Module: EGT206 Data Management

Practical 8: Hands on Redis

Activity1: Redis Installation

1. Go to XpanAI and launch a terminal.

2. Type the following commands to install redis on your machine:

   apt-get update

   apt-get install redis-server

```
The following NEW packages will be installed:
  libjemalloc1 redis-server redis-tools
0 upgraded, 3 newly installed, 0 to remove and 67 not upgraded.
Need to get 519 kB of archives.
After this operation, 1507 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
```

   [Type 'y' when prompted]

3. To start redis:

   redis-server

```
root@expanai-xpanaiadmin-0:~# redis-server
410:C 15 May 08:19:58.464 # Warning: no config file specified, using the default config. In order to specify a config file
use redis-server /path/to/redis.conf
                _._
           _.-``__ ''-._
      _.-``    `.  `_.  ''-._           Redis 3.0.6 (00000000/0) 64 bit
  .-`` .-```.  ```\/    _.,_ ''-._
 (    '      ,       .-`  | `,    )     Running in standalone mode
 |`-._`-...-` __...-.``-._|'` _.-'|     Port: 6379
 |    `-._   `._    /     _.-'    |     PID: 410
  `-._    `-._  `-./  _.-'    _.-'
 |`-._`-._    `-.__.-'    _.-'_.-'|
 |    `-._`-._        _.-'_.-'    |           http://redis.io
  `-._    `-._`-.__.-'_.-'    _.-'
 |`-._`-._    `-.__.-'    _.-'_.-'|
 |    `-._`-._        _.-'_.-'    |
  `-._    `-._`-.__.-'_.-'    _.-'
      `-._    `-.__.-'    _.-'
          `-._        _.-'
              `-.__.-'

410:M 15 May 08:19:58.465 # Server started, Redis version 3.0.6
410:M 15 May 08:19:58.466 # WARNING you have Transparent Huge Pages (THP) support enabled in your kernel. This will create
latency and memory usage issues with Redis. To fix this issue run the command 'echo never > /sys/kernel/mm/transparent_huge
page/enabled' as root, and add it to your /etc/rc.local in order to retain the setting after a reboot. Redis must be restar
ted after THP is disabled.
410:M 15 May 08:19:58.466 * The server is now ready to accept connections on port 6379
```

4. Start a new terminal and type following command:
   redis-cli
   This will open a redis prompt.

```
127.0.0.1:6379>
```

   In the above prompt, 127.0.0.1 is your machine's IP address and 6379 is the port on which Redis server is running.

5. Now type the following PING command.
   ping

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

   This shows that Redis is successfully installed on your machine.

Activity2: Strings

Redis string is a sequence of bytes. You can store anything up to 512 megabytes in one string.

1. To assign key “name” with a value of “practical8” using set command.

```
SET name practical8
```

2. To retrieve value of key “name” using get command.

```
GET name
```

3. To delete an entry using del command.

```
DEL name
```

4. The INCR command parses the string value as an integer, increments it by one, and sets the obtained value as the new value.

```
SET counter 100
INCR counter
```

5. If you do not wish to use an extra line of code to create multiple entries, you can use the extended function mset.

```
MSET name1 john name2 mary name3 peter
```

6. Retrieve the values of several entries at once using mget.

```
MGET name1 name2 name3
```

Exercise

1. Assign following key-value pair.

Key	Value
Module1	English
Module2	Physics
Module3	Programming
Module5	Maths

```
127.0.0.1:6379> MSET Module1 English Module2 Physics Module3 Programming Module5 Maths
OK
```

2. Check if key “Module4” exists or not in the database using EXISTS command. It will return 1 if the key exists, or 0 if the key does not exist.

```
127.0.0.1:6379> EXISTS Module4
(integer) 0
```

3. Check the value type stored at key “Module1” using TYPE command.

```
127.0.0.1:6379> TYPE Module1
string
```

4. Rename the key “Module5” to “Module4” using RENAME command.

```
127.0.0.1:6379> rename Module5 Module4
OK
```

5. Get the first 5 character of key “Module3” using GETRANGE command.

```
127.0.0.1:6379> GETRANGE Module3 0 4
"Progr"
```

Activity3: Key expiration

Key expiration lets you set a timeout for a key, also known as a “time to live”, or “TTL”. When time to live elapses, the key is automatically destroyed.

