NoSQL (Not Only SQL)

CMT220
Databases & Modelling

Cardiff School of Computer Science & Informatics
http://www.cs.cf.ac.uk

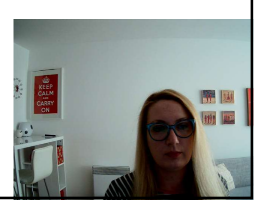
CARDIFF UNIVERSITY
PRIFYSGOL CAERDYDD

1

1

Lecture

- in this module we learnt about relational databases
- in this lecture we will learn about the latest type(s) of databases
  - NoSQL



CARDIFF UNIVERSITY
PRIFYSGOL CAERDYDD

2

Big data

3

Big data

- modern data collection technologies (social media, smartphones, sensors, etc.) act as force multipliers for data growth
- big data is a broad term for datasets so large or complex that traditional data processing applications (e.g. RDBMS) are inadequate
- big data project is defined by 3V + C:

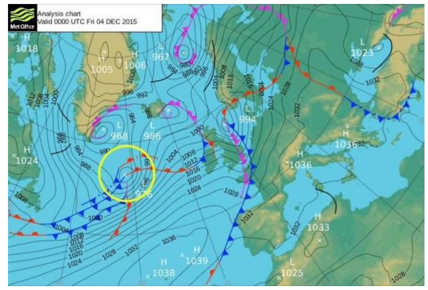
1. velocity — data is streamed at an unprecedented speed and must be dealt with in near–real time
2. variety — data in various formats: structured, semi–structured and unstructured
3. volume — data that involves many terabytes or petabytes
4. complexity — data coming from multiple sources need to be connected and correlated

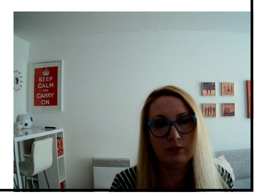
4

4

Velocity

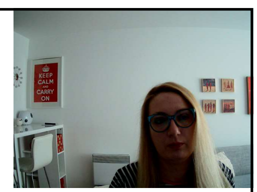
- How fast is the data produced/processed?
- gathering data quickly is of no benefit is we analyse it once a week
- real–time analytics is about using very current data to provide information that will help improve a service or respond to demand swiftly





5

Variety



- data comes in all types of formats
  - from structured, numeric data in traditional databases ...
  - ... to unstructured text documents, email, video, audio, stock ticker data and financial transactions

6

6

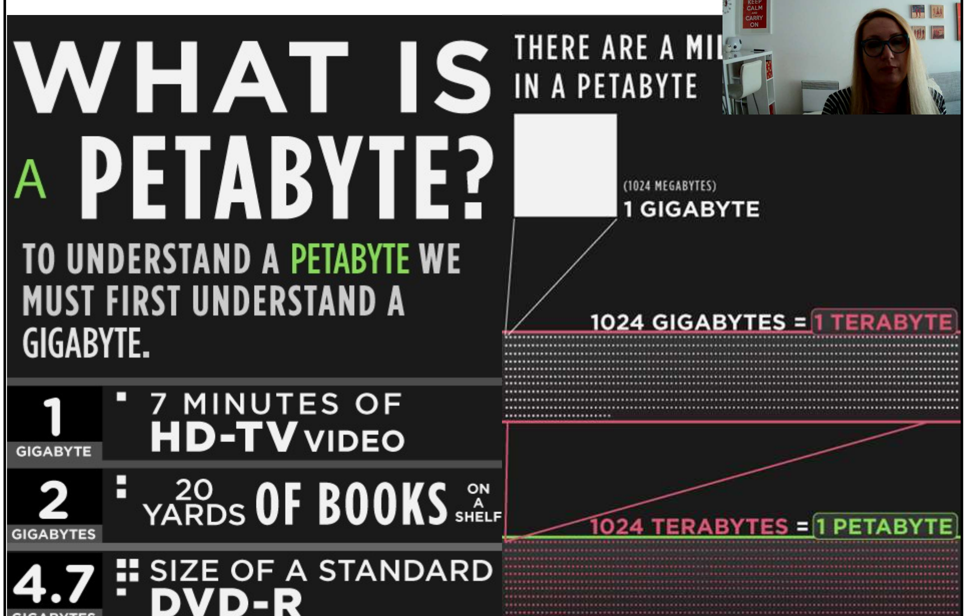
Volume

- How much data?

