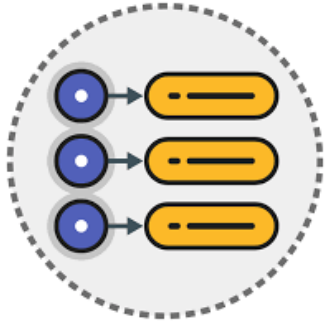


DAY 1 – 3 KEY-VALUE MODEL

Emerging Data Modeling and Management Technology: Key-Value and Document Databases

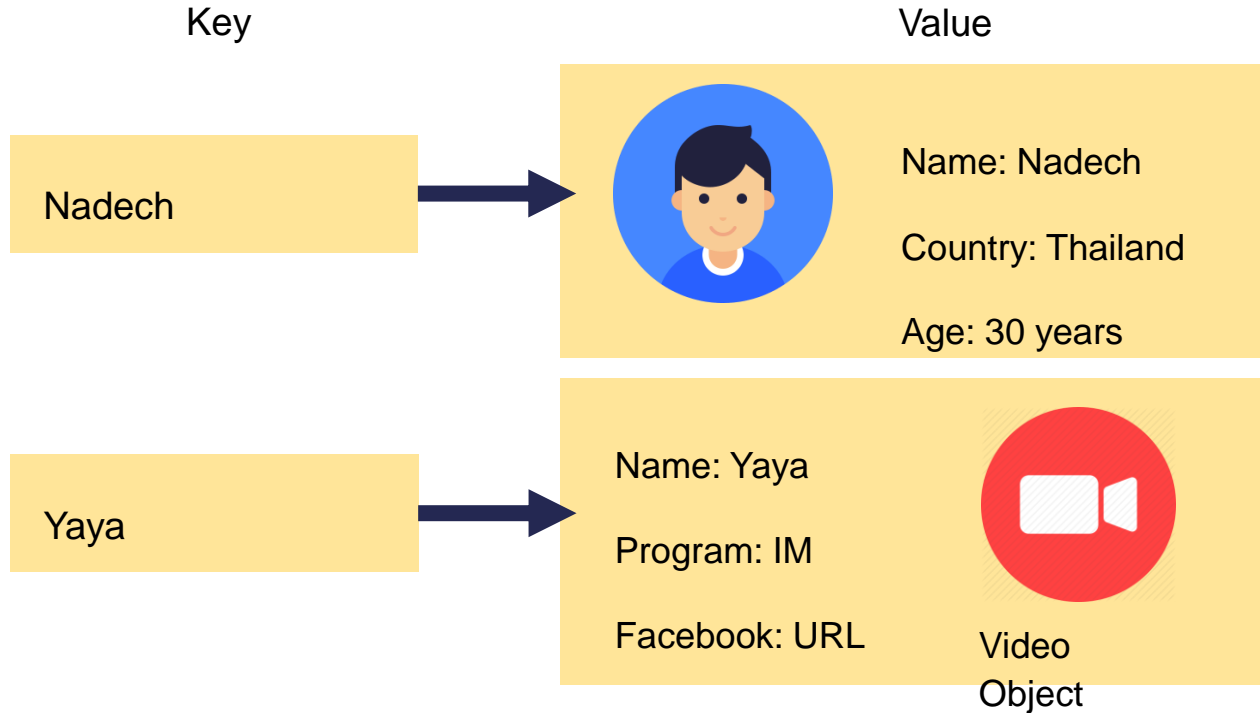
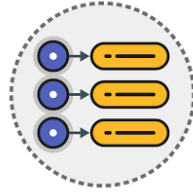
DS&AI : PROFESSIONAL TRAINING COURSE

1 – 3 APRIL 2021 | ASIAN INSTITUTE OF TECHNOLOGY

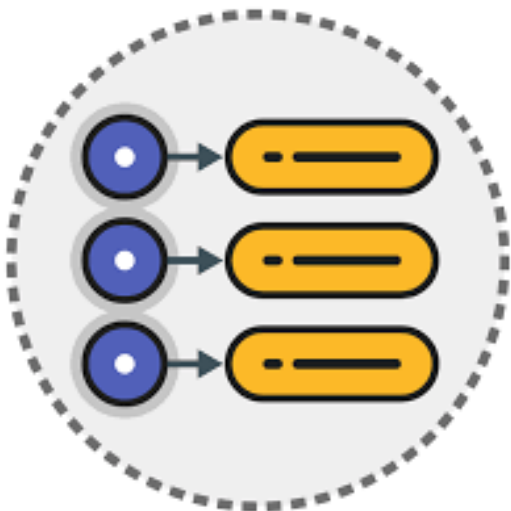


Key-Value Model

Key-Value Model



Key-Value Model



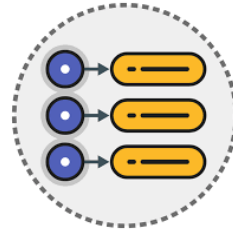
The simplest model: just Keys and Values

- No Schema
- Keys: synthetic or auto-generated
- Values: any object type (e.g., String, JSON, BLOB) stored as uninterpreted block, thus the keys are the only way to retrieve stored data.

Query operations for stored objects are associated with a key:

- PUT, GET, DELETE

Benefits vs. Limitations



BENEFITS

Extremely fast retrieval using the key

Virtually no restriction on the type of data that can be stored:

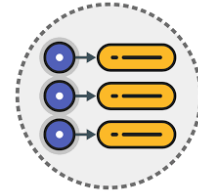
- Text (for example, the HTML code for a Web page)
- Any type of multimedia binary (still images, audio, and video).

