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

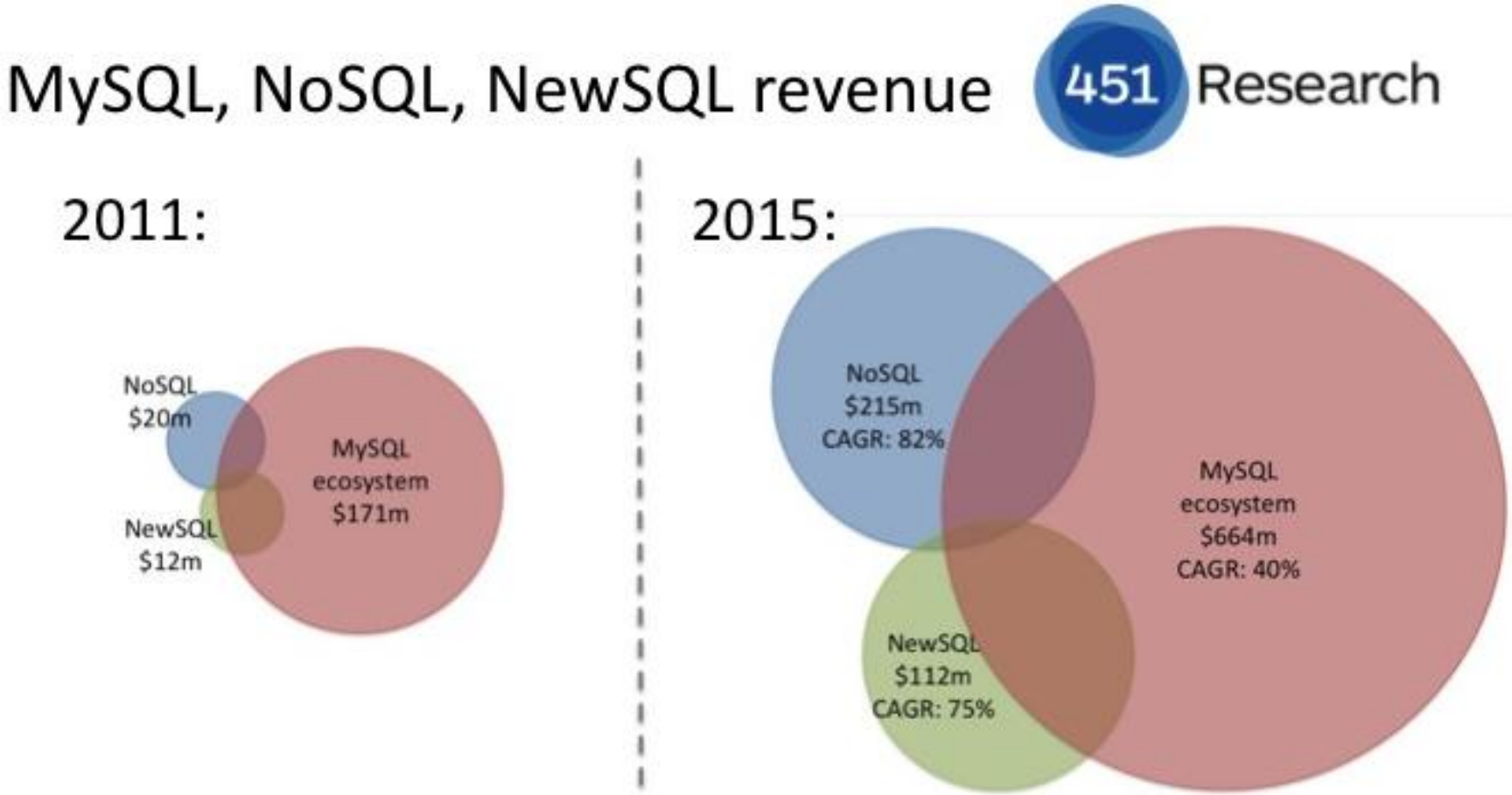
it's name is

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 22nd 2012



Nobody talks about the real origins of databases.

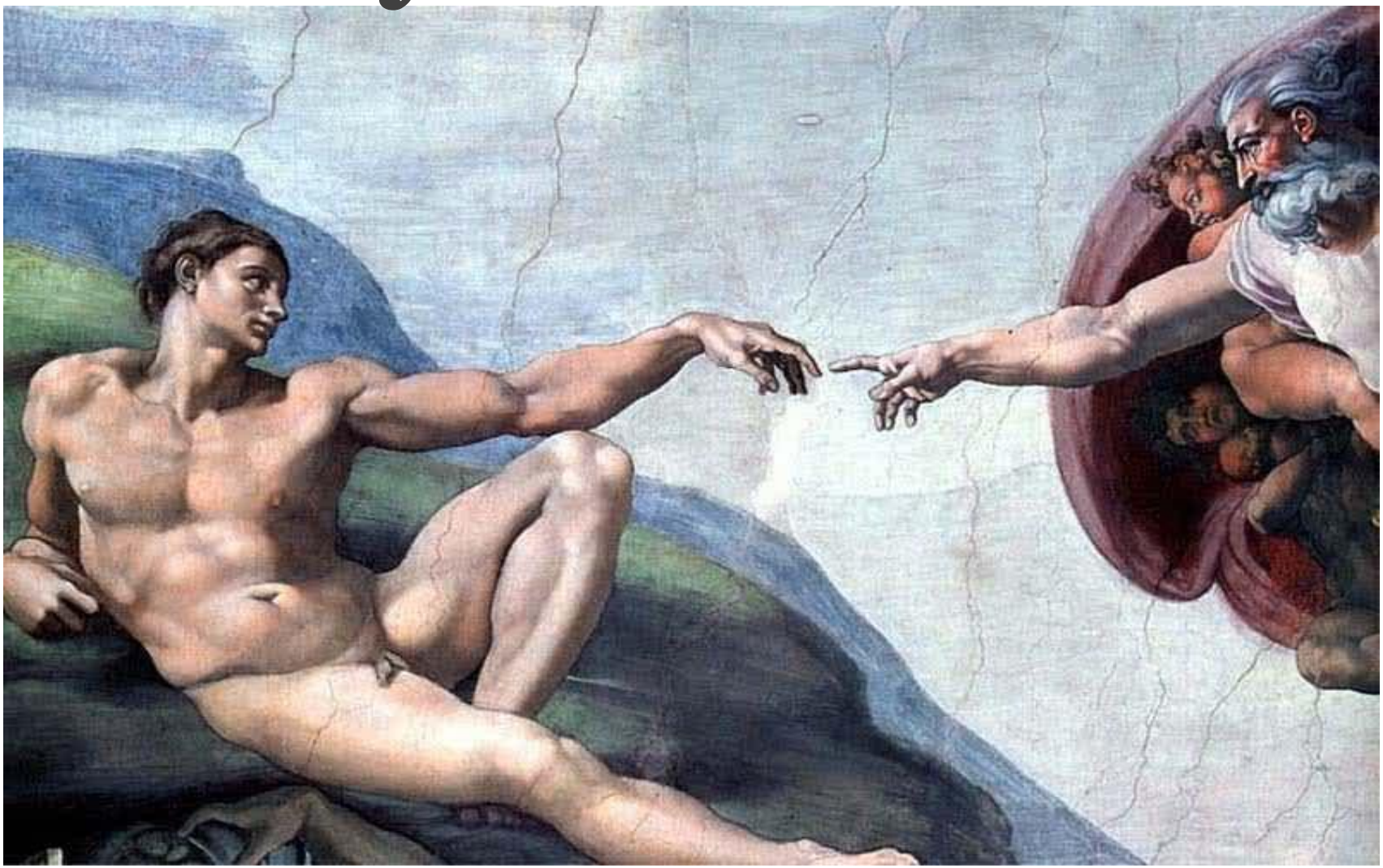
You could marvel about the lot of similarities between the past and today

Are you ready to go back to the past?

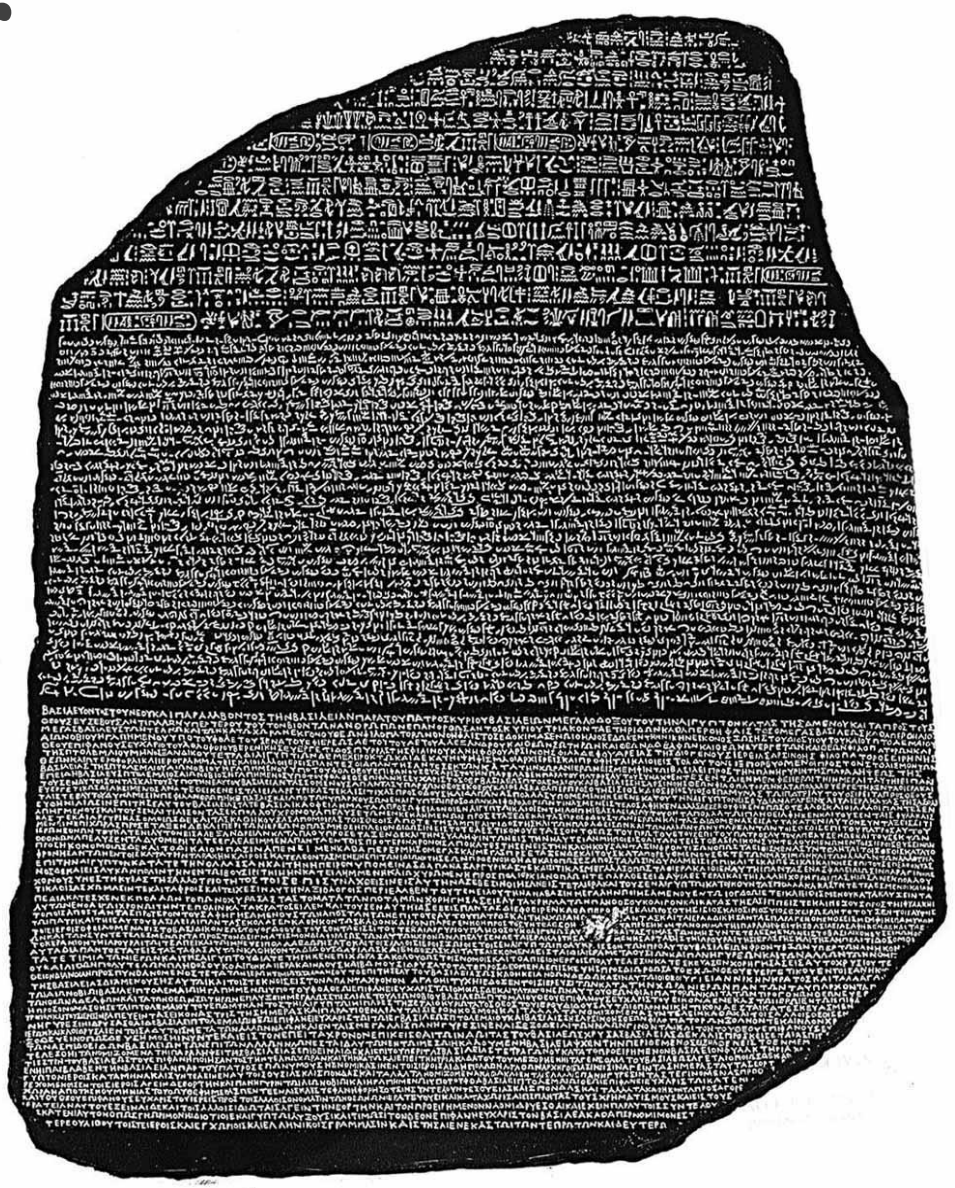
The computer age and the beginning of databases?

