



4-Я ОНЛАЙН КОНФЕРЕНЦИЯ ПО
ТЕСТИРОВАНИЮ ПО
ВАДИМА КСЕНДЗОВА

NoSQL Redis

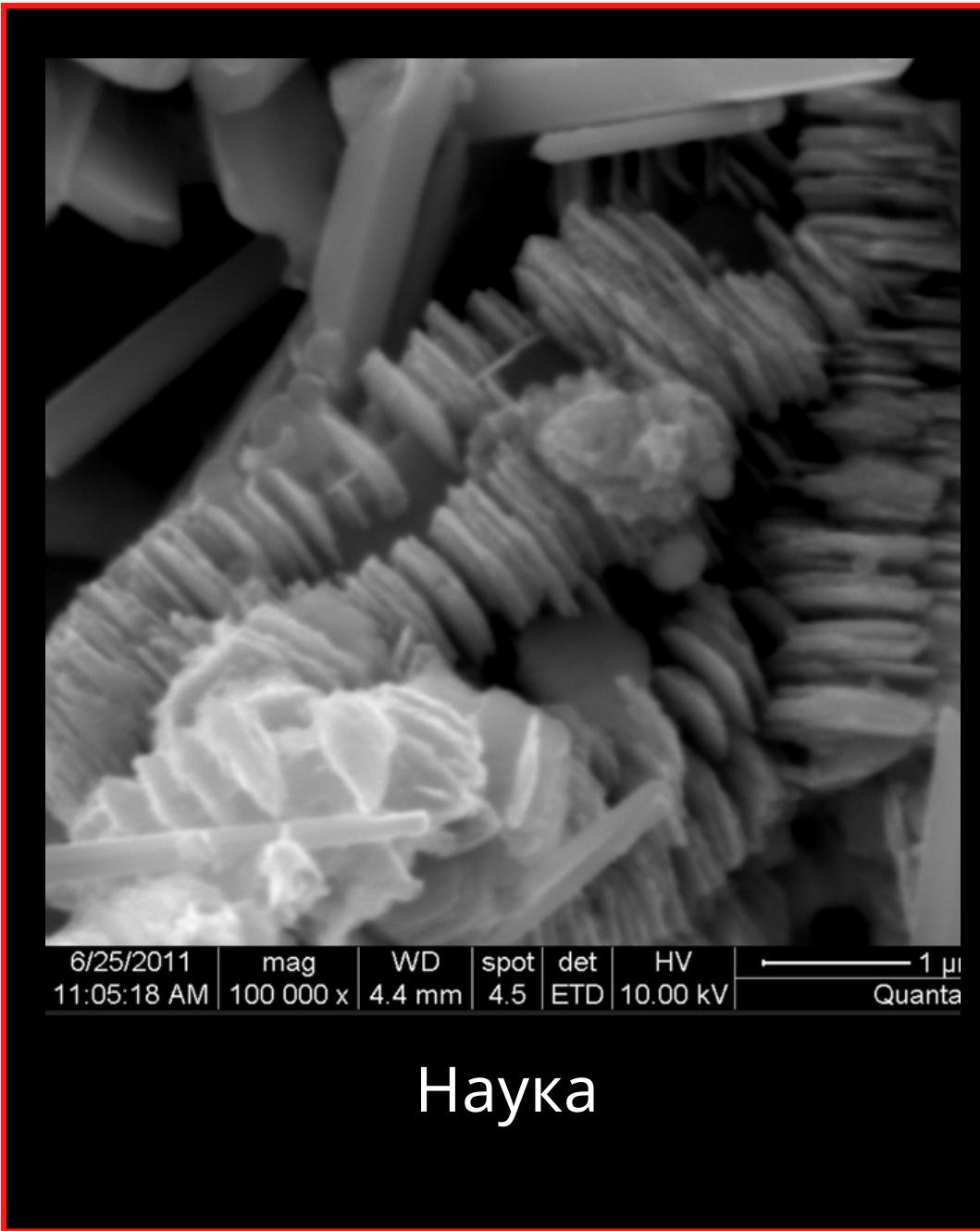
Оксана Соклакова

-  <https://www.linkedin.com/in/oksanasoklakova/>
-  @oxanamara
-  soklakovaon@bk.ru
- Россия, Белгород

