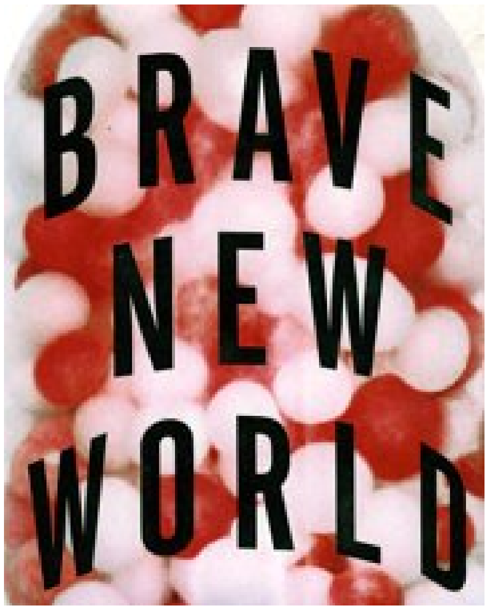
Jim Starkey is Creating a Brave New World by Rethinking Databases for the Cloud

Thursday, August 4, 2011 at 9:11AM

Todd Hoff in NewSQL, nosql



Jim Starkey, *founder of* NuoDB, *in this* thread *on the Cloud Computing group, delivers a* masterful post *on why he thinks the relational model is the best overall compromise amongst the different options, why NewSQL can free itself from the limitations of legacy SQL architectures, and how this creates a brave new lock free world....*

I'll *[Jim Starkey]* go into more detail later in the post for those who care, but the executive summary goes like this: Network latency is relatively high and human attention span is relatively low. So human facing computer systems have to perform their work in a small number of trips between the client and the database server. But the human condition leads inexorably to data complexity. There are really only two strategies to manage this problem. One is to use coarse granularity storage, glombing together related data into a single blob and letting intelligence on the client make sense of it. The other is storing fine granularity data on the server and using intelligence on the server to aggregate data to be returned to the client.

NoSQL uses the former for a variety of reasons. One is that the parallel nature of scalable implementations make maintaining consistent relationship among data elements problematic at best. Another is that clients are easy to run in parallel on multiple nodes, so moving computation from servers to clients makes sense. The downside is that there is only one efficient way to view data. A shopping cart application, for example, can store everything about a single user session in a single blob that is easy to store, fetch, and contains just about every necessary take an order to completion. But it also makes it infeasible to do reporting without moving vast quantities of unnecessary data.

SQL databases support ACID transactions that makes consistent fine granularity data possible, but also requires server side intelligence to manage the aggregation. The SQL language, ugly as it may be, performs the role with sufficient adequacy (we could do better, but not significantly better). Stored procedures using any sort of data manipulation language also works. But distributed intelligence, declarative or procedural, pretty much requires regularity of data. Hence schemas. They don't have to be rigid or constraining, but whatever intelligence is going to operate on bulk data needs a clue of what it looks like and what the relationships are.

So that's the short version. The long version requires a historical understanding of how we got from an abstract to structured views of data.

On of the earliest high performance database systems (they weren't called that yet) was a mainframe product, circa 1969, whose name eventually settled on Model 204 (person node: I did the port from DOS/360 to OS/360 MVT). Model 204 had records that were arbitrary collections of attribute/value pairs. The spooks loved it -- dump everything you know into cauldron and throw queries at it. But it couldn't handle relationships among records without a lot of procedural code.

There were two data models that attempted to capture data structure (relationships). One was the hierarchical model, IBM's IMS and the ARPAnet Datacomputer (personal note here, too). The other was the

CODASYL (aka network) data model where individual records "owned" chains of other records. IDMS, the premier CODASYL product roamed the earth in the late Jurassic period until an alternative emerged from the primordial swamps (sigh, I was on the team that ported IDBMS to the PDP-11, often likened to an elephant on a bicycle). Both the hierarchical and network models required data manipulation languages that an orc's mother couldn't love.

Codd's relational model, once stripped of the unnecessary mathematical trappings, was a very happy compromise between simple regular tables and ad hoc joins for inter-table relationships. And it made for straightforward transactional semantics as well. It was a hit when the first commercial implementation came out and has persevered (I put out DEC's first relational database, Rdb/ELN, in 1984).

The OO data model (encapsulated objects and pointer) pops up and dies with great regularity, as does its close relative, object/relational mapping. Neither is particularly suited for the distributed world, however, but the refugees can generally find work in other companies.

Over the years I've had a professional affiliations with the amorphous data model (Model 204), hierachical (Datacomputer), CODASYL (DBMS-11), and relational (Rdb/ELN, Interbase, Firebird, MySQL, and NimbusDB). And my friend Tom Atwood started at least half of the OO database companies. So, if I can't claim objectivity, I can at least claim in depth personal experience.

Everything is a compromise. And I deeply believe that the relational model is the best compromise among simplicity, power, performance, and flexibility of implementation. It does require data description, but so do all other useful database management systems regardless of what it is called.

The scale problems with legacy relational databases have nothing to do with SQL, the relational model, or ACID transactions. The essence of the problem is theoretical -- the conflation of consistency and serializability. Serializability is a sufficient condition for database consistency but not a necessary condition. The NewSQL generation of products recognize this and refuse to be limited by lock managers of any ilk.

It's a brave new world out there, ladies and gentlemen.

If this seems like magic, take a look at How NimbusDB Works *for more details.* Are Cloud Based Memory Architectures The Next Big Thing? *has a lot of clarifying quotes from Jim Starkey as well.*

Related Articles

- Is NoSQL A Premature Optimization That's Worse Than Death? Or The Lady Gaga Of The Database World?
- Thoughts from GigaOM Structure 2011
- NimbusDB Presents at Under the Radar
- Databases in 21st Century: NimbusDB and the Cloud by Jim Starkey
- 2011 New England Database Summit Video on Facebook
- Jim Starkey: Introducing Falcon (the video and the podcast)
- Special Relativity and the Problem of Database Scalability by Jim Starkey (audio)
