Welcome!

NOSQL CONCEPTS

Miriam Antona
Software engineer

datacamp

Topics covered

- **Chapter 1**: NoSQL vs relational databases / key-value databases
- **Chapter 2**: Document databases
- **Chapter 3**: Column family databases
- **Chapter 4**: Graph databases

About the course

- Conceptual course (no coding required)

NoSQL vs relational databases

Relational databases

- Use tables/rows/columns
- Need a predefined schema/complicated to change
- Slow queries when joining multiple tables
- Vertically scalable
  - scale by adding more power (e.g. CPU, RAM...)
  - more expensive
- Guarantee ACID transactions
- Typically closed source

NoSQL

- Originally non-SQL/non-relational
- Not only SQL
- Non-relational databases
- Don't use tables/rows/columns
- Schema-less/easy changes
- Fast queries
- Horizontal scalable/cheaper
- Most don't support ACID transactions
- Open source

NoSQL vs relational databases

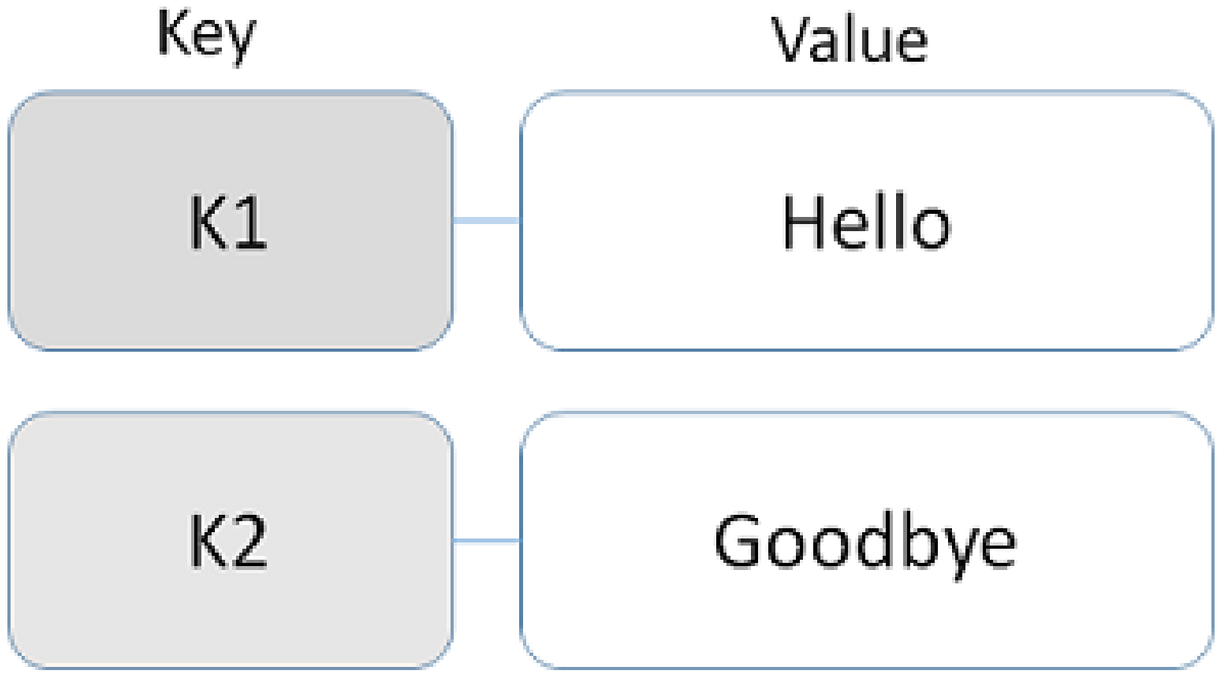
- Are complementary
- Can coexist

Types of NoSQL databases

- Key-value databases
- Document databases
- Column family databases
- Graph databases

Key-value databases

- Simplest NoSQL databases
- Get/Set values with associated key



Key

- Examples:
  - DataCampCourse:123:name
  - Client456
  - 123456789
  - **nosql@courses.me**
  - C09113276F59B26EF3394D90CD31BAA9C
- Any binary sequence
- Unique
- Can be generated by algorithms
- No long keys

Value

- Associated with a key
- Retrieve, set, delete a value by key
- Numbers, strings, JSON, images...
- Size restrictions

Value

- Associated with a key
- Retrieve, set, delete a value by key
- Numbers, strings, JSON, images...
- Size restrictions

key	value
dataCampCourses:123:name	Cleaning data in a SQL Server database
dataCampCourses:123:softLaunchDate	10/01/2020
user:12:address	('123 Sesame Street', 'NY')
user:125:address	{"street" : "123 Sesame Street" , "city" : "NY"}

Datazy example

User preferences

key	value
user:457:preferences	{**"language"** : "en_US" , **"color"** : "green" , **"timezone"** : "GTM-4"}
user:458:preferences	{**"language"** : "es_US" , **"color"** : "blue" , **"timezone"** : "GTM+2"}

- Convention (:)
  - user:id:preferences

Popular key-value databases

redis

DynamoDB

riak

Project Voldemort

A distributed database.