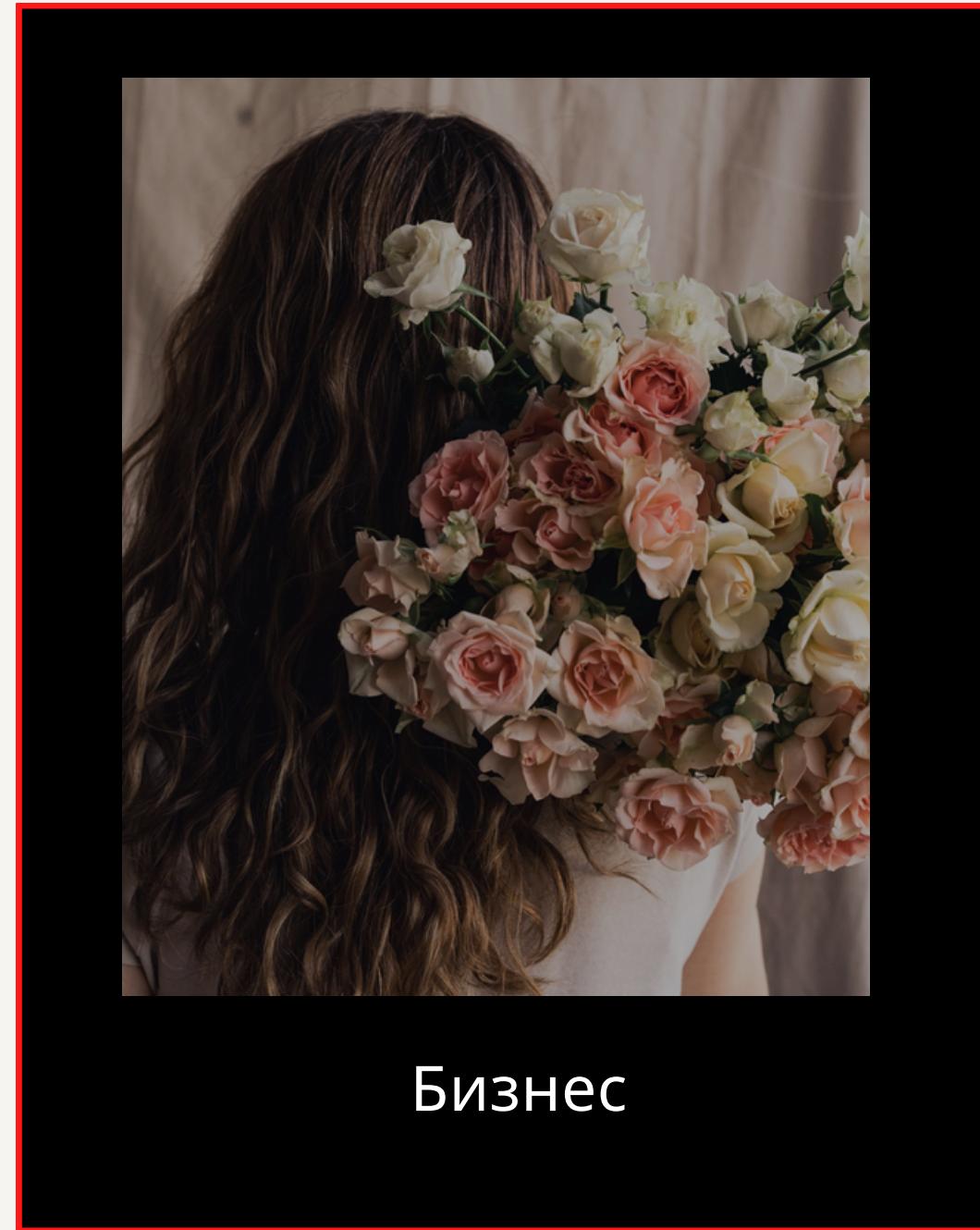


О СЕБЕ

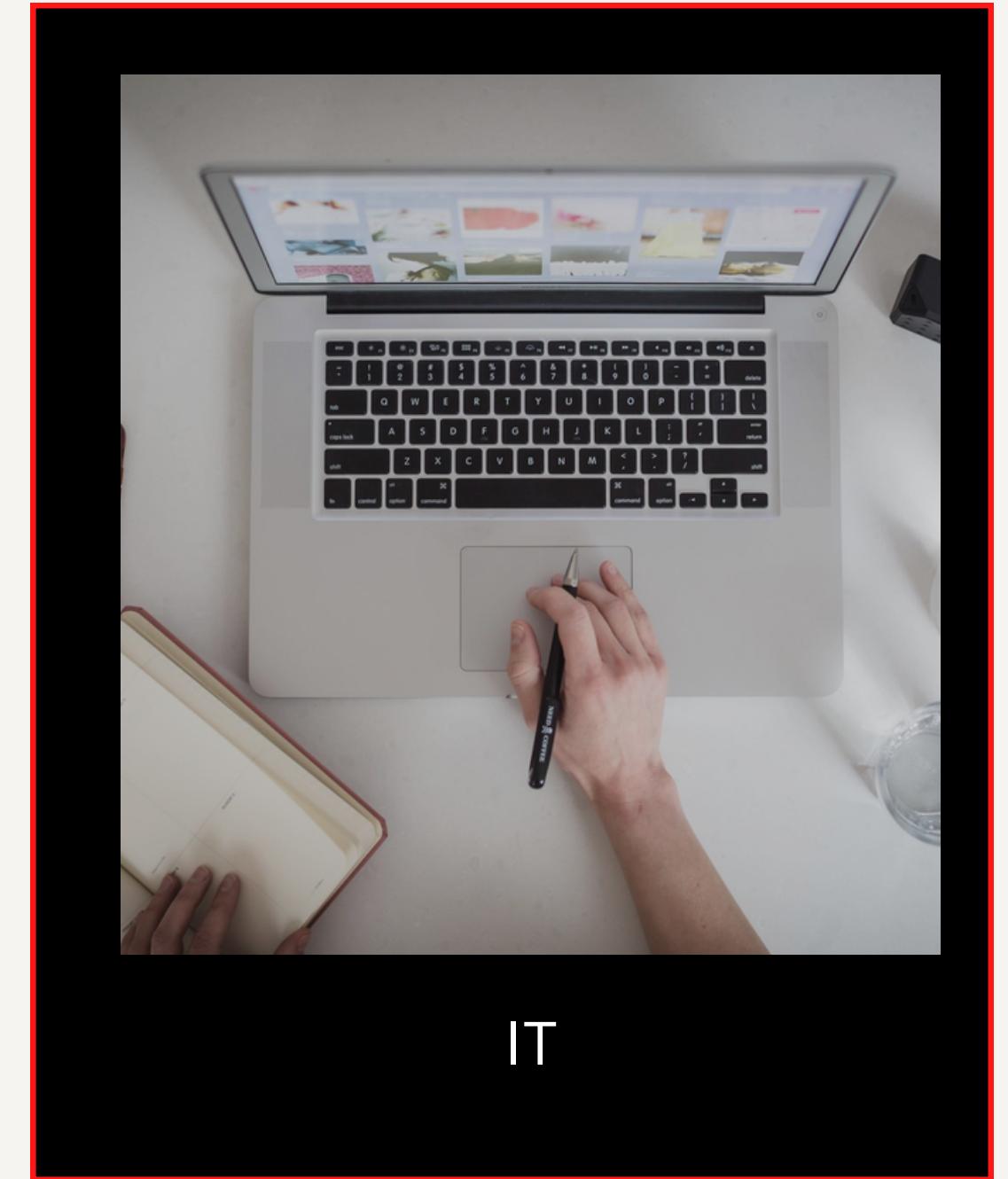
>>>>>>



Наука



Бизнес



IT

NoSQL Redis

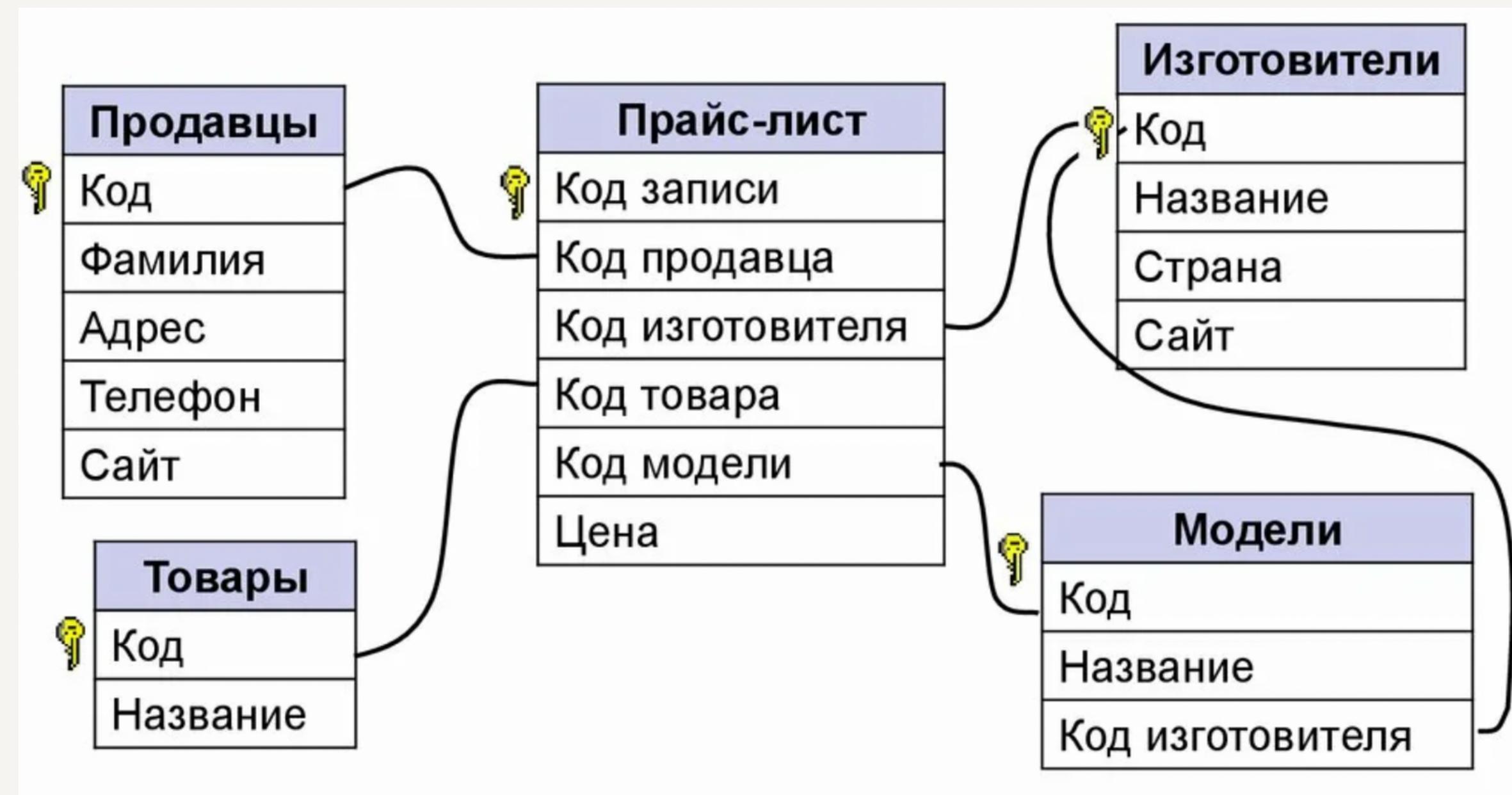
SQL - Structured Query Language

- хранение
- манипулирование
- извлечение данные из
реляционных БД



>>>>>>

Реляционные БД



Entity-Relationship model, модель «сущность – связь»



ACID

требования

- **Atomicity** – Атомарность
 - **Consistency** – Согласованность
 - **Isolation** – Изолированность
 - **Durability** – Надёжность
-

MySQL

Oracle

Postgres



Реляционные БД

- Негибки
- Способны хранить данные только в табличных формах
- Медленные в больших масштабах

150-200 записей/сек

*Postgres



Хотелось бы что-то побыстрее...

>>>>>>



Применение:

- быстрое взаимодействие с пользователем
- обработка больших массивов данных



Что такое In-Memory?

Классические концепции хранения:

- диск - основное хранилище, память - кэш
- время запроса - миллисекунды

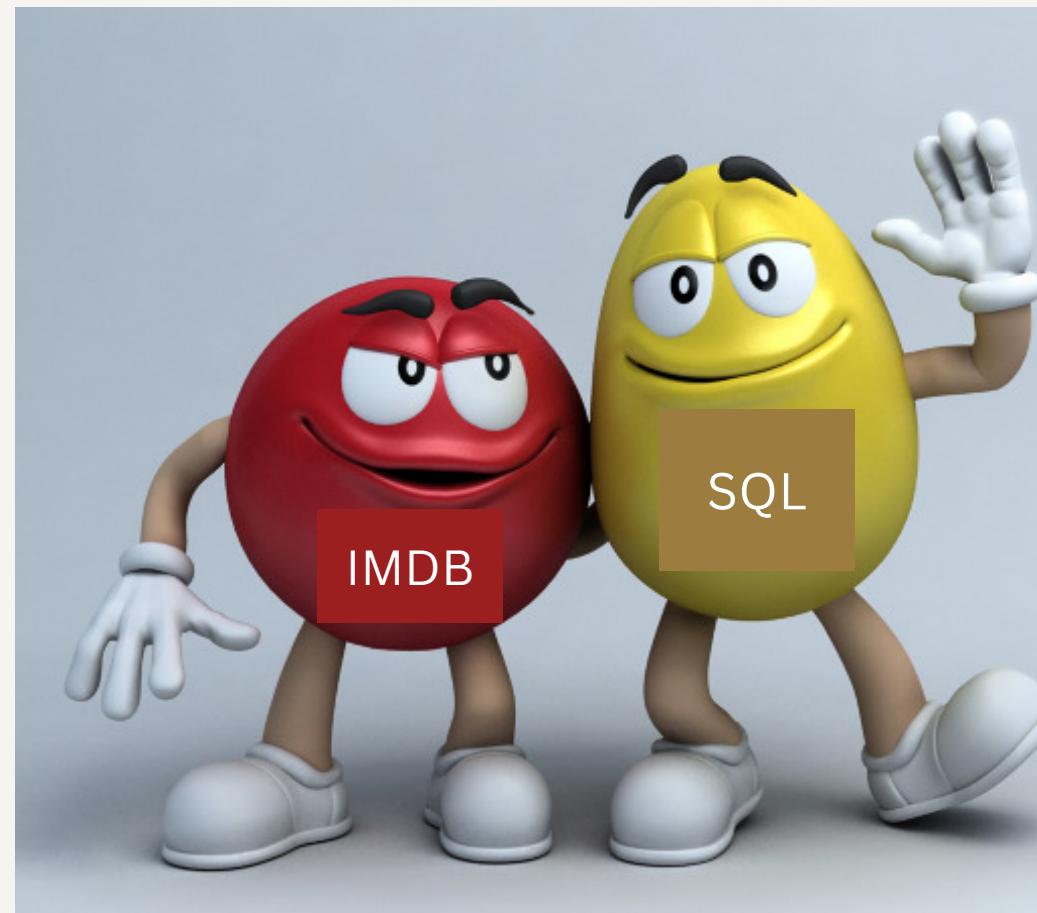
In-memory концепции хранения:

- память - основное хранилище, диск - бэкап
- время запроса - наносекунды и микросекунды

