POLYGLOT PERSISTENCE FOR ADDRESSING **DATA MANAGEMENT** ON THE CLOUD

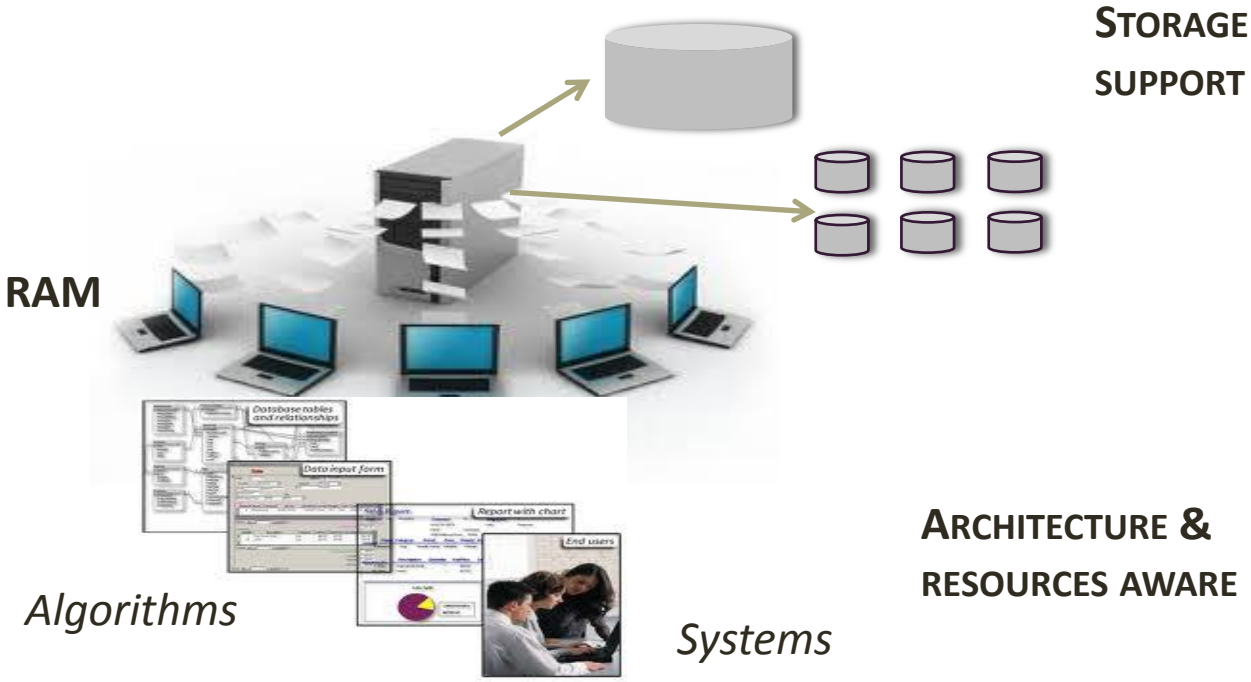
Genoveva Vargas-Solar, CNRS, LIG-LAFMIA, Juan Carlos Castrejon, U. de Grenoble,

