

B4M36DS2, BE4M36DS2: **Database Systems 2**

<http://www.ksi.mff.cuni.cz/~svoboda/courses/181-B4M36DS2/>

Practical Class 6

# RiakKV

**Martin Svoboda**

[martin.svoboda@fel.cvut.cz](mailto:martin.svoboda@fel.cvut.cz)

12. 11. 2018

**Charles University**, Faculty of Mathematics and Physics

**Czech Technical University in Prague**, Faculty of Electrical Engineering

# Riak Overview

## RiakKV

- **Highly available distributed key-value store**
- <http://basho.com/products/riak-kv/>

## Data model

Instance (→ bucket types) → **buckets** → **objects**

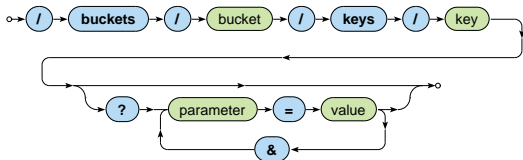
- **Bucket** = logical **collection of objects**
- **Object** = **key-value pair** with metadata
  - **Key** is a Unicode string, **unique within a bucket**
  - **Value** can be anything (text, binary object, image, ...)
  - Each object is also associated with **metadata**

# CRUD Operations

## HTTP API

- All the user requests are submitted as **HTTP requests** with appropriately selected / constructed **methods, URLs, headers,** and **data**

**URL pattern** of HTTP requests for all the CRUD operations



**Optional parameters** (depending on the operation)

# CRUD Operations

## Basic operations on objects

- **Create**: POST or PUT methods
  - **Inserts a key-value pair** into a given bucket
  - Key is specified manually, or will be generated automatically
- **Read**: GET method
  - **Retrieves a key-value pair** from a given bucket
- **Uppdate**: PUT method
  - **Updates a key-value pair** in a given bucket
- **Delete**: DELETE method
  - **Removes a key-value pair** from a given bucket

# HTTP API

## cURL tool

- Allows to **transfer data from / to a server using HTTP** (or other supported protocols)

## Options

- `-X command, --request command`
  - **HTTP request method to be used** (GET, ...)
- `-d data, --data data`
  - **Data to be sent** to the server (implies the **POST method**)
- `-H header, --header header`
  - **Extra headers** to be included when sending the request
- `-i, --include`
  - Prints both headers and (not just) body of a response

# First Steps

## Remotely connect to our NoSQL server

- SSH and SFTP access
- PuTTY and WinSCP on Windows
- **nosql.ms.mff.cuni.cz:42222**

## Check Riak cluster status

- `curl -v http://localhost:10011/ping`
- *And with higher permissions...*
  - `riak ping`
  - `riak-admin test`
  - `riak-admin status`
  - `riak-admin status | grep ring_members`

# Read and Write Operations

## Insert object for a new actor

- Prefix all the bucket names with your `login`

```
curl -i -X PUT
  -H 'Content-Type: text/plain'
  -d 'Ivan Trojan, 1964'
  http://localhost:10011/buckets/login_actors/keys/trojan
```

## Retrieve the previously inserted actor

- Examine both response body and headers

```
curl -i -X GET
  http://localhost:10011/buckets/login_actors/keys/trojan
```

# Bucket Operations

## List all the buckets

- Only buckets with at least one object will be included

```
curl -i -X GET  
http://localhost:10011/buckets?buckets=true
```

## List all the keys in the bucket of actors

- Note that this operation cannot be executed efficiently

```
curl -i -X GET  
http://localhost:10011/buckets/login_actors/keys?keys=true
```



# Update and Delete Operations

## Update our actor object

```
curl -i -X PUT
  -H 'Content-Type: application/json'
  -d '{ "name" : "Ivan Trojan", "year" : 1964 }'
  http://localhost:10011/buckets/login_actors/keys/trojan
```

## Check the updated actor object

- Use different virtual nodes as well
- localhost:10011, localhost:10012, localhost:10013

