BIG DATA MANAGEMENT SYSTEMS: PROJECT #2 – REDIS/KEY-VALUE STORES

May 5, 2019

Zoe Kotti and Chryssa Nampouri
Department of Management Science and Technology
Athens University of Economics and Business
Athens, Greece
{t8150062, t8150096}@aueb.gr

Supervisor: Prof. Damianos Chatziantoniou

Contents

1 Redis Project Description

A key-value store is a system that manages a collection of (*key,value*) pairs, where *key* is unique in this universe. Redis – and other systems – allow the value to be a single value (e.g. string, number), a set of values, a list of values, a hash, etc.

Assume a collection of (key, list) pairs, i.e. the value is a list of values, namely strings. Assume that key is a string as well. Let's call such a collection a *Key-List Store (KL Store)*. This is a special case of a multi-map data structure, where several values are mapped to a key.

Example of a Key-List Store:

Key	List
12	[t12, t67]
34	[t87, t12, t98]
...	...
76	[t121, t72, t99, t179]

Assume two domains of values D_1 and D_2

e.g. D_1 = {all possible customer ids}, D_2 = {all possible transaction ids}

Assume that there is a process P that generates a collection of value1, value2 pairs

$S = \{(u, v) : u \in D_1, v \in D_2\}$

Examples of such processes:

- SELECT custID, transID FROM SALES
- Reading a CSV file and getting for each line forming a pair using columns i and j
- Running any program that produces a stream of pairs of values

Given a collection S described as above, one can define two KLStores, $KL_1(S)$ and $KL_2(S)$ as follows:

$KL_1(S) = \{(x, L_x), \forall x \in U = \{u : (u, v) \in S\}$, L_x = the list of values v, such that $(x, v) \in S\}$

$KL_2(S) = \{(x, L_x), \forall x \in V = \{v : (u, v) \in S\}$, L_x = the list of values u, such that $(u, x) \in S\}$

We want to implement in Python (or some other language) the following functions/methods that get one or more KL stores and "return" (or update) a KL store. All these KL stores should exist in Redis.

1.1 Create_KLStore(name, data-source, query-string, position1, position2, direction)

This function creates in Redis a KL store with name <name> using the data source found in <data-source>. Data sources can be found in an XML file described later and for the scope of this project can be either a CSV file, a relational database or an excel file. In the case of a CSV file, <query-string> is empty and <position1> and <position2> two integer numbers specifying the column positions that will be used to form the (u,v) pairs of S (as described earlier). In the case of an excel, <query-string> contains the index of the worksheet and <position1> and <position2> two integer numbers specifying the column positions that will be used to form the (u,v) pairs of S (as described earlier). In the case of a relational database, <query-string> is an SQL statement in the form SELECT col1, col2 WHERE <etc>. <direction> has the value 1 or 2, specifying whether $KL_1(D)$ or $KL_2(D)$ should be implemented.

1.2 Filter_KLStore(name1, expression)

This function gets a KL store in Redis named <name1> and a string called <expression> representing a valid python boolean expression and applies this expression on each element of each list of <name1>. If the return value is **true**, the element remains in the list, otherwise it is removed. Come up with a convention on how the element of the list is mentioned within the <expression>.

1.3 Apply_KLStore(name1, func)

This function gets a KL store in Redis named <name1> and a python function named <func> – which gets a string and returns a string – and applies <func> on each element of a list, for all lists of the KL store <name1>, transforming thus the lists of the KL store.

1.4 Aggr_KLStore(name1, aggr)

This function gets a KL store in Redis named <name1> and a string named <aggr> that can have one of the values "avg/sum/count/min/max" and aggregates each list of the KL store <name1> according to the specified aggregate, updating the list with just one item, the result of the aggregation. You can implement a more general version of this function that also gets a python function <func> that operates on a list of strings and returns a

string; in this case you should modify the signature of Aggr_KLStore appropriately (e.g. Aggr_KLStore(name1, aggr, func), if <aggr> is an empty string then use <func> for aggregation).

