NoSQL adoption: what's the next step?

Luca Garulli – Founder and CEO
NuvolaBase Ltd

May 2012 29 - 30 in Cologne, Germany
2009

A new grass-root movement of rebels, a few underdogs who believe that the common Relational model is no longer the only solution to every problem.

After 30 long years of “Relational domain” new alternatives have become possible and sustainable
it's name is

NoSQL
2012: 3 years later the revolution has evolved:

many new products, larger adoption
"NoSQL database technologies are largely being adopted for new projects that require additional scalability, performance, relaxed consistency and agility."

– 451 Research of May 22\textsuperscript{nd} 2012
Nobody talks about the real origins of databases.

You could marvel about the lot of similarities between the past and today.
Are you ready to go back to the past?
The computer age and the beginning of databases?
No, let's go further back before computers took over to see how society managed information persistently.
Way too back?
“Stone” was our first persistent database. Before that everything was “volatile” because in-memory only
Pros:

Cheap (it’s free), very-very durable and always available
Cons: not exactly **portable**...
...And this kind of storage had the same problems of modern RDBMS
Slow inserts!
no Market Leader for this technology because stone is FREE and unlimited
So, the world needed a better technology: what next?
Egypt, 5000 years ago

“Papyrus” was the database v. 2.0
Not so cheap
not so durable
but portable

This was the first “mobile” market
Market Leader the Pharaoh?
Wait a sec!
The Pharaoh Market Leader spoke about prophecies, exactly like an “Oracle” would...

(Mhm, I should elaborate it a little bit more)
So, the world needed an even better technology: what next?
Europe, 500 years ago

Modern “Books" became a database v. 2.1 (minor version)
Easy to make copies, not durable as stone, but portable
Users started to make choices:

1) want something really durable? Go with Stone
2) do you want something portable? Papyrus
3) Need also copies? Books
You can’t have all of them. Choose between:

easy to (C)opy

dur(A)bility

(P)ortability

But just pick 1 or 2 of them!
I can’t believe: the origin of CAP theorem?

easy to (C)opy

Stone
dur(A)bility

(P)ortability

Book

Papyrus
We found some interesting similarities with the modern databases: history repeating itself!
Interpreting Technology Hype
by Gartner

- Peak of Inflated Expectations
- Plateau of productivity
- Slope of productivity
- Trough of Disillusionment
- Technology trigger
- Visibility
- Maturity
A potential technology breakthrough kicks things off. Early proof-of-concept stories and media interest trigger significant publicity. Often no usable products exist and commercial viability is unproven.
Early publicity produces a number of success stories—often accompanied by scores of failures. Some companies take action; many do not.
Interpreting Technology Hype

by Gartner

Interest wanes as experiments and implementations fail to deliver. Producers of the technology **shake out or fail**. Investments continue only if the surviving providers **improve their products** to the satisfaction of early adopters.
Bad stories from the trenches

“Goodbye, CouchDB”
May 10th, 2012 by Steven Hazel

“Failing with MongoDB”
November 5, 2011 by Michael Schurter

“A year with MongoDB”
April 2012 on Kiip.me blog

“MongoDB: 9 months on”
11 May 2011 by Clueless Joe
More instances of how the technology can benefit the enterprise start to crystallize and become more widely understood. **Second** and **third-generation** products appear from technology providers. More enterprises fund pilots; conservative companies remain cautious.
Interpreting Technology Hype

by Gartner

Mainstream adoption starts to take off. Criteria for assessing provider viability are more clearly defined. The technology’s broad market applicability and relevance are clearly paying off.
Interpreting Technology Hype

by Gartner

Technology trigger

2009

Peak of Inflated Expectations

2010 - 2011

Trough of Disillusionment

2011

Slope of productivity

2012 - 2014

Plateau of productivity

2015 ->

Don't worry, we're here!

Visibility

Maturity

(c) Luca Garulli Licensed under a Creative Commons Attribution-NoDerivs 3.0 Unported License
Why some users don't succeed using NoSQL?
Have you applied the 3 laws of robotics NoSQL?