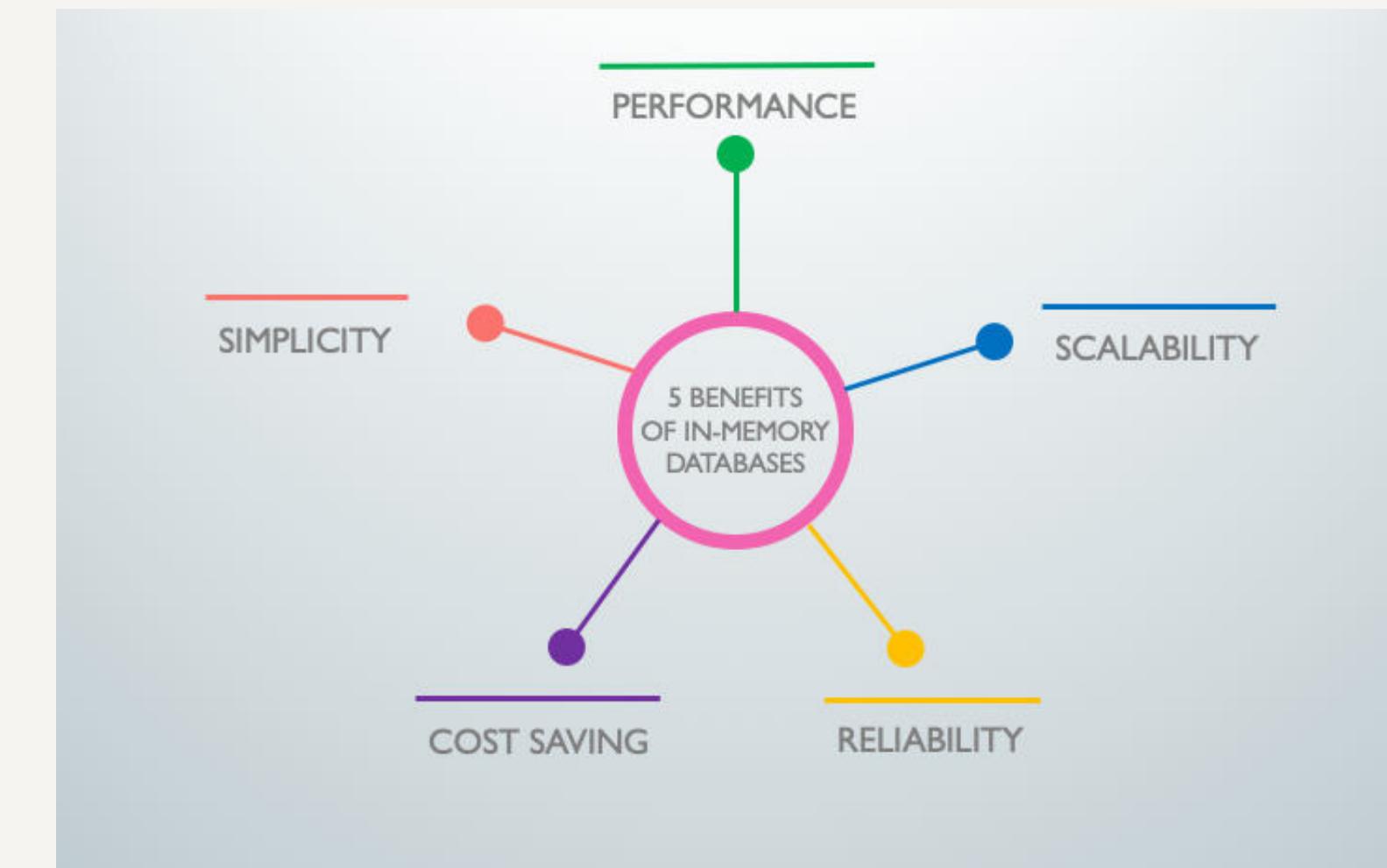
In-Memory

Преимущества баз данных In-Memory :

- быстрая выполнения операций;
- эффективное сохранение зафиксированных данных, которые используются не часто, на жестком диске;
- высокая пропускная способность систем, критичных к производительности.

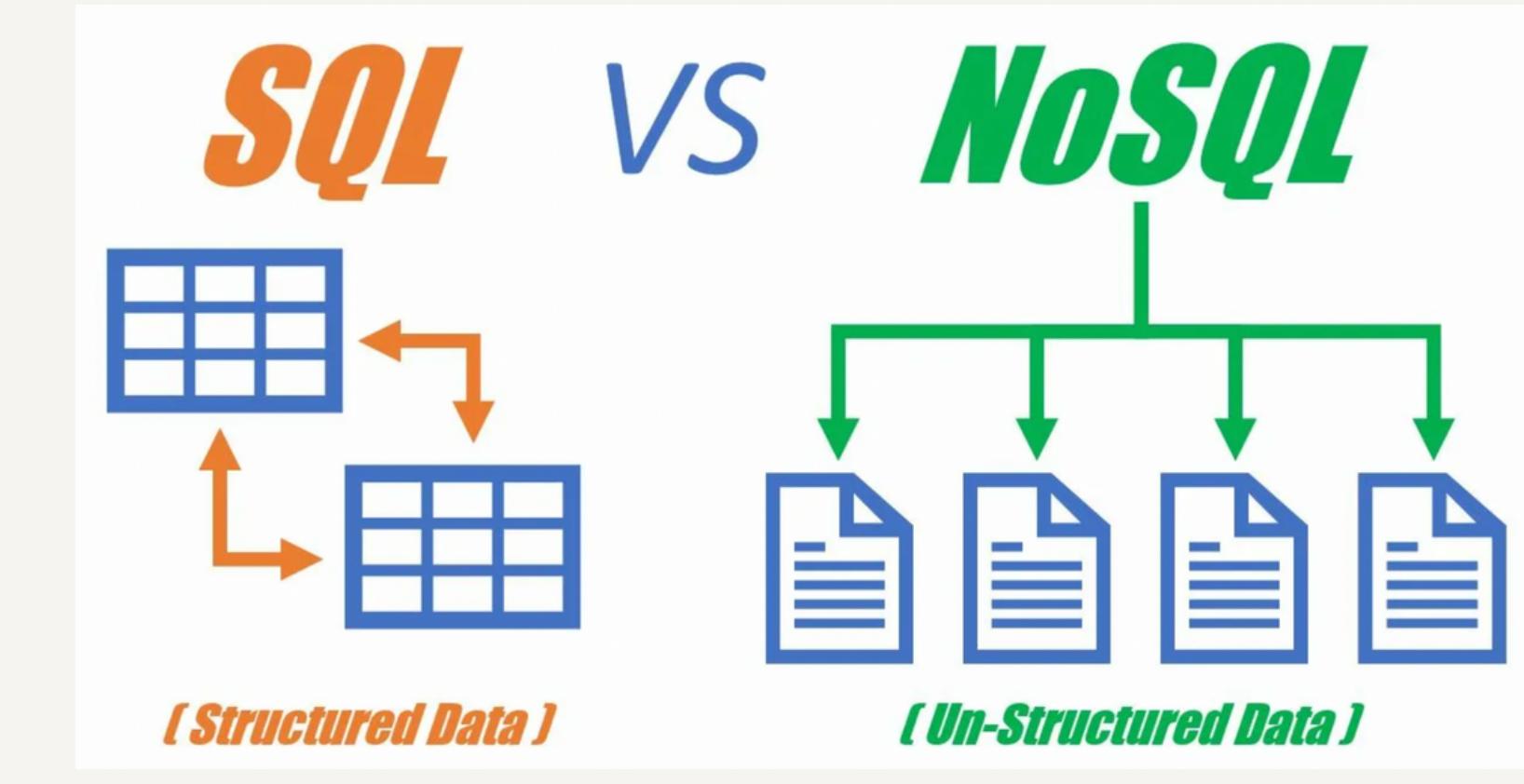
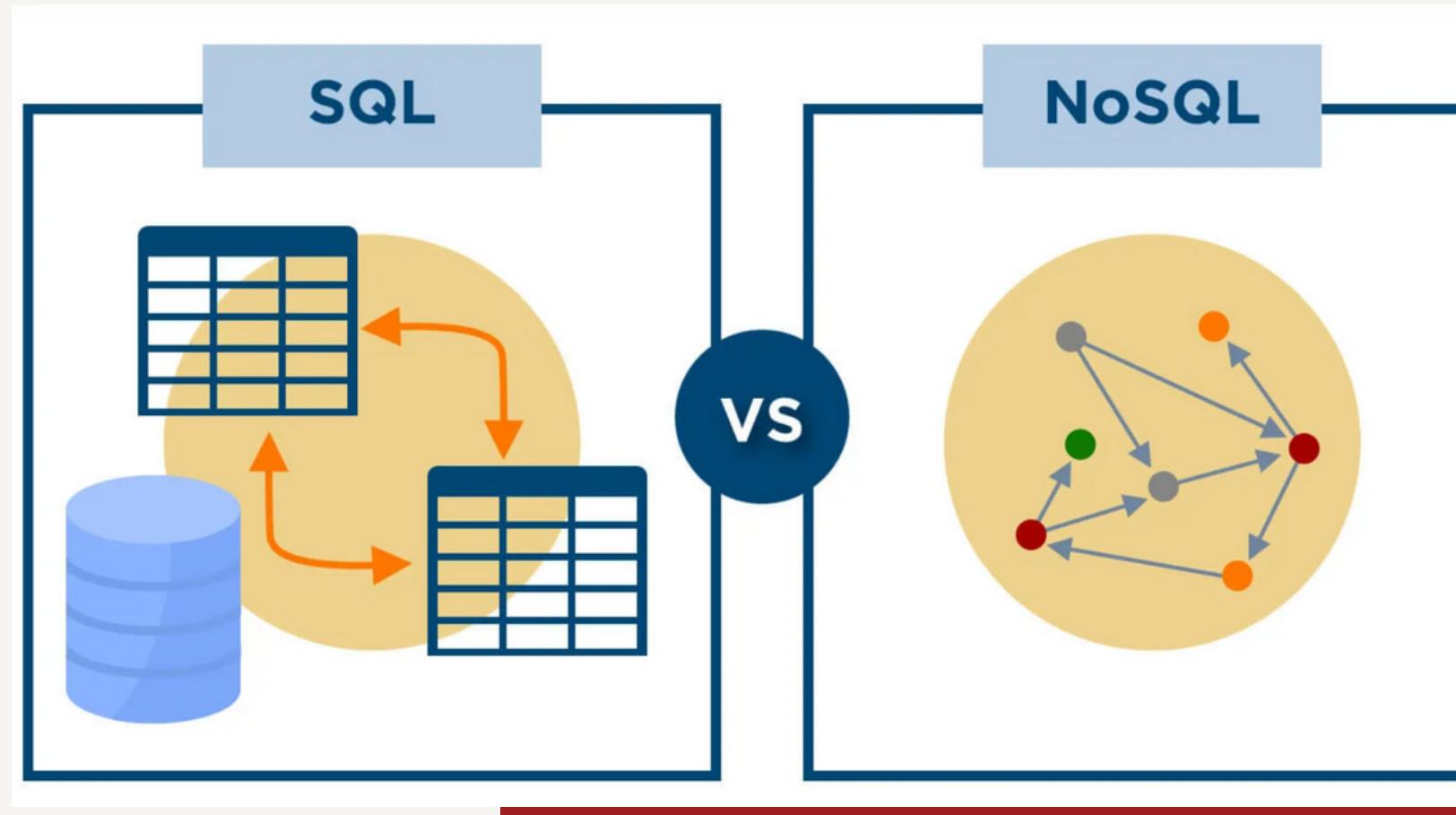


