Politecnico
di Milano

NoSQL databases

Elisabetta Di Nitto
elisabetta.dinitto@polimi.it
13/01/2015

Lecture for the course:
A Multidisciplinary Perspective On Big Data

Credits to Danilo Ardagna, Oscar Locatelli,
Santo Lombardo, Marco Scavuzzo

Politecnico
di Milano

Databases: basic concepts of the relational model

DBMS

- A DBMS does the following
  - Allows an efficient access to data, with a granularity level lower than the one offered by a file system
  - Allows a direct access to data based on their properties
  - Allows concurrency control at the level of the single piece of data (record)
  - Offers languages to describe data and queries on data in a way that is independent on the way they are physically stored
  - Offers sophisticated mechanisms for
    - Reliability
    - Privacy
    - Atomicity of transactions

RDBMS: assumptions and benefits

- Assumptions
  - Well-defined structure for data
  - Data is dense and uniform
  - Indexes can be defined a priori and used for queries
  - Data stays within a few Gigabyte
- Benefits
  - It uses the lowest amount of disk space
  - It is a well-understood model and query language
  - It can support a wide variety of use cases
  - It has schema-enforced data consistency

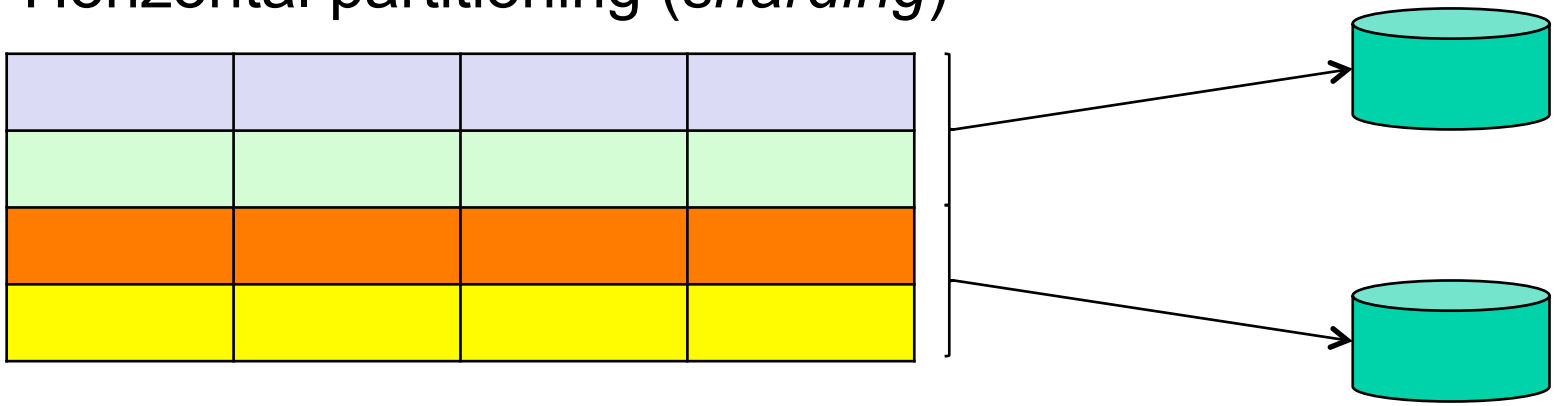
RDBMS: disadvantages and new requirements

- Disadvantages
  - Relatively slow performance
  - Schemas mean a higher programmer overhead for iterating changes
  - High degree of complexity with many tuning knobs
- New requirements
  - In 2009 Google was processing 24 Petabyte per day
  - In 2009 Facebook declared to store about 60 millions of images
  - The Internet archive stores 2 Petabyte of data

