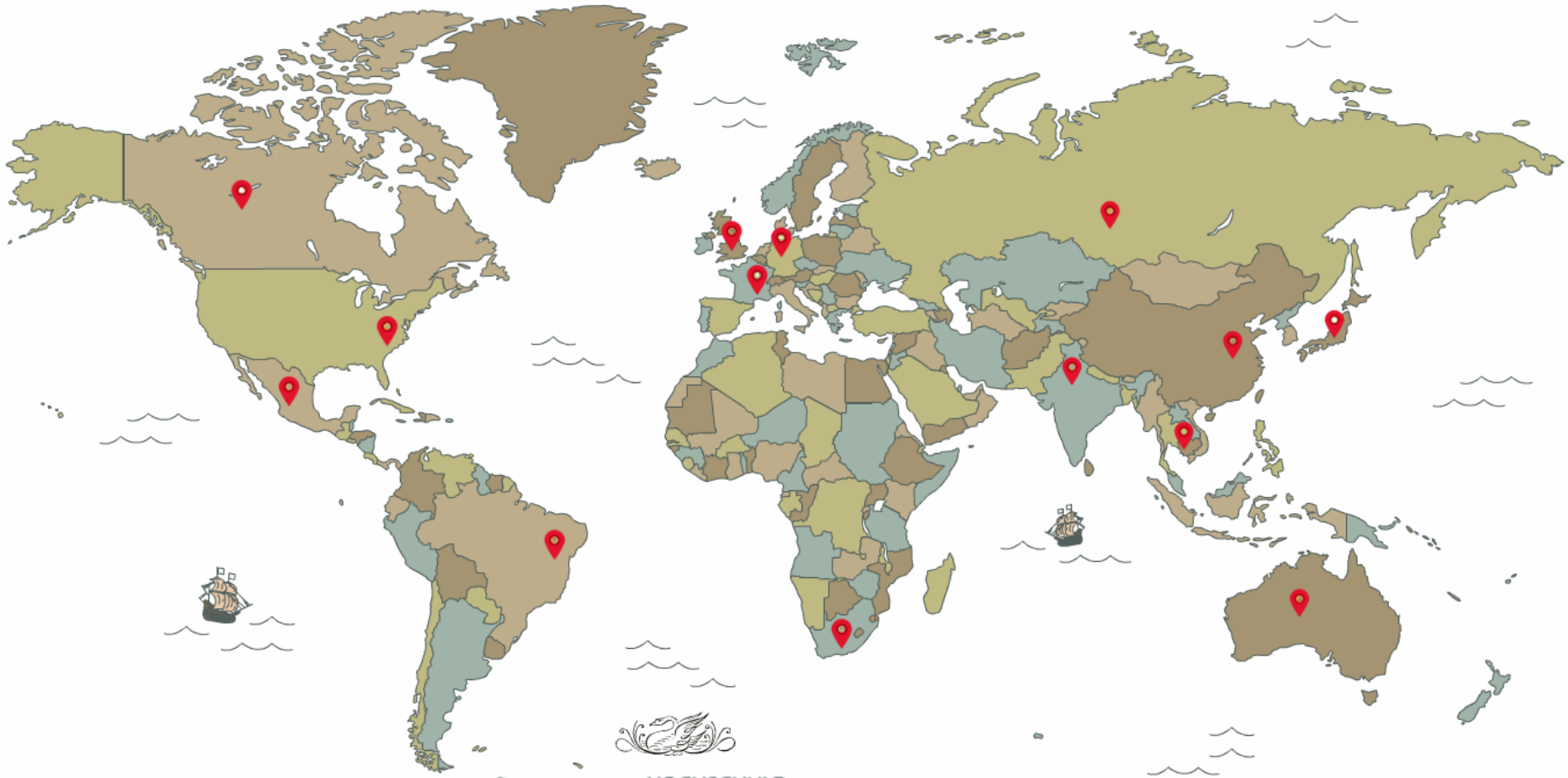
HOCHSCHULE
SRH HEIDELBERG
Intelligence in Learning

Key Value Store Database: Redis

Information Systems
Master Course: Advanced Information Sytems 2016
Lecturer: Prof. Dr. Barbara Sprick, PhD. Ajinkya Prabhune



HOCHSCHULE SRH HEIDELBERG
Intelligence in Learning

Key Value Store Database: Redis

Information Systems
Master Course: Advanced Information Sytems 2016
Lecturer: Prof. Dr. Barbara Sprick, PhD. Ajinkya Prabhune

HOCHSCHULE
SRH HEIDELBERG
Intelligence in Learning

Key Value Store Database: Redis

Information Systems

Master Course: Advanced Information Sytems 2016

Lecturer: Prof. Dr. Barbara Sprick, PhD. Ajinkya Prabhune

Participants

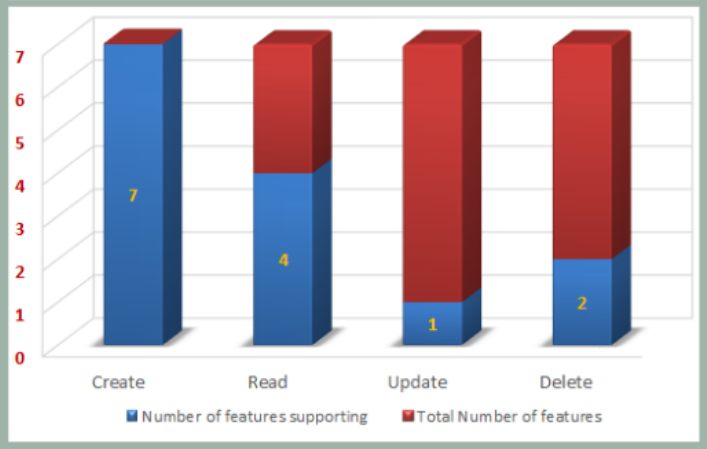
Redis Team
Alpana Chaphalkar (11007146)
Ankit Dixit (11007129)
Jaydeep Galani (11007162)
Mukhtar Baloch (11007170)

Redis Database Introduction

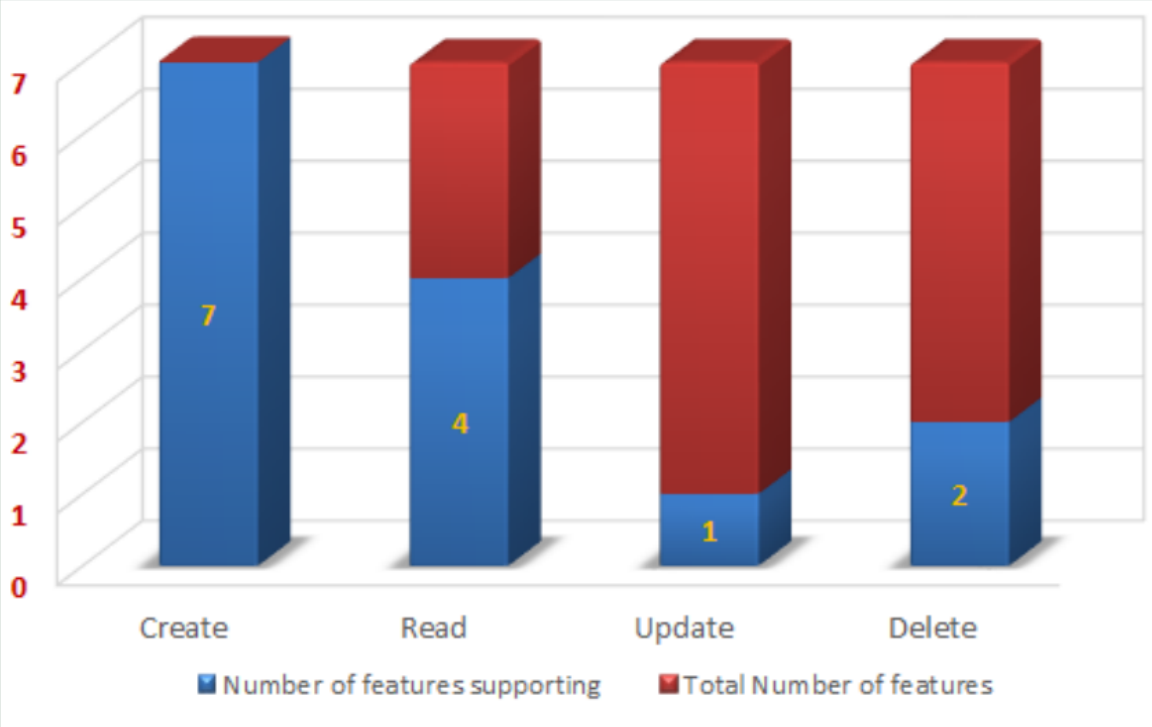
- In Memory Data Store
- Supports Consistency and partition tolerance
- Supports advanced data structures like List, Set, Hash
- Specifically Designed for fast access