долговечность



NoSQL

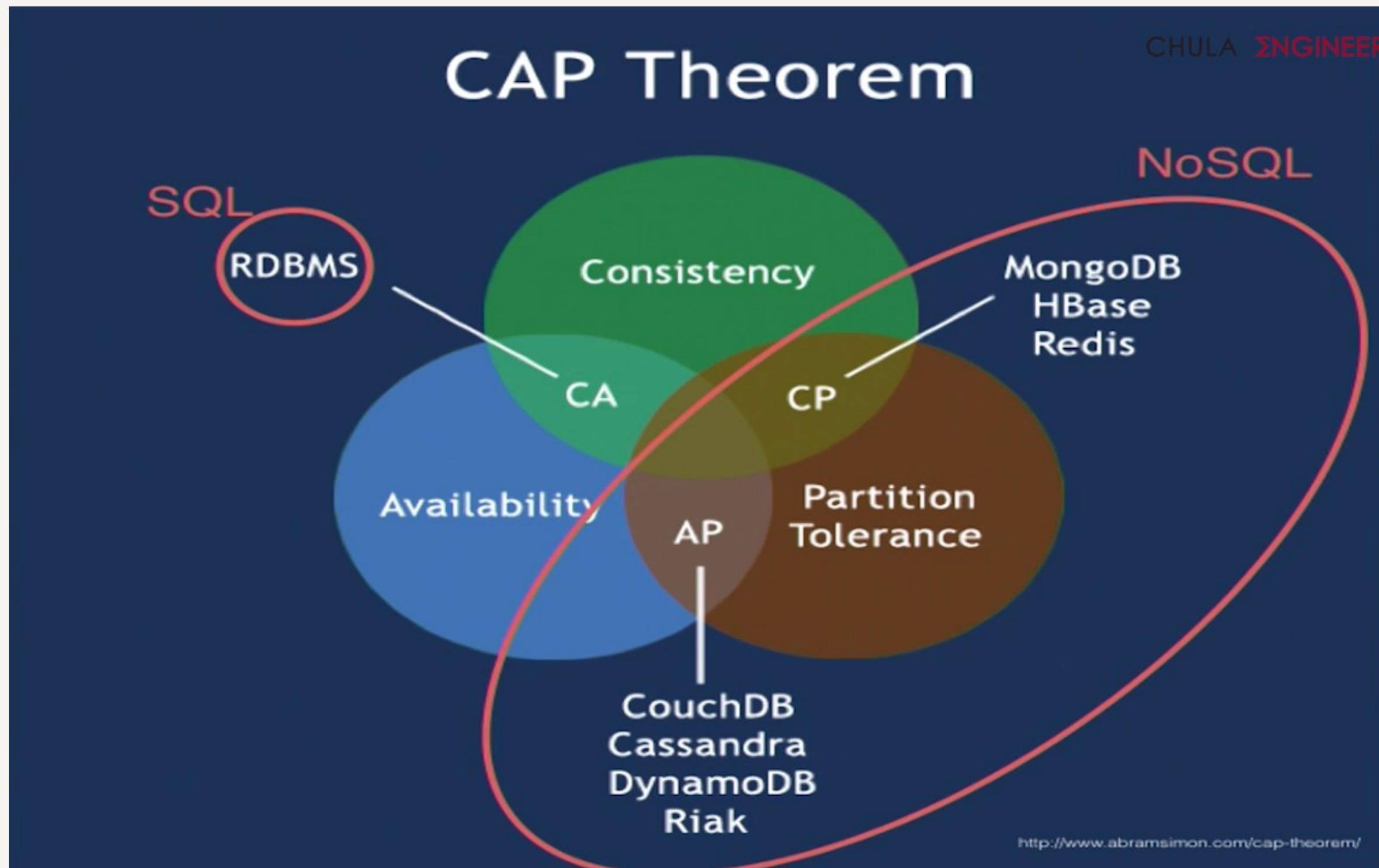
NoSQL (от англ. not only SQL) - не только SQL



Aerospike, Cassandra, Couchbase, DynamoDB, HBase, MarkLogic,
MongoDB, Oracle NoSQL, Redis, Riak



CAP-теорема - теорема Брюера



*Consistency-Availability-Partition tolerance



NoSQL



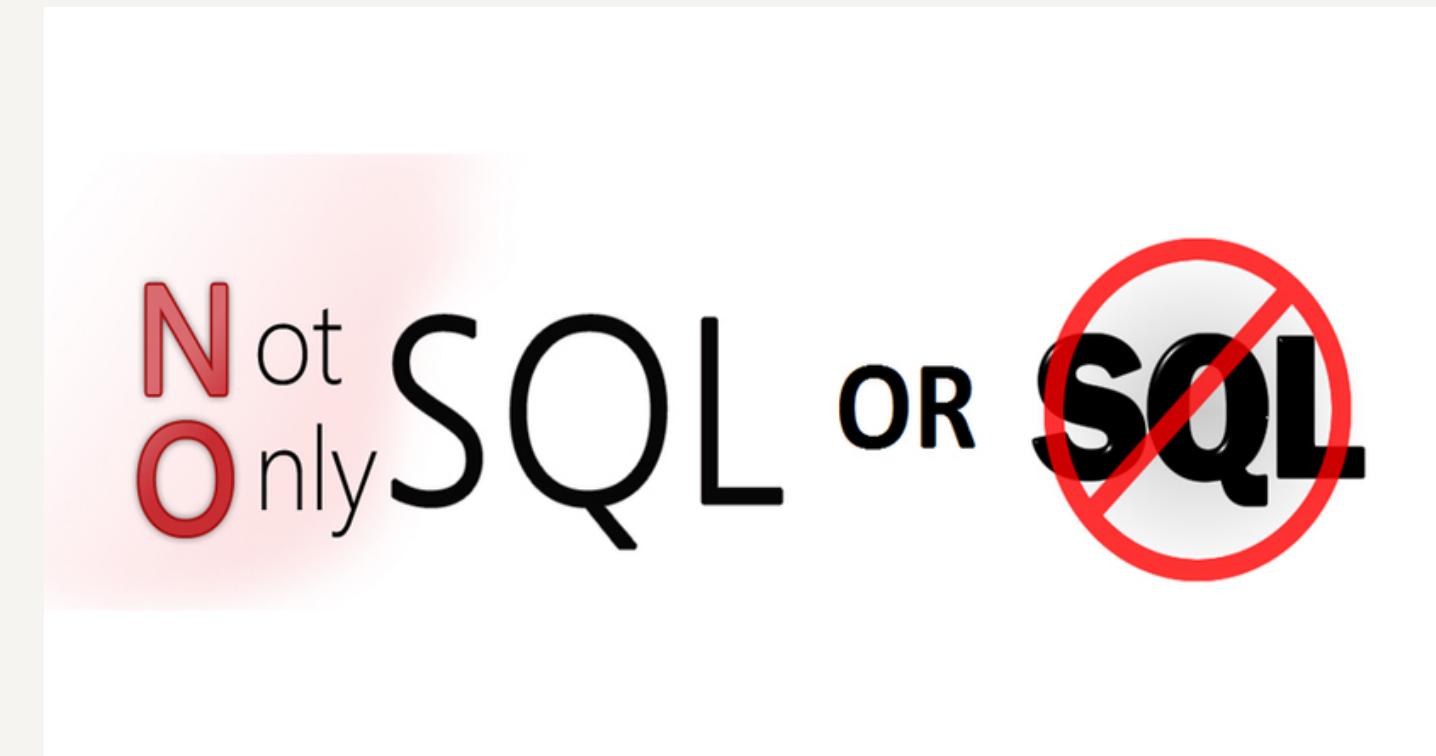
Предлагается пожертвовать
согласованностью данных ради
доступности и масштабируемости



- Высокая гибкость
- Отличная эффективность
- Хорошая масштабируемость
- Структура и тип данных
- Производительность
(10 000 запросов/сек)

