# AT82.02

## DATA MODELING AND MANAGEMENT

LAB5: KEY-VALUE STORE [NOSQL]

Co-funded by the
Erasmus+ Programme
of the European Union

DS&AI

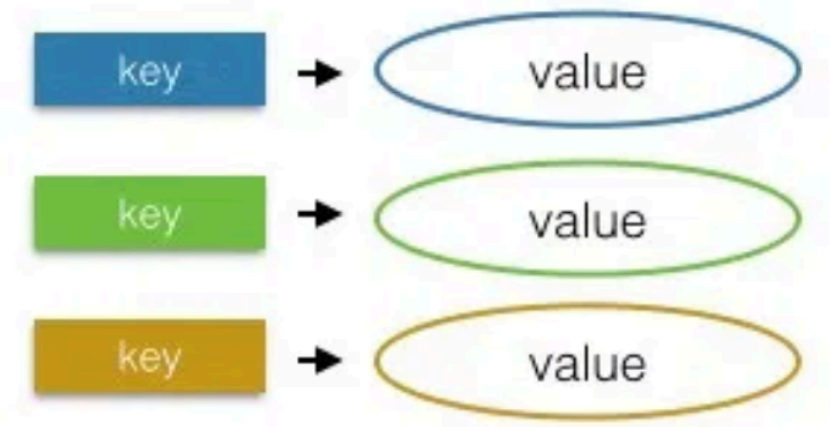
# Outline

- Key-Value Store
- What is Redis ?
- Redis Data Types
- Use Cases: Shopping Cart

# Key-Value Store

**What is key-value stores ?**

- A key-value store is a type of nonrelational database (NoSQL).
- Data is represented as a collection of key–value pairs.
- Collect data as a hash table with a unique key and a pointer to a particular item of data