1. Set a key’s expiration using EXPIRE command.

```
SET key user1
EXPIRE key 5
GET key
```

2. Wait for 5 seconds before you retrieve value of key. The key should have disappeared and return ‘nil’ now.

```
GET key
```

3. You can also create keys with expires. The following example sets a key with the string value user1, having an expire of ten seconds.

```
SET key user1 EX 10
```

Exercise

1. Set a key with string

Key	Value	Expire Time
Module1	English	10 seconds

```
127.0.0.1:6379> SET Module1 English EX 10
OK
```

2. Check the remaining time to live for the key.
(The command returns -2 if the key does not exist.
The command returns -1 if the key exists but has no associated expire.)

```
127.0.0.1:6379> ttl Module1
(integer) 4
127.0.0.1:6379> ttl Module1
(integer) 2
127.0.0.1:6379> ttl Module1
(integer) -2
```

Activity4: Lists

Redis lists are linked lists of string values, sorted by insertion order. You can add elements in Redis lists in the head or the tail of the list.

1. Push two values (100, 101) to head of the list using LPUSH command.

```
LPUSH queuenumber 100
LPUSH queuenumber 101
```

2. Push two values (102, 103) to tail of the list using RPUSH command.

```
RPUSH queuenumber 102
RPUSH queuenumber 103
```

3. Display values within the range (0 to 10) using lrange command. Both the indexes can be negative, telling Redis to start counting from the end: so -1 is the last element, -2 is the penultimate element of the list, and so forth.

```
LRANGE queuenumber 0 10
```

4. Remove the value from head of the list (First In First Out)

```
RPOP queuenumber
```

5. Remove the value from tail of the list (First In Last Out)

```
LPOP queuenumber
```

Exercise

1. Create the following key-value pair.

Key	Value
queue	1,3,5,6,7,8

```
127.0.0.1:6379> RPUSH queue 1
(integer) 1
127.0.0.1:6379> RPUSH queue 3
(integer) 2
127.0.0.1:6379> RPUSH queue 5
(integer) 3
127.0.0.1:6379> RPUSH queue 6
(integer) 4
127.0.0.1:6379> RPUSH queue 7
(integer) 5
127.0.0.1:6379> RPUSH queue 8
(integer) 6
127.0.0.1:6379> LRANGE queue 0 10
1) "1"
2) "3"
3) "5"
4) "6"
5) "7"
6) "8"
```

2. Remove values 1,3 and 8 from the list, so that the list now looks like this.

Key	Value
queue	5,6,7

```
127.0.0.1:6379> LPOP queue
"1"
127.0.0.1:6379> LPOP queue
"3"
127.0.0.1:6379> RPOP queue
"8"
127.0.0.1:6379> LRANGE queue 0 10
1) "5"
2) "6"
3) "7"
```

3. Add values to the list, so that the list now looks like this.

Key	Value
queue	3,4,5,6,7,8,9

```
127.0.0.1:6379> LPUSH queue 4
(integer) 4
127.0.0.1:6379> LPUSH queue 3
(integer) 5
127.0.0.1:6379> RPUSH queue 8
(integer) 6
127.0.0.1:6379> RPUSH queue 9
(integer) 7
127.0.0.1:6379> LRANGE queue 0 10
1) "3"
2) "4"
3) "5"
4) "6"
5) "7"
6) "8"
7) "9"
```

4. Check the length of the list.

```
127.0.0.1:6379> LLEN queue
(integer) 7
```

5. Get the third element from the list.

```
127.0.0.1:6379> LINDEX queue 2
"5"
```

6. Trims the list to contain only 2nd to 5th element.

```
127.0.0.1:6379> LTRIM queue 1 4
OK
127.0.0.1:6379> LRANGE queue 0 10
1) "4"
2) "5"
3) "6"
4) "7"
```

Activity5: Sets

Redis Sets are an unordered collection of unique strings. Sets do not allow repetition of data in a key.