Christine Collet, Grenoble INP, Rafael Lozano, ITESM-CCM

Genoveva.Vargas@imag.fr

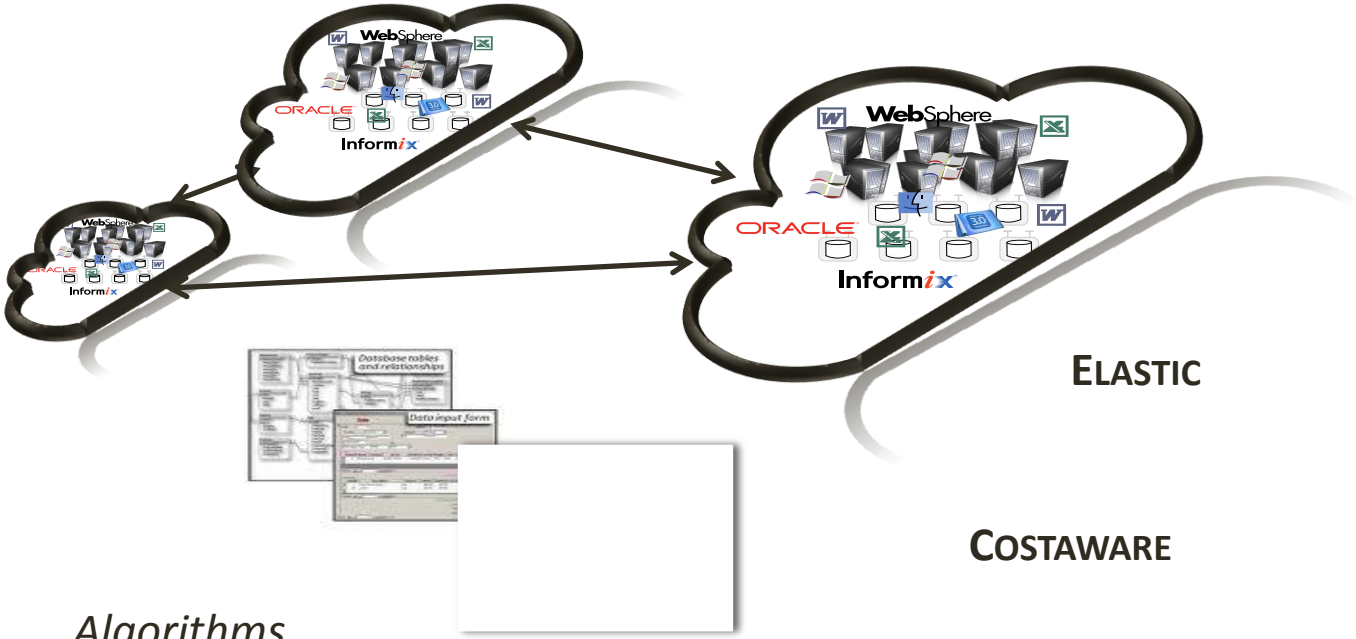
http://vargas-solar.imag.fr

DATA MANAGEMENT WITH RESOURCES CONSTRAINTS



Efficiently manage and exploit data sets according to given specific storage, memory and computation resources

DATA MANAGEMENT WITHOUT RESOURCES CONSTRAINTS



Costly manage and exploit data sets according to unlimited storage, memory and computation resources