Let's practice!

NOSQL CONCEPTS

Advantages and limitations of key-value databases

NOSQL CONCEPTS

Miriam Antona
Software engineer

Advantages - very simple

- Key-value tuple
- No defined schema/types
- Basic operations:
  - Put
    - inserts a new key-value tuple
    - updates a value if the key already exists
  - Get
    - returns the value by a given key
  - Delete
    - removes a key and its value
- Fast operations

Advantages - flexible

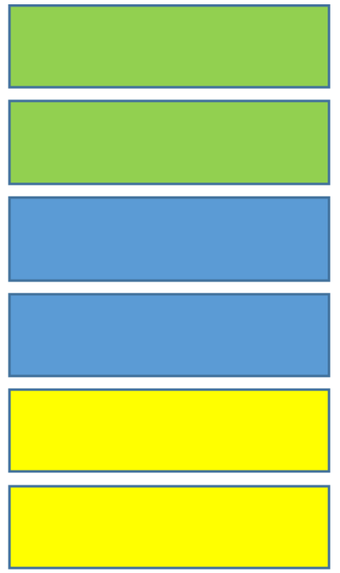
- Allow changes in data types
  - `userID:123 = 123456`
  - `userID:123 = "Miriam"`
- Add additional attributes
  - `user:457:preferences = {"language" : "en:US"}`
  - `user:457:preferences = {"language" : "en:US", "color" : "green","timezone" :"GTM-4"}`

Advantages - information stored in memory

- Fast reads/writes
- Can lose data
- Combination of disk and memory persistence

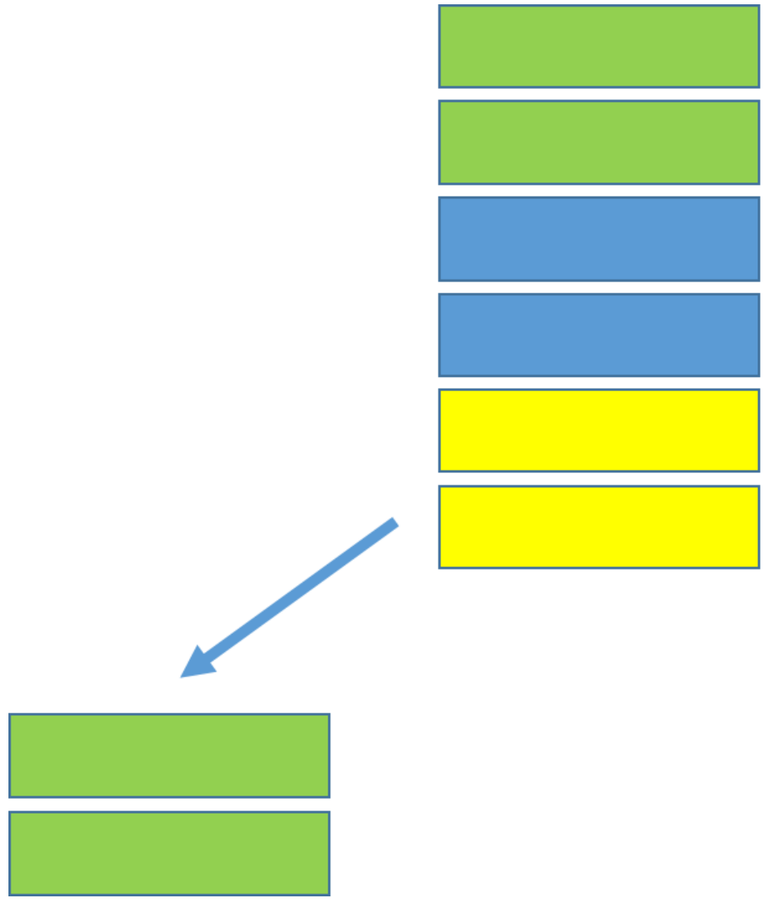
Advantages - scalability

- Can scale horizontally
- Sharding
  - distributes different parts of the data across multiple servers



