**Title:** Redis OM

***Excerpt:*** Redis OM is an object mapper for Redis. It provides client libraries for Java, Node.js, Python, and .NET languages to implement object mapping and querying effectively.

**Permalink:** redis-om

**Category:** Redis

# 

# 

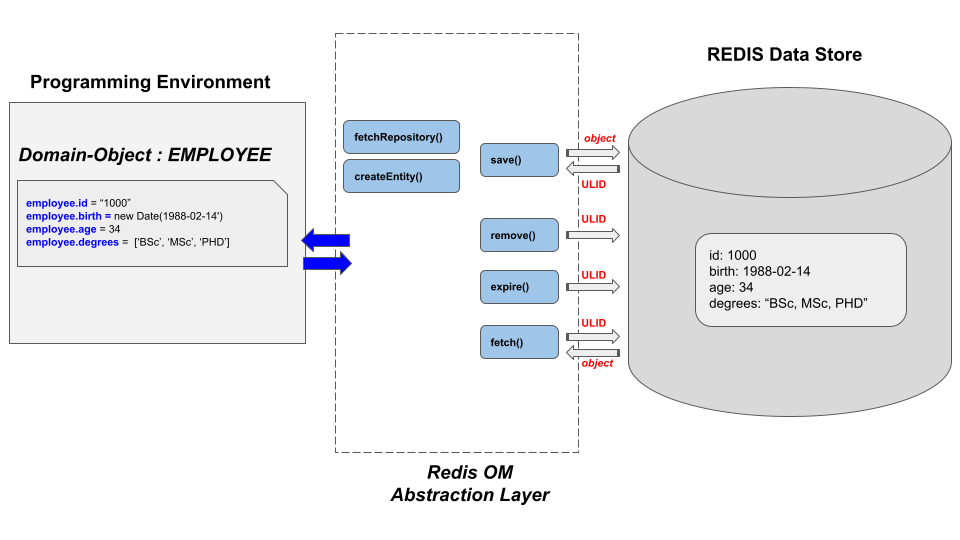
# 

# Introduction to Redis OM

As computer programmers who work with data, it is a regular assignment to arrange data to comply with a specific requirement. Data structures made this task easy for developers by providing proper ways to operate on data. Redis is one of the popular in-memory data stores which supports several robust data structures such as hashes, sorted sets, sets, lists, streams, and etc. This opened the doors for implementing advanced tools like message queues, caches, leaderboards, pub/sub systems, and etc with reduced cost and more efficiently.

The common problem is that it is costly and time-consuming to implement the above tools from scratch. So, the development team has to come up with an abstraction layer to facilitate the mapping between the core Redis data structures and the programming language they use. Even if a custom abstraction layer is implemented, it might introduce a considerable performance lag. Due to that, you will not be able to get the maximum performance out of Redis data structures.

Redis OM is an object mapper for Redis that helps developers to work with Redis data structures within their programming environment without additional effort. It provides a higher-level abstraction layer for object mapping. Usually, a specific domain is modeled in an object-oriented way. So, Redis OM provides a set of language-specific client libraries which can be used to persist domain objects in the Redis data store and fetch those with a language-specific API.



# Domain Object Mapping with Redis OM

The object mapping capability is one of the major features that come with Redis OM. The Redis OM abstraction layer makes consumer life easy by providing a rich language-specific API. It involves a simple, less time consuming process to follow.

Let’s assume that we are working in an ERP domain and need to store employees in the Redis store. With Redis OM in place, first, we need to define our ‘Employee’ schema as follows. We will be using the OM client library for Node.js here.

| class Employee extends Entity {} |
| --- |

| const schema = new Schema(Employee, {  id: { type: 'string' },  age: { type: 'number' },  degrees: { type: 'string[]' } }); |
| --- |

The ‘Entity’ and ‘Schema’ classes are coming from the OM abstraction layer.

Next, an employee instance can be created as follows.

| const employeeObj = employeeRepository.createEntity() employeeObj.id = "1000" employeeObj.age = 45 employeeObj.degrees = ['BSc', 'MSc', 'PHD']  const employeeObjId = await employeeRepository.save(album) |
| --- |

The ‘employeeObjId’ holds the ULID of the created object that is URL-safe, globally unique, and base32-encoded. It is more similar to the usual UUIDs. Furthermore, the domain entities are serialized in two different ways. Basic objects with a uniform structure will be serialized to a Redis hash. The complex objects with tens of fields and nested entities are converted to JSON structure using the RedisJSON module.

# Querying Domain Objects with Redis OM

Redis OM provides a query API to fetch the persistent objects in the Redis store. This query API uses the RedisJSON and RedisSearch modules at its base implementation to provide indexing and querying for domain objects. The performance of the querying is efficient because they’re indexed by default.

We can query all the employee objects in the Redis store using the Node.js OM API as follows.

| employeeRepository.search().return.all() |
| --- |

Furthermore, the querying can be done in different ways using filtering options provided by Redis OM fluent query API. The ‘where’ clause can be used as follows.

| employeeRepository.search().where(<attribute>).eq(<value>).return.all()  employeeRepository.search().where(<attribute>).gt(<value>).return.all()  employeeRepository.search().where(<attribute>).lt(<value>).return.all()  employeeRepository.search().where(<attribute>).eq(<value>).return.all() |
| --- |

It releases a lot of burdens away from developers by focusing more on application development.

# Redis OM Client Libraries

By the time of writing this article, four main Redis OM client libraries have been released for Java spring, .NET, Node.js, and Python languages. The Node.js library is implemented using typescript and it supports both the typescript and javascript domains. The .NET specific OM library enables .NET developers to map and query domain objects using LINQ. Furthermore, the Redis OM python library integrates with the FastAPI framework to provide synchronous and asynchronous support. Java’s popular spring framework has been integrated with the Redis OM to provide probabilistic data structures.

# Conclusion

In short, Redis OM provides a set of language-specific client libraries to map your domain objects to Redis data structures. It is an abstraction layer that helps you to focus more on your application logic without worrying about how to map a Java or Node.js class to a Redis hash. As indicated, Redis OM focuses on object mapping and querying. Domain objects are serialized into hashes or JSON structures based on the complexity of the object. On the other hand, the querying API uses the RedisJSON and RedisSearch modules. To end, Redis OM supports Java, Node.js, Python, and .NET client libraries.