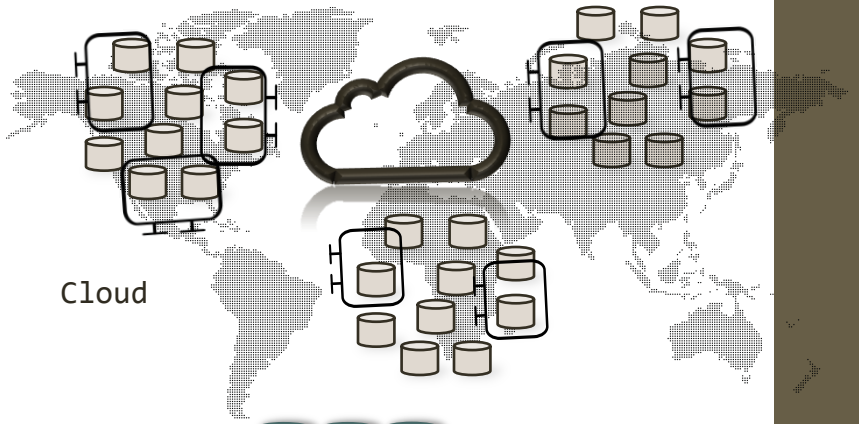
DEALING WITH HUGE AMOUNTS OF DATA

Yota 10^{24}

Zetta 10^{21}

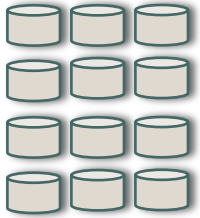
Exa 10^{18}

Peta 10^{15}



Cloud

RAID



Disk

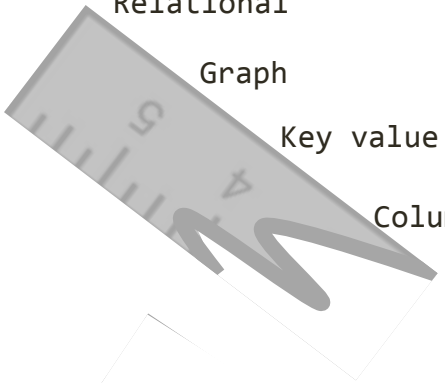
DEALING WITH HUGE AMOUNTS OF DATA

Yota 10^{24}

Zetta 10^{21}

Exa 10^{18}

Peta 10^{15}

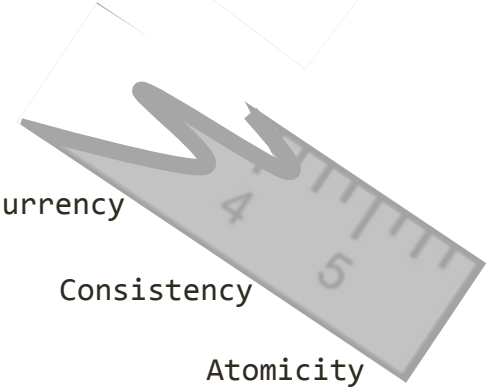


Relational

Graph

Key value

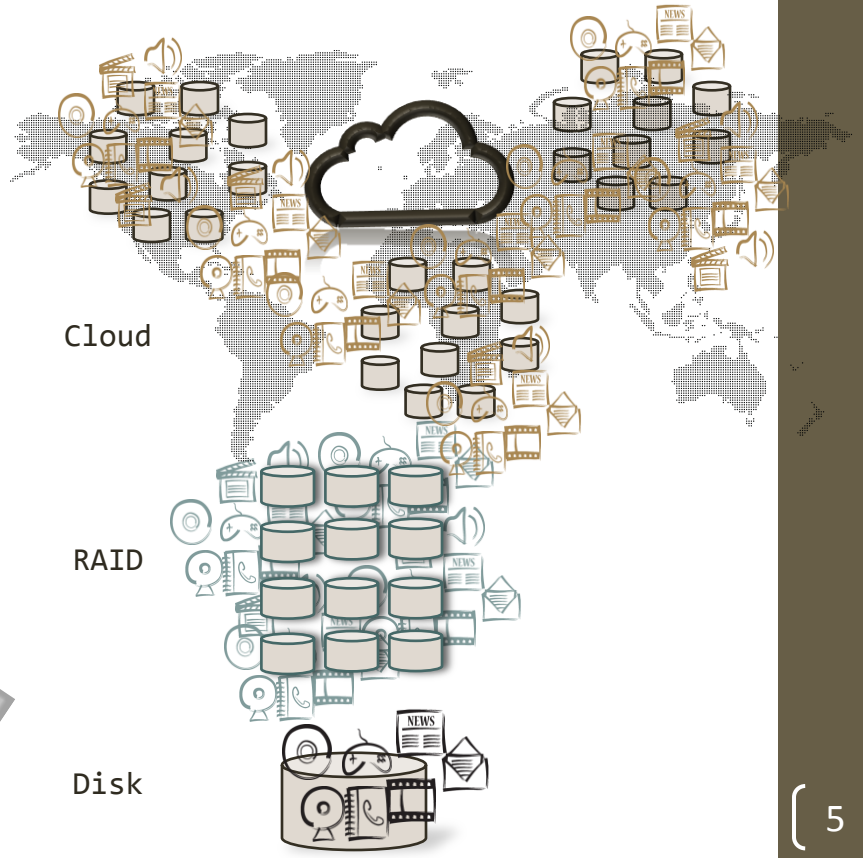
Columns



Concurrency

Consistency

Atomicity



Cloud

RAID

Disk

SQL has Ruled for two decades

- **Store persistent data**

  Storing large amounts of data on disk, while allowing applications to grab the bits they need through queries

- **Application Integration**

  Many applications in an enterprise need to share information. By getting all applications to use the database, we ensure all these applications have consistent, up-to-date data

- **Mostly Standard**

  The relational model is widely used and understood. Interaction with the database is done with SQL, which is a (mostly) standard language. This degree of standardization is enough to keep things familiar so people don't need to learn new things

- **Concurrency Control**

  Many users access the same information at the same time. Handling this concurrency is difficult to program, so databases provide transactions to help ensure consistent interaction.

- **Reporting**

  SQL's simple data model and standardization has made it a foundation for many reporting tools

All this supported by Big Database Vendors and the separation of the DBA profession.

but SQL's dominance is cracking

Relational databases are designed to run on a single machine, so to scale, you need buy a bigger machine

But it's cheaper and more effective to scale horizontally by buying lots of machines.

The machines in these large clusters are individually unreliable, but the overall cluster keeps working even as machines die - so the overall cluster is reliable.

The "cloud" is exactly this kind of cluster, which means relational databases don't play well with the cloud.

The rise of web services provides an effective alternative to shared databases for application integration, making it easier for different applications to choose their own data storage.

Google and Amazon were both early adopters of large clusters, and both eschewed relational databases.

