# Что дальше?
# Перспективы развития баз данных

Quo vadis, Domine?

В настоящее время быстро увеличиваются объёмы хранимых данных. Появились новые системы, плохо обслуживаемые в старых парадигмах. Пример: социальные сети.

Аналитика требует всё более сложных вычислений.

Совершенно необычная задача: анализ ДНК.

Что делать? Нужны ли новые модели данных? Достаточно ли старых СУБД?

Сначала бегло рассмотрим перспективы повышения мощности серверов (закон Мура). Придём к выводу о необходимости развития параллельных систем. Кратко опишем ограничения их эффективности (закон Амдала и теорема Брюера о ключевых свойств распределённых систем).

После этого опять же бегло рассмотрим некоторые модели данных, используемые в рамках направления NoSQL.

# Закон Мура

Цитата из Википедии:

Формулировка 1965 года: “Зако́н Му́ра (англ. Moore's law) — эмпирическое наблюдение, изначально сделанное Гордоном Муром, согласно которому (в современной формулировке) количество транзисторов, размещаемых на кристалле интегральной схемы, удваивается каждые 24 месяца”.

Однако, уже сейчас размер транзистора это всего лишь немногие десятки атомов кремния (их размер примерно 0,2 нанометра).

“В 2003 году Мур опубликовал работу «No Exponential is Forever: But „Forever“ Can Be Delayed!», в которой признал, что экспоненциальный рост физических величин в течение длительного времени невозможен, и постоянно достигаются те или иные пределы. Лишь эволюция транзисторов и технологий их изготовления позволяла продлить действие закона ещё на несколько поколений”.

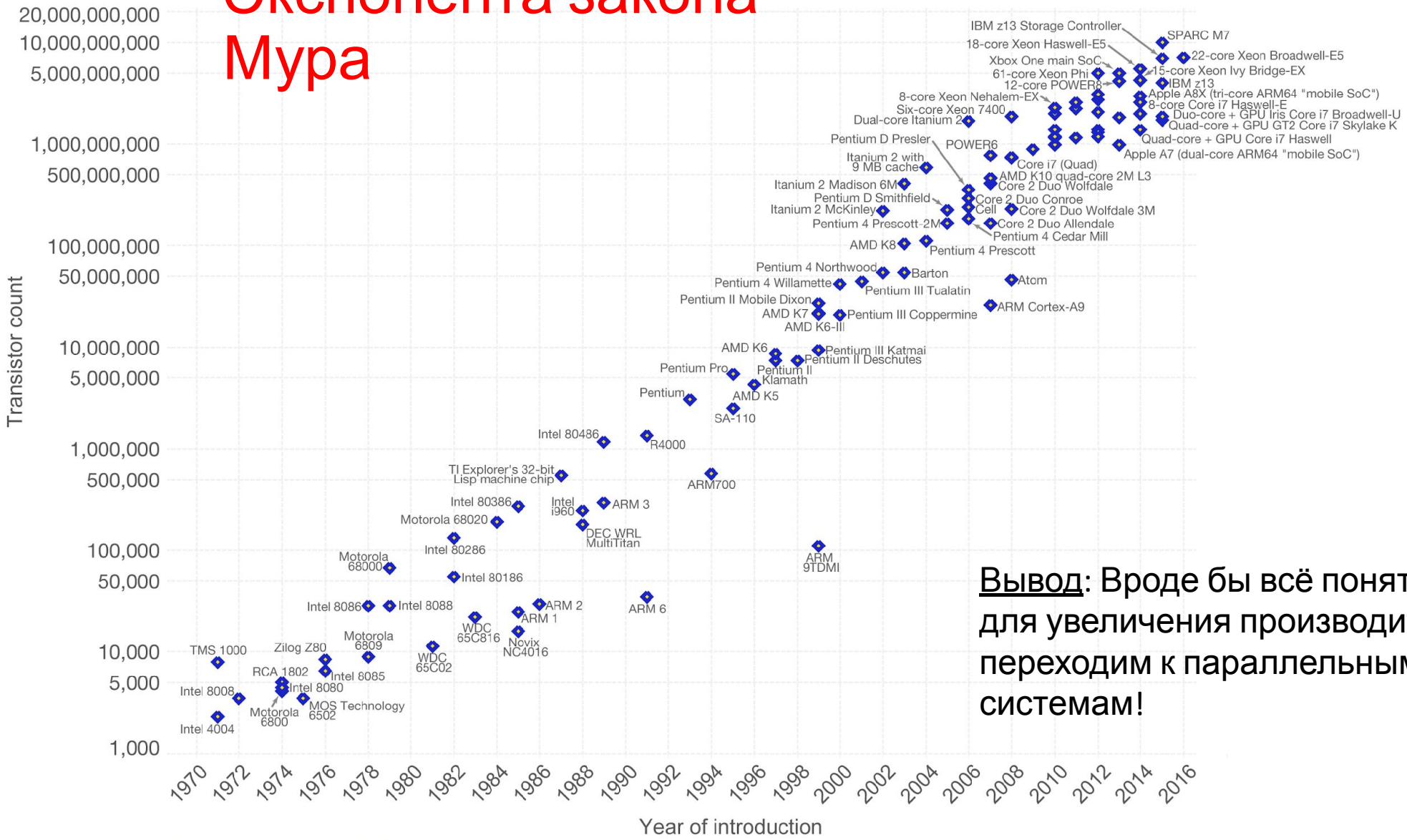
В 2007 году Мур заявил, что закон его имени, очевидно, скоро перестанет действовать из-за атомарной природы вещества (где-то 14-нанометровые технологии или чуть меньше) и

# Moore's Law – The number of transistors on integrated circuit chips (1971-2016)

Our World in Data

Moore's law describes the empirical regularity that the number of transistors on integrated circuits doubles approximately every two years. This advancement is important as other aspects of technological progress – such as processing speed or the price of electronic products – are strongly linked to Moore's law.

Экспонента закона Мура




Вывод: Вроде бы всё понятно – для увеличения производительности переходим к параллельным системам!

