I404B NoSQL

Session 1

NoSQL vs SQL: History and Evolution

Sébastien Combéfis

Fall 2019

Objectives

- From the relational model to NoSQL models
  - Comparison of the two paradigms
  - History of the emergence of NoSQL
  - ACID and BASE properties
- Interest and use of NoSQL models
  - Big Data and the 4Vs
  - Distributed architecture
  - Overview of NoSQL data models

NoSQL

What does NoSQL stand for?

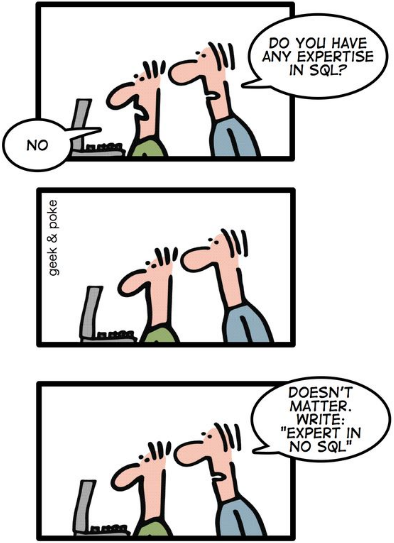
non relational

non SQL

not only SQL

Not your grandfather's database?

HOW TO WRITE A CV



Leverage the NoSQL boom

Relational vs NoSQL

Enterprise Computing

- Many changes in the technology used occurred in enterprises

  Programming languages, architectures, platforms, etc.

- Stability in the way data are stored

  Relational databases have always been used since then

- Some successful challengers do exist in small niche markets

  Architects still choose relational databases

Software and Data

- A company uses software and stores data

  Those two elements are as independent as possible

- Data often lives longer than softwares

  New softwares must support existing data

- Data must be as stable as possible

  Easily understandable and accessible through an API

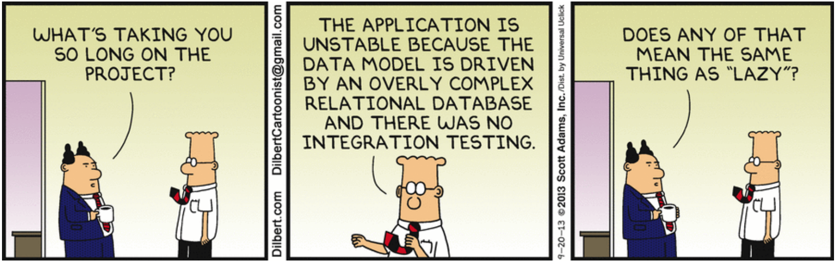
Data

- Need to organise data is at the heart of computer science

  While optimising storage and retrieval

- Several other important sub-functions

  Security, protection against inconsistencies, etc.


WHAT'S TAKING YOU SO LONG ON THE PROJECT?
THE APPLICATION IS UNSTABLE BECAUSE THE DATA MODEL IS DRIVEN BY AN OVERLY COMPLEX RELATIONAL DATABASE AND THERE WAS NO INTEGRATION TESTING.
DOES ANY OF THAT MEAN THE SAME THING AS "LAZY"?
Dilbert.com DilbertCartoonist@gmail.com
9-20-13 ©2013 Scott Adams, Inc./Dist. by Universal Uclick

I hate databases...

- Most developers do not like databases
  - Interact with a DBMS (DataBase Management System)
  - Learn the SQL language (Structured Query Language)
  - Links between data in the database and those in the program
- Database seen as an intrusion of an external element

  With very poor integration with the application code

Emergence of NoSQL

- Need to store large amounts of data

  Moving from large platforms to server clusters

- New database engines emerged under the NoSQL name

  Cassandra, Mongo, Neo4j, Riak, etc.

- Lessening the traditional data consistency constraint

  For performance, scalability, easy programming, etc.

Relational Model Strength

- Relational model based on a standard model
  - Guarantees about data consistency (constraints)
  - Efficient and persistent data storage (better than files)
  - Concurrent read/write access to data (transactions)
- Integration and collaboration of enterprise applications

  Realised with an integration through shared database

End of Relational Model?

- Relational databases are powerful and stable...

  Not ready to disappear in the short and medium term

- ...but they are no longer sufficient

  Unnecessary heaviness to store certain types of data

- Hybrid systems combining several technologies

  Concurrently, cooperatively, in a distributed way, redundantly...