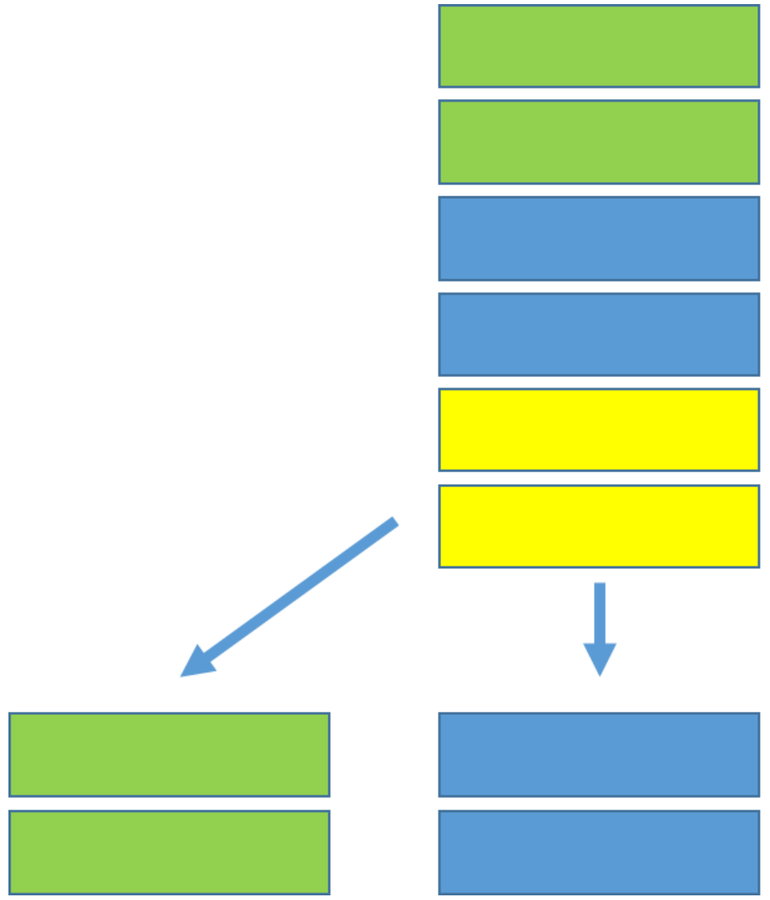
Advantages - scalability

- Can scale horizontally
- Sharding
  - distributes different parts of the data across multiple servers