No, let's go further back before computers took over to see how society managed information persistently

Way too back?



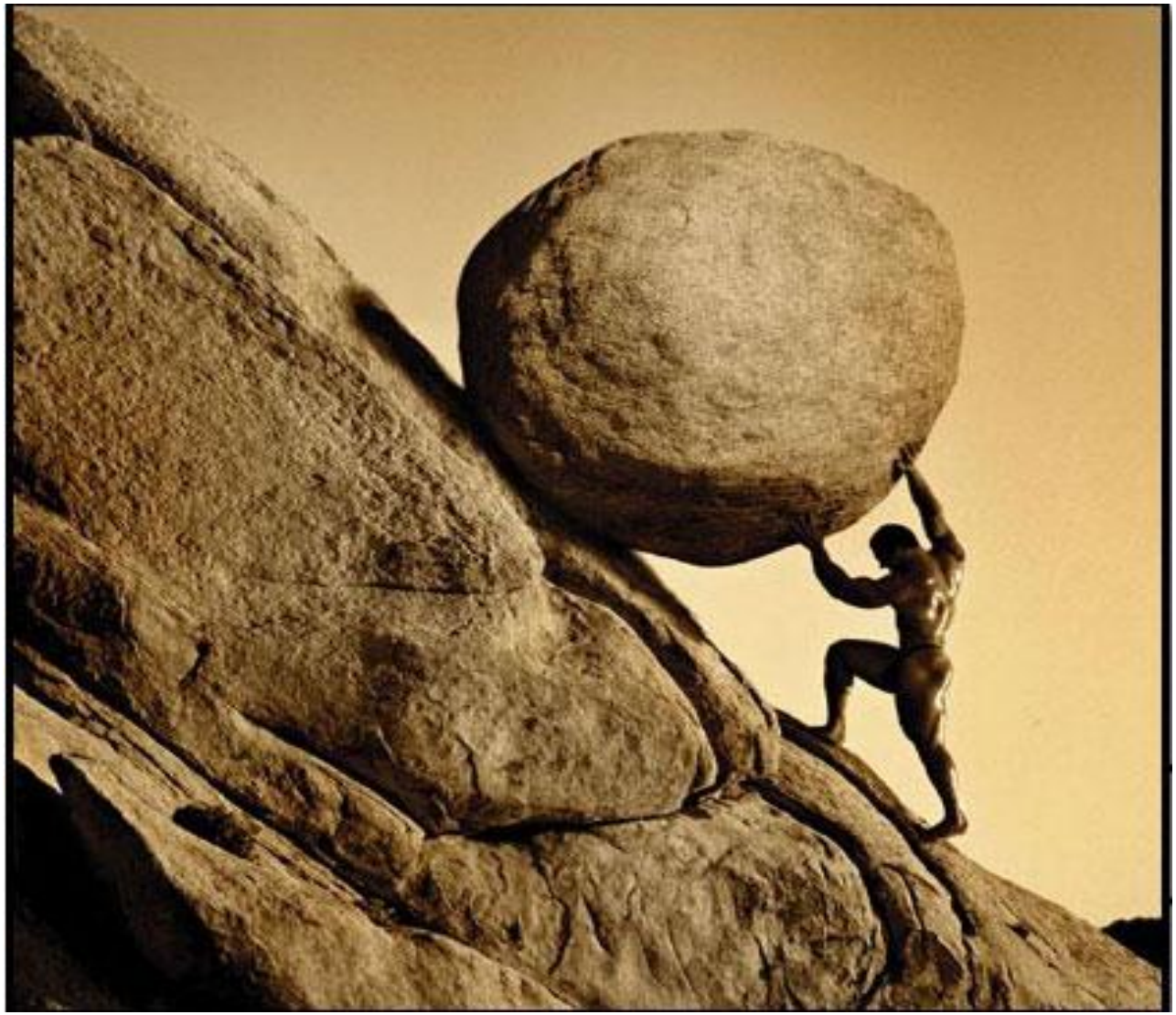
"Stone" was our first persistent database. Before that everything was "volatile" because in-memory only



Pros:

Cheap (it's free), very-very durable and always available

Cons: not exactly portable...

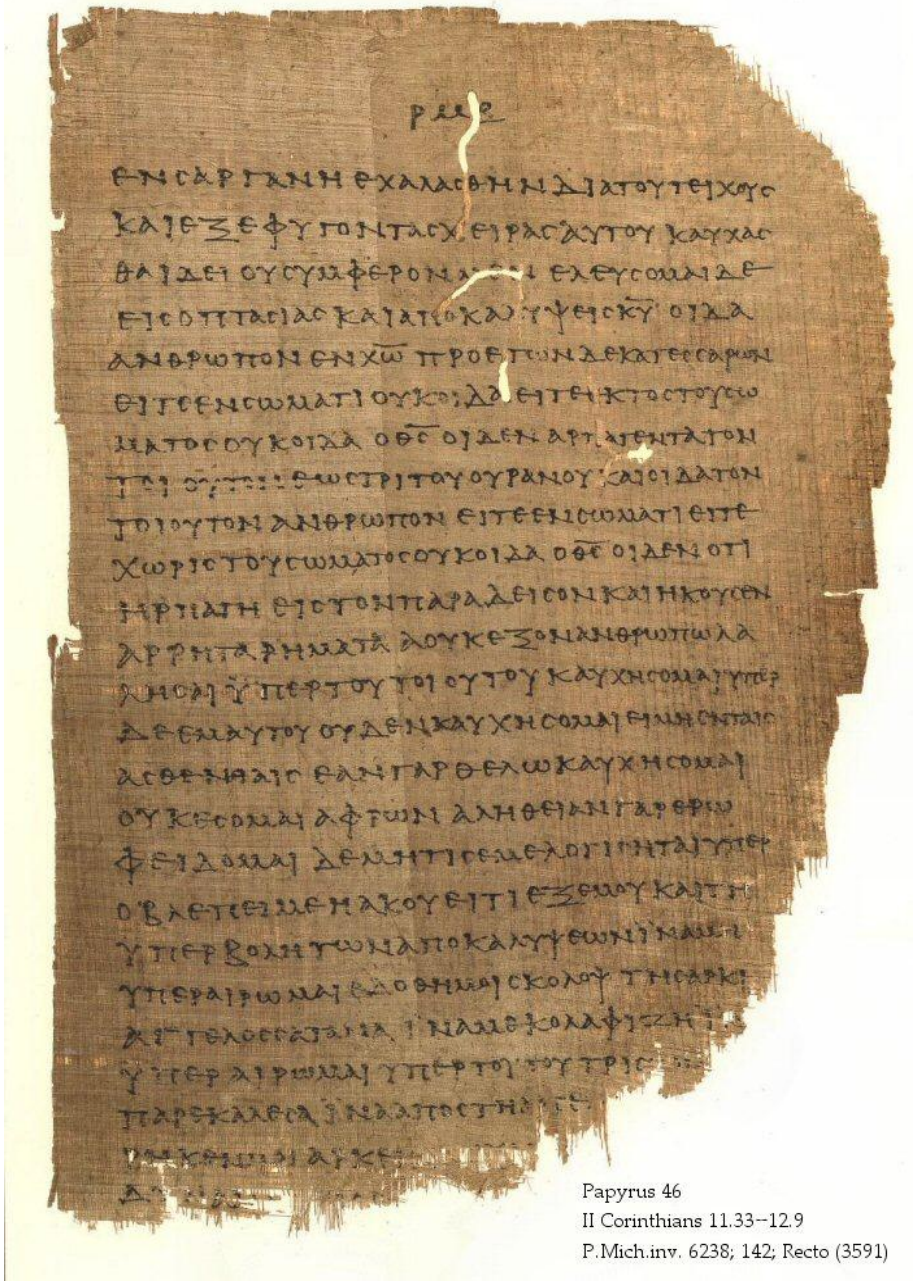


...And this kind of storage had the same problems of modern RDBMS

Slow inserts!

no Market Leader for this technology because stone is FREE and unlimited

So, the world needed a better technology: what next?



Papyrus 46
II Corinthians 11.33–12.9
P.Mich.inv. 6238; 142; Recto (3591)