1.5 LookUp_KLStore(name1, name2)

This function gets two KL stores named <name1> and <name2> and for each element e of a list L in <name1>, performs a lookup for e in the keys of <name2>, gets the list L' of the matched key, and replaces e in L with the elements of L'. This should happen for all lists in <name1>. This is graphically shown in the figure below.

<name1>

Key	List
12	[t12, t13]
18	[t3…]
…	…
56	[t82, t62, t71]

1

<name2>

3

2

4

Key	List
t11	[120, 67, 98]
t12	[68, 139, 65]
t13	[55, 12]
…	…
t99	[45, 89]

New list for 12 → [68, 139, 65, 55, 12]

1.6 ProjSel_KLStore(output_name, pname1, pname2, …, pnamek, expression)

Let's assume n KL stores, $nm\_1, nm\_2, ..., nm\_n$, sharing the same keys, i.e. key column is drawn from the same domain D. The figure below shows such an example for three KL stores.

nm_1	
Key	**List**
t11	[31, 62, 9]
t15	[75, 91]
t22	[55, 12, 112]
t39	[44]
t44	[42, 98]

nm_2	
Key	**List**
t11	[12, 6, 95]
t15	[128]
t22	[43]
t32	[39, 77]
t44	[129]

nm_3	
Key	**List**
t11	[182]
t16	[7, 9]
t22	[56, 29]
t38	[32, 82]
t44	[66, 121, 22]

The goal of this function (operator) is to perform a join on the common keys of some KL stores, creating a new KL store having keys the common keys and corresponding list the concatenation of the individual lists in $nm\_1, nm\_2, ..., nm\_n$. In other words, if there is a key k in all KL stores $nm\_1, nm\_2, ..., nm\_n$, and $L_k^1, L_k^2, ..., L_k^n$ the corresponding lists, then we insert a key-list pair in the new KL store as (k, concatenation($L_k^1, L_k^2, ..., L_k^n$)). In the example above, the new KL store would be the following:

Key	**List**
t11	[31, 62, 9, 12, 6, 95, 182]
t22	[55, 12, 112, 43, 56, 29]
t44	[42, 98, 129, 66, 121, 22]

In addition, we would like to specify at the same time a filtering condition for this new KL store, based on the key and the contents of the lists involved, for example "$key <>' t22'$ and $nm_2 > 20$". Such an expression is ill-defined because lists are not atoms. However, we would like to keep these semantics for reasons that have to do with user experience and understanding at the conceptual level (not discussed here). For the scope of this assignment, translate it as "any element of the list, randomly chosen, for example the first one".

1.7 Dictionary of Data Sources

There is a dictionary in XML describing data sources:

```
<datasources Repository="SomeName">
    <datasource name="DSName" id="someID" type="db/excel/csv">
        <dbconnect>
            <username>username</username>
            <password>password</password>
```

```
            <request>string</request>
            <database>db</database>
        </dbconnect>
        <filename>someFileName</filename>
        <path>somePath</path>
        <delimiter>someDelimiter</delimiter>
    </datasource>
</datasources>
```

2 Function Implementation in Python

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

import csv
import pymysql
import re
import redis
import xlrd
import xml.etree.ElementTree as ET


# Function 1
def Create_KLStore(name, data_source, query_string, position1,
    position2, direction):
    """
    This function creates in Redis a KL store using a data source
    from an XML file.