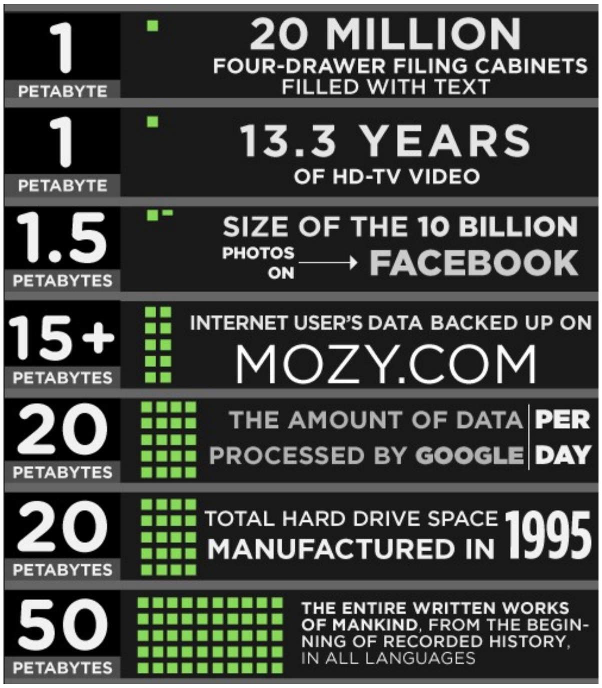
CARDIFF UNIVERSITY
PRIFYSGOL CAERDYDD

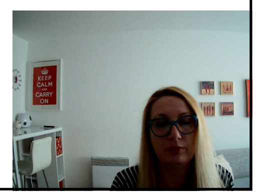
7

7



8



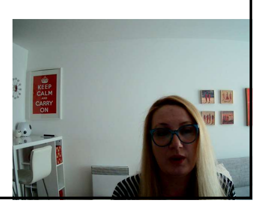


9

WORLDWIDE DATACENTERS NOW ANNUALLY CONSUME

AS MUCH ENERGY AS SWEDEN

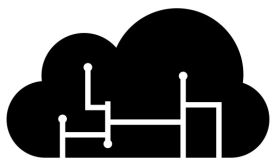
CARDIFF UNIVERSITY PRIFYSGOL CAERDYDD

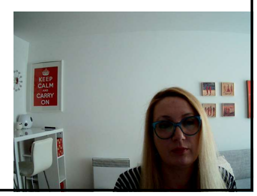


10

Complexity

- today's data comes from multiple sources in a variety of formats
- this makes it difficult to link, match, cleanse and transform data across systems
- however, it is necessary to connect and correlate relationships, hierarchies and multiple data linkages
- otherwise, data can quickly spiral out of control





11

12

What is NoSQL?

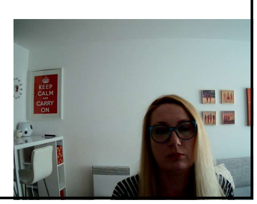
- **N**ot **O**nly SQL databases AKA cloud databases, non–relational databases, Big Data databases ...
- a non–relational and largely distributed database system that enables rapid, ad–hoc organisation and analysis of extremely high–volume, disparate data types
- developed in response to the sheer volume of data being generated, stored and analysed by modern users and their applications
- NoSQL databases have become the first alternative to relational databases, with scalability, availability and fault tolerance being key deciding factors

13

13

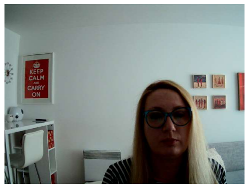
NoSQL databases

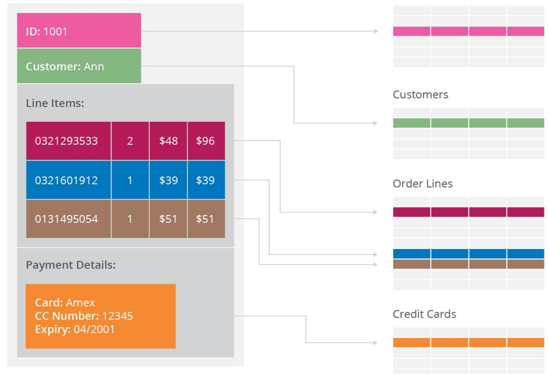
- schema–less data model
- horizontal scalability
- distributed architectures
- the use of languages and interfaces that are "not only"



14

Why NoSQL?