LIMITATIONS

Cannot search within stored values rather than always retrieving by the key

Cannot update parts of a “value” while it’s in the database. You must replace the entire value with a new copy if modifications are needed.

Applications & Use Cases



Best suited for applications where access is only through the key.

They are being used for Web sites that include thousands of pages, large image databases, and large catalogs. They are also particularly useful for keeping Web app session information.

Redis Use Cases

Redis Data Types	Example use cases
Redis String	Session Cache: Many websites leverage Redis Strings to create a session cache to speed up their website experience by caching HTML fragments or pages.
	Queues: Any application that deals with traffic congestion, messaging, data gathering, job management, or packet routing should consider a Redis Queue , as this can help you manage your queue size by rate of arrival and departure for resource distribution.
Redis Lists	Social Networking Sites: Social platforms like Twitter use Redis Lists to populate their timelines or homepage feeds, and can customize the top of their feeds with trending tweets or stories.
	Leaderboards: Forums like Reddit and other voting platforms leverage Redis Lists to add articles to the leaderboard and sort by most voted entries.

Redis Use Cases

Redis Data Types	Example use cases
Redis Sets	Analyzing Ecommerce Sales: Many online stores use Redis Sets to analyze customer behavior, such as searches or purchases for a specific product category or subcategory. For example, an online bookstore owner can find out how many customers purchased medical books in Psychology.
	Inappropriate Content Filtering: For any app that collects user input, it's a good idea to implement content filtering for inappropriate words, and you can do this with Redis Sets by adding words you'd like to filter to a SET key and the SADD command.
Redis Sorted Sets	Q&A Platforms: Many Q&A platforms like Stack Overflow and Quora use Redis Sorted Sets to rank the highest voted answers for each proposed question to ensure the best quality content is listed at the top of the page.
	Task Scheduling Service: Redis Sorted Sets are a great tool for a task scheduling service , as you can associate a score to rank the priority of a task in your queue. For any task that does not have a score noted, you can use the WEIGHTS option

Redis Use Cases

Redis Data Types	Example use cases
Redis Hash	User Profiles: Many web applications use Redis Hashes for their user profiles, as they can use a single hash for all the user fields, such as name, surname, email, password, etc.
	User Posts: Social platforms like Instagram leverage Redis Hashes to map all the archived user photos or posts back to a single user. The hashing mechanism allows them to look up and return values very quickly, fit the data in memory, and leverage data persistence in the event one of their servers dies.

Getting familiarization with Redis Enterprise Cloud (free version)



Key-Value Store

- What is Redis?
- Redis Practice Architecture
- Redis Commands
- Implement Redis for User session and shopping cart

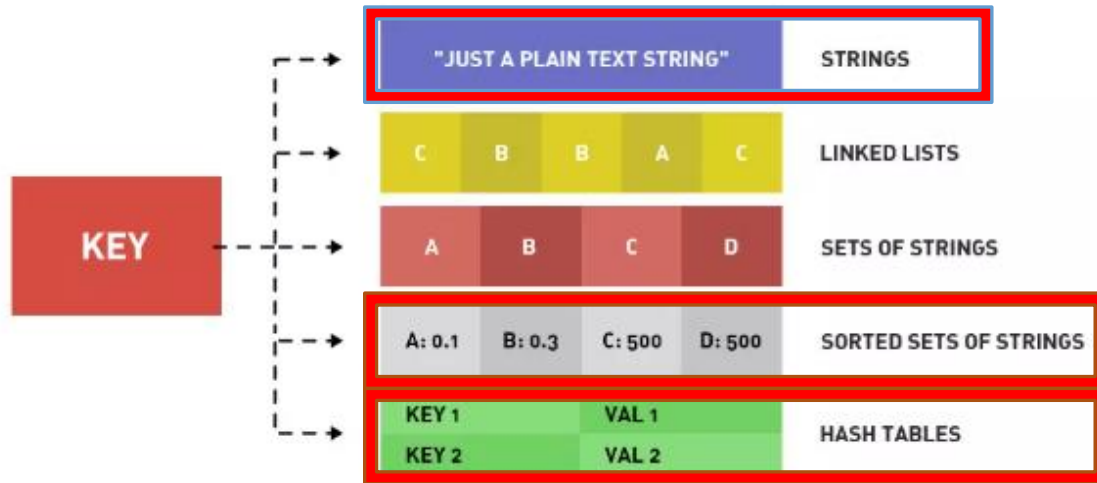
What is Redis?



- 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.

Redis Data Types: String, List, Set, Sorted Set, Hash

Redis Data Types



One key to rule them all.

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.