History

Timeline

1950	Hierarchical model development (IMS)
1970	Appearance of relational model (*Edgar F. T. Codd*)
1980's	Domination of the relational model
2000's	Emergence of the NoSQL term
2011	Emergence of NewSQL

Hierarchical Model (1)

- Building relationships from parents to children

  Limited to one-way relationships

- Database consists of records with fields

  Grouped into record types

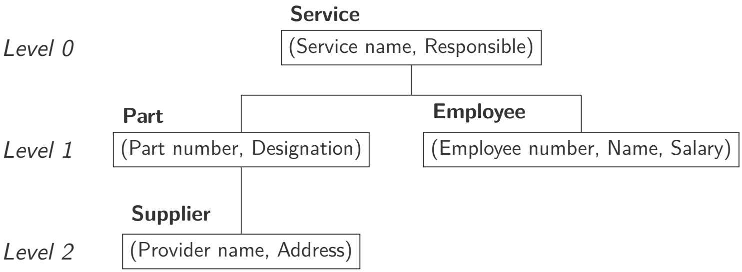
- IMS engine created by IBM (*Information Management System*)

  Used by NASA to manage building materials (started in 1968)

Hierarchical Model (2)

- Hierarchy diagram representing a service

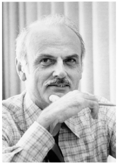
  Fields of the different record types included



Relational Model

- *"A Relational Model of Data for Large Shared Data Banks"*

  Edgar Frank "Ted" Codd, Ph.D. (1923–2003)
  IBM Research, San Jose, California, USA



- ACM A.M. Turing Award 1981

  Theory and practice of DBMS, esp. relational databases

- Organisation of data according to a mathematical model

  Based on set theory and relational algebra

- Isolation of access to data and physical implementation

  Thanks to a high-level declarative language

System R

- First implementation of SQL with the System R prototype

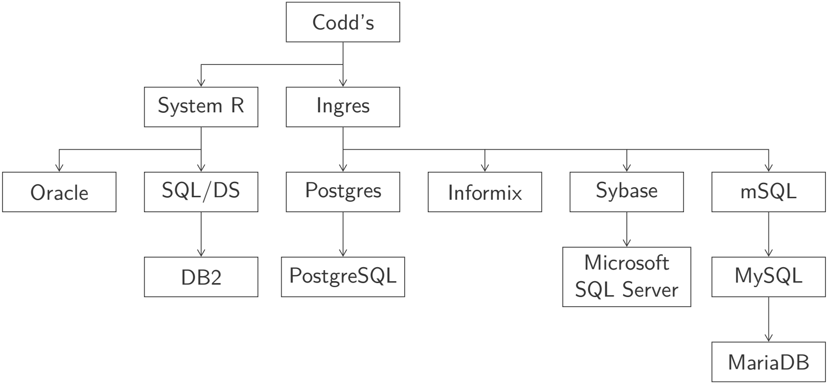
  Developed in 1974 to experiment Codd's concepts

- SEQUEL language (*Structured English Query Language*)
  - Access method: RSS (*Research Storage System*)
  - Optimising SQL processor: RDS (*Relational Data System*)

- Pratt & Whitney first customer of System R in 1977

Pratt & Whitney
A United Technologies Company

Relational Model Evolution



A more complete RDBMS Genealogy has been proposed by HPI (see references).

From OLTP to OLAP

- Online Transactional Processing (OLTP)

  Purely transactional use of data (management)

- Online Analytical Processing (OLAP)

  Dashboard, historical and predictive analysis (statistics)

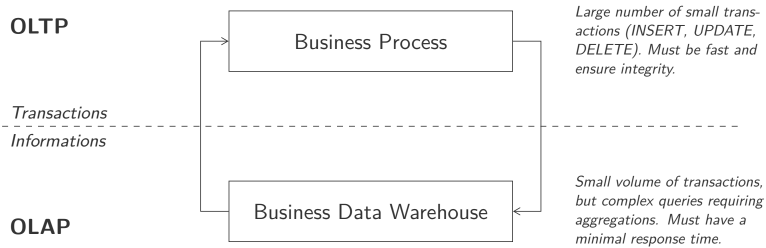
- Limitation of the relational model for OLAP

  Aggregates, query optimisation, indexing... not enough

IT System Data

- Division of an IT system in two parts

  A rather transactional part and a more analytical one



Relational Model Issues

1. Converting information from natural representation to tables
2. Reconstruction of the information from tables
3. Need to model data (semantic) before storing it
4. Rigid schema forcing data from one column to have the same type
5. Difficult to scale (scaling)
6. Difficulty making joins between different systems
7. Several existing dialects of SQL (portability)
8. Some business rules difficult to express in SQL
9. Approximate and fuzzy searches difficult
10. No efficient storage and validation of complex documents

June 11, 2009

- Meetup by Johan Oskarsson at the Hadoop summit @ SFO

  Software developer based in London for Last.fm

- Choosing a short name, memorable, with few Google results

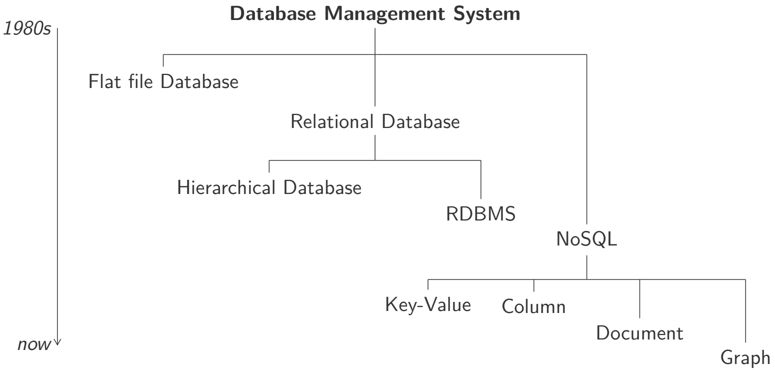
  #NoSQL "open-source, distributed, nonrelational databases"