    :param name: The KL store to create in Redis
    :param data_source: A csv file, a relational database, or an Excel file
    :param query_string: The index of the worksheet when the source is an
        Excel file, or an SQL statement when the source is a relational
        database
    :param position1, position2: Integer numbers specifying column
        positions
    :param direction: 1 for KL1(D) and 2 for KL2(D)
    """
    # Parse XML file
    root = ET.parse(data_source).getroot()

    for source in root.findall('datasource'):
        type = source.get('type')

        if type == 'csv':
            path_csv = source.find('path').text
            filename_csv = source.find('filename').text
            filepath_csv = path_csv + filename_csv
            delimiter = source.find('delimiter').text
        elif type == 'excel':
            path_excel = source.find('path').text
```

```
41              filename_excel = source.find('filename').text
42              filepath_excel = path_excel + filename_excel
43          else:
44              username = source.find('dbconnect/username').text
45              password = source.find('dbconnect/password').text
46              request = source.find('dbconnect/request').text
47              database = source.find('dbconnect/database').text
48
49      # Fetch data from csv file
50      if type == 'csv':
51
52          with open(filepath_csv, 'r') as csv_file:
53
54              csv_reader = csv.reader(csv_file, delimiter = delimiter)
55              # Skip first line with column titles
56              next(csv_reader)
57
58              key_list_dict = {}
59
60              unique_pairs_list = []
61
62              # Direction based on user
63              if direction == 1:
64
65                  for row in csv_reader:
66                      user_id = row[position1]
67                      transaction_id = row[position2]
68
69                      if [user_id, transaction_id] in unique_pairs_list:
70                          continue
71
72                      name.lpush(user_id, transaction_id)
73
74                      unique_pairs_list.append([user_id, transaction_id])
75
76              # Direction based on transaction
77              elif direction == 2:
78
79                  for row in csv_reader:
80                      user_id = row[position1]
81                      transaction_id = row[position2]
82
```

```
                    if [transaction_id, user_id] in unique_pairs_list:
                        continue

                    name.lpush(transaction_id, user_id)

                    unique_pairs_list.append([transaction_id, user_id])
    # Fetch data from Excel file
    elif type == 'excel':

        workbook = xlrd.open_workbook(filepath_excel)

        worksheet = workbook.sheet_by_index(query_string)

        unique_pairs_list = []

        if direction == 1:

            for row in range(1, worksheet.nrows):
                user_id = int(worksheet.cell(row, position1).value)
                transaction_id = int(worksheet.cell(row, position2).value)

                if [user_id, transaction_id] in unique_pairs_list:
                    continue

                name.lpush(user_id, transaction_id)

                unique_pairs_list.append([user_id, transaction_id])

        elif direction == 2:

            for row in range(1, worksheet.nrows):
                user_id = int(worksheet.cell(row, position1).value)
                transaction_id = int(worksheet.cell(row, position2).value)

                if [transaction_id, user_id] in unique_pairs_list:
                    continue

                name.lpush(transaction_id, user_id)

                unique_pairs_list.append([transaction_id, user_id])
    # Fetch data from MySQL database
    else:
```

```
125
126         host = 'localhost'
127         port = 3306
128         query_string = 'SELECT user_id, transaction_id FROM transactions'
129
130         try:
131             connection = pymysql.connect(host=host, port=port,
132                                          user=username, passwd=password,
133                                          db=database)
134             # Prepare a cursor object
135             cursor = connection.cursor()
136             # Execute SQL query
137             cursor.execute(query_string)
138             # Fetch rows
139             result = cursor.fetchall()
140
141             unique_pairs_list = []
142
143             if direction == 1:
144
145                 for item in result:
146                     user_id = int(item[0])
147                     transaction_id = int(item[1])
148
149                     if [user_id, transaction_id] in unique_pairs_list:
150                         continue
151
152                     name.lpush(user_id, transaction_id)
153                     unique_pairs_list.append([user_id, transaction_id])
154
155             elif direction == 2:
156
157                 for item in result:
158                     user_id = int(item[0])
159                     transaction_id = int(item[1])
160
161                     if [transaction_id, user_id] in unique_pairs_list:
162                         continue
163
164                     name.lpush(transaction_id, user_id)
165                     unique_pairs_list.append([transaction_id, user_id])
166
```

