Coding for Data Science and Data Management
Module of Data Management

Relational - NoSQL Databases

Stefano Montanelli
Department of Computer Science
Università degli Studi di Milano
stefano.montanelli@unimi.it

Yesterday vs. Today Needs

- The needs around data storage and management strongly changed over time
- In the 1960s and 1970s
  - We had limited and expensive computing/network resources
  - The data volume to manage was limited
  - The rigid structure of relational databases represented a plus for the design of effective solutions of data organization
  - Constraints were perceived as a support to avoid data inconsistencies

Yesterday vs. Today Needs

- Today, in certain application contexts, the rigid data organization of relational databases is perceived as an obstacle rather than a benefit (e.g., web data analysis)
  - Personal/portable computing devices are widespread with pervasive networking support
  - The data volume is *un*limited
  - The availability of flexible data models represents a plus for effective data organization and storage
  - Constraints are more than necessary in many situations (e.g., FK on a tweet insert)

“One size fits all”

- *The last 25 years of commercial DBMS development can be summed up in a single phrase: “One size fits all”*
- *This phrase refers to the fact that the traditional DBMS architecture (originally designed and optimized for business data processing) has been used to support many data-centric applications with widely varying characteristics and requirements*

“One size fits all”

- *We argue that this concept is no longer applicable to the database market, and that the commercial world will fracture into a collection of independent database engines, some of which may be unified by a common front-end parser*

- Stonebraker, M. and Cetintemel, U., *One Size Fits All: an Idea whose Time has Come and Gone*, in Proc. of ICDE, 2005

Origin of NoSQL databases

- The term NoSQL appears the first time in 1998 by Carlo Strozzi to name his OpenSource, LightWeight database without any SQL interface
- According to a widely-accepted definition, NoSQL means **Not Only SQL**

Origin of NoSQL databases

- In the recent years, NoSQL solutions gained popularity and they have been adopted to address data management requirements in the framework of a number of applications, especially in the Web 2.0 context:
  - Amazon (DynamoDB)
  - Facebook (Cassandra)
  - Google (BigTable)

Definition of NoSQL database

- In the early 2009, Eric Evans, a Rackspace employee, reused this term to refer DBs that are:
  - non-relational
  - distributed
  - not conforming to ACID properties

Definition of NoSQL database

- Alternative definition:

  «NoSQL are next generation databases mostly addressing some of the points: being non-relational, distributed, open source and horizontally scalable»

  http://nosql-database.org

Main features of NoSQL DBs

- **Horizontal scalability**
  NoSQL DBs are designed to scale horizontally, which means that the system can be expanded while it is operational, by adding more nodes for storage and processing as data volume increases

Main features of NoSQL DBs

- **Availability, Replication**
  NoSQL DBs are designed to enforce availability through data replication in a transparent way
  - Read performance is improved
  - Write performance is more complex since an update must be applied to all the copies

Main features of NoSQL DBs

- **Eventual consistency**
  NoSQL DBs usually improve write performance through the use of a more “relaxed” form of consistency called eventual consistency (a.k.a. optimistic replication)

Strict vs. eventual consistency

- Example of strict consistency
  - Your bank balance is € 50
  - You deposit € 100
  - Your bank balance, queried from any ATM anywhere, is € 150
  - Your daughter withdraws € 40 with your ATM card
  - Your bank balance, queried from any ATM anywhere, is € 110

Strict vs. eventual consistency

- Example of eventual consistency
  - I watch the weather report and learn that it's going to rain (the report is a reliable source of information)
  - I tell you that it's going to rain
  - Your neighbor tells his wife that it's going to be sunny
  - You tell your neighbor that it is going to rain
  - Your neighbor tells his wife that it's going to rain
- Eventually, all the people know the truth, but in the meantime the wife came away with an outdated information

