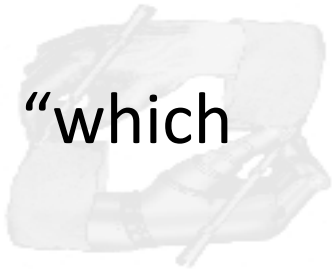
Course Introduction

Designing Data Bases with Advanced Data Models

Dr. Fabio Fumarola

Enterprise computing evolution

- We've spent several years in the world of enterprise computing
- We've seen many things change in:
  - Languages, Architectures,
  - Platforms, and Processes.
- But in one thing we stayed constant – "relational database stored the data"
- The data storage question for architects was: "which relational database to use"



The stability of the reign

DTK

- Why?
- An organization's data lasts much longer than its programs (COBOL?)
- It's valuable to have stable data storage which is accessible from many applications
- In the last decades RDBMS have been successful in solving problems related to storing, serving and processing data.

Before going deep on the course arguments let's understand the evolution of decision support systems.

FROM BUSINESS TO DECISION SUPPORT

From business to decision support: ‘60

- Starting from ’60 data were stored using magnetic disks.
- Supported analysis where static, only aggregated and pretty limited
- For instance, it was possible to extract the total amount of last month sales

From business to decision support: ‘80

- With relational databases and SQL, data analysis start to be somehow dynamics.
- SQL allows us to extract data at detailed and aggregated level
- Transactional activities are stored in Online Transaction Process databases
- OLTP are used in several applications such as orders, salary, invoices…

From business to decision support: ‘80

- The best hypothesis was that the described modules are included into Enterprise Resource Planning (ERP) software
- Examples of such vendors are SAP, Microsoft, HP and Oracle.
- Normally, what happen is that each module is implemented as an ad-hoc software with is own database.
- Cons: Data representation and integration.

OLTP design

- Such kind of databases are designed to be:
  - Strongly normalized,
  - Fast in inserting data.
- However, data normalization:
  - Do not foster the read of huge quantity of data,
  - Increase the number of tables used to store records.

Foster decision support

DTK

- In order to support “Decisions” we need to extract de-normalized data via several JOINS.
- Moreover, operational databases offer a no/limited visibility on historical data.

Considerations:

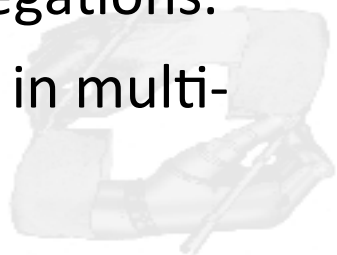
- These factors make hard data analysis made on OLTP databases…

From business to decision support: ‘90

- Thus in ’90 bore databases designed to support analysis from OLTP databases.
- This is the arise of Data Warehouses
- DWs are central repositories of integrated data from one or more disparate sources.
- They store current and historical data and are used for creating trending reports for management reporting such as annual and quarterly comparisons.

Types of DWs systems

- Data Mart
  - is a simple form of a data warehouse that is focused on a single subject (or functional area), such as sales, finance or marketing.
- Online analytical processing (OLAP):
  - is characterized by a relatively low volume of transactions.
  - Queries are often very complex and involve aggregations.
  - OLAP databases store aggregated, historical data in multi-dimensional schemas.