- Several common characteristics to these databases
  - Do not use the relational model, nor SQL
  - Open source
  - Designed to be run on large clusters
  - Based on needs of web properties in the 21st century
  - No schema, possible to add field without control

DBMS Timeline

- Popular database management systems from 1980s to now

  With the co-existence of RDBMS and NoSQL today



The NoSQL World

http://nosql-database.org
http://nosql.mypopescu.com/kb/nosql

NoSQL Interest

- Increased productivity during the development
  - Time saving when mapping database to the memory
  - Less code to write, debug, maintain and evolve
- Amount of data on a large scale
  - Fast storage of large amounts of data
  - Database distributed on server clusters

NewSQL

- New tendency to combine strengths of SQL and NoSQL

  Guarantees from relational model with flexibility of NoSQL

- Often referred to as "SQL on Steroids" by the community
  - Based on the relational model and the SQL language
  - Scalability, flexibility and high performance from NoSQL

- ACID properties satisfied with horizontal scaling

  The power of NoSQL when heavy OLTP transaction volumes

ACID and BASE

ACID Properties

- Set of properties on transactions in databases

  Atomicity, Consistency, Isolation and Durability

- Definition by Reuter et Härder in 1983
  - One transaction makes all or nothing (*e.g.* if power failure)
  - Database changes from one valid state to another valid state
  - Concurrent execution of transactions as if they were sequential
  - A committed transaction confirmed and stored

BASE Properties

- Managing consistency loss by maintaining reliability

  Basically Available, Soft state et Eventual consistency

- Constraints relaxed compared to ACID properties
  - Always a response: failure or inconsistent data possible
  - The state can change over time, even when no input
  - The system will sooner or later be consistent

CAP Theorem

- CAP theorem stated by Eric Brewer for distributed systems

  Consistency, Availability et Partition tolerance

- Initially only calculation distribution and now data distribution

  Clusters or grids to increase the total computing power

- Three guarantees not satisfiable for a distributed system
  - **Consistency** of all data on all the nodes
  - **Availability** of all data even when losing of a node
  - **Partition tolerance** to a failure not disconnecting the cluster

Shared Something vs Nothing

- Shared-Nothing distributed-computing architecture
  - Nodes do not share any memory or storage
  - Each update request is satisfied by a single node
  - Elimination of single-point of failure, very easy to scale
- Shared-Something distributed-computing architecture
  - Hybrid approach between shared-everything and shared-nothing
  - Typically shared-memory nodes and interconnection network

ACID or BASE? (1)

- ACID desired in a *"shared something"* environment

  Pessimistic: *force consistency and end of transactions*

A	everything or nothing, commit ou rollback
C	no inconsistent data
I	no knowledge of concurrent transactions
D	committed transaction persistance

- BASE implemented in a *"shared nothing"* environment

  Optimistic: *accept temporary inconsistencies*

BA	guaranteed by replication
S	consistency to be guaranteed by the application
E	stale data possible, eventual consistency

ACID or BASE? (2)

- Conjecture related to the CAP theorem

  Only two of the three CAP requirements can be met

- Three situations are possible
  - CA $\sim$ ACID: one unique central server (with replication?)
  - CP: either *"w N, r 1"*, or *"w 1, r N"* (too slow?)
  - AP $=$ BASE: no strong consistency guaranteed

Big Data

Big Data

- Increase of the volume of data handled

  In particular companies and organisations related to the internet

- Exponential increase to petabytes of data (10^{15})
  - Scientific data, medical databases
  - Social networks, phone operators
  - Economic and social indicators
  - National territory defence agencies

- Challenging to manage and process this huge amount of data

  Not within the reach of the traditional relational model

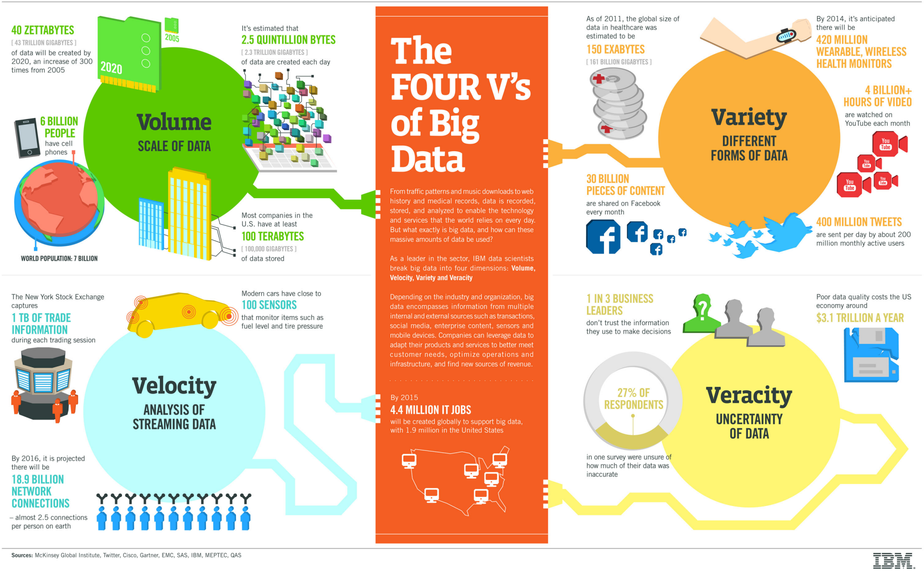
4Vs (1)