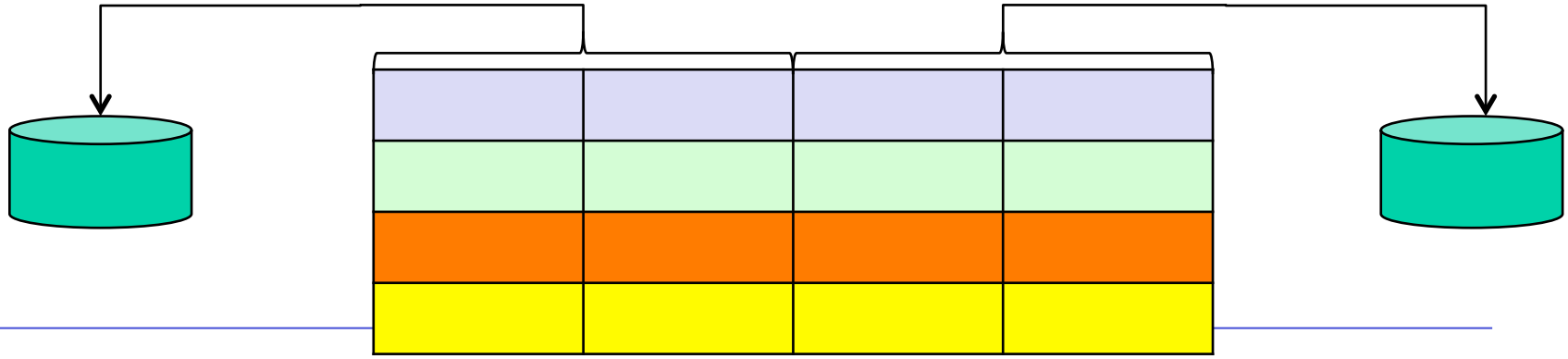
NoSQL: basic concepts

Data partitioning

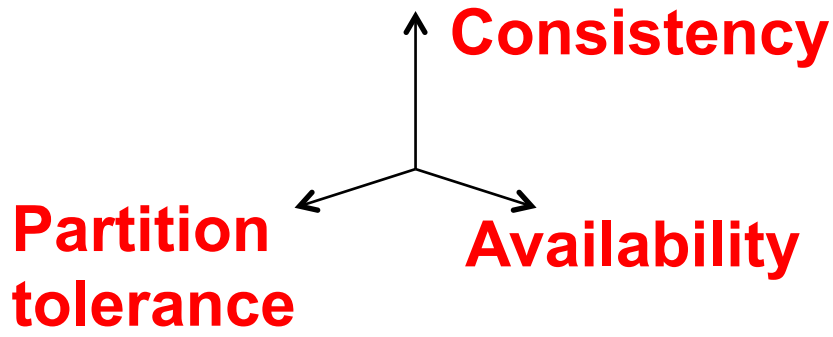
- In case of big data set: partition data storing them on different servers
- Horizontal partitioning (*sharding*)