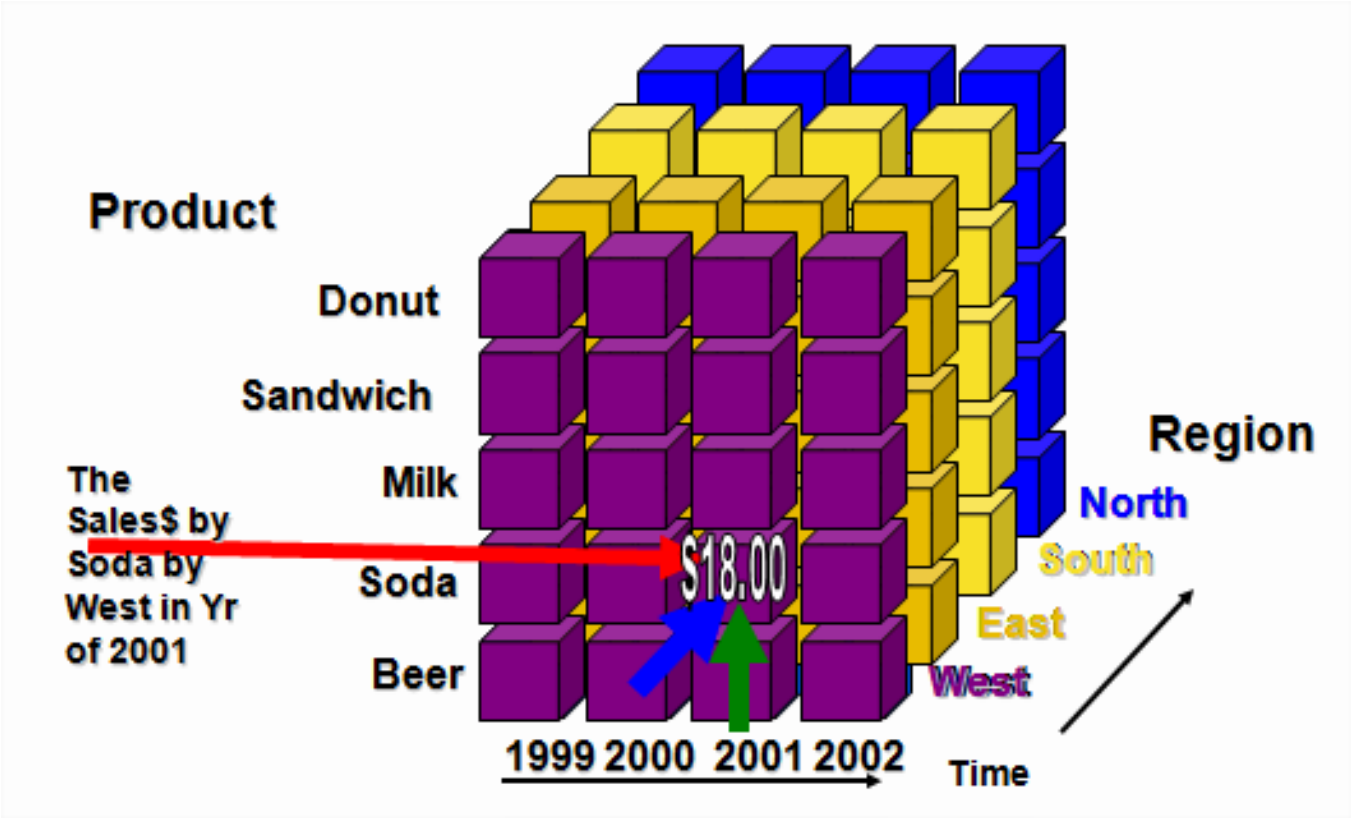
Types of DWs systems

- Predictive analysis
  - Predictive analysis is about finding and quantifying hidden patterns in the data using complex mathematical models that can be used to predict future outcomes.
  - Predictive analysis is different from OLAP in that OLAP focuses on historical data analysis and is reactive in nature, while predictive analysis focuses on the future.

Data Warehouse

- Data stored in DWs are the starting point for the Business Intelligence (BI).
- Def. "*Business intelligence (BI) is the set of techniques and tools for the transformation of raw data into meaningful and useful information for business analysis purposes*".
- With the evolution of BI systems we moved from SQL based analysis to Visual Instruments.

Example DW Cube

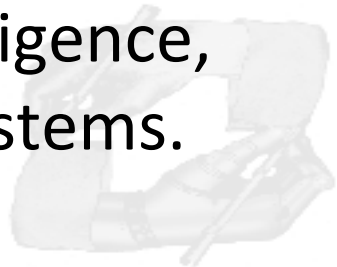


OLAP features

- OLAP systems support data exploration through *drill-down, drill-up, slicing e dicing* operations.
- However, we still have an historical point of view of what happened in the business but now on what is happening.
- We cannot make prediction on the future

From business to decision support: ‘00

- Starting from 2000 it arises the necessity to do predictive analysis.
- The techniques in this scenario are in the field of Data Mining
- Data Mining is the computational process of discovering patterns in large dataset involving methods at the intersection of artificial intelligence, machine learning, statistics, and database systems.

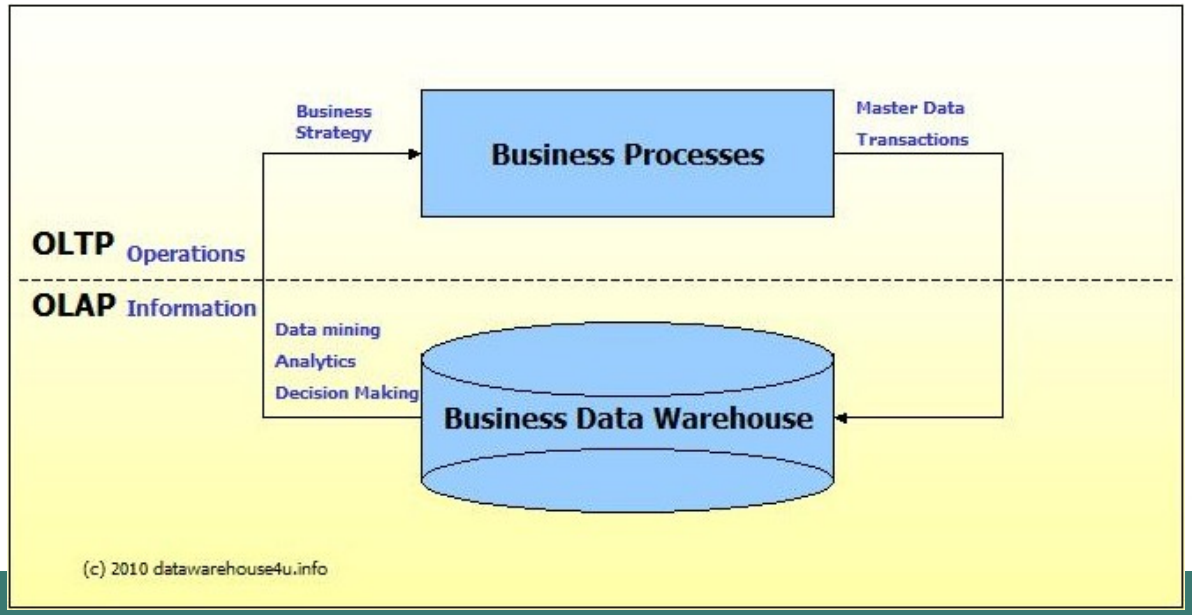


From business to decision support: ‘00

- The overall goal of the data mining process is to extract novel information from a data set and transform it into understandable knowledge.
- Aside from the raw analysis step, it involves database and data management aspects, data pre-processing, model and inference considerations, interestingness metrics, complexity considerations, post-processing of discovered structures, visualization, and online updating.