```
167         except Exception as e:
168             print(e)
169 
170         finally:
171             connection.close()
172 
173 # Function 2
174 def Filter_KLStore(name1, expression):
175     """
176     This function applies an expression on all elements of all lists of
177     a KL store in Redis. If the return value is true, the element remains
178     in the list, otherwise it is removed.
179     The particular element is mentioned as 'element' in the expression.
180 
181     :param name1: A KL store in Redis
182     :param expression: A string representing a valid boolean expression
183     """
184     to_delete_list = []
185 
186     for key in name1.keys():
187         key = key.decode('utf-8')
188 
189         for i in range(0, name1.llen(key)):
190             value = name1.lindex(key, i).decode('utf-8')
191 
192             if eval(expression.replace('element', value)):
193                 continue
194 
195             to_delete_list.append([key, value])
196 
197     for set in to_delete_list:
198         name1.lrem(set[0], 0, set[1])
199 
200 # Function 3
201 def func(value):
202     """
203     This function appends the character '3' to the end of a String.
204 
205     :param value: The String to modify
206     :return: The modified String
207     """
208     value = value + '3'
```

```
209     return value
210 
211 def Apply_KLStore(name1, func):
212     """
213     This function performs an element-wise operation using a Python
214     function.
215 
216     :param name1: The name of the KL store in Redis
217     :param func: A Python function that transforms the lists of
218         the KL store
219     """
220     for key in name1.keys():
221         for i in range(name1.llen(key)):
222             value = name1.lrange(key, i, i)
223             # Remove 'b' character in front of a string literal
224             value = value[0].decode('utf-8').split()
225 
226             value[0] = str(value[0])
227 
228             newValue = func(value[0])
229             # Update value in position i with new value
230             name1.lset(key, i, newValue)
231 
232 # Function 4
233 def Aggr_KLStore(name1, aggr):
234     """
235     This function aggregates each list of a KL store according to
236     the specified aggregate, updating the list with just one item,
237     the result of the aggregation.
238 
239     :param name1: A KL store in Redis
240     :param aggr: A string that can have one of the values
241         'avg/sum/count/min/max'
242     """
243     for key in name1.keys():
244         values = []
245 
246         key = key.decode('utf-8')
247 
248         for i in range(0, name1.llen(key)):
249             values.append(int(name1.lindex(key, i).decode('utf-8')))
250 
```

```
        if aggr == 'avg':
            result = round(sum(values)/len(values), 3)
        elif aggr == 'sum':
            result = sum(values)
        elif aggr == 'count':
            result = len(values)
        elif aggr == 'min':
            result = min(values)
        else:
            result = max(values)

        name1.ltrim(key, 0, 0)

        name1.lset(key, 0, result)

# Function 5
def LookUp_KLStore(name2, name3):
    """
    This function performs a lookup for each element of a list.

    :param name2: The name of the first KL store in Redis
    :param name3: The name of the second KL store in Redis
    """
    for key in name2.keys():
        valueSize = name2.llen(key)

        for i in range(valueSize):
            value_name2 = name2.lrange(key, i, i)

            for j in range(name3.llen(value_name2[0])):
                value_name3 = name3.lrange(value_name2[0], j, j)
                # Remove 'b' character in front of a string literal
                value_name3 = value_name3[0].decode('utf-8').split()

                value_name3[0] = str(value_name3[0])

                name2.rpush(key, value_name3[0])

        for k in range(valueSize):
            name2.lpop(key)