- Big Data characterised by an unlimited amount of datasets

  Data very complex to collect and to store

- Data follows the 4Vs
  - **Volume** of Pbytes, or even Ebytes, of data
  - **Velocity** for data creation, storage, analysis and visualisation
  - **Variety** of sources and types for data (image, video, sound...)
  - **Veracity** of data, obsolescence, integrity and security
- Two other Vs that are also important
  - **Validity** of data, correct and accurate to take decision
  - **Volatility** how long is data valid and should be stored

4Vs (2)

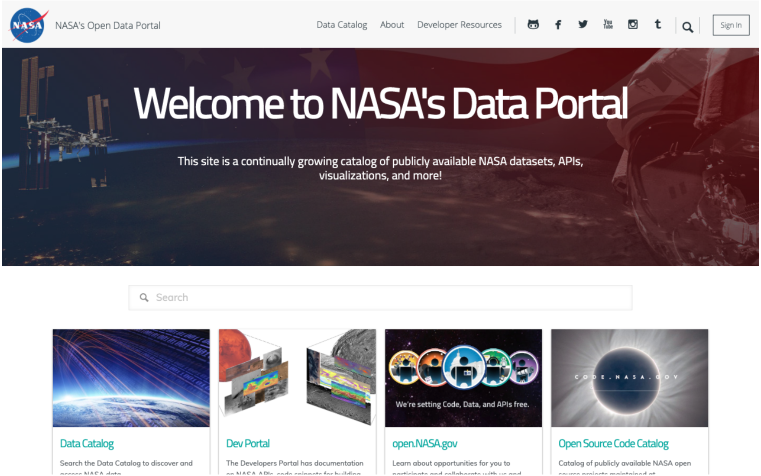

The FOUR V's of Big Data
From traffic patterns and music downloads to web history and medical records, data is recorded, stored, and analyzed to enable the technology and services that the world relies on every day. But what exactly is big data, and how can these massive amounts of data be used?
As a leader in the sector, IBM data scientists break big data into four dimensions: Volume, Velocity, Variety and Veracity
Depending on the industry and organization, big data encompasses information from multiple internal and external sources such as transactions, social media, enterprise content, sensors and mobile devices. Companies can leverage data to adapt their products and services to better meet customer needs, optimize operations and infrastructure, and find new sources of revenue.
By 2015
4.4 MILLION IT JOBS
will be created globally to support big data, with 1.9 million in the United States
Volume
SCALE OF DATA
40 ZETTABYTES
[43 TRILLION GIGABYTES]
of data will be created by 2020, an increase of 300 times from 2005
2005
2020
6 BILLION PEOPLE
have cell phones
WORLD POPULATION: 7 BILLION
It's estimated that
2.5 QUINTILLION BYTES
[2.3 TRILLION GIGABYTES]
of data are created each day
Most companies in the U.S. have at least
100 TERABYTES
[100,000 GIGABYTES]
of data stored
Velocity
ANALYSIS OF STREAMING DATA
The New York Stock Exchange captures
1 TB OF TRADE INFORMATION
during each trading session
Modern cars have close to
100 SENSORS
that monitor items such as fuel level and tire pressure
By 2016, it is projected there will be
18.9 BILLION NETWORK CONNECTIONS
– almost 2.5 connections per person on earth
Variety
DIFFERENT FORMS OF DATA
As of 2011, the global size of data in healthcare was estimated to be
150 EXABYTES
[161 BILLION GIGABYTES]
30 BILLION PIECES OF CONTENT
are shared on Facebook every month
By 2014, it's anticipated there will be
420 MILLION WEARABLE, WIRELESS HEALTH MONITORS
4 BILLION+ HOURS OF VIDEO
are watched on YouTube each month
400 MILLION TWEETS
are sent per day by about 200 million monthly active users
Veracity
UNCERTAINTY OF DATA
1 IN 3 BUSINESS LEADERS
don't trust the information they use to make decisions
27% OF RESPONDENTS
in one survey were unsure of how much of their data was inaccurate
Poor data quality costs the US economy around
$3.1 TRILLION A YEAR
Sources: McKinsey Global Institute, Twitter, Cisco, Gartner, EMC, SAS, IBM, MEPTEC, QAS
IBM