Data source: Wikipedia (https://en.wikipedia.org/wiki/Transistor_count)
The data visualization is available at OurWorldinData.org. There you find more visualizations and research on this topic.

# Закон Амдала

Две крайние ситуации параллельной работы:

- каждый шаг алгоритма зависит от результатов вычислений или статуса выполнения предыдущего шага; параллельные вычисления или бесполезны, или мало эффективны.
- не существует зависимости или имеется малая зависимость между параллельно выполняемыми подзадачами (их результаты не влияют или мало влияют друг на друга);

Это ситуация чрезвычайной параллельности ( embarrassingly parallel ).

Накладные расходы реализации распараллеливания: в первую очередь синхронизация и смена контекста. Для плохо распараллеливаемых задач они снижают производительность.

Легко распараллеливаемые задачи могут масштабироваться не только по ядрам одного сервера, но и горизонтально на множество серверов. Конечно, в этом случае необходимо решить проблемы сохранения и объединения результатов.

Закон Амдала это модель потенциального выигрыша в производительности при распараллеливании вычислений.

Ускорение за счёт распараллеливания алгоритма на p процессоров, по сравнению с последовательным вариантом определяется отношением времени последовательного вычисления $T_{посл}(n)$ ко времени параллельного вычисления $T_p(n)$:

$S_p(n) = T_{посл}(n) / T_p(n)$.

Здесь n применяется для параметризации вычислительной сложности решаемой задачи.

Эффективность использования процессоров в параллельном алгоритме

# Рост производительности с увеличением числа серверов 1/2

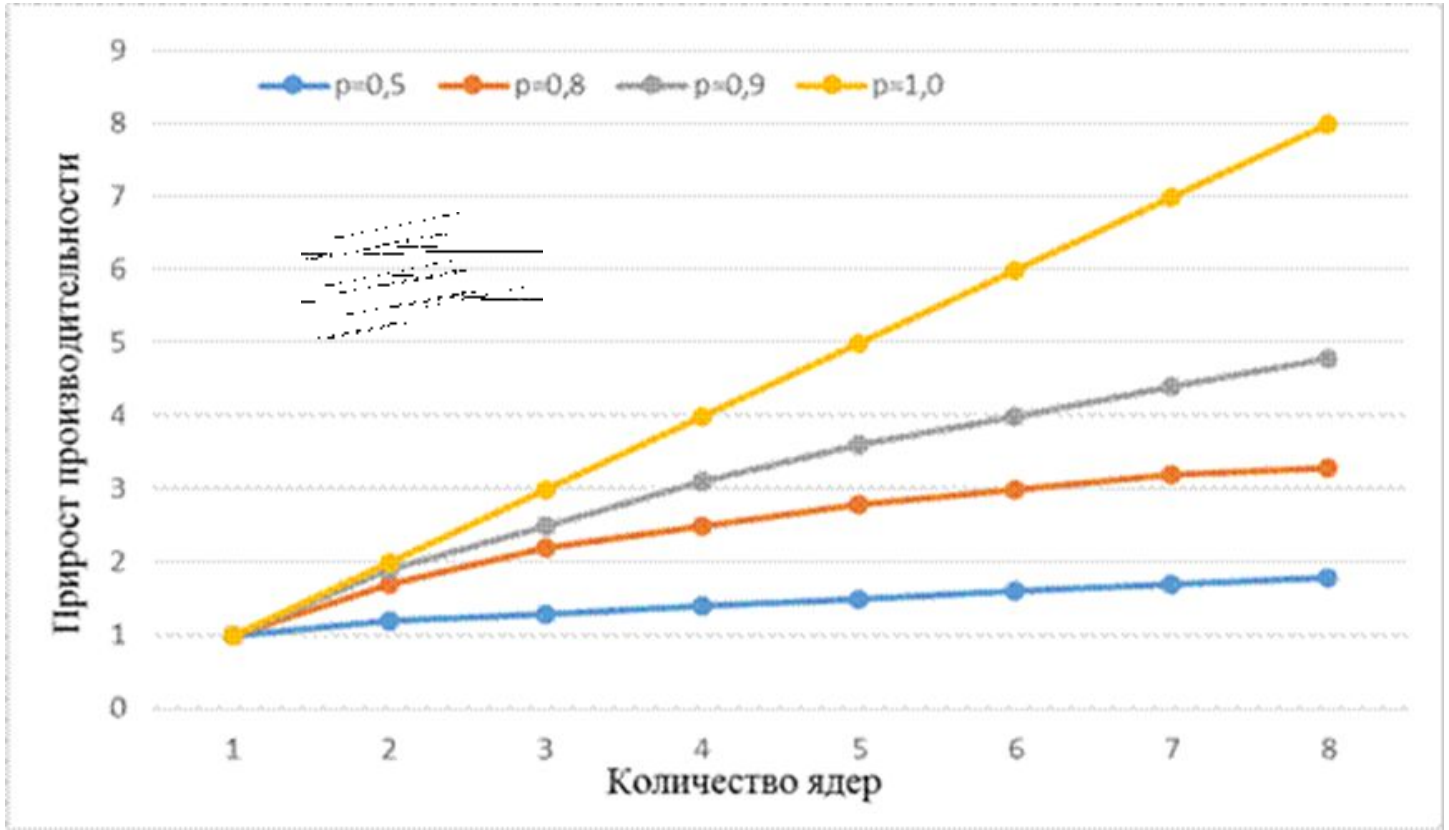
Ускорение S от параллельного выполнения программы

