Redis

COSC 516 – Cloud Databases

Redis introduction

- Remote Dictionary Server (Redis) is an in-memory data store.
- Redis is a scalable and durable data structure store, used as a cache, message broker and key-value databases.
- It is commonly used as a key-value databases to process, analyze real-time data with sub-millisecond latency, so It's perfect for social media streaming, gaming and IOT.
- The organizations that use use Redis such as include Twitter, snapchat and discord and instacart.

in-memory dataset

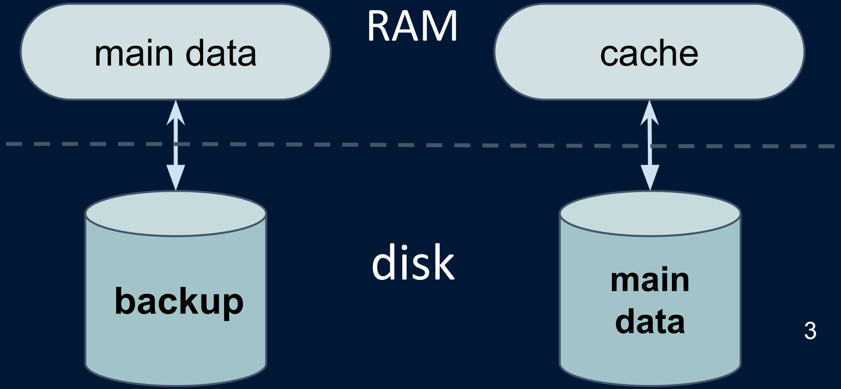
- Remote Dictionary Server (Redis) is an in-memory data store.

❑ in-memory data system : store the data entire in main memory.

  + low latency
  + lower memory and CPU requirement.
  - data volatile

❑ on-disk data system(traditional) : store the data on disk.

  - slower
  - require more memory and CPU
  + Persistence



Built-in replication

- Redis replication is based on leader follower replication, also known as master-replica. The replica instance will be exact copy of the master regardless of what happened to the master.
- Replica are able to connect to others replicas in a cascading-liking like structure, and it’s scalable.
- The default replication mode is asynchronous replication, which is the use cases for most of event.
- It is non-blocking on both of the master and replica side, which means they handle the queries in spite of the on-going synchronization

Built-in replication

- Three mechanism of connection status
  - Well connected → Any change in the mast will be sent to the replica through a stream of command
  - Not connected → the replica proceed with a partial resynchronization.
  - Not connected & partial resychronization is not possible → fully resychronization, which create a snapchat of master data and sent to the replica

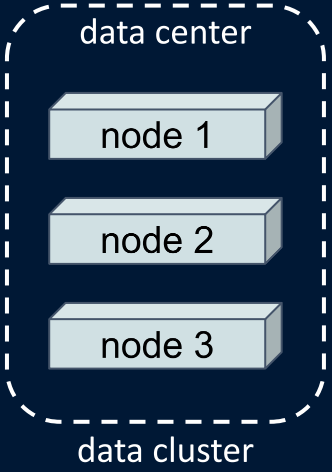
Redis Persistence

- Persistence of the database is the way in which data can be written or transferred to durable storage even when the system in shut down.
  - Redis Database (RDB): perform point-in-time snapshots of your dataset
  - Append only file (AOF): log every write operation received by the server.
  - No persistence : disable the persistence completely.
  - RDB + AOF : combine RDB and AOF in the same instance.

In general, you should use both persistence method. In certain cases, if you care a lot about your data but still can live a few minutes of data loss in case of disaster, you can use RDS alone.

Redis cluster

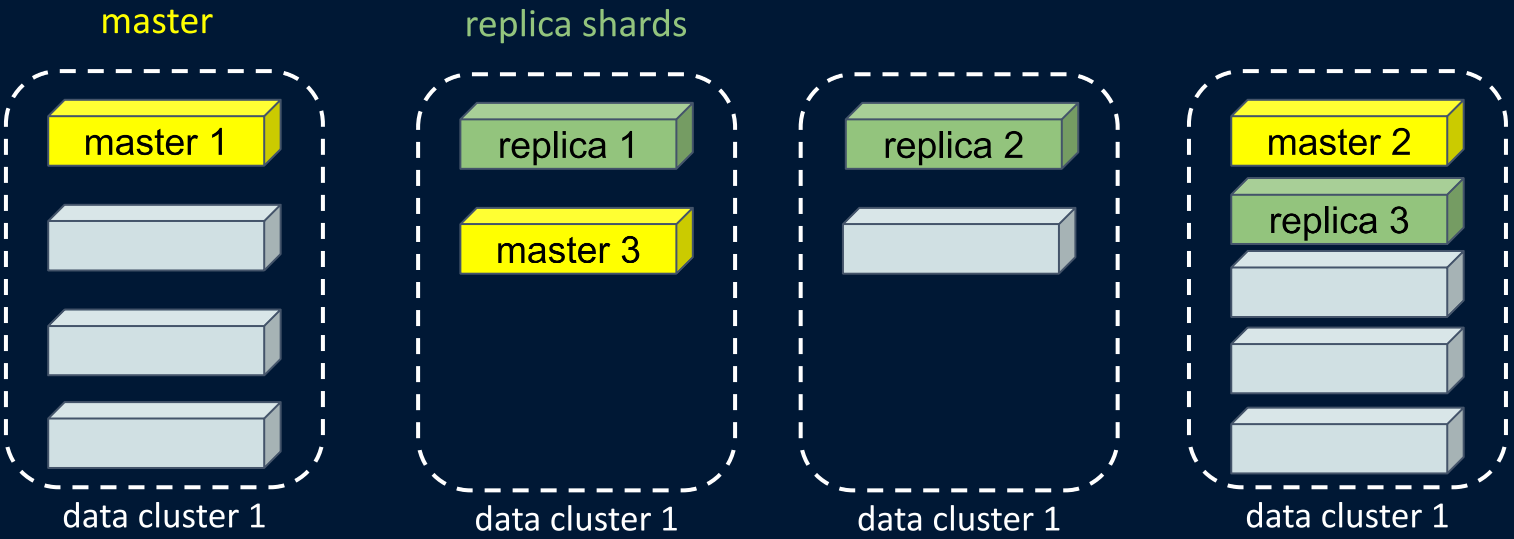
- Redis enterprise offer redis cluster which is a set of redis node that has redis installed on its OS.
- Redis cluster is self-managed.
- Install the redis on the node and it would provide you the UI to interact with.



