Redis

Intro

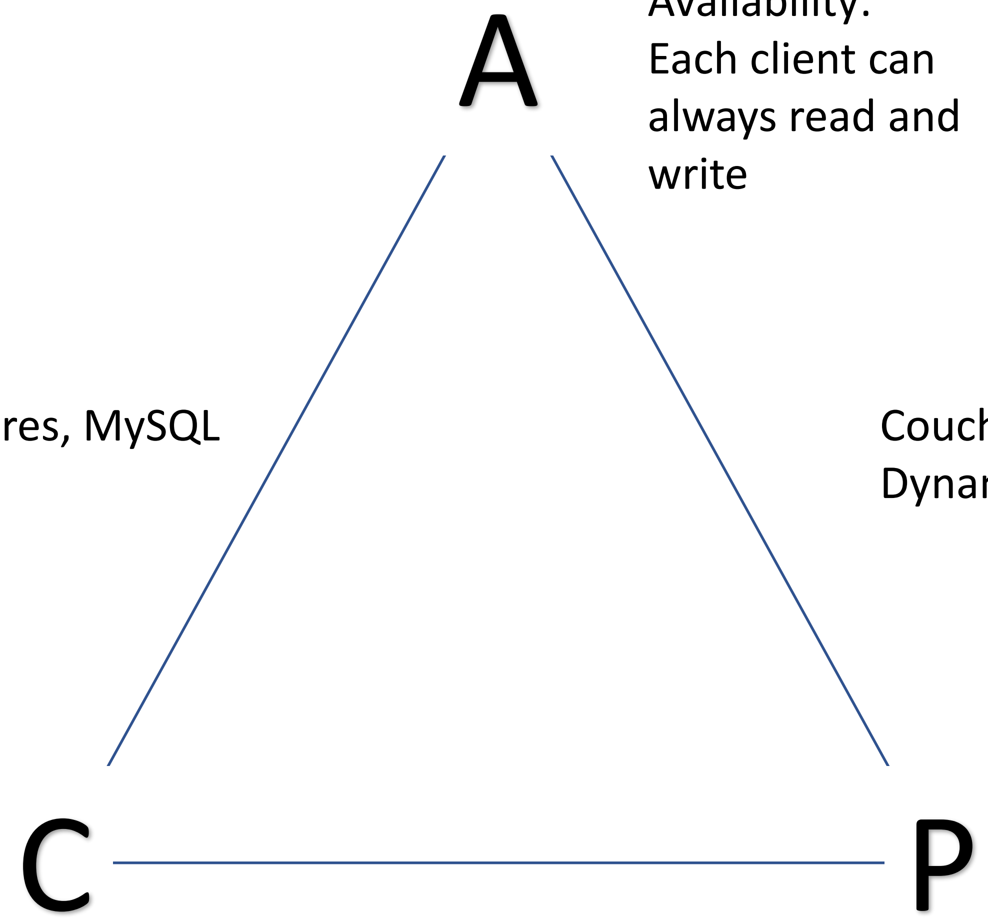
What is Redis

- REmote DIctionary Serve
- No SQL but not like others
- Open Source
- Written in C
- Use Linux its most tested

History

- Early 2009 - Salvatore Sanfilippo, an Italian developer, started the Redis project
- June 2009 - Redis was deployed in production for the LLOOGG real-time web analytics website
- March 2010 - VMWare hired Sanfilippo to work full-time on Redis (remains BSD licensed)

Where does it fit in CAP Theorem



Availability:
Each client can
always read and
write

RDBMS, Postgres, MySQL

CouchDB, Cassandra,
DynamoDB**,** Riak

Consistency: All
clients always
have the same
view of data

Big Table, Hbase, MongoDB, **Redis**

Partition Tolerance:
Works well despite
physical network
partitions

ACID or BASE

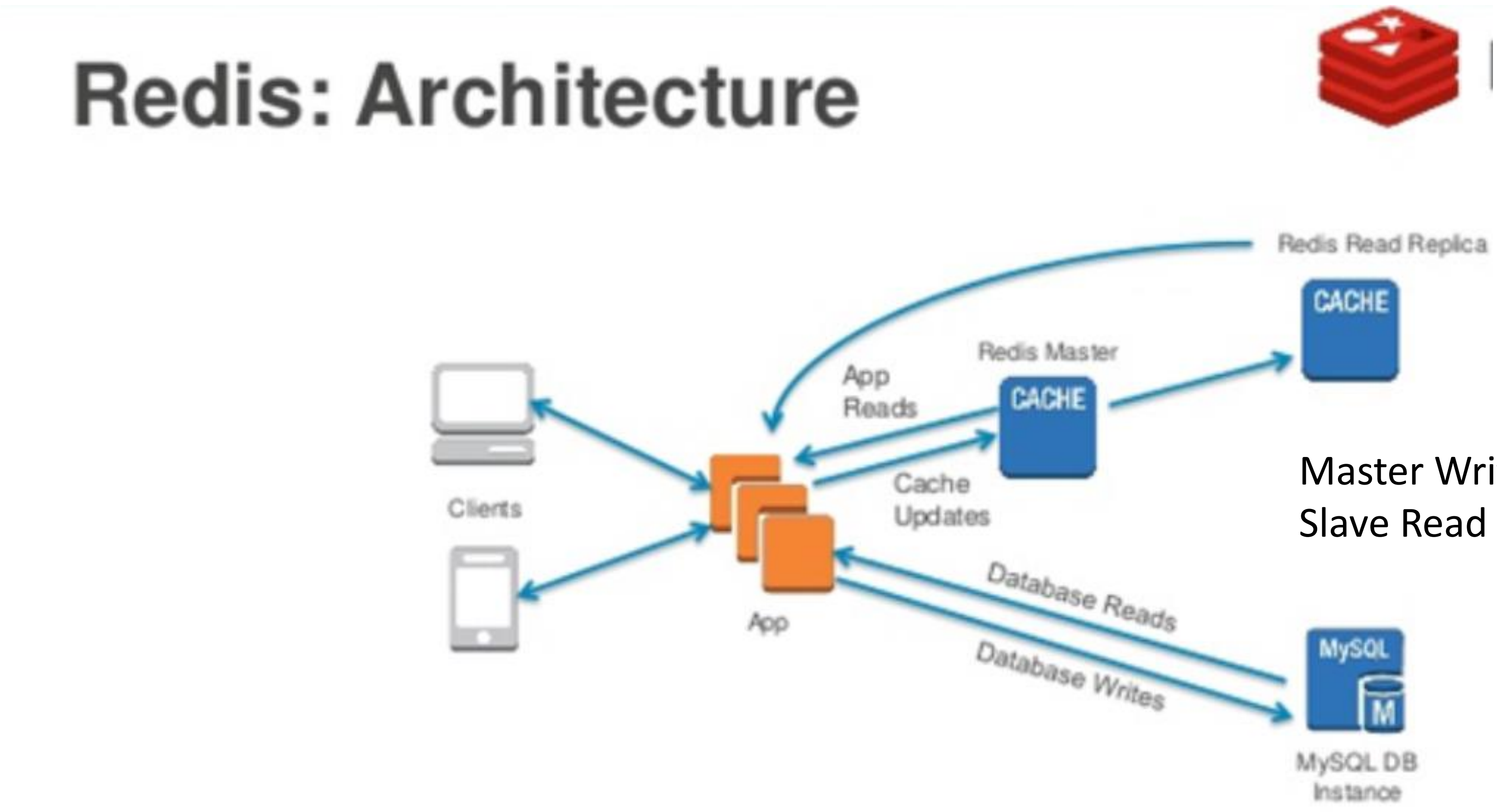
- Atomic: Each task in a transaction succeeds or the entire transaction is rolled back.
- Consistent: A transaction maintains a valid state for the database before and after its completion and cannot leave the database in an inconsistent state.
- Isolated: A transaction not yet committed must not interfere with another transaction and must remain isolated.
- Durable: Committed transactions persist in the database and can be recovered in case of database failure.