$S=1/(p + (1-p) / n)$

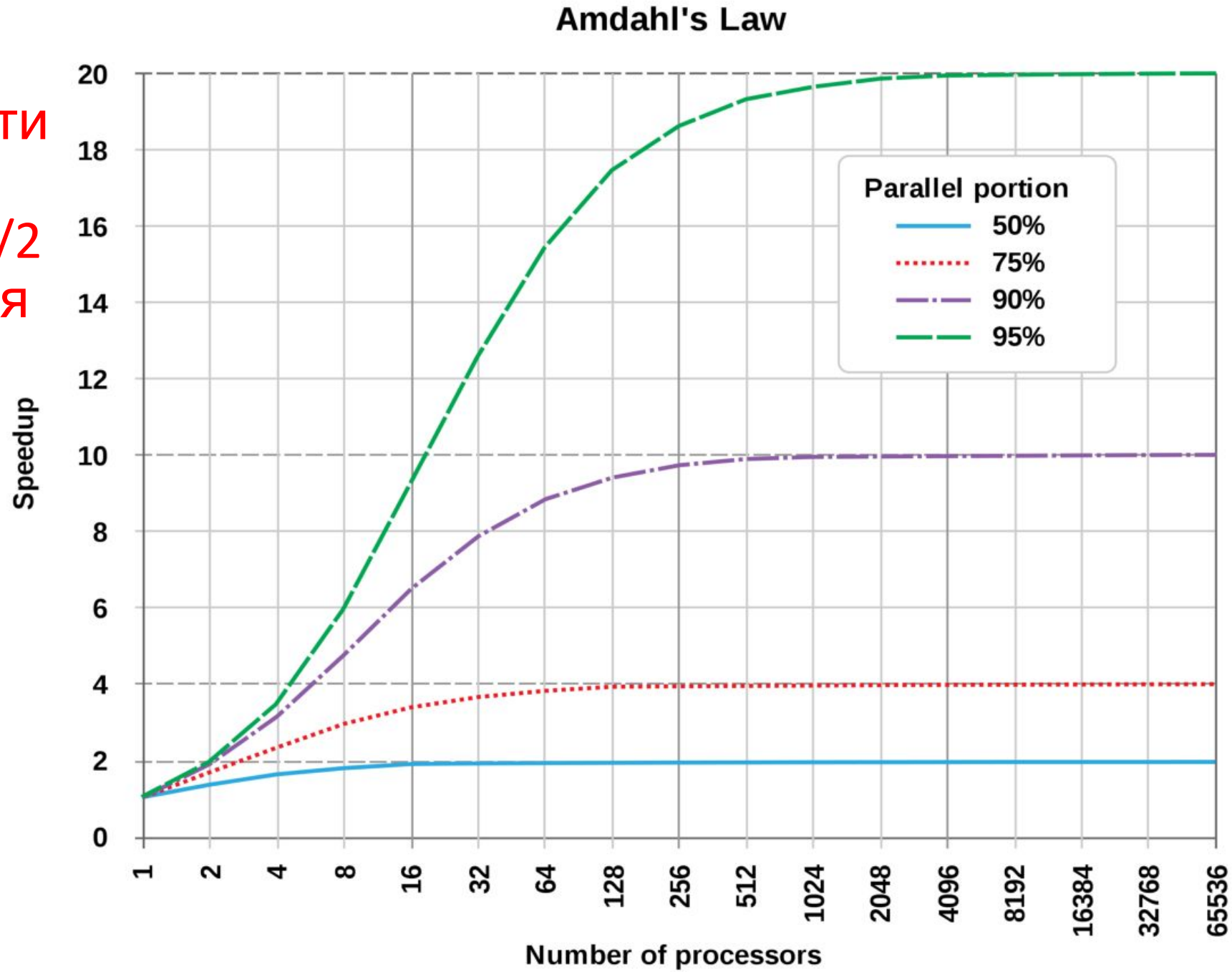
где p – доля последовательных (не распараллеливаемых) действий в программе, а (1-p) – доля распараллеливаемых действий в программы, которые могут быть равномерно формально распределены по n ядрам.

Только полностью распараллеливаемый алгоритм, не содержащий последовательных вычислений (p=1), позволяет получить линейную зависимость производительности от количества серверов.

Если доля последовательных вычислений в алгоритме равна 25 %, (p=0.75) то увеличение числа процессоров до 10 дает ускорение в всего в 3,077 раза, а увеличение числа процессоров до 1000 даст ускорение в 3,988 раза.



# Рост роизводительности с увеличением числа серверов 2/2 (логарифмическая шкала)



# **Теорема Брюера --** теорема CAP (Consistency-Availability-Partition tolerance)

В 2000 году Эрик Брюер выдвинул гипотезу, касающуюся ключевых свойств распределённых систем, которую затем доказали в MIT.

**“В распределённой системе невозможно обеспечить одновременное выполнение трёх условий: корректности, доступности, устойчивости к сбоям узлов”.**

Аналог: Предложение клиенту: "быстро, дёшево, качественно — выбирайте любые два".

# Три свойства распределённой системы

**Корректность (Consistency)**

Говорит о том, что система всегда выдаёт только логически непротиворечивые ответы. То есть не бывает такого, что вы добавили в корзину товар, а после рефреша страницы его там не видите.

**Доступность (Availability)**

Означает, что сервис отвечает на запросы, а не выдаёт ошибки о том, что он недоступен.

**Устойчивость к сбоям сети (Partition tolerance)**

Означает, что распределённая по кластеру система работает в расчёте на случаи произвольной потери пакетов внутри сети.

# Виды моделей NoSQL

- Графовые, например, neo4j, работают со сложно связанными данными в частности в областях социальные сети, графы знаний.
- Ключ-значение (key-value), например, Berkeley Db хорошо поддерживают горизонтальный шардинг.
- Столбцовые (column), например, Oracle
- Документно-ориентированные, например, Mongo DB

# Что такое JSON (1/3)

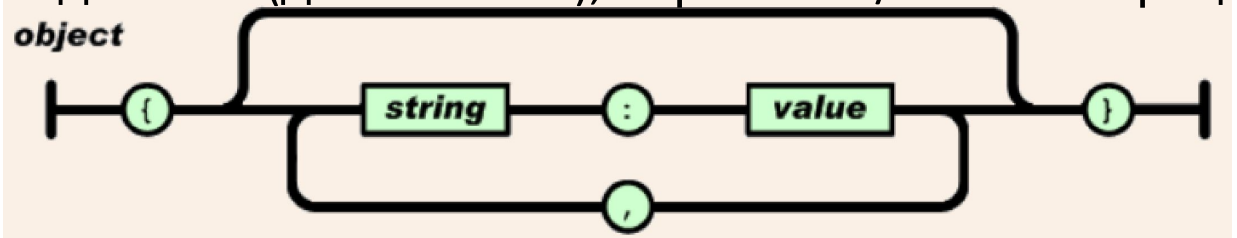
JSON (JavaScript Object Notation) -- текстовый формат, независимый от языка реализации.