Redis scalability

- Redis cluster and database scales horizontally with a deployment topology.

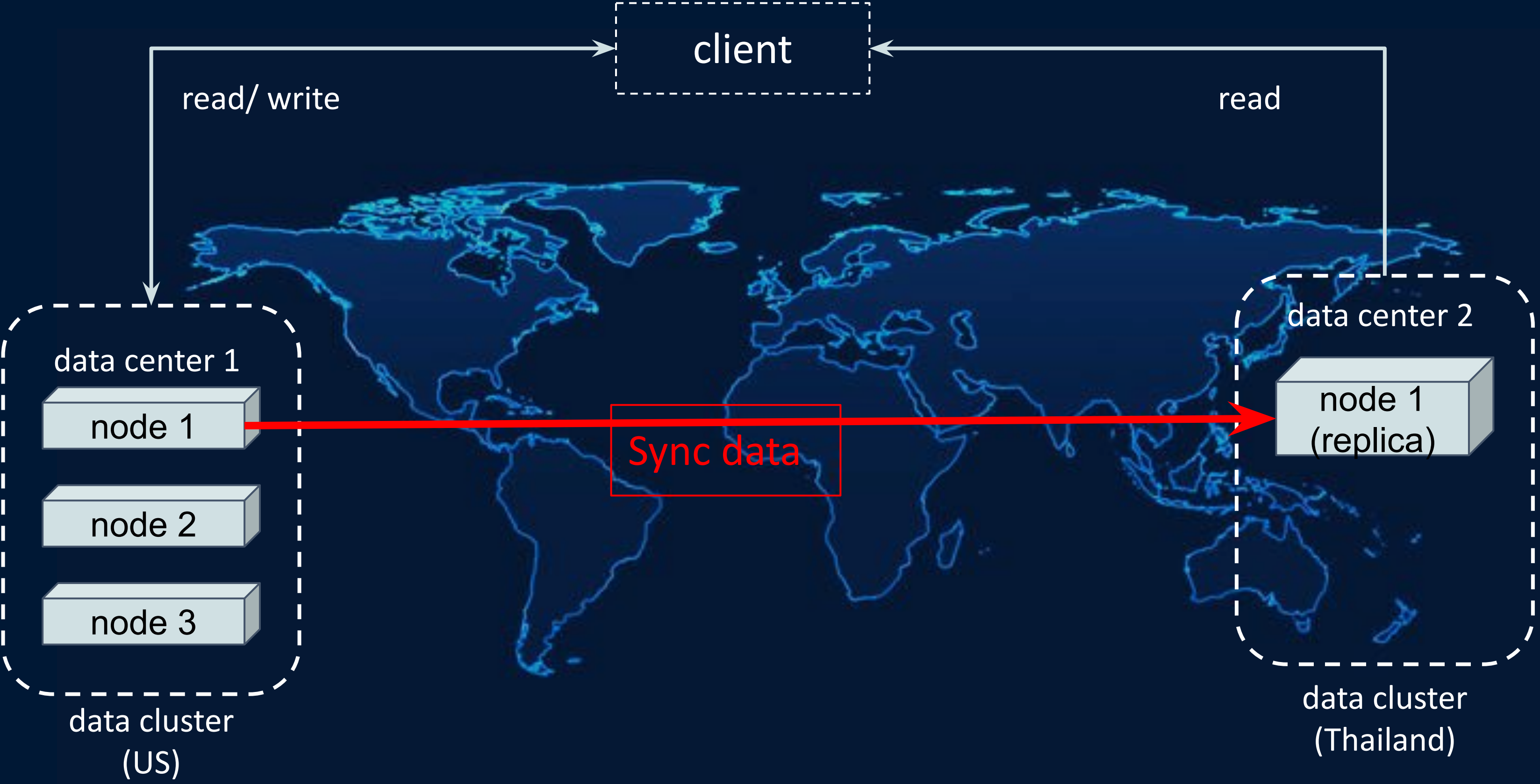
Suppose we want to create the database with shards and replication enables, the redis cluster will automatically shard and choose the node and create a total of 6 shards.