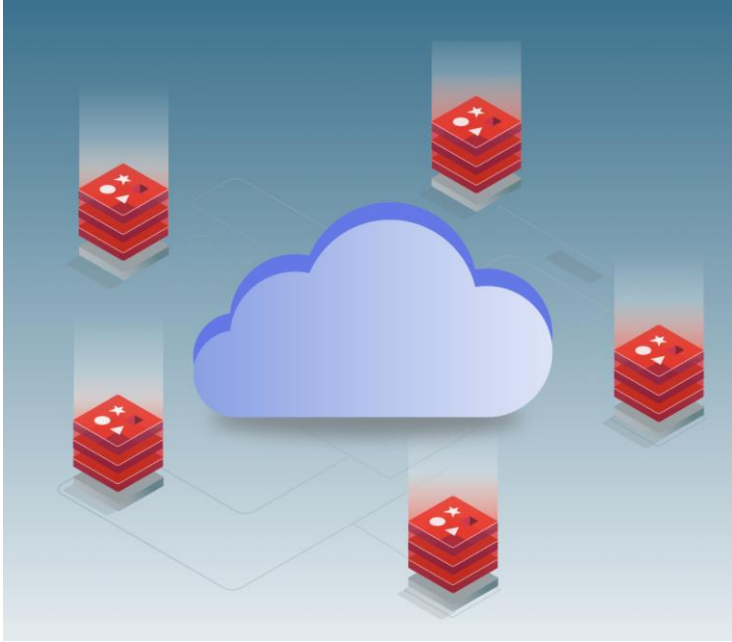
SET: A collection of unique strings with no ordering.

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

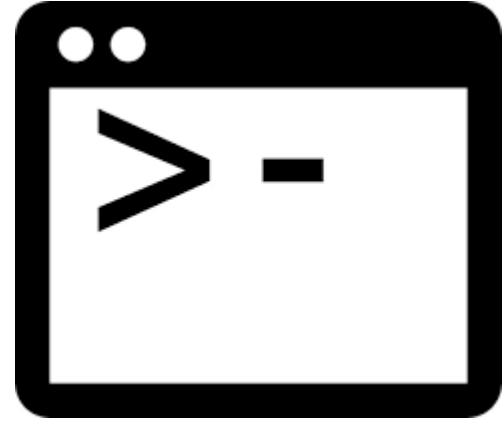
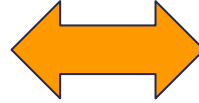
HASH: Unordered hash table of keys to values

CRUD Operations

Redis Practice Lab Architecture




Redis Enterprise Cloud



Redis-cli




Redis Enterprise Cloud (Free Version)

Register at <https://redislabs.com/try-free/>

 Redis Enterprise Cloud

New Subscription

Select Cloud Provider



Select Region

us-east-1

A plan is limited by any of the parameters indicated whichever comes first

GB - Plan memory limit. Applies to the aggregate memory of all databases and replicas.

Ops/sec - Plan throughput limit. Applies to aggregate throughput of all databases.

Databases - Max number of plan databases, including replicas.

Essentials

Low Throughput

High Availability: ☐ Off ☒ Replication (optional) ☐ Multi-AZ

30MB	100MB	250MB	500MB
1 Databases Free	4 Databases \$7/mo	8 Databases \$18/mo	12 Databases \$36/mo

Cloud Essentials
30MB
1 Databases

Free



DMM_training

Create

High-availability	Replication in same AZ
Data persistence to EBS	
Daily and instant backups	
Connections	30
Security groups	0
Source IP auth. rules	1
Support	Basic

Price excludes taxes

Redis Enterprise Cloud



Redis Enterprise Cloud ▾

Create Database

Database Name	<input type="text" value="shopping"/>		
Subscription	#1330887 Essentials/AWS/us-east-1/Standard/30MB		
Protocol	<div>Redis ▾</div>		
Replication ⓘ	Disabled ⓘ		
Data Persistence ⓘ	None		
Access Control & Security	<div><div><input checked="" type="radio"/> Redis Password ⓘ</div><div><div>.....</div><div>⌵</div></div></div>		
Data Eviction Policy ⓘ	<div>volatile-lru ▾</div>		
Modules	<div><input type="checkbox"/></div>		
Alert Settings ⓘ	<input checked="" type="checkbox"/> Total size of datasets under this plan has reached	<div><input type="text" value="80"/></div>	% of plan limit
	<input checked="" type="checkbox"/> Number of connections has reached	<div><input type="text" value="80"/></div>	% of plan limit

Cancel

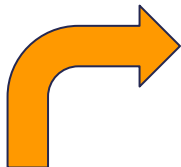
Activate

Redis Enterprise Cloud

Databases



shopping



View Database



Database Name

shopping

Metrics

Slowlog

Configuration

Subscription #1330887 Essentials/AWS/us-east-1/Standard/30MB

Protocol Redis

Used Memory 6.30 MB

Replication *i* Disabled

Activated On 01/17/2021 03:25:17

Last Changed

Endpoint redis-11851.c9.us-east-1-2.ec2.cloud.redislabs.com:11851

Data Persistence None

Access Control & Security Default User Password

Host

Port

Subscription #1330887 Essentials/AWS/us-east-1/Standard/30MB DMM_training

1

21%

Name Type Memory Plan usage Endpoint Modules Options Status

shopping

Redis

6.30 MB

21%

redis-11851.c9.us...



Install Redis-cli without Installing Redis-server

1. **Install Nodejs:** <https://nodejs.org/en/download/>
2. **install the Node.js version of redis-cli:**
`> npm install -g redis-cli`
3. **Connect to redis server:**
`> rdcli -h redis-11851.c9.us-east-1-2.ec2.cloud.redislabs.com -p 11851 -a [your redis password]`

URL: 1. [Install Redis-cli without Installing Redis-server](#)
2. [Install Redis](#) (Install Redis-server in your local machine + Redis-cli)