Egypt, 5000 years ago

"Papyrus" was the database v. 2.0

Not so cheap 

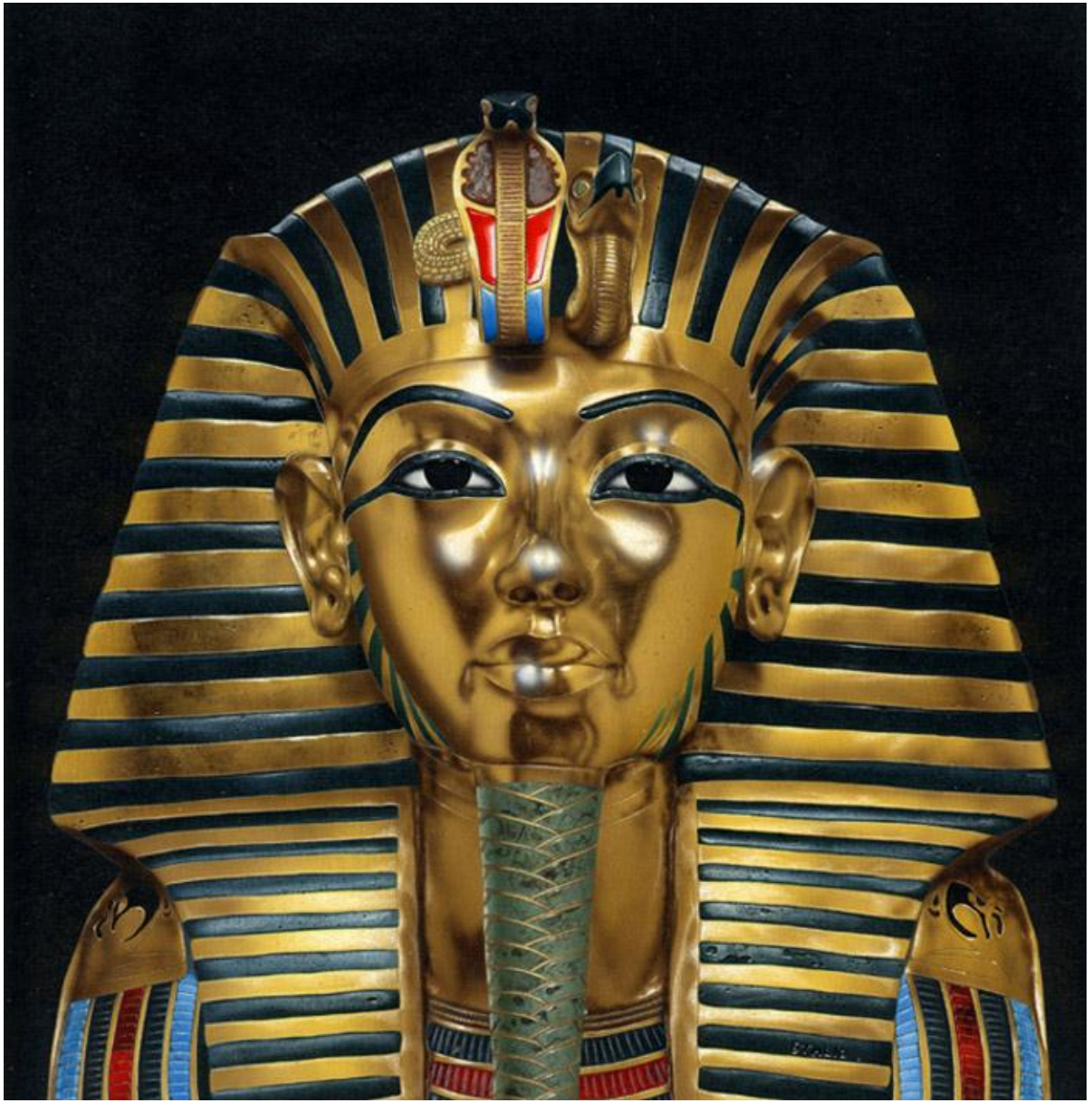
not so durable 

 but portable

This was the first

"mobile" market

Market Leader the Pharaoh ?



Wait a sec!

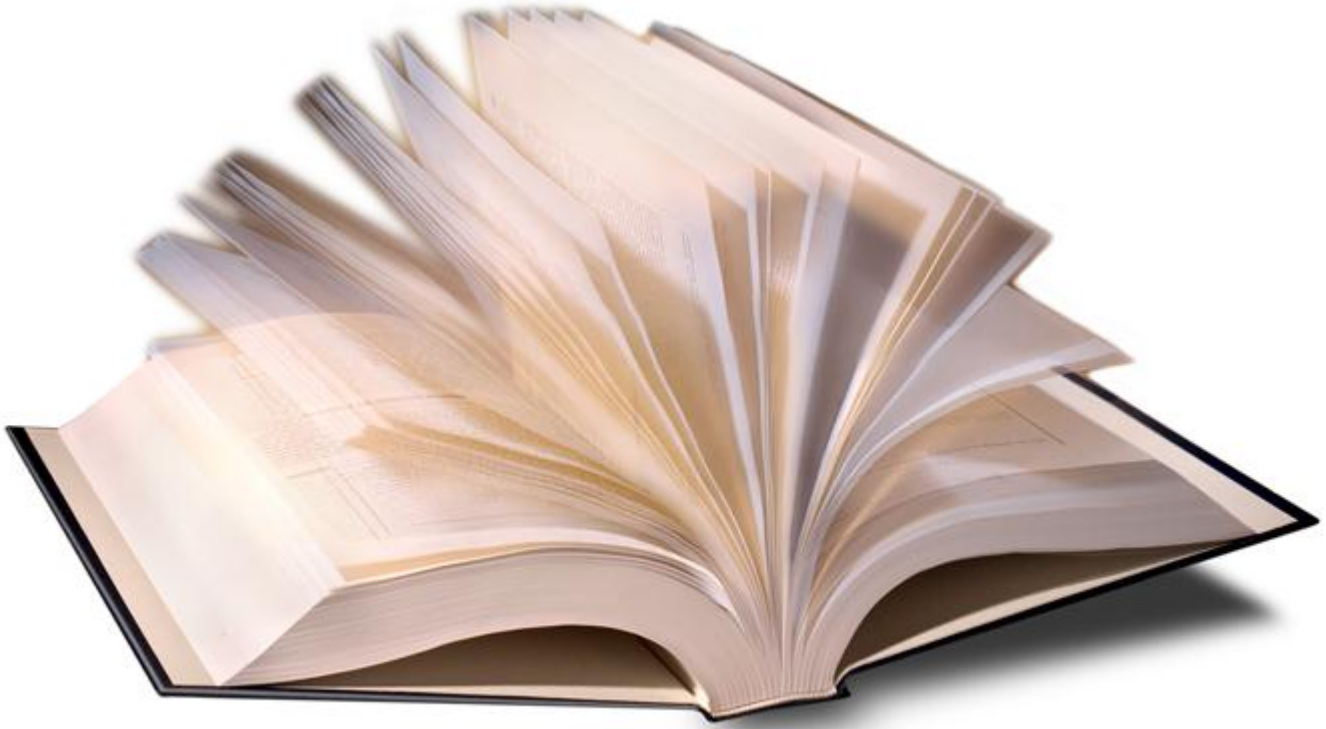
The Pharaoh Market Leader spoke about prophecies, exactly like an "Oracle" would...

(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 database v. 2.1 (minor version)



Easy to make

 copies,

not durable as 

stone, but

 portable

Users started to make **choices**:

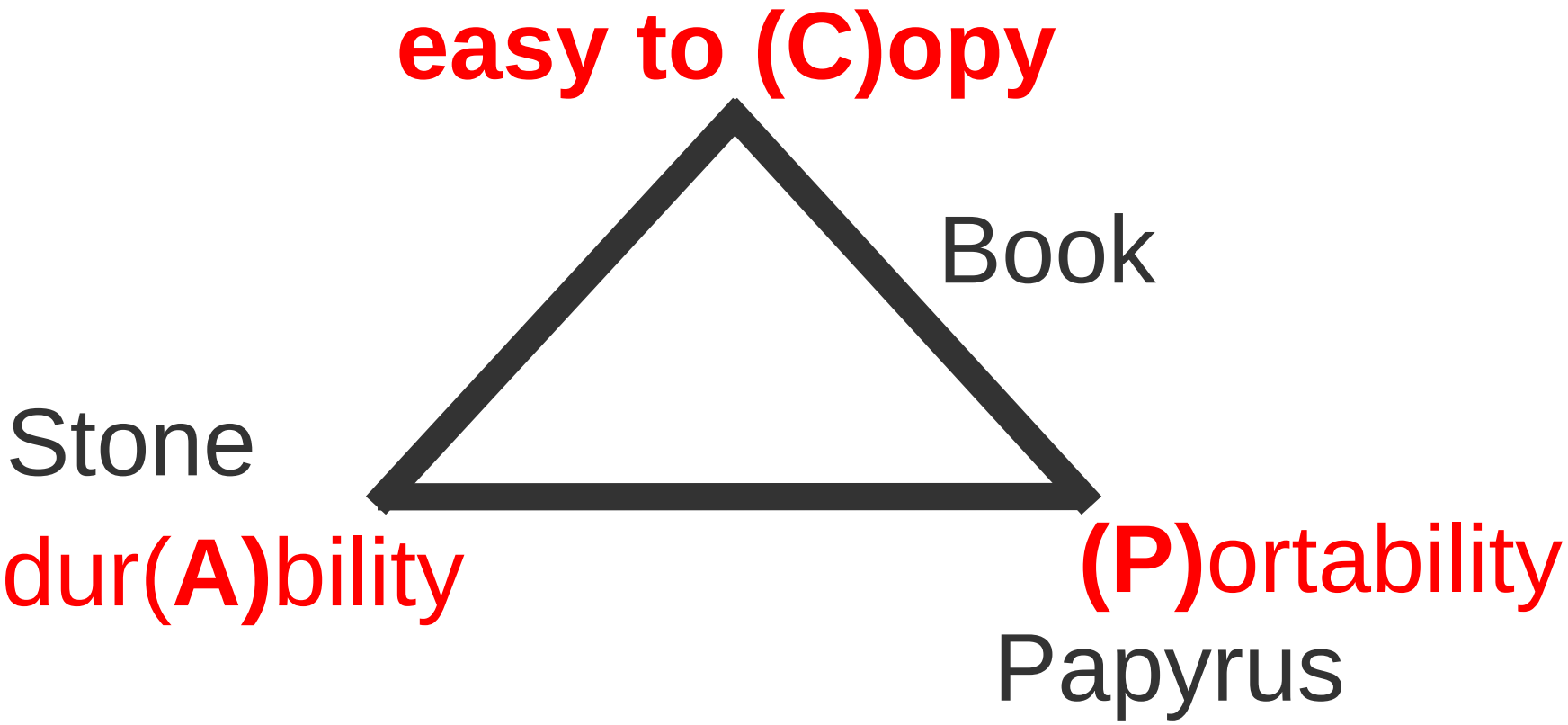
1) want something really durable?
Go with Stone
2) do you want something portable?
Papyrus
3) Need also copies? Books

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!

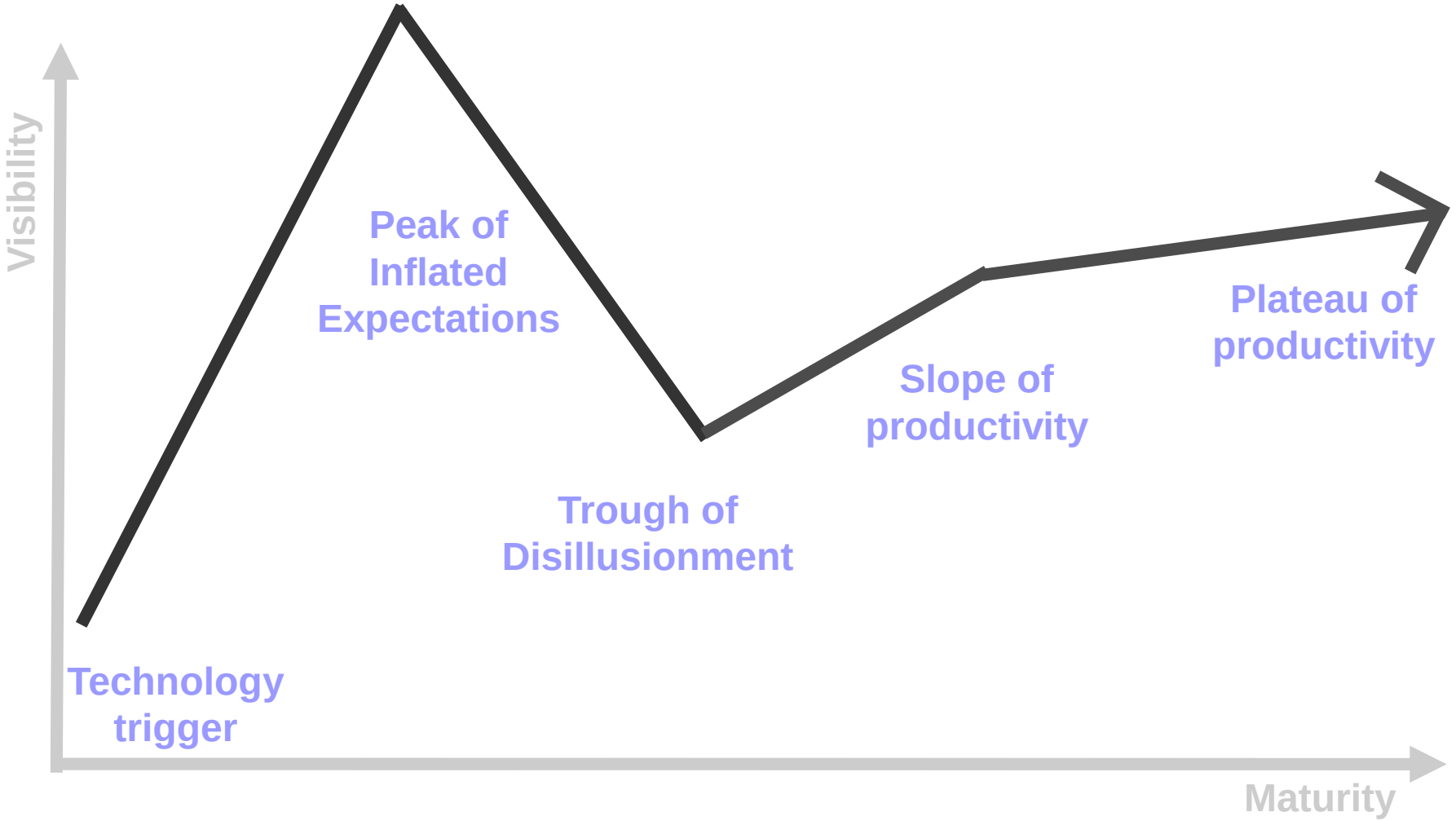
I can't believe: the origin of CAP theorem ?