Flexibility of replication-of

- “Active-Passive” geo-distributed deployment using the replica-of capability, and you can synchronize the data between source and destination databases (replica).
- The destination databases can act like a disaster recovery database.
- Flexibility of replica-of
  - source database can replicate to many destinations
  - destination databases can replicate from many source databases
  - multi-region deployment in the same cloud
  - multi-cloud

Flexibility of replication-of

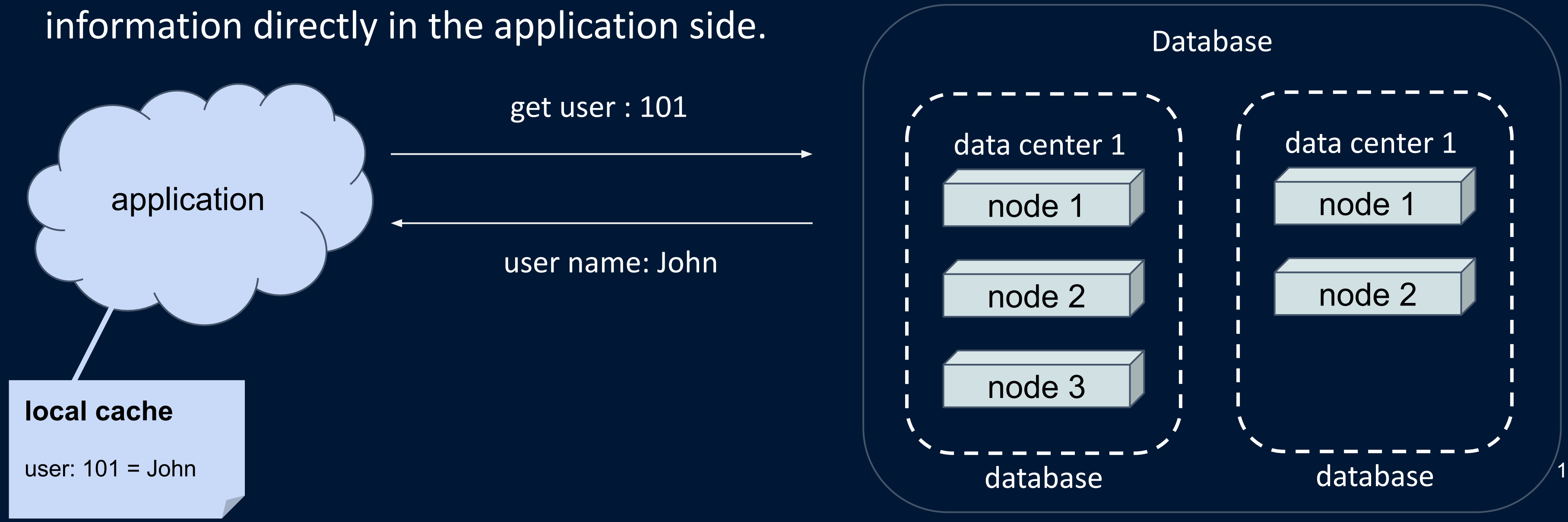


https://stock.adobe.com/search?k=world+map+dark+blue

Client side caching

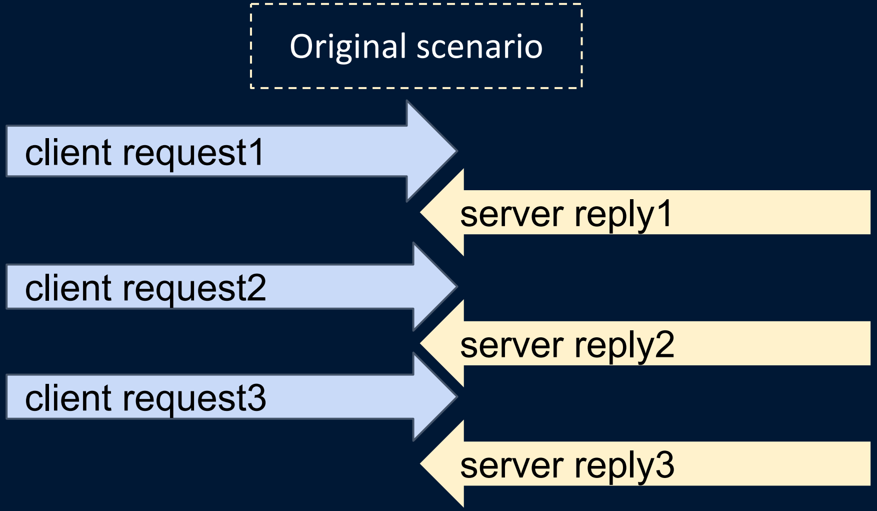
- Client side caching is a technique used to create high performance.

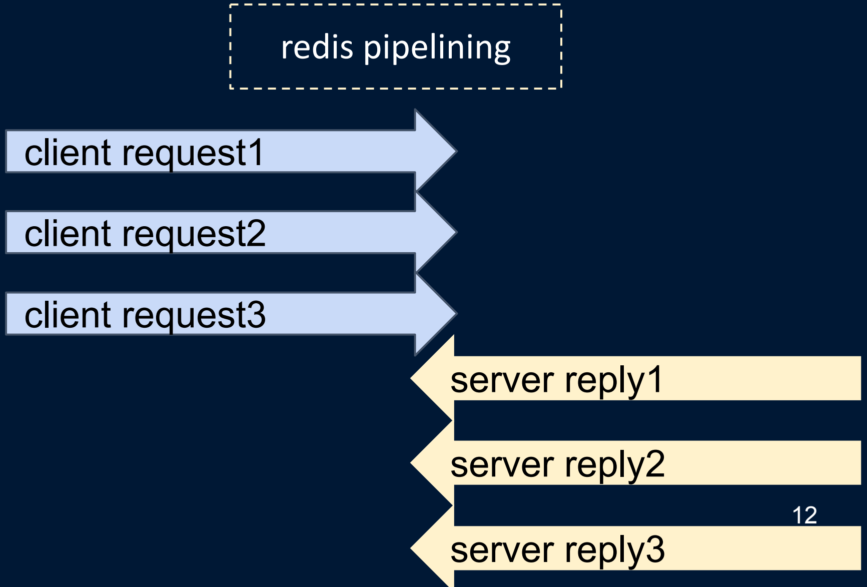
It use the memory available on application server, server that are usually connected compared to the database nodes, to store some subset of the database information directly in the application side.



