UNIVERSITÀ DEGLI STUDI DI MILANO

Introduzione ai db NoSQL

LA STATALE

Prof. Stefano Montanelli

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 collection)
- Personal/portable computing devices are widespread with pervasive networking support
- The data volume is unlimited
- 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*
- *We argue that this concept is no longer applicable to the database market, and that the commercial world will fracture into a collection of independent database engines, some of which may be unified by a common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**
- 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

Origin of NoSQL databases

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

Availability, Replication

- NoSQL DBs are designed to enforce availability through data replication in a transparent way: i) improved read performance; ii) complex write performance (since an update must be applied to all the copies)

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)

Eventual consistency - example

- User A watches the weather report and learns that it’s going to rain (the report is a reliable source of information)
- User A tells User B that the weather is going to rain
- User C tells User D that the weather is going to be sunny
- User B tells User C that the weather is going to rain
- User C tells User D that the weather is going to rain

Eventually, all the people know the truth, but in the meantime User D 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

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

Classification of NoSQL databases

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)

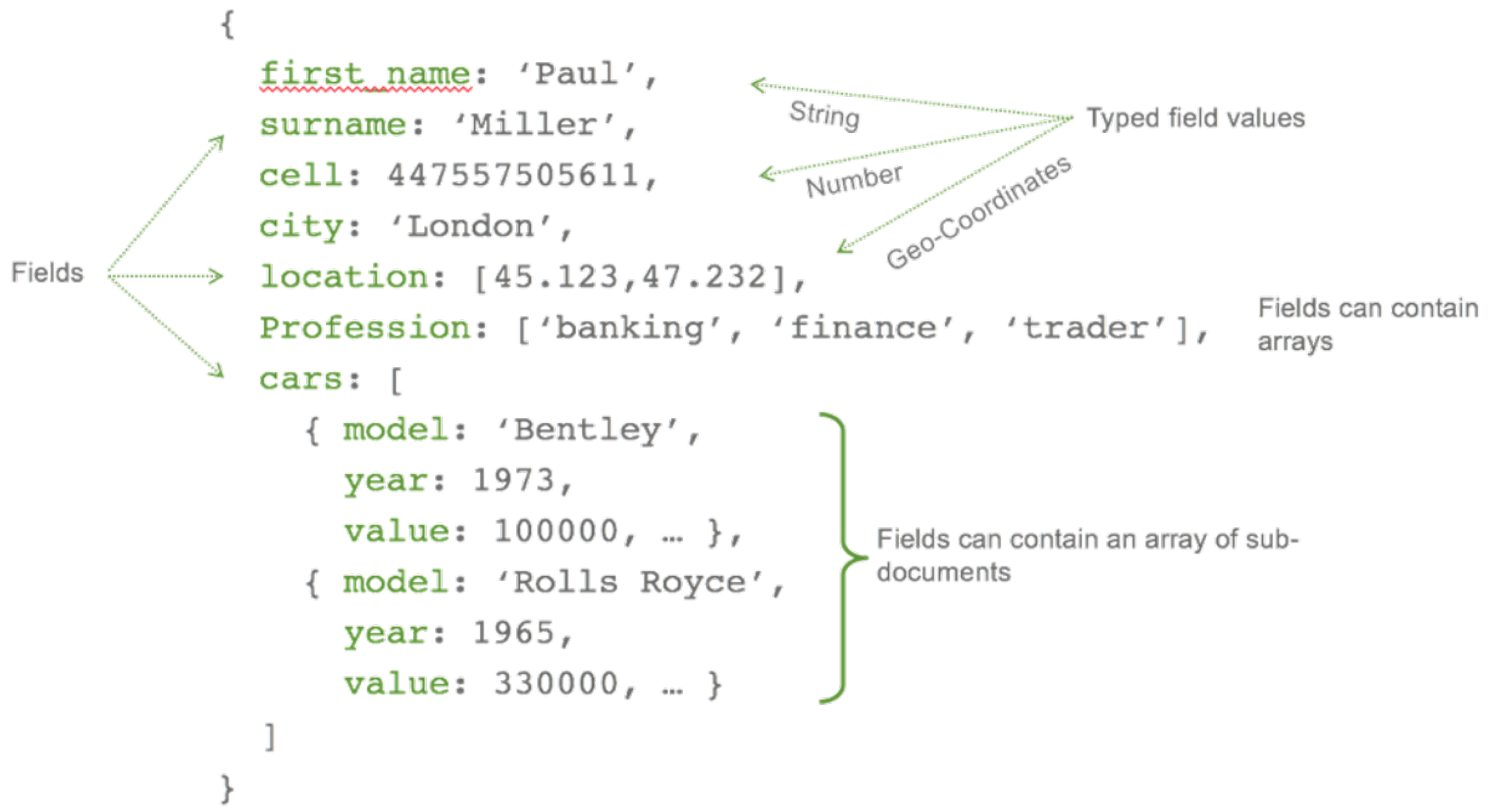
Document-based systems

- 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



```
{
  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