We found some interesting similarities with the modern databases: history repeating itself!

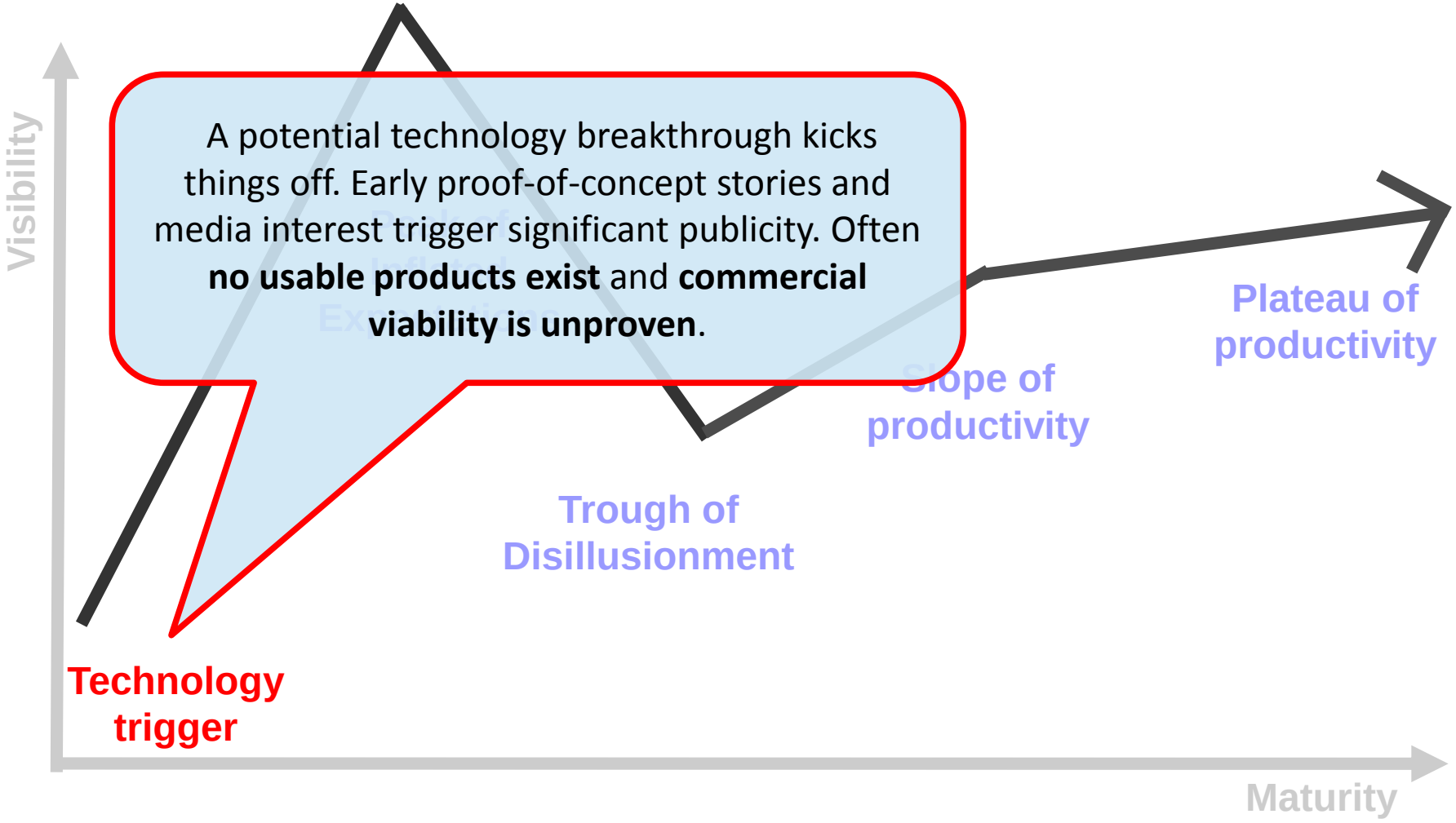
Interpreting Technology Hype

by Gartner



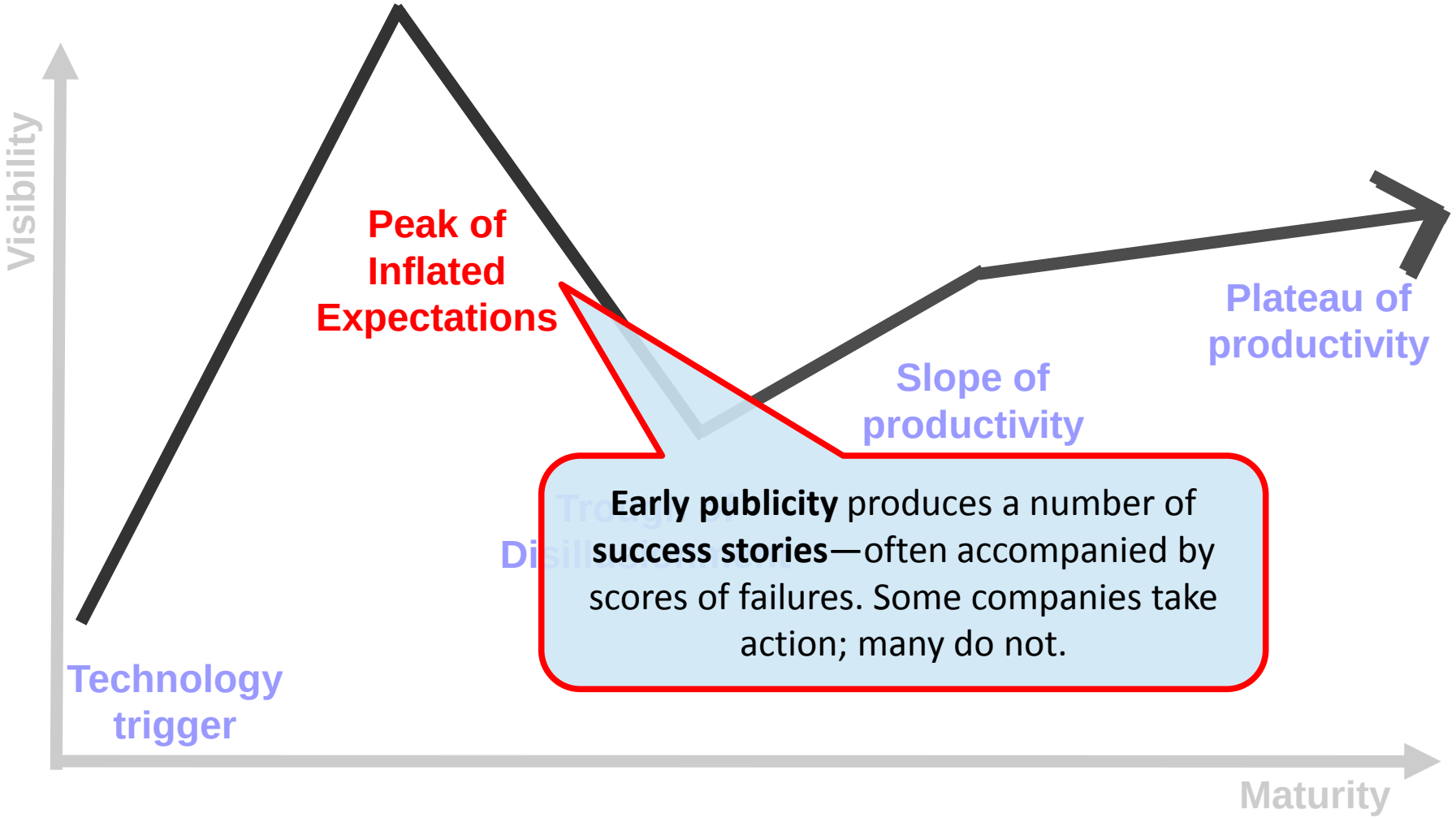
Interpreting Technology Hype

by Gartner



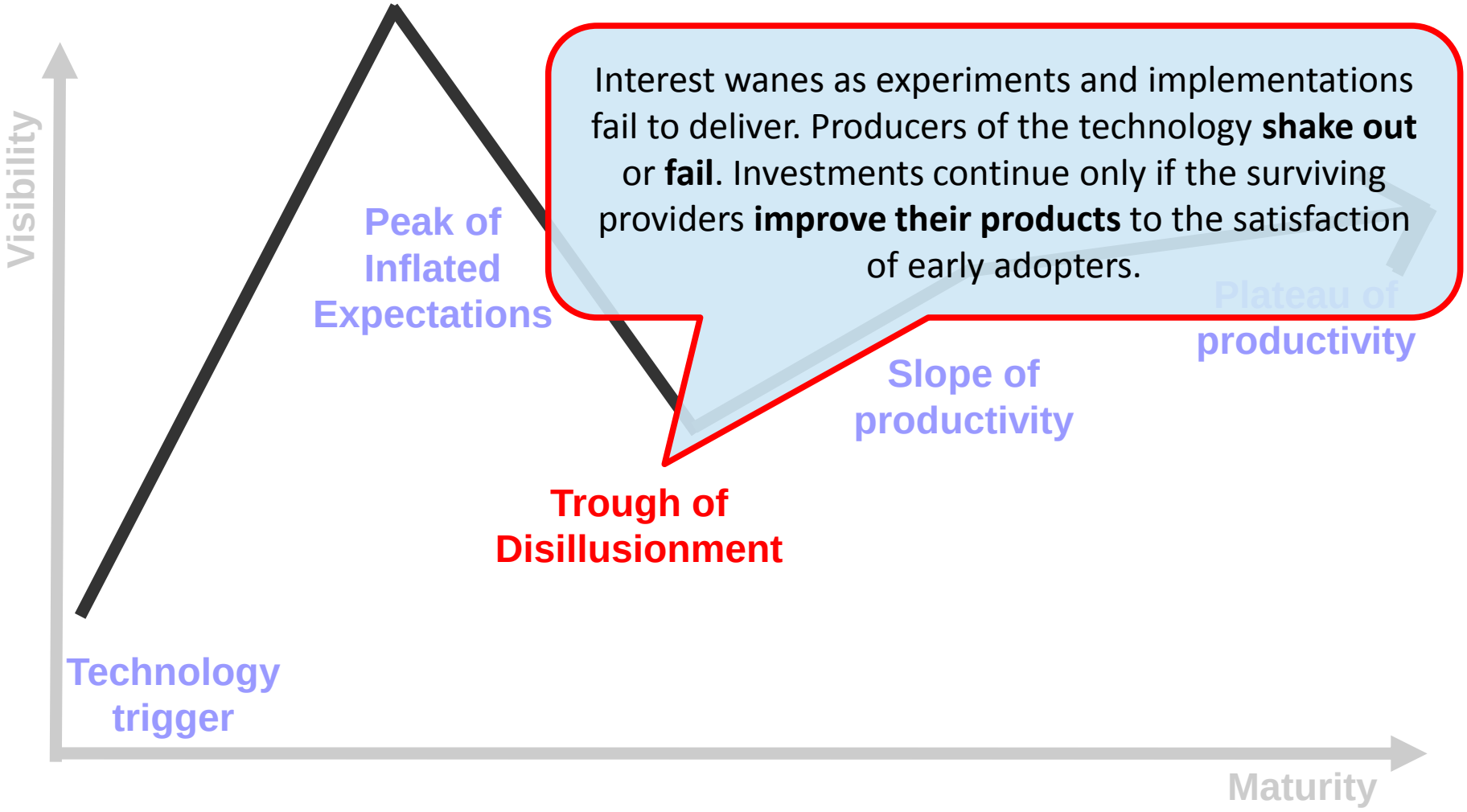
Interpreting Technology Hype

by Gartner



Interpreting Technology Hype

by Gartner



Bad stories from the trenches

"Goodbye, CouchDB"
May 10th, 2012 by Steven Hazel

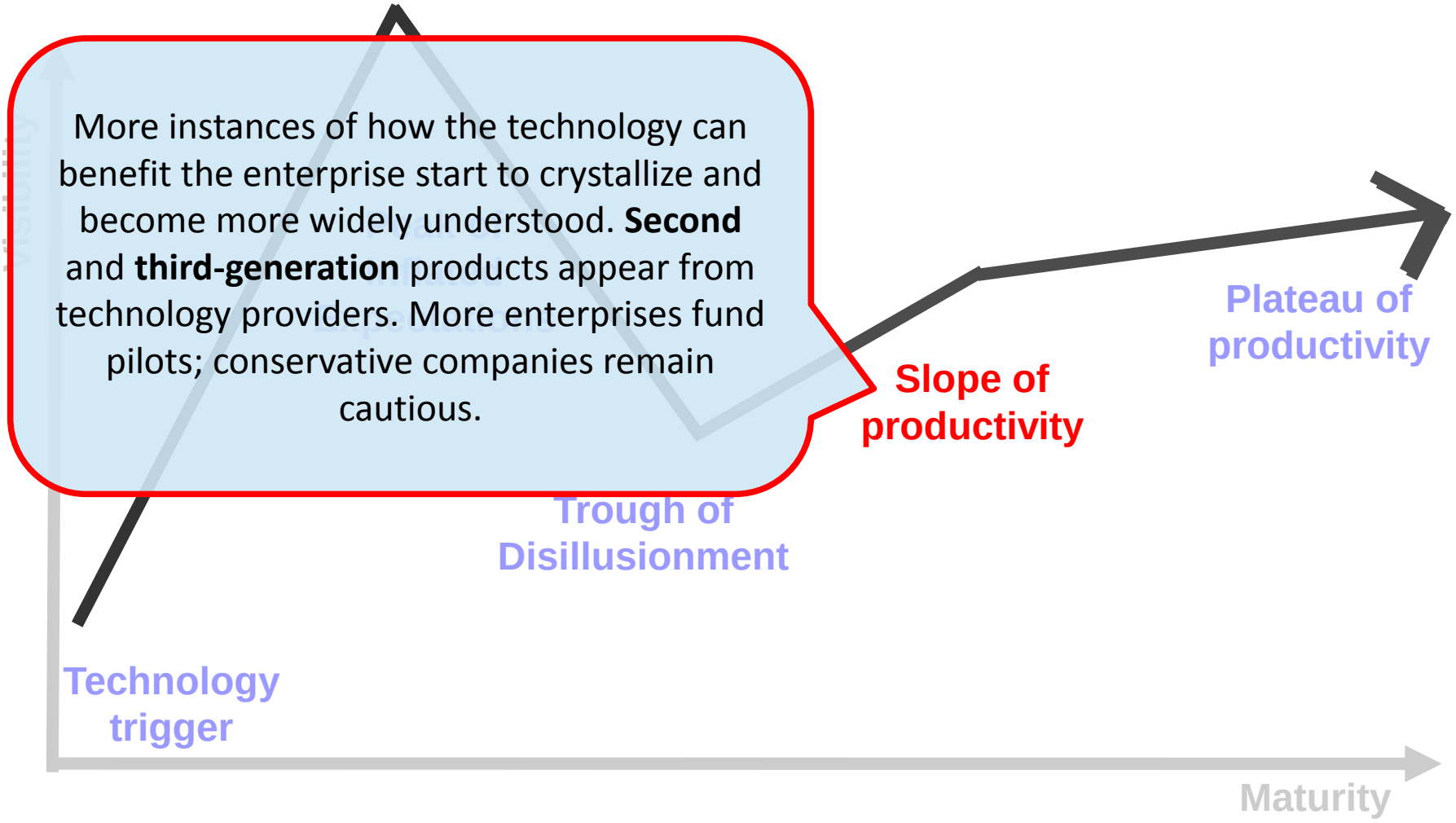
"Failing with MongoDB"
November 5, 2011 by Michael Schurter