From business to decision support: Recap

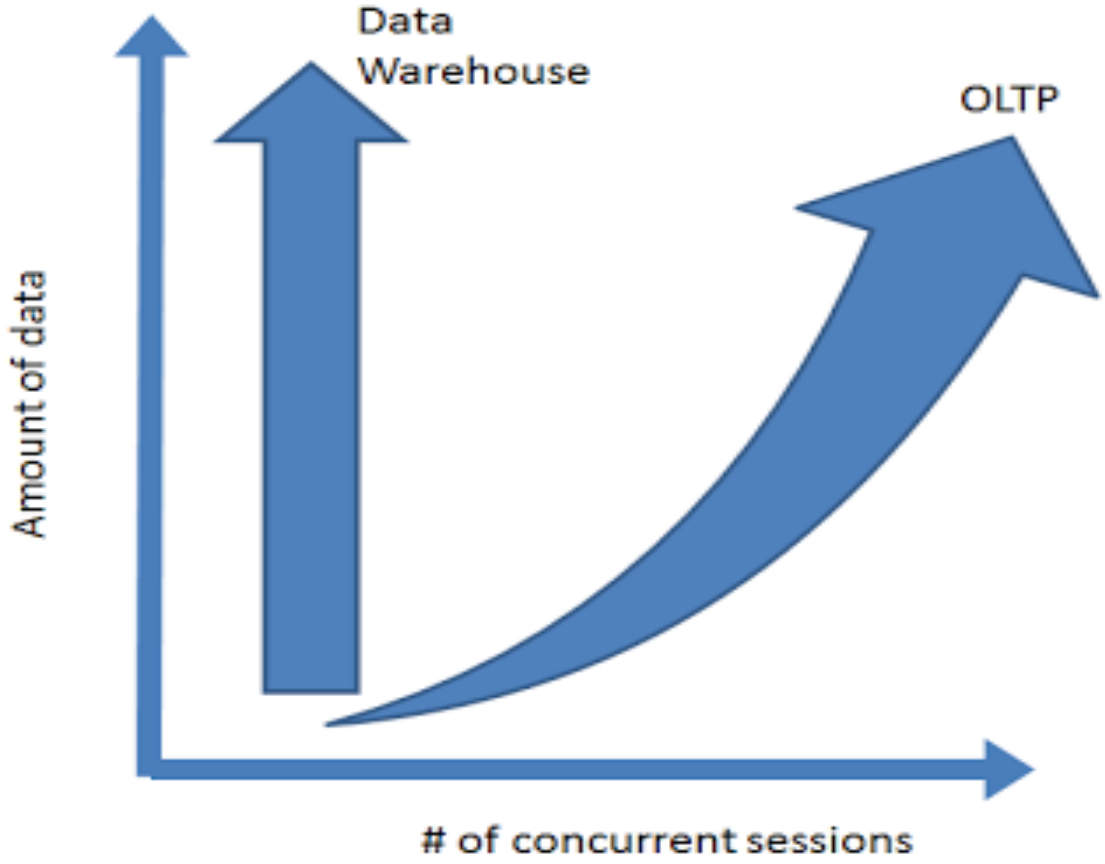
- In the last decades RDBMS have been successful in solving problems related to storing, serving and processing data.

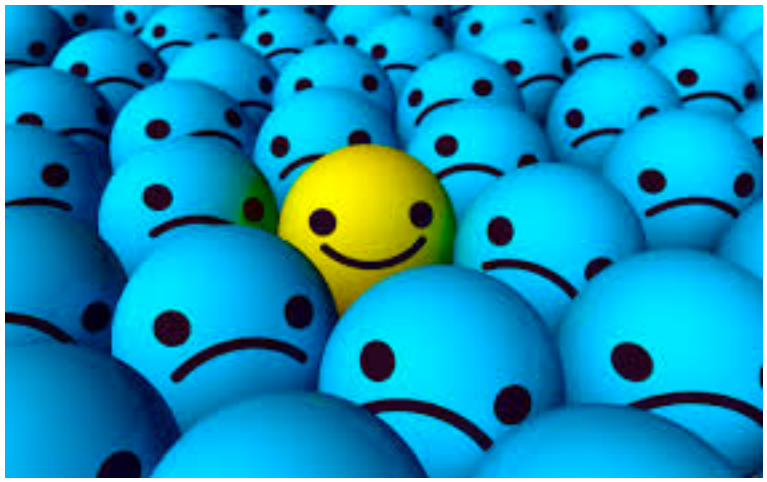


From business to decision support: Recap

- Vendors such as Oracle, Vertica, Teradata, Microsoft and IBM proposed their solution based on Relational Algebra and SQL.
- With Data Mining we are able to extract knowledge from data which can be used to support predictive analysis (Data Mining is not only prediction!!!).

Challenges of Scale Differ





But

THERE IS SOMETHING THAT DOES NOT WORK!

1. Scaling Up Databases

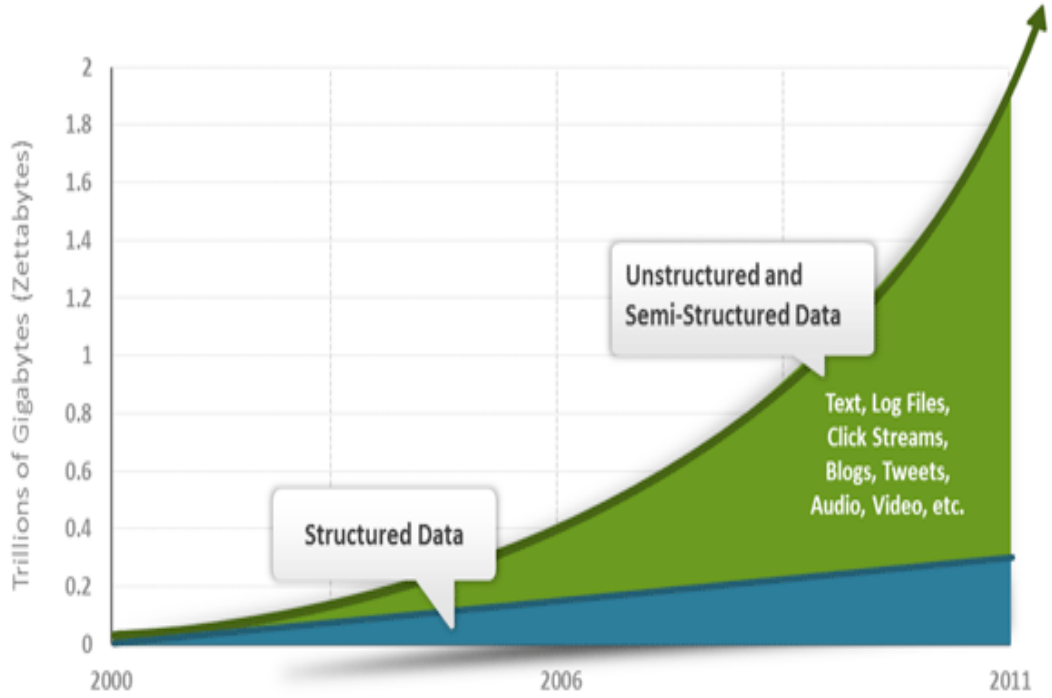
A question I'm often asked about Heroku is: "How do you scale the SQL database?" There's a lot of things I can say about using caching, sharding, and other techniques to take load off the database. But the actual answer is: we don't. SQL databases are fundamentally non-scalable, and there is no magical pixie dust that we, or anyone, can sprinkle on them to suddenly make them scale.

