Redis Project
Relational databases & Key-Value systems

Athens University of Economics and Business
Dpt. Of Management Science and Technology
Prof. Damianos Chatziantoniou

Lamprini Koutsokera (8130074) | lkoutsokera@gmail.com
Stratos Gounidellis (8130029) | stratos.gounidellis.gmail.com

BDSMasters

SQL Server vs. Redis

	Microsoft SQL Server	redis
Description	Microsoft's relational DBMS	In-memory data structure store, used as database
Database model	Relational DBMS	Key-value store
Implementation language	C++	C
Data scheme	yes	schema-free
Triggers	yes	no
Replication methods	yes, depending the SQL-Server Edition	Master-slave replication
Partitioning methods	tables can be distributed across several files, sharding through federation	Sharding

Project

from a relational database to a key-value system

References: [1]

From **theory** to **practice**

Tools


redis


python™

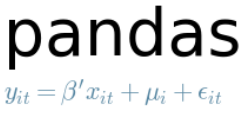

jupyter

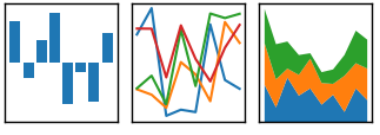

ANACONDA®


ubuntu®

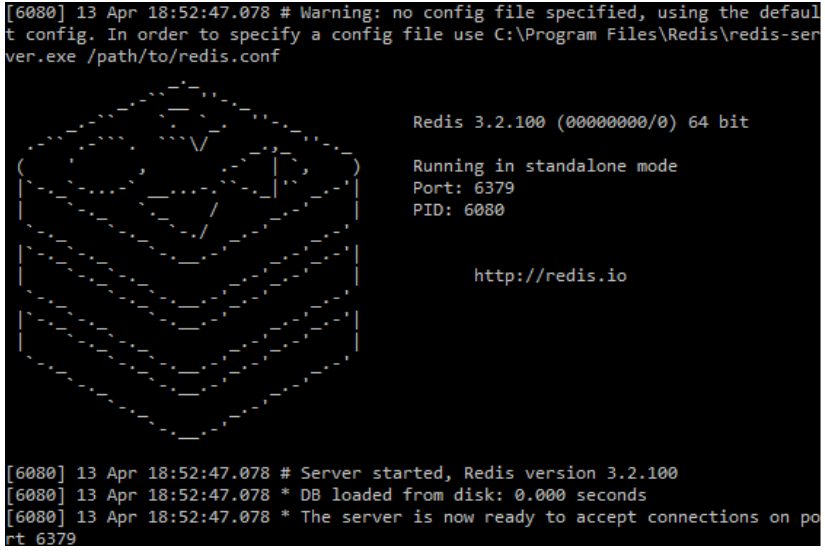

NumPy


GitHub


pandas
$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$



Required installations



Redis installation

```
wget http://download.redis.io/redis-stable.tar.gz
tar xvzf redis-stable.tar.gz
cd redis-stable
make
```

```
127.0.0.1:6379> ping
PONG
127.0.0.1:6379>
```

check if Redis is working

Required python packages installation

```
pip install -r requirements.txt
```

```
numpy==1.11.3
pandas==0.19.2
redis==2.10.5
tabulate==0.7.7
```

References: [2]

From **software configuration** to **coding**

Python coding [1] – table parsing

Relational Data Insertion

```
class RedisTableParser(object):
    """RedisTableParser: Implementation of the methods needed
       to successfuly create a table in the Redis database.
    """
    def sqlTableToRedis(self, tableFile):
        """Create a Redis Table parsing data from an SQL Table
        through a file.

        :param self: An instance of the class RedisTableParser.
        :param tableFile: A file that contains data from an SQL
            Table.
        """
    def recordsInsertion(r, string, fields, table, tableId):
        """Insert in redis database the records.

        :param r: An instance of connection to redis.
        :param string: A string delimeted with ";",
            containing a record.
        :param fields: The attributes of the table.
        :param table: The name of the table to be inserted.
        :param tableId: The table counter.
        """
```

