**Caching**

Caching is the process of storing frequently accessed or computationally expensive data in a temporary storage area, called a cache, to speed up subsequent access to that data. The cache is typically located closer to the consumer of the data, such as within the same system or on a faster storage medium, compared to the original source of the data.

**Data Retrieval**: When a system or application needs certain data, it first checks if it's available in the cache.

**Cache Hit**: If the data is found in the cache (known as a cache hit), it can be retrieved much faster than if it had to be fetched from the original source, such as a database or a remote server.

**Cache Miss**: If the data is not found in the cache (known as a cache miss), the system fetches the data from the original source and stores it in the cache before returning it to the requester.

**REDIS**

It's an in-memory data structure store, used as a database, cache, and message broker.  
Redis is known for its high performance, scalability, and versatility.

**Key Features of Redis**

In-memory storage: Redis primarily stores data in RAM, which allows for extremely fast read and write operations.

Persistence options: Although Redis primarily stores data in memory, it provides different persistence options to ensure data durability, including snapshotting and journaling.

Replication and high availability: Redis supports master-slave replication, allowing for data to be replicated across multiple nodes, ensuring high availability and fault tolerance.

Pub/Sub messaging: Redis includes support for Publish/Subscribe messaging, allowing for the implementation of message brokers and real-time communication systems.

Atomic operations: Redis commands are atomic, meaning they either succeed or fail completely, ensuring data integrity.

**Note: It does offer persistence options that allow you to permanently store data, similar to traditional databases. Redis provides two main mechanisms for persistence:**

Snapshotting: Redis can create point-in-time snapshots of the dataset at regular intervals. These snapshots are written to disk as RDB (Redis DataBase) files. You can configure Redis to take snapshots automatically based on time or changes to the dataset.

Append-only file (AOF) logging: Redis can also log every write operation to an append-only file. This file contains a log of all write operations performed on the dataset. The AOF file can be replayed to reconstruct the dataset in case of server restart or failure.

**Why Data Retrieval is faster in Redis**

**Access Speed, NO Disk I/O, Reduced Latency**

**Redis With Spring Boot**

1. **Third Party Libraries**
2. **Use CRUD Repository**
3. **We Can go with the annotations**

**Third Party Libraries:**

1. **Jedis:** Jedis is a Java client library for Redis that provides a simple and straightforward API for interacting with Redis. It directly communicates with the Redis server using blocking I/O operations.
2. **Lettuce:** Lettuce is a high-performance Redis client library for Java that provides non-blocking and asynchronous I/O operations, making it suitable for high-concurrency applications. Lettuce is designed to be lightweight and scalable, leveraging features such as connection pooling and command pipelining to achieve high throughput and low-latency Redis interactions.It supports reactive programming models and integrates well with reactive frameworks like Spring WebFlux.Lettuce is a good choice if you need high-performance Redis interactions or if you're building reactive applications.

**Note:** While both Jedis and Lettuce can be used to interact with Redis from a Java application, the choice between them depends on factors such as performance requirements, concurrency model, and preferred programming style. If you're already using Spring Data Redis for your Spring Boot application, you might not need to use Jedis or Lettuce directly, as Spring Data Redis provides integration with both of these libraries under the hood.

**RedisTemplate**

The RedisTemplate is a central class in the Spring Data Redis module. It provides a high-level abstraction for interacting with Redis data structures and commands within a Spring application. The RedisTemplate simplifies the integration of Redis into Spring applications by encapsulating common operations and providing a consistent programming interface.

**CRUD Repository for Redis:** Same as JPA

**Redis Template vs Repository**

Use RedisTemplate when:

* You need fine-grained control over Redis operations and want to execute Redis commands directly.
* Your application interacts with Redis in a more low-level manner, requiring flexibility and customization.

Use Spring Data Redis repositories when:

* You prefer a higher level of abstraction and want to work with domain-specific objects rather than Redis commands.
* Your application follows a more traditional CRUD (Create, Read, Update, Delete) pattern and can benefit from automatic mapping between Java objects and Redis data structures

**Annotation Based:**This approach utilizes caching annotations such as @Cacheable, @CachePut, and @CacheEvict to add caching capabilities to your Spring components (typically service methods).