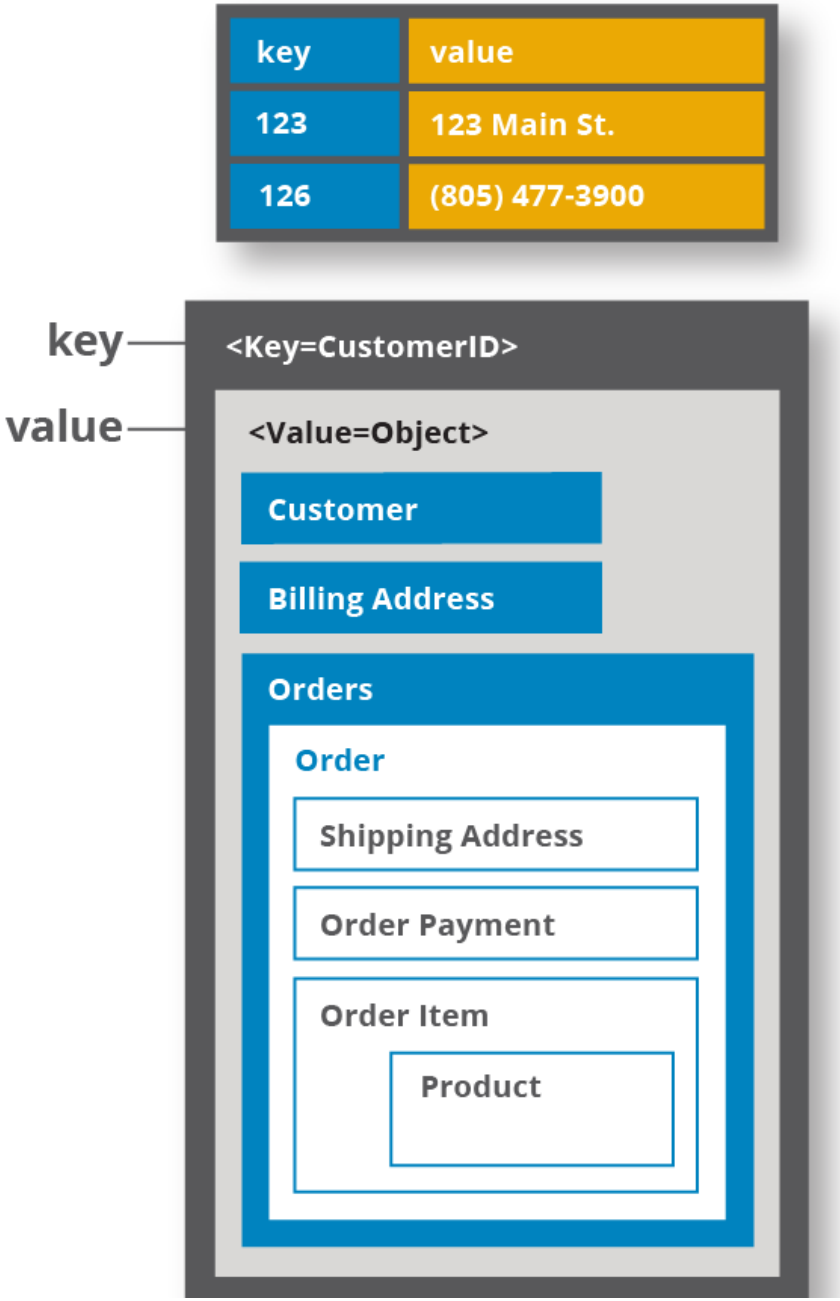



**How do key-value stores work?**

- Key-value stores have no query language.
- Retrieve and update data using simple get, set and delete commands

# Key-Value Store

- The key could be anything, but it need to be the unique identifier for an item of data
- A value is either the data being identified that could be anything including a list or another key-value pair.
- Key-value stores are flexible and offer very fast performance for reads and writes
- Not adequate for complex applications



# Key-Value Store

- Some popular key-value stores are:




Amazon DynamoDB

ORACLE

BERKELEY DB

# What is Redis?


- A Key-Value Store.
- Stores and manipulates all data in memory that can be used as a database, cache, and message broker.
- Supports basic data structures such as strings, hashes, lists, sets, and sorted sets with range queries.
- More advanced data structures like bitmaps, hyperloglogs, and geospatial indexes with radius queries are also supported.

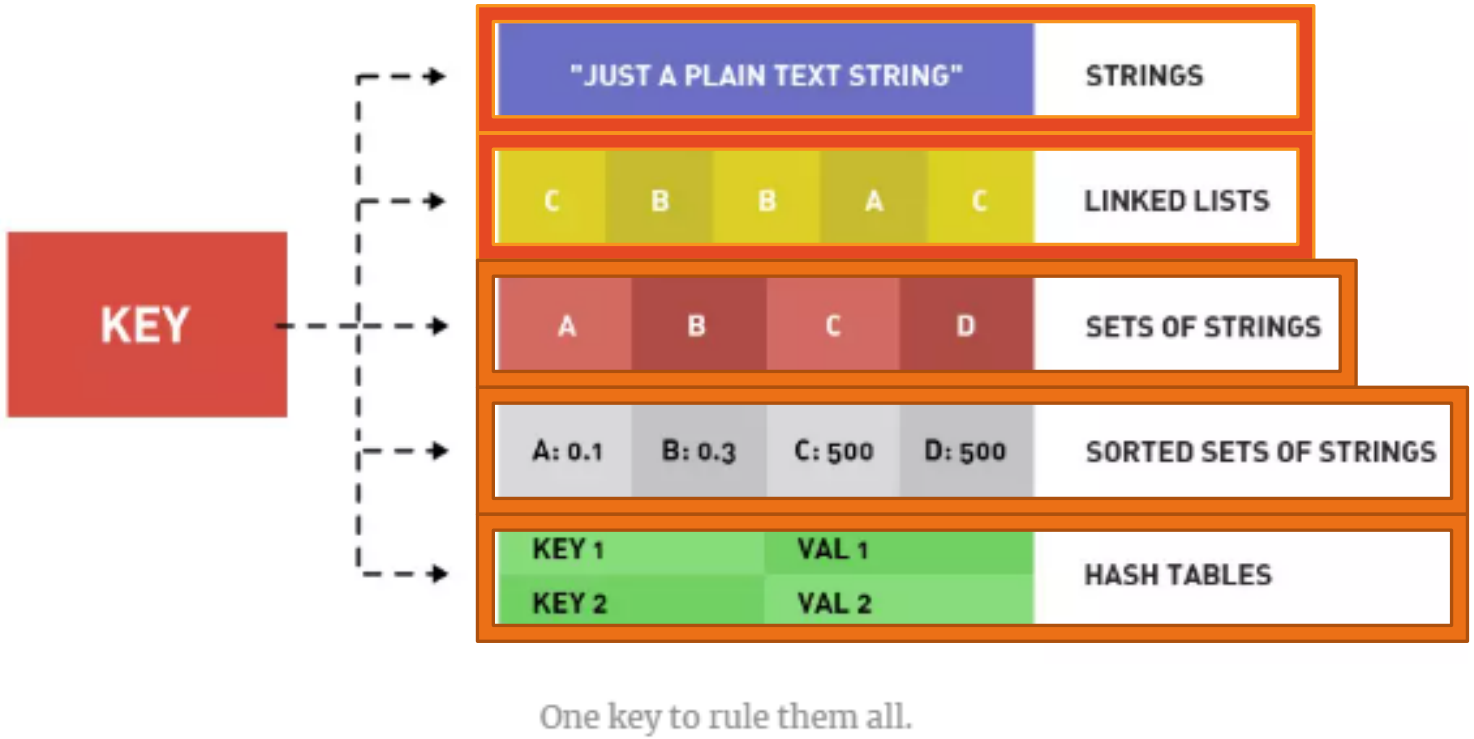
# How to install Redis?