"A year with MongoDB"
April 2012 on Kiip.me blog

"MongoDB: 9 months on"
11 May 2011 by Clueless Joe

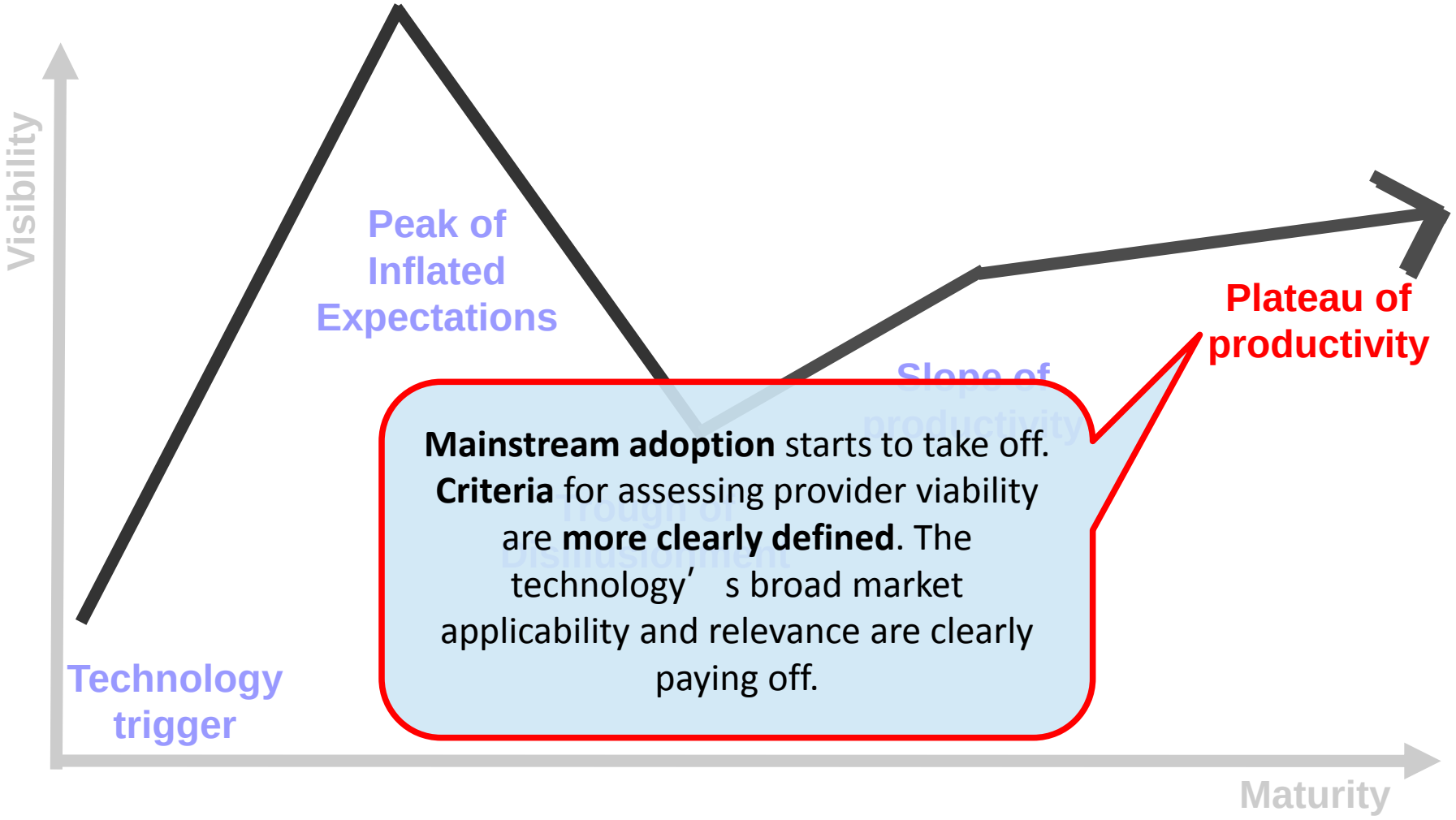
Interpreting Technology Hype

by Gartner



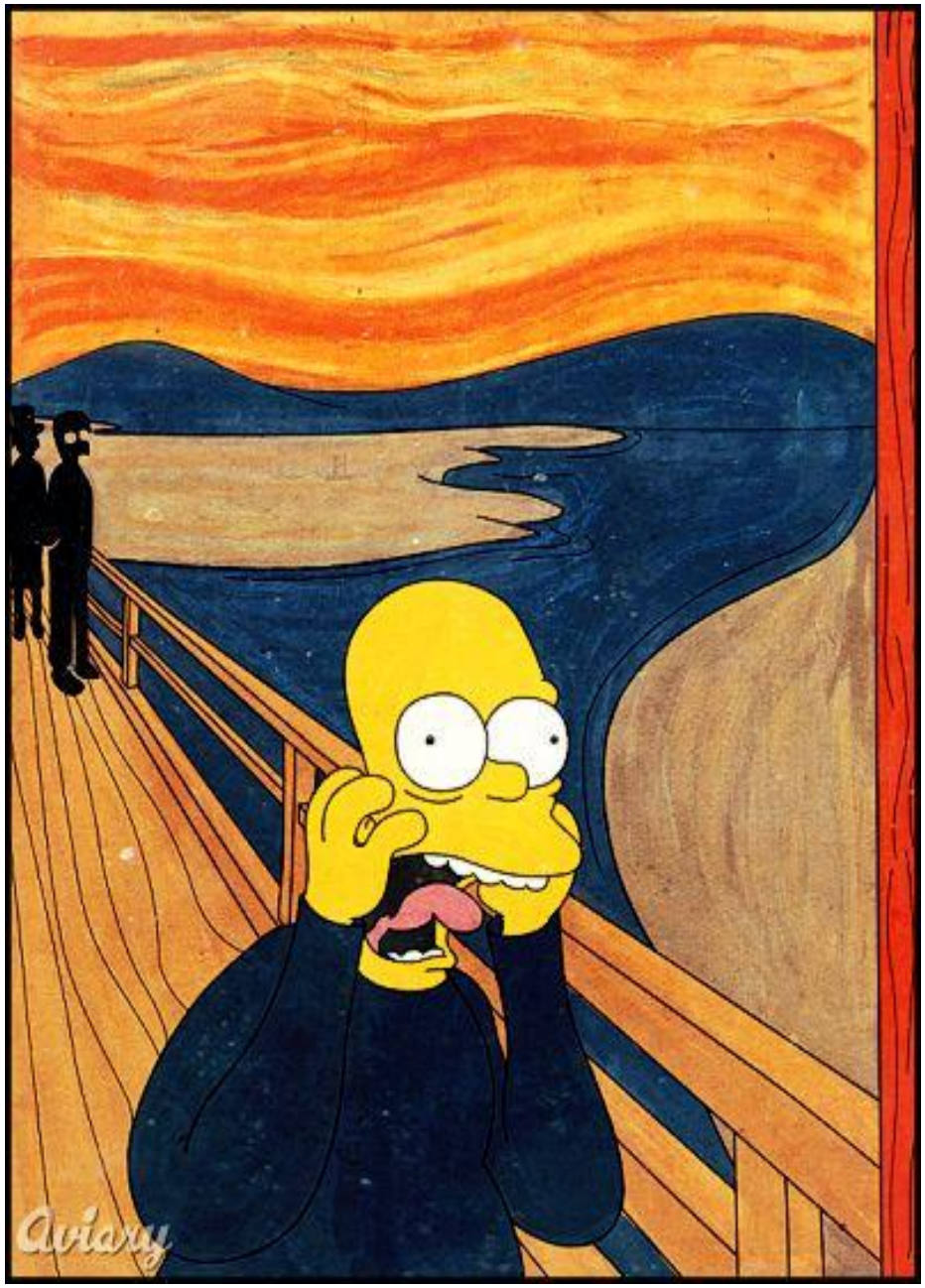
Interpreting Technology Hype

by Gartner

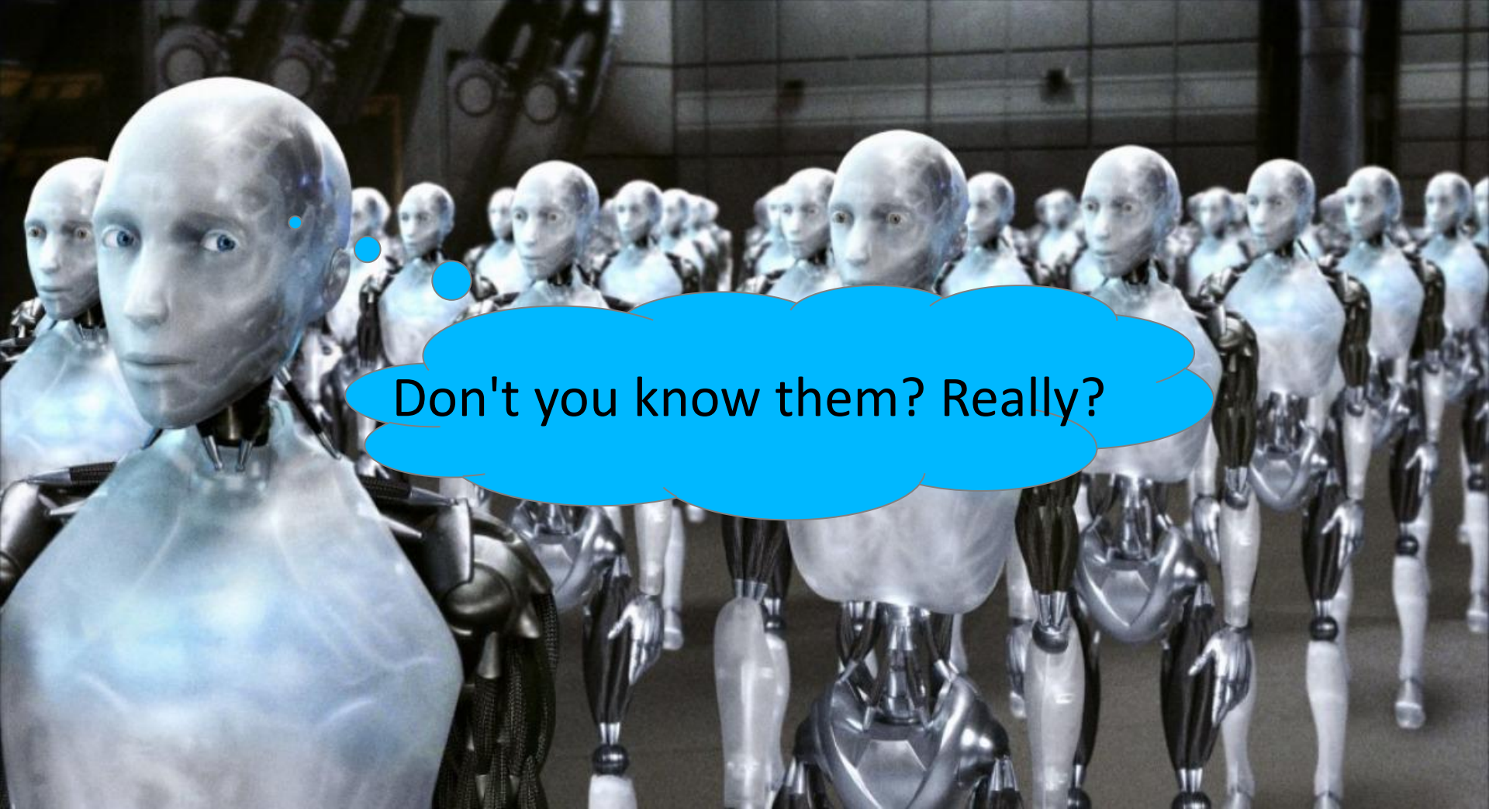


Interpreting Technology Hype

by Gartner

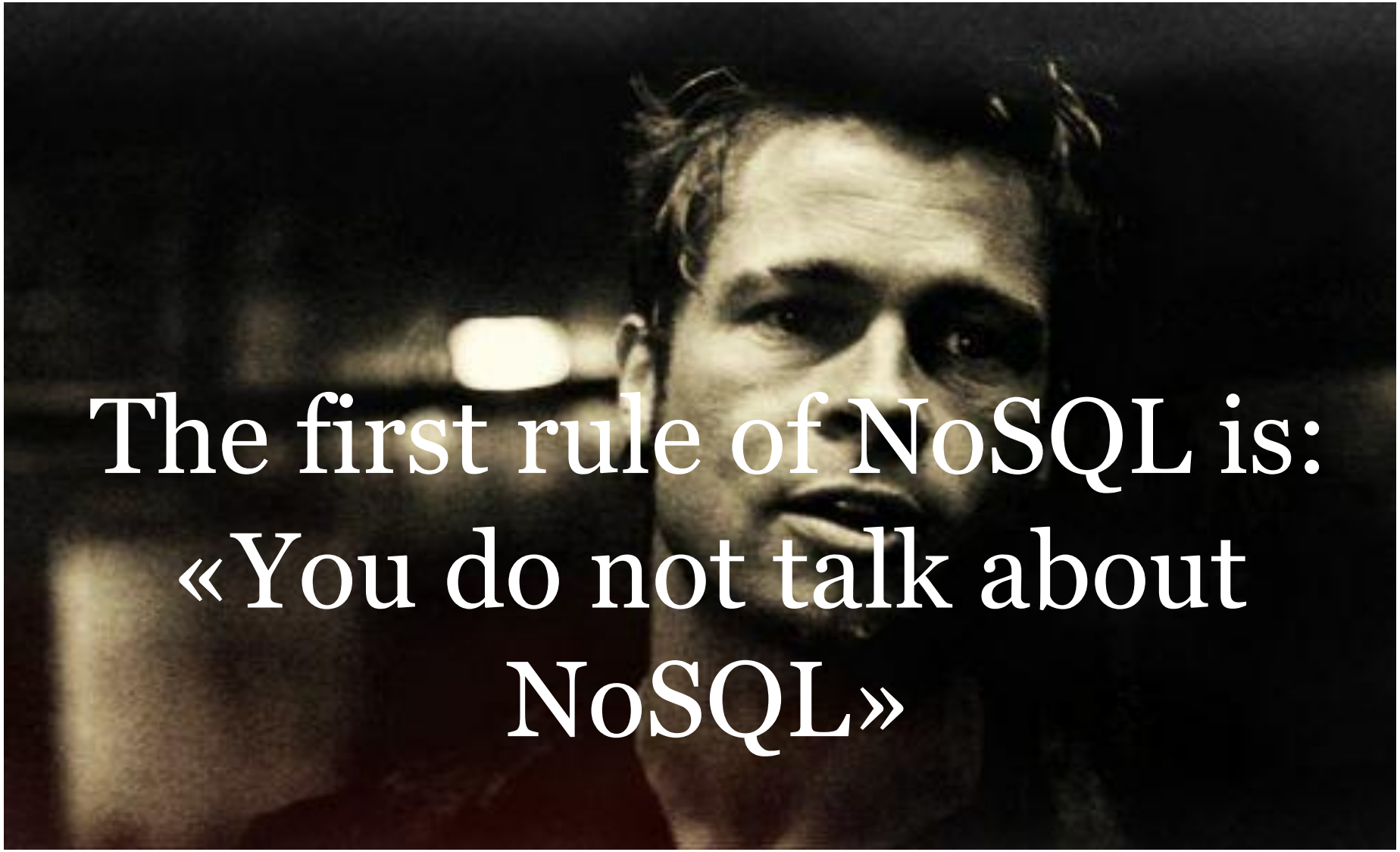


Why some users don't succeed using NoSQL?