- Basically Available: The system is guaranteed to be available in event of failure.
- Soft State: The state of the data could change without application interactions due to eventual consistency.
- Eventual Consistency: The system will be eventually consistent after the application input. The data will be replicated to different nodes and will eventually reach a consistent state. But the consistency is not guaranteed at a transaction level.

Choose K:V Stores if:

1. Simple schema
2. High velocity read/write with no frequent updates
3. High performance and scalability
4. No complex queries involving multiple keys or joins

Redis Architecture



Redis: Architecture

Master Writes
Slave Read Only

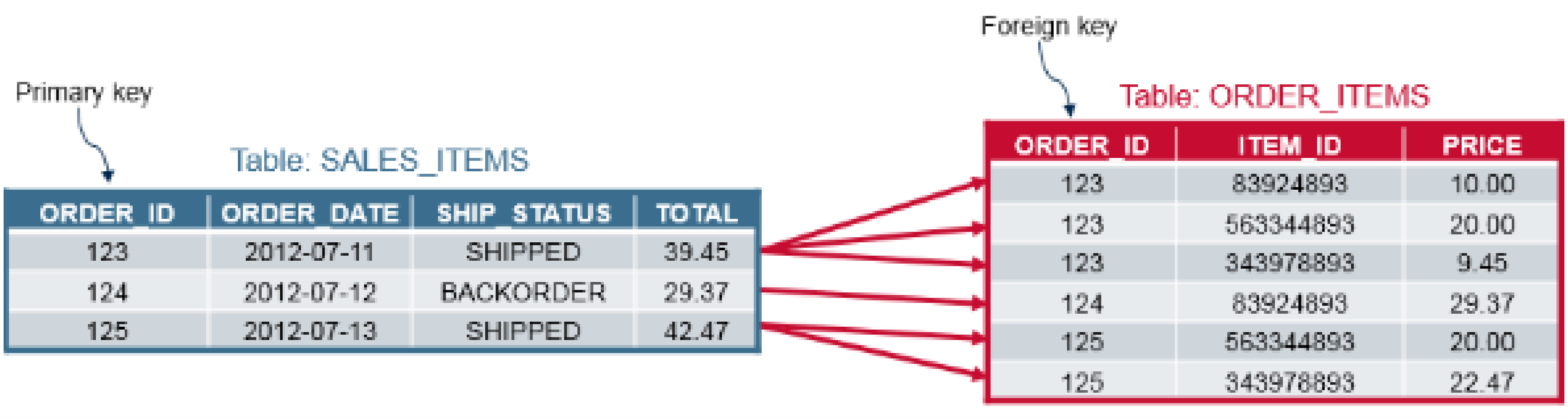
What happens if Redis runs out of memory?

- Redis will either be killed by the Linux kernel OOM killer, crash with an error, or will start to slow down.
- Redis has built-in protections allowing the user to set a max limit to memory usage, using the maxmemory option in the configuration file to put a limit to the memory Redis can use.
- The INFO command will report the amount of memory Redis is using so you can write scripts that monitor your Redis servers checking for critical conditions before they are reached.

Redis is single threaded

- CPU use is typically not an issue
- Memory usually runs out first
- Network capacity limits as well

Normalized Relational



Primary key

Table: SALES_ITEMS

ORDER_ID	ORDER_DATE	SHIP_STATUS	TOTAL
123	2012-07-11	SHIPPED	39.45
124	2012-07-12	BACKORDER	29.37
125	2012-07-13	SHIPPED	42.47