1. Go to https://redis.io/download, download and extract file. For windows, download the stable version at https://github.com/MSOpenTech/redis/releases
2. Start Redis using command:
   $ src/redis-server
3. Check Service:
   $ src/redis-cli ping
   PONG

**Try Redis:**

- Interactive Tutorial: https://try.redis.io/

# Redis Data Types




One key to rule them all.

Redis Command: [https://redis.io/commands](https://redis.io/commands)

**STRING:** Binary-safe string data with max allowed size 512 MB

**LIST:** Lists in Redis are implemented using a linked list. They are collections of string elements, sorted by insertion order.

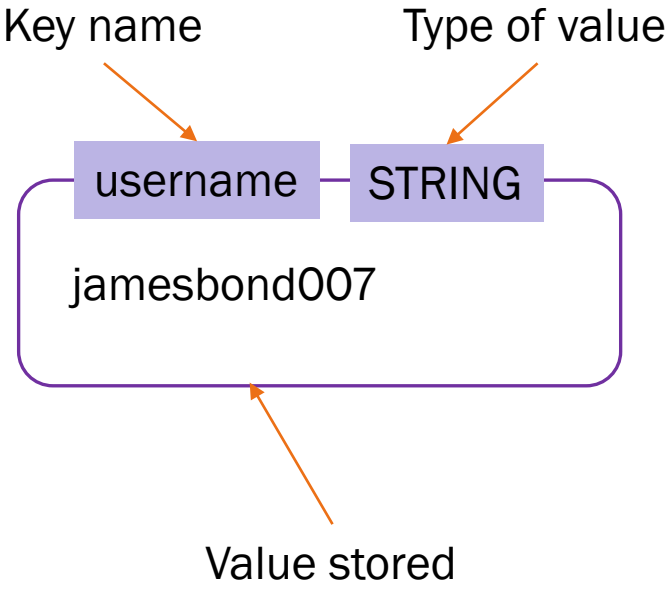
**SET:** A collection of unique strings with no ordering.