Have you applied the 3 laws of ~~robotics~~ NoSQL?

NoSQL 1st law



NoSQL 2nd law

there is no “golden hammer”

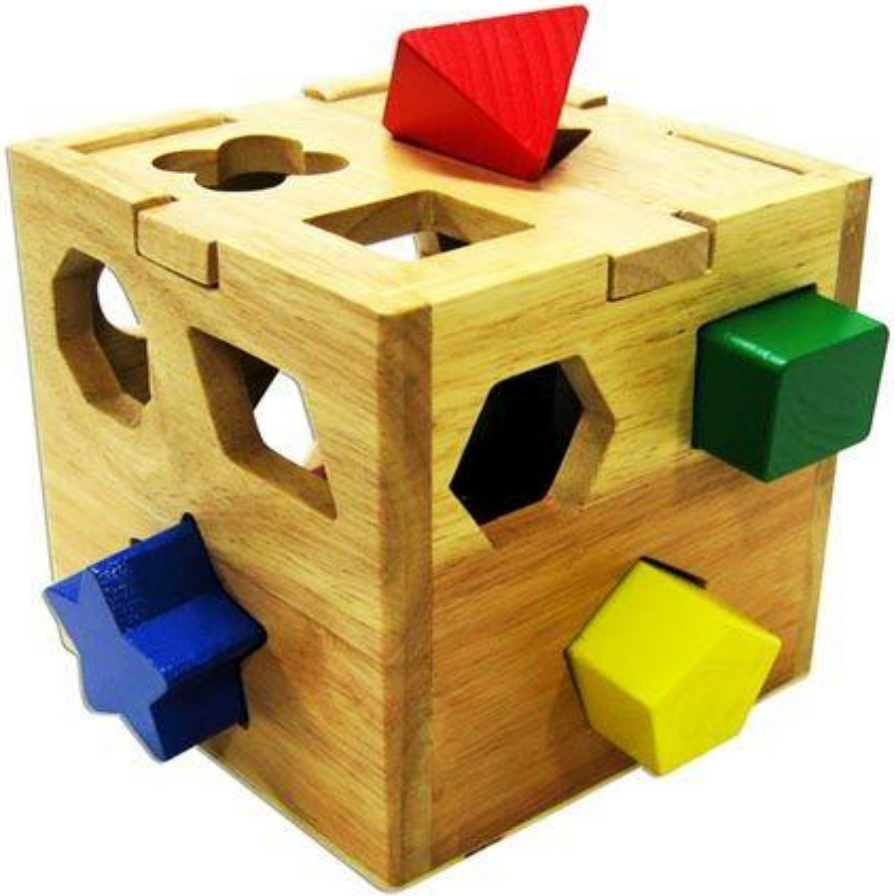
“if all you have is a **hammer**, everything looks like a **nail**”

NoSQL 2nd law

explained

If all you have is a		**everything looks like**
rdbms	→	tables
key value	→	keys and values
document	→	documents
graph db	→	vertexes and nodes

NoSQL 3rd law



“**one size** doesn't fit all”