# Function 6
```

```
293 def ProjSel_KLStore(output_name, pnames, expression):
294     """
295     This function performs a join on the common keys of some KL stores,
296     and stores them with their values in a new KL store.
297     Then it applies a filtering condition for this new KL store,
298     based on the key and the contents of the lists involved.
299 
300     :param output_name: The name of the KL store that will be created
301     :param pnames: The names of the KL stores that will be used for the
302         output
303     :param expression: A valid Python boolean expression with the following
304         convention: Within the expression, key and KL stores involved
305         should be prepended by some special symbol(s),
306         e.g. "##key <> 't22' and ##age > 20"
307     """
308     keys_pname1 = pnames[0].keys()
309 
310     for key in keys_pname1:
311         counter = 1
312 
313         for i in range(1, len(pnames)):
314             if key in pnames[i].keys():
315                 counter += 1
316 
317         if counter == len(pnames):
318             for i in range(len(pnames)):
319 
320                 for j in range(0, pnames[i].llen(key)):
321                     output_name.lpush(key,
322                         pnames[i].lindex(key, j).decode('utf-8'))
323 
324     expression_list = expression.split()
325 
326     key_values = []
327 
328     for ex in expression_list:
329         ex = ex.replace("'", "")
330         if re.search(r'^\W+', ex):
331             ex_sub = re.sub(r'^\W+', '', ex)
332 
333             if ex_sub == '':
334                 continue
```

```
335
336             key_values.append([ex, ex_sub])
337
338     to_delete_list = []
339
340     for key in output_name.keys():
341         key = key.decode('utf-8')
342
343         for i in range(0, output_name.llen(key)):
344             value = output_name.lindex(key, i).decode('utf-8')
345
346             expression2 = expression
347             expression2 = expression2.replace(key_values[0][0],
348                 "'" + key + "'")
349
350             expression2 = expression2.replace(key_values[1][0], value)
351
352             if eval(expression2):
353                 continue
354
355             to_delete_list.append([key, value])
356
357     for set in to_delete_list:
358         output_name.lrem(set[0], 0, set[1])
```

3 Function Testing in Python

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

import redis
import redis_functions as rf


def Print_KLStore(current, name):
    print()
    print('----------' + current +' KL Store----------')

    for key in name.keys():
        print(key, name.lrange(key, 0, -1))

def Test_Create_KLStore():
    name1 = redis.Redis(host='localhost', port=6379, db=0)
    name1.flushdb()

    data_source = 'data-sources/Redis_Data_Source.xml'
    query_string = ''
    position1 = 0
    position2 = 1
    direction = 1

    rf.Create_KLStore(name1, data_source, query_string,
        position1, position2, direction)

    current = 'Initial'
    Print_KLStore(current, name1)

    return name1

def Test_Filter_KLStore():
    name1 = Test_Create_KLStore()

    expression = 'element == 87654321'

    rf.Filter_KLStore(name1, expression)

    current = 'Final'
```

```
41      Print_KLStore(current, name1)
42
43  def Test_Apply_KLStore():
44      name1 = Test_Create_KLStore()
45
46      rf.Apply_KLStore(name1, rf.func)
47
48      current = 'Final'
49      Print_KLStore(current, name1)
50
51  def Test_Aggr_KLStore():
52      aggregates = ['avg', 'sum', 'count', 'min', 'max']
53
54      for aggr in aggregates:
55          print()
56          print('Aggregation: ' + aggr)
57          name1 = Test_Create_KLStore()
58
59          rf.Aggr_KLStore(name1, aggr)
60
61          current = 'Final'
62          Print_KLStore(current, name1)
63
64  def Test_LookUp_KLStore():
65      name2 = redis.Redis(host='localhost', port=6379, db=1)
66      name3 = redis.Redis(host='localhost', port=6379, db=2)
67
68      name2.flushdb()
69      name3.flushdb()
70
71      name2.lpush(12, 't12', 't13')
72      name2.lpush(18, 't32')
73      name2.lpush(56, 't82', 't62', 't71')
74
75      name3.lpush('t11', 120, 67, 98)
76      name3.lpush('t12', 68, 139, 65)
77      name3.lpush('t13', 55, 12)
78      name3.lpush('t32', 22, 13, 2, 6)
79      name3.lpush('t71', 4)
80      name3.lpush('t82', 100, 0)
81      name3.lpush('t99', 45, 89)
82
```

```
83      current = 'Initial'
84      Print_KLStore(current, name2)
85
86      current = 'Initial'
87      Print_KLStore(current, name3)
88
89      rf.LookUp_KLStore(name2, name3)
90
91      current = 'Final'
92      Print_KLStore(current, name2)
93
94  def Test_ProjSel_KLStore():
95      pname1 = redis.Redis(host='localhost', port=6379, db=3)
96      pname2 = redis.Redis(host='localhost', port=6379, db=4)
97      pname3 = redis.Redis(host='localhost', port=6379, db=5)
98      output_name = redis.Redis(host='localhost', port=6379, db=6)
99
100     pname1.flushdb()
101     pname2.flushdb()
102     pname3.flushdb()
103     output_name.flushdb()
104
105     pname1.lpush('t11', 31, 62, 9)
106     pname1.lpush('t15', 75, 91)
107     pname1.lpush('t22', 55, 12, 112)
108     pname1.lpush('t39', 44)
109     pname1.lpush('t44', 42, 98)
110
111     pname2.lpush('t11', 12, 6, 95)
112     pname2.lpush('t15', 128)
113     pname2.lpush('t22', 43)
114     pname2.lpush('t32', 39, 77)
115     pname2.lpush('t44', 129)
116
117     pname3.lpush('t11', 182)
118     pname3.lpush('t16', 7, 9)
119     pname3.lpush('t22', 56, 29)
120     pname3.lpush('t38', 32, 82)
121     pname3.lpush('t44', 66, 121, 22)
122
123     pnames = [pname1, pname2, pname3]
124
```

```
125     current = 'Initial'
126     Print_KLStore(current, pname1)
127 
128     current = 'Initial'
129     Print_KLStore(current, pname2)
130 
131     current = 'Initial'
132     Print_KLStore(current, pname3)
133 
134     expression = "##key != 't22' and ##value > 98"
135 
136     rf.ProjSel_KLStore(output_name, pnames, expression)
137 
138     current = 'Final'
139     Print_KLStore(current, output_name)
140 
141 # Run test functions
142 Test_Create_KLStore()
143 Test_Filter_KLStore()
144 Test_Apply_KLStore()
145 Test_Aggr_KLStore()
146 Test_LookUp_KLStore()
147 Test_ProjSel_KLStore()
```