Redis Datamodel

Sr. No.	Application Features	Data Structure Names
1	User Login	Users ID
		User Login Profile
2	User Subscription	Subscription List
3	User Favorite List	Favorite List
4	Application Status	Application Status
5	User Search History	User Search History
6	Location Price Trend	Estate Types List
		Price Trend
7	Location Crime Trend	Crime Trend

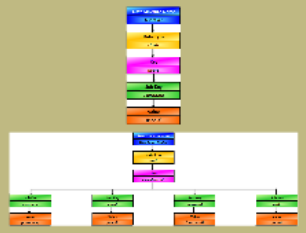


Sr. No.	Application Features	Data Structure Names
1	User Login	Users ID
		User Login Profile
2	User Subscription	Subscription List
3	User Favorite List	Favorite List
4	Application Status	Application Status
5	User Search History	User Search History
6	Location Price Trend	Estate Types List
		Price Trend
7	Location Crime Trend	Crime Trend



User Login

Data Structure



CRUD Operations

Create

> HMSET users (username) (user_id)
> HMSET uses:(user_id) username (username) email (email) password (password) auth (auth)

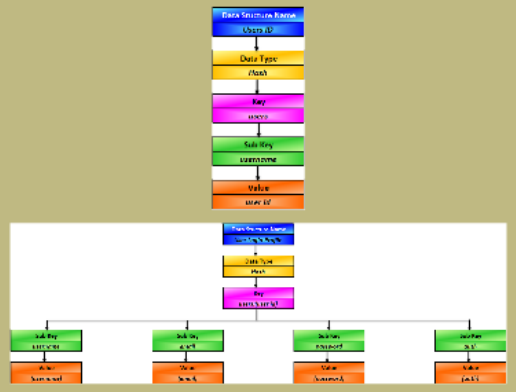
Read

> HGETALL users
> HGETALL users:(user_id)

Update

> HSET users:(user_id) password (password)

Data Structure



CRUD Operations

Create

> HMSET users (username) (user_id)
> HMSET uses:(user_id) username (username) email (email) password (password) auth (auth)

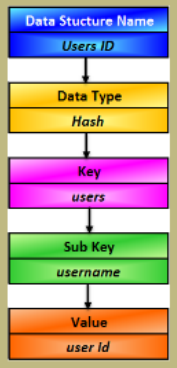
Read

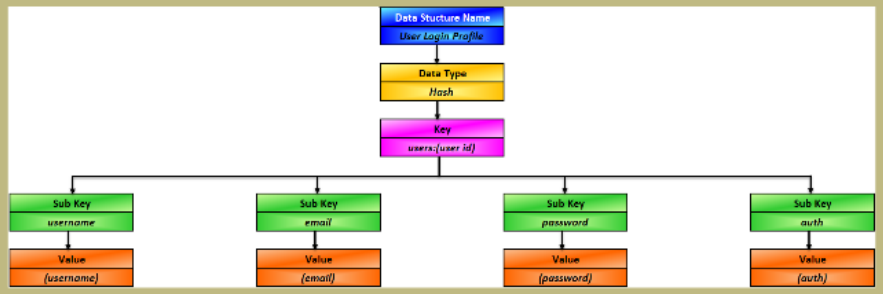
> HGETALL users
> HGETALL users:(user_id)

Update

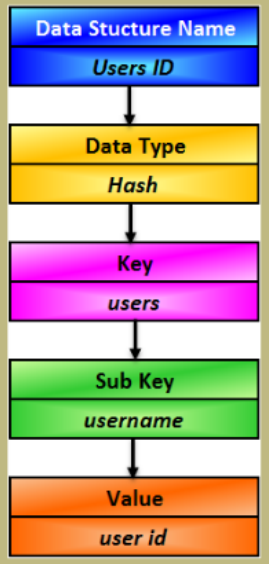
> HSET users:(user_id) password (password)

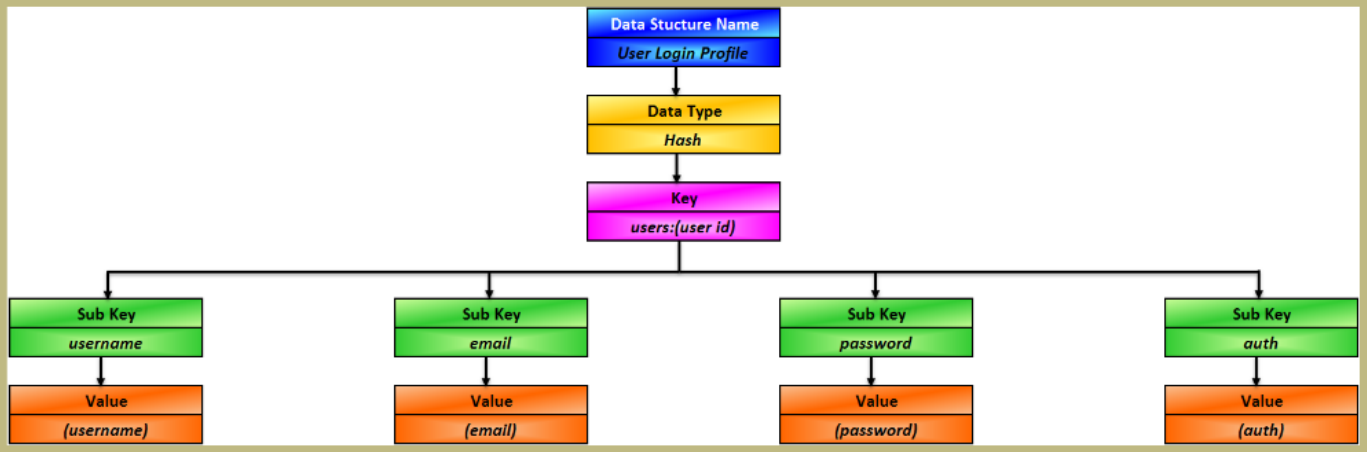
Data Structure





Data Structure





CRUD Operations

Create

> HMSET users (username) (user_id)
> HMSET uses:(user_id) username (username) email (email) password (password) auth (auth)

Read

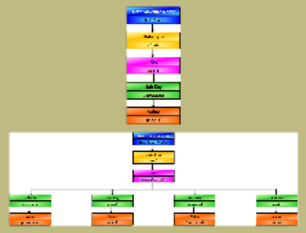
> HGETALL users
> HGETALL users:(user_id)

Update

> HSET users:(user_id) password (password)

User Login

Data Structure



CRUD Operations

Create

> HMSET users (username) (user_id)
> HMSET uses:(user_id) username (username) email (email) password (password) auth (auth)

Read

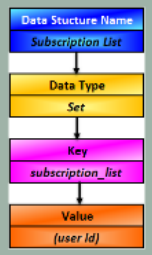
> HGETALL users
> HGETALL users:(user_id)

Update

> HSET users:(user_id) password (password)

Subscription List

Data Structure



CRUD Operations

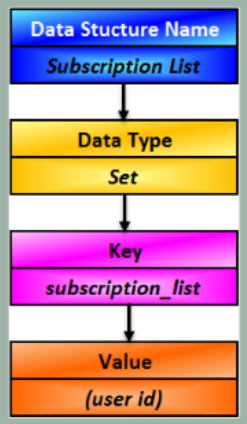
Create
> SADD subscription_list (user_id)

Read
> SMEMBERS subscription_list

Delete
> SREM subscription_list (user_id)

Subscription List

Data Structure



CRUD Operations

Create

> SADD subscription_list (user_id)