1. Add new members to a set using SADD command.

```
SADD myset 1 2 3
```

2. Remove the specified member from the set using SREM command.

```
SREM myset 2
```

3. Gets all the members in a set using SMEMBERS command.

```
SMEMBERS myset
```

Exercise

1. Create the following key-value pair.

Key	Value
colorset1	blue orange green
colorset2	red purple yellow

```
127.0.0.1:6379> SADD colorset1 blue orange green
(integer) 3
127.0.0.1:6379> SADD colorset2 red purple yellow
(integer) 3
```

2. Test if “red” value belongs to set “colorset1”.

```
127.0.0.1:6379> SISMEMBER colorset1 red
(integer) 0
```

3. Check the size of the set “colorset2”.

```
127.0.0.1:6379> SCARD colorset2
(integer) 3
```

4. Remove value “orange” from the set “colorset1”.

```
127.0.0.1:6379> SREM colorset1 orange
(integer) 1
```

5. Combine “colorset2” and “colorset1” together.

```
127.0.0.1:6379> SUNION colorset1 colorset2
1) "red"
2) "blue"
3) "green"
4) "yellow"
5) "purple"
```

Activity6: Hashes

Hashes are collection of key-value pairs. Every hash can store up to more than 4 billion field-value pairs.

1. Sets the value of one or more fields on a hash using HSET command.

```
HSET user1 username ryan
```

2. Returns the value at a given field using HGET command.

```
HGET user1 username
```

3. Sets the value of one or more fields on a hash using HMSET command.

```
HMSET user2 username nathan birthyear 2005 verified 1
```

4. Returns the values at one on more given fields using HMGET command.

```
HMGET user2 username birthyear verified
```

Exercise

1. Create the following key-value pair.

Key	Value
carspecs	brand: Toyota model: Alphard colour: silver year: 2020

```
127.0.0.1:6379> HMSET carspecs brand Toyota model Alphard colour silver year 2020
OK
```

2. Delete hash field “year:2020”.

```
127.0.0.1:6379> HDEL carspecs year
(integer) 1
```

3. Determine whether hash field “horsepower” exists or not.

```
127.0.0.1:6379> HEXISTS carspecs horsepower
(integer) 0
```

Activity7: Sorted Sets

Redis Sorted Sets are similar to Redis Sets with the unique feature of values stored in a set. The difference is, every member of a Sorted Set is associated with a score, that is used in order to take the sorted set ordered, from the smallest to the greatest score.

1. To add members to a sorted set using ZADD command.

```
ZADD tutorials 1 redis
ZADD tutorials 2 mongodb
ZADD tutorials 3 mysql
```

2. You can also update the member value using ZADD command.

```
ZADD tutorials 4 mysql
```

3. To gets the number of members in the sorted sets using ZCARD command.

```
ZCARD tutorials
```

Exercise

1. Assign following key-value pair.

Key	Value	Score
leaderboard	user1	60
	user2	80
	user3	55
	user4	90
	user5	20

```
127.0.0.1:6379> ZADD leaderboard 60 user1
(integer) 1
127.0.0.1:6379> ZADD leaderboard 80 user2
(integer) 1
127.0.0.1:6379> ZADD leaderboard 55 user3
(integer) 1
127.0.0.1:6379> ZADD leaderboard 90 user4
(integer) 1
127.0.0.1:6379> ZADD leaderboard 20 user5
(integer) 1
```

2. Get the members in sorted set “leaderboard”, ordered by scores in descending order.

```
127.0.0.1:6379> ZREVRANGEBYSCORE leaderboard 100 0 WITHSCORES
 1) "user4"
 2) "90"
 3) "user2"
 4) "80"
 5) "user1"
 6) "60"
 7) "user3"
 8) "55"
 9) "user5"
10) "20"
```

3. How many members in sorted set “leaderboard’ have score between 60 and 90?

```
127.0.0.1:6379> ZCOUNT leaderboard 60 90
(integer) 3
```

4. Increment the score of user3 by 20.

```
127.0.0.1:6379> ZINCRBY leaderboard 20 user3
"75"
```

5. Remove user5 from the sorted set.

```
127.0.0.1:6379> ZREM leaderboard user5
(integer) 1
```

6. Remove users with score lower than 60 from the sorted set.

```
127.0.0.1:6379> ZREMRANGEBYSCORE leaderboard 0 60
(integer) 1
```

7. Get the score of user4 from the sorted set “leaderboard”.

```
127.0.0.1:6379> ZSCORE leaderboard user4
"90"
```