NoSQL Systems @ RomaTre

Luca Rossi
luca.rossi.917@gmail.com

Francesca Bugiotti
franbugiotti@yahoo.it

Outline

- **What is NoSQL**
  - *"One size does not fit all"*
  - Main themes
  - Timeline
  - Adoption
- **SOS - Save Our Systems**
  - NoSQL Issues
  - What we are working on

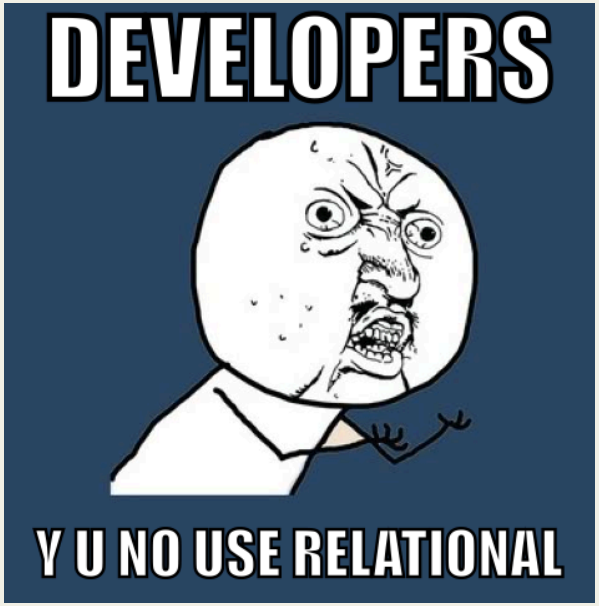
"One size does not fit all"

- **Mismatch** between RDBMS features and applications requirements
  - Some features are unneeded
  - Others are lacking (and desirable)

- **Choose the right tool for your needs**

Why not Relational?

- **Unneeded complexity**
  - Rigid schemas
  - ACID properties



- **Hard to scale**
  - Complexity of setting up database clusters
  - Running on commodity hardware

NoSQL Timeline

- **2003** – Memcached
- **2006** – *Google BigTable*
- **2007** – *Amazon Dynamo*

Pioneers

- **2007** – HBase
- **2008** – Cassandra
- **2009** – Redis, Riak, MongoDB, Voldemort, …

Open source projects

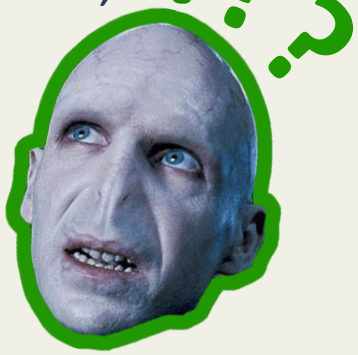
NoSQL Timeline

- **2003** – Memcached
- **2006** – *Google BigTable*
- **2007** – *Amazon Dynamo*

Pioneers

- **2007** – HBase
- **2008** – Cassandra
- **2009** – Redis, Riak, MongoDB, **Voldemort**, …

Open source projects



NoSQL Timeline

- **2003** – Memcached
- **2006** – *Google BigTable*
- **2007** – *Amazon Dynamo*

Pioneers

- **2007** – HBase
- **2008** – Cassandra
- **2009** – Redis, Riak, MongoDB, Voldemort, …

Open source projects

…

- **Late 2011** – Oracle NoSQL

Commercial players

Rise of NoSQL

- **Strong adoption by both** big companies **and** startups
- **Different focuses:**
  - **Big companies**: scalability, performance
  - **Startups**: flexibility, performance, low cost
- **An impressive lineup:**
  - **Facebook:** Cassandra, HBase
  - **Google:** BigTable
  - **Amazon:** SimpleDB, DynamoDB
  - **Twitter:** FlockDB, HBase
  - **LinkedIn:** Voldemort, SenseiDB
  - …

Some recurring features

- Replication and distribution of data across many servers
- Flexible data models
  - Ability to dynamically add new attributes to data records
- Simple interfaces (no SQL)
  - Focus on simple operations
- No standardization ☹