NoSQL 3rd 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

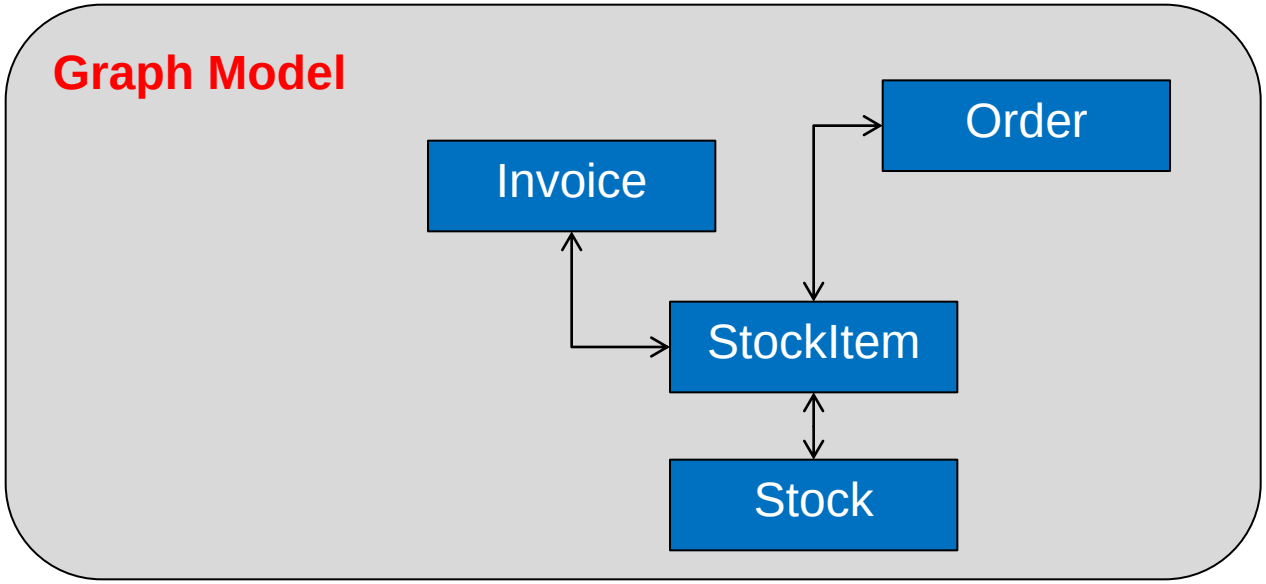
Multi-Model storage 2/2

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

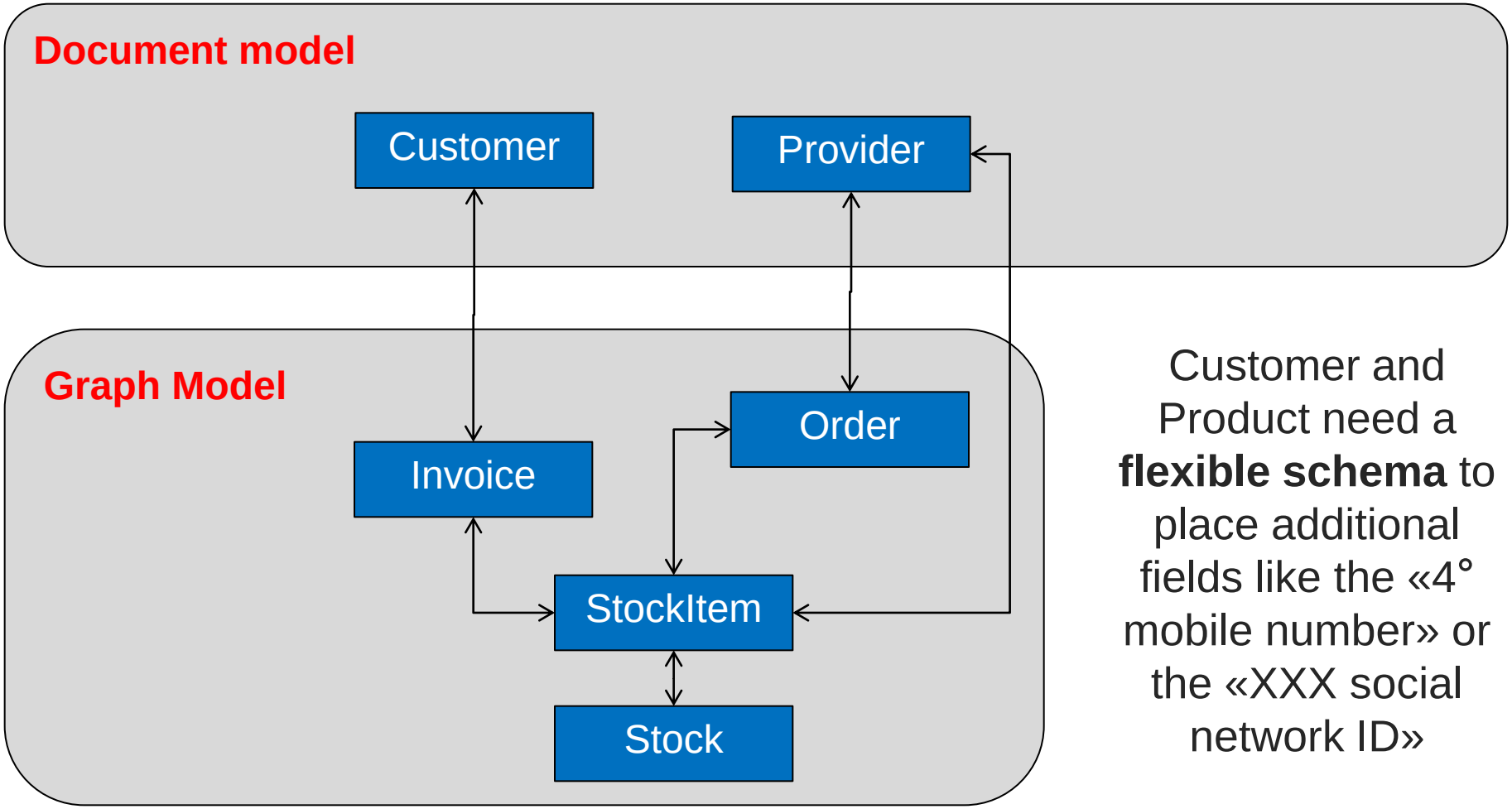
Only one product to know and manage

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!*

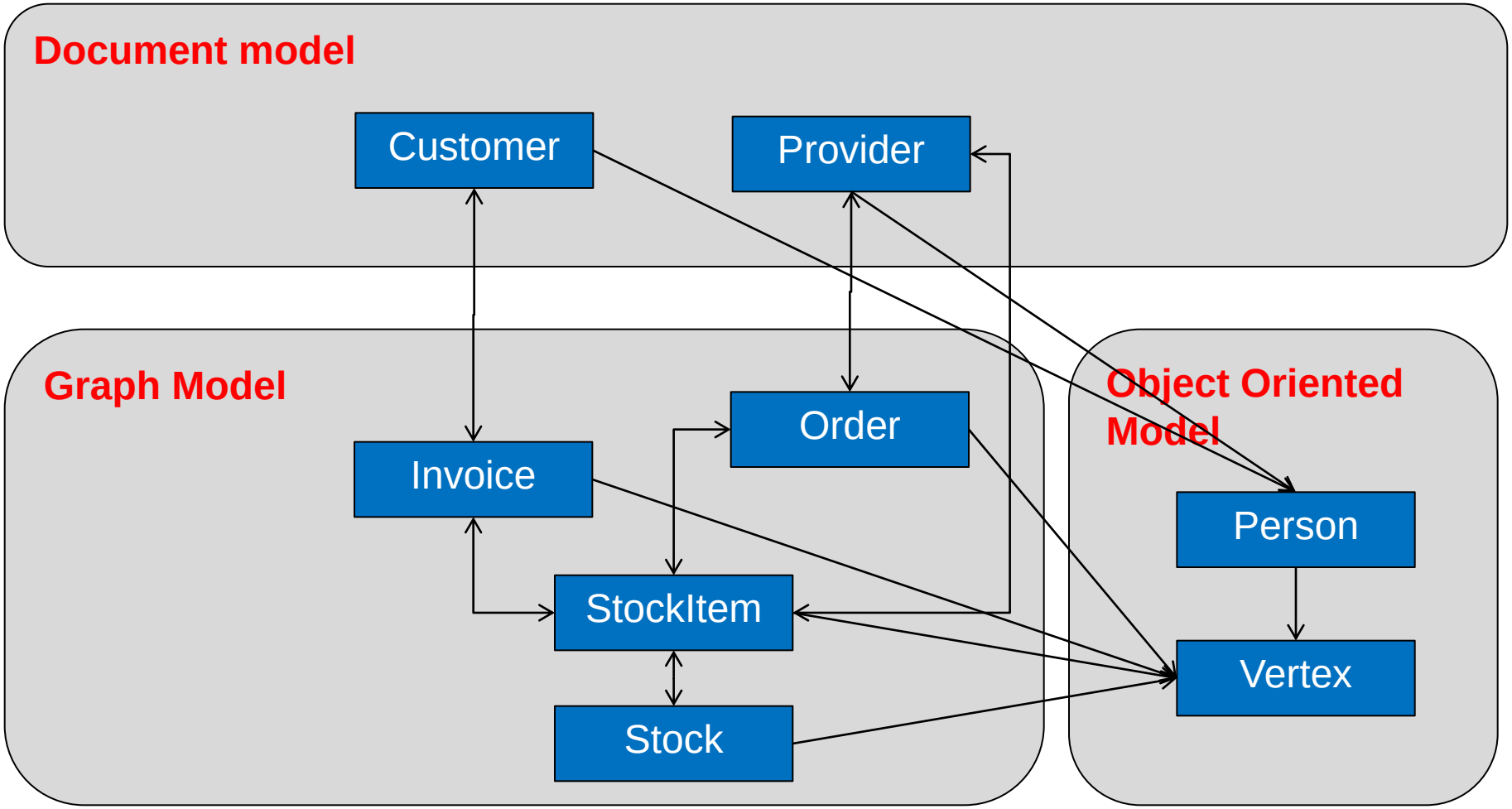


Multi-Model example 2/4

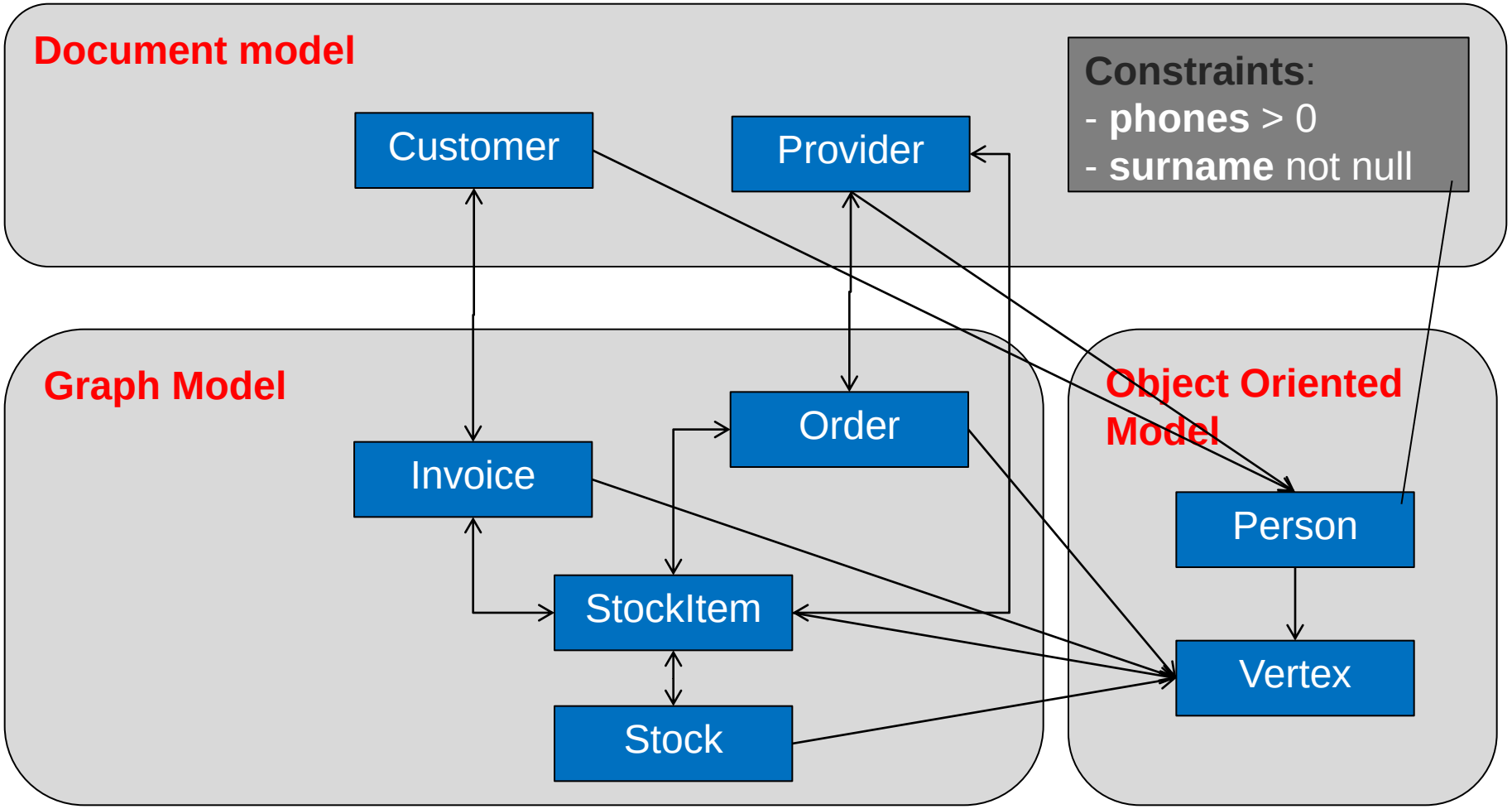


Customer and Product need a **flexible schema** to place additional fields like the «4° mobile number» or the «XXX social network ID»

Multi-Model example 3/4



Multi-Model example 4/4



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

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