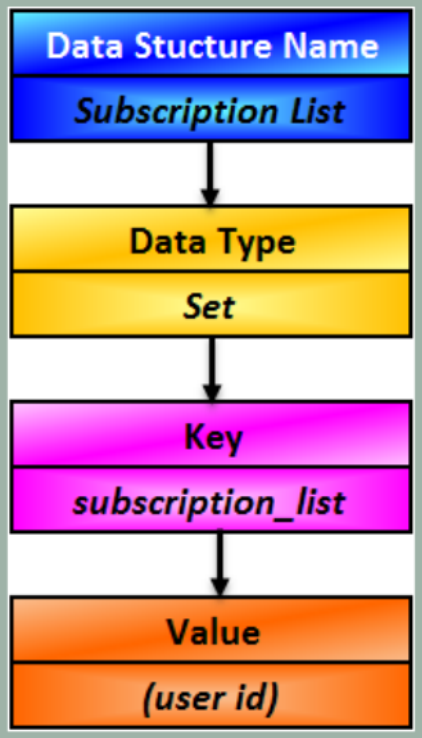
Read

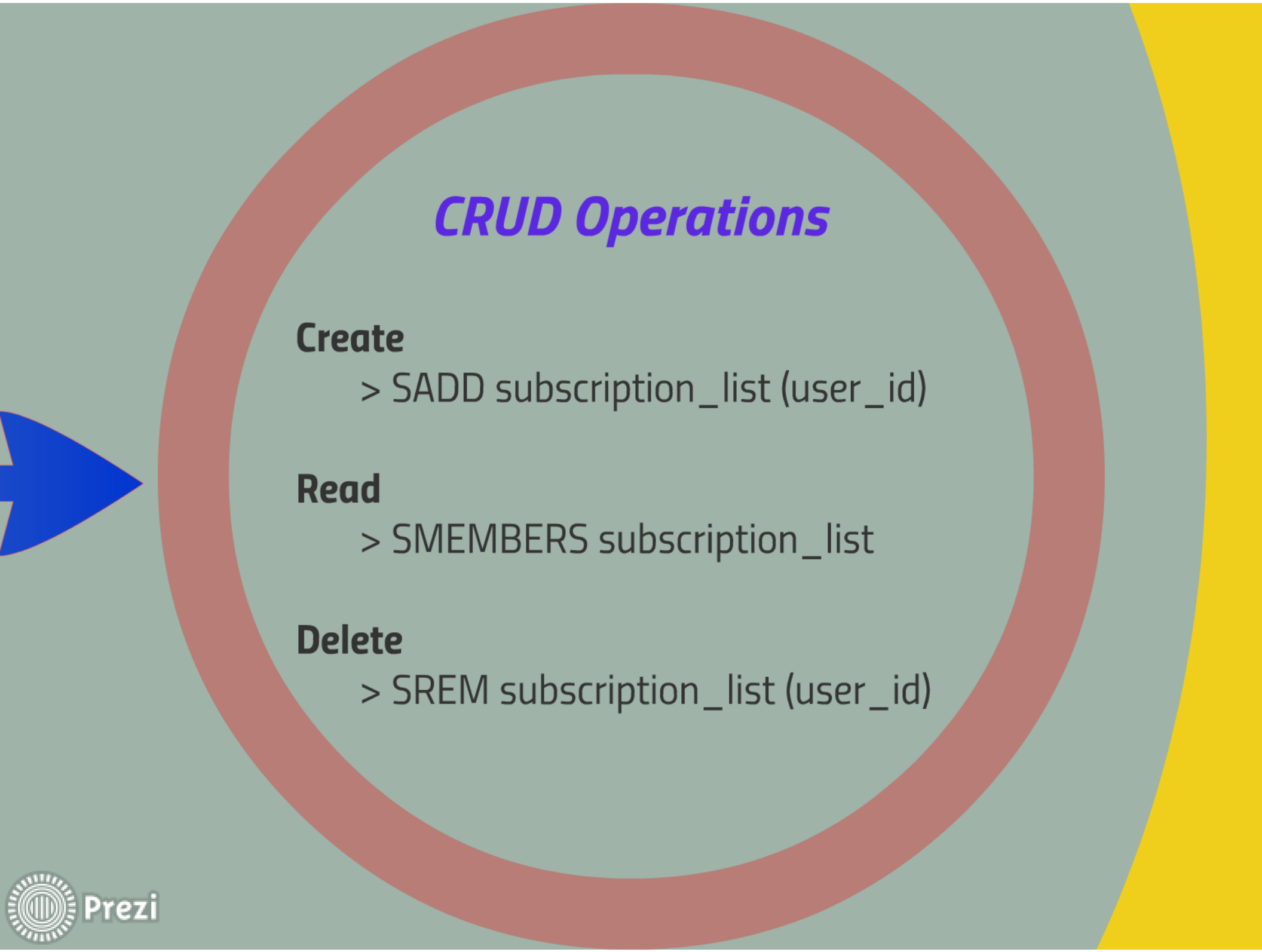
> SMEMBERS subscription_list

Delete

> SREM subscription_list (user_id)

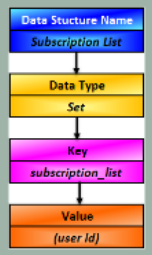
Data Structure




CRUD Operations
Create
> SADD subscription_list (user_id)
Read
> SMEMBERS subscription_list
Delete
> SREM subscription_list (user_id)

Subscription List

Data Structure



CRUD Operations

Create
> SADD subscription_list (user_id)

Read
> SMEMBERS subscription_list

Delete
> SREM subscription_list (user_id)

User Favorite List

Data Structure

CRUD Operations

Create

HMSET favourite_list:(user id):(random number) estate_id (estate id) time_stamp (date:time) estate_latitude (estate latitude) estate_longitude (estate longitude) user_latitude (user latitude) user_longitude (user longitude)

Read

HGETALL favourite_list:(user id):(random number)

Delete

DEL favourite_list:(user id):(random number)

User Favorite List

Data Structure

CRUD Operations

Create

HMSET favourite_list:(user id):(random number) estate_id (estate id) time_stamp (date:time) estate_latitude (estate latitude) estate_longitude (estate longitude) user_latitude (user latitude) user_longitude (user longitude)

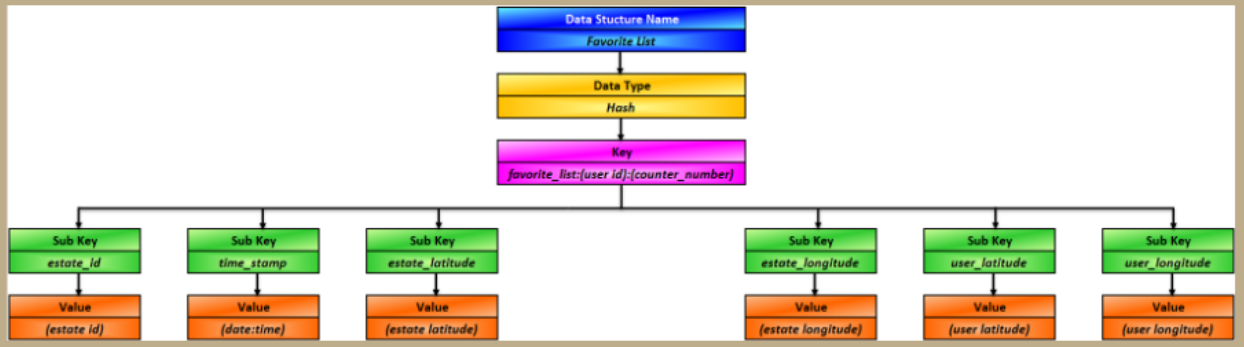
Read

HGETALL favourite_list:(user id):(random number)

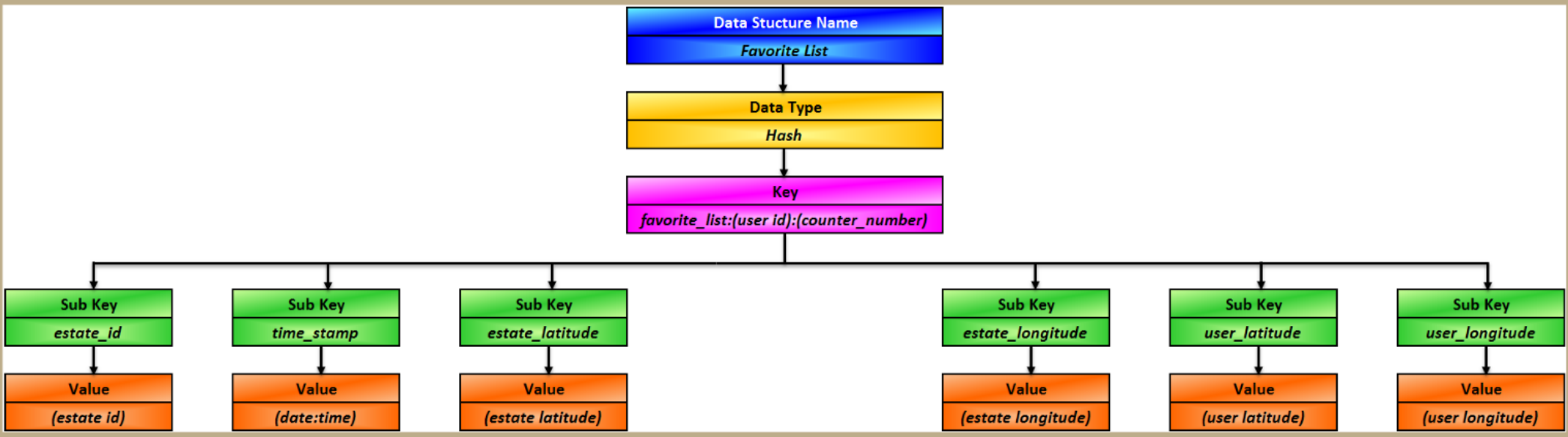
Delete

DEL favourite_list:(user id):(random number)