Adam Wiggins Heroku

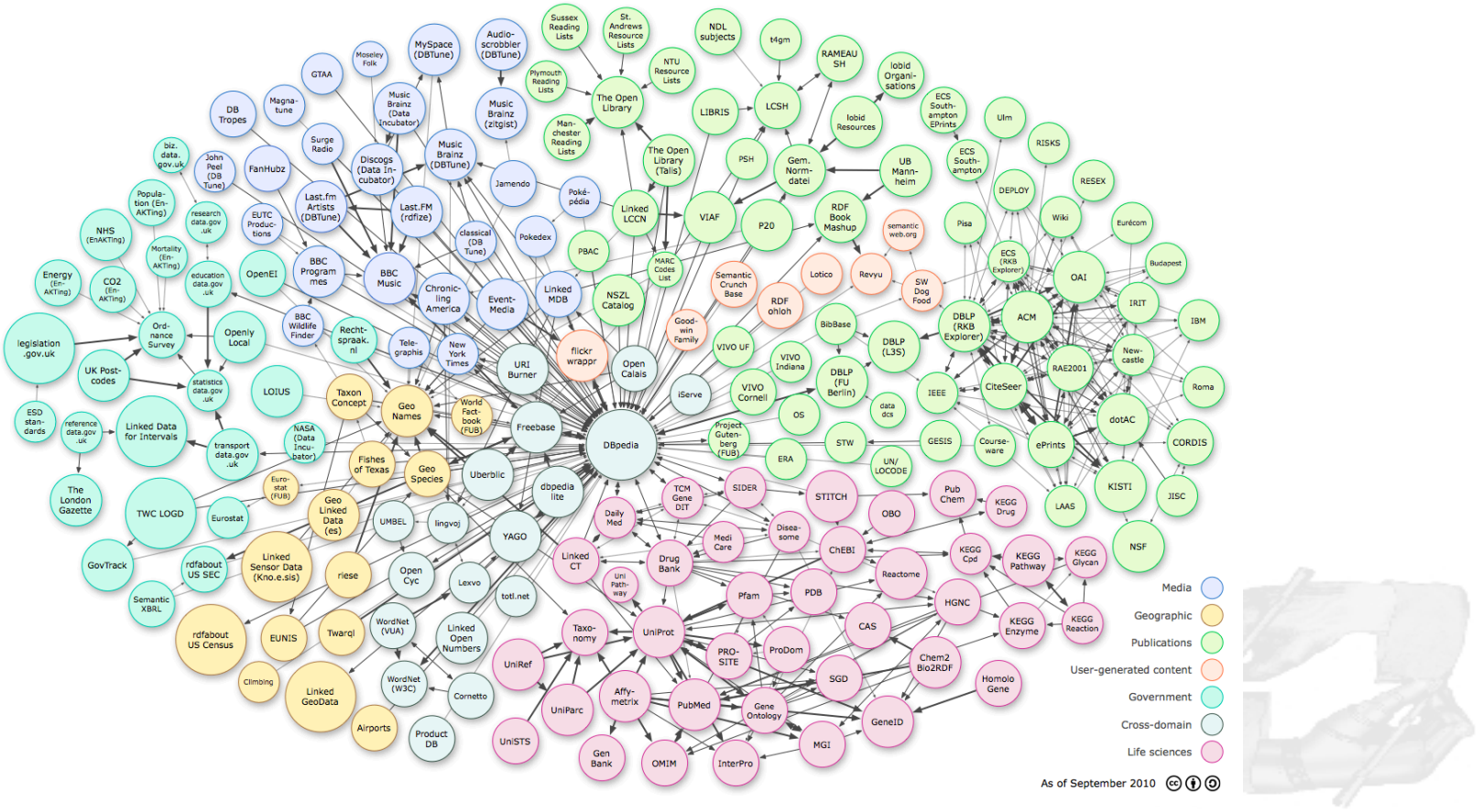
Adam Wiggins, Heroku Patterson, David; Fox, Armando (2012-07-11). **Engineering Long-Lasting Software: An Agile Approach Using SaaS and Cloud Computing**, Alpha Edition (Kindle Locations 1285-1288). Strawberry Canyon LLC. Kindle Edition.