JSON основан на двух универсальных структурах данных, поддерживаемых почти асеми языками программирования:

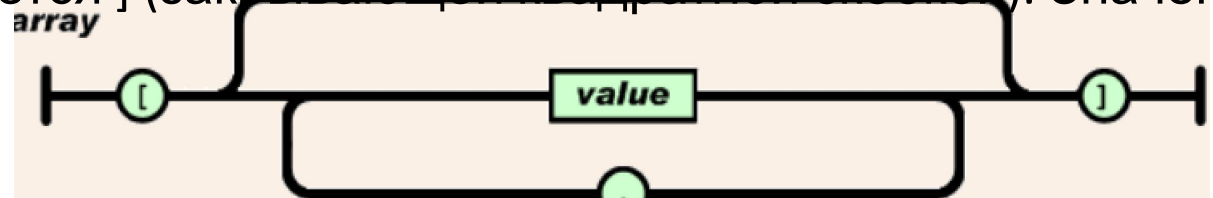
- Коллекция пар ключ/значение. В разных языках, эта концепция реализована как объект, запись, структура, словарь, хэш, именованный список или ассоциативный массив.
- Упорядоченный список значений. В большинстве языков это реализовано как массив, вектор, список или последовательность.

Синтаксис:

- Объект - неупорядоченный набор пар ключ/значение. Объект начинается с { (открывающей фигурной скобки) и заканчивается } (закрывающей фигурной скобкой). Каждое имя сопровождается : (двоеточием), пары ключ/значение разделяются , (запятой).



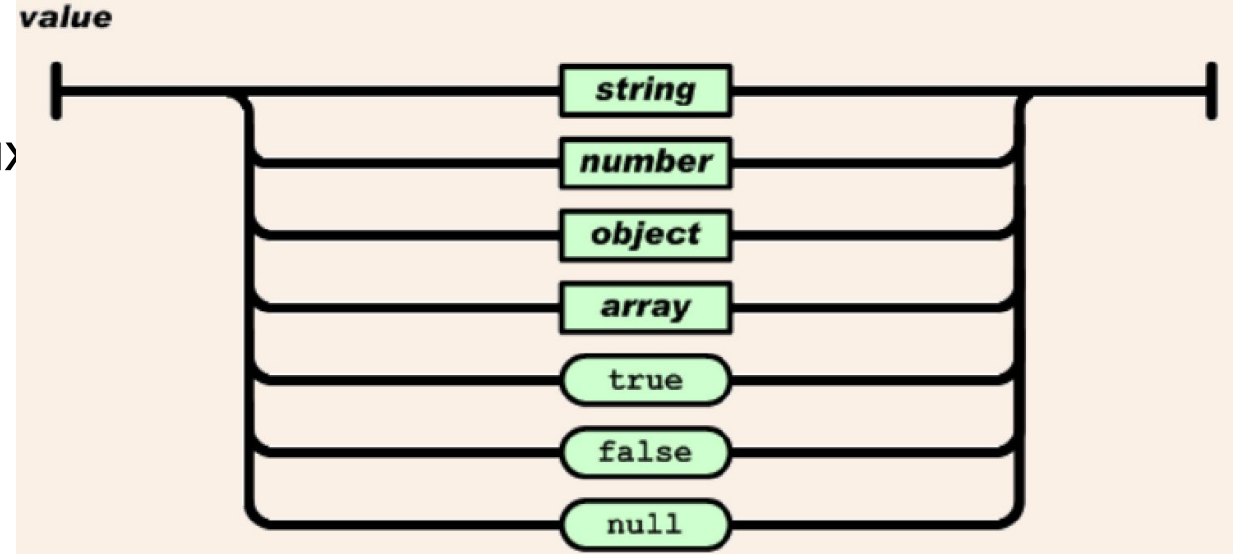

- Массив - упорядоченная коллекция значений. Начинается с [ (открывающей квадратной скобки) и заканчивается ] (закрывающей квадратной скобкой). Значения разделены , (запятой).