Google → **Bigtable**

Amazon → **Dynamo**

Their efforts have been a large inspiration to the NoSQL community

so now we have NoSQL databases

There is no standard definition of what NoSQL means. The term began with a workshop organized in 2009, but there is much argument about what databases can truly be called NoSQL.

But while there is no formal definition, there are some common characteristics of NoSQL databases

- they don't use the relational data model, and thus don't use the SQL language
- they tend to be designed to run on a cluster
- they tend to be Open Source
- they don't have a fixed schema, allowing you to store any data in any record

examples include

We should also remember Google's Bigtable and Amazon's SimpleDB. While these are tied to their host's cloud service, they certainly fit the general operating characteristics

so this means we can

Reduce Development Drag

A lot of effort in application development is tied up in working with relational databases. Although Object/ Relational Mapping frameworks have eased the load, the database is still a significant source of developer hours. Often we can reduce this effort by choosing an alternative database that's more suited to the problem domain.

We often come across projects who are using relational databases because they are the default, not because they are the best choice for the job. Often they are paying a cost, in developer time and execution performance, for features they do not use.

Embrace Large Scale

The large scale clusters that we can support with NoSQL databases allow us to store larger datasets (people are talking about petabytes these days) to process large amounts of analytic data.

Alternative data models also allow us to carry out many tasks more efficiently, allowing us to tackle problems that we would have balked at when using only relational databases

ROADMAP

POLYGLOT PERSISTENCE

PUTTING POLYGLOT PERSISTENCE IN PRACTICE

CONCLUSION AND OUTLOOK

POLYGLOT PERSISTENCE

- ***Polyglot Programming:*** applications should be written in a mix of languages to take advantage of different languages are suitable for tackling different problems
- ***Polyglot persistence***: any decent sized enterprise will have a variety of different data storage technologies for different kinds of data
  - a new strategic enterprise application should no longer be built assuming a relational persistence support
  - the relational option might be the right one - but you should seriously look at other alternatives

M. Fowler and P. Sadalage. *NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence*. Pearson Education, Limited, 2012.