Connect to Redis Enterprise Cloud using redis-cli



```
Command Prompt - rdcli -h redis-11851.c9.us-east-1-2.ec2.cloud.redislabs.com -p 11851 -a Po1pmWbqEVS...  
(c) 2019 Microsoft Corporation. All rights reserved.  
  
C:\Users\purin>node -v  
v14.16.0  
  
C:\Users\purin>rdcli -h redis-11851.c9.us-east-1-2.ec2.cloud.redislabs.com -p 11851 -a  
Po1pmWbqEVSVCc2qOTjuwfdFxXygE8q  
redis-11851.c9.us-east-1-2.ec2.cloud.redislabs.com:11851> ping  
PONG  
redis-11851.c9.us-east-1-2.ec2.cloud.redislabs.com:11851>
```

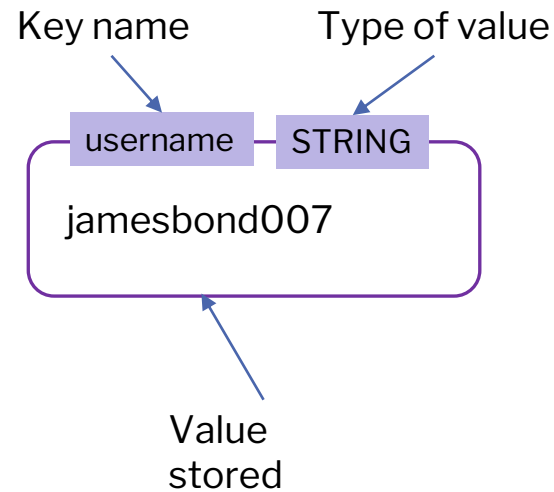
Test connection:

- `rdcli -h redis-11851.c9.us-east-1-2.ec2.cloud.redislabs.com -p 11851 -a Po1pmWbqEVSVCc2qOTjuwfdFxXygE8q`
- `redis-11851.c9.us-east-1-2.ec2.cloud.redislabs.com:11851> ping`
- **PONG**

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 login:session-1 "{user_id:1}"
OK

> MSET login:session-1 "{user_id:1}"
login:session-2 "{user_id:2}"

OK

> GET login:session-2
"{user_id:2}"

> MGET login:session-1 login:session-2
1) "{user_id:1}"
2) "{user_id:2}"

> DEL login:session-2
(integer) 1
```

```
> SET view_count 10
OK

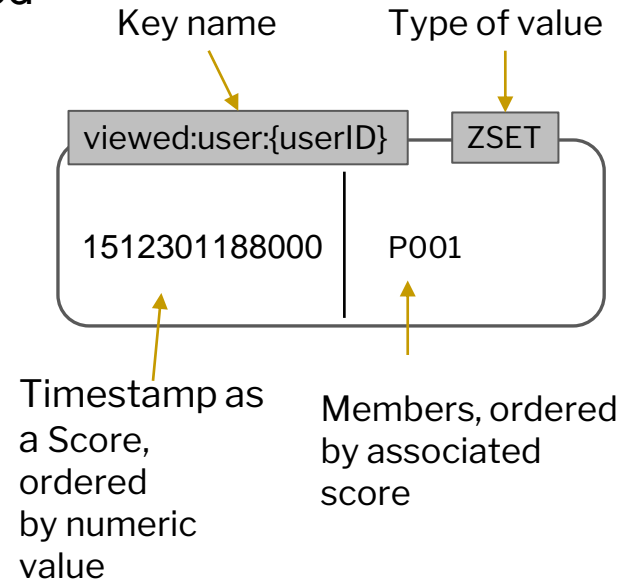
> INCR view_count
(integer) 11

> INCR view_count
(integer) 12
```

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 viewed:user:1 1590215629000 P001
(integer) 1 //success

> ZADD viewed:user:1 1590215629002 P002
(integer) 1

> ZADD viewed:user:1 1590215629004 P003
(integer) 1

> ZADD viewed:user:1 1590215629006 P004
(integer) 1

> ZADD viewed:user:1 1590215629008 P001
(integer) 0 //cannot add duplicate value
```

```
> ZRANGE viewed:user:1 0 -1
1) "P002"
2) "P003"
3) "P004"
4) "P001" // recently

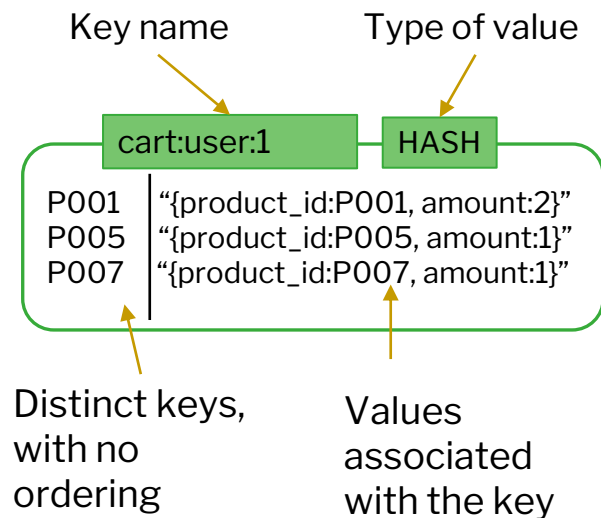
//keep last 3 viewed products
> ZREMRANGEBYRANK viewed:user:1 0 -4
(integer) 1

> ZRANGE viewed:user:1 0 -1
1) "P003"
2) "P004"
3) "P001"
```

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 cart:user:1 P001 "{product_id:P001, amount:2}"  
(integer) 1 //success
```

```
> HSET cart:user:1 P005 "{product_id:P005, amount:1}"  
(integer) 1
```

```
> HGET cart:user:1 P005  
"{product_id:P005, amount:1}"
```

```
> HGETALL cart:user:1  
1) "P001"  
2) "{product_id:P001, amount:2}"  
3) "P005"  
4) "{product_id:P005, amount:1}"
```