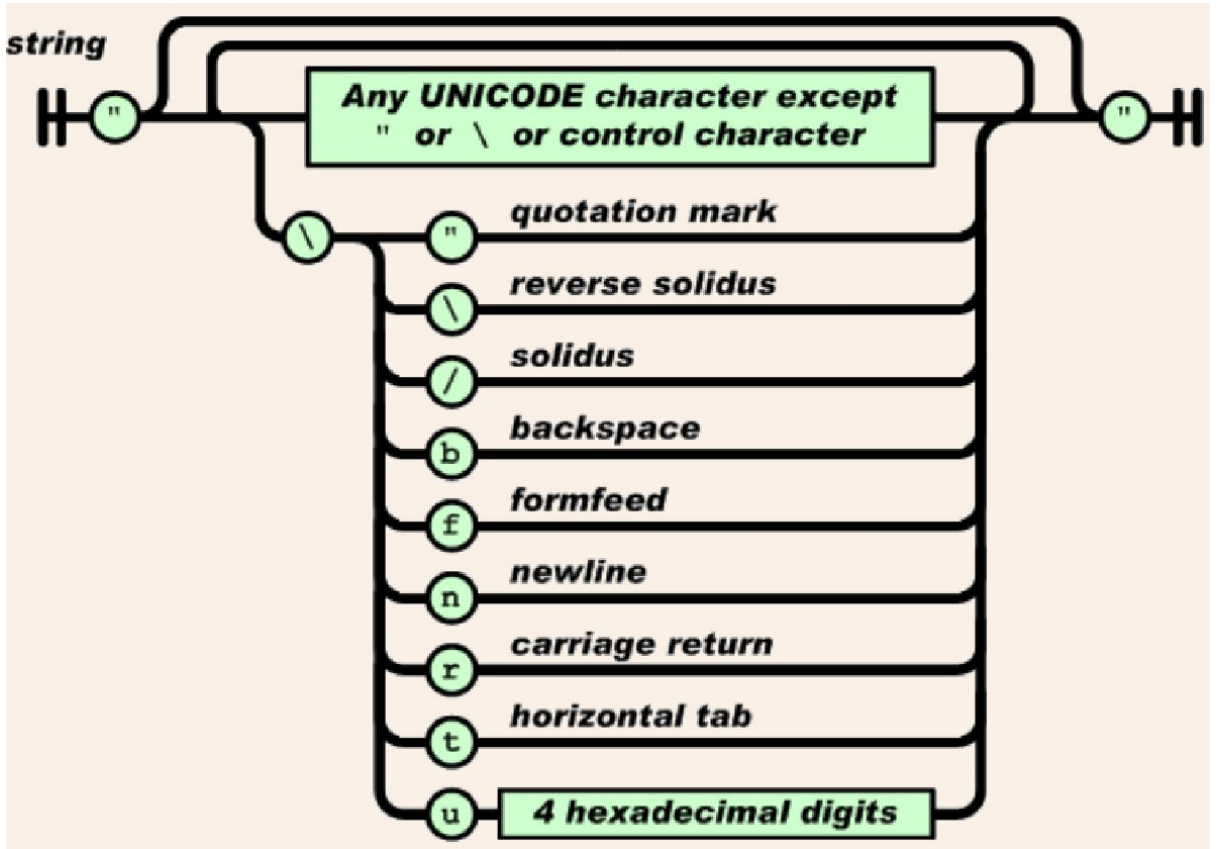
# Что такое JSON (2/3)

Значение может быть строкой в двойных кавычках, числом, true, false, null, объектом или массивом.

Эти структуры могут быть вложенными.




Строка - коллекция нуля или больше символов Unicode, заключенная в двойные кавычки, используя \ (обратную косую черту) в качестве символа экранирования. Символ представляется как односимвольная строка.
Похожий синтаксис используется в C и Java.

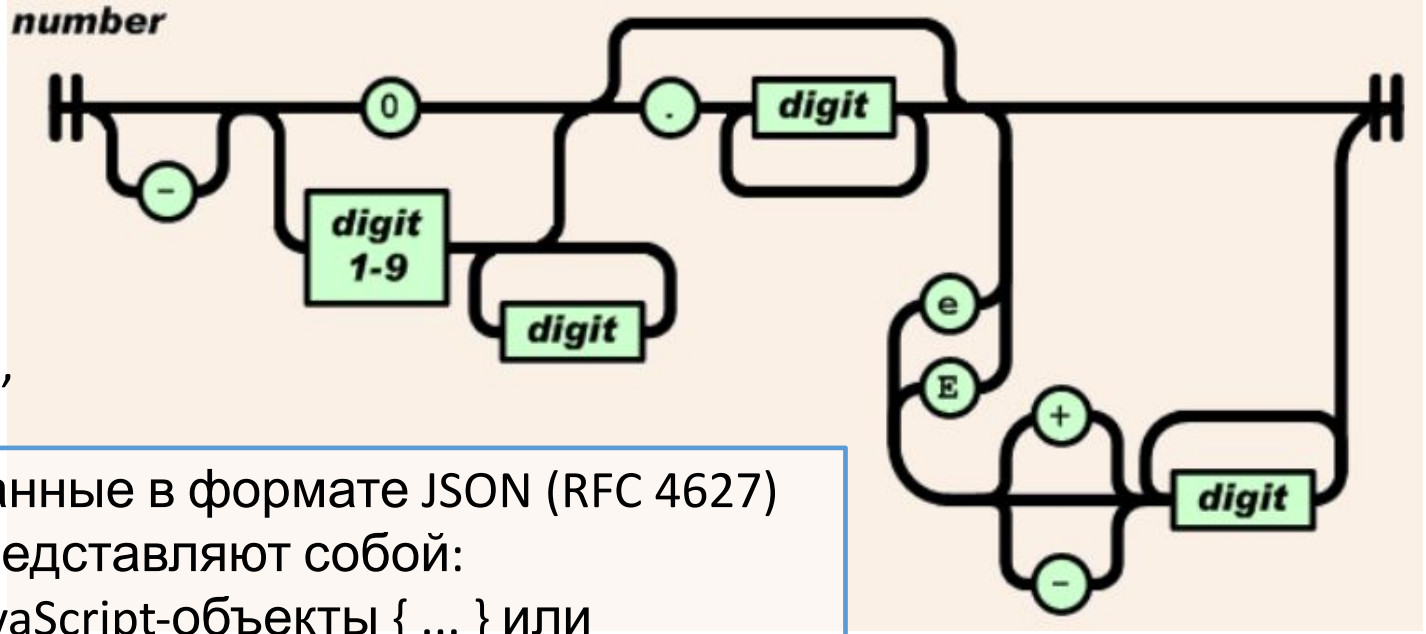


# Что такое JSON (3/3)

Число представляется так же, как в C или Java, кроме того, что используется только десятичная система счисления.

Пример:

```
{
    "firstName": "John",
    "lastName": "Smith",
    "address": {
        "streetAddress": "21 2nd Street",
        "city": "New York",
        "state": "NY",
        "postalCode": 10021
    },
    "phoneNumbers": [
        "212 555-1234",
        "646 555-4567"
    ]
}
```




- Данные в формате JSON (RFC 4627)
- представляют собой:
- JavaScript-объекты { ... } или
- Массивы [ ... ] или
- Значения одного из типов:
  - строки в двойных кавычках,
  - число,
  - логическое значение true/false,
  - null.

