**Вопросы к лабораторной работе №12**

1. Поясните понятие in-memory Database.

База данных в памяти ([англ.](https://ru.bmstu.wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) In-memory database, IMDB) — это [система управления базами данных](https://ru.bmstu.wiki/%D0%A1%D0%A3%D0%91%D0%94_(%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D0%A3%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F_%D0%91%D0%B0%D0%B7%D0%B0%D0%BC%D0%B8_%D0%94%D0%B0%D0%BD%D0%BD%D1%8B%D1%85)), которая при хранении компьютерных данных в основном полагается на [оперативную память](https://ru.bmstu.wiki/RAM_(Random_Access_Memory)). Она выделяется на фоне СУБД, в которых используется механизм дискового хранения. Базы данных в памяти быстрее, чем базы данных, оптимизированные для использования дисковых накопителей, поскольку доступ к диску медленнее, чем доступ к памяти, а внутренние алгоритмы оптимизации проще и выполняют меньше инструкций [ЦП](https://ru.bmstu.wiki/CPU_(Central_Processing_Unit)). Доступ к данным из памяти исключает время на поиск при их запросе, что обеспечивает более быструю и предсказуемую работу, чем при использовании дискового накопителя.

Базы данных в памяти предназначены для достижения минимального времени отклика, а также чрезвычайно высокой пропускной способности для систем, критичных к производительности. Это возможно потому, что данные сохраняются и обрабатываются в форме, используемой приложением, что устраняет накладные расходы, связанные с переводом и кэшированием. Технология IMDB способна поддерживать развертывание на уровне приложений, управление данными в режиме реального времени и большинство свойств ACID.

1. Поясните понятие хэш-таблица.

Хеш-таблицей называется структура данных, предназначенная для реализации ассоциативного массива, такого в котором адресация реализуется посредством хеш-функции. Хеш-функция – это функция, преобразующая ключ key в некоторый индекс i равный h(key), где h(key) – хеш-код (хеш-сумма, хеш) key. Весь процесс получения индексов хеш-таблицы называется хешированием.

1. Каким образом обеспечивается персистентность данных в СУБД Redis?

**Персистентность**. Redis сохраняет снимки базы данных на диск. Можно настроить период сохранения данных в зависимости от количества обновленных значений. Также можно использовать режим дозаписи. Для случаев, когда, например, Redis используется в качестве кэша, сохранение на диск можно отключить вовсе.

1. Поясните назначение команд СУБД Redis: set, get, del, getset.

**set**: устанавливает key для хранения строки value. Если key уже существует, его value перезаписывается, независимо от его типа. Любое предыдущее время жизни, связанное с ключом, отбрасывается при успешной операции набора.

**get**: получить value по key. Если ключ не существует, то возвращается специальное значение nil. Ошибка возвращается, если значение, хранящееся в key файле, не является строкой, поскольку [GET](https://redis.io/commands/get) обрабатывает только строковые значения.

**getset**: атомарно устанавливает в key новое значение value и возвращает старое значение, хранящееся в key. Возвращает ошибку, если key существует, но не содержит строкового значения.

**del**: удаляет указанные ключи. Ключ игнорируется, если он не существует.

1. Поясните назначение команд СУБД Redis: incr, decr, incrBy, decrBy

**incr:** увеличивает число, хранящееся в key на единицу. Если ключ не существует, он устанавливается в 0 значение перед выполнением операции. Ошибка возвращается, если ключ содержит значение неправильного типа или содержит строку, которая не может быть представлена в виде целого числа. Эта операция ограничена 64-битными целыми числами со знаком.

**decr:** уменьшает число, хранящееся в key на единицу. Если ключ не существует, он устанавливается в 0 значение перед выполнением операции. Ошибка возвращается, если ключ содержит значение неправильного типа или содержит строку, которая не может быть представлена в виде целого числа. Эта операция ограничена **64-битными целыми числами со знаком.**

**incrby:** увеличивает число, хранящееся в key, путем приращения на указанное число. Если ключ не существует, перед выполнением операции ему присваивается значение 0. Возвращается ошибка, если ключ содержит значение неправильного типа или содержит строку, которую невозможно представить как целое число. Эта операция ограничена 64-битными целыми числами со знаком.

**decrby:** уменьшает число, хранящееся в ключе, на указанное число. Если ключ не существует, перед выполнением операции ему присваивается значение 0. Возвращается ошибка, если ключ содержит значение неправильного типа или содержит строку, которую невозможно представить как целое число. Эта операция ограничена 64-битными целыми числами со знаком.

1. Поясните назначение команд СУБД Redis: mset, mget.

**mset:** устанавливает для нескольких ключей соответствующие значения. MSET заменяет существующие значения новыми значениями, так же, как и set. MSET является атомарным, поэтому все заданные ключи устанавливаются сразу.

**mget**: возвращает значения всех заданных ключей. Для каждого ключа, который не содержит строкового значения или не существует, возвращается специальное значение nil. Из-за этого операция никогда не заканчивается неудачей.

1. Поясните назначение команд СУБД Redis: hset, hget.

*Хеш в Redis – это тип данных, используемый для хранения сопоставления между ключом и значением. Хэши Redis подходят для хранения сложных объектов данных, поскольку они могут содержать несколько пар поле(field)-значение(value).*

*key: field1: value1, field2: value2 …*

**hset:** устанавливает пары field-value в хэше (key) (раньше только одну пару). Если key не существует, создается новый ключ, содержащий хэш. Если field хэш уже существует, он перезаписывается. Начиная с Redis **4.0.0,** HSET является вариативным и позволяет использовать **несколько** field/value пар.

**hget:** возвращает значение (value), связанное с field в указанном хэше.

1. Поясните назначение команд СУБД Redis: hmset, hmget.

**hmset:** устанавливает для указанных полей соответствующие значения в хэше. Эта команда перезаписывает все указанные поля, уже существующие в хэше. Если key не существует, создается новый ключ, содержащий хэш. Согласно Redis 4.0.0, HMSET считается устаревшим (hset теперь тоже может устанавливать несколько полей сразу).

**hmget:** возвращает значения, связанные с указанными fields в хэше. Для каждого field, что не существует в хэше, возвращается значение nil. Поскольку несуществующие ключи рассматриваются как пустые хэши, запуск HMGET против несуществующего key вернет список nil значений.

1. Поясните назначение команды СУБД Redis: exists.

Возвращает число существующих ключей среди заданных.

Начиная с Redis 3.0.3, можно указать несколько ключей вместо одного. В этом случае он возвращает общее количество существующих ключей. Обратите внимание, что возврат 1 или 0 для одного ключа – это всего лишь частный случай вариативного использования, поэтому команда полностью обратно совместима.

Если один и тот же существующий ключ упоминается в аргументах несколько раз, он будет подсчитан несколько раз. Так что если somekey существует, EXISTS somekey somekey вернет 2.

10. Поясните назначение команды СУБД Redis multi.

Отмечает начало блока транзакции. Последующие команды будут поставлены в очередь для атомарного выполнения с использованием EXEC.

11. Поясните назначение метода sendCommand.

Если команда Redis еще не реализована (в nodejs redis), метод sendCommand можно использовать для отправки пользовательских команд на сервер.