Student -SQL Table

SSN	FName	LName	Address	Age
12938	Nikos	Papadopoulos	Hydras 28, Athens	42
18298	Maria	Nikolaou	Kifisias 33, Marousi	34
81129	Dimitris	Panagiotou	Alamanas 44, Petralona	29

Student -SQL Table in text file

```
Student
SSN
FName
LName
Address
Age
;
12938;Nikos;Papadopoulos;Hydras 28, Athens;42
18298;Maria;Nikolaou;Kifisias 33, Marousi;34
```

```
1) "Student_SSN_3"
2) "Student_SSN_2"
3) "StudentId"
4) "Student_PrimaryKeys"
5) "Student_Age_1"
6) "Student_Address_1"
7) "Student_Age_3"
8) "Student_LName_1"
9) "Student_Address_3"
10) "Student_FName_2"
```

```
127.0.0.1:6379> get Student_SSN_1
"12938"
127.0.0.1:6379> get Student_FName_1
"Nikos"
127.0.0.1:6379> get Student_LName_1
"Papadopoulos"
127.0.0.1:6379> get Student_Address_1
"Hydras 28, Athens"
127.0.0.1:6379> get Student_Age_1
"42"
```

References: [3]

Python coding [2] – query parsing

SQL Query

```
SELECT Student.FName, Student.LName, Grade.Mark
FROM Student, Grade
WHERE Student.SSN=Grade.SSN
ORDER BY Student.Age ASC
LIMIT 2
```

SQL Query in text file

```
Student.FName, Student.LName, Grade.Mark
Student, Grade
Student.SSN=Grade.SSN
Student.Age ASC
2
```


```python
try:
    Student_FName_List = sorted(r.keys(pattern='Student_FName*'))
    Student_LName_List = sorted(r.keys(pattern='Student_LName*'))
    Grade_Mark_List = sorted(r.keys(pattern='Grade_Mark*'))
    Student_Age_List = sorted(r.keys(pattern='Student_Age*'))
    Student_SSN_List = sorted(r.keys(pattern='Student_SSN*'))
    Grade_SSN_List = sorted(r.keys(pattern='Grade_SSN*'))

    resultsArray = np.zeros(4)

    for Student_FName, Student_LName, Student_Age, Student_SSN in zip(Student_FName_List, Student_LName_List, Student_Age_List, Student_SSN_List):
        for Grade_Mark, Grade_SSN in zip(Grade_Mark_List, Grade_SSN_List):
            if r.get(Student_SSN) == r.get(Grade_SSN):
```

References: [3]

Python coding [3] – query parsing

Query Execution

```
class RedisQueryParser(object):
    """RedisQueryParser: Implementation of the methods needed
       to successfuly retrieve the expected results from the
       Redis database.
    """
    def checkNumeric(inputString):
        """Check whether a given string is numeric or not.

        :param inputString: A string from the query text file.
        :return: True, if the inputString is numeric.
            Otherwiser, return False.
        """
    def parseSqlQuery(queryFile):
        """Determine the clauses included in the query text file

        :param queryFile: A file with the query clauses.
        :return: A tuple with the different clauses.
        """
    def pythonFileInitialize():
        """Initialize the python file to be created with some
            basic imports and methods' calls.

        :return: A string with initialization of the python file
        """
```

```
def convertToRedisWhere(whereQuery, startString,
                        endString, flag=True, forCheck=None):
    """Tailor the WHERE clause according to the syntax and the logic
        of Python.

    :param whereQuery: A string with the WHERE clause.
    :param startString: A string with the character(-s) the
        search term should start.
    :param endString: A string with the character(-s) the
        search term should end.
    :param flag: Boolean variable to check whether the search term
        has already been tailored.
    :param forCheck: Either None or a List with the tables in
        FORM clause of the query.
    :return: A string with the transformed WHERE clause.
    """
def convertStringToNumber(self, whereQuery, startString, endString):
    """Tailor the WHERE clause according to the syntax and the logic
        of Python (numeric values).

    :param self: An instance of the class RedisQueryParser.
    :param whereQuery: A string with the WHERE clause.
    :param startString: A string with the character(-s) the
        search term should start.
    :param endString: A string with the character(-s) the
        search term should end.
    :return: A string with the transformed WHERE clause, based on the
        numeric values.
    """
```

