Stonebraker:
da teoria para a prática bem sucedida em sistemas de bancos de dados
(uma testemunha ocular dessa trajetória – Marta Mattoso)

Marta Mattoso
Linha de BD, PESC, COPPE
Michael Stonebraker recebeu em junho, o prêmio Turing da ACM de 2014. “For fundamental contributions to the concepts and practices underlying modern database systems.”

Stonebraker

Outros premiados em BD

Ted Codd, 1970
Jim Gray, 1998
“Brought Relational Database Systems from Concept to Commercial Success

Set the Research Agenda for the Multibillion-Dollar Database Field for Decades”

**ACM Press Release**

- Database systems are critical applications of computing and preserve much of the world’s important data.
- Stonebraker invented many of the concepts that are used in almost all modern database systems. He demonstrated how to engineer database systems that support these concepts and released these systems as open software, which ensured their widespread adoption.
- Source code from Stonebraker’s systems can be found in many modern database systems.
- During a career spanning four decades, Stonebraker founded numerous companies successfully commercializing his pioneering database technology work.

---

**NO CENTRO DE MUDANÇAS DE TECNOLOGIA E POLÊMICAS**

As batalhas de Stonebraker
Stonebraker timeline
ACM Press Release

• Stonebraker developed Ingres, proving the viability of the relational database theory.
• Ingres was one of the first two relational database systems (the other was IBM System R).
• Major contributions including query language design, query processing techniques, access methods, and concurrency control, and showed that query rewrite techniques (query modification) could be used to implement relational views and access control.

Anos 70
WW I

• Apostando no modelo relacional: Ingres
• IBM contra-ataca com DB2 (System R)
• Perdendo a batalha para a Oracle

contra a hegemonia dos fabricantes
Stonebraker timeline
ACM Press Release

• Open source DBMS
• Object-relational model of database architecture integrating important ideas from object-oriented programming into the relational database context.
• Concepts introduced in Ingres and Postgres can be found in nearly all major database systems today. Ingres and Postgres were well engineered, built on UNIX, released as open software, and form the basis of many modern commercial database systems including Illustra, Informix, Netezza and Greenplum.

Anos 80

• Orientação a Objetos e SGBD-R
• A guerra entre os OR e os OO
• Desenvolvimento do Postgres-OR, Illustra
• Combatendo pelos OR

contra a hegemonia dos fabricantes de R puro
Anos 90
WW III

contra a hegemonia dos líderes mercado SGBDR

- Movimento código aberto
- Postgres e variações
- Otimização extensível, adaptável

Stonebraker timeline
ACM Press Release

D-INGRES
- Lasting technical results on distributed query processing and transaction coordination protocols through development of Distributed Ingres, one of the first distributed database systems.

XPRES
- XPRS, a parallel version of Postgres that explored the “shared nothing” approach to parallel database management.

Mariposa
- Mariposa, a massively-distributed federated database system, explored ideas such as opportunistic data replication and decentralized query processing.
Developed database architectures for specialized purposes

<table>
<thead>
<tr>
<th>Aurora/StreamBase</th>
<th>• real-time processing over streaming data sources</th>
</tr>
</thead>
<tbody>
<tr>
<td>C-Store/Vertica</td>
<td>• column-oriented storage architecture resulted in systems optimized for complex queries</td>
</tr>
<tr>
<td>H-Store/VoltDB</td>
<td>• high throughput, distributed main-memory online transaction processing system</td>
</tr>
<tr>
<td>SciDB</td>
<td>• extreme-scale data management and data analysis system for science</td>
</tr>
</tbody>
</table>

contra a hegemonia dos líderes mercado SGBDR

Anos 2000
WW III

• “no size fits all”
• combatendo os “elefantes”
• OLAP (consultas analíticas) x OLTP (transações venda bilhetes)
Relational database management systems (DBMSs) have been remarkably successful in capturing the DBMS marketplace. To a first approximation they are “the only game in town,” and the major vendors (IBM, Oracle, and Microsoft) enjoy an overwhelming market share.

- They are selling “one size fits all”; i.e., a single relational engine appropriate for all DBMS needs.
- Moreover, the code line from all of the major vendors is quite elderly, in all cases dating from the 1980s. Hence, the major vendors sell software that is a quarter century old, and has been extended and morphed to meet today’s needs.

**MS: these legacy systems are at the end of their useful life. They deserve to be sent to the “home for tired software.”**


**Anos 2010**

- NoSQL: Abandonar ACID não é uma boa ideia
- SGBDR-P mais eficiente que Hadoop/Hive etc. para bancos de dados

**critica os sistemas NoSQL e ecossistema Hadoop**
New SQL: An Alternative to NoSQL and Old SQL for New OLTP Apps


• **NoSQL.** There have been a variety of startups in the past few years that call themselves NoSQL vendors. Most claim extreme scalability and high performance, achieved through relaxing or eliminating transaction support and moving back to a low-level DBMS interface, thereby eliminating SQL.

• In my opinion, these vendors have a couple of issues when presented with New OLTP. First, most New OLTP applications want real ACID. Replacing real ACID with either no ACID or “ACID lite” just pushes consistency problems into the applications where they are far harder to solve. Second, the absence of SQL makes queries a lot of work. In summary, NoSQL will translate into “lots of work for the application”—i.e., this will be the full employment act for programmers for the indefinite future.

We see the following trajectory in Hadoop data management:

• Step 1: Adopt Hadoop for pilot projects.
• Step 2: Scale Hadoop to production use.
• Step 3: Observe an unacceptable performance penalty.
• Step 4: Morph to a real parallel DBMS. MapReduce is an internal interface in a parallel DBMS, and one that is not well suited to the needs of a DBMS

MapReduce is an internal interface in a parallel DBMS, and one that is not well suited to the needs of a DBMS
Impala is architected exactly like all of the shared-nothing parallel SQL DBMSs, serving the data warehouse market. Specifically, notice the MapReduce layer has been removed, and for good reason. MapReduce is not a useful internal interface inside a SQL (or Hive) DBMS

- Google announced MapReduce is yesterday’s news and they have moved on, building their software offerings on top of better systems such as Dremmel, BigTable, and F1/Spanner. In fact, Google must be "laughing in their beer" about now. They invented MapReduce to support the Web crawl for their search engine in 2004. A few years ago they replaced MapReduce in this application with BigTable, because they wanted an interactive storage system and MapReduce was batch-only. Hence, the driving application behind MapReduce moved to a better platform a while ago. Now Google is reporting they see little-to-no future need for MapReduce.

- From the point of view of a parallel SQL/Hive DBMS, HDFS is a "fate worse than death."

I wonder how long it will take the rest of the world to follow Google’s direction and do likewise...

http://cacm.acm.org/magazines/2015/1/181612-a-valuable-lesson-and-whither-hadoop
What do you think are some of the key trends shaping the world of data management in 2012?

• “The first one is one size no longer fits all, and that is going to give heartburn to the current [relational database] elephants.
• The second one is "big data" means three different things and you’ve got to remember which one you’re talking about.
• The third one is that ACID [atomicity, consistency, isolation, durability] is a really good idea, so don’t throw the baby out with the bathwater.
• The fourth thing is, Think memory. It’s the new disk.
• The last thing is that -- in stages -- the cloud is really the answer to save money.”

Stonebraker's advice

• For undergraduates just getting into computer science, "Learn how to code, and code well, because whatever you do is going to involve implementation."

• For Ph.D. students trying to figure out where to focus their attention, he suggests talking to real-world computer users. "They’re happy to tell you why they don’t like or do like any given technology, so they’re a wonderful source of problems to work on.”