- Vertical partitioning

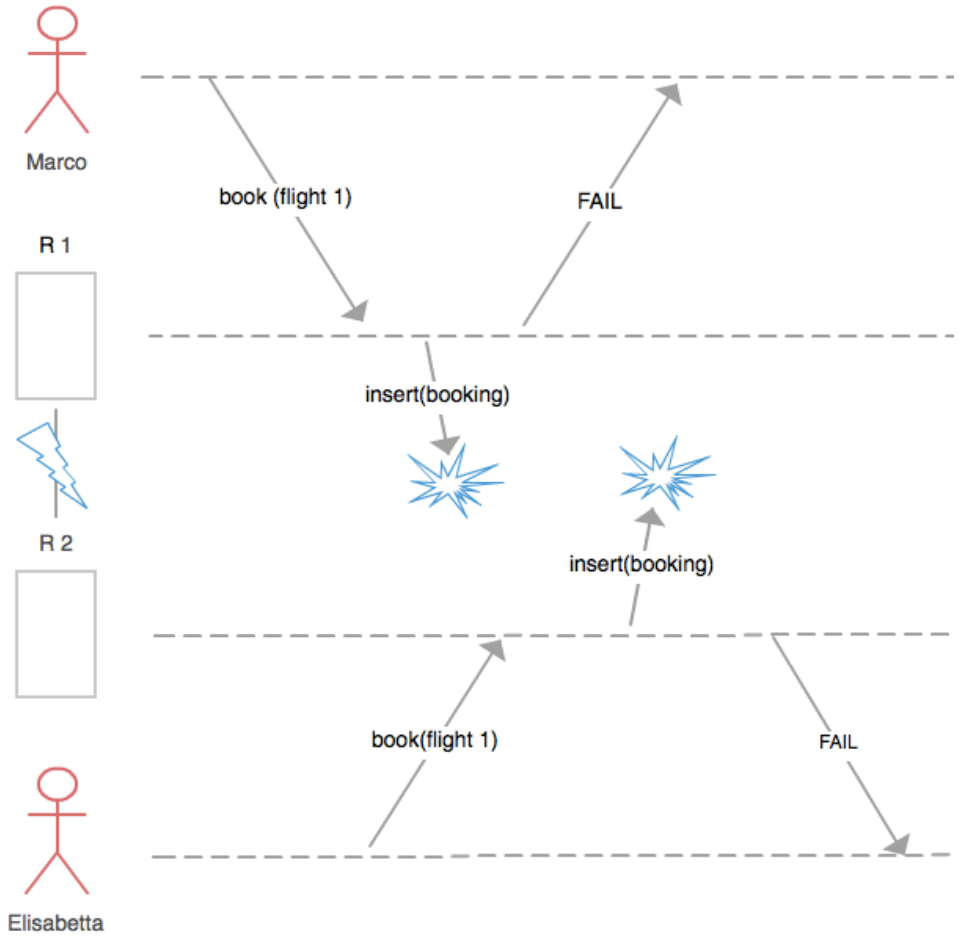


DBMS and CAP theorem



- Three possible guarantees
  - *Consistency*: all nodes see the same data at the same time
  - *Availability*: every request receives a response about whether it was successful or failed
  - *Partition tolerance*: the system continues to operate despite arbitrary message loss or failure of part of the system
- **CAP theorem**: it is impossible for a distributed computer system to simultaneously provide all three guarantees

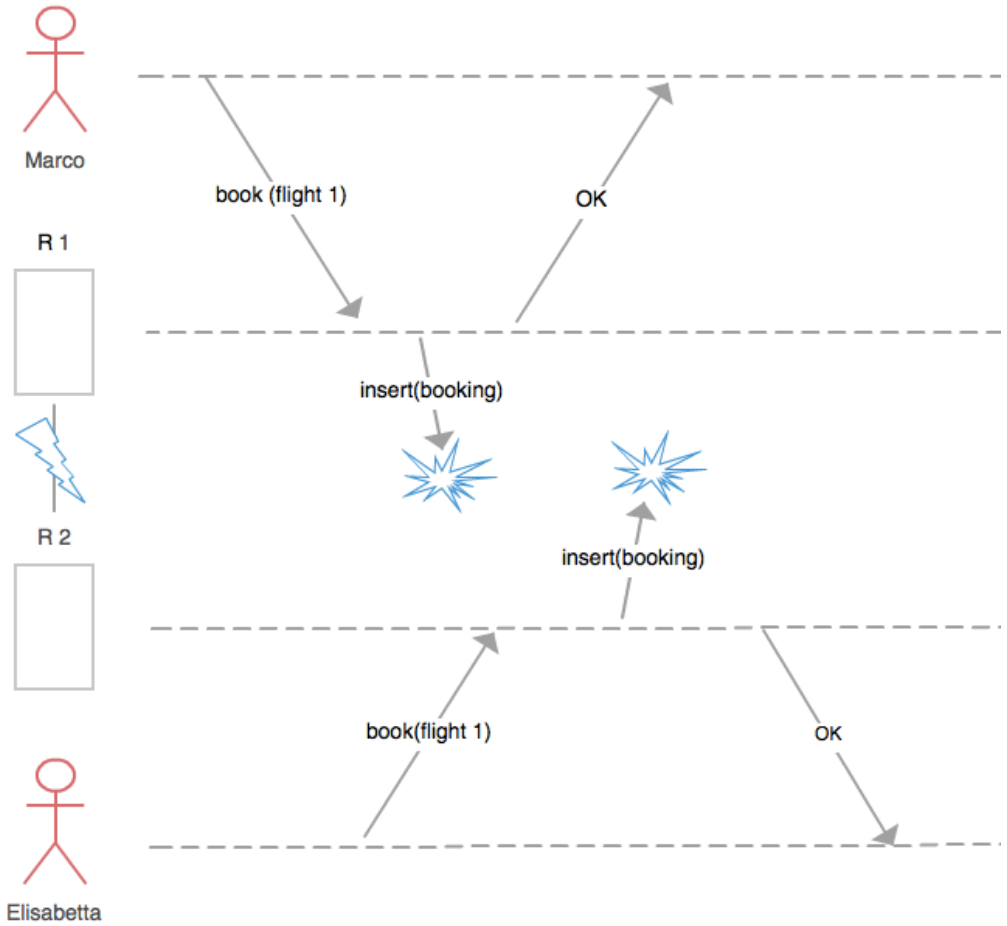
CP (Consistency & Partition tolerance) = CAP-Consistency system example