2. Data Variety

- RDBMs have problems with Unstructured and Semi-Struct



3. Connectivity



4. P2P Knowledge

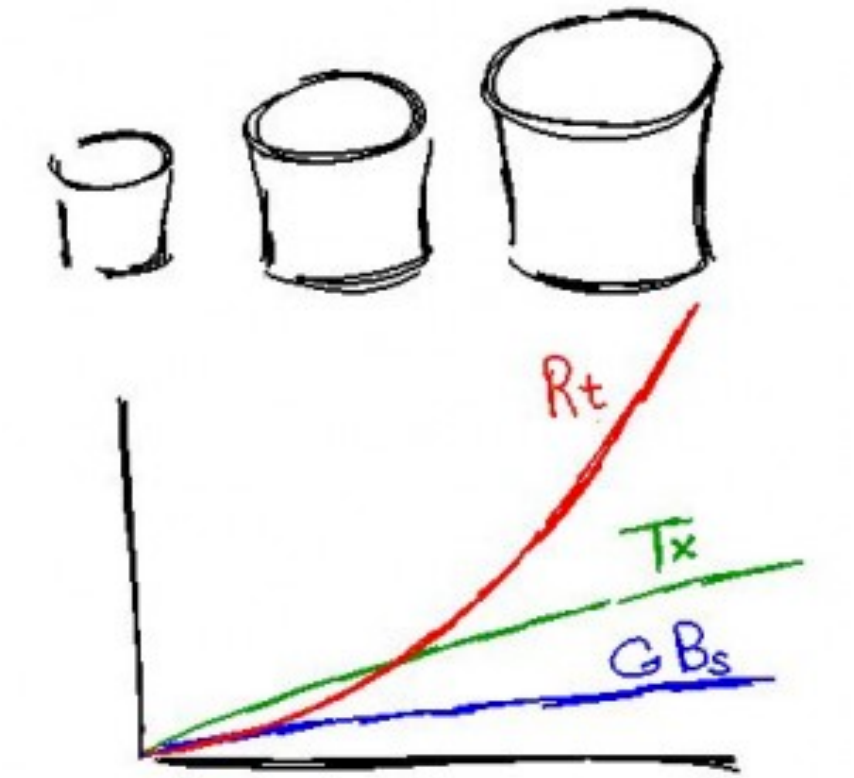
5. Concurrency

6. Concurrency

6. Diversity

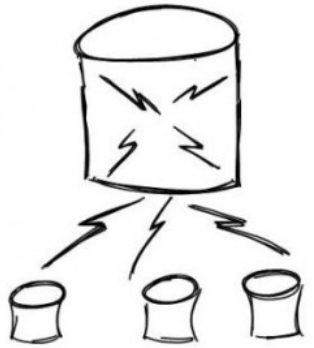
7. Cloud

What is the problem with RDBMs



http://codefutures.com/database-sharding/

- Caching
- Master/Slave
- Master/Master
- Cluster
- Table Partitioning
- Federated Tables
- Sharding
- Distributed DBs



What is the problem with RDBMs

- RDBMS can somehow deal with this aspects, but they have issues related to:
  - expensive licensing,
  - requiring complex application logic,
  - Dealing with evolving data models
- There were a need for systems that could:
  - work with different kind of data format,
  - Do not require strict schema,
  - and are easily scalable.

Help!!!

NOSQL: THE NEW CHALLENGER!

NoSQL

DTK

- It is born out of a need to handle large data volumes
- It forces a fundamental shift to building large hardware platforms through clusters of commodity servers.
- This need raises from the difficulties of making application code play well with relational databases

NoSQL

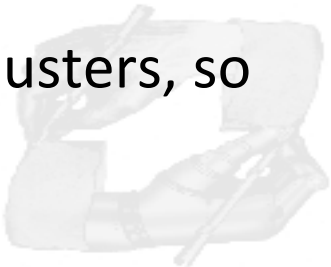
- The term “NoSQL” is very ill-defined.
- It’s generally applied to a number of recent non relational databases: Cassandra, Mongo, Neo4j, Hbase and Redis,…
- They embrace
  - schemaless data,
  - run on a cluster,
  - and have the ability to trade off traditional consistency for other useful properties

Why are NoSQL Databases Interesting

1. Application development productivity:
   - A lot of application development is spent on mapping data between in memory data structures and relational databases
   - A NoSQL database may provide a data model that can simplify that interaction resulting in less code to write, debug, and evolve.

Why are NoSQL Databases Interesting

2. Large-scale data:
   - Organizations are finding it valuable to capture mode data and process it more quickly.
   - They are finding it expensive to do so with relational databases.
   - NoSQL database are more economic if ran on large cluster of many smaller an cheaper machines.
   - Many NoSQL database are designed to run on clusters, so they better fit on Big Data scenarios.

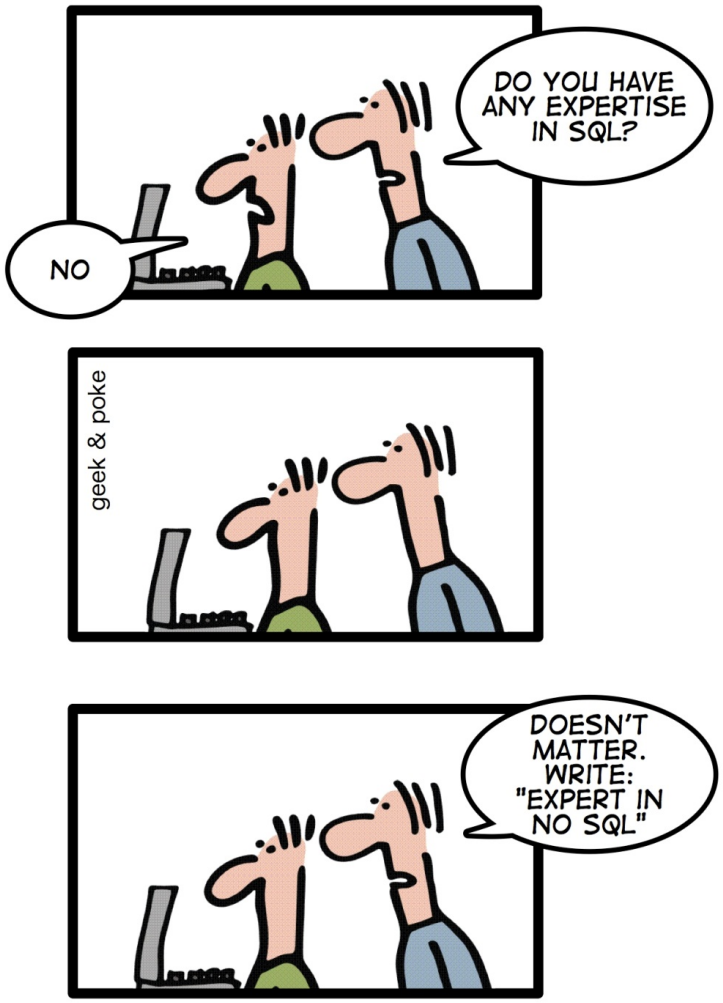


Connectedness

Internet Hypertext, RSS, Wikis, blogs, wikis, tagging, user generated content, RDF, ontologies