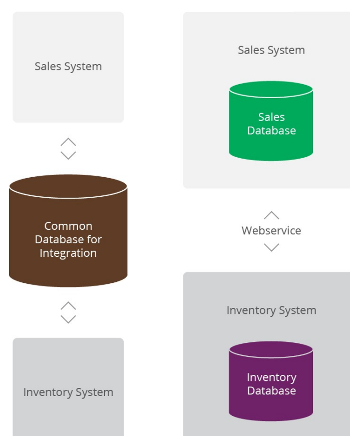


- impedance mismatch between the relational data structures and the in–memory data structures of the application
- with NoSQL databases developers do not have to convert in–memory structures to relational structures

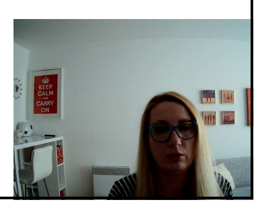
15

15

Why NoSQL?



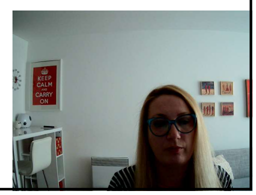
- moving away from using databases as integration points
- encapsulating databases with applications and integrating using services instead
- the rise of the web as a platform created a vital factor change in data storage
- need to support large volumes of data by running on clusters
- relational databases were not designed to run efficiently on clusters



16

Why NoSQL?

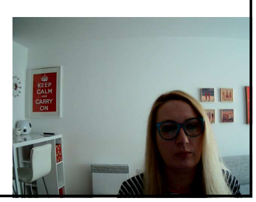
- NoSQL technology was originally created and used by Internet leaders such as Facebook, Google, Amazon and others
- they required a DBMS that could write and read data anywhere in the world
- ... while scaling and delivering performance across massive data sets and millions of users
- today, almost every organisation has the need to deliver applications that utilise Web, mobile and IoT technologies



17

Aggregate data model

- relational data modelling is vastly different from the types of data structures that application developers use
- movement away from relational modelling and towards aggregate models
- an aggregate is a collection of data that we interact with as a unit
- the unit of data can reside on any machine and when retrieved gets all the related data along with it
- aggregates make it easier for the database to manage data storage over clusters
- ... but aggregate–oriented databases make inter–aggregate relationships more difficult to handle



18

Distribution models

- aggregate–oriented databases make distribution of data easier, since all related data is contained in the aggregate
- the distribution mechanism has to move the aggregate and not have to worry about related data
- two styles of distributing data:

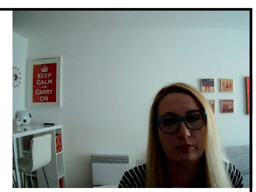
1. **sharding** distributes different data across multiple servers, so each server acts as the single source for a subset of data
2. **replication** copies data across multiple servers, so each bit of data can be found in multiple places

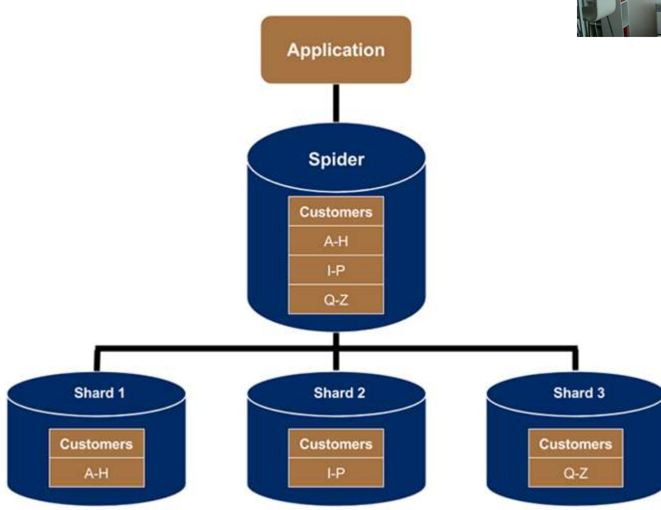
CARDIFF UNIVERSITY PRIFYSGOL CAERDYDD

19

19

Distribution model: sharding





CARDIFF UNIVERSITY PRIFYSGOL CAERDYDD

20

20

Distribution model: replication

- data are copied across multiple servers
- two forms of replication:

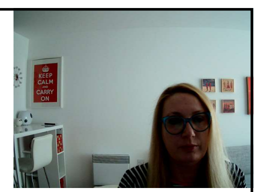
1. master–slave replication makes one node the authoritative copy that handles writes while slaves synchronise with the master and may handle reads
2. peer–to–peer replication allows writes to any node; the nodes coordinate to synchronize their copies

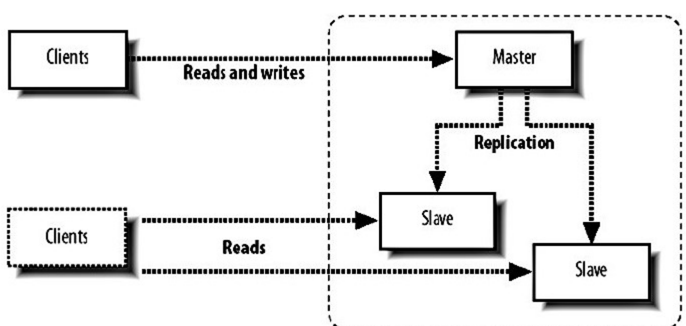
- master–slave replication reduces the chance of update conflicts
- peer–to–peer replication avoids loading all writes onto a single server creating a single point of failure

21

21

Master–slave replication

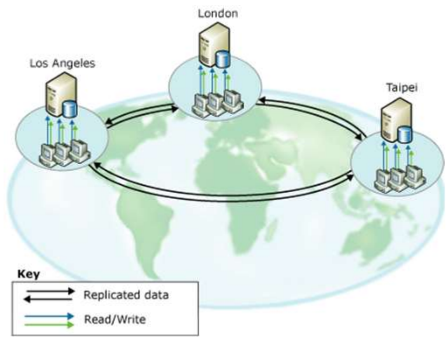




22

22

Peer–to–peer replication

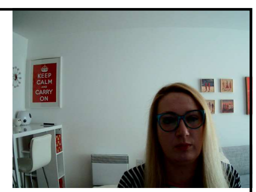


CARDIFF UNIVERSITY PRIFYSGOL CAERDYDD

23

23

Types of NoSQL databases



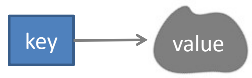
1. key–value store – all data consists of an indexed key and a value
2. document database – expands on the basic idea of key–value stores
   - documents contain more complex data and each document is assigned a unique key
3. column-family store – store data tables as sections of columns of data, rather than rows of data
4. graph database – designed for data whose relations are well represented as a graph

CARDIFF UNIVERSITY PRIFYSGOL CAERDYDD

24

24

Key–value store

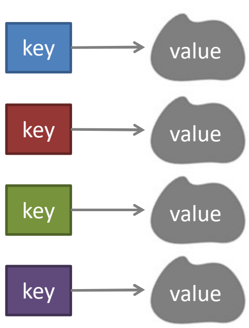


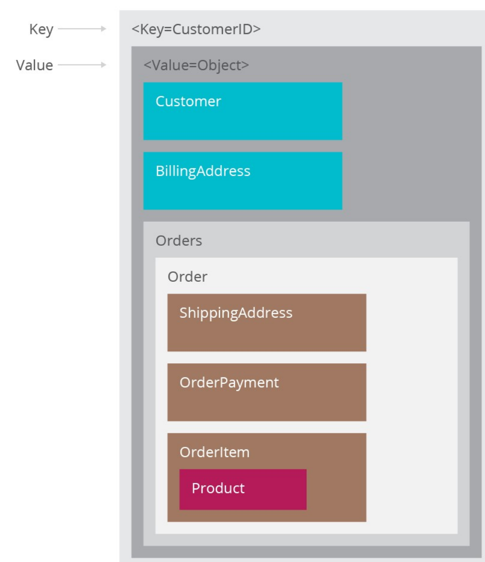
- simplest NoSQL database to use from an API perspective
- store data in a schema–less way
- the value is a blob that is just stored, without caring what's inside; it is the responsibility of the application to understand what was stored
- the client can either get the value for the key, put a value for a key or delete a key from the store
- key–value stores always use primary–key access, so they generally have great performance and can be scaled easily
- e.g. Memcached, Berkeley DB, Amazon DynamoDB, Couchbase