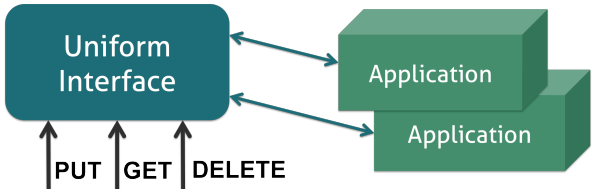
…and some recurring problems

- **What if:**
  - I want to use many DBMSes at the same time
  - I want to migrate my data
  - I want to decouple my app from a specific technology
- **Reverse the canonical problem:**
  - One size (DBMS) does not fit all (apps)…
  - …but one size (your app) should fit all (the DBMSes)

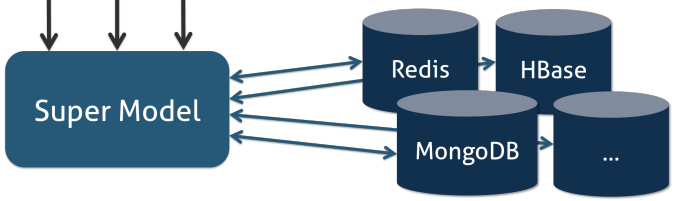
SOS – Save our systems

- **Goal:**
  - Seamless access to different NoSQL databases.
- **Requirements:**
  - Lightweight: small footprint on performances
  - Coherent: with main NoSQL themes and features
    - Hint: do not reimplement SQL
      - Seriously, someone has done it
  - Scalable: easily extendable to different technologies and DBMSes

SOS – Save our systems



Common Interface
to access different NoSQL systems



Common Data Model
instances are mapped to the DBMSes of choice

- **SOS is a *Database Access Layer* between the app and the data store**
  - It collects data from the interface and seamlessly manages its translation and deployment to specific DBMSes
- **Implementations provided for three data stores belonging to different families:**
  - HBase (column-based store)
  - Redis (key-value store)
  - MongoDB (document store)

Usage example

```
public class Studente {

    private String matricola;
    private String nome;
    private String cognome;

    private Set<Verbalizzazione> verbs;
    ...
```

```
Studente luca = new Studente(...);

DatabaseHandler db = new HBaseHandler();
db.put("studenti", luca.getMatricola(), luca);
```

Usage example

```
public class Studente {

    private String matricola;
    private String nome;
    private String cognome;

    private Set<Verbalizzazione> verbs;
    ...
```

```
Studente luca = new Studente(...);

DatabaseHandler db = new HBaseHandler();
db.put("studenti", luca.getMatricola(), luca);
```

Collection name

Object Id

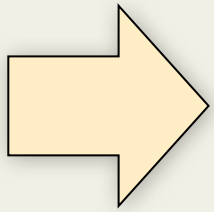
Object

Usage example

```java
public class Studente {

   private String matricola;
   private String nome;
   private String cognome;
   private Set<Verbalizzazione> verbs;
   ...
```

Java



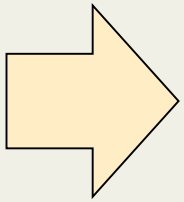
```
{
    matricola = ”281283”,
    nome = “Luca”,
    cognome = ”Rossi”,
    verbalizzazioni = [
        {
            id = ”10001”,
            corso = {
                id = ”20001”,
                nome = ”Basi di Dati”
            };
            data = ”12/06/2011”,
            voto = 28
        },
        {
            id = ”10002”,
            corso = {
                id = ”20004”,
                nome = ”Crittografia”,
            };
            data = ”21/05/2011”,
            voto = 30
        }
    ]
}
```