Main features of NoSQL DBs

- **Horizontal fragmentation**
- NoSQL data files can have many millions of records since file records are horizontally partitioned (*data sharding*) to distribute the data access load
- Suitable combination of data sharding and shard replication improve load balancing and system availability

Main features of NoSQL DBs

- **Replication models**
- **Master-slave**: one copy is the master copy and all writes must be applied to the master copy and then propagated to the slaves (using eventual consistency techniques)
- The reads can be executed only on the master copy or can be allowed also at the slaves (but no guarantee to read latest updated values)

Main features of NoSQL DBs

- **Replication models**
- **Master-master**: all reads and writes can be at any of the copies, then no guarantee that reads at different nodes with replicated data see the same value and different concurrent writes on the same data item at different nodes can generate temporary inconsistent values of the item
- A reconciliation method is implemented to resolve conflcting writes of the same data item

Main features of NoSQL DBs

- **Schema is not required**
- NoSQL DBs allow self-describing, semistructured (e.g., JSON) data. There could be a partial schema, BUT the schema is NOT required
- Data constraints must be programmed at the application level, into the program code

Main features of NoSQL DBs

- **Less powerful query languages**
- Applications using NoSQL often require to locate (single) records in single files based on key values and not via complex attribute conditions
- An appropriate set of operations are provided in the form of API by NoSQL systems called **CRUD** (create, read, update, delete) for reading/writing records

Classification of NoSQL database

- We can distinguish NoSQL databases in the following main categories:
  - *Key-Value stores*

    (e.g., DynamoDB, Cassandra, Berkeley DB Redis)
  - *Document-based systems*

    (e.g., MongoDB, CouchDB)
  - *Column-family systems*

    (e.g., Google BigTable, Amazon's SimpleDB)
  - *Graph-oriented systems*

    (e.g., Neo4j, Sones, InfinityGraph)

Key-Value stores

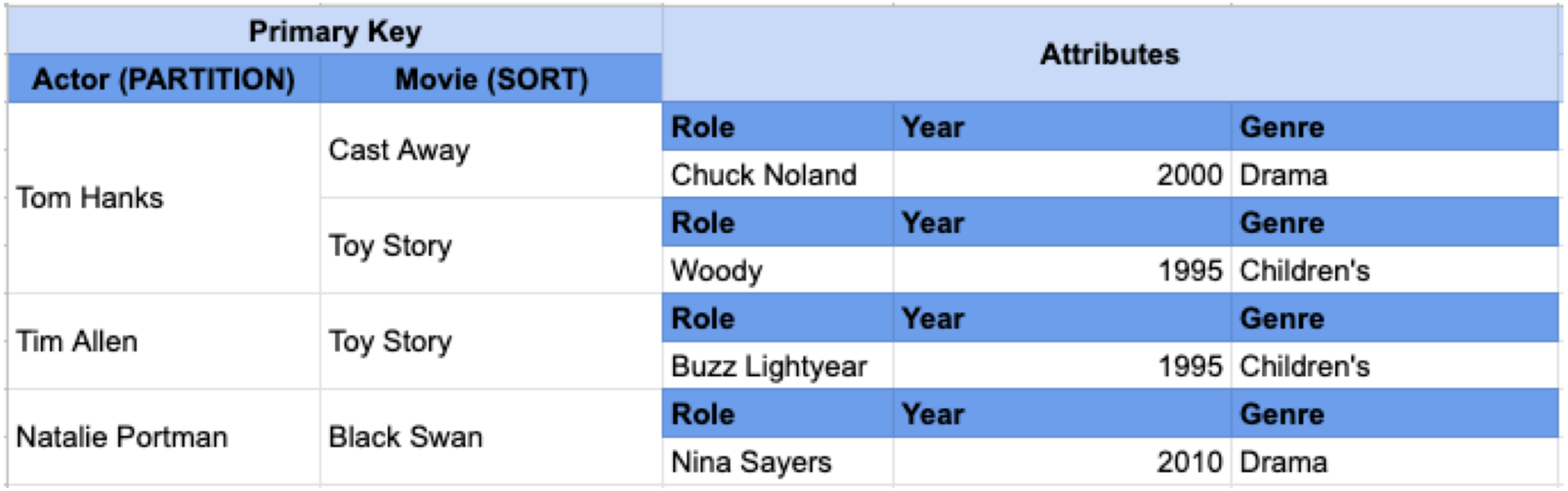
- The target is high-performance, availability, scalability
- Data are modeled as pairs <*key-value*>, where *key* is a unique identifier and *value* is a data item that can be rapidly located by the key (hash function)
- A key-value system defines the key space and it allocates the storage space associated with a given key, by managing the subsequent data retrieval