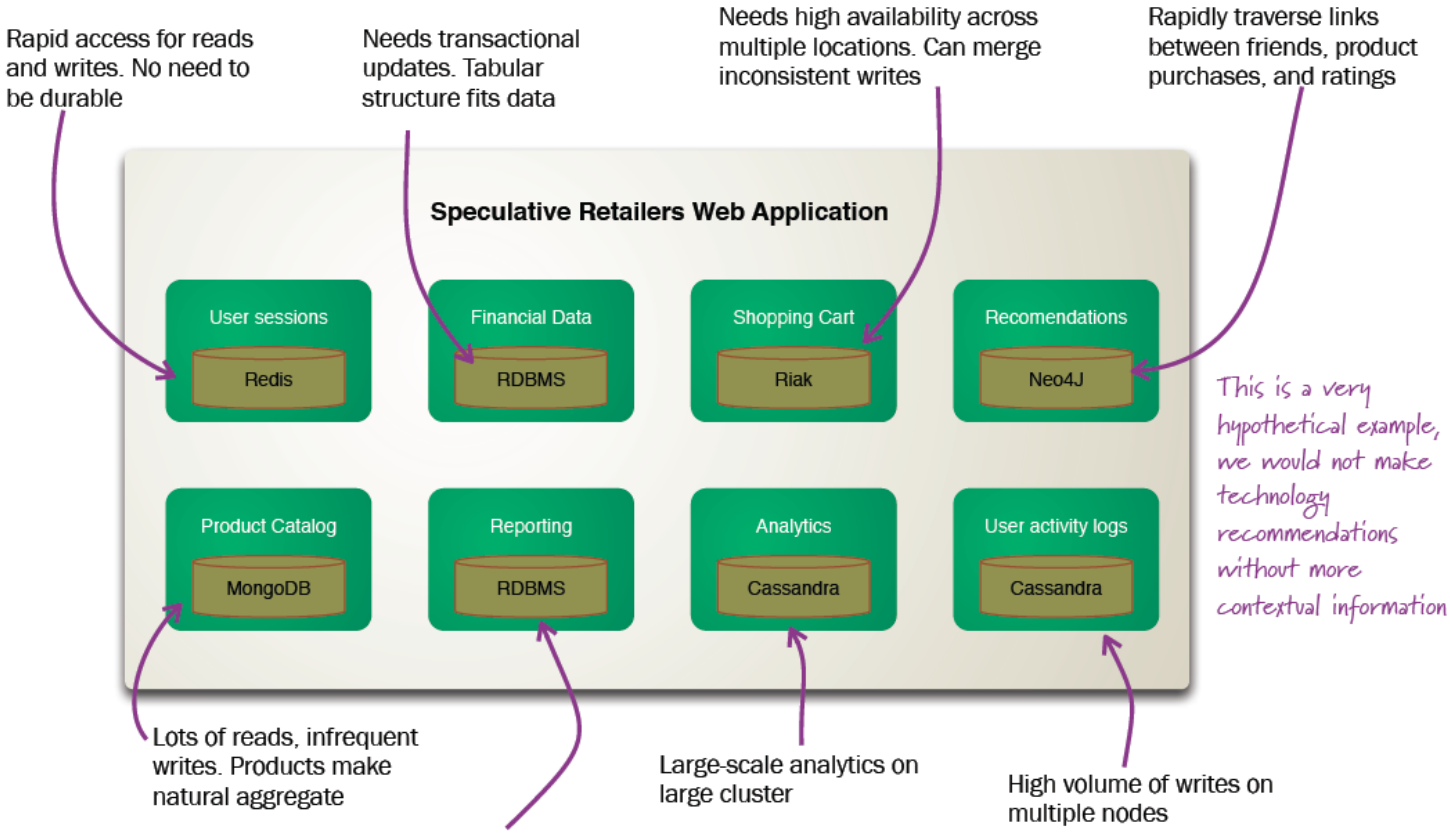
Polyglot Persistence

using multiple data storage technologies, chosen based upon the way data is being used by individual applications. Why store binary images in relational database, when there are better storage systems?

Polyglot persistence will occur over the enterprise as different applications use different data storage technologies. It will also occur within a single application as different parts of an application's data store have different access characteristics.

http://martinfowler.com/bliki/PolyglotPersistence.html

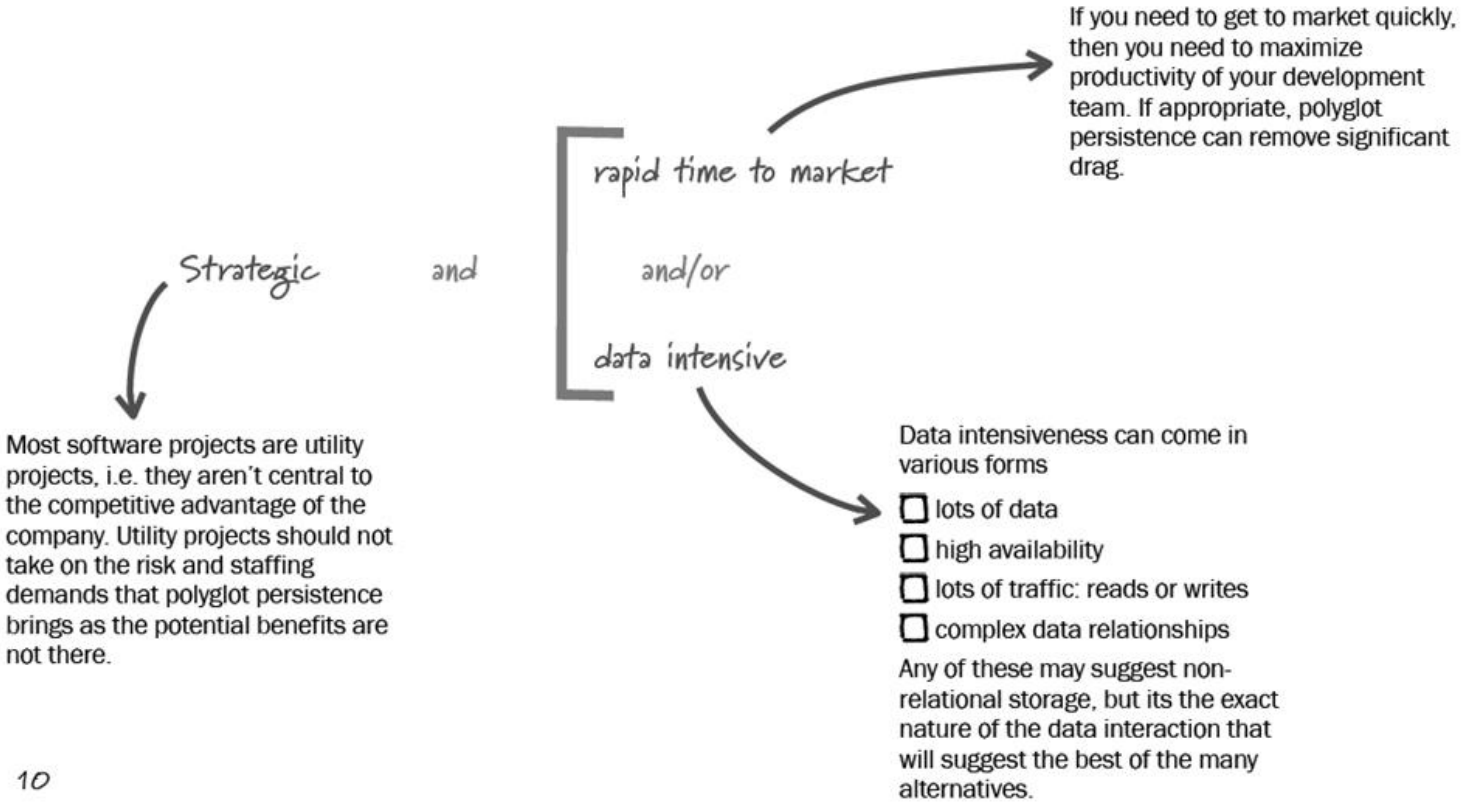
what might Polyglot Persistence look like?