Non ACID Consistency Management

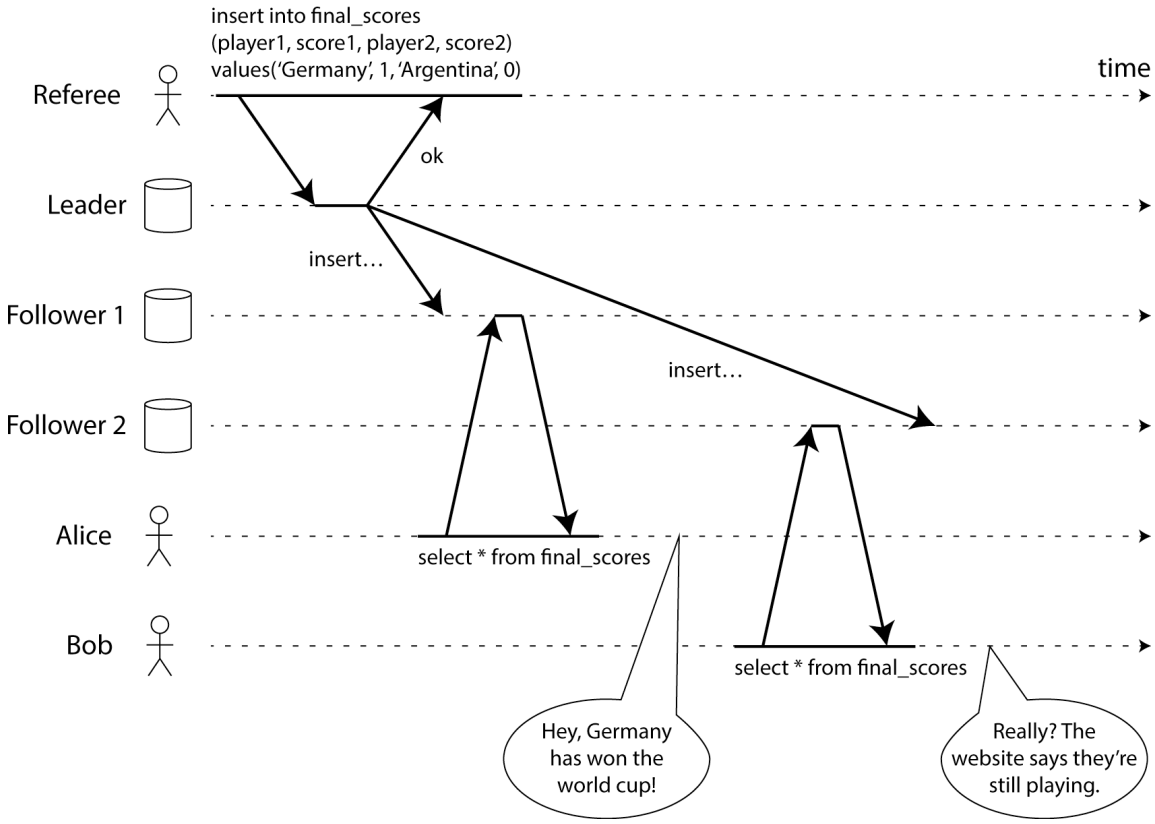
- Eventual Consistency:
  - Changes made at one replica will be transmitted asynchronously to the others (e.g., DNS)
  - Discrepancies in data state between replicas, and thus between users and locations, for a temporary period may occur

AP (Availability & Partition tolerance) = CAP-Availability system example



Tradeoff Consistency/Latency

- Example of a database executing non linearizable write ops. (without partitions)



Bob gets stale data

Bob realizes it through a separate communication channel (audio)

DBMS and CAP theorem

- Relational databases favor Consistency and Availability
- NoSQL databases favor Availability and Partition tolerance
  - Don’t use them when you need consistency guarantees!