JSON

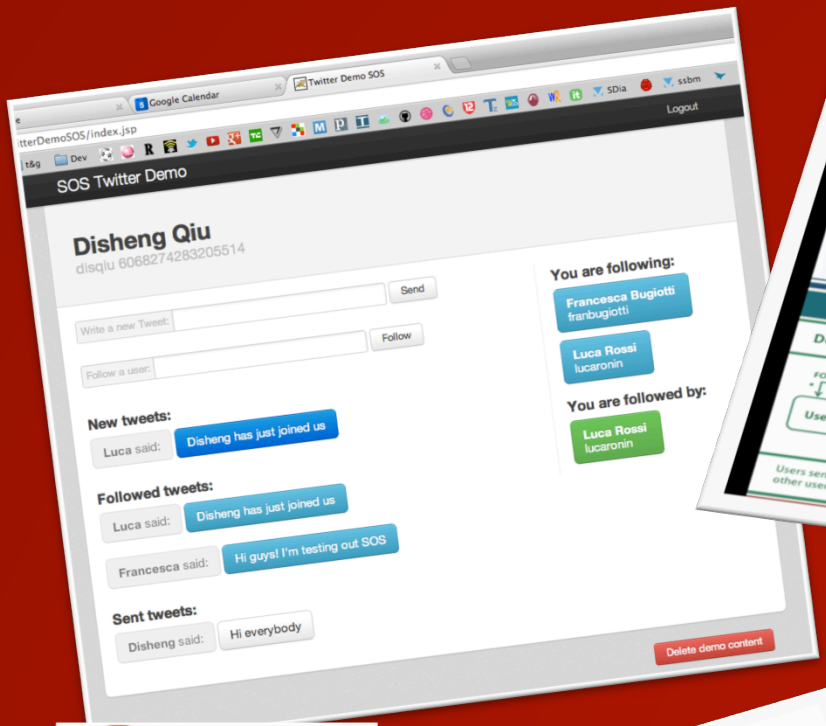
Usage example

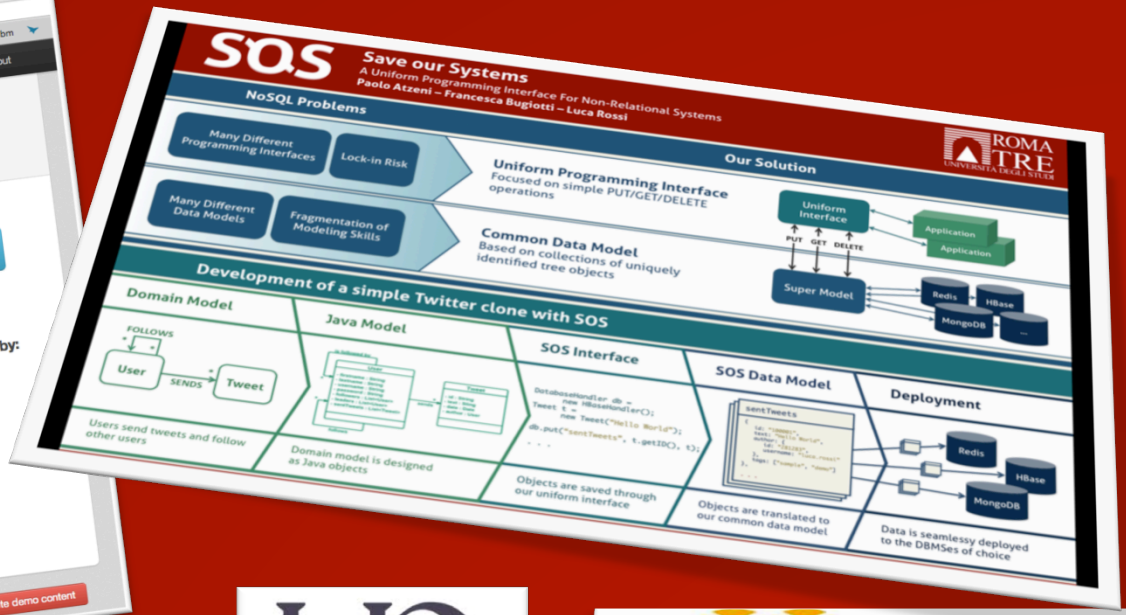
```
{
    matricola = ”281283”,
    nome = “Luca”,
    cognome = ”Rossi”,
    verbalizzazioni = [
        {
           id = ”10001”,
           corso = {
               id = ”20001”,
               nome = ”Basi di Dati”
           };
           data = ”12/06/2011”,
           voto = 28
        },
        {
           id = ”10002”,
           corso = {
               id = ”20004”,
               nome = ”Crittografia”,
           };
           data = ”21/05/2011”,
           voto = 30
        }
    ]
}
```

JSON



HBase	
_top	verbalizzazioni[]
matricola = “281283” nome = “Luca” cognome = “Rossi”	[0].id = “10001” [0].corso.id = “20001” [0].corso.nome = “Basi di Dati” [0].data = “12/06/2011” [0].voto = 28 [1].id = “10001” [1].corso.id = “20004” [1].corso.nome = “Crittografia” [1].data = “21/05/2011” [1].voto = 30





redis

mongoDB

HBASE

amazon web services™

We have theses!