Key-Value stores

- The system does not care of the type and format of the values which can be highy heterogeneous (e.g., string of bytes, array of bytes)
- The interpretation of the data structure is demanded to the application
- Supported operations (key(k) - value (v)):
  - PUT k,v: writes value v in the space identified by k
  - GET k: retrieves the value in the space identified by k
  - DELETE k: deletes the value in the space identified by k

Key-Value stores

- Example with DynamoDB



Primary Key		Attributes		
Actor (PARTITION)	Movie (SORT)			
Tom Hanks	Cast Away	Role	Year	Genre
		Chuck Noland	2000	Drama
	Toy Story	Role	Year	Genre
		Woody	1995	Children's
Tim Allen	Toy Story	Role	Year	Genre
		Buzz Lightyear	1995	Children's
Natalie Portman	Black Swan	Role	Year	Genre
		Nina Sayers	2010	Drama

Document-based systems

- Document-based systems are based on the notion of collection and document

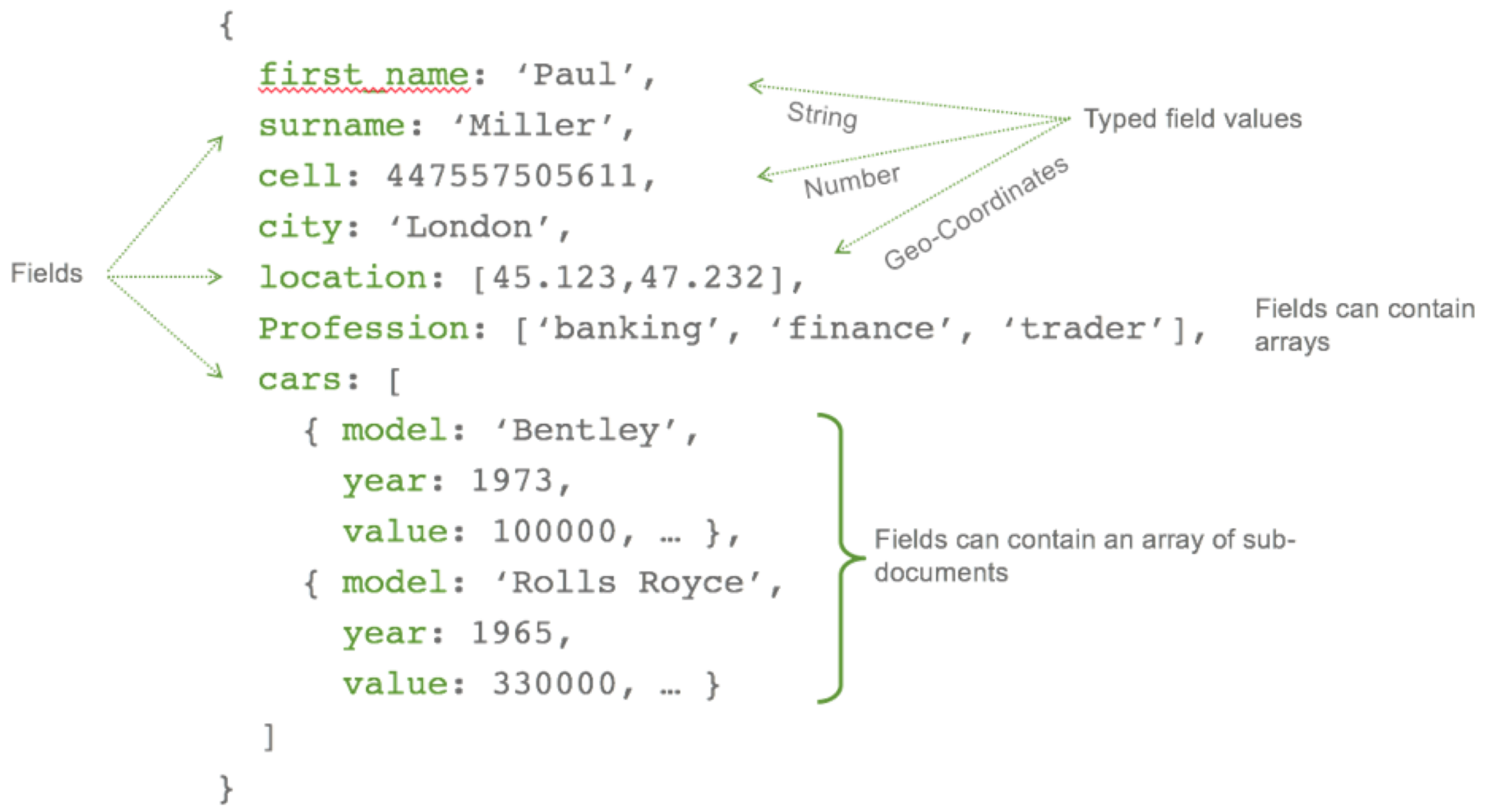
- A **document** is a data item with a flexible notion of structure
  - A document is self-describing both schema and data
  - Documents have a hierarchical structure organization of inter-related data elements
  - Documents can be specified in various formats (e.g., JSON - JavaScript Object Notation documents)