**SORTED SET:** A collection of unique strings ordered by user defined scoring

**HASH:** Unordered hash table of keys to values

# Redis STRING

Redis String type is the simply type of value. Integer and float number can be stored as STRING type in Redis.




| Command | Meaning |
|---|---|
| SET | Set the value stored at the given key |
| GET | Retrieves the data stored at the given key |
| DEL | Delete the value stored at the given key (use for all types) |

Note: MSET and MGET commands are used to set or retrieve the value of multiple keys in a single command

# Redis STRING example

> SET username Anna
OK

> GET username
“Anna”

> DEL username
(integer) 1

> MSET a 10 b 20 c 30
OK

> MGET a b c
1) "10“
2) "20“
3) "30"

> SET view_count 10
OK

> INCR view_count
(integer) 11

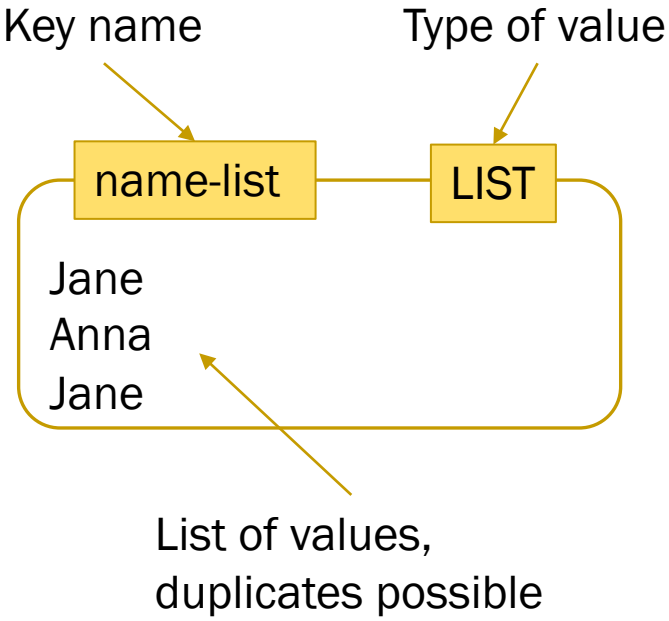
> INCR view_count
(integer) 12

> INCRBY view_count 20
(integer) 32

# Redis LIST

Redis LIST is a sequence of order string, implemented using Linked List

| Command | Meaning |
|---|---|
| RPUSH / LPUSH | Pushes the value onto the right/left end of list |
| RPOP / LPOP | Pops the value from the right/left end of list and return it |
| LRANGE | Retrieves a range of values from the list |
| LINDEX | Retrieves the value at a given position in the list |
| LLEN | Return the length of a list |



# Redis LIST example

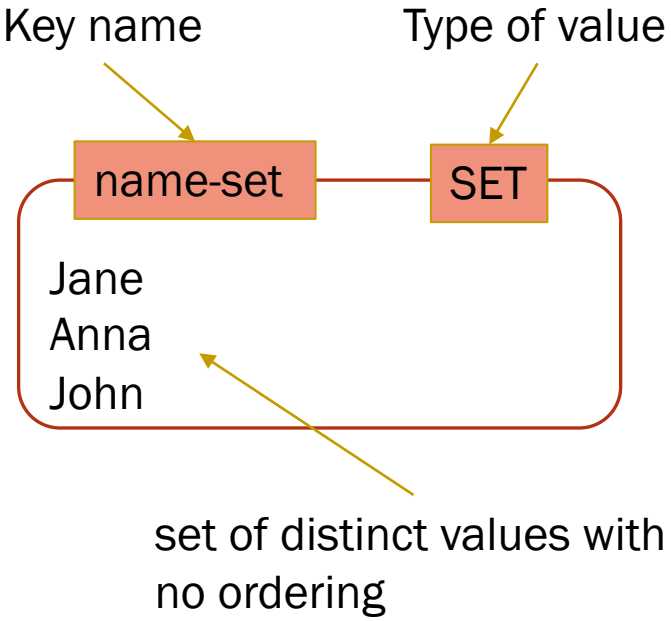
```
> RPUSH name-list Jane
(integer) 1

> RPUSH name-list Anna
(integer) 2

> LPUSH name-list Jane
(integer) 3

> LRANGE name-list 0 -1
1) "Jane"
2) "Jane"
3) "Anna"
```

```
> LINDEX name-list 2

“Anna”

> LPOP name-list
“Jane”


> LRANGE name-list 0 -1
1) "Jane"
2) "Anna"
```

# Redis SET

Redis SET is a sequence of strings like Redis LIST , but Redis set collects unique strings with no ordering.

