**12. Понятие in-memory Database**

In-memory база данных (или in-memory хранилище) - это база данных, которая хранит свои данные в оперативной памяти (RAM), в отличие от традиционных баз данных, которые хранят данные на диске. Использование оперативной памяти для хранения данных позволяет обеспечить очень высокую скорость доступа к данным, поскольку оперативная память обладает очень низким временем доступа. In-memory базы данных широко используются в приложениях, где требуется высокая производительность и скорость обработки данных, например, в кэширующих системах, системах реального времени, аналитических системах и т. д.

**13. Понятие хэш-таблица**

Хэш-таблица - это структура данных, которая реализует ассоциативный массив, а именно, структуру данных, позволяющую хранить пары "ключ-значение" и выполнять операции добавления, удаления и поиска элементов по ключу. Основная идея хэш-таблицы заключается в использовании хэш-функции для вычисления индекса (хэша) для каждого ключа, что позволяет достигнуть очень быстрого доступа к данным. В идеальном случае, при использовании хорошей хэш-функции, время доступа к элементу в хэш-таблице является постоянным (O(1)), что делает хэш-таблицу очень эффективной для реализации различных алгоритмов и структур данных.

**14. Персистентность данных в СУБД Redis**

Redis обеспечивает персистентность данных с помощью механизма записи журнала транзакций и периодической записи снимков (snapshots) данных на диск. Журнал транзакций (transaction log) записывает все операции записи (например, set, incr, del) в журнальный файл, а периодические снимки (snapshots) представляют собой полные копии данных в определенные моменты времени. В случае сбоя, Redis использует записи журнала транзакций для восстановления данных, а также может использовать последний снимок данных для ускорения процесса восстановления.

**15. Назначение команд СУБД Redis: set, get, del, getset**

* **SET key value**: устанавливает значение **value** для ключа **key**.
* **GET key**: возвращает значение, связанное с ключом **key**.
* **DEL key [key ...]**: удаляет ключи **key**.
* **GETSET key value**: устанавливает значение **value** для ключа **key** и возвращает старое значение, связанное с ключом **key**.

Эти команды позволяют устанавливать, получать, удалять и обновлять значения ключей в Redis.

**16. Назначение команд СУБД Redis: incr, decr, incrBy, decrBy**

* **INCR key**: увеличивает значение ключа **key** на 1.
* **DECR key**: уменьшает значение ключа **key** на 1.
* **INCRBY key increment**: увеличивает значение ключа **key** на **increment**.
* **DECRBY key decrement**: уменьшает значение ключа **key** на **decrement**.

Эти команды позволяют выполнять атомарные операции инкремента и декремента на целочисленных значениях ключей в Redis.

**17. Назначение команд СУБД Redis: mset, mget**

* **MSET key value [key value ...]**: устанавливает несколько пар ключ-значение одновременно.
* **MGET key [key ...]**: возвращает значения для нескольких ключей.

Эти команды позволяют работать с несколькими ключами и значениями одновременно.

**18. Назначение команд СУБД Redis: hset, hget**

* **HSET key field value**: устанавливает значение **value** для поля **field** в хэше, связанном с ключом **key**.
* **HGET key field**: возвращает значение поля **field** из хэша, связанного с ключом **key**.

Эти команды позволяют устанавливать и получать значения полей в хэшах Redis.

**19. Назначение команд СУБД Redis: hmset, hmget**

* **HMSET key field value [field value ...]**: устанавливает несколько полей и их значений одновременно в хэше, связанном с ключом **key**.
* **HMGET key field [field ...]**: возвращает значения для нескольких полей из хэша, связанного с ключом **key**.

Эти команды позволяют работать с несколькими полями и их значениями одновременно в хэшах Redis.

**20. Назначение команды СУБД Redis exists**

* **EXISTS key**: проверяет наличие ключа **key** в базе данных.

Эта команда позволяет проверить, существует ли ключ в базе данных Redis.

**21. Назначение команды СУБД Redis multi**

* **MULTI**: начинает транзакцию, в рамках которой можно выполнять несколько команд, которые будут выполнены как единое атомарное действие.
* **EXEC**: выполняет все команды, добавленные в транзакцию с помощью **MULTI**.

Эти команды позволяют выполнять множество операций как одну транзакцию, что гарантирует атомарность выполнения операций.

**22. Назначение метода sendCommand**

Метод **sendCommand** в библиотеке Node.js для Redis позволяет отправлять произвольные команды Redis на сервер Redis. Этот метод предоставляет возможность отправлять команды, которые не предусмотрены встроенными методами библиотеки, или для отправки специфических команд, которые необходимы в вашем приложении. Метод **sendCommand** принимает имя команды Redis и массив аргументов, связанных с этой командой.