WHY NOSQL

HOW TO WRITE A CV



Leverage the NoSQL boom

The Value of Relational Databases

- Relational databases have become such an embedded part of our computing culture.
- What are the benefits they provide?

Getting at Persistent Data

DTK

- The most obvious value of a database is keeping large amount of persistent data
- Two areas of memory:
  - Main memory: fast, volatile, limited in space and lose data when it loses the power
  - Backing store: larger but slower, commonly seen as a disk

Getting at Persistent Data

DTK

- The most obvious value of a database is keeping large amount of persistent data
- Two areas of memory:
  - Main memory: fast, volatile, limited in space and lose data when it loses the power
  - Backing store: larger but slower, commonly seen as a disk

Getting at Persistent Data

- The backing store can be organized in all sort of ways.
- For many productivity applications (such as word processors) it is a file in the file system.
- For most enterprise applications, however, the backing store is a database.
- A database allows more flexibility than a file system.

Concurrency

- Concurrency is notoriously difficult to get right.
- Object oriented is not the right programming model to deal with concurrency.
- Since enterprise applications can have a lot of concurrent users, there is a lot of rooms for bad things to happen.
- Relation databases have transactions that help mitigating this problem, but….

Concurrency

- You still have to deal with transactional error when you try to book a room that is just gone.
- The reality is that the transactional mechanism has worked well to contain the complexity of concurrency.
- Transaction with rollback allows as to deal with errors.

Integration

- Enterprise applications live in a rich ecosystem
- multiple application written by different teams need to collaborate in order to get things done
- This collaboration is done via data sharing.
- A common way to do this is **shared database integration** [Hohpe and Woolf] where multiple applications store their data into a single database
- Using a single database, allows all the application to share data easily, while the database concurrency control applications such as users.

A (Mostly) Standard Model

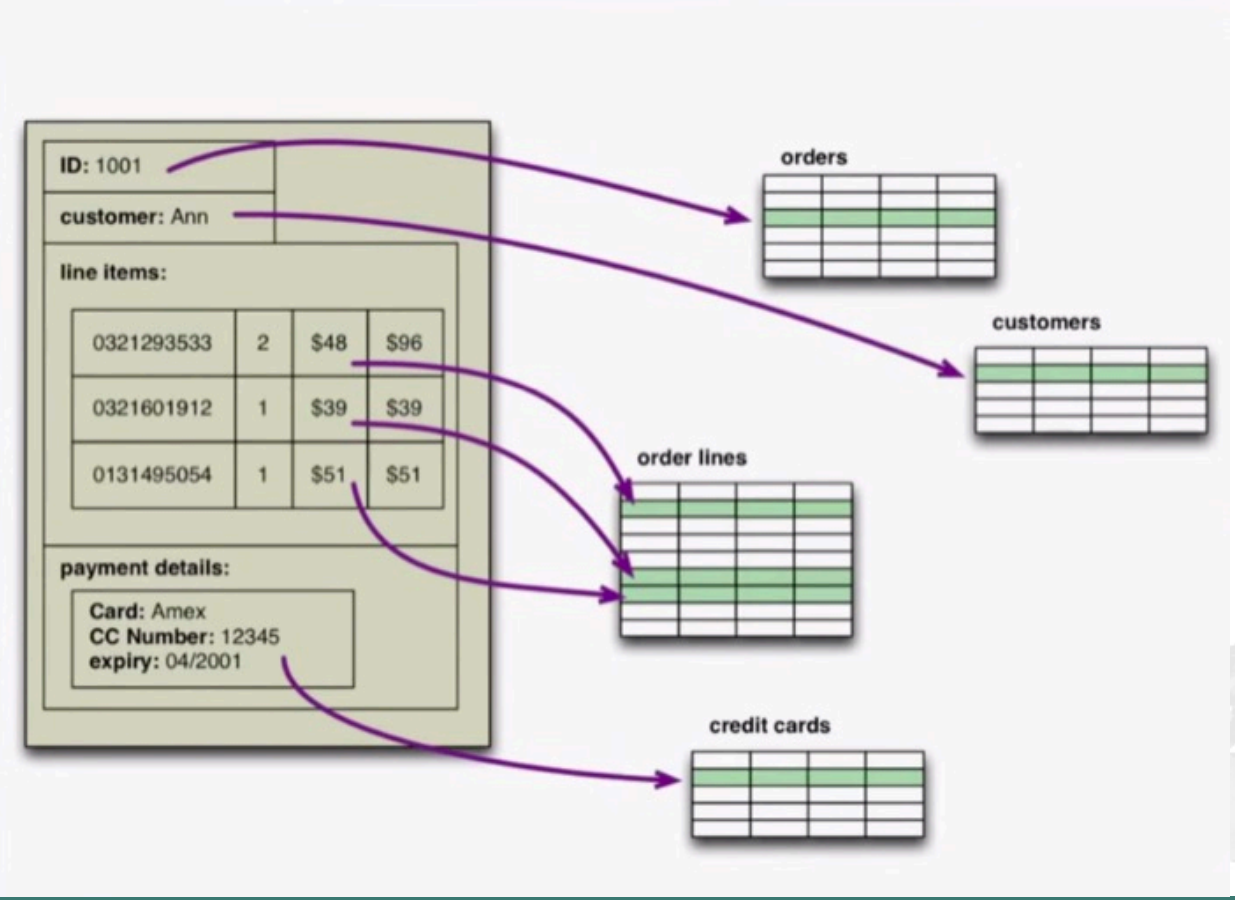
DTK

- Relational database have succeeded because they have a standard model
- As a result, developers and database professionals can apply the same knowledge in several projects.
- Although there are differences between different RDBMs, the core mechanism remain the same.