References: [3]

Python coding [4] – query parsing

Query Execution

```
def checkNumericBeforeOperator(dictReplaceAfterNew, whereQuery,
                               startString):
    """Tailor the WHERE clause according to the syntax and the logic
       of Python (numeric values).

    :param dictReplaceAfterNew: A dictionary with the indexes of the
       numeric values found in the WHERE clause.
    :param whereQuery: A string with the WHERE clause.
    :param startString: A string with the character(-s) the
       search term should start.
    :return: A string with the transformed WHERE clause, based on the
       numeric values.
    """
def selectFromToRedis(selectQuery, fromQuery, whereQuery,
                      selectQuerySplitOrder):
    """Parse and edit the SELECT and FROM clauses in order to be translate
       to python according to its syntax and logic rules.

    :param selectQuery: A string with the SELECT clause.
    :param fromQuery: A list with the tables in the FROM clause.
    :param whereQuery: A string with the WHERE clause.
    :param selectQuerySplitOrder: A list with the attributes included in
       the ORDER BY clause.
    :return: A tuple with the string including the lists to be created,
       the updated "SELECT" clause, the attributes that should be
       retrieved from redis (and their number) that are not included
       in the SELECT clause but they are included in the WHERE clause
       and the attributes that should be retrieved from redis.
    """
```

```
def orderQueryToRedis(orderQuery, selectQuery):
    """Parse and edit the ORDER clause in order to be translated
       to python according to its syntax and logic rules.

    :param orderQuery: A string with the ORDER clause.
    :param selectQuery: A string with the SELECT clause.

    :return: A tuple with the field according to which the results will
       be ordered, a variable to check whether the order will
       be ascending or descending, the updated "SELECT" clause and a
       variable to check whether the order field is included in the SELECT
       clause or not.
    """
def whereToRedis(self, fromQuery, whereQuery):
    """Parse and edit the WHERE clause in order to be translated
       to python according to its syntax and logic rules.

    :param self: An instance of the class RedisQueryParser.
    :param fromQuery: A list with the tables in the FROM clause.
    :param whereQuery: A string with the WHERE clause.

    :return: A string with the python-like WHERE clause.
    """
```

References: [3]

Python coding [5] – query parsing

Query Execution

```
def pythonFileArrayResults(selectQuerySplit, whereQuery, counterTab):
    """Create the content of the python file responsible for
       saving the results properly in a numpy array.

    :param selectQuerySplit: A list with the attributes in the
       SELECT clause.
    :param whereQuery: A string with the WHERE clause.

    :return: A string with the content of the python file,
       which will save the results of the query in a numpy
       array.
    """
def pythonFileForLoop(selectQuerySplit, selectQuery,
                      keysList, fromQuery):
    """Construct the main for loop of the output python file,
       in order to iterate over the results retrieved from
       the Redis database.

    :param selectQuerySplit: A list with the attributes in the
       SELECT clause.
    :param selectQuery: A string with the SELECT clause.
    :param counterWhere: The number of attributes contained in
       the WHERE clause but not in the SELECT clause.
    :param keysList: A string with the necessary content
       to iterate over the different attributess.

    :return: A string with the content of the python file,
       which will iterate over the results.
    """
```

```
def pythonFileLimitOrderQuery(
        orderQuery, orderFlag, limitQuery,
        orderField, orderFieldExists, randomCheck):
    """Construct the main for loop of the ouput python file,
       in order to iterate over the results retrieved from
       the Redis database.

    :param orderQuery: A string with the ORDER clause.
    :param orderFlag: A boolean variable to check whether the
       ordering will be ascending or descending.
    :param limitQuery: A string with the LIMIT clause, i.e.
       the number of results to be printed.
    :param orderField: The field according to which the
       results will be ordered.
    :param orderFieldExists: A boolean variable to check whether the
       ordering field is included also in the SELECT clause or not.
    :param randomCheck: A boolean variable to check whether the
       results should be printed in random order.

    :return: A string with the content of the python file,
       related mainly with the formatting of the way the results
       are printed.
    """
```