Или в одну строчку:

{ “first_name” : “Sammy”, “last_name”: “Shark”, “online” : true,}

# Пример документа на социальном новостном сайте

```
{   _id: ObjectID('4bd9e8e17cefd644108961bb'),
    title: 'Adventures in Databases',
    url: 'http://example.com/databases.txt',
    author: 'msmith',
    vote_count: 20,

    tags: ['databases', 'mongodb', 'indexing'],

    image: {
        url: 'http://example.com/db.jpg',
        caption: '',
        type: 'jpg',
        size: 75381,
        data: "Binary"
    },

    comments: [
        {   user: 'bjones',
            text: 'Interesting article!'
        },
        {   user: 'blogger',
            text: 'Another related article is at http://example.com/db/db.txt'
        }
    ]
}
```

Поле _id – первичный ключ

1 Теги хранятся в виде массива строк

2 Атрибут указывает на другой документ

3 Комментарии хранятся в виде массива объектов, представляющих один комментарий

# Два документа в одной коллекции MongoDB

- Поля firstname и lastsity имеются у обоих документов коллекции
- Поле likes имеется только у первого документа, а поле addresses только у второго

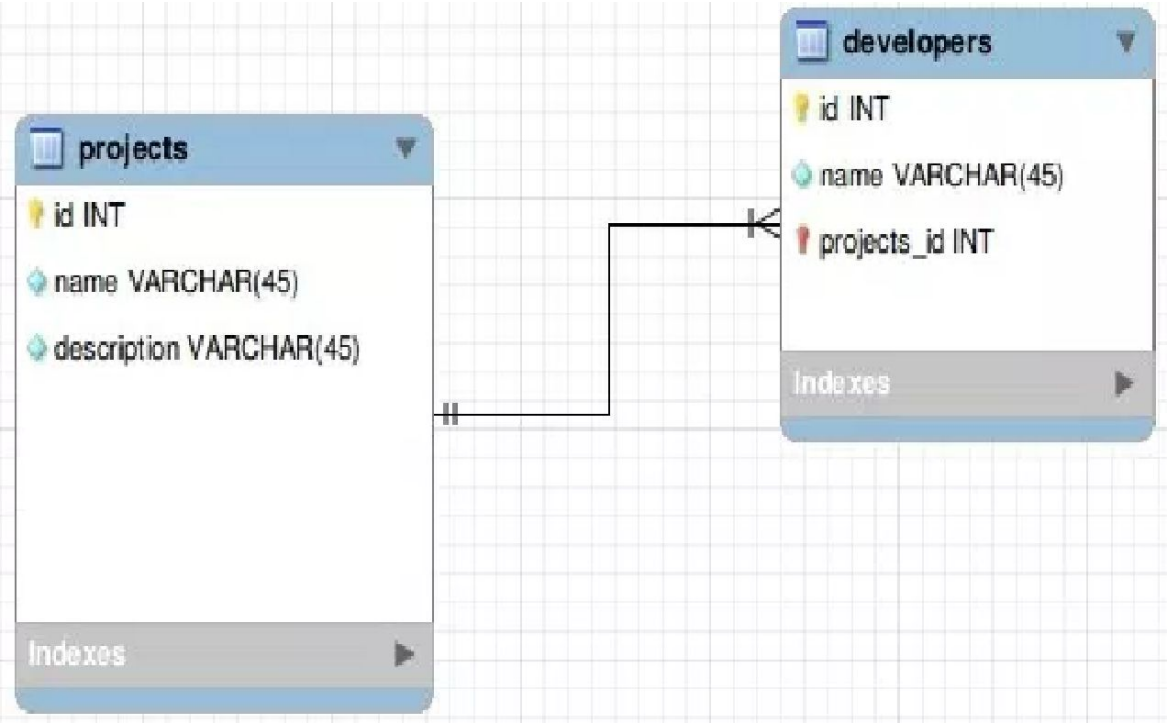
```
{ "firstname": "Martin",
  "likes": [ "Biking",
  "Photography" ],
  "lastcity": "Boston",
  "lastVisited":
}
```

```
{
  "firstname": "Pramod",
  "citiesvisited": [ "Chicago", "London", "Pune", "Bangalore" ],
  "addresses": [
    { "state": "AK",
      "city": "DILLINGHAM",
      "type": "R"
    },
    { "state": "MH",
      "city": "PUNE",
      "type": "R" }
    ],
   "lastcity": "Chicago"
}
```

# Сравнения реляционных БД и MongoDB 1/2

| Реляционная БД | MongoDB |
| --- | --- |
| База данных | База данных |
| Таблица | Коллекция |
| Ряд | Документ |
| Колонка | Поле |
| Объединенние таблиц | Встроенные документы (embedded) |
| Первичный ключ (primary key) | Первичный ключ (primary key). По умолчанию MongoDB генерирует Default key_id |

# Сравнения реляционных БД и MongoDB 2/2




В MongoDB это одна коллекция проектов со следующей структурой:

```
{
  _id: PROJECT_ID
  name: NAME_OF_PROJECT,
  developers: [
    {
      name:'DEVELOPER_NAME',
    },
    {
      name: 'DEVELOPER_NAME'
    }
  ]
}
```

# Сравнение запросов

sql

```
SELECT cust_id,
      SUM(price) as total
FROM orders
WHERE status = 'A'
GROUP BY cust_id
HAVING total > 250
```

mongo

```
db.orders.aggregate( [
  { $match: { status: 'A' } },
  {
    $group: {
      _id: "$cust_id",
      total: { $sum: "$price" }
    }
  },
  { $match: { total: { $gt: 250 } } }
] )
```

# Что такое блокчейн (Blockchain)

- Это технология хранения и передачи данных в защищённом режиме и без посредников.
- Технология Blockchain была создана в 2009 году вместе с криптовалютой Bitcoin.
- Создателем виртуальной валюты и Blockchain считают Сатоши Накамото. Однако, это псевдоним, и кто за ним стоит, один или несколько человек, не известно.
- Есть предположение, что на создание блокчейн были потрачены тысячи часов.
- Блокчейн состоит из блоков, связываемых в цепочки.
- Блоки – это данные о транзакциях, сделках и контрактах внутри системы, представленные в зашифрованном виде.
- Все блоки выстроены в цепочки, то есть они связаны между собой. Для записи нового блока, необходимо считать информацию о старых блоках. Доступ к данным есть у всех, но содержание доступно только лиуам имеющим соответствующий пароль

Два вида цепочек:

1. **Публичный** блокчейн. Открытая база данных. Такой вид блокчейна используется в криптовалюте Bitcoin. Каждый участник может записывать и читать данные.
2. **Приватный или частный** блокчейн имеет ограничения по записи/чтению данных. Могут устанавливаться приоритетные узлы. Подвид Private Blockchain – эксклюзивный блокчейн. Определяется группа лиц, занимающаяся обработкой транзакций.