Redis pipelining

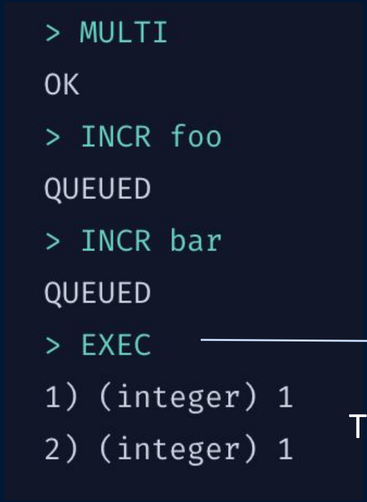
- Redis is a TCP server using client-server with a request/reponse protocol.
- Pipelining is a technique for improving performance by sending multiple commands to the server without waiting for the replies.





Redis transaction

- Redis transaction allow the execution of a group of command that placed between MULTI and EXEC.
- All the commands are serialized and executed sequentially, and redis will guarantee that the command are executed as a single operation.



```
> MULTI
OK
> INCR foo
QUEUED
> INCR bar
QUEUED
> EXEC
1) (integer) 1
2) (integer) 1
```

This will trigger the execution of all the command in the transaction.

The replies will be in the same order the command were executed.

Redis Stack

Redis offers a rich stack which can cater to Big Data processing, advanced aggregation and AI modelling needs. It bundles Redis modules including:

- RedisJSON: Queryable JSON docs
- RediSearch: Querying across hashes and JSON documents
- RedisGraph: Graph data models with the Cypher query language
- RedisTimeSeries: Time series data support (ingestion & querying)
- RedisAI: Executing Deep Learning/Machine Learning models and managing their data.

This stack is supported by client libraries of several languages, e.g. Python, NodeJs, etc
For trying Redis out, the simplest use case is RediSearch.

Data Structures in Redis

Redis is a data structure server. At its core, Redis provides a collection of native data types that help you solve a wide variety of problems, from caching to queuing to event processing. To store data, you need to design a schema that can be modelled by them.

- Strings
- Hashes
- Sets
- Lists

Keys

Redis keys are binary safe, this means that you can use any binary sequence as a key, from a string like "foo" to the content of a JPEG file. The empty string is also a valid key.

Try to stick with a schema. For instance "object-type:id" is a good idea, as in "user:1000".

Dots or dashes are often used for multi-word fields, as in "comment:4321:reply.to" or "comment:4321:reply-to".

The maximum allowed key size is 512 MB.

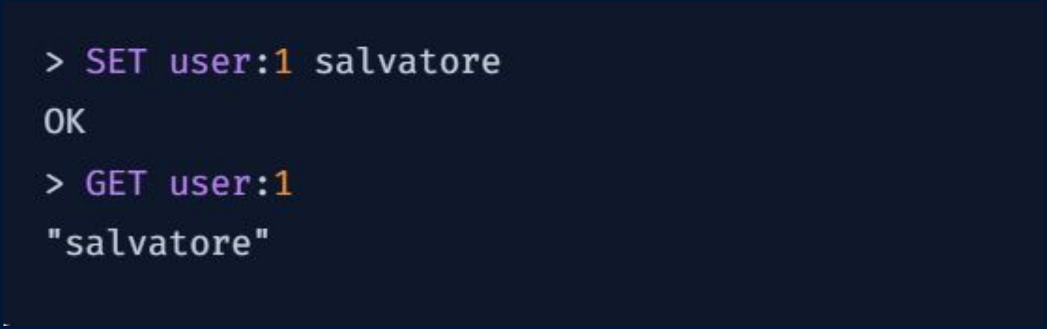
Strings

Redis strings store sequences of bytes, including text, serialized objects, and binary arrays. As such, strings are the most basic Redis data type. They're often used for caching, but they support additional functionality that lets you implement counters and perform bitwise operations, too.

Maximum size of a string is 512 MB.

Strings - Example

Store and then retrieve a string in Redis:



```
> SET user:1 salvatore
OK
> GET user:1
"salvatore"
```

