The future is:

~~NoSQL Databases~~

Polyglot Persistence

a note on the future of data storage in the enterprise, written primarily for those involved in the management of application development.

Martin Fowler

Pramod Sadalage

ThoughtWorks®

 Rendered: February 8, 2012 11:26

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.

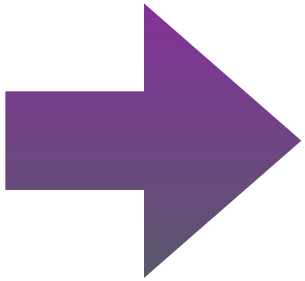
- **Reporting**

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

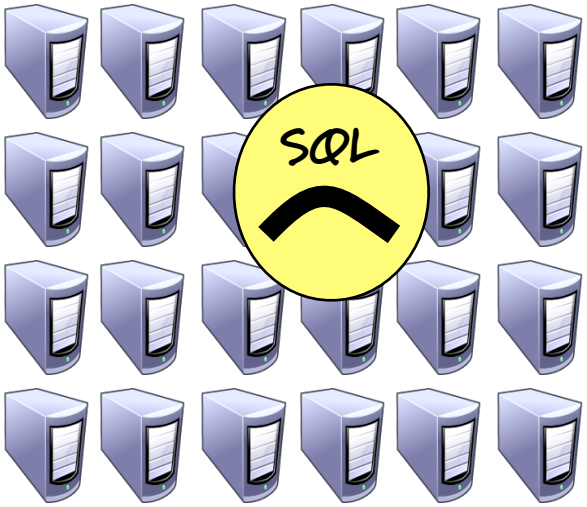
All this supported by Big Database Vendors and the separation of the DBA profession.

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.

Guardian

New functionality uses Mongo rather than relational DB. They found Mongo's document data model significantly easier to interact with for their kind of application. [more...]

DNC

Searching 300 Million voters information for 1 person with addresses, emails, phones is tough with a relational data store. MongoDB was used to store the documents about the person. [more...]

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

Danish Health Care

Centralized record of drug prescriptions. Currently held in MySQL databases, but concerned about scale for both response time and availability. Migrated data to Riak. [more...]

McLaren

Streaming of telemetric data into MongoDB for later analysis. Orders of magnitude faster than relational (SQL Server). [more...]

but this does not mean relational is dead

- ### the relational model is still relevant

  The tabular model is suitable for many kinds of data, particularly when you need to pick apart data and re-assemble it in different ways for different purposes.

- ### ACID transactions

  In order to run effectively on a cluster, most NoSQL databases have limited transactional capability. Often this is enough... but not always.

- ### Tools

  The long dominance of SQL means that many tools have been written to work with SQL databases. Tooling for alternative datastores is much more limited.

- ### Familiarity

  NoSQL systems are still new, so people aren't familiar with using them. So we shouldn't be using them on utility projects where their benefits would have less impact.