25

25

Key–value store

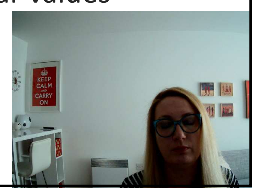




26

Document database

- document databases expand on the basic idea of key–value stores by storing store documents in the value part
- they store, retrieve & manage documents
  - semi–structured data
  - e.g. XML, JSON, BSON, etc.
  - self–describing, hierarchical data structures, which can consist of maps, collections and scalar values
- e.g. MongoDB, CouchDB , Terrastore, OrientDB, RavenDB



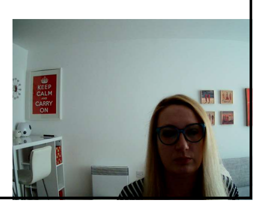
27

Document database

<Key=CustomerID>

```
{
   "customerid": "fc986e48ca6"
   "customer":
   {
   "firstname": "Pramod",
   "lastname": "Sadalage",
   "company": "ThoughtWorks",
   "likes": [ "Biking","Photography" ]
   }
   "billingaddress":
   { "state": "AK",
     "city": "DILLINGHAM",
     "type": "R"
    }
}
```

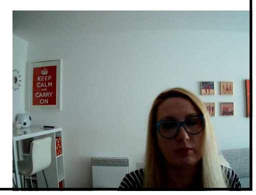
Key



28

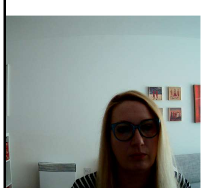
Column–family store

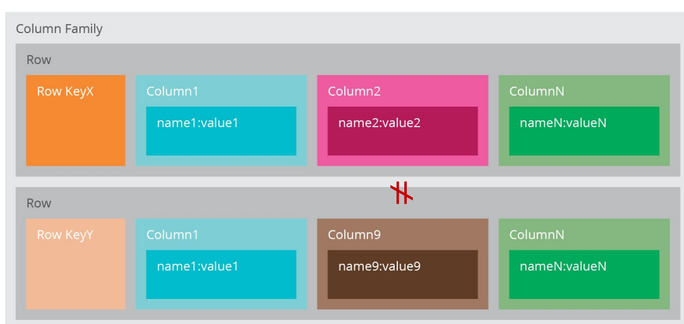
- AKA column store or wide–column store
- column family is a group of related data that is often accessed together
  - e.g. for a customer, we would often access their profile information at the same time, but not their orders
- column–family databases store data in column families as rows that have many columns associated with a row key
- very high performance and a highly scalable architecture
- e.g. Cassandra, HBase, HyperTable



29

Column–family store vs. relational database





- similarity: each column family corresponds to a container of rows in a table where the key identifies the row and the row consists of multiple columns
- difference: various rows do **not** have to have the same columns, and columns can be added to any row at any time without having to add it to other rows

30

30

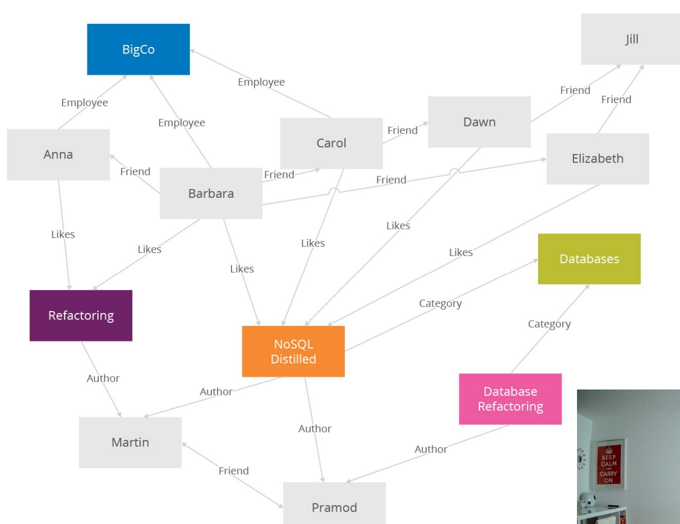
Graph database

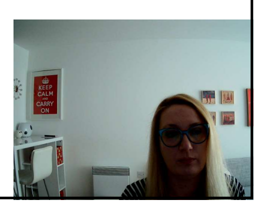
- graph databases store entities and relationships between these entities
  - entities are also known as nodes, which have properties
  - relationships are known as edges, which can also have properties
  - edges have directional significance
- the organisation of the graph lets the data to be stored once and then interpreted in different ways based on relationships
- e.g. Neo4J, Infinite Graph, OrientDB