4 Results

4.1 Test_Create_KLStore()

```
D:\GitHub\bdms\redis>python test_redis_functions.py

----------Initial KL Store----------
b'3' [b'87654321', b'34567890', b'23456789']
b'1' [b'12345670']
b'2' [b'98765432']
b'5' [b'87654321', b'30124567', b'20134567']
b'0' [b'12345679', b'12345678']
```

4.2 Test_Filter_KLStore()

```
D:\GitHub\bdms\redis>python test_redis_functions.py

----------Initial KL Store----------
b'3' [b'87654321', b'34567890', b'23456789']
b'1' [b'12345670']
b'2' [b'98765432']
b'5' [b'87654321', b'30124567', b'20134567']
b'0' [b'12345679', b'12345678']

----------Final KL Store----------
b'5' [b'87654321']
b'3' [b'87654321']
```

4.3 Test_Apply_KLStore()

```
D:\GitHub\bdms\redis>python test_redis_functions.py

----------Initial KL Store----------
b'3' [b'87654321', b'34567890', b'23456789']
b'1' [b'12345670']
b'2' [b'98765432']
b'5' [b'87654321', b'30124567', b'20134567']
b'0' [b'12345679', b'12345678']

----------Final KL Store----------
b'1' [b'123456703']
b'2' [b'987654323']
b'5' [b'876543213', b'301245673', b'201345673']
b'0' [b'123456793', b'123456783']
b'3' [b'876543213', b'345678903', b'234567893']
```