When is polyglot persistence pertinent?

- Application essentially composing and serving web pages
  - They only looked up page elements by ID, they had no need for transactions, and no need to share their database
  - A problem like this is much better suited to a key-value store than the corporate relational hammer they had to use
- Scaling to lots of traffic gets harder and harder to do with vertical scaling
  - Many NoSQL databases are designed to operate over clusters
  - They can tackle larger volumes of traffic and data than is realistic with a single server

what kinds of projects are candidates for polyglot persistence?



polyglot persistence provides lots of new opportunities for enterprises

i.e. **problems**

- **Decisions**

  We have to decide what data storage technology to use, rather than just go with relational

- **Immaturity**

  NoSQL tools are still young, and full of the rough edges that new tools have.

  Furthermore since we don't have much experience with them, we don't know how to use them well, what the good patterns are, and what gotchas are lying in wait.

- **Organizational Change**

  How will our data groups react to this new technology?

- **Dealing with Eventual Consistency Paradigm**

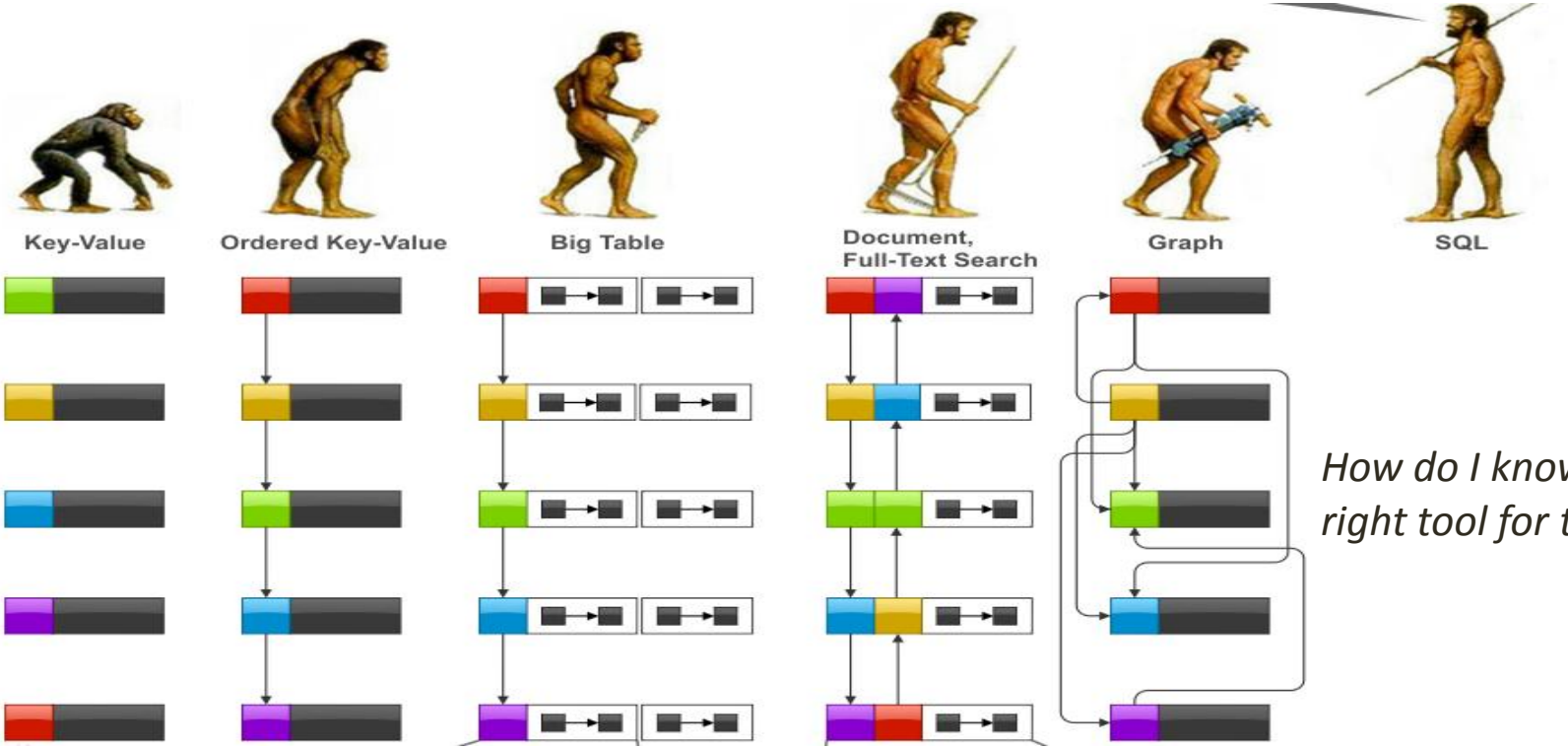
  How will different stakeholders in the enterprise data deal with data that could be stale and how do you enforce rules to sync data across systems