```
> HLEN cart:user:1
```

```
2
```

```
> HDEL cart:user:1 P005  
(integer) 1
```

```
> HGETALL cart:user:1
```

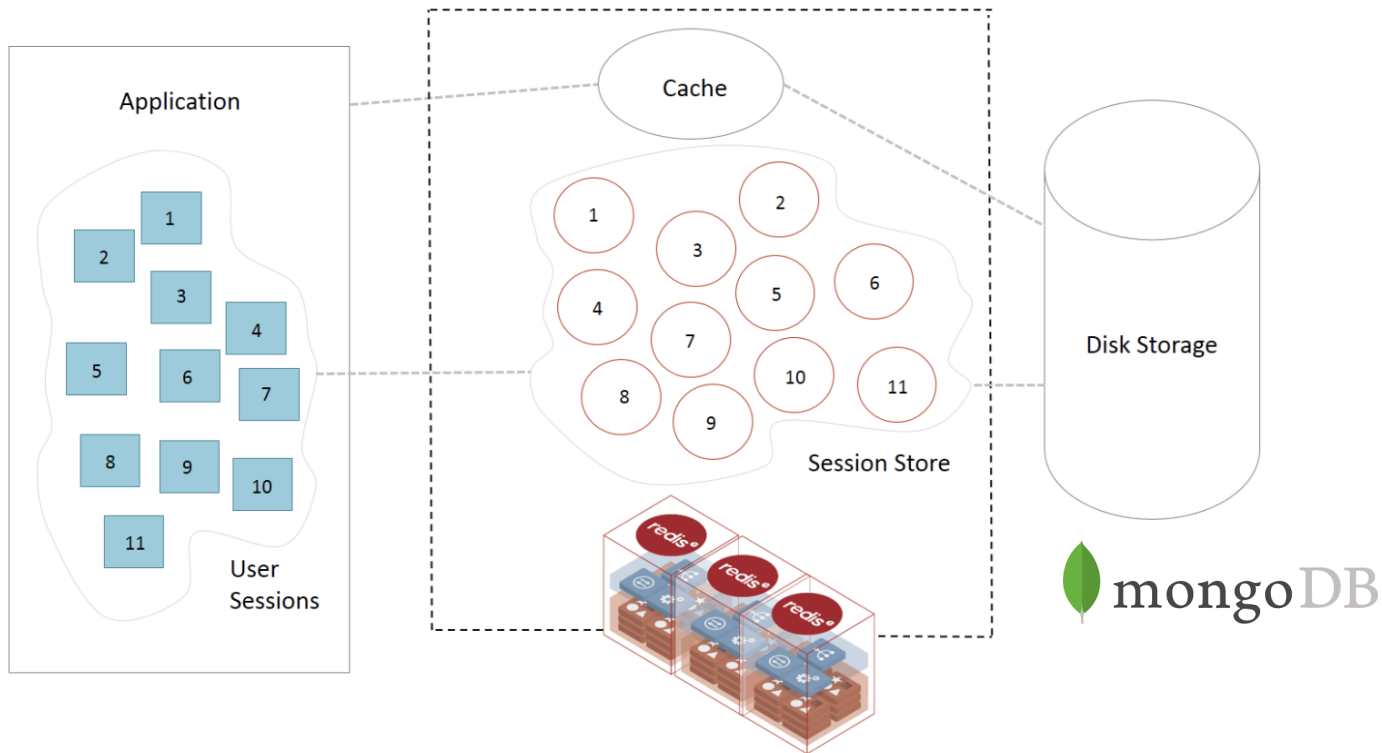
```
1) "P001"
```