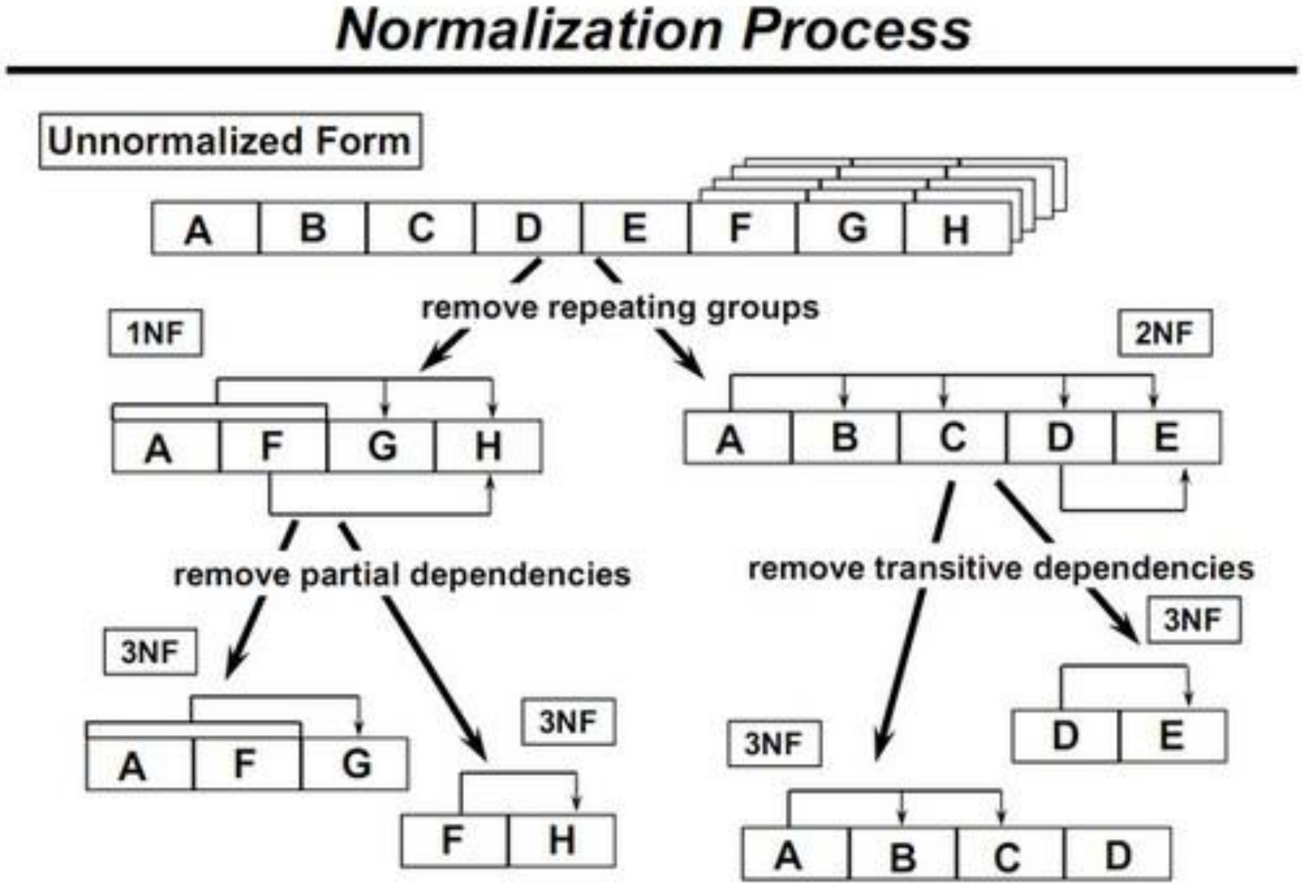
Foreign key

Table: ORDER_ITEMS

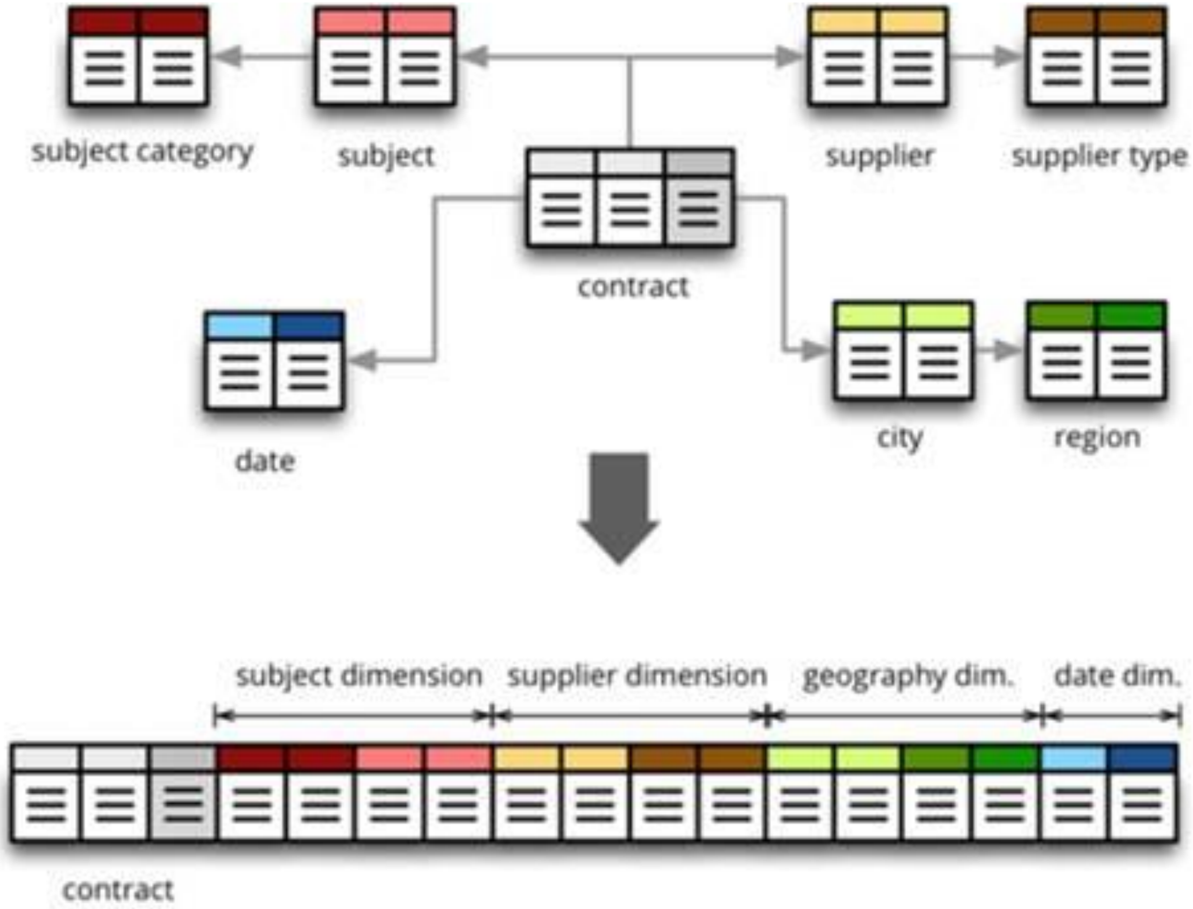
ORDER_ID	ITEM_ID	PRICE
123	83924893	10.00
123	563344893	20.00
123	343978893	9.45
124	83924893	29.37
125	563344893	20.00
125	343978893	22.47