Hashes

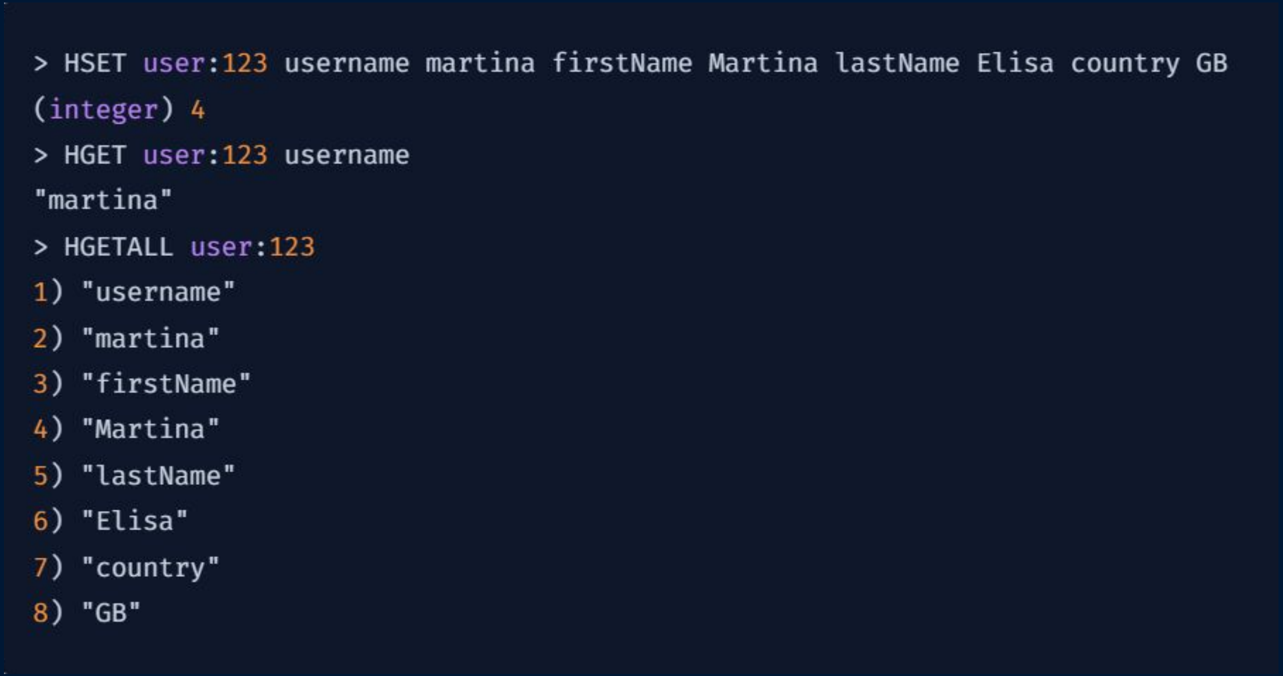
Redis hashes are record types structured as collections of field-value pairs. You can use hashes to represent basic objects and to store groupings of counters, among other things.

Most Redis hash commands are O(1).
A few commands - such as HKEYS, HVALS, and HGETALL - are O(n), where n is the number of field-value pairs.

Every hash can store up to 4,294,967,295 (2^32 - 1) field-value pairs. In practice, your hashes are limited only by the overall memory on the VMs hosting your Redis deployment.

Hashes - User Profile Example

Represent a basic user profile as a hash:



```
> HSET user:123 username martina firstName Martina lastName Elisa country GB
(integer) 4
> HGET user:123 username
"martina"
> HGETALL user:123
1) "username"
2) "martina"
3) "firstName"
4) "Martina"
5) "lastName"
6) "Elisa"
7) "country"
8) "GB"
```

Sets

A Redis set is an unordered collection of unique strings (members). You can use Redis sets to efficiently:

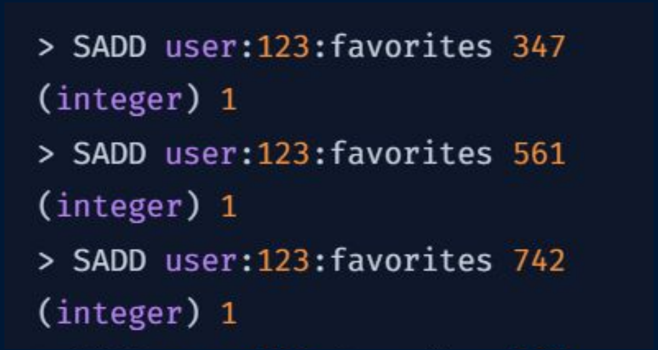
- Track unique items (client IPs e.g.)
- Represent relations (e.g., the set of all users with a given role).
- Perform common set operations such as intersection, unions, and differences.

The max size of a Redis set is 2^32 - 1 (4,294,967,295) members.

Most set operations, including adding, removing, and checking whether an item is a set member, are O(1).

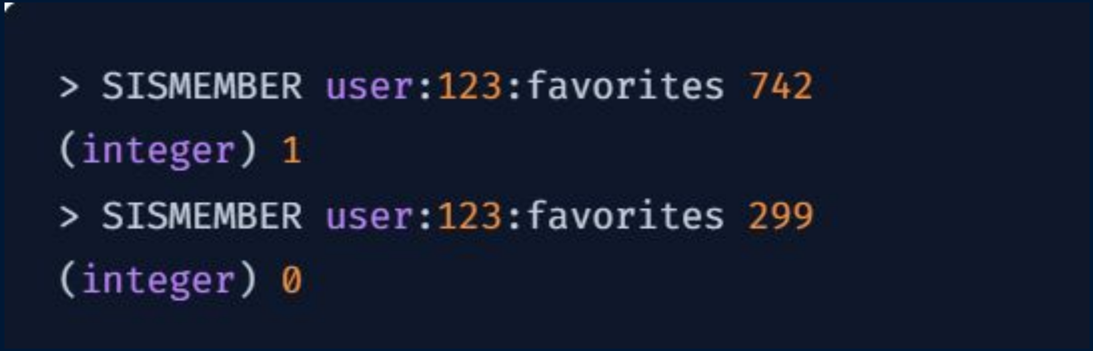
Example

Store the set of favorited book IDs for users 123



```
> SADD user:123:favorites 347
(integer) 1
> SADD user:123:favorites 561
(integer) 1
> SADD user:123:favorites 742
(integer) 1
```

Check whether user 123 likes books 742 and 299



```
> SISMEMBER user:123:favorites 742
(integer) 1
> SISMEMBER user:123:favorites 299
(integer) 0
```

Lists

Redis lists are linked lists of string values. Redis lists are frequently used to:

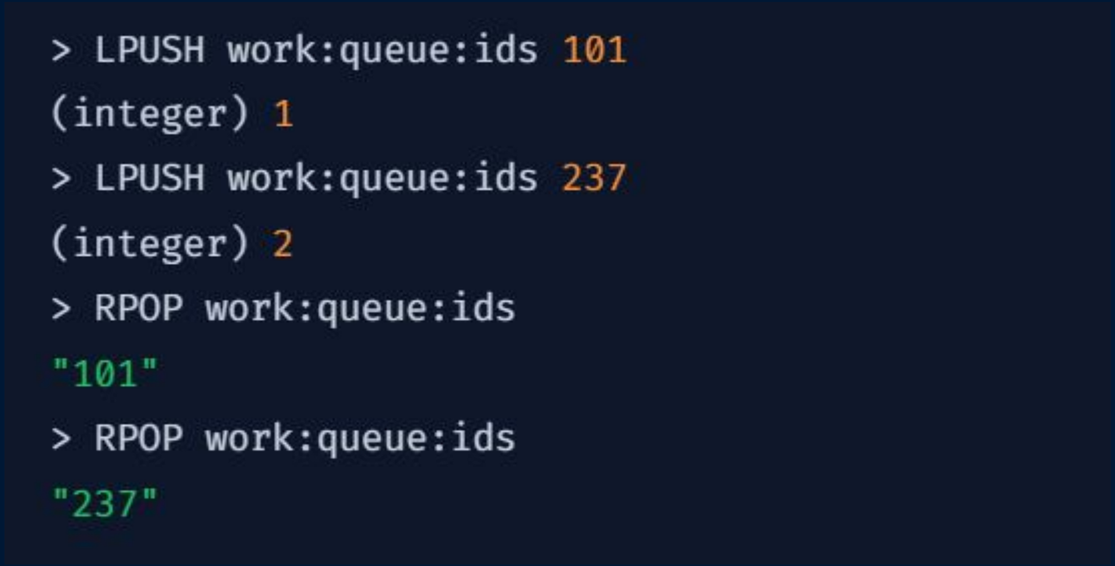
- Implement stacks and queues.
- Build queue management for background worker systems.

The max length of a Redis list is 2^32 - 1 (4,294,967,295) elements.

List operations that access its head or tail are O(1), which means they're highly efficient. However, commands that manipulate elements within a list are usually O(n). Exercise caution when running these commands, mainly when operating on large lists.

Lists - Queue Example

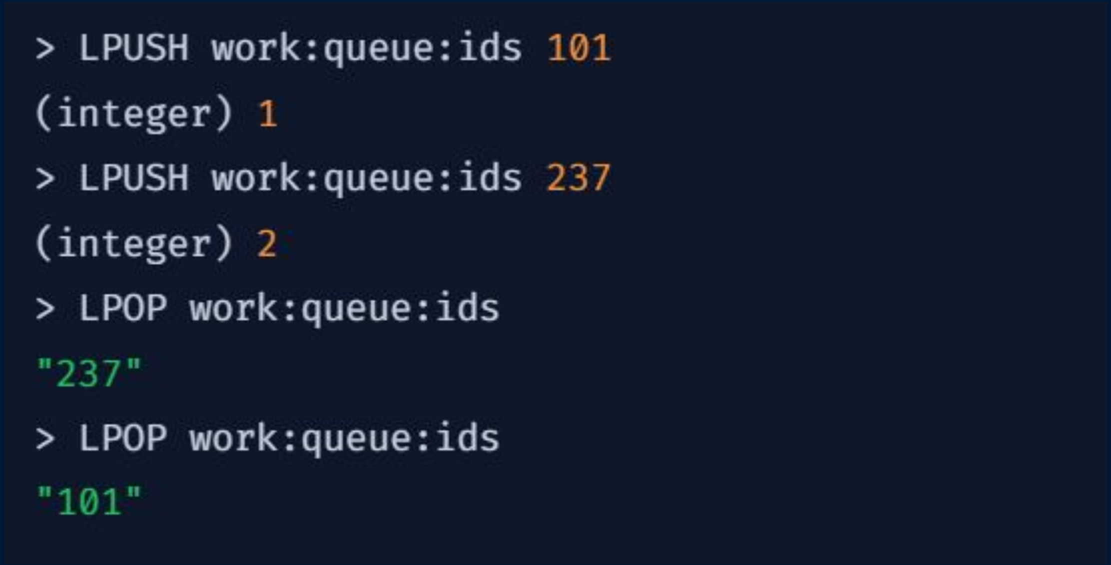
Treat a list like a queue (first in, first out):



```
> LPUSH work:queue:ids 101
(integer) 1
> LPUSH work:queue:ids 237
(integer) 2
> RPOP work:queue:ids
"101"
> RPOP work:queue:ids
"237"
```

Lists - Stack Example

Treat a list like a stack (first in, last out):



```
> LPUSH work:queue:ids 101
(integer) 1
> LPUSH work:queue:ids 237
(integer) 2
> LPOP work:queue:ids
"237"
> LPOP work:queue:ids
"101"
```

Objective

- Understand Redis Database workflow and interaction with the user
- Overview Redis from a scalability and persistence perspective
- Learn about combining a batch of Redis operations
- Briefly explore the various modules offered by the Redis Stack
- Discuss some of the different data structures in Redis and their features