What is NoSQL?

- No use of SQL as query language:
  - Manage large volumes of data that do not necessarily follow a fixed schema
  - Data is partitioned among different machines and JOIN operations are not usable
- ACID guarantees may be relaxed:
  - E.g., eventual consistency
  - Transactions limited to single data items
- Distributed, fault-tolerant architecture
  - Data held in a redundant manner on several servers
  - Horizontal scalability

Categories of NoSQL

Main categories of NoSQL based on their logical data model

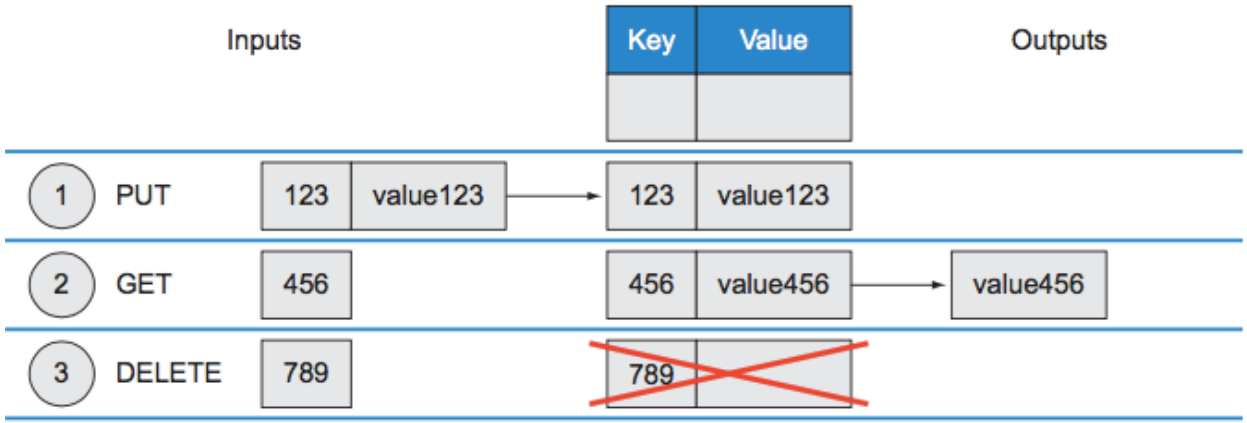
- Key-Value
- Column-family
- Document-based

Key Value NoSQL Databases

- Common data model: map/dictionary
- Putting and requesting values (BLOBs) is done per key
- Keys are sorted lexicographically
- Favor high scalability over consistency
- No ad-hoc querying or analytics features
- Length of keys is limited, length of values not limited

Key Value query language

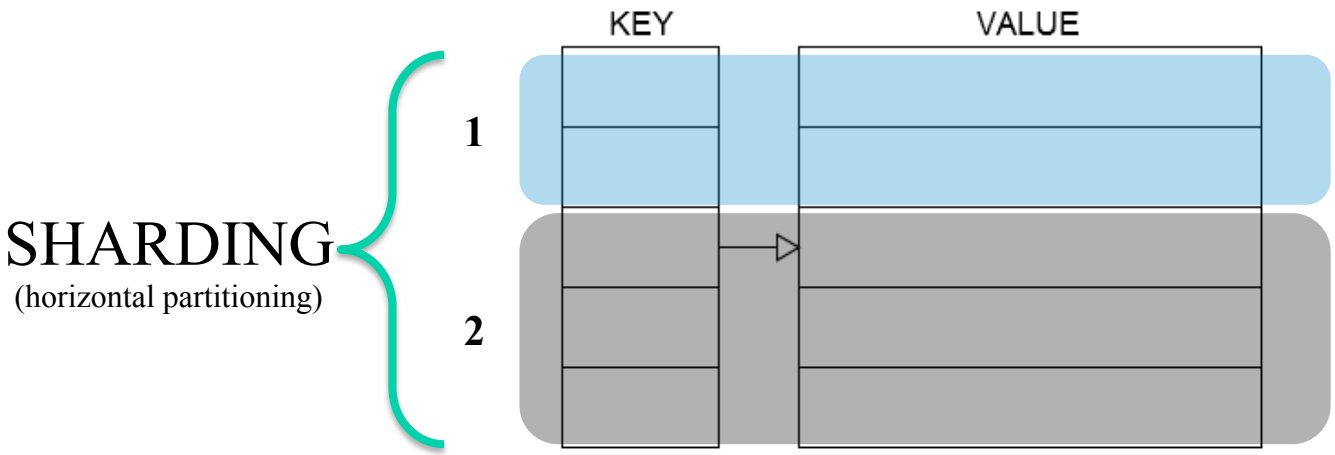
- Query operations are limited to
  - put(key,value)
  - get(key)
  - delete(key)