FLORIDA POLYTECHNIC UNIVERSITY

Reverse it


Normalization Process
Unnormalized Form
A
B
C
D
E
F
G
H
remove repeating groups
1NF
A
F
G
H
2NF
A
B
C
D
E
remove partial dependencies
3NF
A
F
G
3NF
F
H
remove transitive dependencies
3NF
D
E
3NF
A
B
C
D

Denormalized

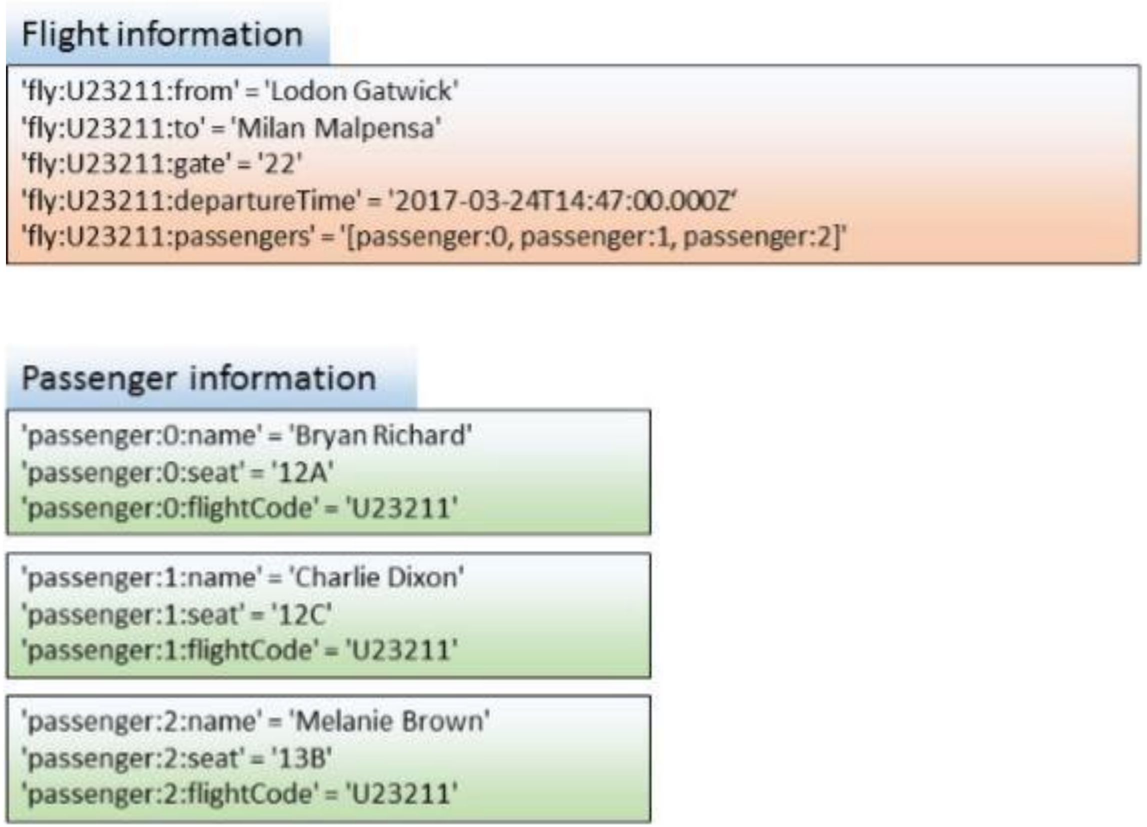


How is Redis Different

- Complex Data types
- All Data resides in memory
    - An empty instance uses ~ 3MB of memory.
    - 1 Million small Keys -> String Value pairs use ~ 85MB of memory.
    - 1 Million Keys -> Hash value, representing an object with 5 fields, use ~ 160 MB of memory.
- Redis is either memory or network bound CPU is usually not a factor

Data Model

- Key
  - Printable ASCII
- Value
  - Primitives
    - Strings
  - Containers (of strings)
    - Hashes
    - Lists
    - Sets
    - Sorted Sets



Flight information

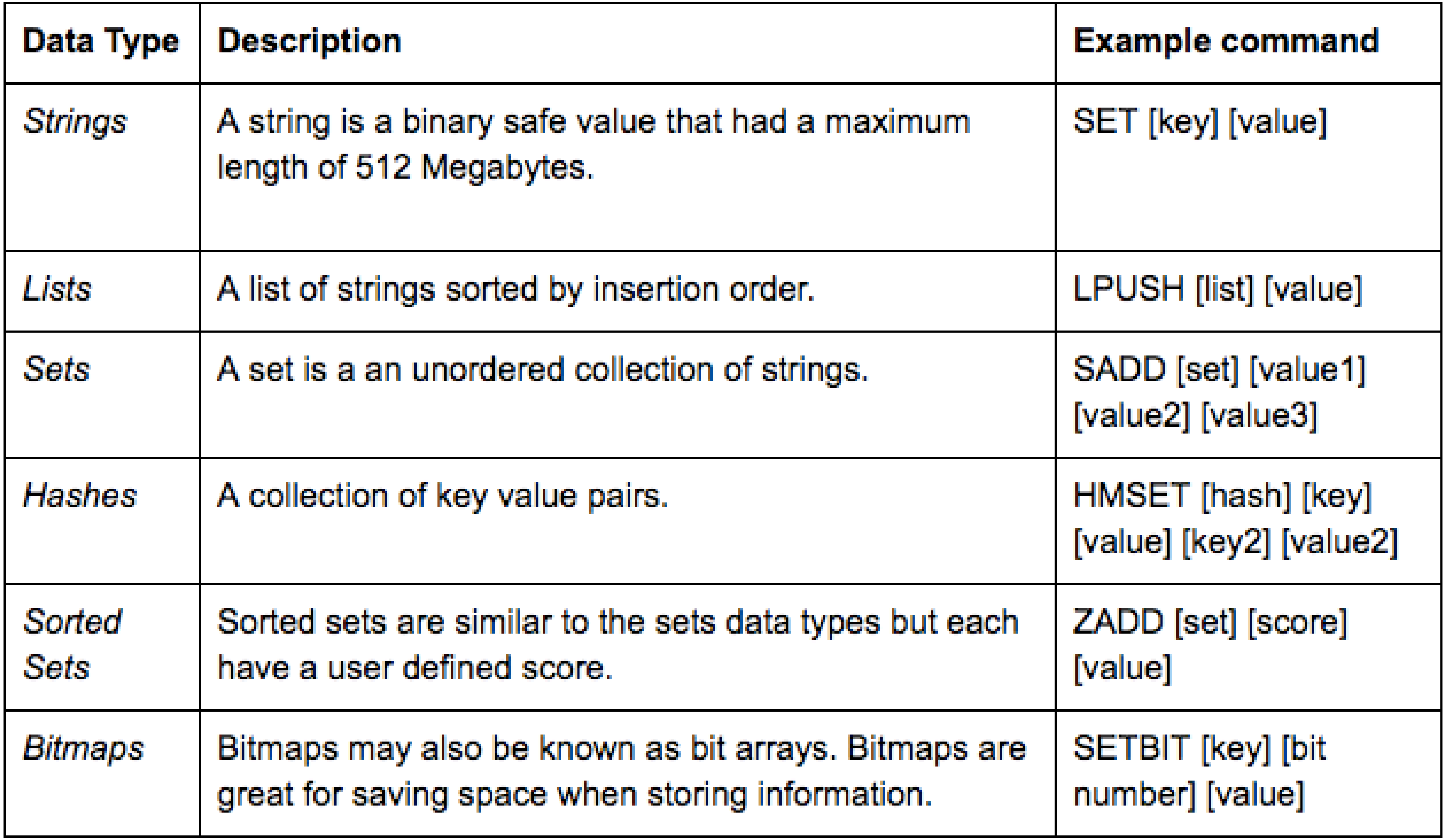
'fly:U23211:from' = 'Lodon Gatwick'
'fly:U23211:to' = 'Milan Malpensa'
'fly:U23211:gate' = '22'
'fly:U23211:departureTime' = '2017-03-24T14:47:00.000Z'
'fly:U23211:passengers' = '[passenger:0, passenger:1, passenger:2]'