Document-based systems

- Document-based systems are based on the notion of collection and document
- A **collection** is a *schema-free* group of similar documents
  - A collection can group documents with similar, but different structure
  - Two documents in a collection can have different data elements

Document-based systems

- Example



```
{
  first_name: 'Paul',
  surname: 'Miller',
  cell: 447557505611,
  city: 'London',
  location: [45.123,47.232],
  Profession: ['banking', 'finance', 'trader'],
  cars: [
    { model: 'Bentley',
      year: 1973,
      value: 100000, … },
    { model: 'Rolls Royce',
      year: 1965,
      value: 330000, … }
  ]
}
```

Fields
String
Number
Geo-Coordinates
Typed field values
Fields can contain arrays
Fields can contain an array of sub-documents

Column-family systems

- Data are stored in tables
- The rows in a table are self-describing, with a unique row-key
- A table is associated with one or more colums, having a name, which are specified at the table creation time
- Each column can be associated with many qualifiers, not necessarily specified at the creation time

Column-family systems

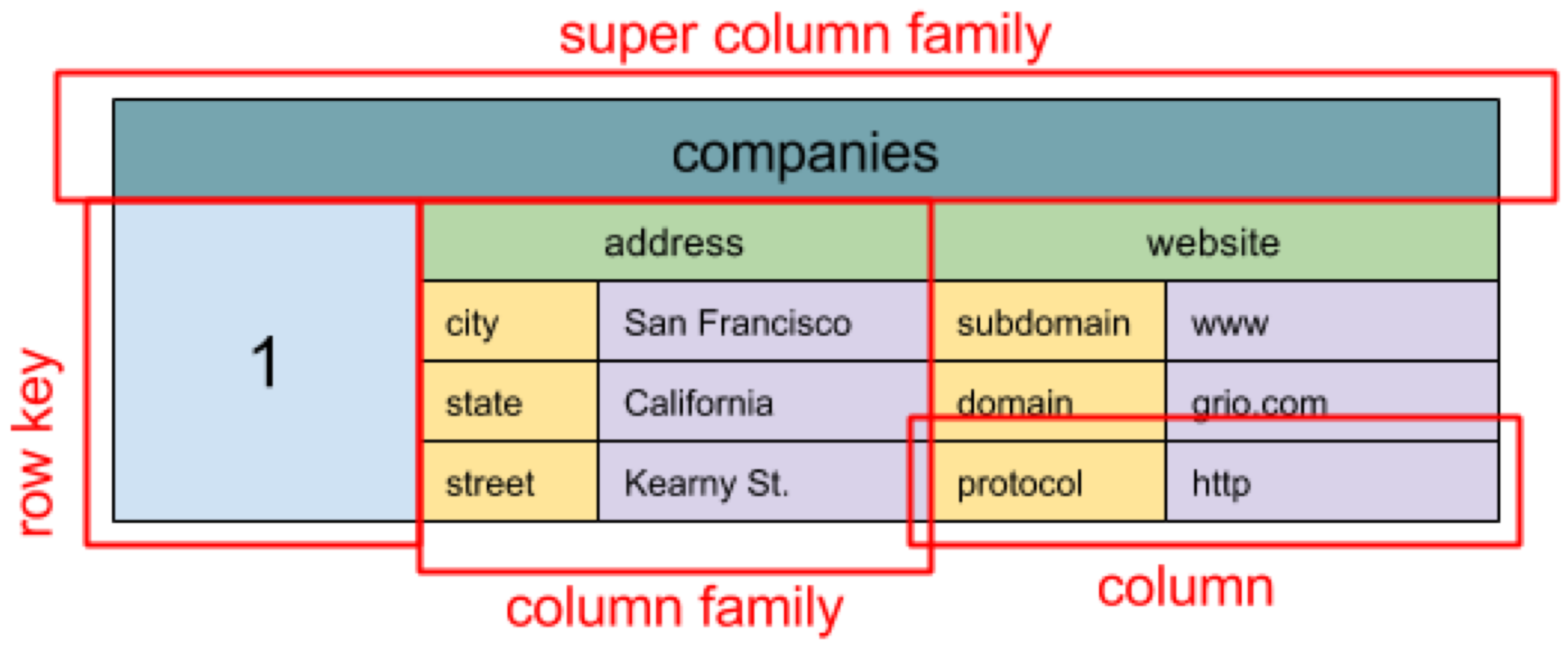
- Column qualifiers make the model self-describing since they are dynamically specified when new rows are inserted in a table
- Different rows have the same columns but can have different qualifiers for the same column

Column-family systems

- Column families group together several columns (i.e., attributes of relations) that are related and should be retrieved together
- This kind of systems are suitable to speed-up the scan operations of large datasets by organizing in the same table-record the join result of a conventional relational DB