Data Structure



Data Structure



CRUD Operations

Create

HMSET favourite_list:(user id):(random number) estate_id (estate id) time_stamp (date:time) estate_latitude (estate latitude) estate_longitude (estate longitude) user_latitude (user latitude) user_longitude (user longitude)

Read

HGETALL favourite_list:(user id):(random number)

Delete

DEL favourite_list:(user id):(random number)

User Favorite List

Data Structure

CRUD Operations

Create

HMSET favourite_list:(user id):(random number) estate_id (estate id) time_stamp (date:time) estate_latitude (estate latitude) estate_longitude (estate longitude) user_latitude (user latitude) user_longitude (user longitude)

Read

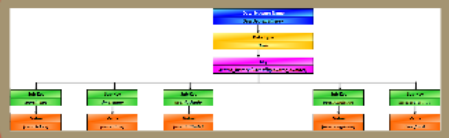
HGETALL favourite_list:(user id):(random number)

Delete

DEL favourite_list:(user id):(random number)

User Search History

Data Structure



CRUD Operations

Create

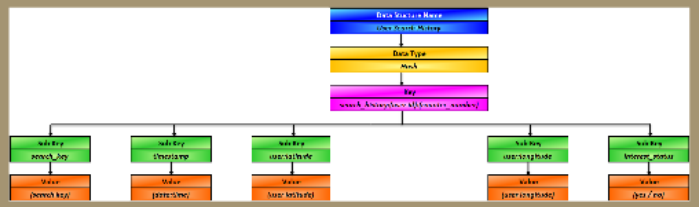
> HMSET search_history:(user_id):(random_number) search_key (search_key) timestamp (timestamp) user_latitude (user_latitude) user_longitude (user_longitude) interest_status (0/1)

Read

> HGETALL search_history:(user_id):(random_number)

User Search History

Data Structure



CRUD Operations

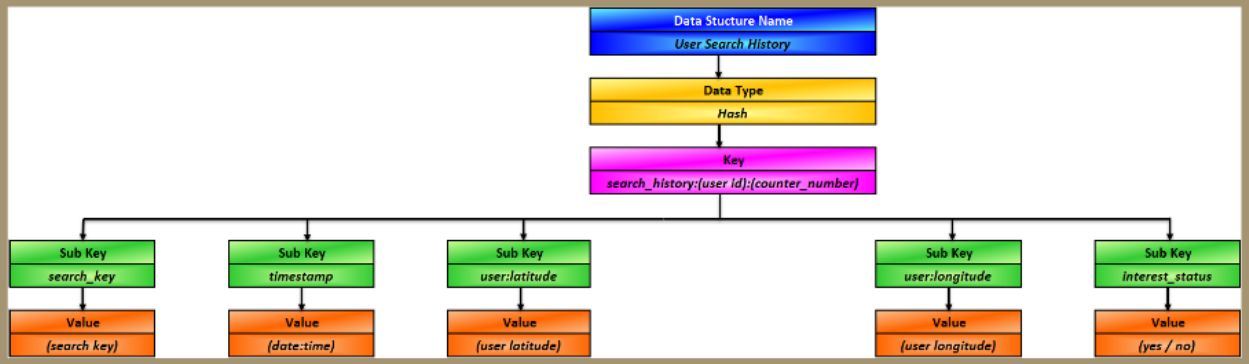
Create

> HMSET search_history:(user_id):(random_number) search_key (search_key) timestamp (timestamp) user_latitude (user_latitude) user_longitude (user_longitude) interest_status (0/1)

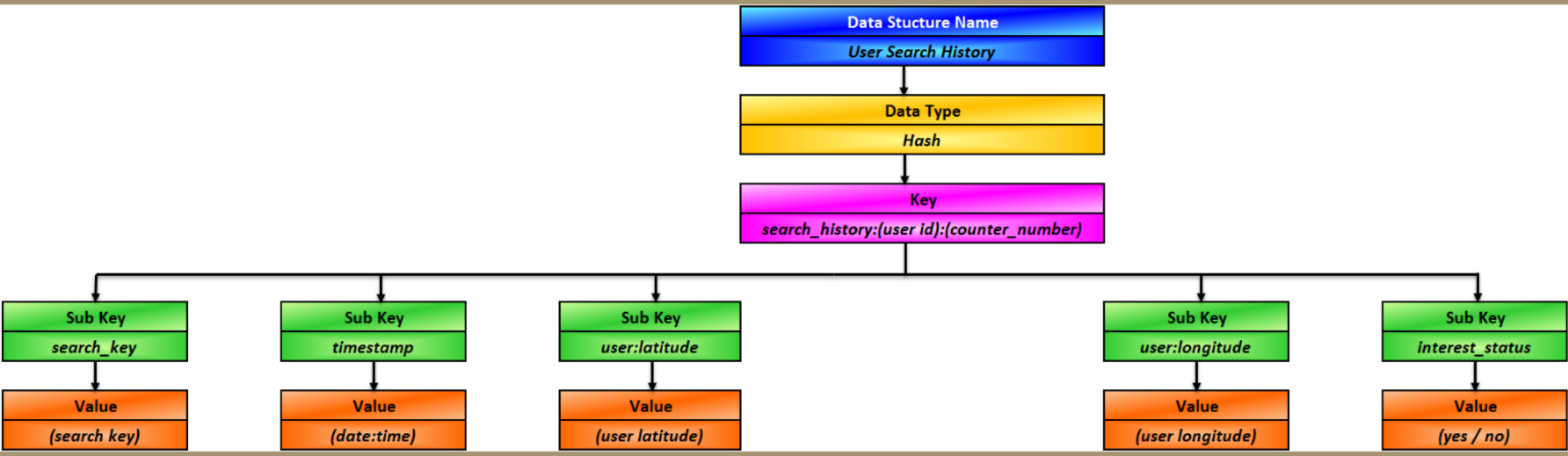
Read

> HGETALL search_history:(user_id):(random_number)

Data Structure



Data Structure



CRUD Operations

Create

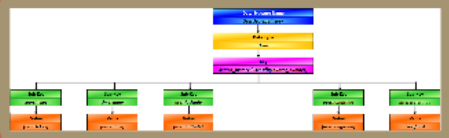
> HMSET search_history:(user_id):(random_number) search_key (search_key) timestamp (timestamp) user_latitude (user_latitude) user_longitude (user_longitude) interest_status (0/1)

Read

> HGETALL search_history:(user_id):(random_number)

User Search History

Data Structure



CRUD Operations

Create

> HMSET search_history:(user_id):(random_number) search_key (search_key) timestamp (timestamp) user_latitude (user_latitude) user_longitude (user_longitude) interest_status (0/1)

Read

> HGETALL search_history:(user_id):(random_number)

Trends

Data Structure

CRUD Operations

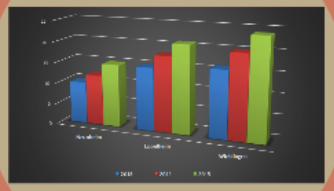
Create

> LPUSH estate_types (estate type)
> HMSET price_trend:(year):(estate_type) (address_id) (price)

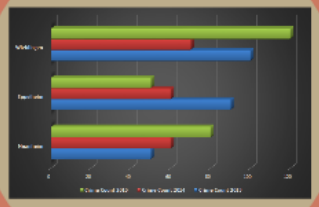
Read

> LRANGE estate_types 0 -1
> HGETALL price_trend:(year):(estate_type)

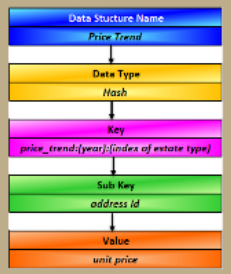
Price Trend

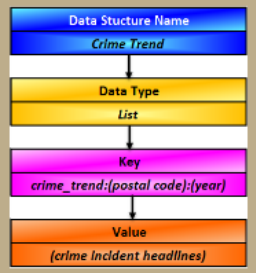


Crime Trend



Data Structure





CRUD Operations

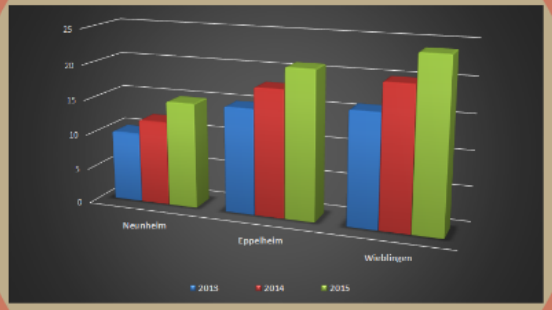
Create

> LPUSH estate_types (estate type)
> HMSET price_trend:(year):(estate_type) (address_id) (price)