Don't you know them? Really?
You must apply the 3 laws of NoSQL to avoid a blood bath
The first rule of NoSQL is: «You do not talk about NoSQL»
NoSQL 2\textsuperscript{nd} law

there is no “golden hammer”

“if all you have is a \textit{hammer},
everything looks like a \textit{nail}”
NoSQL 2nd law explained

If all you have is a rdbms, everything looks like tables.

If all you have is a key value, everything looks like keys and values.

If all you have is a document, everything looks like documents.

If all you have is a graph db, everything looks like vertexes and nodes.
NoSQL 3rd law

“one size doesn't fit all”
NoSQL 3\textsuperscript{rd} law
explained: choose the right model(s)
(not exhaustive)
Problem:
what about if you need multiple models?

Very often the domain can be split in multiple sub-domains:

this is the Polyglot Persistence
Polyglot Persistence

Use **multiple storage** solutions to avoid compromising on the business data model...

Don’t change your model, change storage solution or integrate it with an additional product supports the model you need!
Multi-Model storage 1/2

one product, multiple faces

Because the Polyglot Persistence some NoSQL vendors support multiple models in the same product
Multi-Model storage 2/2

This is the best way to achieve the NoSQL goal choosing the right model for each piece of domain with no compromises

Only one product to know and manage
Multi-Model example 1/4

To model the main entities of a selling product we choose the Graph one because it has the ability to traverse items and allow fast retrieving of relationships. *NOTE: This is just an example!*

Graph Model

- Invoice
- StockItem
- Order
- Stock

NOTE: This is just an example!
Customer and Product need a flexible schema to place additional fields like the «4° mobile number» or the «XXX social network ID»
Multi-Model example 3/4

Document model

- Customer
- Provider

Graph Model

- Invoice
- Order
- StockItem
- Stock

Object Oriented Model

- Person
- Vertex
Multi-Model example 4/4

Document model

- Customer
- Provider

Constraints:
- phones > 0
- surname not null

Graph Model

- Invoice
- Order
- StockItem
- Stock

Object Oriented Model

- Person
- Vertex

Constraints:
- phones > 0
- surname not null
Lessons learned
Lesson learned 1/2

If you changed your domain to fit the selected NoSQL solution,

What you selected was probably wrong or not the very best solution!

#fail
Lesson learned 2/2

Selecting a NoSQL product because it’s simply the most famous or because your preferred Social Network is using it means that nothing has changed:

You’re making the same mistake that generations of developers made in the last 30 years by selecting the Relational DBMS for every use case!
Lesson learned 2/2b

This is against the NoSQL vein!

#superfail
Other 9 factors to consider

Obviously, the model is not the only thing to evaluate when you choose the right product:

1. Maturity
2. Constraints (lock mgmt, write strategy, reliability, etc.)
3. Who is using it in production?
4. Has anyone ever used it with a volume of data of similar to mine?
5. Is it Open Source? Is there an active community?
6. Commercial Support
7. Current skill of your team
8. TCO (Total Cost of Ownership)
9. Test it before to select even with micro-benchmarks represent your use case!
Future directions: NoSQL

2\textsuperscript{nd} and 3\textsuperscript{rd} generation of NoSQL products are providing more features RDBMS already have:

* persistence for memory-only db
* transaction or similar
* better locking to improve concurrency
* finer indexing systems
Future directions: RDBMS

Nth generation of RDBMS are providing NoSQL features like:

* schema-less
* improved horizontal scalability
* raw API for fast insertion
* native support for array/collection
* full-text, queueing, etc.
The common goal

faster, more scalable and flexible solution

NewSQL

NoSQL

RDBMS
NoSQL
what
risks & challenges
with such scenario?
In many cases companies continue to use RDBMS as primary storage leaving to the NoSQL solutions the «secondary role» of distributed and/or scalable cache.
RDBMS and NewSQL products are trying to provide a technical answer to face the "BigData" and all the problems of performance and scalability.

Companies could stay with the «improved» (thanks to NoSQL) RDBMS products.
NoSQL Challenge

The NoSQL challenge is to gain the trust of users and customers to be used not only as a secondary storage, but playing the first role in the game of the persistence.
That's all folks

Enjoy NoSQL Matters 2012 2\textsuperscript{nd} day!

Many thanks!
Hey, I’m a developer! If you want to hear something more technical don’t miss «Design your application using Persistent Graphs and OrientDB»
Today 14:45 House 6 Room 2