# Особенности технологии блокчейн

Что такое блокчейн? Немного подробнее

- Это цепочка блоков транзакций хранящихся не на едином сервере, а одновременно у всех участников сети в виде связанных между собой копий. Информация в системе шифруется одним из самых надежных криптографическим методом – хешированием SHA-256.
- Децентрализованная база данных, устойчивая к взлому отдельных узлов, без возможности удаления/изменения данных. Каждый блок, появляющийся в системе, связан с предыдущим (в его названии заключены ссылки на предыдущий блок).
- Каждый узел сети в любой момент времени имеет актуальные данные. Прозрачность системы дает возможность отследить любую транзакцию любому пользователю. любой участник системы может увидеть записи в книге, но удалить или изменить их – нет. Однако, использовать данную информацию для мошенничества не выйдет.
- Между собой данные объединяются в неразрывную цепочку с помощью криптографии.
- Согласование данных в блокчейн похоже на работу Torrent-трекера. Вспомним, что торренты работают в режиме P2P (peer to peer – то есть, все участники сети равноправны). При скачивании файла с трекера не используется центральный сервер. Файл скачивается с компьютеров других участников торрента. Если в пиринговой сети нет участников, то файл нельзя скачать. И в блокчейн все операции проводятся между участниками напрямую за счет того, что все они подключены к

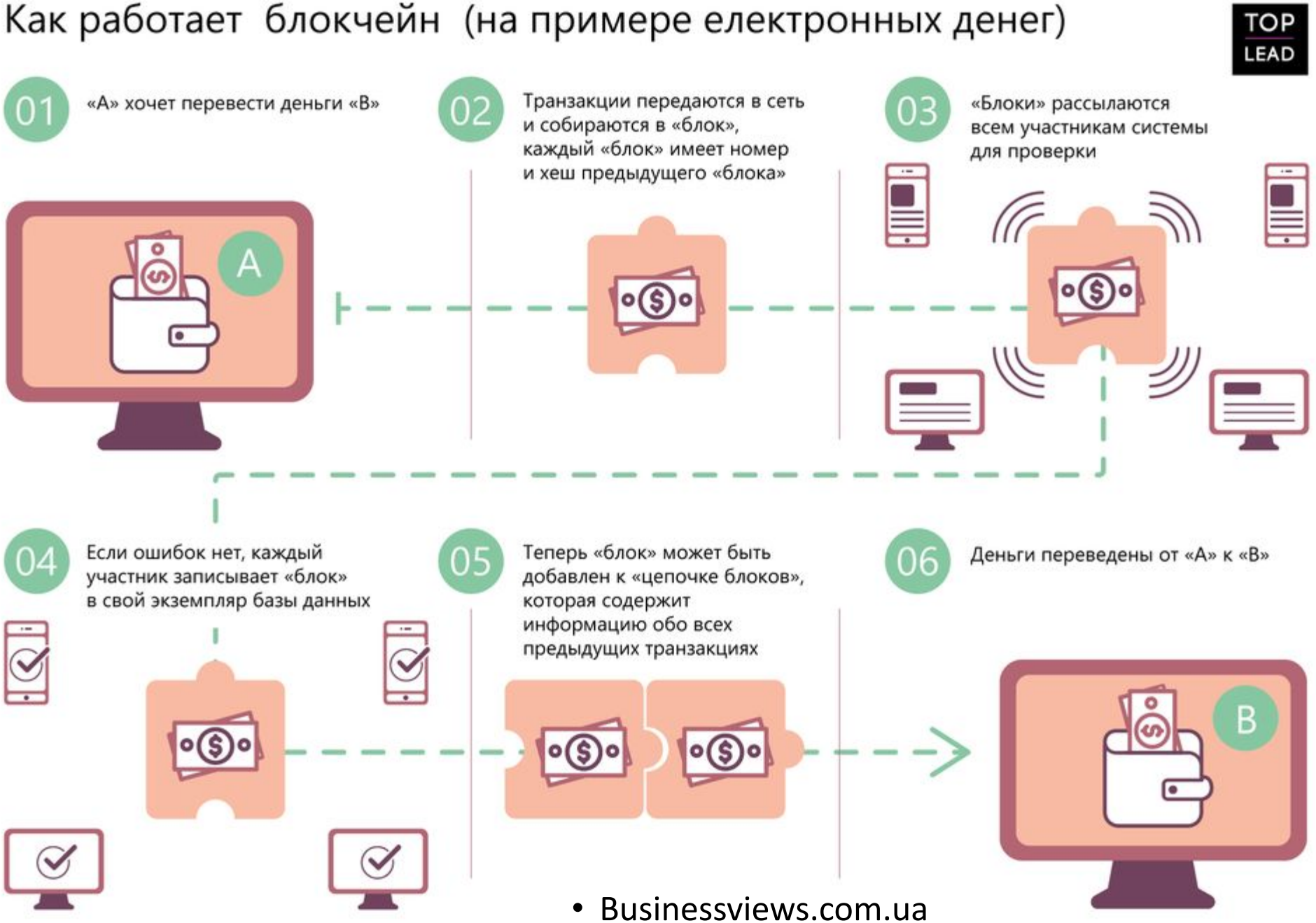
# Строим свою блокчейн базу данных

Создаём многопользовательскую учётную книгу, которую невозможно поделать. Предполагается, что есть доступ к своим строкам по записи и чтению, а к остальным только по чтению. Режим 24*7.