```
2) "{product_id:P001, amount:2}"
```

```
> DEL cart:user:1  
(integer) 1
```

Project Practice: Modeling and Managing

Designing Cache and Session Store with Redis

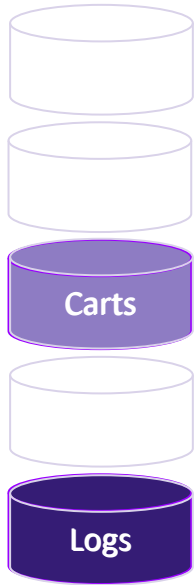


Redis in E-Commerce System

Functions

1. User Session Management : Login Sessions
2. User Behavior Log Management: Viewed Products Log
3. Shopping Cart Management

Summary Key-Value Data Model



Data		Key	Value	Data type
Login Session	sessionID user_id	login:{sessionID}	user_id	String
	timestamp sessionID	recent	timestamp sessionID	Sorted Set
Viewed Products Log	userID timestamp produc_id	viewed:{email}	timestamp produc_id	Sorted Set
Shopping Cart	userID product_id amount price	cart:{email}	product_id amount price	Hash

STRING

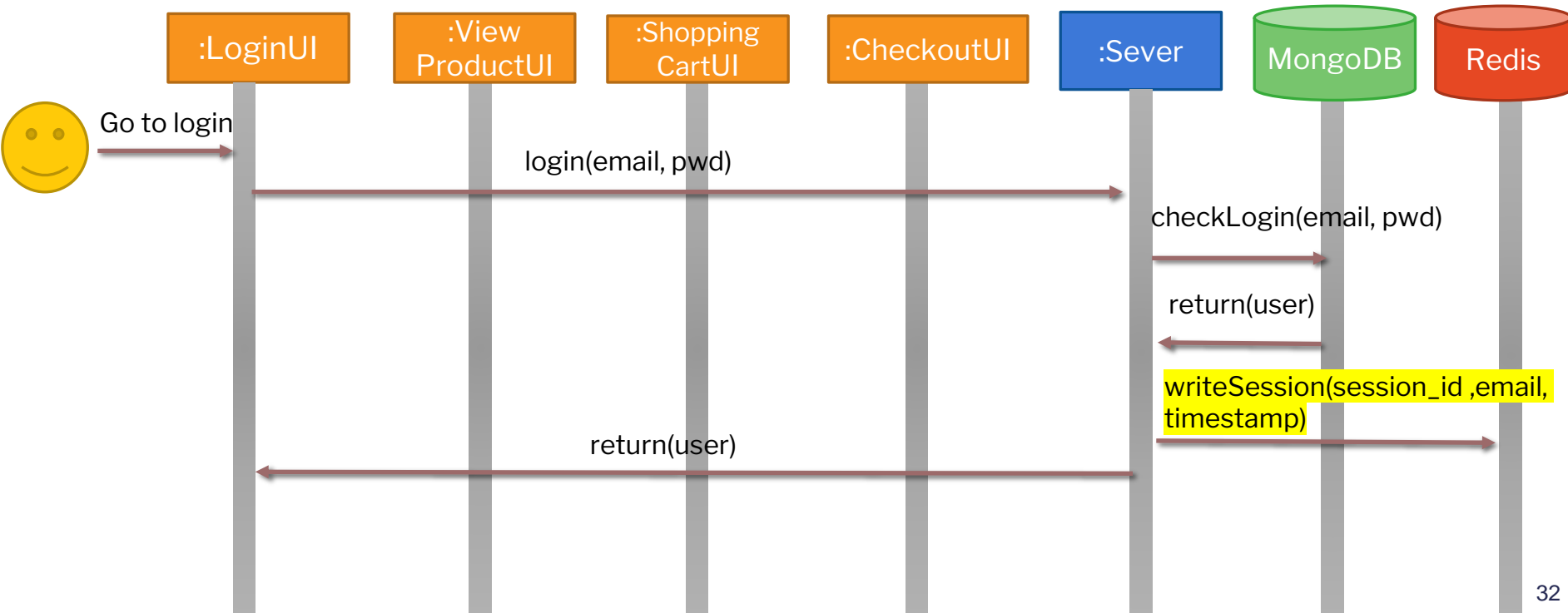
LIST

SET

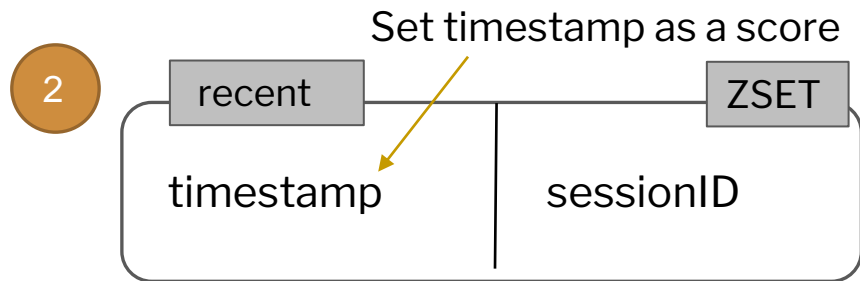
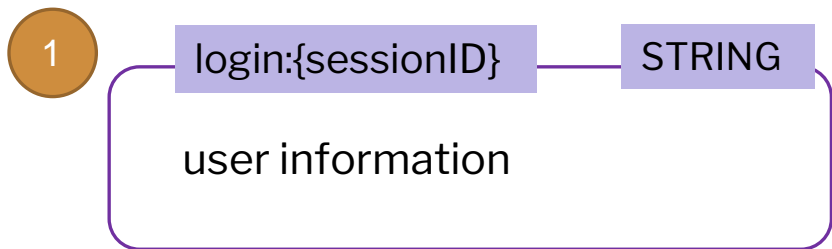
SORTED SET

HASH

U1. Login Session Management



(U1) writeSession(session_id ,user_id, timestamp)



Example Data:

Key	Value
login:session-1	"{email:'june@gmail.com'}"
login:session-2	"{email:'tcrawford@hotmail.com'}"

Key	Value
recent	1511533205001 session-1
recent	1511532142401 session-2

Example Commands:

```
SET login:session-1 "{email:'june@gmail.com'}"  
GET login:session-1  
DEL login:session-1
```

```
ZADD recent 1511533205001 session-1  
ZRANGE recent 0 -1  
ZREMRANGEBYRANK recent 0 -51
```

U1. Login Session Management

//1 user information

Create login session as a string: use command **SET** and set **login:{sessionID}** as a key.

```
> SET login:qHsXwI9URyRms8117mjH0w "{email:'june@gmail.com'}"
```

//2. log access

Create recent login session: use command **ZADD** and set **recent** as a key.