Passenger information

'passenger:0:name' = 'Bryan Richard'
'passenger:0:seat' = '12A'
'passenger:0:flightCode' = 'U23211'

'passenger:1:name' = 'Charlie Dixon'
'passenger:1:seat' = '12C'
'passenger:1:flightCode' = 'U23211'

'passenger:2:name' = 'Melanie Brown'
'passenger:2:seat' = '13B'
'passenger:2:flightCode' = 'U23211'



Data Type	Description	Example command
Strings	A string is a binary safe value that had a maximum length of 512 Megabytes.	SET [key] [value]
Lists	A list of strings sorted by insertion order.	LPUSH [list] [value]
Sets	A set is a an unordered collection of strings.	SADD [set] [value1] [value2] [value3]
Hashes	A collection of key value pairs.	HMSET [hash] [key] [value] [key2] [value2]
Sorted Sets	Sorted sets are similar to the sets data types but each have a user defined score.	ZADD [set] [score] [value]
Bitmaps	Bitmaps may also be known as bit arrays. Bitmaps are great for saving space when storing information.	SETBIT [key] [bit number] [value]

Lists

- Maximum length of a list is 2^{32} - 1 elements (4294967295, more than 4 billion of elements per list).

- redis 127.0.0.1:6379> LPUSH database redis (integer) 1 redis
- 127.0.0.1:6379> LPUSH database mongodb (integer) 2 redis
- 127.0.0.1:6379> LPUSH database mysql (integer) 3 redis
- 127.0.0.1:6379> LRANGE database 0 10 1) "mysql" 2) "mongodb" 3) "redis"

Sets

- The maximum length of a list is 2^{32} - 1 elements (4294967295, more than 4 billion of elements per set).
- redis 127.0.0.1:6379> SADD database redis (integer) 1 redis 127.0.0.1:6379> SADD database mongodb (integer) 1 redis 127.0.0.1:6379> SADD database mysql (integer) 1 redis 127.0.0.1:6379> SADD database mysql (integer) 0 redis 127.0.0.1:6379> SMEMBERS database 1) "mysql" 2) "mongodb" 3) "redis"

Hash

- HMSET user:1 id 1 username alexbraunton password a67b9b40f76 email alex@example.com verified Y
- 127.0.0.1:6379> KEYS *
  - 1) "firstkey"
  - 2) "user:1“
- 127.0.0.1:6379> HGETALL user:1
- 1) "id"
- 2) "1"
- 3) "username"
- 4) "alexbraunton"
- 5) "password"

6) "a67b9b40f76"
7) "email"
8) "alex@example.com"
9) "verified"
10) "Y"

Sorted Sets

- Maximum length of a list is 2^{32} - 1 elements (4294967295, more than 4 billion of elements per set).
-
  ```
  redis 127.0.0.1:6379> ZADD database 1 redis (integer) 1 redis
  127.0.0.1:6379> ZADD database 2 mongodb (integer) 1 redis
  127.0.0.1:6379> ZADD database 3 mysql (integer) 1 redis
  127.0.0.1:6379> ZADD database 3 mysql (integer) 0 redis
  127.0.0.1:6379> ZADD database 4 mysql (integer) 0 redis
  127.0.0.1:6379> ZRANGE database 0 10 WITHSCORES 1) "redis" 2)
  "1" 3) "mongodb" 4) "2" 5) "mysql" 6) "4"
  ```