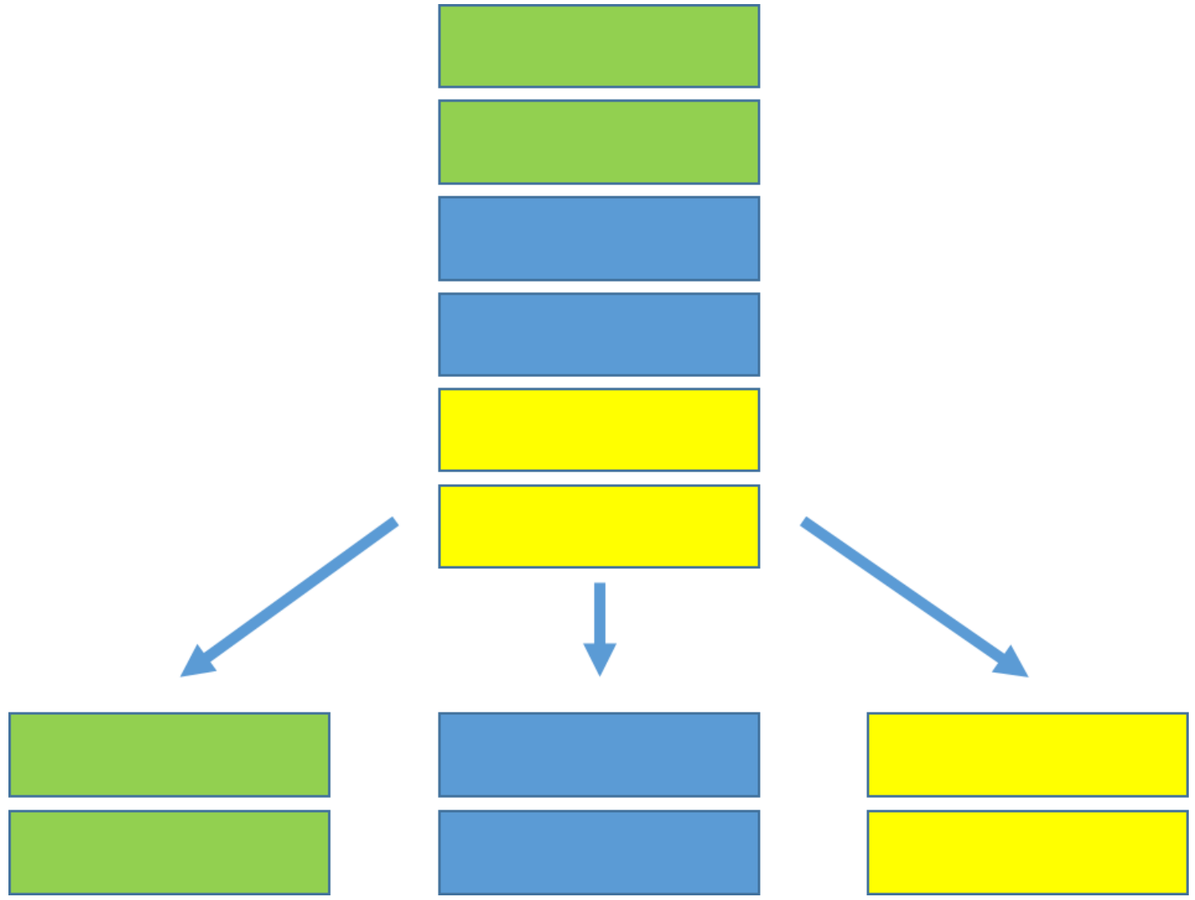
Advantages - scalability

- Can scale horizontally
- Sharding
  - distributes different parts of the data across multiple servers



Advantages - scalability

- Can scale horizontally
- Sharding
  - distributes different parts of the data across multiple servers



Limitations

- Just search by key
  - Problem if we don't know the key
  - Some key-value databases added functionalities
    - search by value
    - add secondary indexes
    - search by several keys simultaneously
  - Not complex queries

Let's practice!

NOSQL CONCEPTS

When to use key-value databases

NOSQL CONCEPTS

Miriam Antona
Software engineer

datacamp

Suitable cases

- User sessions
  - key: session ID
  - value: session information

Suitable cases

- User profiles and user preferences
  - key: user ID
  - value: user profile/preferences

- language
- color
- timezone

Suitable cases

- Shopping carts
  - key: user ID
  - value: shopping cart information

Suitable cases

- Real-time recommendations

Suitable cases

- Advertising

Suitable cases

- Store the information as the **value** in a single object
- Information is saved with one operation

```
SET user:457:preferences {"language":"en_US","color":"green","timezone":"GTM-4"}
```

- Information is retrieved with one operation

```
GET user:457:preferences
```

- Fast

Unsuitable cases

- Search data by its value

Unsuitable cases

- Search data by its value

key	value
user:1:address	{"**street**" : "123 Sesame Street" , "**city**" : "New York City"}
user:2:address	{"**street**" : "742 Evergreen Terrace" , "**city**" : "Springfield"}
user:3:address	{"**street**" : "221b Baker Street" , "**city**" : "London"}
user:4:address	{"**street**" : "4 Privet Drive" , "**city**" : "Little Whinging"}
...	...

- Related data

Let's practice!

NOSQL CONCEPTS

Redis case study

NOSQL CONCEPTS

Miriam Antona
Software engineer

Redis - overview

- ***Re**mote **Di**ctionary **S**erver*
- Popular **key-value database**
- **Fast in-memory data** structure store
  - In-memory dataset
  - Also allows to persist data to disk
- **Used as:**
  - Database
  - Cache
  - Message broker

- Open source
- **Redis Labs:** 400+ employees

Redis - data structures

- Strings

```
SET name Ann
```

- Lists

```
RPUSH my_numbers 1 2 3
```

- Sets

```
SADD my_set 1 2 3
```

- Hashes

```
HMSET user:123 name Ann surname Smith
```

- ...

Redis - data structures

- Strings

```
SET name Ann
```

- Lists

```
RPUSH my_numbers 1 2 3
```

- Sets

```
SADD my_set 1 2 3
```

- Hashes

```
HMSET user:123 name Ann surname Smith
```

- ...

[1] https://www.redis.io/commands

Redis - features

- Atomic **operations**
- **Transactions**
- **Lua scripting** for complex operations
- **Programming languages:** Python, R, C#, Java, JavaScript, PHP...
- **Asynchronous replication**

Redis - popular uses

- **Caching** (query results, images, files...)
- **Session storage** (user profiles, credentials...)
- **Chatting, messaging, and queues** (chat rooms, real-time comments, social media feeds...)
- **Real-time analytics** (social media analytics, advertisement)
- **Gaming leaderboards** (ranked lists in real-time)
- etc.

Redis - on the cloud

- Amazon Web Services **Elasticache** for Redis
- **Microsoft Azure Cache** for Redis in Azure
- Alibaba **ApsaraDB** for Redis in Alibaba Cloud

Redis - customers

Editoo case study

- Small business
- Online tool to create custom magazines
  - personal
  - business

Problem:

- **High latency** due to more people using the application
- Their RDBMS couldn't handle that increase

Solution: Use Redis

- Store user sessions
- Caching database queries

Editoo case study

Results:

- Reduction in downtime
- Higher performance
- Future migrations from its relational databases into Redis

Editoo case study

Results:

- Reduction in downtime
- Higher performance
- Future migrations from its relational databases into Redis

[1] https://redislabs.com/case-studies/

Let's practice!

NOSQL CONCEPTS