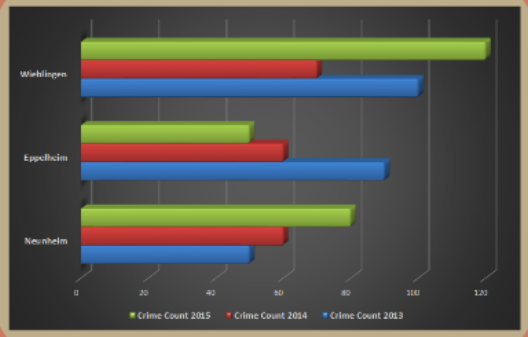
Read

> LRANGE estate_types 0 -1
> HGETALL price_trend:(year):(estate_type)

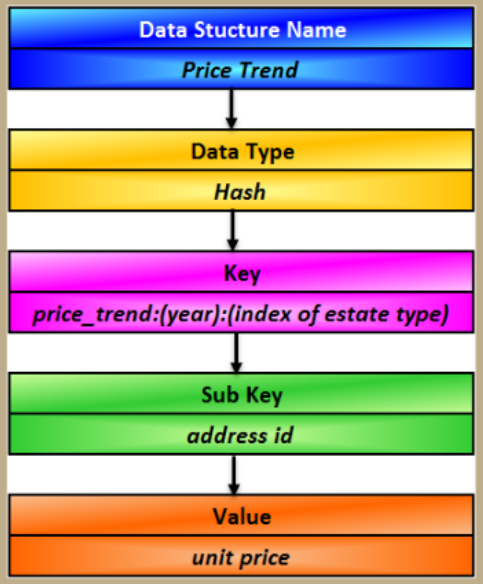
Price Trend

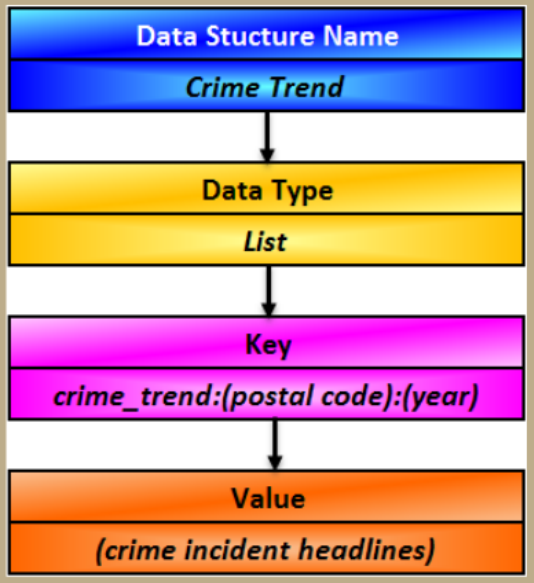


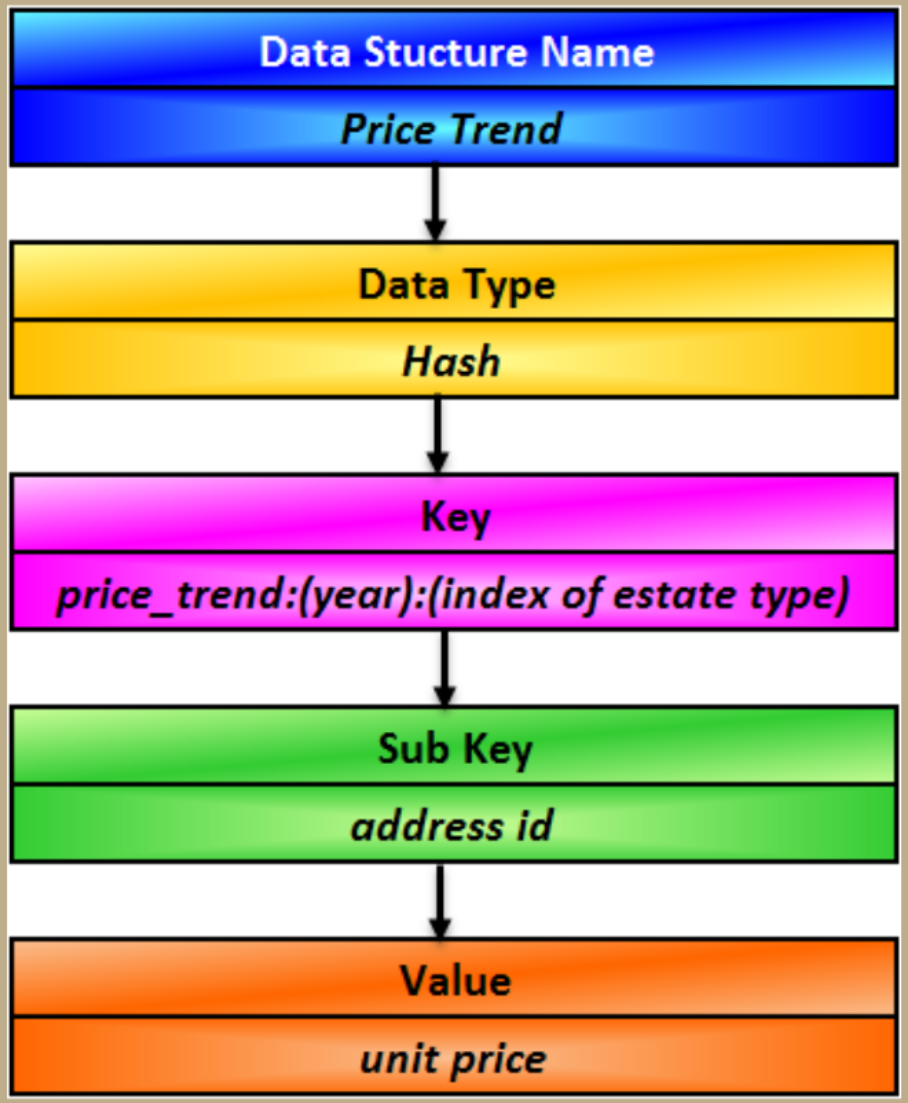
Crime Trend

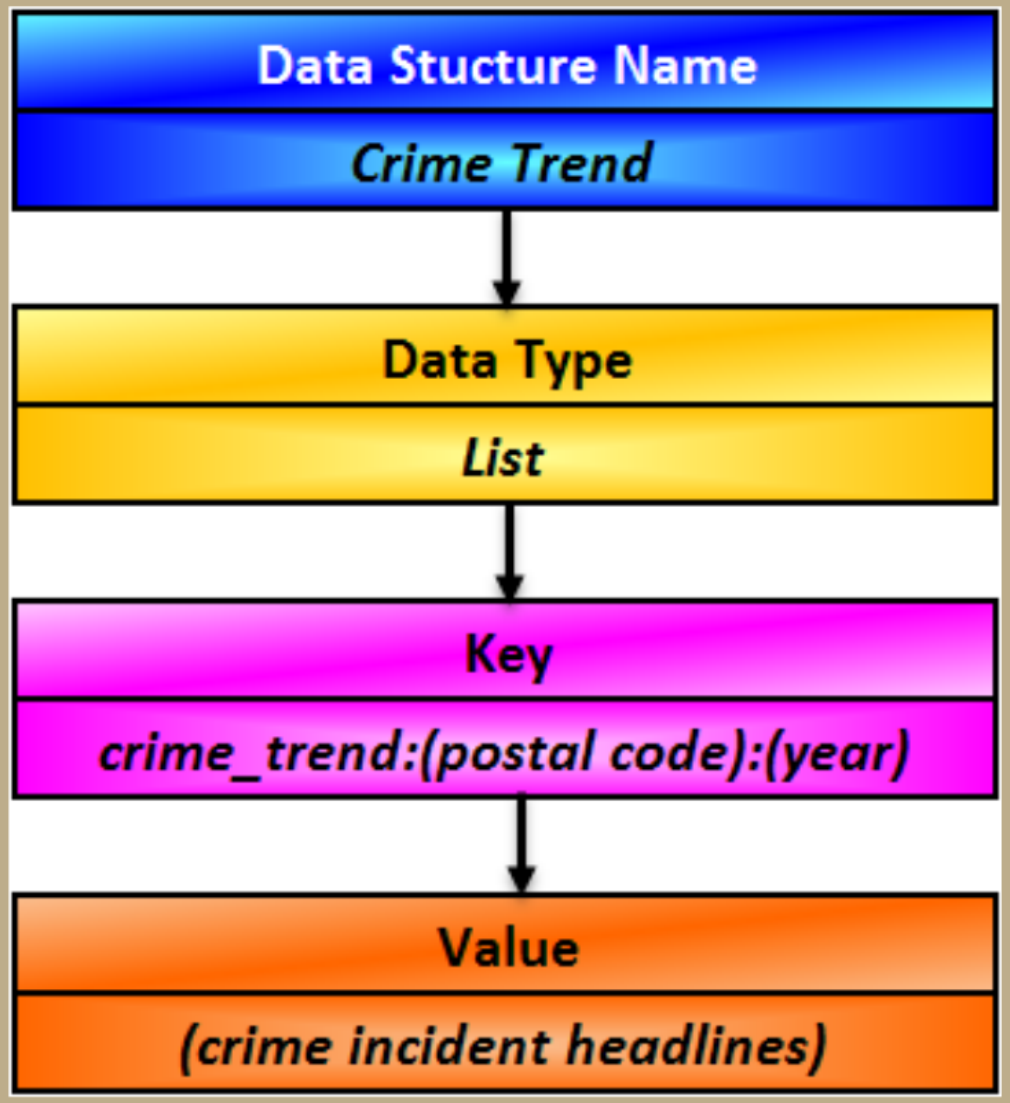


Data Structure






Data Stucture Name
Price Trend
Data Type
Hash
Key
price_trend:(year):(index of estate type)
Sub Key
address id
Value
unit price