Column-family systems

- Example



Graph-oriented systems

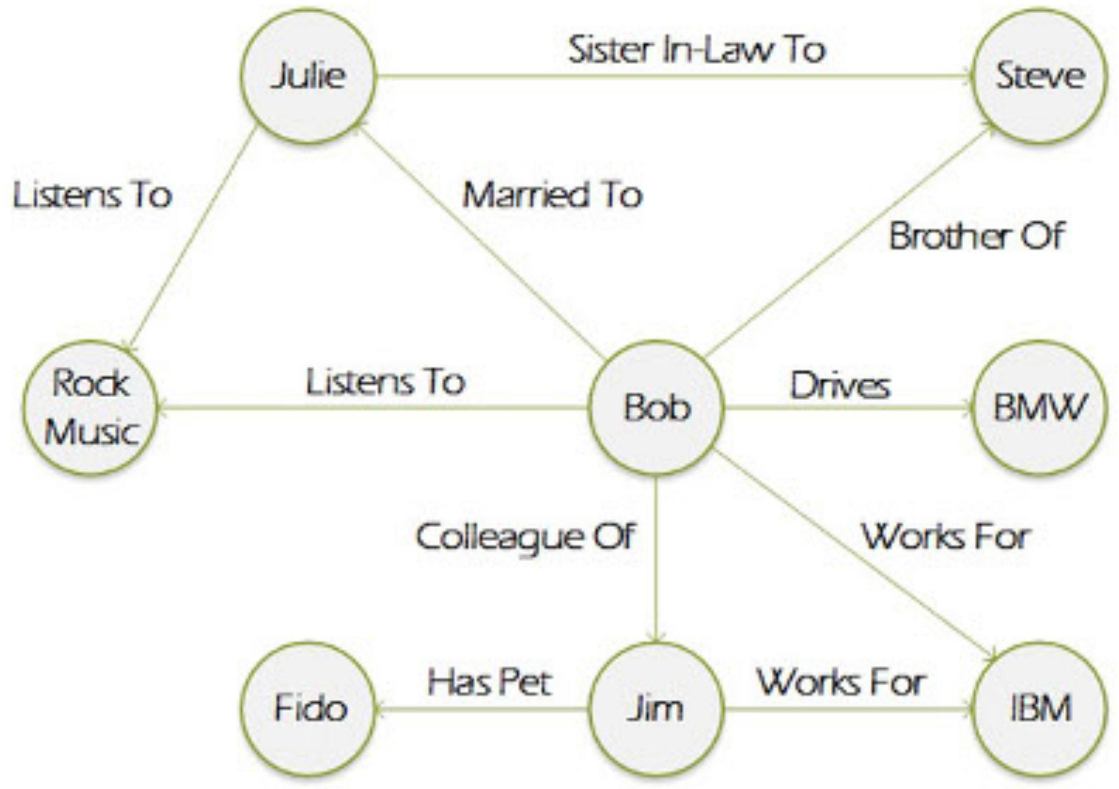
- Data are organized as graphs, i.e., collections of labeled nodes and edges
- Nodes represent data entities and they can have arbitrary properties with a unique key identificator
- Nodes can have one or more labels. Nodes with the same label are grouped into collections for query purposes

Graph-oriented systems

- Edges represent direct relationships between nodes and can also have arbitrary properties
- Relationships have a type whose role is analogous to the node labels
- Relations are used to reach a node from another node (i.e., graph traversal is similar to the join operation)

Graph-oriented systems

- Example



Coding for Data Science and Data Management
Module of Data Management

NoSQL for Big Data

Stefano Montanelli
Department of Computer Science
Università degli Studi di Milano
stefano.montanelli@unimi.it

NoSQL and Big Data

- NoSQL are generally considered as the natural data storage solution for supporting big-data applications
- This is mainly due to the NoSQL support to horizontal scalability that allows to expand the storage capabilities by adding more nodes when data volume increases

Definitions of Big Data

- “any voluminous amount of structured, semistructured and unstructured data that has the potential to be mined for information”
- “data sets that are so large or complex that traditional data processing application software is inadequate to deal with them”
- “massive amounts of data collected over time that are difficult to analyze and handle using common database management tools”

The Vs of Big Data

- **Volume**: large data volumes - scale;
- **Variety**: many different data types and sources to consider, both structured and unstructured (e.g., relational data, documents, web data, emails, graphs, images, videos);
- **Velocity**: speed at which data is created, gathered, captured, processed, and eventually set aside;

The Vs of Big Data

- **Veracity**: quality degree and credibility of the data to be analyzed; usually data are inconsistent, incomplete, inaccurate
- **Value**: data relevance to customers, business value, actionable knowledge for enforcing statistical analysis, data mining, machine learning, visualization

Examples of Big Data applications

- E-commerce data of a commercial company
- Browsing history and web actions of a user
- Pollution data collected within a metropolitan area through sensing devices
- Car traffic data within a phone cell
- Tweet data about a certain hashtag