Разница NoSQL/SQL

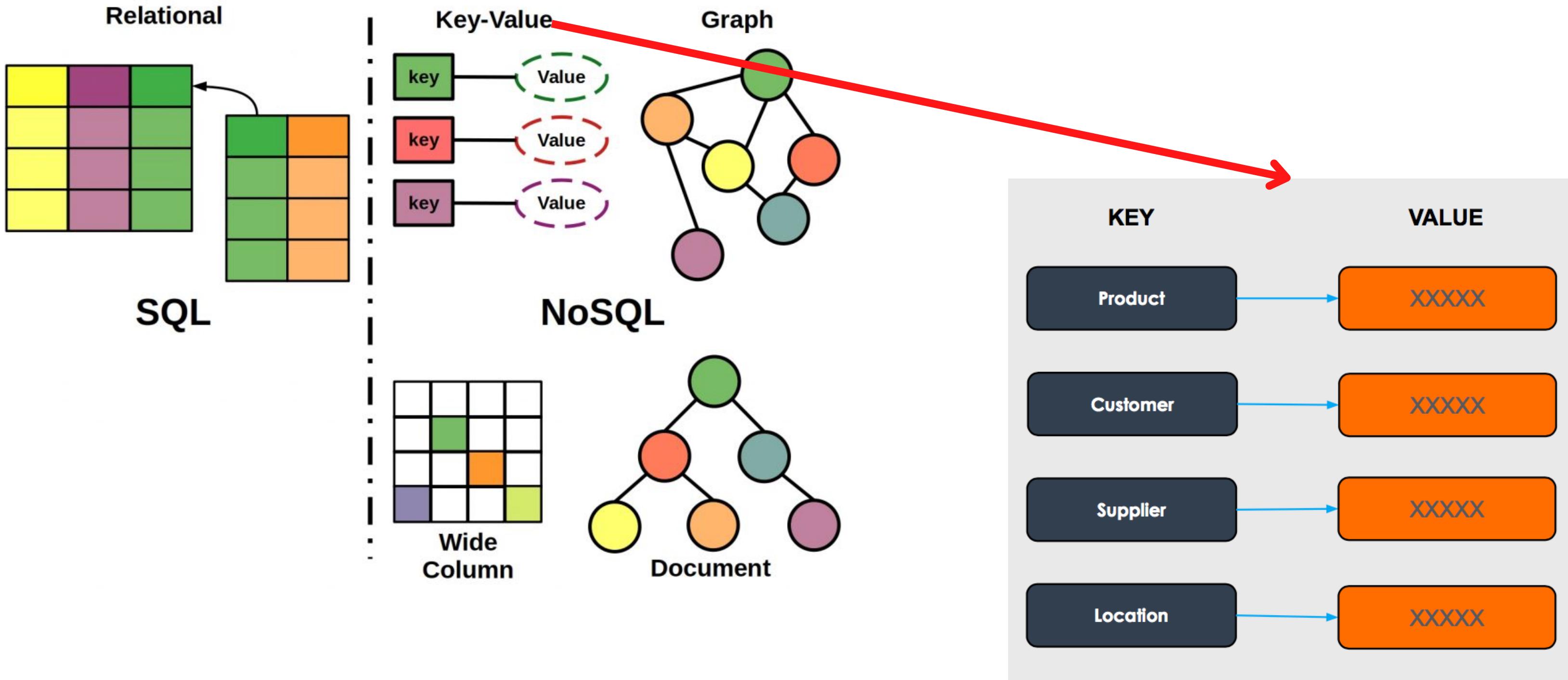
- Больше производительность, скорость и масштаб
- Гибкий подход к записям данных, нет жёстких требований к структуре
- NoSQL имеет высокую защиту от хакерских атак.
- Для неструктурированных данных, например, по типу «ключ-значение», «дерево документов» или «граф»



>>>>>>

- Для структурированных данных, которые вписываются в табличный вид
- Стабильность
- SQL более простые и удобные в последующей работе благодаря своей структуризации

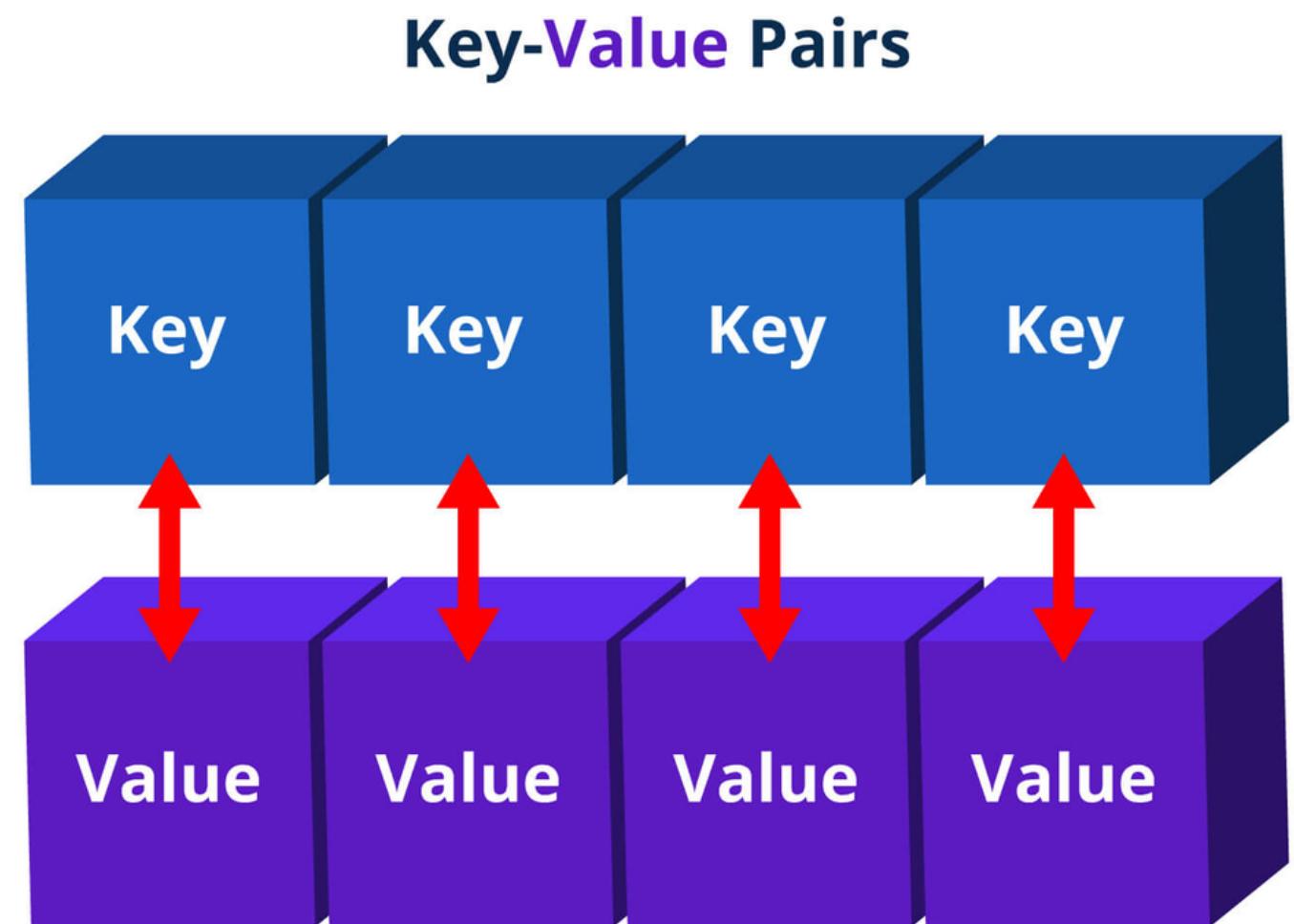
Key-Value

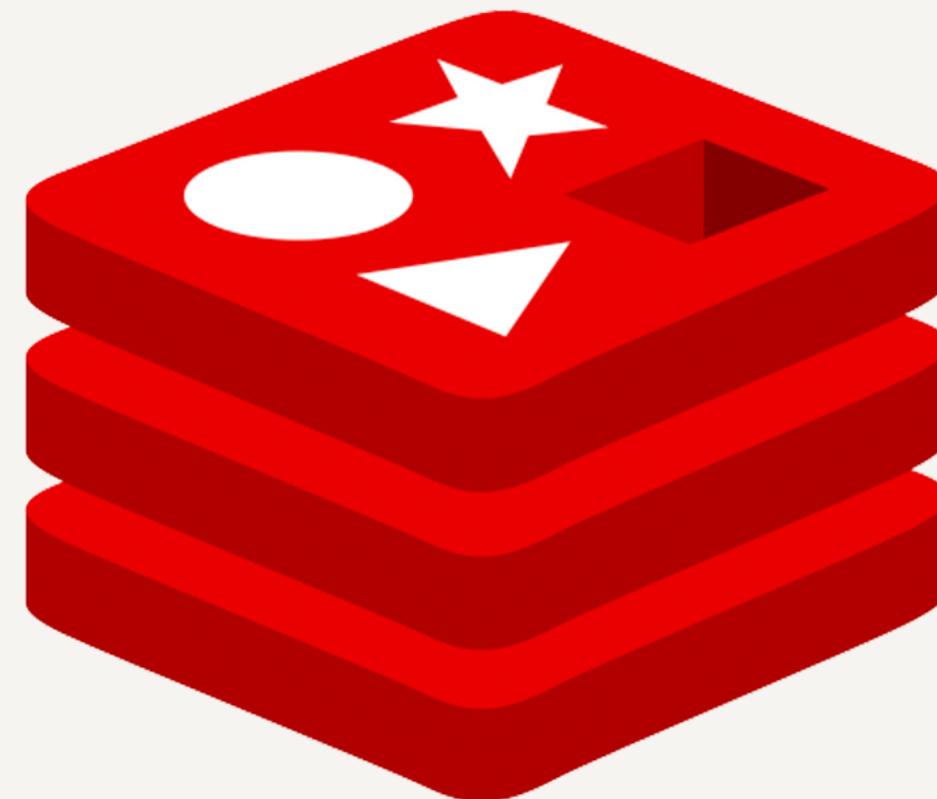


Key-Value



- Для быстрого сохранения базовых данных
- Быстрые, работоспособные
- Простые операции запроса, вставки и удаления
- Поиск по значению отсутствует
- Главные плюсы - масштабируемость, простота