ROADMAP

PUTTING POLYGLOT PERSISTENCE IN PRACTICE

CONCLUSION AND OUTLOOK

Use the right tool for the right job…



How do I know which is the right tool for the right job?

(Katsov-2012)

PROBLEM

- How to specify data **requirements** for cloud environments?
- For a set of data requirements, how to choose an appropriate combination of cloud **storage system implementation** and **deployment provider**?
- How to **generate/manage** everything that's required to work with the selection that I make?

EXISTING SOLUTIONS

- Integration of cloud storage platforms (Livenson-2011)
  - Cloud Data Management Interface (CDMI) (SNIA-2011) proxy to integrate **blob** and **queue** data stores
- Data integration over NoSQL stores (Curé-2011)
  - Integration of relational and NoSQL databases (*Document*, *column*)
  - Focus on efficient answering of queries
- Storage provider selection (Ruiz-2011, Ruiz-2012)
  - **Characterize** storage providers features (Ex: *performance*, *cost*)
  - Specify requirements for application datasets (Ex: *expected size*, *access latency*, *concurrent clients*)
  - Based on the previous information, an **assignment** of datasets to different storage systems is proposed

EXISTING SOLUTIONS

- Modeling as a Service (Bruneliere-2010)
  - Deploy and execute model-driven services over the Internet (*SaaS*)
- Design and deploy applications in the cloud (Peidro-2011)
  - Promotes **graphical models** to capture cloud requirements
  - Models automatically deployed to PaaS and IaaS environments
- Application design/execution in multiple clouds (Ardagna-2012)
  - MDE quality-driven method for design, development and operation
  - **Monitoring** and feedback system

LIMITATIONS OF EXISTING SOLUTIONS

- Support for a limited set of cloud storage interfaces
- Data integration can be highly based on the relational model
- Limited information for the selection of data storage systems
- Consideration for high-level cloud models (*SaaS*) but limited support for low-level models (*PaaS* and *IaaS*)

OBJECTIVES

- Provide adequate **notations** and **environments** to characterize cloud data storage requirements
- Selection of cloud data storage implementations and deployment providers
- Management of the required artifacts to work with different combinations of cloud storage implementations and providers

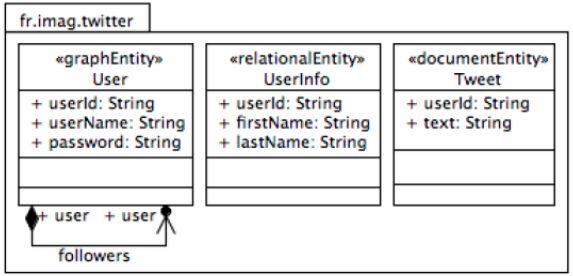
MODEL2ROO: APPROACH

```
@org.springframework.data.neo4j.annotation.NodeEntity
public class User {}

@org.springframework.roo.addon.jpa.
activerecord.RooJpaActiveRecord
public class UserInfo {}

@org.springframework.roo.addon.layers.
repository.mongo.RooMongoEntity
public class Tweet {}
```