Open Data

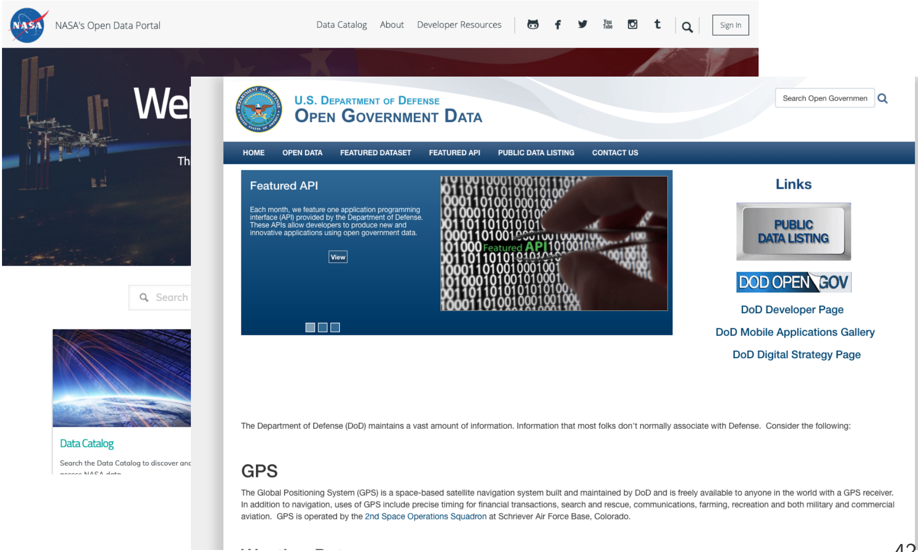
- Some open data should be freely available to everyone
  - To use and to republish as they wish
  - Can be related to linked data, resulting in linked open data
- Sometimes data only accessible under specific conditions
  - Open may be a problem with commercially valuable data
  - Access restriction, license, copyright, patent, charge, etc
- A large variety of sources do provide open data

  Government, public or private companies, research centres, etc.