THE UNIVERSITY OF BRITISH COLUMBIA

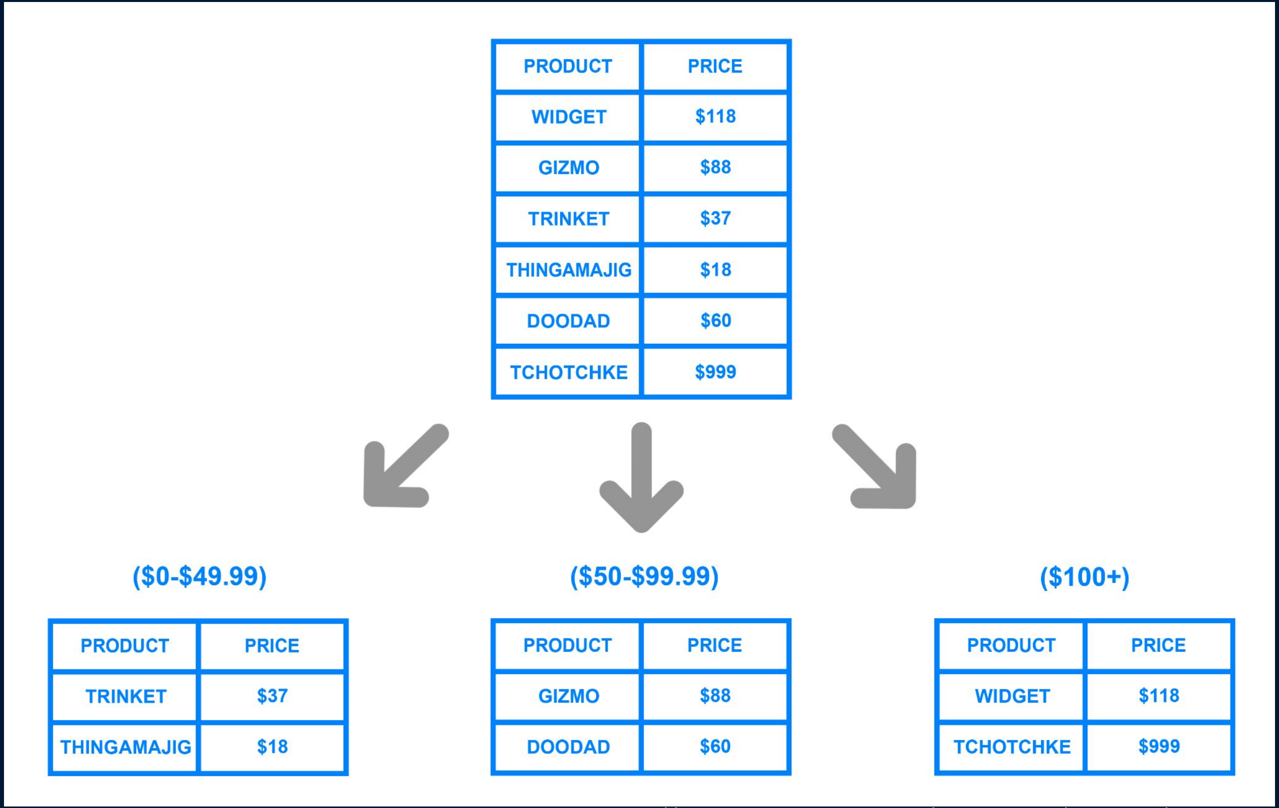
Supplementary : ACID

- ACID is a set of property of database transaction intended to guarantee data validity despite error, power failure and other accidents.

In the context of database, a sequence of database operation that satisfied the ACID properties is called a transaction.

- Atomicity: Redis ensures atomicity because you cannot go to the next transaction till your previous transactions are fully completed.
- Consistency: Redis does not guarantee consistency because Redis is at risk of losing writes.
- Isolation: Redis ensures isolation because transactions are kept separated from each other and ensure that transactions previously are completed before the next transaction goes through.
- Durability : Redis cannot guarantee durability, but when you enable “appendonly” this may increase your Redis database’s durability while sacrificing performance.

Supplementary: Redis shard horizontally



PRODUCT	PRICE
WIDGET	$118
GIZMO	$88
TRINKET	$37
THINGAMAJIG	$18
DOODAD	$60
TCHOTCHKE	$999

($0-$49.99)

PRODUCT	PRICE
TRINKET	$37
THINGAMAJIG	$18

($50-$99.99)

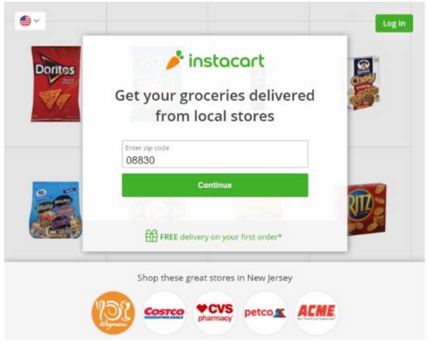
PRODUCT	PRICE
GIZMO	$88
DOODAD	$60

($100+)

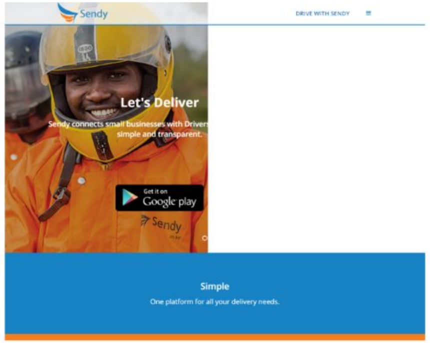
PRODUCT	PRICE
WIDGET	$118
TCHOTCHKE	$999

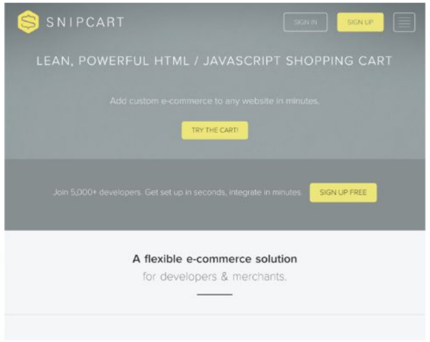
https://www.digitalocean.com/community/tutorials/understanding-database-sharding