31

31

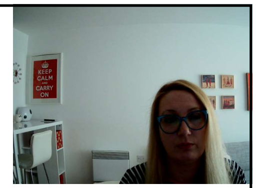
Graph database





32

Graph database



- most of the value from the graph databases comes from the relationships and their properties
- relationships are first–class citizens in graph databases
- there is no limit on the number and types of relationships a node can have
- relationships have a type, a start node, an end node, but can also have properties of their own
- these properties can be used to query the graph
- e.g. when did they become friends, what is the distance between the nodes, what aspects are shared between the nodes, etc.

33

33

NoSQL vs. SQL summary

	SQL databases	NoSQL databases
Types	**One** type with minor variations	**Many** different types
Development history	Developed in **1970s** to deal with the first wave of data storage applications	Developed in late **2000s** to deal with limitations of SQL databases, especially scalability, multi-structured data, geo-distribution and agile development
Examples	MySQL, PostgreSQL, Microsoft SQL Server, Oracle	MongoDB, Cassandra, HBase, Neo4j
Data storage models	Related data are stored in separate **tables**, and then joined together when more complex queries are executed.	**Varies** based on database type. Key-value stores function similarly to SQL databases, but have only two columns (key & value). Document databases store all relevant data together in single document, e.g. in JSON or XML, which can nest values hierarchically.

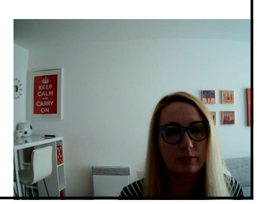
34

Cont.	SQL databases	NoSQL databases
Schemas	Structure and data types are **fixed in advance**.	Typically **dynamic**. Applications can add new fields on the fly, and unlike SQL table rows, dissimilar data can be stored together as necessary.
Scaling 	**Vertically**, meaning a single server must be made increasingly powerful in order to deal with increased demand.	**Horizontally**, meaning that to add capacity, a database administrator can simply add more commodity servers or cloud instances. The database automatically spreads data across servers as necessary.
Development model	Mix of **open-source** (e.g. PostgreSQL, MySQL) and **closed source** (e.g. Oracle)	**Open-source**
Supports transactions	**Yes**, updates can be configured to complete entirely or not at all	In **certain circumstances** and **at certain levels** (e.g. document level vs. database level)
Data manipulation	Specific language (**SQL**) using SELECT, INSERT and UPDATE statements	Through **object-oriented APIs**
Consistency	Can be configured for **strong consistency**	**Depends** on product. Some provide strong consistency (e.g. MongoDB) whereas others offer eventual consistency (e.g. Cassandra).

35

NoSQL vs. SQL databases

- designed to support different application requirements
- they typically co-exist in most enterprises
- it is **not** a question of either ... or!



36

NoSQL vs. SQL databases

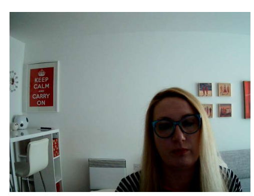
- key decision points on when to use which:

Use SQL when you need/have...	Use NoSQL when you need/have...
Centralised applications (e.g. business management)	**Decentralised** applications (e.g. Web, mobile and IoT)
Moderate to high availability	**Continuous** availability; no downtime
Moderate velocity data	**High** velocity data (devices, sensors, etc.)
Data coming in from **one/few locations**	Data coming in from **many locations**
Primarily **structured** data	Structured, with **semi/unstructured**
Complex/nested transactions	**Simple** transactions
Primary concern is scaling **reads**	Concern is to scale both **writes** and **reads**
Philosophy of scaling **up** for more users/data	Philosophy of scaling **out** for more users/data
To maintain **moderate** data volumes with purge	To maintain **high** data volumes; retain forever

37

A history of databases in No–tation

- 1970: NoSQL = We have no SQL
- 1980: NoSQL = Know SQL
- 2000: NoSQL = No SQL!
- 2005: NoSQL = Not only SQL
- 2013: NoSQL = No, SQL!



38