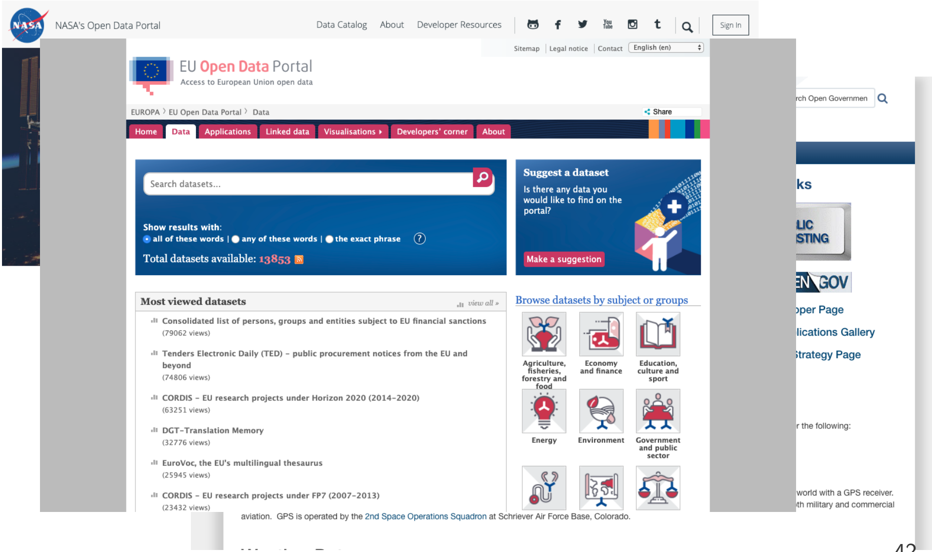
Open Data Source



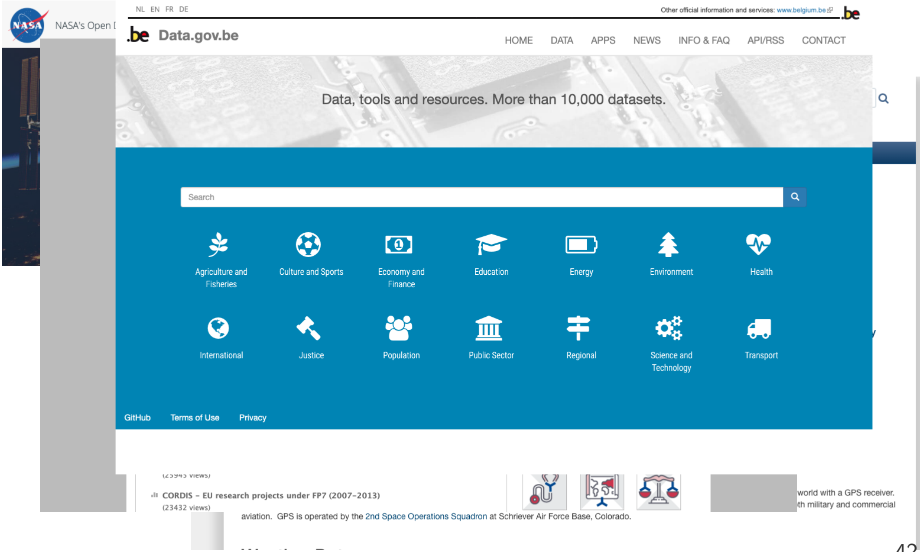
Open Data Source



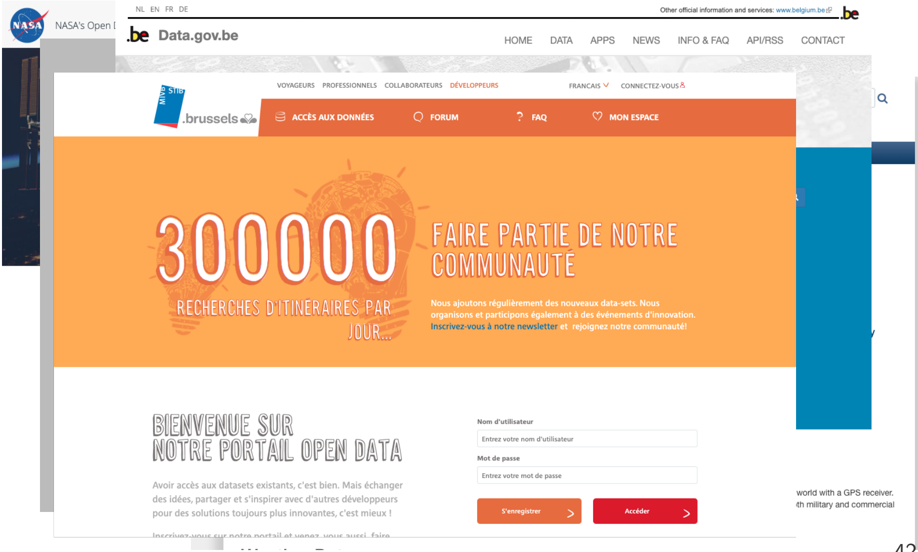
Open Data Source



Open Data Source



Open Data Source



Main Actors

Google

THE APACHE SOFTWARE FOUNDATION

facebook

amazon.com

YAHOO!

Microsoft

IBM

Technological Evolution

- First DBMS built around mainframes

  With the limitations of storage capacities of those time

- Several technological evolutions removed those constraints
  - Generalisation of network interconnections
  - Increased available bandwidth over internet
  - Decreased cost of commodity machines

Google FS

- Proprietary distributed file system developed by Google

  Google File System (GFS) presented in 2003

- Redundant and resilient storage on a cluster of machines

  Average and "disposable" power (commodity hardware)

- FGS has several nice characteristics
  - Designed for machine-machine interactions
  - Executed in the user space, not in OS kernel space
  - Manage files of several gigabytes
  - Automatic replication of data by chunkservers

MapReduce

- Programming paradigm and associated implementation
  - Processing and generation of large amounts of data
  - Parallel algorithm distributed on a cluster
- Implementation based on two functions
  - `Map` performs an operation on a list (sort, filter...)
  - `Reduce` groups data into one single result (sum, max...)

Apache Hadoop

- Hadoop open source implementation of MapReduce in Java

  By Doug Cutting, named after his son's toy elephant

- Hadoop Distributed FileSystem (HDFS)

  Inspired by the overview publication on GFS

- Framework used by many companies
  - Supported by Microsoft (on Windows Azure and Server)
  - Yahoo! cluster with 4000 machines, soon 10000 with v2.0
  - Facebook announces installation of HDFS with 100 petabytes

BigTable

- GFS-based data management system

  Proprietary solution again developed by Google

- Data consistency management and distribution on GFS

  Just working like a gigantic distributed hash table

- Several open source implementation HBase (Apache)

  For example used by eBay, Yahoo! and Twitter

Dynamo

- Distributed and proprietary key-value pairs storage (Amazon)

  Implemented by Amazon in Simple Storage Service (S3)

- Four key principles of Dynamo storage system
  - Incremental scalability with no influence on operator/system
  - Symmetry with all the nodes being equal
  - Complete decentralisation with no central role
  - Heterogeneity by sharing work according to resources

- Creation of several NoSQL engines based on Dynamo

  Cassandra, Riak, Voldemort project (LinkedIn)...

Data Model

Impedance mismatch (1)

- Object-relational impedance mismatch with SQL

  Moving from relational to object is done with an impedance

- Difference between relational model and memory structure

  Relations and tuples versus complex data structures

- Appearance of object oriented programming languages

  Object-Relational Mapping (ORM) such as Hibernate...

Impedance mismatch (2)

```python
class Address:
    def __init__(self, street, number, zipcode, city):
        self.__address = (street, number, zipcode, city)

    def __str__(self):
        return '{}, {}\n{} {}'.format(*self.__address)

ecam = Address("Promenade de l'Alma", 50,
               1200, "Woluwé-Saint-Lambert")
wolubilis = Address("Cours Paul-Henri Spaak", 1,
               1200, "Woluwé-Saint-Lambert")
```

Address

Id	Street	Number	CityID
1	Promenade de l'Alma	50	1
2	Cours Paul-Henri Spaak	1	1

City

Id	Zipcode	Name
1	1200	Woluwé-Saint-Lambert

Integration vs Application (1)