Data Stucture Name
Crime Trend
Data Type
List
Key
crime_trend:(postal code):(year)
Value
(crime incident headlines)

CRUD Operations

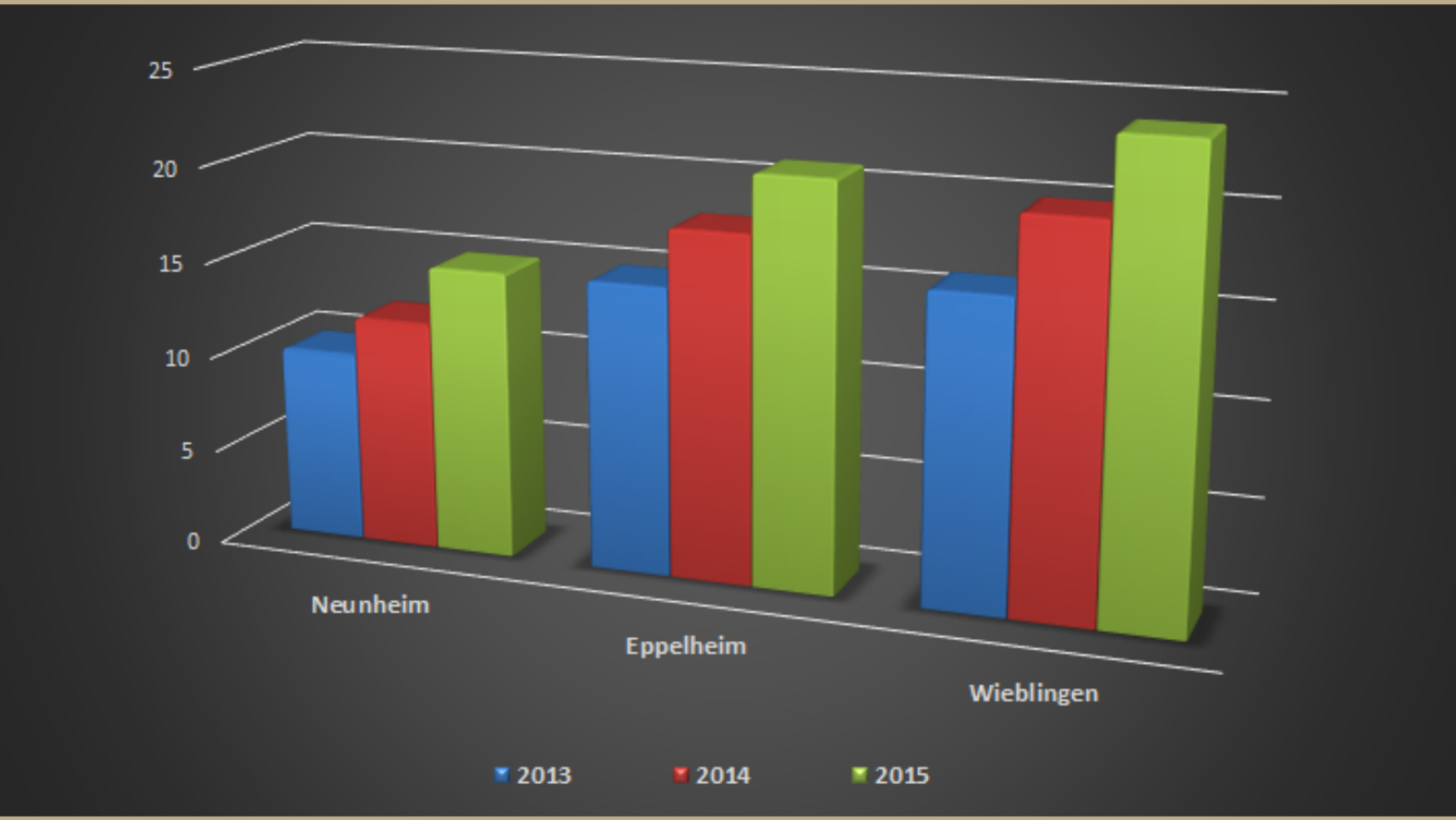
Create

> LPUSH estate_types (estate type)
> HMSET price_trend:(year):(estate_type) (address_id) (price)