- Самый плохой вариант: Храним ее на одном сервере с защищенным доступом. Как напакостить – получить доступ с чужим логином и паролем. Можно убить сервер.
- Вариант 2: Добавляем шифрование данных.
- Вариант 3: Размещаем все блоки (т.е. страницы) книги на всех узлах сети. При попытке изменения содержимого система обращается за консультацией ко всем узлам. Их может быть сотни тысяч и более. Блоки разделяются между участниками с помощью пиринговых сетей, вроде используемых для раздачи торрентов. Доступ ко всем узлам требует гораздо больших ресурсов.
- Вариант 4: Добавляем ссылки из каждой страницы на все предыдущие страницы. Попытки вырвать или вклеить в книгу какую-нибудь страницу обнаруживаются при консультации с другими версиями книги. В каждый блок добавляется временная отметка в виде хэш-функции. Блоки в определенном порядке складываются в цепочки. Если попытаться переставить блоки в последовательности, то система обнаружит несоответствие структуры и хеш-суммы.

Чтобы не допустить изменения временной метки и пересчёт хэш-суммы, так, чтобы она была воспринята системой как правильная, блокчейн использует несколько способов защиты: Proof of Work (PoW, доказательство работы) и Proof of Stake (PoS, доказательство владения).

# Как работает блокчейн (на примере електронных денег)




- Businessviews.com.ua

# Плюсы и минусы блокчейна

Плюсы:

- Участники транзакции не могут обмануть друг друга.
- Не нужны посредники вроде банка, которые достаточно хорошо оплачиваются.
- Нет центрального узла, разрушив который можно сбить с ног всю систему.
- Все операции прозрачны для ее участников так как все данные вносятся в одну базу.
- Сеть может быть полностью анонимной, то есть пользователи могут быть зашифрованы блокчейн-адресами в виде случайных чисел, которые к тому же будут меняться при каждой транзакции.

Минусы:

- Очень высокие требования к быстродействию машин в узлах и к производительности сети.
- Постоянный рост объёма хранимой информации (например, блокчейн криптовалюты «биткоин» (BTC) сейчас имеет объем более 170 гигабайт).
- Отсутствие законодательной базы. Нужны стандарты.
- Большие затраты энергии.

Угрозы:

квантовый компьютер взломает электронную подпись, используемую в блокчейне биткойна, уже к 2027 году?*

# Немного практики (1/3)

- Ольга Нерода

Технология блокчейн достаточно проста для понимания. Но совсем без технических терминов обойтись не удастся. Например, придется разобраться, как работает хеширование.

Хеш — это криптографический алгоритм, который принимает на вход любые данные (файл, текст, картинка, двоичный код) и генерирует из него последовательность букв и цифр фиксированной длины. При этом, одинаковые файлы всегда дают в результате одинаковый хеш, а разные — в идеале, разный (бывают очень редкие случаи совпадения и это считается уязвимостью криптографической функции).

Например, я написала сообщение Hello world и хочу передать его своему другу. Но мне надо узнать, дошло ли оно до него в неизменном виде. Можно попросить передать сообщение обратно и сравнить, не изменилось ли оно. Но это не очень рационально: например, если наше сообщение кроме текста содержит видео в HD и весит кучу гигабайт. Поэтому для подтверждения того, что сообщение дошло без искажений, используют хеш. Если у вас и у вашего получателя хеш совпадает, значит сообщение не было изменено.

Сейчас мы с вами попробуем захешировать сообщение. Вбейте в Google «sha-256 онлайн» и в калькулятор по одной из ссылок введите любую фразу. SHA-256 — это один из алгоритмов хэширования.

Например, хеш от фразы Hello world выглядит так:

64ec88ca00b268e5ba1a35678a1b5316d212f4f366b2477232534a8aeca37f3c

А ниже хеш той же фразы hello world, но все буквы строчные:

b94d27b9934d3e08a52e52d7da7dabfac484efe37a5380ee9088f7ace2efcde9

https://thequestion.ru/questions/317223/chto-takoe-blokchein-prostymi-slovami

# Немного практики (2/3)

Ни одного совпадения! Вы также можете поэкспериментировать с любыми словами и фразами. Получите ровно те же результаты, что и я: одинаковые для одинаковых строк, и разные — для разных.

С хешем разобрались, теперь можно перейти к самому понятию блокчейн.

Блокчейн — это цепочка криптографически связанных блоков.

Давайте создадим свой блокчейн. В нем может быть что угодно, но для простоты запишем денежные транзакции.

В первом блоке у нас будут следующие записи:

Аня → Коля :: 10 рублей

Коля → Ира :: 20 рублей

Аня → Саша :: 30 рублей

Эту информацию можно перенести в Блокнот и сохранить файл с названием 1.txt. Первый блок в блокчейне криптовалют называет Genesis block, и он тоже прописывается вручную.

Копируем информацию первого блока, идем на сайт, где ранее хешировали фразу Hello world, и вставляем эти 3 строчки. Вот такой хеш у нас получился:

1a1ca4dd39417b14ea97868428da0adfd12321d1d975bec7666d10120b4f0cd8

# Немного практики (2/3)

Давайте создадим новый файл 2.txt и запишем в него еще несколько транзакций, а последней строчкой добавим результат хеширования предыдущего файла. Вот так:

Оля→ Паша :: 50 рублей

Юра → Катя :: 20 рублей

1a1ca4dd39417b14ea97868428da0adfd12321d1d975bec7666d10120b4f0cd8

Сохраняем и считаем хеш этой записи. Информацию о переводах + предыдущий хеш! Получается:

235ae3d827b9b47b2d710bb23eb1540de8101b6a24a1235a875bd6b65c986569

По аналогии создаем третий файл:

Костя → Петя :: 80 рублей

Артем → Юля :: 5 рублей

235ae3d827b9b47b2d710bb23eb1540de8101b6a24a1235a875bd6b65c986569

И все последующие файлы. Их может быть сколько угодно. И набор таких файлов-блоков позволяет представить, как выглядит простейшая концепция блокчейна.

Видео про Bitcoin:

https://www.youtube.com/watch?time_continue=4&v=RuZ80TPUF_A