Impedance Mismatch

- It is the difference between the relational model and the in-memory data structures.
- Relational data organizes data into table and rows (relation and tuples).
  - A tuple is a set of name-value pairs
  - A relation is a set of tuples.
- All the SQL operations consume and return relations.

Impedance Mismatch: Example



Impedance Mismatch

- Tuples and relation provides elegance and simplicity, but it also introduces limitations.
- In particular, the values in a relational tuple have to be simple.
- They cannot contain any structure, such as nested record or a list.
- This limitation is not true for in memory data-structures.

Impedance Mismatch

DTK

- As a result, if we want to use richer in-memory data structure, we have to translate it to a relational representation to store in on disk.
- While object-oriented language succeeded, object-oriented databases faded into obscurity.
- Impedance mismatch has been made much easier to deal with Object-Relational Mapping (ORM) frameworks such as Eclipse-Link, Hibernate and other JPA implementations.

Impedance Mismatch

DTK

- ORMs remove a lot of work, but can become a problem when people try to ignore:
  - the database, and
  - query performance suffer
- This is where NoSQL database works greatly, why?

Application and Integration DBs

- This is a event that happen several times in SW projects.
- In this scenario, the database acts as an integration database.
- The downsides to share database are.

Application and Integration DBs

- The downsides to share database are:
  - Its structure tend to be more complex than any single application needs,
  - If an application want to make changes to its data storage, it needs to coordinate with all the other applications,
  - Performance degradation due to huge number of access
  - Errors in database usage since it is accessed by application written by different teams.
- This is different from single application databases

Application and Integration DBs

- Interoperability concerns can now shit to interfaces of the application allowing interaction over HTTTP.
- This is what happen with micro-services (http://www.tikalk.com/java/micro-services/)
- Micro-services and Web services in general enable a form of communications based on data.
- Data is represented as documents using before XML and now JSON format.

Application and Integration DBs

- If you are going to use service integration using text over HTTP is the way to go.
- However, if we are dealing with performance there are binary protocols.

MessagePack

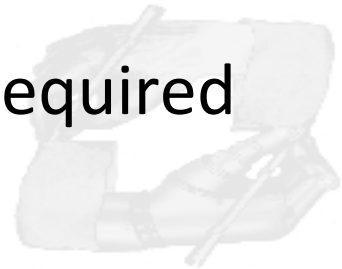
Apache Thrift ™

Scala Pickling

Attack of the Clusters

DTK

- In 2000s several large web properties dramatically increase in scale:
  - Websites started tracking activity and structure in detail (analytics)
  - Large sets of data appeared: links, social networks, activity in logs, mapping data.
  - With this growth in data came a growth in users
- Coping with this increase in data and traffic required more computing resources.



Attack of the Clusters

- There were to choices:
  - Scaling up
  - Scaling out
- Scaling up implies bigger machines, more processors, disk storage and memory ($$$).
- Scaling out was the alternative:
  - Use a lot of small machine in a cluster.
  - It is cheap and also more resilient (individual failures)

Attack of the Clusters

- This revealed a new problem, relational databases are note designed to run on clusters.
- Clustered relational databases (e.g. Oracle RAC or Microsoft SQL Server) work on the concept of shared disk subsystem.
- RDBMs can also run on separate server with different sets of data (sharding)

Attack of the Clusters

DTK

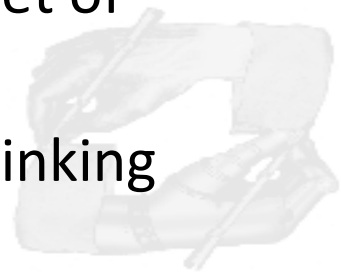
- However, it needs an application to control the sharded-database.
- Also we lose querying, referential integrity, transactions or consistency control cross shard.
- These technical issues are exacerbated by licensing cots.
- This mismatch between DBs and clusters led some organizations to consider different solutions

The emergence of NoSQL

- Two companies in particular – Google and Amazon – have been very influential.
- They were capturing a large amount of data and their business is on data management
- Both companies produces influential papers:
  - BigTable from Google
  - Dynamo DB from Amazon

The emergence of NoSQL

- As part of innovation in data management system, several new technologies where built:
  - 2003 - Google File System,
  - 2004 - MapReduce,
  - 2006 - BigTable,
  - 2007 - Amazon DynamoDB
  - 2012 Google Cloud Engine
- Each solved different use cases and had a different set of assumptions.
- All these mark the beginning of a different way of thinking about data management.