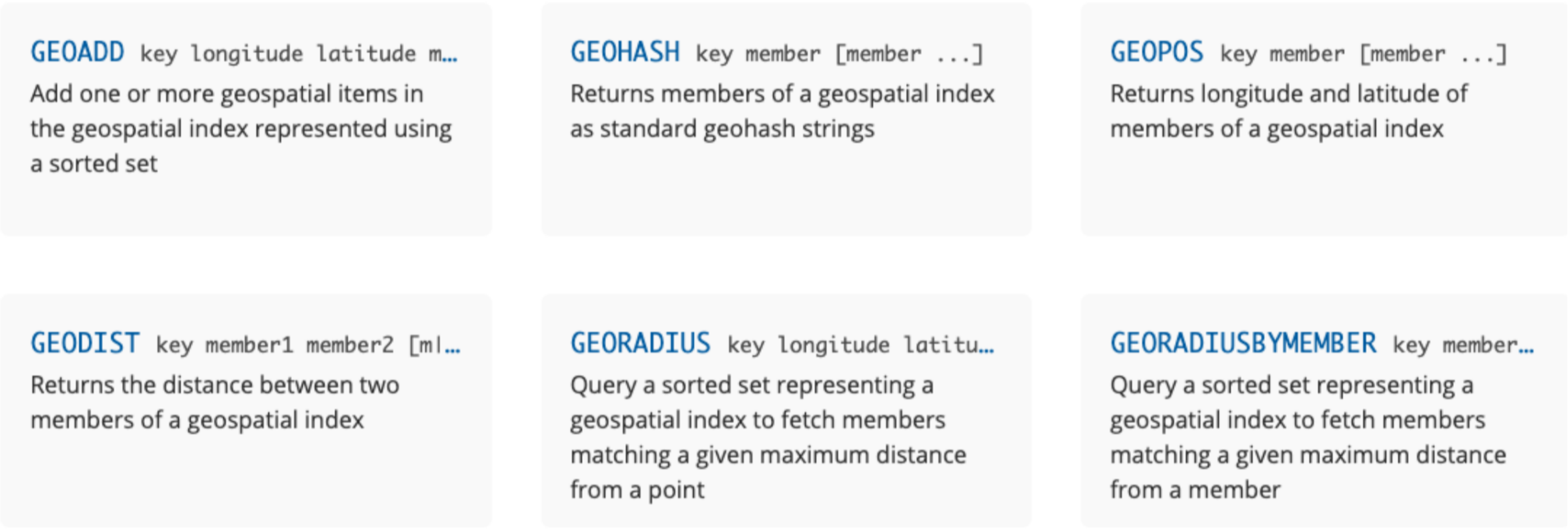
Features

- **Speed**: Redis stores the whole dataset in primary memory that's why it is extremely fast.
- **Persistence**: While all the data lives in memory, changes are asynchronously saved on disk using flexible policies based on elapsed time and/or number of updates since last save.
- **Data Structures**: Redis supports various types of data structures such as strings, hashes, sets, lists, sorted sets with range queries, bitmaps, hyperloglogs and geospatial indexes with radius queries.
- **Supported Languages**: Redis supports a lot of languages such as ActionScript, C, C++, C#, Clojure, Common Lisp, D, Dart, Erlang, Go, Haskell, Haxe, Io, Java, JavaScript (Node.js), Julia, Lua, Objective-C, Perl, PHP, Pure Data, Python, R, Racket, Ruby, Rust, Scala, Smalltalk and Tcl.
- **Master/Slave Replication**: Redis follows a very simple and fast Master/Slave replication.
- **Sharding**: Redis supports sharding.
- **Portable**: Redis is written in ANSI C and works in most POSIX systems like Linux, BSD, Mac OS X, Solaris

Advantages

- **Exceptionally fast** − Redis is very fast and can perform about 110000 SETs per second, about 81000 GETs per second.
- **Supports rich data types** − Redis natively supports most of the datatypes that developers already know such as list, set, sorted set, and hashes. This makes it easy to solve a variety of problems as we know which problem can be handled better by which data type.
- **Operations are atomic** − All Redis operations are atomic, which ensures that if two clients concurrently access, Redis server will receive the updated value. Redis operations working on the different Data Types are atomic, so it is safe to set or increase a key, add and remove elements from a set, increase a counter etc.
- **Multi-utility tool** − Redis is a multi-utility tool and can be used in a number of use cases such as caching, messaging-queues (Redis natively supports Publish/Subscribe), any short-lived data in your application, such as web application sessions, web page hit counts, etc.

GeoSpatial



GEOADD `key longitude latitude m…`

Add one or more geospatial items in the geospatial index represented using a sorted set

GEOHASH `key member [member ...]`

Returns members of a geospatial index as standard geohash strings

GEOPOS `key member [member ...]`

Returns longitude and latitude of members of a geospatial index

GEODIST `key member1 member2 [m|…`

Returns the distance between two members of a geospatial index

GEORADIUS `key longitude latitu…`

Query a sorted set representing a geospatial index to fetch members matching a given maximum distance from a point

GEORADIUSBYMEMBER `key member…`

Query a sorted set representing a geospatial index to fetch members matching a given maximum distance from a member

GeoSpatial

- GEOADD cities -74.0059413 40.7127837 "New York" -118.2436849 34.0522342 "Los Angeles“
- GEOSEARCH cities FROMMEMBER Washington BYRADIUS 500 km WITHCOORD WITHDIST WITHHASH
- GEOSEARCH cities FROMLONLAT -77.0368707 38.9071923 BYRADIUS 500 km WITHCOORD WITHDIST WITHHASH
- GEODIST cities “New York” “Los Angeles” mi can use km or ft
- GEORADIUS cities -74.0059413 40.7127837 100 mi WITHDIST