this leads us to a world of

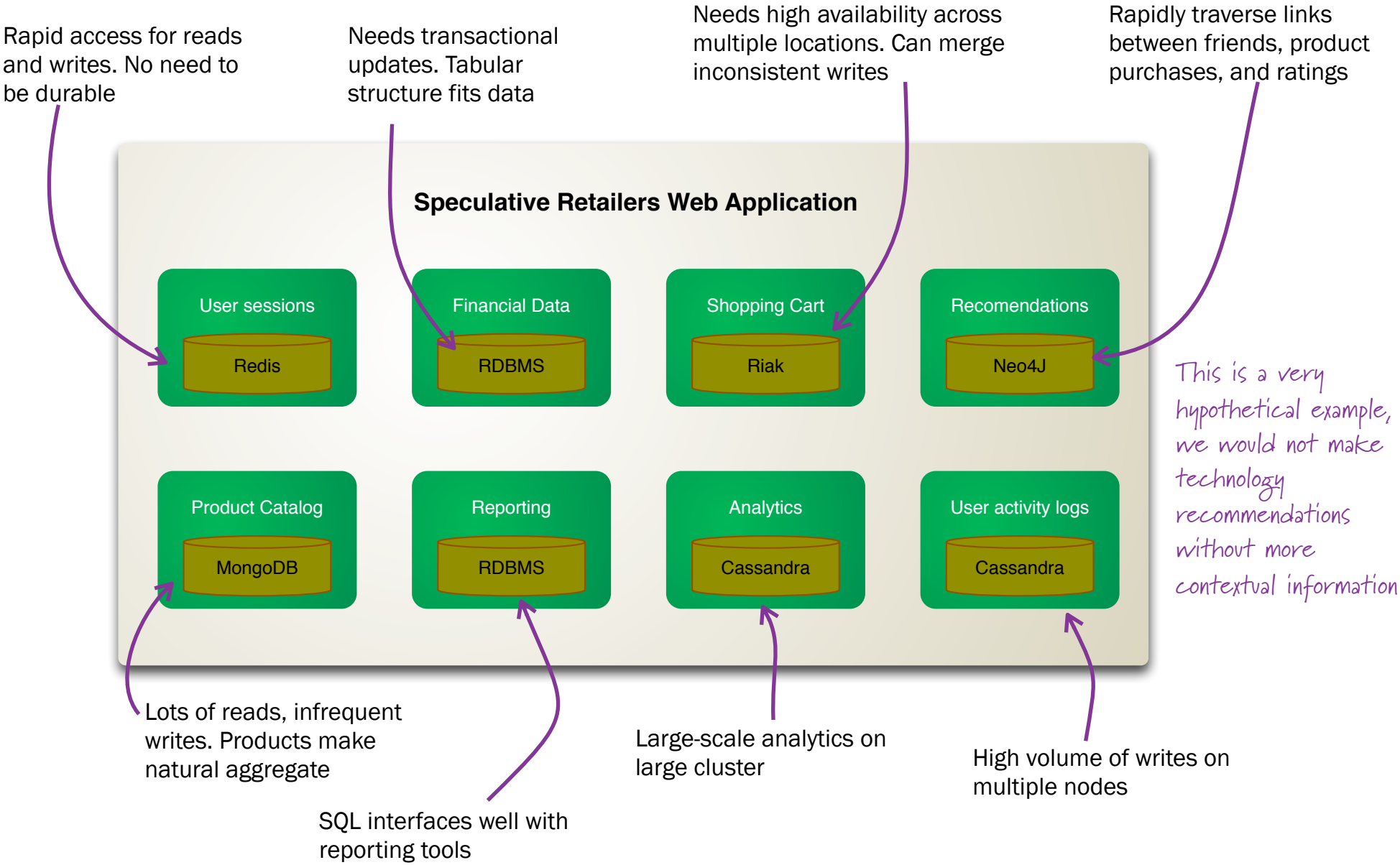
Polyglot Persistence

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?



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.

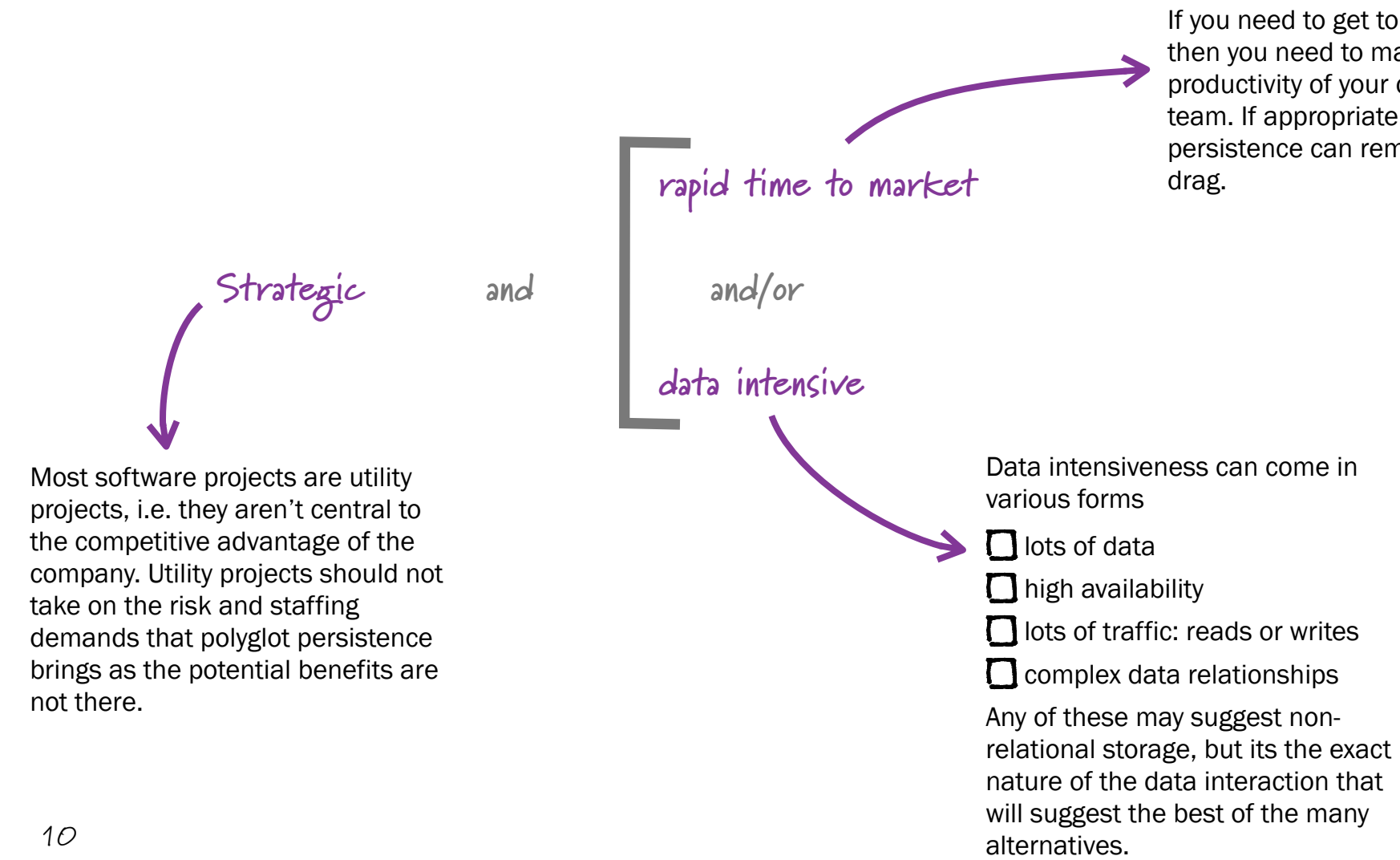
- ☐ **Organizational Change**

  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

what kinds of projects are candidates for polyglot persistence?



for more information...

On the Web

We are both active writers on our websites. Martin writes at http://martinfowler.com and Pramod at http://www.sadalage.com/.

Forthcoming Book

We are currently working on a introductory book to NoSQL databases, to be titled: NoSQL Distilled (see http://martinfowler.com/bliki/NosqlDistilled.html)

Consulting and Delivery

ThoughtWorks has carried out several projects delivering production systems using NoSQL technologies. To see if NoSQL is a good fit for your needs, and how we can help your delivery, contact your local ThoughtWorks office, which you can find at http://thoughtworks.com

ThoughtWorks®