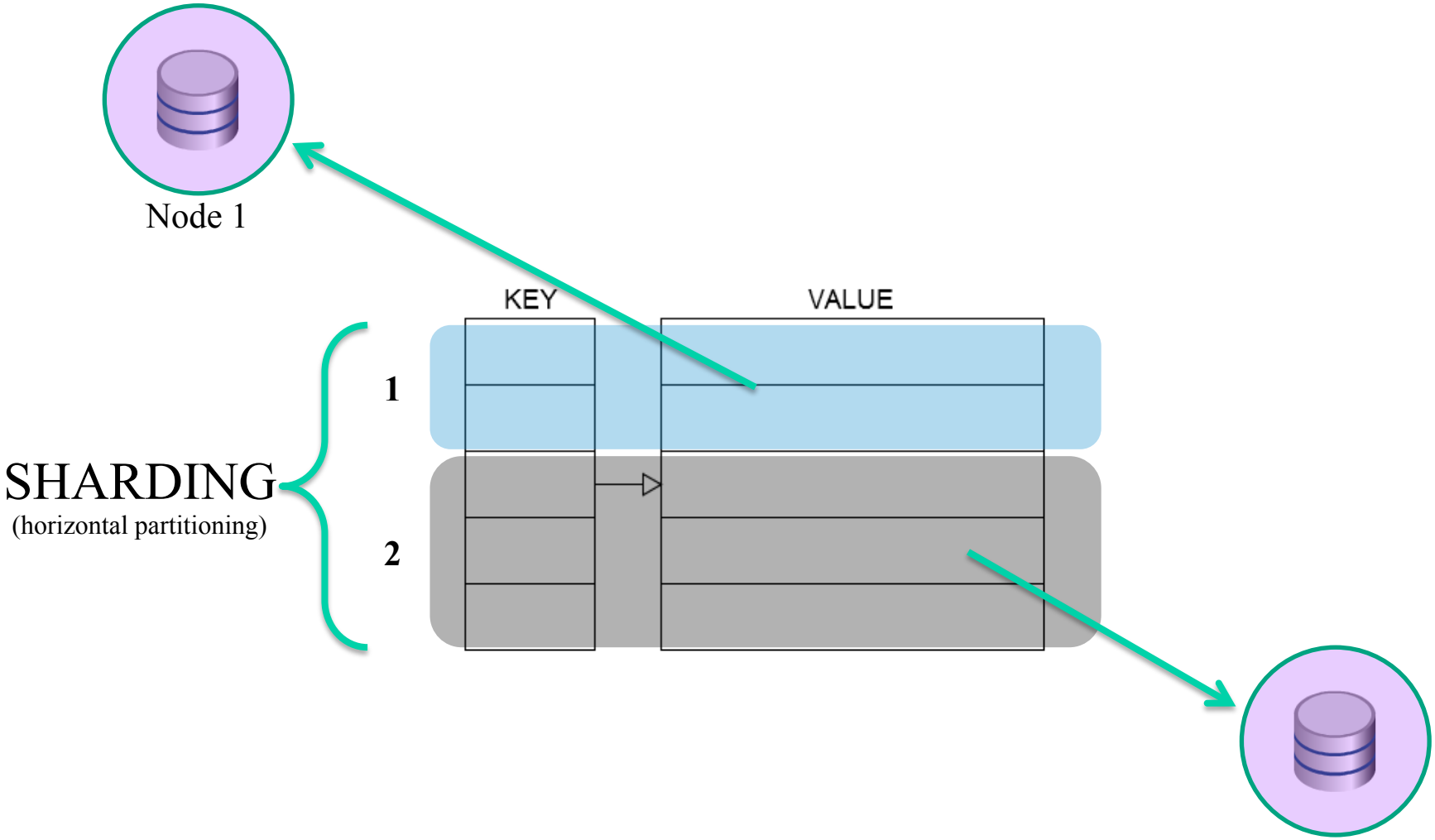
- Redis also provides matching for key-ranges
  - Matching of numeric ranges
  - Matching of regular expressions