References: [3]

Python coding [6] – query parsing

Query Execution

```
def sqlQueryToRedis(self, selectQuery, fromQuery, whereQuery, orderQuery,
                    limitQuery):
    """Call the methods required to build the output file.

    :param self: An instance of the class RedisQueryParser.
    :param selectQuery: A string with the SELECT clause.
    :param fromQuery: A string with the FROM clause.
    :param whereQuery: A string with the WHERE clause.
    :param orderQuery: A string with the ORDER clause.
    :param limitQuery: A string with the LIMIT clause.

    :return: A string with the final complete content of the
        python file.
    """
def checkSyntax(outputPython):
    """Check the syntax of the created python file.

    :param outputFile: The name of the output file to be created.
    """
def writePythonFile(outputFile, sourceCode):
    """Write the source code on the python file specified.

    :param outputFile: The name of the output file to be created.
    :param sourceCode: The source code to be written in the output
        python file.
    """
```

References: [3]

Python coding [7] – code metrics

Relational Data Insertion

type	number	%	previous	difference
code	66	62.26	66	=
docstring	26	24.53	26	=
comment	1	0.94	1	=
empty	13	12.26	13	=

Query Execution

type	number	%	previous	difference
code	419	60.81	418	+1.00
docstring	207	30.04	207	=
comment	1	0.15	1	=
empty	62	9.00	62	=

Unit Testing

type	number	%	previous	difference
code	72	64.29	72	=
docstring	21	18.75	21	=
comment	1	0.89	1	=
empty	18	16.07	18	=

Assumptions - Restrictions

- The **text file** follows the structure described below:
  - first line (**SELECT**): a list of table_name.attribute_name, delimited by the character ",".
  - second line (**FROM**): a list of table names, delimited by the character ",".
  - third line (**WHERE**): a simple condition, consisting only of ***AND***, ***OR***, ***NOT***, **=**, **<>**, **>**, **<**, **<=**, **>=** and ***parentheses***.
  - fourth line (**ORDER BY**): a simple clause, containing either an attribute name and the way of ordering (***ASC*** or ***DESC***) or ***RAND()***.
  - fifth line (**LIMIT**): a number, specifying the number of rows to be displayed.
- The **ORDER BY** clause contains only one attribute.
- The sql query is correct according to the **sql syntax**.
- The **names** of the tables and the attributes are correct.
- In case a clause is **skipped** then the corresponding line remains **blank**.

Query examples / results [1]

1

```
Student.FName, Student.LName, Grade.Mark
Student, Grade
Student.SSN=Grade.SSN
```

	Student_FName	Student_LName	Grade_Mark
0	Christina	Poluzou	6
1	Eleni	Petrou	4
2	Giannis	Papaspurou	6
3	Michalis	Nikolaou	7
4	Stratos	Gounidellis	2
5	Lamprini	Koutsokera	8
6	Dimitris	Panagiotou	7
7	Maria	Nikolaou	9
8	Damianos	Chatziantoniou	5
9	Dimitra	Papadimitriou	7
10	Nikos	Papadopoulos	10

```
Total rows:  11
The results have been saved in the file resultsRedis.csv!
```

2

```
Student.FName, Student.Address, Student.LName
Student
Student.FName < "Nikos1"
Student.FName asc
2
```

	Student_FName	Student_Address	Student_LName
0	Christina	Krustalli 32, Kifisia	Poluzou
1	Damianos	Kolokotroni 59B, Athina	Chatziantoniou

```
Total rows:  2
The results have been saved in the file resultsRedis.csv!
```