| Command | Meaning |
|---|---|
| SADD | Adds the value to the set |
| SMEMBERS | Returns the entries set of values |
| SISMEMBERS | Check of the value is in the set |
| SREM | Remove the value from the set, If it exists |



# Redis SET example

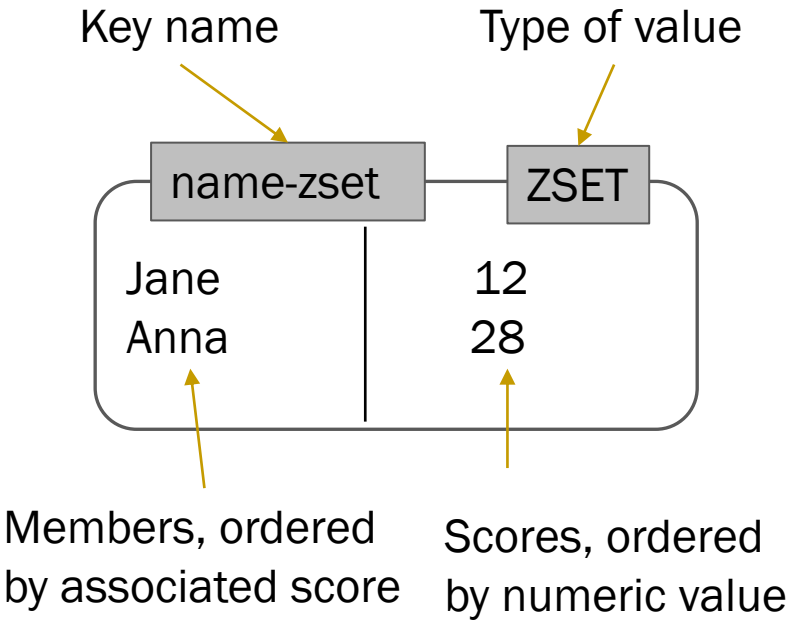
```
> SADD name-set Jane
(integer) 1     //success

> SADD name-set Anna
(integer) 1

> SADD name-set John
(integer) 1

> SADD name-set Jane
(integer) 0
//cannot add duplicate value
```

```
> SMEMBERS name-set
1) "Jane"
2) "Anna"
3) "John"

> SISMEMBER name-set Pang
(integer) 0

> SISMEMBER name-set Anna
(integer) 1

> SREM name-set Anna
(integer) 1
```

# Redis SORTED SET (ZSET)

Redis SORTED SET is a collection of unique strings ordered by user defined scoring. Every element in a sorted set is associated with a floating-point value (called score)

| Command | Meaning |
|---|---|
| ZADD | Adds the value with the given score to the ZSET |
| ZRANGE | Retrieves the values in the ZSET from their position in the sorted order |
| ZRANGEBYSCORE | Retrieves the values in the ZSET based on a range of scores |
| ZREM | Remove the value from the ZSET, If it exists |