Key-Value and sharding

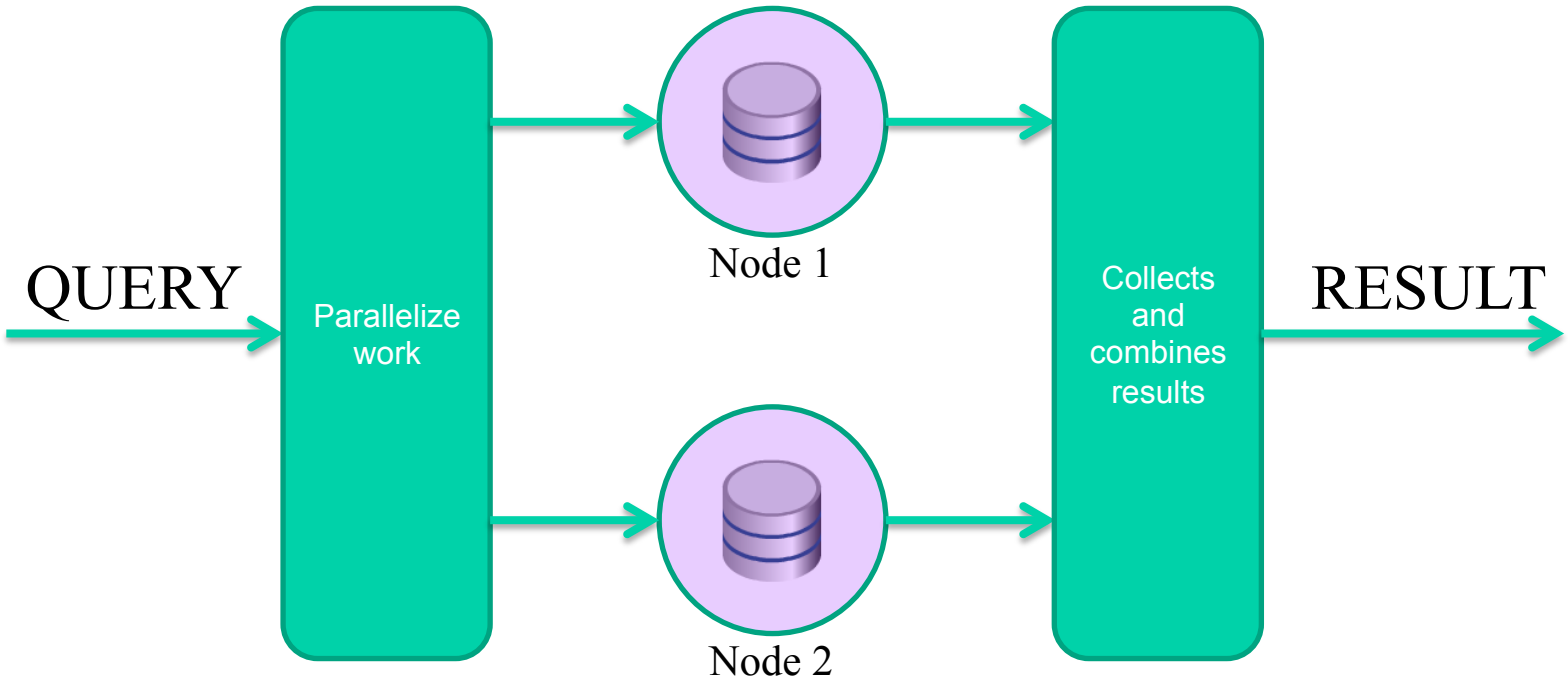
- **Shared Nothing** architecture: each key-value pair is decoupled from the others
- Can be easily moved on different machines