3

```
Stu.FName, Student.Address, Student.LName
Student, Grade
Stu.FName asc, Student.LName desc
```

```
Exception:
ERROR! Please check the syntax of the query. Output python file is not created! :(
```

Query examples / results [2]

4

```
Student.FName, Student.LName, Grade.Mark
Student, Grade
(Student.SSN=Grade.SSN and Grade.Mark > 5) and (Student.FName='Maria' or Student.FName='Nikos')
Grade.Mark asc
8
```

	Student_FName	Student_LName	Grade_Mark
0	Nikos	Papadopoulos	10
1	Maria	Nikolaou	9

```
Total rows:  2
The results have been saved in the file resultsRedis.csv!
```

5

```
Student.FName, Student.LName, Student.Age
Student


7
```

	Student_FName	Student_LName	Student_Age
0	Christina	Poluzou	35
1	Eleni	Petrou	47
2	Giannis	Papaspurou	32
3	Michalis	Nikolaou	27
4	Stratos	Gounidellis	22
5	Lamprini	Koutsokera	22
6	Dimitris	Panagiotou	17

```
Total rows:  7
The results have been saved in the file resultsRedis.csv!
```

Query examples / results [3]

6

```
Student.FName, Student.LName, Student.Address
Student

RAND()
3
```

	Student_FName	Student_LName	Student_Address
0	Eleni	Petrou	Kountouriwtou 29, Peiraias
1	Christina	Poluzou	Krustalli 32, Kifisia
2	Michalis	Nikolaou	Dionusou 66, Marousi

```
Total rows:  3
The results have been saved in the file resultsRedis.csv!
```

	Student_FName	Student_LName	Student_Address
0	Stratos	Gounidellis	Ilia Iliou 2, Neos Kosmos
1	Giannis	Papaspurou	Patision 13, Athina
2	Michalis	Nikolaou	Dionusou 66, Marousi

```
Total rows:  3
The results have been saved in the file resultsRedis.csv!
```

7

```
Grade.SSN, Student.FName, Student.LName
Student, Grade
Student.SSN=Grade.SSN and (Grade.Mark <> 8 and Grade.Mark >= 6)
```

	Grade_SSN	Student_FName	Student_LName
0	12385	Christina	Poluzou
1	13647	Giannis	Papaspurou
2	12358	Michalis	Nikolaou
3	12129	Dimitris	Panagiotou
4	18298	Maria	Nikolaou
5	10036	Dimitra	Papadimitriou
6	12938	Nikos	Papadopoulos

```
Total rows:  7
The results have been saved in the file resultsRedis.csv!
```

Style check, code analysis & unit testing

[1] pep8 → [2] flake8 → [3] pylint

```
class TestRredisQueryParser(unittest.TestCase):
    """TestRredisQueryParser: Implementation of the methods needed
       to successfuly test the expected results from the
       SQL Query Parsing.
    """
     def test_readSqlQuery(self):
         """Test whether a given query is read correctly or not.
         """
    def test_selectFromToRedis(self):
        """Test whether the SELECT clause is converted correctly or not.
        """
    def test_orderQueryToRedis(self):
        """Test whether the ORDER BY clause is converted correctly or not.
        """

    def test_whereQueryToRedis(self):
        """Test whether the WHERE clause is converted correctly or not.
        """
    def test_exceptionSyntaxError(self):
        """Test whether the syntax of the created python file is correct.
        """
```

References

[1] Db-engines.com. (n.d.). *Memcached vs. Microsoft SQL Server vs. Redis Comparison*. [online] Available at: https://db-engines.com/en/system/Memcached%3bMicrosoft+SQL+Server%3bRedis [Accessed 13 Apr. 2017].

[2] Redis.io. *Redis Quick Start.* https://redis.io/topics/quickstart [Accessed 12 Apr. 2017].

[3] Peter Cooper. *Redis 101 - A whirlwind tour of the next big thing in NoSQL data storage.* https://www.scribd.com/document/33531219/Redis-Presentation [Accessed 12 Apr. 2017].