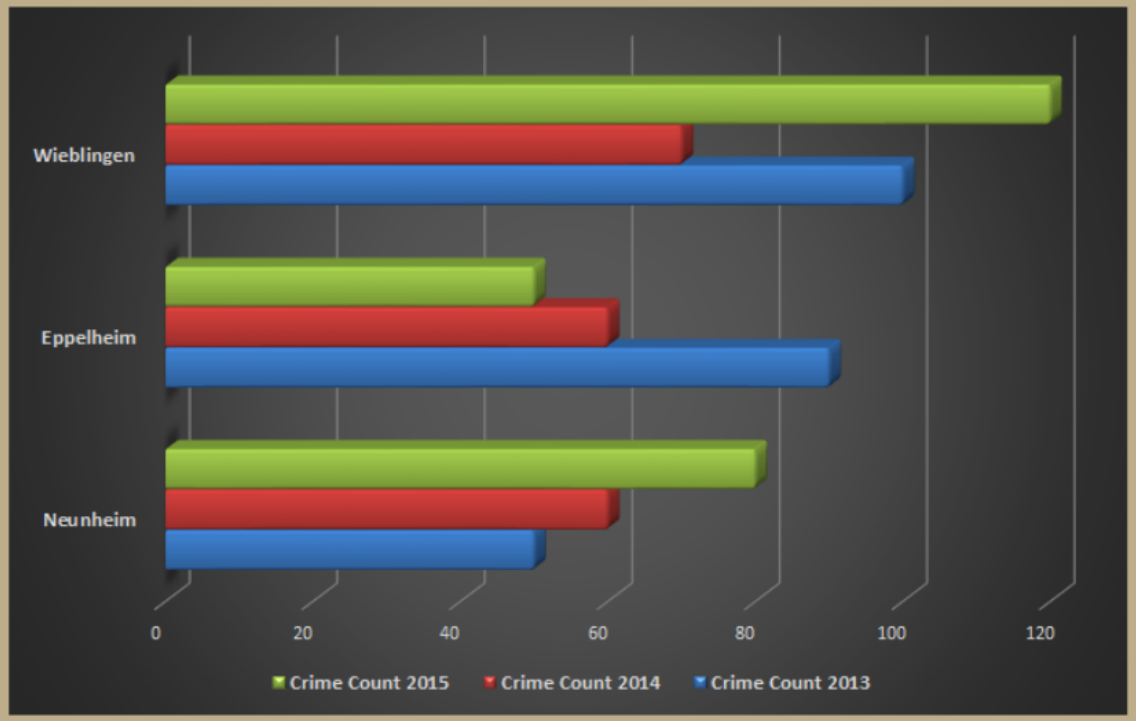
Read

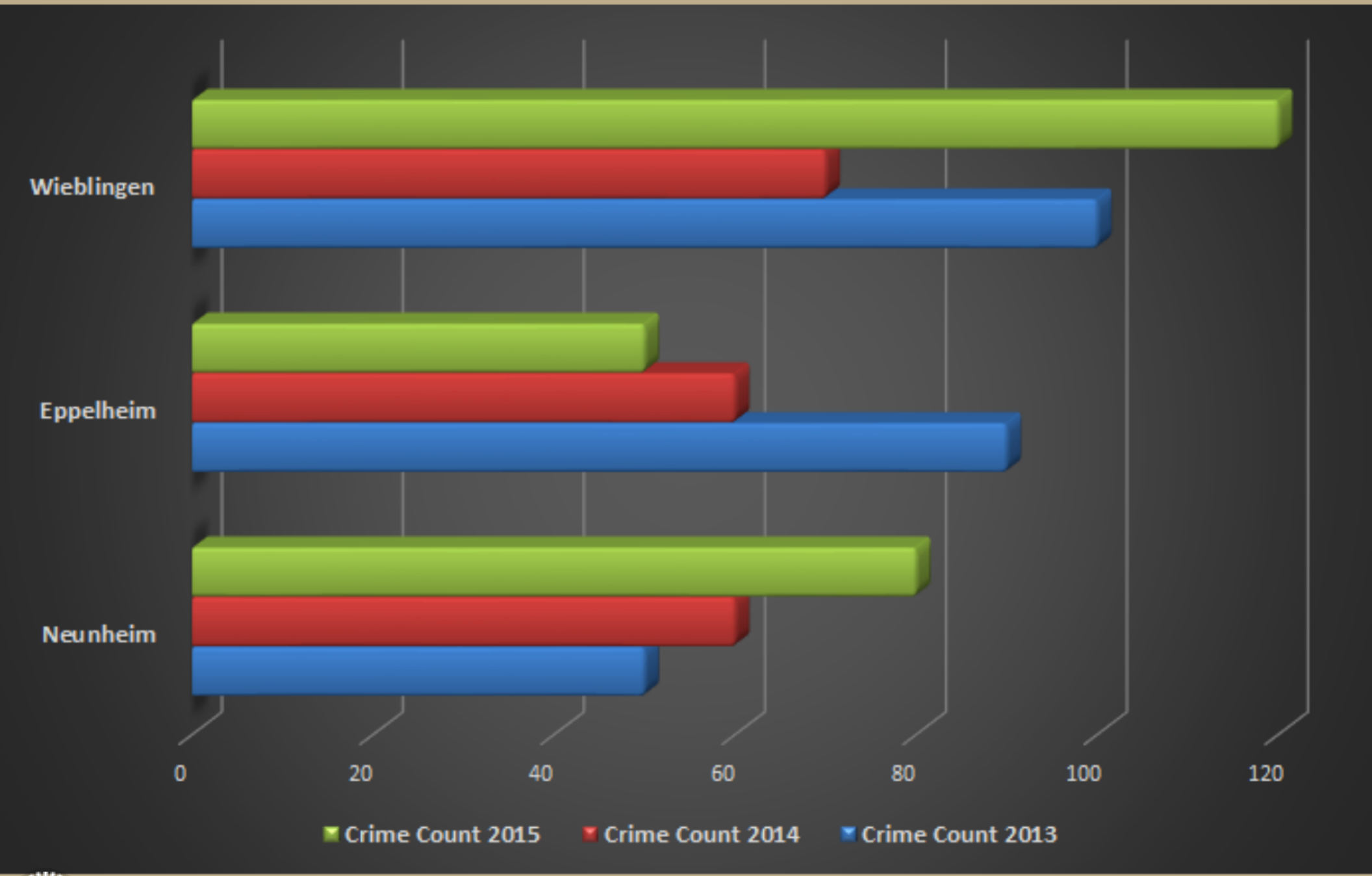
> LRANGE estate_types 0 -1
> HGETALL price_trend:(year):(estate_type)

Price Trend



Crime Trend




Wieblingen
Eppelheim
Neunheim
0
20
40
60
80
100
120
Crime Count 2015
Crime Count 2014
Crime Count 2013

Trends

Data Structure

CRUD Operations

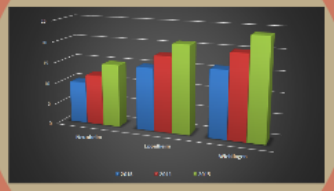
Create

> LPUSH estate_types (estate type)
> HMSET price_trend:(year):(estate_type) (address_id) (price)

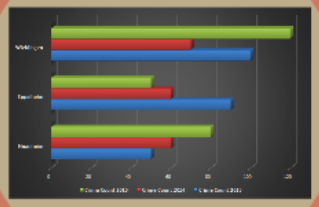
Read

> LRANGE estate_types 0 -1
> HGETALL price_trend:(year):(estate_type)