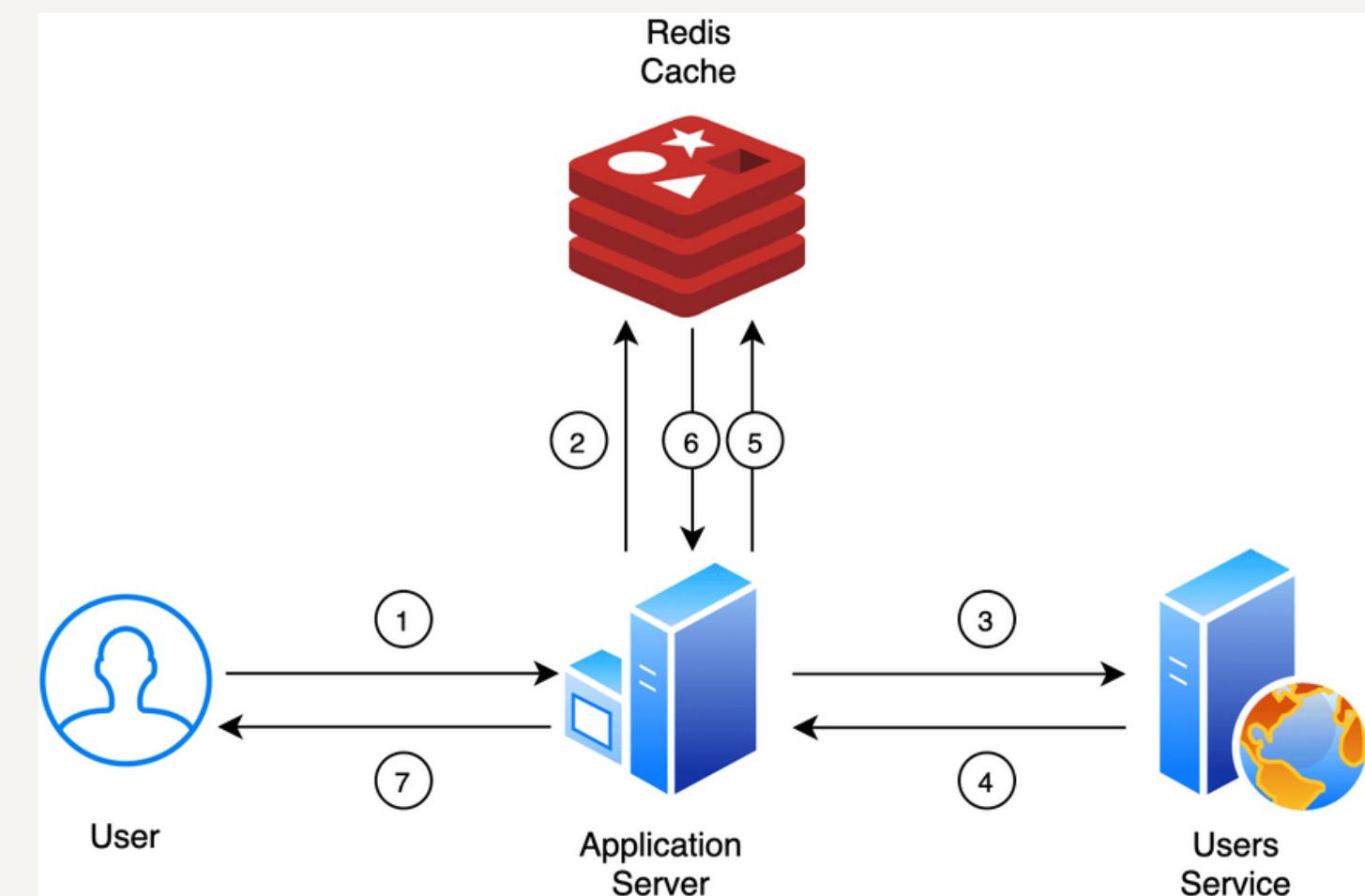


redis

NoSQL Key-value In-memory DataBase Redis

REmote Dictionary Server - сервер структур данных

- Язык разработки: C
- Библиотеки для работы с Redis есть во всех современных языках программирования
- «ключ – значение»
- 100 тыс. SET- и GET- запросов/сек
- Базируется в памяти, но с использованием долговременного хранилища
- Не поддерживает SQL
- Хорошо масштабируется



Применение

сервисы для
путешественников

геокодирование

таблицы
результатов
в игровых
приложениях

для хранения сессий

фильтрация
запрещённого контента

записи и хранения метрик продуктов и продаж

форумы

социальные сети

электронная коммерция

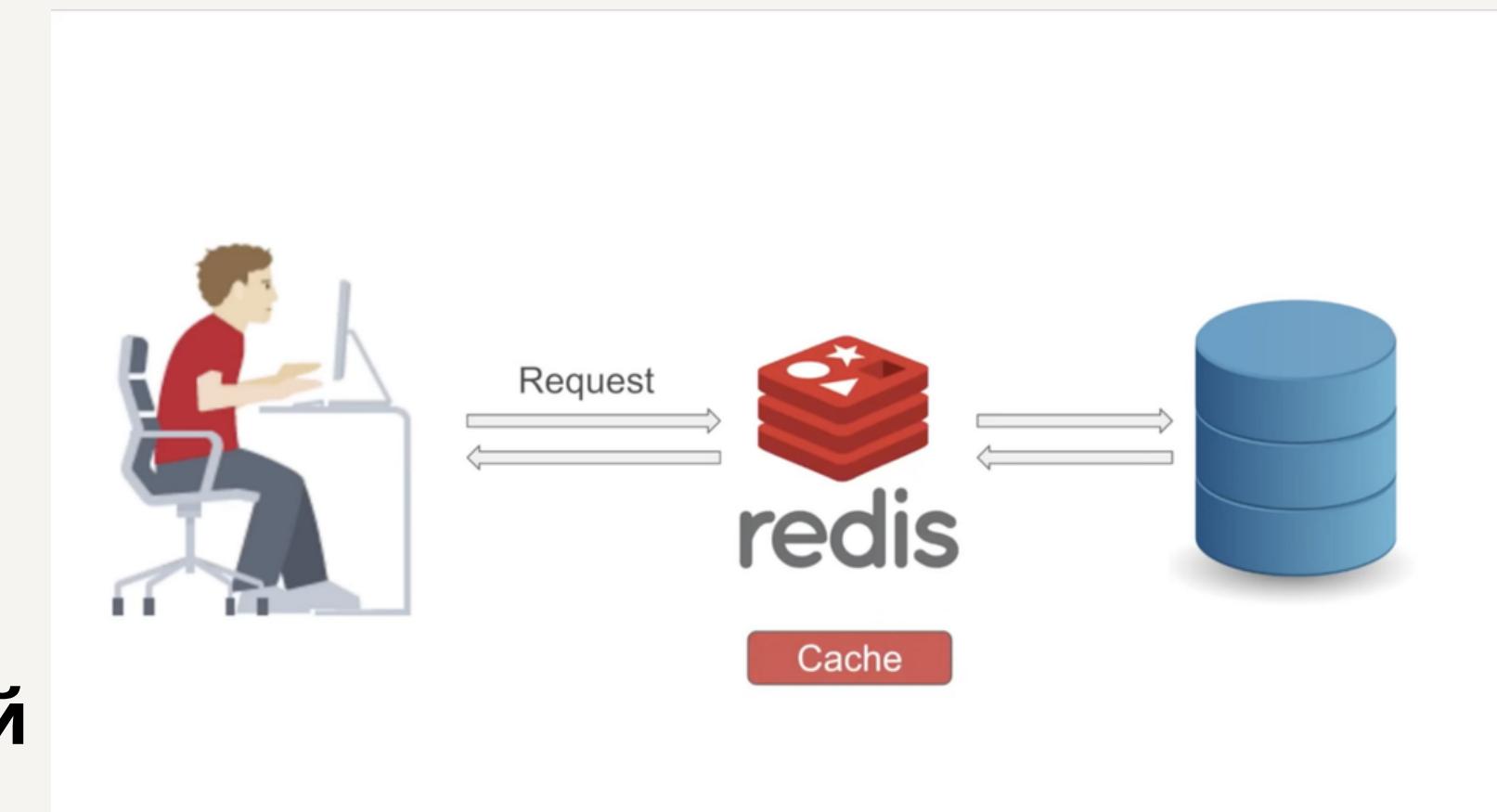
очереди

лидерборды

билинг по
объёму
оказанных услуг

анализ продаж

планировщик задач



Установка

<https://redis.io>



The screenshot shows the Redis website homepage. At the top, there's a dark header bar with the Redis logo, navigation links (GET STARTED, DOCS, COMMANDS, RESOURCES, COMMUNITY, SUPPORT), a search bar, a 'Download' button, and a 'Try Redis Cloud' button. Below the header, the word 'Redis' is prominently displayed in large black letters. To its left is a sidebar with links for Redis (Get started, Data types, Redis CLI, Redis clients, Persistence, Scaling) and Redis Stack (Get started, Stack clients, RedisInsight, JSON, Search, Probabilistic). At the bottom of the sidebar are two buttons: 'Get Started' (blue) and 'Read the docs' (white). The main content area features a dark background with white text, showing a Redis command-line interface session:

```
●▲★  
redis> PING  
"PONG"  
redis> HSET user:1 name antirez vocation artist  
(integer) 2  
redis> SET e 2.71  
"OK"  
redis> INCRBYFLOAT e 0.43  
"3.14"  
redis> RENAME e pi  
"OK"  
redis>
```

Установка

<https://redis.io>

Install Redis

How you install Redis depends on your operating system and whether you'd like to install it bundled with Redis Stack and Redis UI. See the guide below that best fits your needs:

- [Install Redis from Source](#)
- [Install Redis on Linux](#)
- [Install Redis on macOS](#)
- [Install Redis on Windows](#)
- [Install Redis with Redis Stack and RedisInsight](#)

Once you have Redis up and running, and can connect using `redis-cli`, you can continue with the steps below.



Установка

Install Redis on macOS



You can download the last Redis source files here. For additional options, see the [Redis downloads](#) section below.

Stable (7.0)

Redis 7.0 includes several new user-facing features, significant performance optimizations, and many other improvements. It also includes changes that potentially break backwards compatibility with older versions.

- [Download 7.0.5](#)
- [7.0 Release Notes](#)
- [More installation options →](#)

Source files

Use Homebrew

Homebrew
Менеджер недостающих пакетов для macOS

Search Homebrew

Русский

Установка Homebrew

```
$ /bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
```

Вставьте эту строку в терминал.

Перед выполнением скрипта объясните, что он собирается сделать. Другие варианты установки можно найти [здесь](#).