The emergence of NoSQL

- It is irony that the term “NoSQL” appeared in late 90s from a relational database made by Carlo Strozzi.
- The name comes from the fact that it does not used SQL as query language.
- However, the usage of “NoSQL” as we consider today come from a meetup on 2009 in San Francisco.
- They want a term that can be used as Twitter hashtag. #NoSQL

NoSQL Characteristics

1. They don’t use SQL. (HBase, Cassandra, Redis…)
2. They are generally open-source projects.
3. Most of them are designed to run on clusters.
4. RDBMs used ACID transactions to handle consistency across the whole database. NoSQL resort to other options (CAP theorem).
5. No all are cluster oriented (Graph DBs)
6. NoSQL operate without a schema. (schema free)

The emergence of NoSQL

DTK

- NoSQL does not stands for Not-Only SQL.
- It is better to NoSQL as a movement rather than a techonology.
- RDBMs are not going away.
- The change is that relational databases are an option
- This point of view is often referred to as **polyglot persistence**

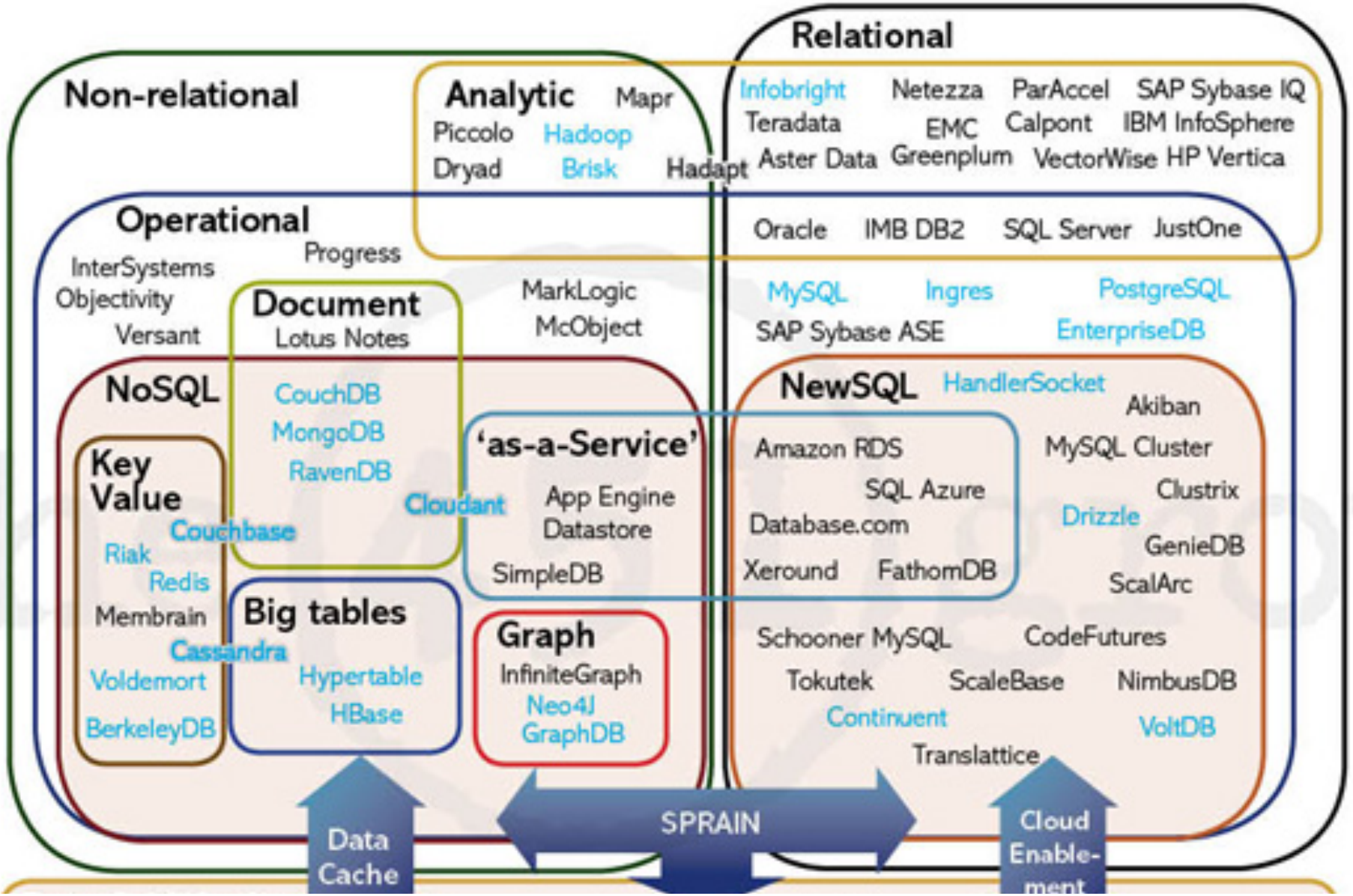
The emergence of NoSQL

- Instead of just picking a relational database, we need to understand:
  1. The nature of the data we are storing, and
  2. How we want to manipulate it.
- In order to deal with this change most organizations need to shift from integration database to application database.
- In this course we concentrate on Big Data running on clusters.

The emergence of NoSQL

DTK

- The Big Data concerns have created an opportunity for people to think freshly about their storage needs.
- NoSQL help developer productivity by simplifying their database access even if they have no need to scale beyond single machine.


Relational
Non-relational
Analytic
Mapr
Infobright
Netezza
ParAccel
SAP Sybase IQ
Teradata
EMC
Calpont
IBM InfoSphere
Piccolo
Hadoop
Aster Data
Greenplum
VectorWise
HP Vertica
Dryad
Brisk
Hadapt
Operational
Oracle
IMB DB2
SQL Server
JustOne
Progress
InterSystems
Objectivity
Versant
Document
Lotus Notes
MarkLogic
McObject
MySQL
Ingres
PostgreSQL
SAP Sybase ASE
EnterpriseDB
NoSQL
CouchDB
MongoDB
RavenDB
Cloudant
'as-a-Service'
App Engine
Datastore
SimpleDB
NewSQL
HandlerSocket
Akiban
Amazon RDS
SQL Azure
Database.com
Xeround
FathomDB
MySQL Cluster
Clustrix
Drizzle
GenieDB
ScalArc
Key Value
Couchbase
Riak
Redis
Membrain
Cassandra
Voldemort
BerkeleyDB
Big tables
Hypertable
HBase
Graph
InfiniteGraph
Neo4J
GraphDB
Schooner MySQL
CodeFutures
Tokutek
ScaleBase
NimbusDB
Continuent
VoltDB
Translattice
Data Cache
SPRAIN
Cloud Enable-ment