Customers

Engine Yard

github SOCIAL CODING

guardian.co.uk

digg™

DISQUS

mercado Libre.com

BLIZZARD ENTERTAINMENT

studiVZ

stackoverflow

bump TECHNOLOGIES

eHarmony

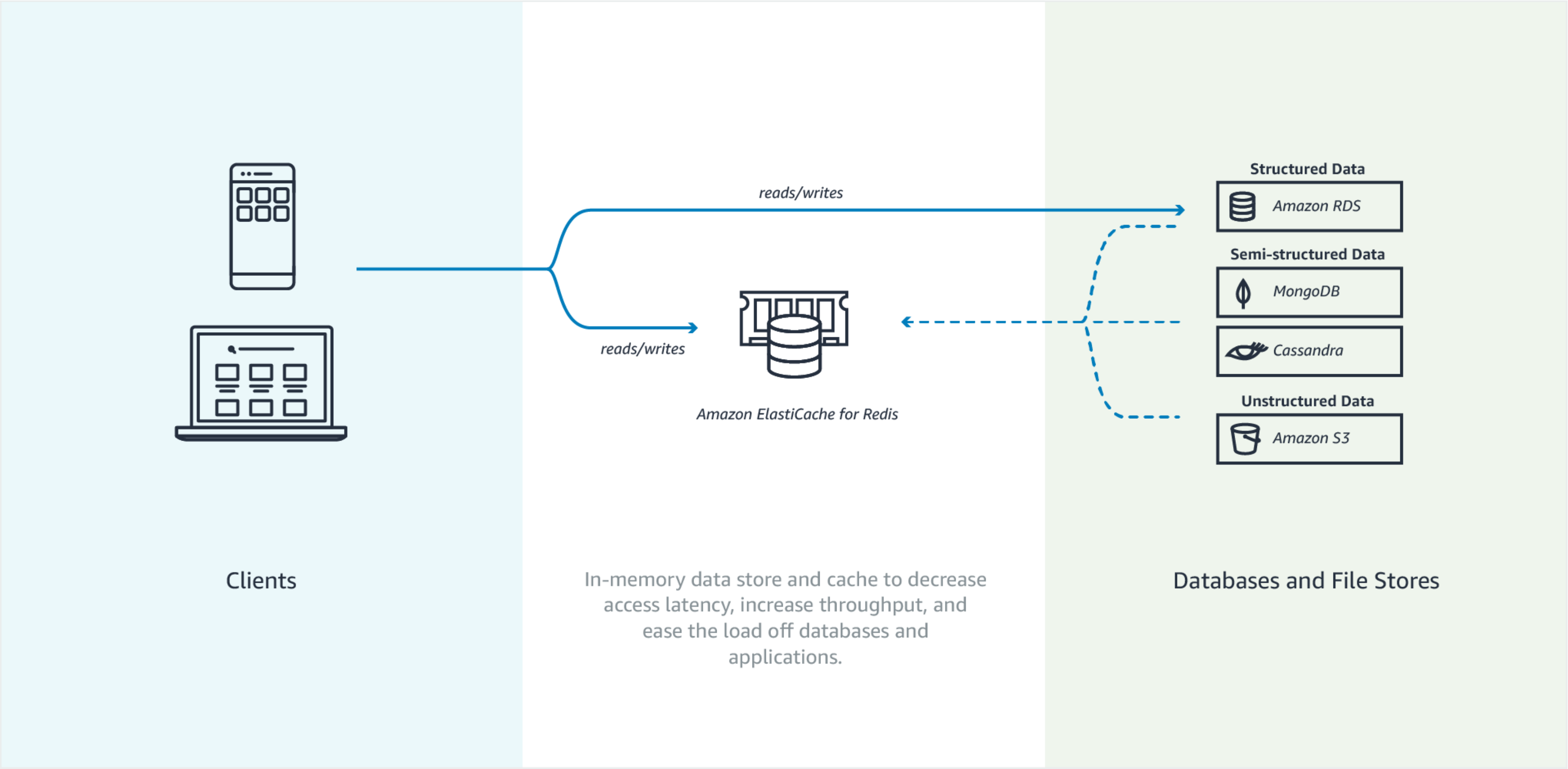
- Makes up to 15 million matches per day and chose Redis Enterprise to achieve a responsive and interactive customer experience, with the least operational overhead.

- Redis Enterprise fulfills many roles – from real-time analytics to supporting low latency match searches, to powering news feeds and profile data – delivering a superior experience to the desktop as well as mobile users. With its automated, seamless scaling and cost-effective low-touch manageability, Redis Enterprise provides a robust technology platform for their team.

What types of workloads?

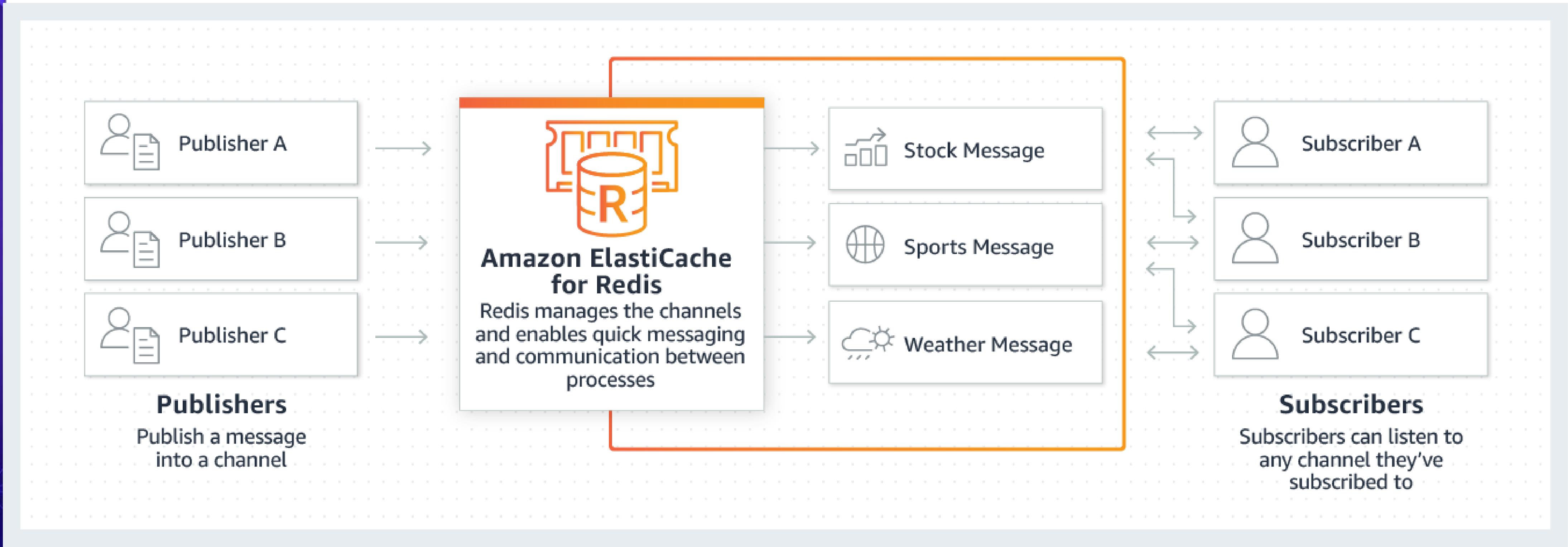
- Page Caching / Session caching / Cookie Storage / Search Engine
- Counter / Analytics / Fraud Detection
- Data that expires / Leaderboards
- Ad targeting
- Messaging