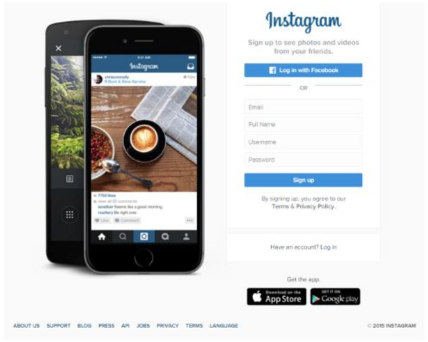
Supplementary : List of company using Redis


instacart
Get your groceries delivered from local stores
Continue
FREE delivery on your first order*
Shop these great stores in New Jersey
www.instacart.com

GitHub
Where software is built
Powerful collaboration, code review, and code management for open source and private projects. Public projects are always free.
Sign up for GitHub
Want to use GitHub on your servers?
github.com


Sendy
Let's Deliver
Sendy connects small businesses with Drivers simple and transparent.
Simple
One platform for all your delivery needs.
sendyit.com

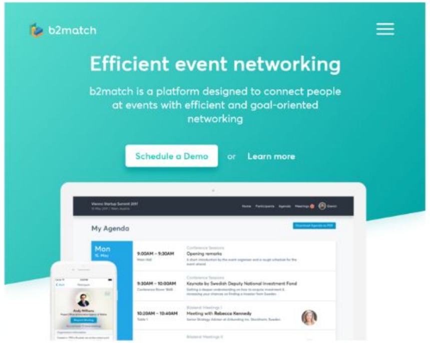

SNIPCART
LEAN, POWERFUL HTML / JAVASCRIPT SHOPPING CART
Add custom e-commerce to any website in minutes.
TRY THE CART!
A flexible e-commerce solution
for developers & merchants.
snipcart.com



instagram.com

shopify
More ways to sell than ever before
Ecommerce · Social Commerce · Buy Button · Point of Sale
Start your free trial
A gorgeous online store
www.shopify.com

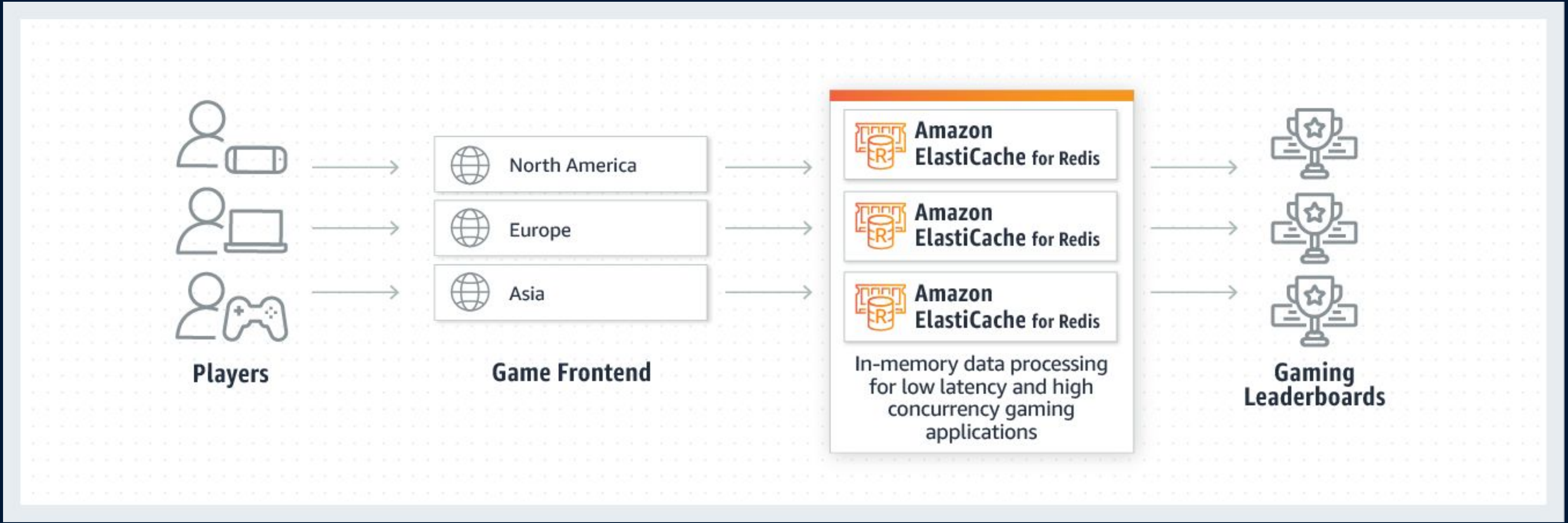
debitoor
Invoicing software & easy accounting
Built for small businesses, sole traders, and freelancers.
Start now for free
debitoor.com


b2match
Efficient event networking
b2match is a platform designed to connect people at events with efficient and goal-oriented networking
Schedule a Demo
Learn more
www.b2match.com

https://techstacks.io/tech/redis

Supplementary: Gaming scenario

Real-time gaming leaderboards are easy to create with Amazon ElastiCache for Redis. Just use the Redis Sorted Set data structure, which provides uniqueness of elements while maintaining the list sorted by their scores. Creating a real-time ranked list is as simple as updating a user's score each time it changes. You can also use Sorted Sets to handle time series data by using timestamps as the score.



North America
Europe
Asia
Players
Game Frontend
Amazon ElastiCache for Redis
Amazon ElastiCache for Redis
Amazon ElastiCache for Redis
In-memory data processing for low latency and high concurrency gaming applications
Gaming Leaderboards

https://aws.amazon.com/elasticache/redis/#Gaming_Leaderboards