Key-Value and sharding

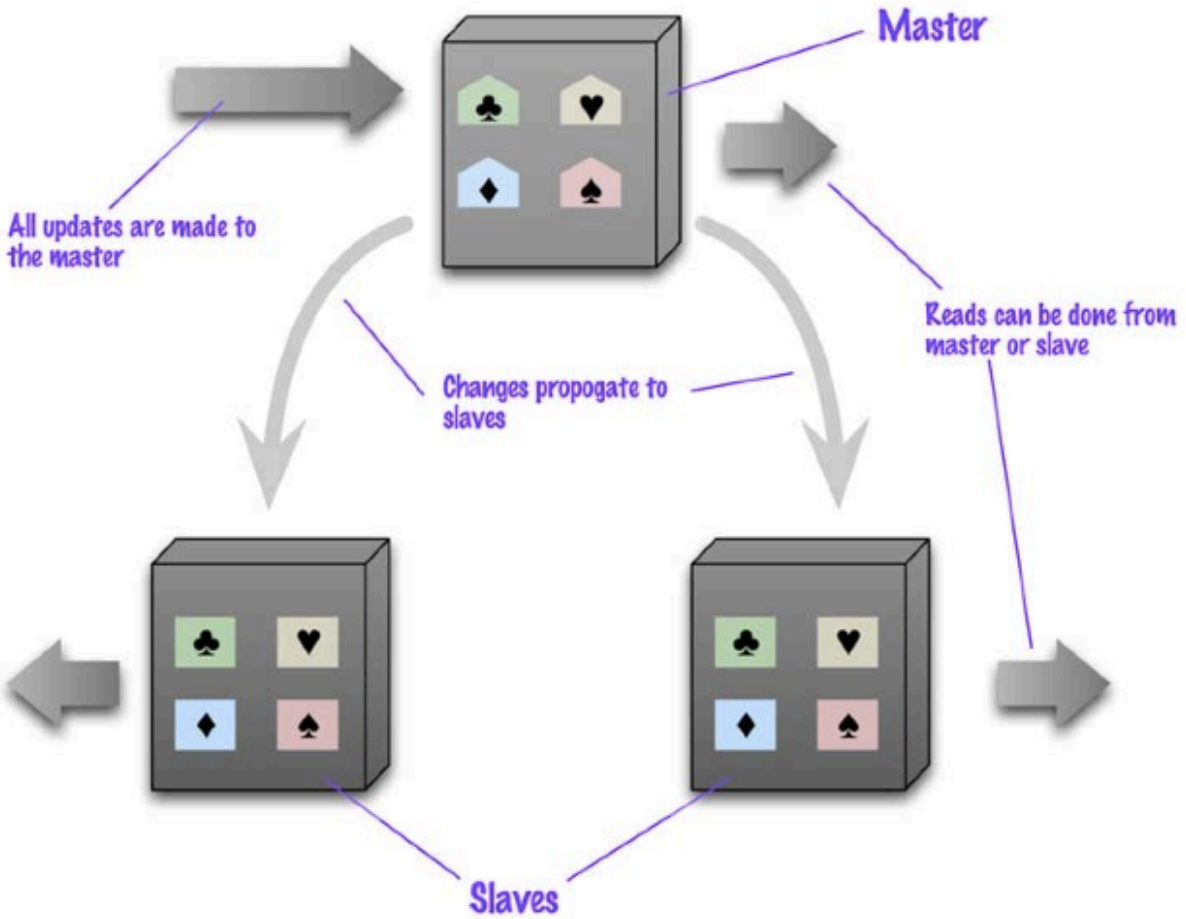


Approach to query execution



PARALLEL EXECUTION

Active-Passive Replication



Key Value storage typical usages

- Session storage
- User Profiles or preferences
- Shopping carts
- Single user analytics

Evolution of key-value