## Remove the actor object

```
curl -i -X DELETE
  http://localhost:10011/buckets/login_actors/keys/trojan
```

# Sample Data

## Insert objects for new actors

- Put the data into `login_actors` bucket
- Use `application/json` content type

```
{ "name" : "Ivan Trojan", "year" : 1964 }
```

```
{ "name" : "Jiří Macháček", "year" : 1966 }
```

```
{ "name" : "Jitka Schneiderová", "year" : 1973 }
```

```
{ "name" : "Zdeněk Svěrák", "year" : 1936 }
```

# Sample Data

## Insert objects for new movies

- Put the data into `login_movies` bucket
- Use `application/json` content type once again

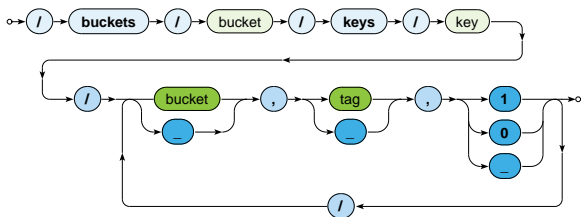
```
{  
  "title" : "Vratné lahve", "year" : 2006,  
  "actors" : [ "Zdeněk Svěrák", "Jiří Macháček" ]  
}
```

```
{  
  "title" : "Samotáři", "year" : 2000,  
  "actors" : [ "Jitka Schneiderová", "Ivan Trojan", "Jiří Macháček" ]  
}
```

```
{  
  "title" : "Medvídek", "year" : 2007,  
  "actors" : [ "Jiří Macháček", "Ivan Trojan" ]  
}
```

# Links and Link Walking

**Links** = directed relationships between objects



## Parameters

- *Bucket* – assumes only a given target bucket
- *Tag* – considers only a given link tag
- *Keep* – whether the objects should be included in the result

# Links and Link Walking

## Create new links actor → movie

```
curl -i -X PUT
  -H 'Content-Type: application/json'
  -H 'Link: </buckets/login_movies/keys/samotari>; riactag="tmovie"'
  -H 'Link: </buckets/login_movies/keys/medvidek>; riactag="tmovie"'
  -d '{ "name" : "Ivan Trojan", "year" : 1964 }'
  http://localhost:10011/buckets/login_actors/keys/trojan
```

## Check the updated actor object

- Verify the presence of links in particular

## Traverse the links from the actor

```
curl -i -X GET
  http://localhost:10011/buckets/login_actors/keys/trojan
  /login_movies,tmovie,1
```

# Links and Link Walking

**Add all the links movie  $\rightarrow$  actor**

**Express a more complicated link walking query**

- Find all the actors who appeared in movies where *Trojan* stared

# Search 2.0

## Create a full-text index for the bucket of actors

```
curl -i -X PUT
  -H 'Content-Type: application/json'
  -d '{ "schema" : "_yz_default" }'
  http://localhost:10011/search/index/login_iactors
```

```
curl -i -X PUT
  -H 'Content-Type: application/json'
  -d '{ "props" : { "search_index" : "login_iactors" } }'
  http://localhost:10011/buckets/login_actors/props
```

## Verify the new bucket properties

```
curl -i -X GET
  http://localhost:10011/buckets/login_actors/props
```

# Search 2.0

## Reinsert objects for all the actors

- Note that names of fields were changed...
  - Suffixes recognizable by the JSON extractor were added
  - Czech accented characters were removed

```
{ "name_s" : "Ivan Trojan", "year_i" : 1964 }
```

```
{ "name_s" : "Jiri Machacek", "year_i" : 1966 }
```

```
{ "name_s" : "Jitka Schneiderova", "year_i" : 1973 }
```

```
{ "name_s" : "Zdenek Sverak", "year_i" : 1936 }
```



# Search 2.0

## Find all the actors born in 1964

```
curl -i -X GET  
  'http://localhost:10011/search/query/  
  login_iactors?wt=json&omitHeader=true&q=year_i:1964'
```

## Express a more complicated full-text query

- Find all the actors who were born in 1960 or later and their name contains substring *de*

# References

## Riak documentation

- <http://docs.basho.com/riak/kv/2.1.4/>

## Search queries (Apache Solr query syntax)

- <http://docs.basho.com/riak/kv/2.1.4/developing/usage/search/>
- [https://lucene.apache.org/solr/guide/6\\_6/the-standard-query-parser.html](https://lucene.apache.org/solr/guide/6_6/the-standard-query-parser.html)