Terminal

>>>>>

Установка

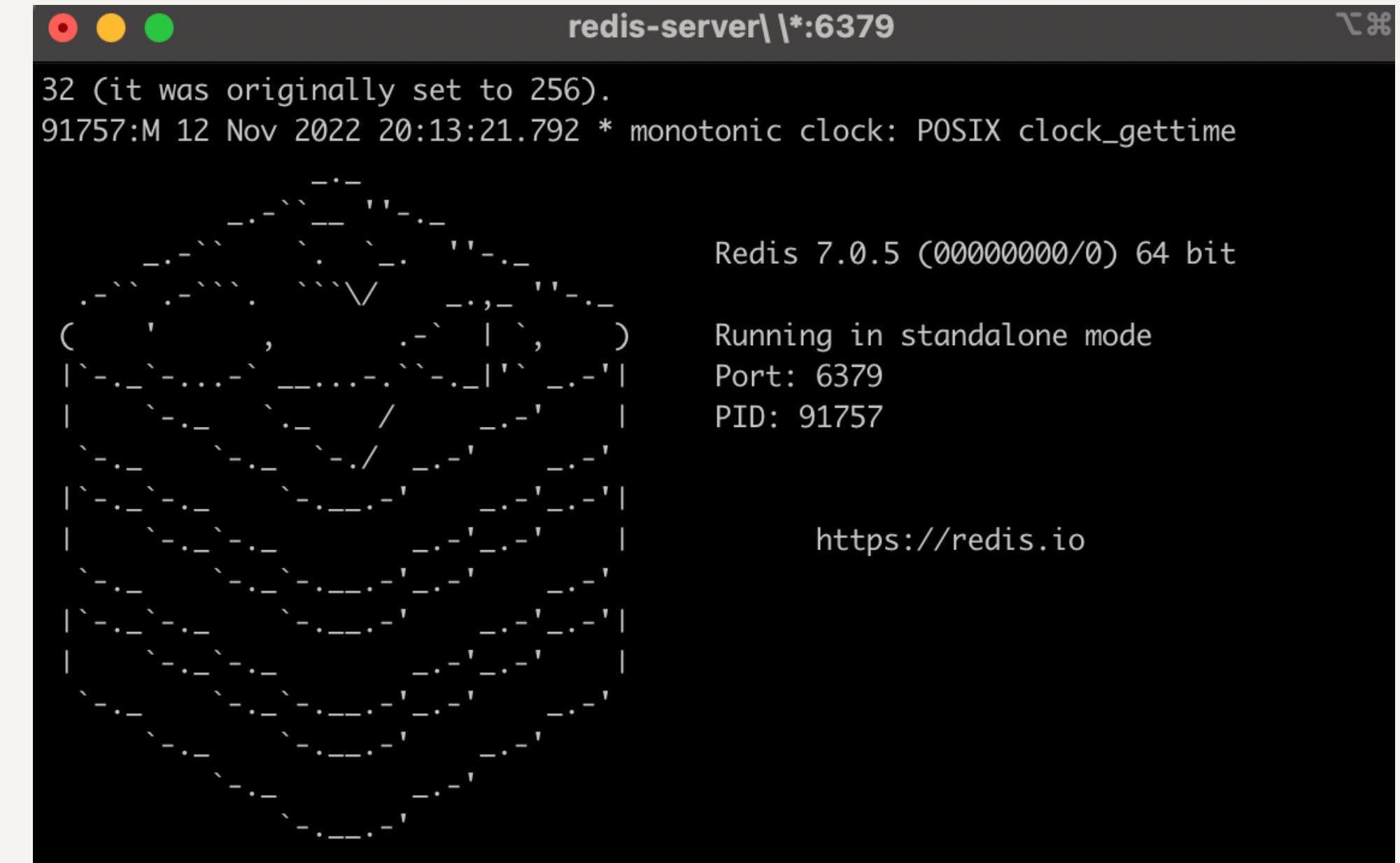
```
brew install redis
```

This will install Redis on your system.

Starting and stopping Redis in the foreground

To test your Redis installation, you can run the `redis-server` executable from the command line:

```
redis-server
```



The terminal window shows the Redis server starting up. It displays the Redis logo (a stylized tree composed of characters like ' ', ' ', and ' '), followed by the text "Redis 7.0.5 (00000000/0) 64 bit". Below this, it says "Running in standalone mode", "Port: 6379", and "PID: 91757". At the bottom right, there is a link "https://redis.io". The terminal window has a dark background with light-colored text and icons.

Готово к соединению:



Connect

```
$redis = new Redis();
$redis->connect('127.0.0.1',
    6379);
```

// Получить доступ к БД Redis:

redis-cli

```
[oksanasoklakova@MacBook-Air-Oksana ~ % $redis = new Redis();
$redis->connect('127.0.0.1', 6379);
oksanasoklakova@MacBook-Air-Oksana ~ %
```

```
==> Successfully started `redis` (label: homebrew.mxcl.redis)
oksanasoklakova@MacBook-Air-Oksana ~ % brew services info redis
redis (homebrew.mxcl.redis)
Running: ✓
Loaded: ✓
Schedulable: ✘
User: oksanasoklakova
PID: 93297
oksanasoklakova@MacBook-Air-Oksana ~ % redis-cli
127.0.0.1:6379> lpush demos redis-macOS-demo
(integer) 1
127.0.0.1:6379> rpop demos
"redis-macOS-demo"
127.0.0.1:6379> EXISTS foo
(integer) 0
127.0.0.1:6379>
```



Структуры и типы данных Redis

hello world	String
011011010110111101101101	Bitmap
{23334}{6634728}{916}	Bitfield
{a: "hello", b: "world"}	Hash
[A>B>C>C]	List
{A<B<C}	Set
{A:1, B:2, C:3}	Sorted set
{A: (50.1, 0.5)}	Geospatial
01101101 01101111 01101101	Hyperlog
{id1=time1.seq({ a: "foo", a: "bar"}) }	Stream

Типы данных:

- Строки (Strings)
- Множества (Sets)
- Сортированные множества (Sorted Sets)
- Списки (Lists)
- Хеш (Hash)



STRING строки

Установка строкового значения
по ключу и чтение:

```
127.0.0.1:6379> EXISTS foo
(integer) 0
127.0.0.1:6379> SET foo "Hello Vadim 31 gr!"
OK
127.0.0.1:6379> GET foo
"Hello Vadim 31 gr!"
127.0.0.1:6379>
```

Общие команды для строк:

SET: задаёт значение ключа.

GET: извлекает значение ключа.

DEL: удаляет ключ и его значение.

INCR: выполняет автоматический
инкремент ключа.

INCRBY: выполняет инкремент ключа на
указанную величину.

EXPIRE: срок хранения ключа (в
секундах).

Строки используются для хранения отсортированных по ключу данных.



STRING строки

```

127.0.0.1:6379> EXISTS mar
(integer) 0
127.0.0.1:6379> SET mar "QA"
OK
127.0.0.1:6379> GET mar
"QA"
127.0.0.1:6379> GETSET mar "QA engineer"
"QA"
127.0.0.1:6379> GET mar
"QA engineer"
127.0.0.1:6379>

```

Установка нового значения с
возвращением старого:

Установка срока жизни значения:

Это реализация кэша

```

127.0.0.1:6379> SET foo "Hello!" EX 5
OK
127.0.0.1:6379> GET foo
"Hello!"
127.0.0.1:6379> GET foo
(nil)

```

>>>>>>>

STRING строки

Инкремент и декремент (счетчики)

```
// Установка значения
$redis->set('rate:GBP', 92);

// Установим строку
$redis->set('description:GBP', 'Pound sterling');

// Установить значение, если таковое не существует
// Если существует - ничего не делать
// SET if Not eXists
$redis->setnx('rate:GBP', 100);

// Установить сразу несколько значений
// Так же сработает msetnx
$redis->mset([
    'rate:EUR' => 80,
    'rate:JPY' => 60
]);
```

>>>>>>

Увеличивает или уменьшает число, сохраненное в ключе, на единицу.

Это строковая операция, поскольку Redis не имеет выделенного целочисленного типа

```
127.0.0.1:6379> SET mar 1
OK
127.0.0.1:6379> INCR mar
(integer) 2
127.0.0.1:6379> GET mar
"2"
127.0.0.1:6379> DECR mar
(integer) 1
127.0.0.1:6379> GET mar
"1"
127.0.0.1:6379>
```

Комплексные типы данных

HASHES хэш-таблицы

ассоциативный массив

```
"1"
127.0.0.1:6379> HSET obj mar 45
(integer) 1
127.0.0.1:6379> HGET obj mar
"45"
127.0.0.1:6379> HSET obj bar 77
(integer) 1
127.0.0.1:6379> HGETALL obj
1) "mar"
2) "45"
3) "bar"
4) "77"
127.0.0.1:6379>
```

Общие команды:

HMSET: устанавливает несколько значений.

HSET: устанавливает поле со строковым значением.

HGET: извлекает значение заданного поля.

HMGET: извлекает значения нескольких полей.

HGETALL: извлекает все значения.



HSETS множества

```
127.0.0.1:6379> SADD myset "4 conf"
(integer) 1
127.0.0.1:6379> SADD myset "testing soft"
(integer) 1
127.0.0.1:6379> SMEMBERS myset
1) "testing soft"
2) "4 conf"
127.0.0.1:6379>
127.0.0.1:6379> SCARD myset
(integer) 2
```

Множество – это неупорядоченный набор значений.

Общие команды:

SADD: добавление одного или нескольких значений.

SMEMBERS: запрос всех значений множества.

SINTER: запрос общих значений нескольких множеств.

SISMEMBER: поиск значения во множестве.

SRANDMEMBER: запрос случайного значения множества.