# Redis SORTED SET (ZSET) example

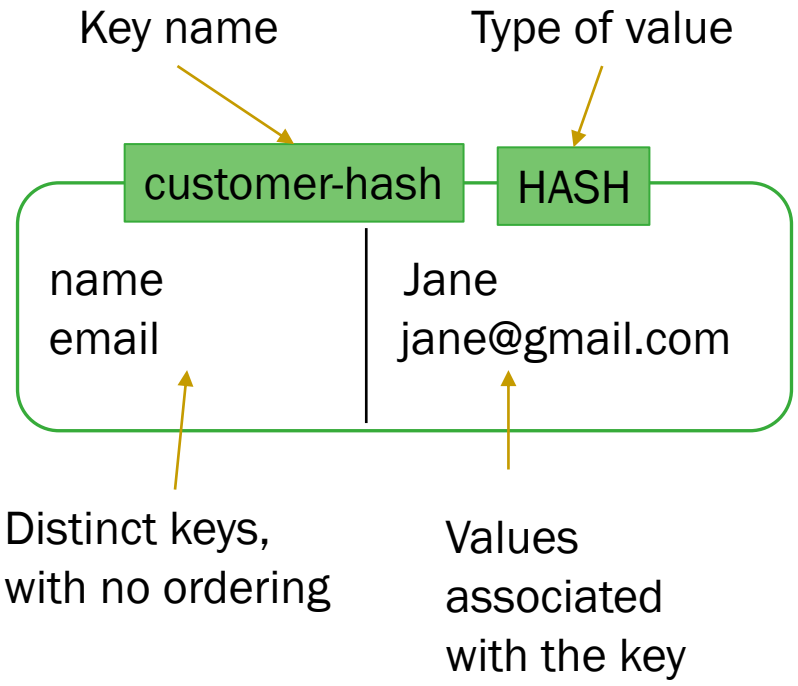
```
> ZADD name-zset  12 Jane
(integer) 1      //success

> ZADD name-zset  28 Anna
(integer) 1

> ZADD name-zset 28 Anna
(integer) 0
//cannot add duplicate value
```

```
> ZRANGE name-zset 0 -1 withscores
1) "Jane"
2) 12.0
3) "Anna"
4) 28.0

> ZREM name-zset Anna
(integer) 1

> ZRANGE name-zset 0 -1 withscores
1) "Jane"
2) 12.0
```

# Redis HASH

Redis HASH is a collection of key-value pairs. The value which stored in HASH can be strings and numbers.

| Command | Meaning |
|---|---|
| HSET | Stores the value at the key in the hash |
| HGET | Retrieves the value at the given hash key |
| HGETALL | Retrieves the entire hash |
| HDEL | Remove the key from the hash, if it exists |
| HLEN | Returns the number of fields contained in the hash stored at key. |




Note: HMSET and HMGET commands are used to set or retrieve the value of multiple keys in a single command

# Redis HASH example

```
> HSET customer-hash name Jane
(integer) 1      //success

> HSET customer-hash email
jane@gmail.com
(integer) 1

> HGET customer-hash name
"Jane“

> HLEN customer-hash
2
```

```
> HGETALL customer-hash
1) "name“
2) "Jane“
3) "email“
4) "jane@gmail.com"

> HDEL customer-hash email
(integer) 1

> HGETALL customer-hash
1) "name“
2) "Jane“
```

# Use Case: Shopping Cart

1.Admin add new products into AIT shopping website for three items as follows.

| Product ID | Name | Price | Category |
|---|---|---|---|
| P001 | ML Textbook | 300 | book |
| P002 | DMM Textbook | 250 | book |
| P003 | Pencil | 15 | stationery |

| P001 | HASH |
|---|---|
| product_id | P001 |
| name | ML Textbook |
| price | 300 |
| category | book |

| P002 | HASH |
|---|---|
| product_id | P002 |
| name | DMM Textbook |
| price | 250 |
| category | book |

| P003 | HASH |
|---|---|
| product_id | P003 |
| name | Pencil |
| price | 15 |
| category | stationery |

# Use Case: Shopping Cart (2)

Create Product 1 in HASH:  use command HSET and set P001 as a key.

```
> HSET P001 name "ML Textbook"
> HSET P001 price 300
> HSET P001 category "book"
```

Create Product 2 in HASH:  use command HSET and set P002 as a key.

```
> HSET P002 name "DMM Textbook"
> HSET P002 price 250
> HSET P002 category "book"
```

Create Product 3 in HAS :  use command HMSET and set P003 as a key.

```
> HMSET P003 name "Pencil" price 15 category "stationery"
```