AWS Elasticube Architecture


reads/writes
Structured Data
Amazon RDS
Semi-structured Data
MongoDB
Cassandra
reads/writes
Amazon ElastiCache for Redis
Unstructured Data
Amazon S3
Clients
In-memory data store and cache to decrease access latency, increase throughput, and ease the load off databases and applications.
Databases and File Stores

FLORIDA POLYTECHNIC UNIVERSITY

Chat Messaging



Publisher A

Publisher B

Publisher C

Publishers

Publish a message into a channel

Amazon ElastiCache for Redis

Redis manages the channels and enables quick messaging and communication between processes

Stock Message

Sports Message

Weather Message

Subscriber A

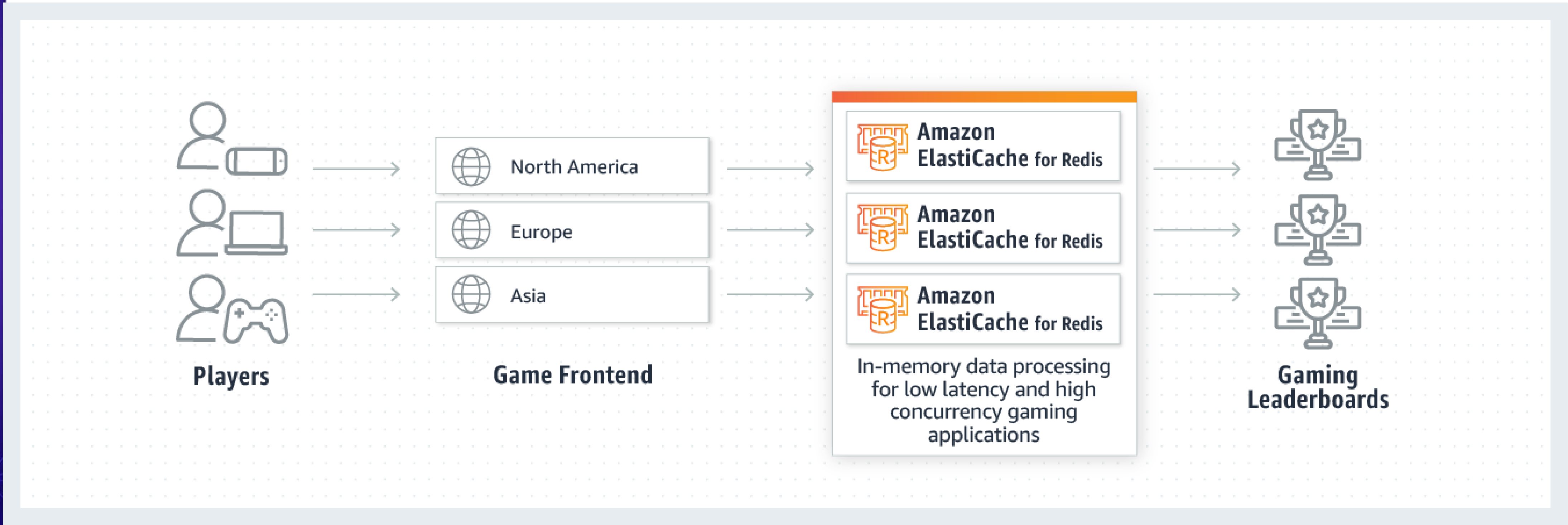
Subscriber B

Subscriber C

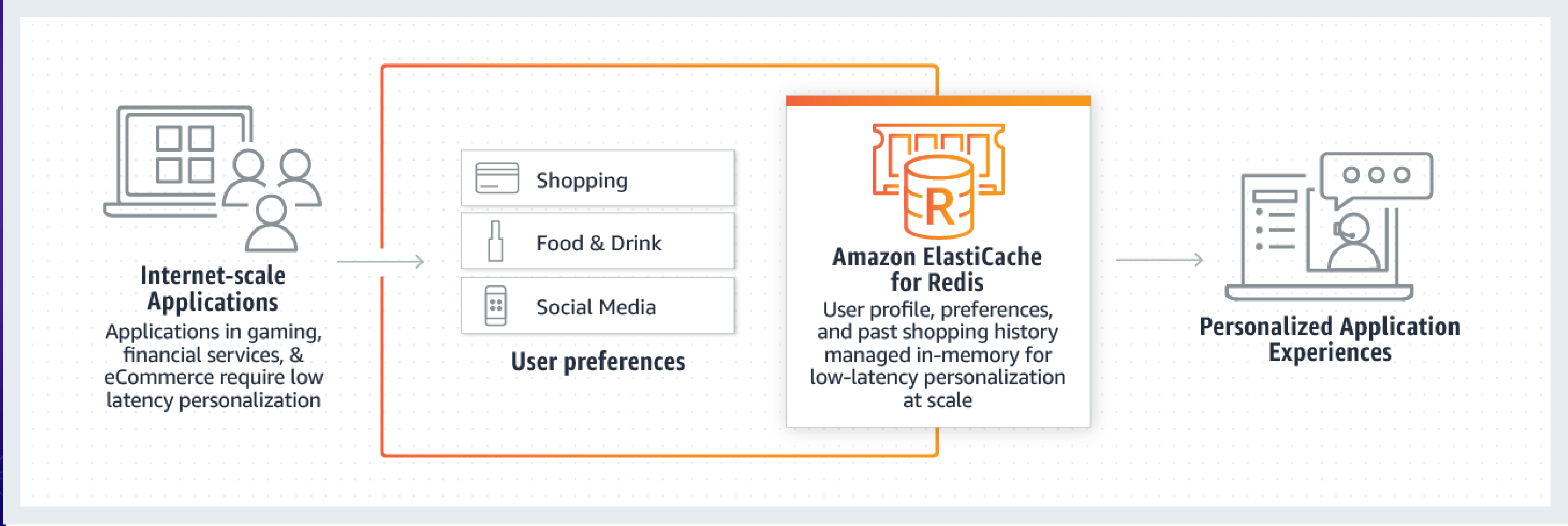
Subscribers

Subscribers can listen to any channel they've subscribed to

Game Leaderboard


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

Session Store


Internet-scale Applications
Applications in gaming, financial services, & eCommerce require low latency personalization
Shopping
Food & Drink
Social Media
User preferences
Amazon ElastiCache for Redis
User profile, preferences, and past shopping history managed in-memory for low-latency personalization at scale
Personalized Application Experiences

FLORIDA POLYTECHNIC UNIVERSITY