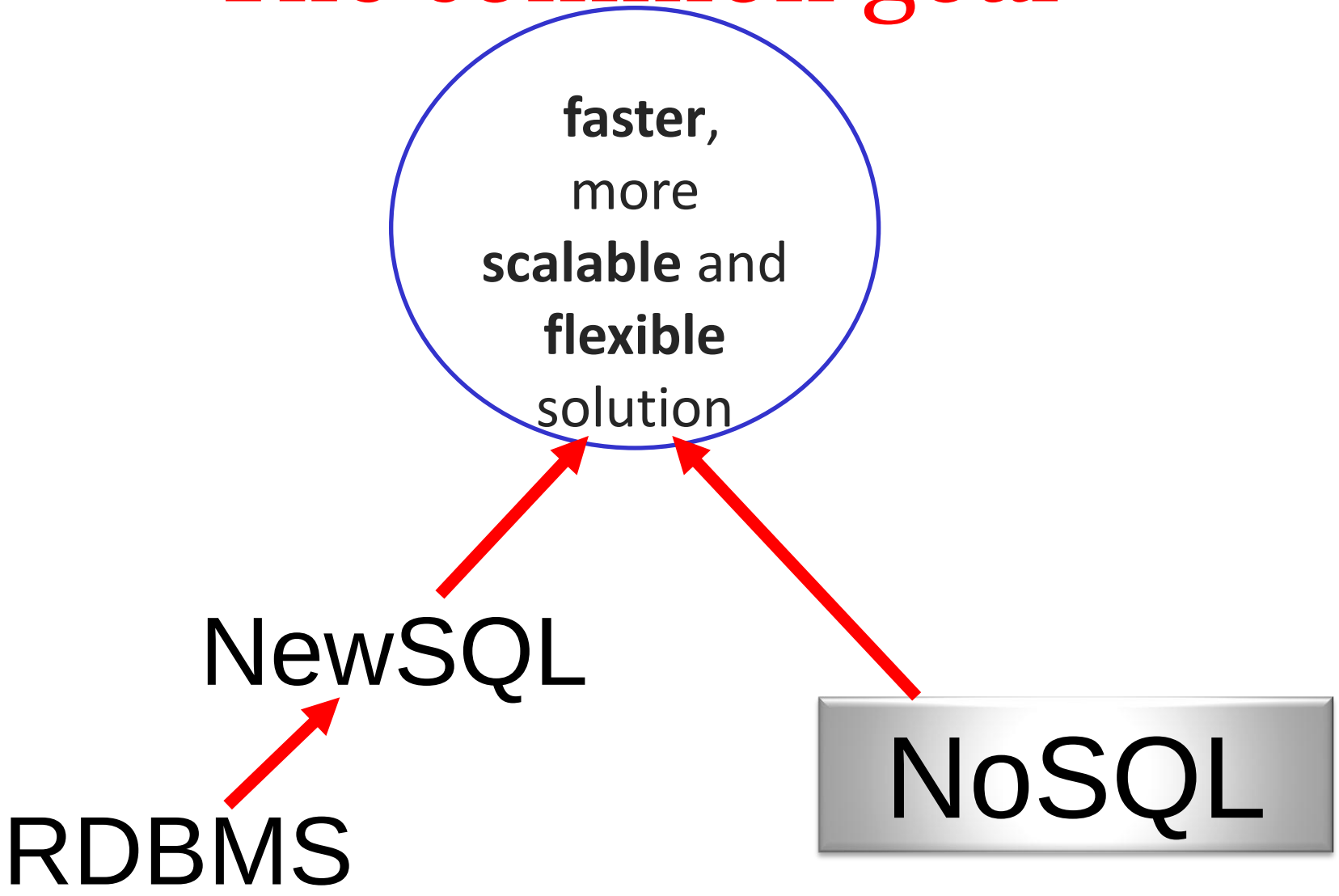
* persistence for memory-only db
* transaction or similar
* better locking to improve concurrency
* finer indexing systems

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



NoSQL
what
risks & **challenges**
with such scenario?

NoSQL Risks 1/2

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

NoSQL Risks 2/2

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

2nd day!

Many thanks!

Luca Garulli

Author of

Document-Graph NoSQL
Open Source project

CEO at

Ltd, London UK

www.twitter.com/lgarulli

Luca Garulli

Author of OrientDB®

Document-Graph NoSQL Open Source project

CEO at

Ltd, London UK

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

www.twitter.com/lgarulli