```
> ZADD recent 1511533205001 session-1
```

//expire session each 1 hour

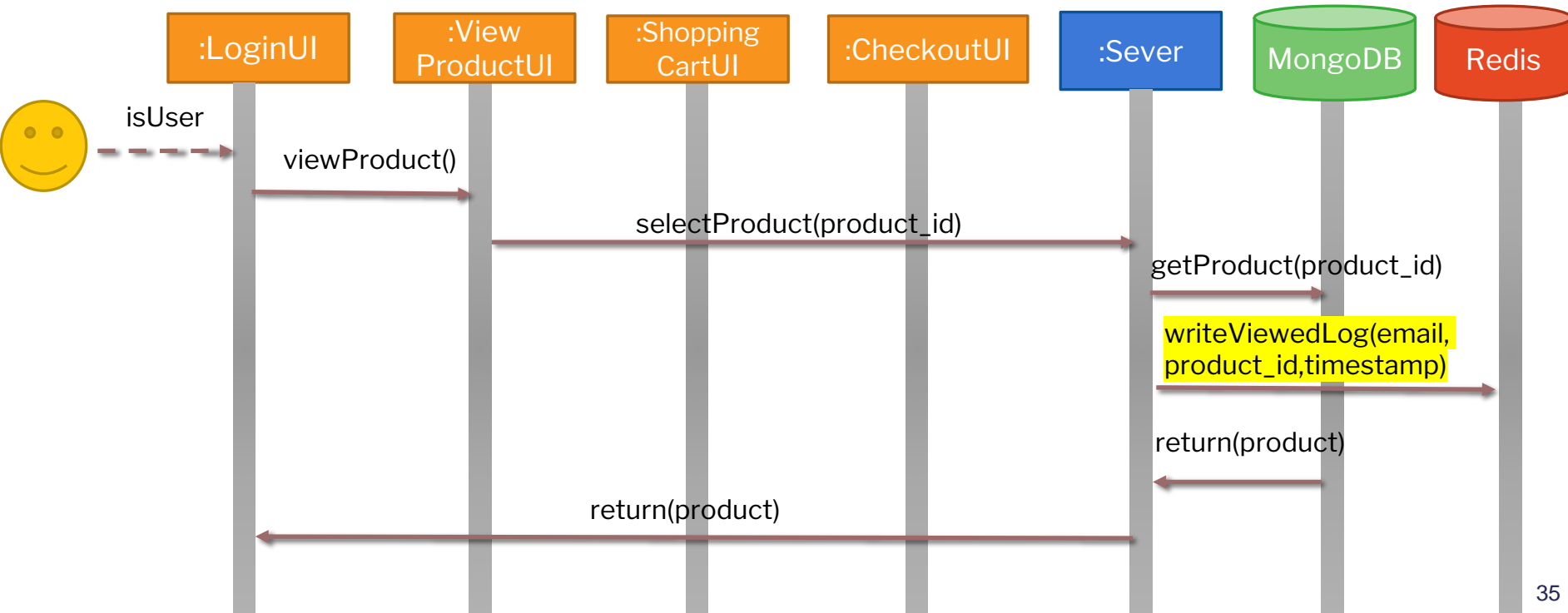
Set expire session: use command **EXPIRE key second**

```
> EXPIRE login:qHsXwI9URyRms8117mjH0w 3600
```

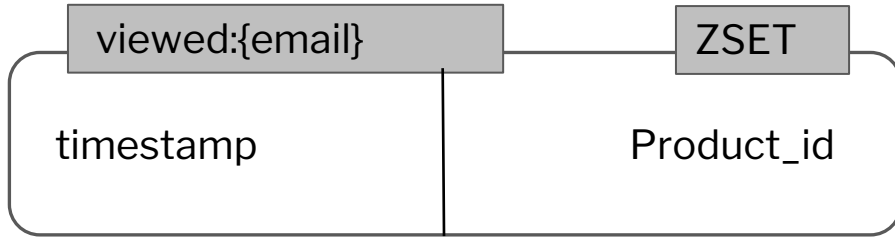
// keep only the top 100 recent session

```
> ZREMRANGEBYRANK recent 0 -101
```

U2. Recently Viewed Products Log



(U2.) writeViewedLog(user_id, product_id, timestamp)



Store viewed product log inside a sorted set with a key like “viewed:{email}” In sorted set, we store timestamp as a score and product_id

Example Data:

Key	Value
viewed:june@gmail.com	1511533205001 P001
viewed:tcrawford@hotmail.com	1511532142401 P005

Example Commands:

```
ZADD viewed:june@gmail.com 1511533205001 P001
```

```
ZRANGE viewed:tcrawford@hotmail.com 0 -1 withscores
```

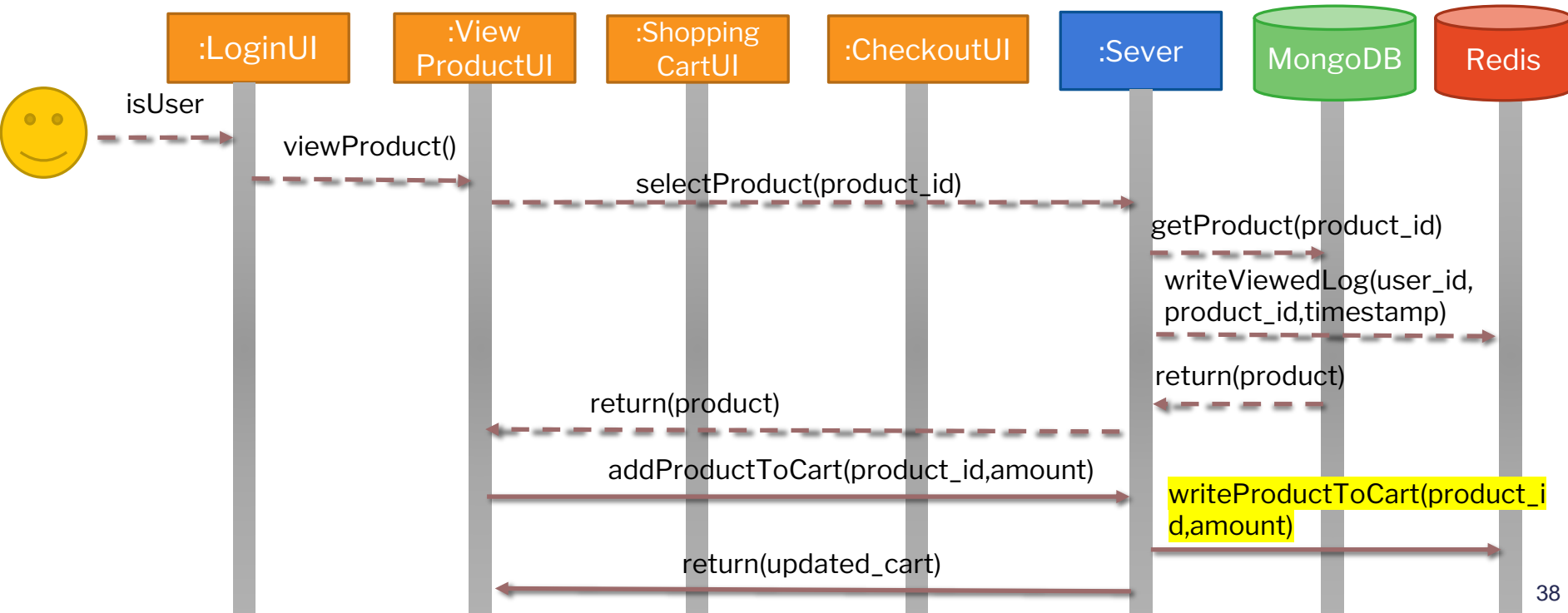
```
ZREMRANGEBYRANK viewed:user:1 0 -11
```

Task: Recently Viewed Products Log

Instruction:

1. Write viewed product log of product_id P001 and P005 for tom@gmail.com. use command **ZADD** and set **viewed:tom@gmail.com** as a key.
2. Remove old items of viewed product, keeping the most recent 25 items.

U3. Shopping Cart Management



U3. writeProductToCart(product_id,amount)



cart:{email}

HASH

product_id: "CartItem Detail"
product_id: "CartItem Detail"
product_id: "CartItem Detail"

Store a cart object inside a hash with a key like "cart:user:{userID}". Inside this hash, we store the product_id as a key and the value is the item details in JSON format.

Example Data:

Key	Value
cart:june@gmail.com	P003: "{product_id:P003,amount:2}" P004: "{product_id:P004,amount:10}"

Example Commands:

```
HSET cart:june@gmail.com P004 "{product_id:P004,amount:10}"
```

```
HDEL cart:june@gmail.com P004
```

```
HGETALL cart:user:1
```

U3. writeProductToCart(product_id,amount)

Add Product P001 in HASH: use command **HSET** and set **cart:{email}** as a key.

```
> HSET cart:june@gmail.com P001 "{product_id:P001,amount:2}"
```

Add Product P002 and P003 in HASH : use command **HMSET**

```
> HMSET cart:june@gmail.com P002 "{product_id:P002,amount:1}" P003  
"{product_id:P003,amount:5}"
```

Delete Product Item in Hash: use command **HDEL**

```
> HDEL cart:june@gmail.com P001
```

Get all data in Hash: use command **HGETALL**

```
> HGETALL cart:june@gmail.com
```


Redis Use Cases

Redis Data Types	Example use cases
Redis String	Session Cache: Many websites leverage Redis Strings to create a session cache to speed up their website experience by caching HTML fragments or pages.
	Queues: Any application that deals with traffic congestion, messaging, data gathering, job management, or packet routing should consider a Redis Queue , as this can help you manage your queue size by rate of arrival and departure for resource distribution.
Redis Lists	Social Networking Sites: Social platforms like Twitter use Redis Lists to populate their timelines or homepage feeds, and can customize the top of their feeds with trending tweets or stories.
	Leaderboards: Forums like Reddit and other voting platforms leverage Redis Lists to add articles to the leaderboard and sort by most voted entries.

Redis Use Cases

Redis Data Types	Example use cases
Redis Sets	Analyzing Ecommerce Sales: Many online stores use Redis Sets to analyze customer behavior, such as searches or purchases for a specific product category or subcategory. For example, an online bookstore owner can find out how many customers purchased medical books in Psychology.
	Inappropriate Content Filtering: For any app that collects user input, it's a good idea to implement content filtering for inappropriate words, and you can do this with Redis Sets by adding words you'd like to filter to a SET key and the SADD command.
Redis Sorted Sets	Q&A Platforms: Many Q&A platforms like Stack Overflow and Quora use Redis Sorted Sets to rank the highest voted answers for each proposed question to ensure the best quality content is listed at the top of the page.
	Task Scheduling Service: Redis Sorted Sets are a great tool for a task scheduling service , as you can associate a score to rank the priority of a task in your queue. For any task that does not have a score noted, you can use the WEIGHTS option

Redis Use Cases

Redis Data Types	Example use cases
Redis Hash	User Profiles: Many web applications use Redis Hashes for their user profiles, as they can use a single hash for all the user fields, such as name, surname, email, password, etc.
	User Posts: Social platforms like Instagram leverage Redis Hashes to map all the archived user photos or posts back to a single user. The hashing mechanism allows them to look up and return values very quickly, fit the data in memory, and leverage data persistence in the event one of their servers dies.