Price Trend



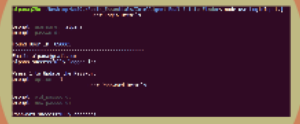
Crime Trend



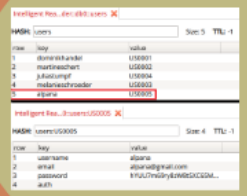
User Login Feature Implementation

Codes in Node.js

Console Output

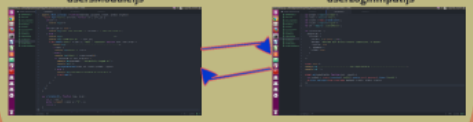


Database Results

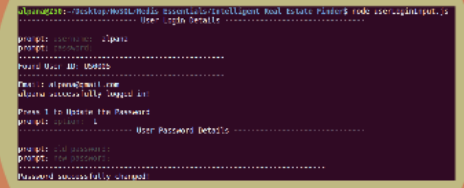


Implementation

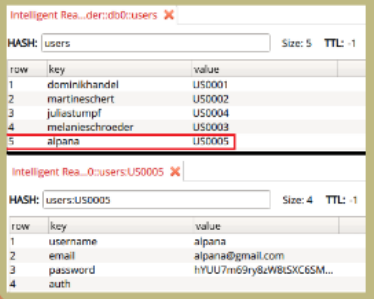
Codes in Node.js



Console Output

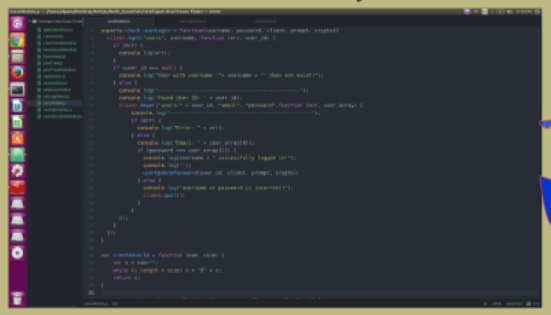


Database Results

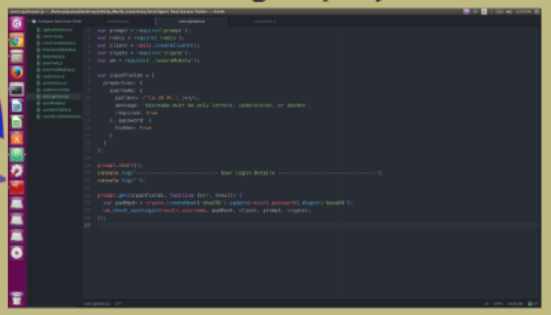


Codes in Node.js

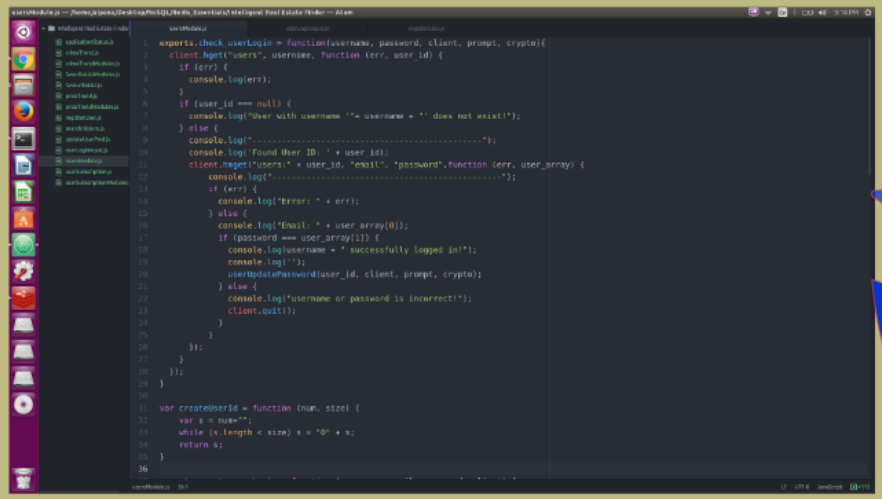
usersModule.js



userLoginInput.js



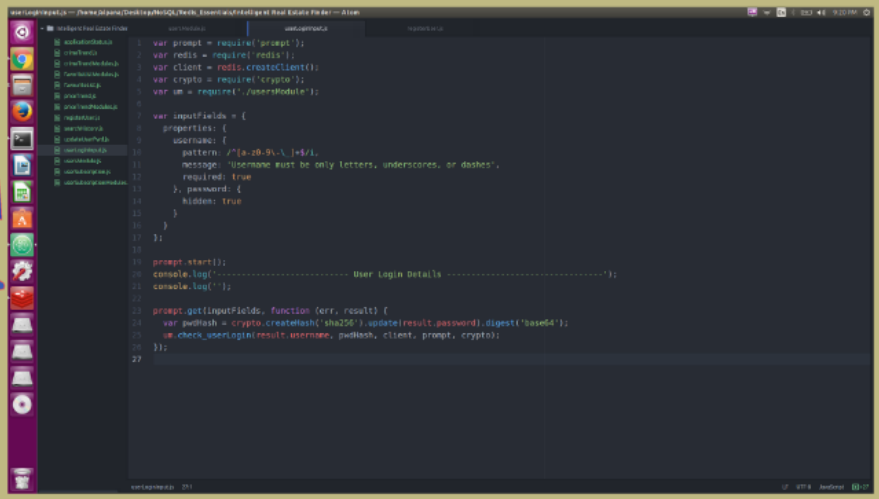
usersModule.js



```
exports.check_userLogin = function(username, password, client, prompt, crypto){
  client.hget("users", username, function (err, user_id) {
    if (err) {
      console.log(err);
    }
    if (user_id === null) {
      console.log("User with username '"+ username + "' does not exist!");
    } else {
      console.log("-----------------------------------------------");
      console.log('Found User ID: ' + user_id);
      client.hmget("users:" + user_id, "email", "password",function (err, user_array) {
          console.log("-----------------------------------------------");
          if (err) {
            console.log("Error: " + err);
          } else {
            console.log("Email: " + user_array[0]);
            if (password === user_array[1]) {
              console.log(username + " successfully logged in!");
              console.log('');
              userUpdatePassword(user_id, client, prompt, crypto);
            } else {
              console.log("username or password is incorrect!");
              client.quit();
            }
          }
      });
    }
  });
}

var createUserId = function (num, size) {
    var s = num+"";
    while (s.length < size) s = "0" + s;
    return s;
}
```

userLoginInput.js



```
var prompt = require('prompt');
var redis = require('redis');
var client = redis.createClient();
var crypto = require('crypto');
var um = require('./usersModule');

var inputFields = {
  properties: {
    username: {
      pattern: /^[a-z0-9\-\_]+$/i,
      message: 'Username must be only letters, underscores, or dashes',
      required: true
    }, password: {
      hidden: true
    }
  }
};

prompt.start();
console.log('------------------------- User Login Details -----------------------------');
console.log('');

prompt.get(inputFields, function (err, result) {
  var pwdHash = crypto.createHash('sha256').update(result.password).digest('base64');
  um.check_userLogin(result.username, pwdHash, client, prompt, crypto);
});
```

Console Output

```
alpana@Z50:~/Desktop/NoSQL/Redis_Essentials/Intelligent Real Estate Finder$ node userLoginInput.js
--------------------------- User Login Details -------------------------------

prompt: username:  alpana
prompt: password:
--------------------------------------------
Found User ID: US0005
--------------------------------------------
Email: alpana@gmail.com
alpana successfully logged in!

Press 1 to Update the Password
prompt: option:  1
------------------------- User Password Details ----------------------------

prompt: old_password:
prompt: new_password:
---------------------------------------------------------------------
Password successfully changed!
```

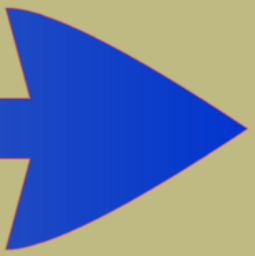
```
alpana@Z50:~/Desktop/NoSQL/Redis_Essentials/Intelligent Real Estate Finder$ node userLoginInput.js
--------------------------- User Login Details -----------------------------

prompt: username:  alpana
prompt: password:
--------------------------------------------
Found User ID: US0005
--------------------------------------------
Email: alpana@gmail.com
alpana successfully logged in!

Press 1 to Update the Password
prompt: option:  1
------------------------- User Password Details ----------------------------

prompt: old_password:
prompt: new_password:
------------------------------------------------------------------
Password successfully changed!
```

Database Results



Intelligent Rea...der::db0::users

HASH: users Size: 5 **TTL:** -1

row	key	value
1	dominikhandel	US0001
2	martineschert	US0002
3	juliastumpf	US0004
4	melanieschroeder	US0003
5	alpana	US0005

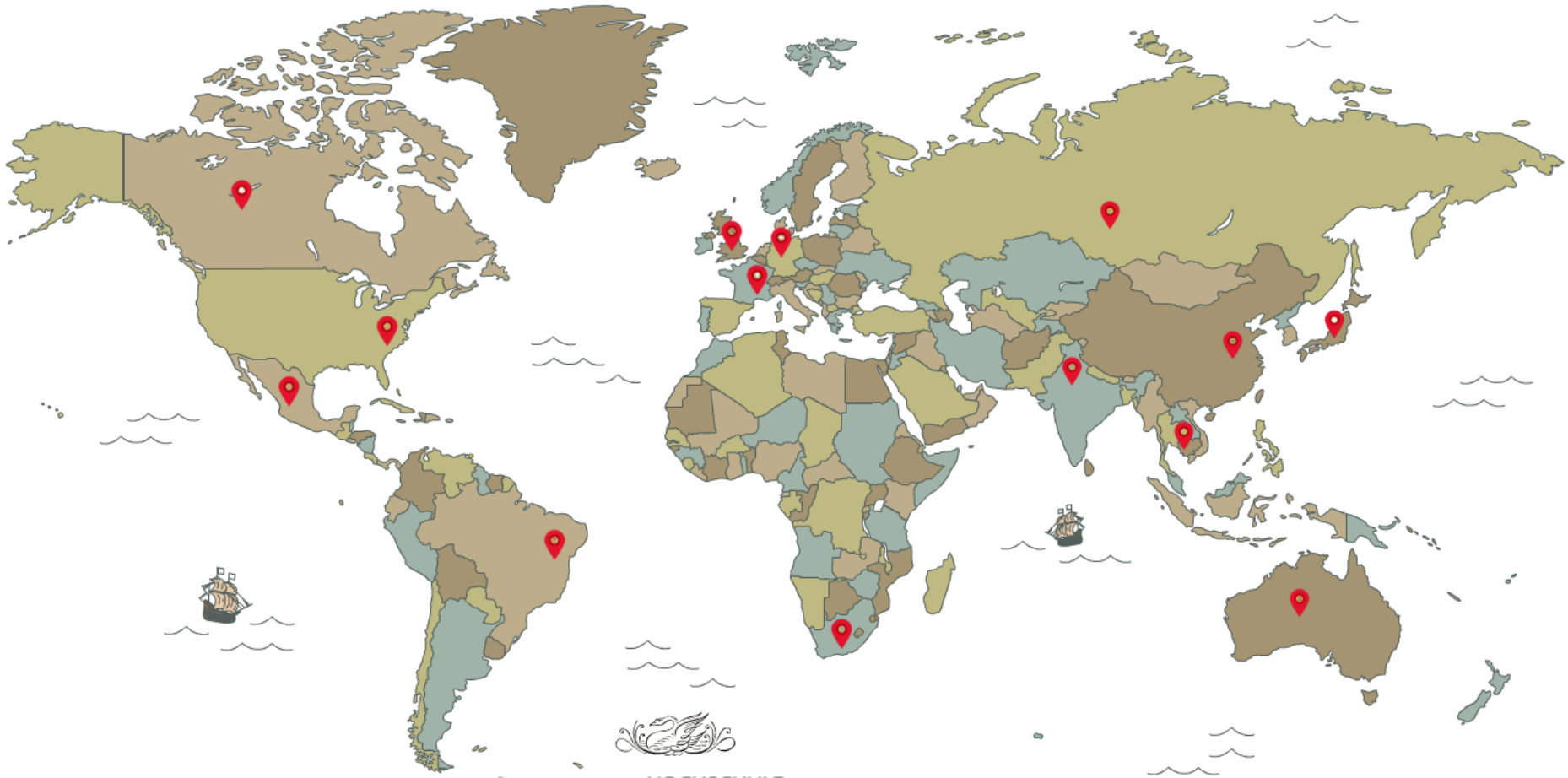
Intelligent Rea...0::users:US0005

HASH: users:US0005 Size: 4 **TTL:** -1

row	key	value
1	username	alpana
2	email	alpana@gmail.com
3	password	hYUU7m69ry8zW8tSXC6SM...
4	auth	

Ideas on extending the application

- Detail Analysis of Price Trend and Crime Trend
- For example, using PUB / SUB functionality of Redis database, to broadcast live updates and other information to all the users who are currently logged in.



HOCHSCHULE SRH HEIDELBERG
Intelligence in Learning

Key Value Store Database: Redis

Information Systems
Master Course: Advanced Information Sytems 2016
Lecturer: Prof. Dr. Barbara Sprick, PhD. Ajinkya Prabhune