- Coordination of several applications around data

  Sharing data in a single common database

- Difficult to change the structure of the database

  Non-trivial to ensure data integrity

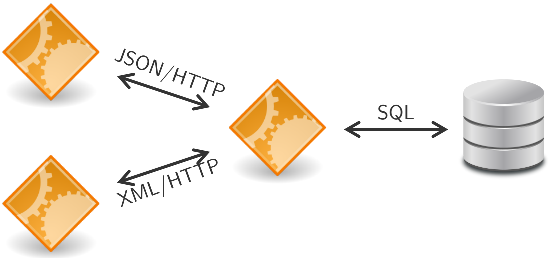
Integration vs Application (2)

- Access to the database by a single application

  Provides an access interface to other applications

- Web services deployment and services oriented architecture

  Greater format flexibility for exchanged data



Data Model (1)

- Model with which data are perceived and manipulated

  Different from the disk storage model

- Relational model consists in tables with rows

  Columns with values that can reference other rows

- Moving to a model representing a collection of aggregates

  Unit of information processed, stored and exchanged atomically

Data Model (2)

- Four main data models in the NoSQL world

  Detailed analysis of one example for each model

- No single and unambiguous classification

  Some databases cover several models

Data Model	**Database examples**
Key-Value	BerkeleyDB, Memcached, **Redis**, Riak...
Document	CouchDB, **MongoDB**, OrientDB...
Column	Amazon Simple DB, **Cassandra**, HBase...
Graph	FlockDB, HyperGraphDB, **Neo4j**, OrientDB...

Aggregate

- Operation on complex and structured data units

  To overcome the limitations of the relational model tuples

- Possible to nest lists and other structures in aggregate

  Different "objects" handled as units

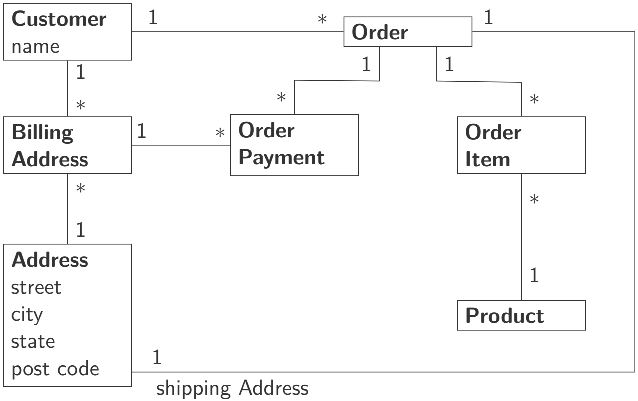
- Aggregate is the unit to handle and manage concurrency

  Facilitating the distribution of data on clusters

Relation vs Aggregate (1)

- Fully normalised model without any duplicate data

  May required a lot of entities and associations



Relation vs Aggregate (2)

Customer

Id	Name
1	Martin

Order

Id	CustomerId	ShippingAddressId
99	1	77

Product

Id	Name
27	NoSQL Distilled

BillingAddress

Id	CustomerId	AddressId
55	1	77

OrderItem

Id	OrderId	ProductId	Price
100	99	27	32.45

Address

Id	City
77	Chicago

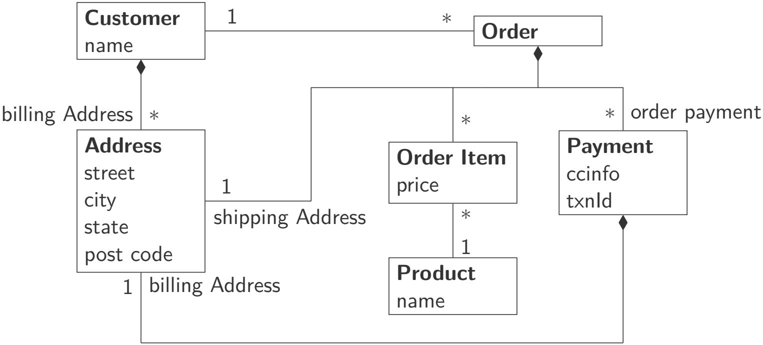
OrderPayment

Id	OrderId	CardNumber	BillingAddressId	txnId
33	99	1000-1000	55	abelif879rft

Relation vs Aggregate (3)

- Model consisting of two main aggregates

  Customer and Order composed of "sub-aggregates"



Relation vs Aggregate (4)

```
# Customer
{
    "id": 1,
    "name": "Martin",
    "billingAddress": [{"city": "Chicago"}]
}

# Order
{
    "id": 99,
    "customerId": 1,
    "orderItems": [{
        "productId": 27,
        "price": 32.45,
        "productName": "NoSQL Distilled"
    }],
    "shippingAddress": [{"city": "Chicago"}],
    "orderPayment": [{
        "ccinfo": "1000-1000",
        "txnId": "abelif879rft",
        "billingAddress": {"city": "Chicago"}
    }]
}
```

Key-Value

- Aggregate stored in key-value form

  The key acts as the unique identifier of each aggregate

- An aggregate is retrieved thanks to its key

  Key-value stores work as lookup tables

Document

- Aggregate stored in document form

  Each document is uniquely identified by an ID

- Retrieving the whole document or part of a document

  From queries on the fields of the aggregate

- Creation of an index based on the content of documents

  To speed up search operations in the database

Column

- Columns are stored on the disk instead of rows

  Column storage can be seen as a two-level map

- Key-value structure with row identifier as key

  The second level contains information about the columns

Graph

- Possible to have relations between aggregates

  With automatic update possibility

- Useful for small records with a lot of links

  Set of nodes connected by edges

- Social networks, preferences, eligibility rules...