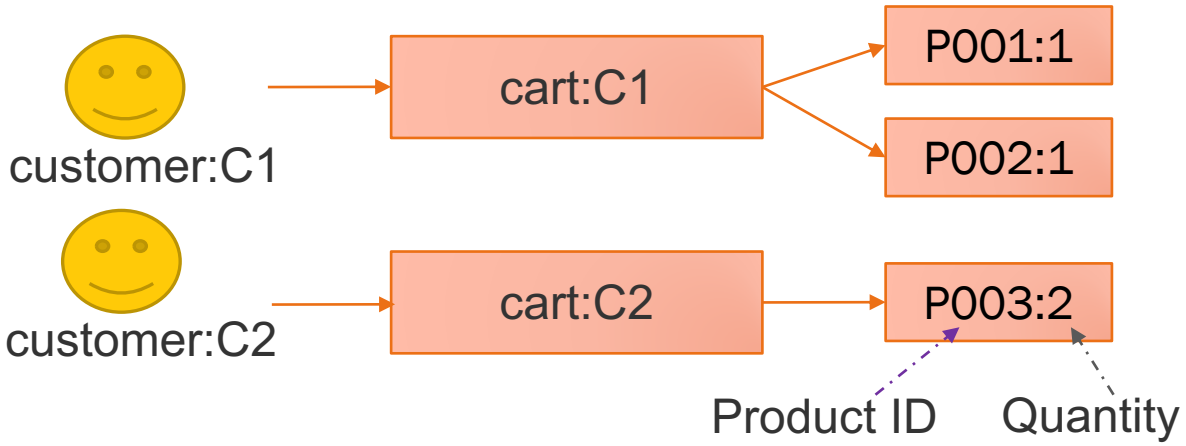
# Use Case: Shopping Cart (3)

Customer

| Customer ID | Name | Address |
|---|---|---|
| C1 | Jane | AIT |
| C2 | Anna | AIT |

Cart_product

| Customer ID | Product ID | Quantity |
|---|---|---|
| 1 | P001 | 1 |
| 1 | P002 | 1 |
| 2 | P003 | 2 |



# Use Case: Shopping Cart (3)

Customer

customer:C1 HASH

| | |
|---|---|
| id | C1 |
| name | Jane |
| address | AIT |

customer:C2 HASH

| | |
|---|---|
| id | C2 |
| name | Anna |
| address | AIT |

Shopping Cart

cart:C1 SET

P001:1
P002:1

cart:C2 SET

P003:2

# Use Case: Shopping Cart (4)

2. Jane and Anna would like to register to AIT shopping website. She fills out her information in the registration form including name and address.
Create User:  use command **HSET**

```
> HSET customer:C1 id C1
> HSET customer:C1 name Jane
> HSET customer:C1 address AIT
```

Create User:  use command **HMSET**

```
> HMSET customer:C2 id C2 name Anna address AIT
```

3. After admin approve, Jane go to AIT shopping website and pickup new products into her shopping cart.
Get all data in Hash: use command **HGETALL**

```
> HGETALL customer:C1
```

# Use Case: Shopping Cart (5)

Add product P001 with quantity= 1 in a SET: use command **SADD**

```
> SADD cart:C1 P001:1
```

Add product P002 with quantity= 1 in a SET : use command **SADD**

```
> SADD cart:C1 P002:1
```

Get all product in a cart using **SMEMBERS**

```
> SMEMBERS cart:C1
1) "P002:1"
2) "P001:1"
```

4. Jane change her mind. She would like to buy only "DMM Textbook" , so she delete "ML Textbook" in his shopping cart.

Delete "ML Textbook" at product id P001 in a cart with command **SREM**.

```
> SREM cart:C1 P001:1
```

# Use Case: Shopping Cart (6)

5. She checks all her orders in shopping cart again and prepare for payment.
Get all data in a cart (collected in SET): use command SMEMBERS

```
> SMEMBERS cart:C1
1)  "P002"
 > HGETALL P002
1) "name"
2) "DMM Textbook"
3) "price“
4) "250"
5) "category"
6) "book"
```

After finish payment process,  all data in a cart will be deleted.

```
> DEL cart:C1
```

# References


1. Introduction of Redis data types: https://matt.sh/introduction-to-redis-data-types
2. Redis Data Types: https://www.tutorialspoint.com/redis/redis_data_types.htm
3. Redis command: https://redis.io/commands
4. Interactive tutorial: http://try.redis.io/
5. Install-cli-comands and data types: https://auth0.com/blog/introduction-to-redis-install-cli-commands-and-data-types/
6. The ReJSON Redis Module: http://rejson.io/
7. https://www.slideshare.net/arthurshvetsov/redis-basics


Thank you.