NIE-PDB | **Advanced Database Systems** | 2024/25 Winter

Assignment **04** – **Riak**

Assignment

- Create a **shell script** that will work with data in our **Riak database** via its **HTTP interface** using the `cURL` tool
- Insert about **3 key/value objects** into at least **3 buckets**, each containing objects of different entity types
  - Always **include content headers**
  - One bucket must contain objects with **XML values** (`text/xml`)
  - One bucket must contain objects with **JSON values** (`application/json`)
  - As for the third bucket, choose any content type you like
- Design your XML and JSON values so that they can be **indexed by Yokozuna**
  - I.e., do not use Czech or other national accented characters
  - Use each of the following **type suffixes** at least once
    - `_s` or `_ss` (string), `_i` or `_is` (integer), and `_b` or `_bs` (boolean)
- Associate your **search index** with at least the two XML and JSON buckets
- Express at least 2 index **search queries**
  - Use each of the following constructs at least once: wildcards, ranges, logical operators
- **Remove all your objects** at the end of your script (i.e., empty all your buckets)

Requirements

- Our Riak cluster is accessible via nodes running at `https://nosql.kti.in.fit.cvut.cz:10021/` or `10022` or `10023`
  - You must be connected to our NoSQL server via PuTTY / SSH, though
  - These nodes will otherwise not be reachable from outside since the listed ports are blocked
- Use **bucket type** with name `pdb241_login` for all buckets you would like to create
  - Of course, replace `login` with your actual login name
  - E.g.: `/types/pdb241_svobom25/buckets/actors/keys/trojan` for a particular actor object in a bucket of actors
  - This bucket type already exists, you thus do not need to take care of its creation
- However, do not access your buckets directly so that the bucket type can easily be changed
  - I.e., write `$RIAK_TYPE` instead of a fixed name when **referencing your bucket type** in URLs
  - `$RIAK_TYPE` is a variable provided from outside of your script (see below)
  - Its value will correspond to your actual login name `pdb241_login`, and so your bucket type name
  - E.g.: `/types/$RIAK_TYPE/buckets/actors/keys/trojan`
- Similarly, it is necessary to use the following **variables** to provide the necessary parameters
  - `$RIAK_HOST` for Riak instance host name
  - `$RIAK_USER` for your login name and `$RIAK_PASSWORD` for your password
- Your requests will therefore correspond to the following pattern
  - ```
    curl -i -X GET -u $RIAK_USER:$RIAK_PASSWORD \
    https://$RIAK_HOST:10021/types/$RIAK_TYPE/buckets/actors/keys/trojan
    ```
- When working with your XML and JSON values, it is better to wrap them by single quotes and not double quotes
  - The reason is that double quotes are needed by XML attributes and JSON strings
    ```

  - E.g.: `'{ name_s : "Ivan Trojan", year_i : 1964 }'`
- Use a **search index** with name corresponding to the pattern `pdb241_login_index` (i.e., your login name suffixed with `_index`)
  - Do not create this index, it already exists
  - Note that you have to associate it with your buckets before you start inserting any objects
  - Otherwise your already existing objects will not become indexed
- Note that variables enclosed in single quotes will not be replaced by their values
  - You therefore need to place them outside of these single quotes
  - E.g.: `'{ "props" : { "search_index" : "'$RIAK_USER'_index" } }'`
- As expected, do not access your search index directly in your URLs
  - Use `$RIAK_USER""_index` instead
  - This approach with a pair of double quotes is necessary in order not to treat our fixed suffix `_index` as a part of a variable name
- When preparing your **search conditions**, you must work carefully
  - The reason is that certain characters are treated specifically both by shell as well as in URLs
  - First, prepare your actual search condition at the logical level
    - E.g.: `(year_i:[1960 TO *])`
  - Second, encode unsafe characters: space `%20`, left bracket `%5B`, and right bracket `%5D`
    - E.g.: `(year_i:%5B1960%20TO%20*%5D)`
  - Finally, escape round parentheses
    - E.g.: `\(year_i:%5B1960%20TO%20*%5D\)`
  - Note that you also need to escape ampersands in query parameters
    - E.g.: `...\&q=\(year_i:%5B1960%20TO%20*%5D\)`
- Your search requests will therefore correspond to the following pattern
  - `curl -i -X GET -u $RIAK_USER:$RIAK_PASSWORD \ https://$RIAK_HOST:10021/search/query/$RIAK_USER_index?wt=json\&omitHeader=true\ \&q=\(year_i:%5B1960%20TO%20*%5D\)`
- Always comment the intended **meaning of search queries in natural language**
  - Comments are written as `# comment`
- Each search query must be evaluated to a **non-empty set of matching results**
- Make sure your **shell script** is **executable** at all (i.e., has the `X` permission assigned)
  - If you are using WinSCP, just locate your file, go to the properties context menu and add the X permission for the owner
  - If you are using PuTTY, execute the following command `chmod u+x script.sh`
- Also make sure your script can be **executed repeatedly** without failures
- Only use Linux style of **line endings**
  - I.e., use `LF = chr(10) = "\n"` instead of `CRLF = chr(13).chr(10) = "\r\n"` on Windows or `CR = chr(13) = "\r"` on Mac
- If something is not working as expected, try to execute Riak **ping query**
  - `curl -i -X GET -u $RIAK_USER:$RIAK_PASSWORD https://$RIAK_HOST:10021/ping`

Submission

- **script.sh**: Bash script allowing to execute all the HTTP requests

Execution

- First of all, define appropriate values for all the three required variables
  - `export RIAK_HOST="nosql.kti.in.fit.cvut.cz"` (our nosql server itself)
  - `export RIAK_USER="pdb241_login"` (your actual login name)

  - `export RIAK_PASSWORD="MyPassword"` (your actual password)
  - `export RIAK_TYPE=$RIAK_USER` (your bucket type)
- Then, execute the following shell command to evaluate the whole Riak script as such
  - `./script.sh`
- Do not put the aforementioned variable exports into the `script.sh` file itself

Tools

- **RiakKV** (3.0.10) – https://www.tiot.jp/en/solutions/riak/
  - Already installed on the NoSQL server

References

- **Riak KV 3.0.4 Documentation**
  - https://www.tiot.jp/riak-docs/riak/kv/3.0.4/
- **Riak KV 3.0.4 Search 2.0 (Yokozuna) Documentation**
  - https://www.tiot.jp/riak-docs/riak/kv/3.0.4/developing/usage/search/
- **Apache Solr Query Parser 7.3 Documentation**
  - https://lucene.apache.org/solr/guide/7_3/the-standard-query-parser.html