4.4 Test_Aggr_KLStore()

```
D:\GitHub\bdms\redis>python test_redis_functions.py

Aggregation: avg

----------Initial KL Store----------
b'3' [b'87654321', b'34567890', b'23456789']
b'1' [b'12345670']
b'2' [b'98765432']
b'5' [b'87654321', b'30124567', b'20134567']
b'0' [b'12345679', b'12345678']

----------Final KL Store----------
b'1' [b'12345670.0']
b'2' [b'98765432.0']
b'5' [b'45971151.667']
b'0' [b'12345678.5']
b'3' [b'48559666.667']

Aggregation: sum

----------Initial KL Store----------
b'3' [b'87654321', b'34567890', b'23456789']
b'1' [b'12345670']
b'2' [b'98765432']
b'5' [b'87654321', b'30124567', b'20134567']
b'0' [b'12345679', b'12345678']

----------Final KL Store----------
b'1' [b'12345670']
b'2' [b'98765432']
b'5' [b'137913455']
b'0' [b'24691357']
b'3' [b'145679000']
```

```
Aggregation: count

----------Initial KL Store----------
b'3' [b'87654321', b'34567890', b'23456789']
b'1' [b'12345670']
b'2' [b'98765432']
b'5' [b'87654321', b'30124567', b'20134567']
b'0' [b'12345679', b'12345678']

----------Final KL Store----------
b'1' [b'1']
b'2' [b'1']
b'5' [b'3']
b'0' [b'2']
b'3' [b'3']

Aggregation: min

----------Initial KL Store----------
b'3' [b'87654321', b'34567890', b'23456789']
b'1' [b'12345670']
b'2' [b'98765432']
b'5' [b'87654321', b'30124567', b'20134567']
b'0' [b'12345679', b'12345678']

----------Final KL Store----------
b'1' [b'12345670']
b'2' [b'98765432']
b'5' [b'20134567']
b'0' [b'12345678']
b'3' [b'23456789']

Aggregation: max

----------Initial KL Store----------
b'3' [b'87654321', b'34567890', b'23456789']
b'1' [b'12345670']
b'2' [b'98765432']
b'5' [b'87654321', b'30124567', b'20134567']
b'0' [b'12345679', b'12345678']

----------Final KL Store----------
b'1' [b'12345670']
b'2' [b'98765432']
b'5' [b'87654321']
b'0' [b'12345679']
b'3' [b'87654321']
```

4.5 Test_LookUp_KLStore()

```
D:\GitHub\bdms\redis>python test_redis_functions.py

----------Initial KL Store----------
b'56' [b't71', b't62', b't82']
b'12' [b't13', b't12']
b'18' [b't32']

----------Initial KL Store----------
b't99' [b'89', b'45']
b't82' [b'0', b'100']
b't11' [b'98', b'67', b'120']
b't32' [b'6', b'2', b'13', b'22']
b't71' [b'4']
b't12' [b'65', b'139', b'68']
b't13' [b'12', b'55']

----------Final KL Store----------
b'56' [b'4', b'0', b'100']
b'12' [b'12', b'55', b'65', b'139', b'68']
b'18' [b'6', b'2', b'13', b'22']
```

4.6 Test_ProjSel_KLStore()

```
D:\GitHub\bdms\redis>python test_redis_functions.py

----------Initial KL Store----------
b't11' [b'9', b'62', b'31']
b't22' [b'112', b'12', b'55']
b't39' [b'44']
b't15' [b'91', b'75']
b't44' [b'98', b'42']

----------Initial KL Store----------
b't32' [b'77', b'39']
b't11' [b'95', b'6', b'12']
b't22' [b'43']
b't15' [b'128']
b't44' [b'129']

----------Initial KL Store----------
b't38' [b'82', b'32']
b't11' [b'182']
b't22' [b'29', b'56']
b't16' [b'9', b'7']
b't44' [b'22', b'121', b'66']

----------Final KL Store----------
b't11' [b'182']
b't44' [b'121', b'129']
```