LISTS списки

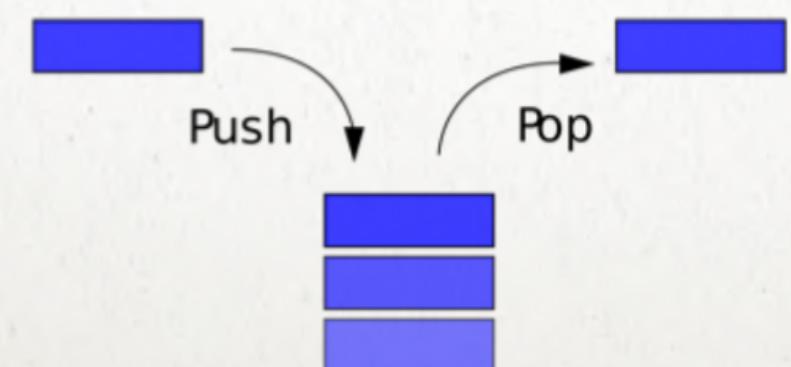
```
127.0.0.1:6379> LPUSH mylist "first"
(integer) 1
```

```
127.0.0.1:6379> LPUSH mylist "second"
(integer) 2
```

```
127.0.0.1:6379> LPOP mylist
"second"
```

```
127.0.0.1:6379>
  LINsert mylist BEFORE first "zero"
(integer) 2
```

```
127.0.0.1:6379> LRANGE mylist 0 -1
1) "zero"
2) "first"
```



Список – это последовательность значений, упорядоченных по порядку их создания

Общие команды:

LPUSH: добавление значения в начало списка.

RPUSH: добавление значения в конец списка.

LPOP: запрос и удаление первого элемента списка.

RPOP: запрос и удаление последнего элемента списка.

LREM: удаление всех элементов.

LRANGE: извлечение диапазона.

LTRIM: удаление элементов, не входящих в заданный диапазон.



ZSETS упорядоченные множества

```
127.0.0.1:6379> ZADD myzset 2 "one"  
(integer) 1
```

```
127.0.0.1:6379> ZADD myzset 1 "two"  
(integer) 1
```

```
127.0.0.1:6379>  
ZRANGE myzset 0 -1 WITHSCORES  
1) "two"  
2) "1"  
3) "one"  
4) "2"
```

Общие команды:

- **ZADD**: добавляет значение в множество с сортировкой.
- **ZREVRANGE**: выводит значения множества с сортировкой по индексу.
- **ZREM**: удаляет значения.



Транзакции

```
127.0.0.1:6379> MULTI
OK
127.0.0.1:6379(TX)> INCR mar
QUEUED
127.0.0.1:6379(TX)> INCR bar
QUEUED
127.0.0.1:6379(TX)> EXEC
1) (integer) 2
2) (integer) 1
127.0.0.1:6379>
```

Redis поддерживает транзакции согласно двум принципам:

1. Команды должны выполняться в заданном порядке. Во время процесса их нельзя прерывать другими запросами.
2. Операции нужно обрабатывать в полном объеме.

Транзакции начинаются с команды **MULTI** и запускаются командой **EXEC**.



Подписки

Одни клиенты сервиса Redis подписываются на определенные каналы сообщений

```
127.0.0.1:6379> SUBSCRIBE channel1
```

```
Reading messages... (press Ctrl-C to quit)
```

- 1) "subscribe"
- 2) "channel1"
- 3) (integer) 1

>>>>>>>

Другие клиенты могут посыпать сообщения в каналы. Подписаные на данные каналы клиенты получат сообщения

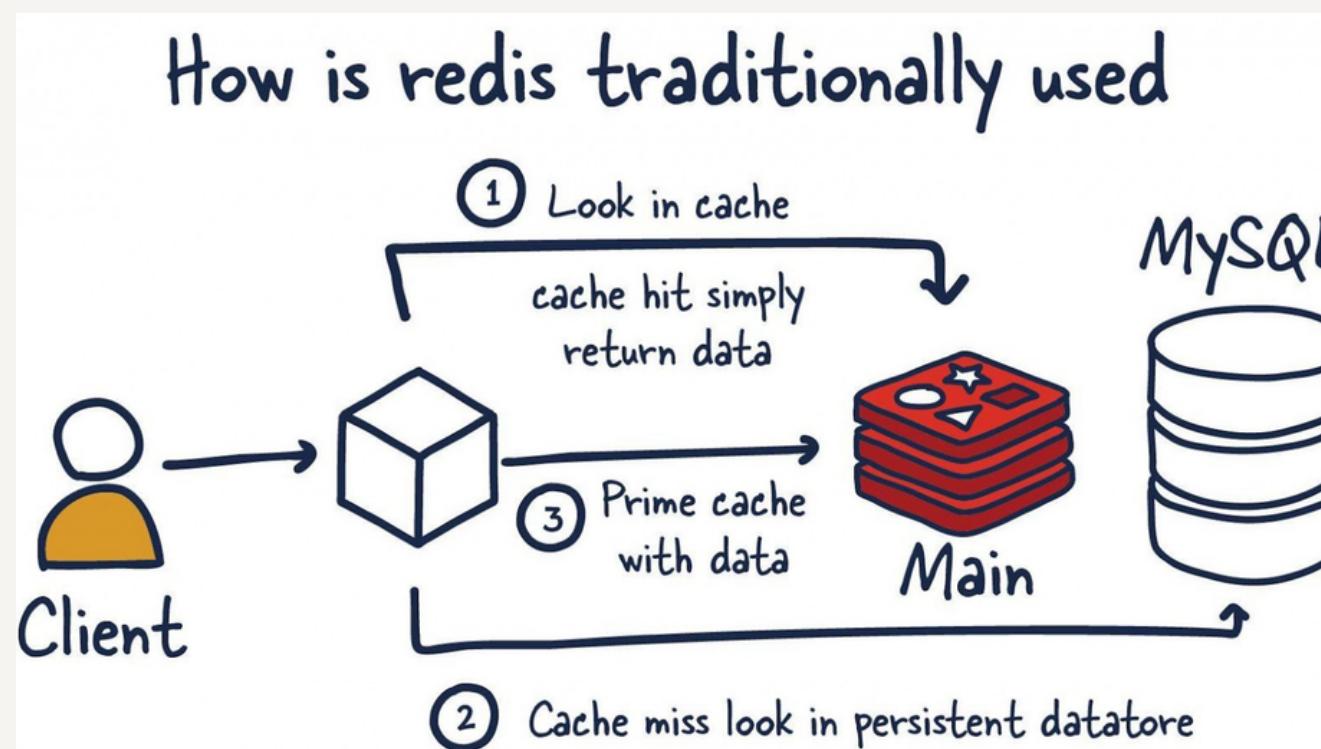
```
127.0.0.1:6379>
PUBLISH channel1 "Hello!"
```

```
(integer) 1
```

```
...
3) "Hello!"
```

Применение и статистика

- Хранение сессий пользователей.
- Кэширование данных.
- Онлайн чаты и системы обмена сообщениями.
- Различные очереди задач
- Мгновенно сортируемые «таблицы лидеров»
- Отображение часто меняющихся и часто используемых данные (котировки)



Redis по сравнению с аналогами легче, быстрее, дешевле в обслуживании

По состоянию на 2021 – 2022 гг. Redis входит в десятку наиболее востребованных баз данных по версии портала DB-Engines.

Можно назвать несколько ведущих компаний, которые используют Redis: Pinterest, Uber, Slack, Airbnb, Twitter, StackOverflow.

Список использованной литературы

1. <https://redis.io>
2. <https://habr.com/ru/company/yoomoney/blog/332462/?ysclid=l9y2aqqbb6939882864>
3. <https://www.youtube.com/watch?v=AimUYjKs3pQ>
4. https://www.youtube.com/watch?v=XCss_NVAa1g
5. <https://habr.com/ru/company/yoomoney/blog/332462/?ysclid=l9y2aqqbb6939882864>
6. <https://www.oslogic.ru/knowledge/916/nscd-sluzhba-kotoraya-keshiruet-zaprosy-sluzhby-imyon/?ysclid=l9y2b4zflj836558029>
7. <https://www.oslogic.ru/knowledge/916/nscd-sluzhba-kotoraya-keshiruet-zaprosy-sluzhby-imyon/?ysclid=l9y2dck3jj51188930>
8. <https://github.com/mallika2011/In-Memory-Key-Value-Storage?ysclid=l9y2f929dd241338938>
9. <https://www.oslogic.ru/knowledge/916/nscd-sluzhba-kotoraya-keshiruet-zaprosy-sluzhby-imyon/?ysclid=l9y2dck3jj51188930>
10. <https://github.com/mallika2011/In-Memory-Key-Value-Storage?ysclid=l9y2f929dd241338938>
11. <https://habr.com/ru/company/headzio/blog/505792/?ysclid=l9y2fs1kga158380911>
12. <https://itnan.ru/post.php?c=1&p=562192&ysclid=l9y2hdlx3b278786186#11>
13. <https://otus.ru/nest/post/715/?ysclid=l9y2ppkzsc30167853>
14. <https://otus.ru/nest/post/715/?ysclid=l9y2ppkzsc30167853>
15. <https://drive.google.com/file/d/1xANb4AlVNjNJkLGwvVhx5m9q1VIG6DRv/view?pli=1>
16. <https://www.oracle.com/cis/database/what-is-a-relational-database/>
17. <https://ru.wikipedia.org/wiki/Redis>
18. <https://studfile.net/preview/6878033/page:12/>
19. <https://www.xelent.ru/blog/otlichiya-relyatsionnykh-i-nerelyatsionnykh-baz-dannykh/?ysclid=laOuau61wt682213628>
20. https://translated.turbopages.org/proxy_u/en-ru.ru.123ef5fd-63638a3c-689c9028-74722d776562/https/en.wikipedia.org/wiki/In-memory_computing
21. <https://habr.com/ru/company/headzio/blog/505792/?ysclid=la0vckf44537350600>
22. <https://servermall.ru/blog/kak-vybrat-server-dlya-baz-dannykh/>
23. <https://ppt-online.org/488613>
24. <https://habr.com/ru/company/wunderfund/blog/685894/?ysclid=lae7lze8k0740520685>
25. <https://cloud.yandex.ru/blog/posts/2022/07/redis-overview>
26. <https://www.8host.com/blog/ustanovka-i-ispolzovanie-redis/>



Спасибо за внимание!