  "What are all the things Theo and Yannis both like?"

neo4j OrientDB


NoSQL Characteristics
Lizcano Photography

Schema-Less Database

- NoSQL databases do not have data schema

  Unlike the rigid structure imposed by the relational model

- Unrestricted addition of data of any type

  Such as key, document, column, edge and properties

- Possible to store non-uniform data

  Which eliminates the need to have `NULL` values

Implicit Schema

- Assumptions about the data structure in the code

  The database remains ignorant, it is the application that checks

- Danger if multiple applications on the same database

  They have to agree on the data schema

- Data migration must always be done carefully

  Should it be with the relational or NoSQL models

Developer Centered

- Developer centered development methodology
  - Design and implementation of the application architecture
  - Data modelling
- Two different approaches RDBMS vs NoSQL
  - Relational data models are defined thanks to theory
  - Application queries and configuration to be supported

NoSQL Architecture

- Building database on relational with DBMS
  - Description of data structures and storage
  - Data recovery process and reliability
  - Data in tables (record and column) and not repeated
  - Importance of primary keys
- Data management much more flexible with NoSQL
  - Distribution across multiple servers, platforms, processors
  - Gradual evolution of the (implicit) data schema

Data Operation and Relation

- CRUD standard operations with RDBMS

  Create, Read, Update, Delete

- Much more diverse and varied operations in NoSQL
  - Large number of additions and updates
  - Operations on other entities than rows of tables

- NoSQL not adapted to data with a lot of relations

  RDBMS have one-to-one, one-to-many and many-to-many

References

- Emmanuel Rodriguez, *What does NoSQL mean?*, April 23, 2019.
  https://medium.com/@emrod718/what-does-nosql-mean-ec79619948a3
- Two-Bit History, *The Most Important Database You've Never Heard of*, October 7, 2017.
  https://twobithistory.org/2017/10/07/the-most-important-database.html
- Edgar E. F. Codd, *A relational model of data for large shared data banks*, 1970, Communications of the ACM, Vol. 13, No. 6, pp.377-387.
- Donald D. Chamberlin, Morton M. Astrahan, Michael W. Blasgen, James N. Gray, W. Frank King, BruceG. Lindsay, Raymond Lorie, James W. Mehl, Thomas G. Price, Franco Putzolu, Patricia G. Selinger, Mario Schkolnick, Donald R. Slutz, Irving L. Traiger, Bradford W. Wade, & RobertA. Yost, *A history and evaluation of System R*, 1981, Communications of the ACM, Vol. 24, No. 10, pp.632-646.
- Hasso Plattner Institut, *RDBMS Genealogy*, October 2018. https://hpi.de/naumann/projects/rdbms-genealogy.html
- Martin Fowler, *NosqlDefinition*, January 9, 2012. https://martinfowler.com/bliki/NosqlDefinition.html
- Syed Sadat Nazrul, *CAP Theorem and Distributed Database Management Systems*, April 24, 2018.
  https://towardsdatascience.com/cap-theorem-and-distributed-database-management-systems-5c2be977950e
- IBM Big Data & Analytics Hub, *The Four V's of Big Data*. https://www.ibmbigdatahub.com/infographic/four-vs-big-data

Credits

- Logos pictures from Wikipedia.
- Oliver Widder (Geek and Poke), January 27, 2011, http://geekandpoke.typepad.com/geekandpoke/2011/01/nosql.html.
- N i c o l a, March 22, 2014, https://www.flickr.com/photos/15216811@N06/13495437084.
- https://openclipart.org/detail/34537/tango-drive-hard-disk.
- https://openclipart.org/detail/59167/magnifying-glass.
- Scott Adams (Dilbert), September 20, 2013, http://dilbert.com/strip/2013-09-20.
- Stanley Zimny, December 3, 2011, https://www.flickr.com/photos/stanzim/10457956743.
- Paul Rand, 1972, https://en.wikipedia.org/wiki/File:IBM_logo.svg.
- SchuminWeb, November 25, 2010, https://en.wikipedia.org/wiki/File:Edgar_F_Codd.jpg.
- Hydrargyrum, July 23, 2015, https://en.wikipedia.org/wiki/File:Pratt_%26_Whitney_UTC_logo.svg.
- biologycorner, August 24, 2009, https://www.flickr.com/photos/biologycorner/3870547053.
- luckey_sun, April 9, 2012, https://www.flickr.com/photos/75279887@N05/6914441342.
- Luke Vu, November 20, 2011, https://www.flickr.com/photos/lukevu/6380430175.
- https://openclipart.org/detail/94723/database-symbol.
- https://openclipart.org/detail/35407/tango-applications-other.
- Diego Lizcano, June 25, 2017, https://www.flickr.com/photos/diegolizcano/35522041701.