High-level abstractions

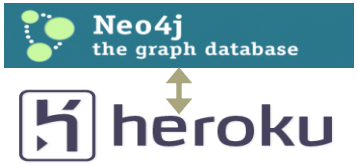


UML class diagram

Spring Roo

Spring Data

Low-level abstractions

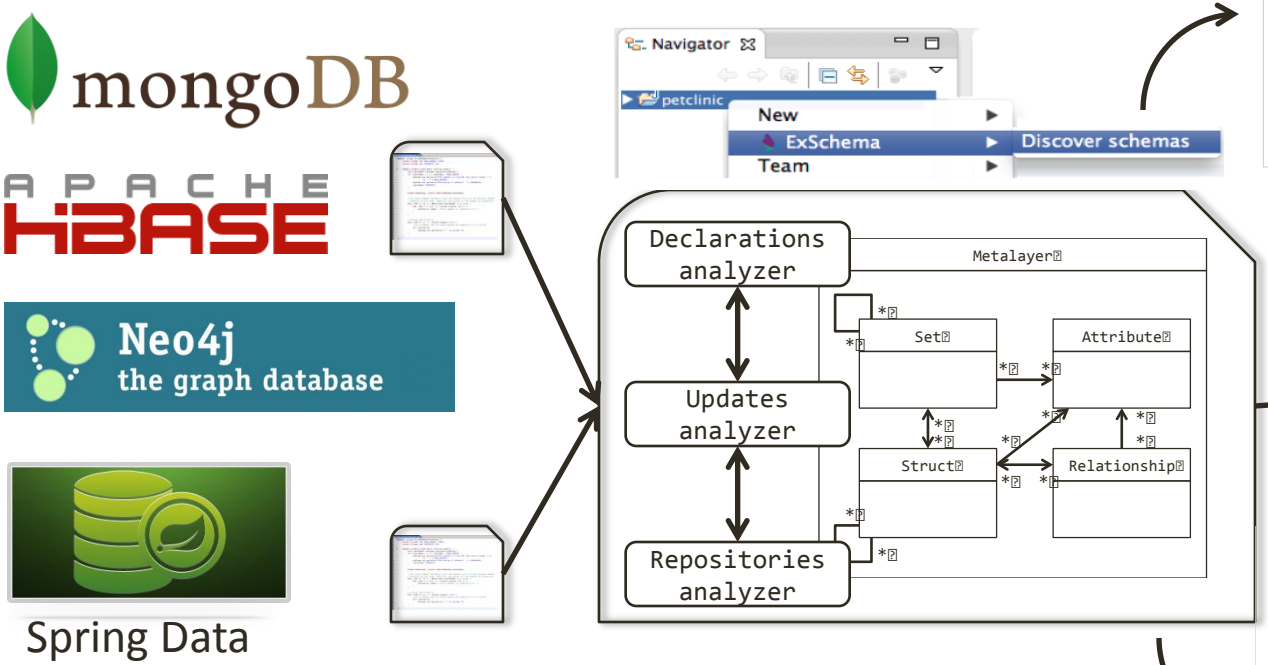


Graph database

Relational database

http://code.google.com/p/model2roo/

EXSCHEMA: APPROACH



http://code.google.com/p/exschema/

ROADMAP

DATA ALL AROUND IN THE ERA OF THE CLOUD

CONCLUSION AND OUTLOOK

Wrap up

- Polyglot persistence is about using different data storage technologies to handle varying data storage needs
  - can apply across an enterprise or within a single application
  - increases complexity in programming and operations, so the advantages of a good data storage fit need to be weighed against this complexity
  - will come at a cost - but it will come because the benefits are worth it if used appropriately
- Encapsulating data access into services reduces the impact of data storage choices on other parts of a system

Thanks Merci

Contact: Genoveva Vargas-Solar, CNRS, LIG-LAFMIA
Genoveva.Vargas@imag.fr
http://vargas-solar.imag.fr

Want to put cloud data managemet in practice ?
cf. http://vargas-solar.imag.fr/academika/cloud-data-management/

http://code.google.com/p/exschema/