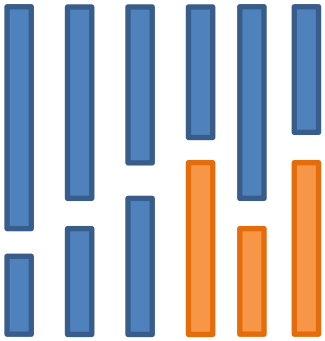


Южно-Уральский государственный университет

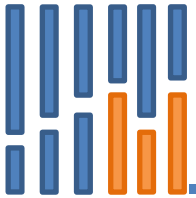
Научный семинар по информационным технологиям под руководством профессора Л.Б. Соколинского

# NoSQL СУБД

Докладчик: **К.В. Бородулин**

Научный руководитель: **Л.Б. Соколинский**

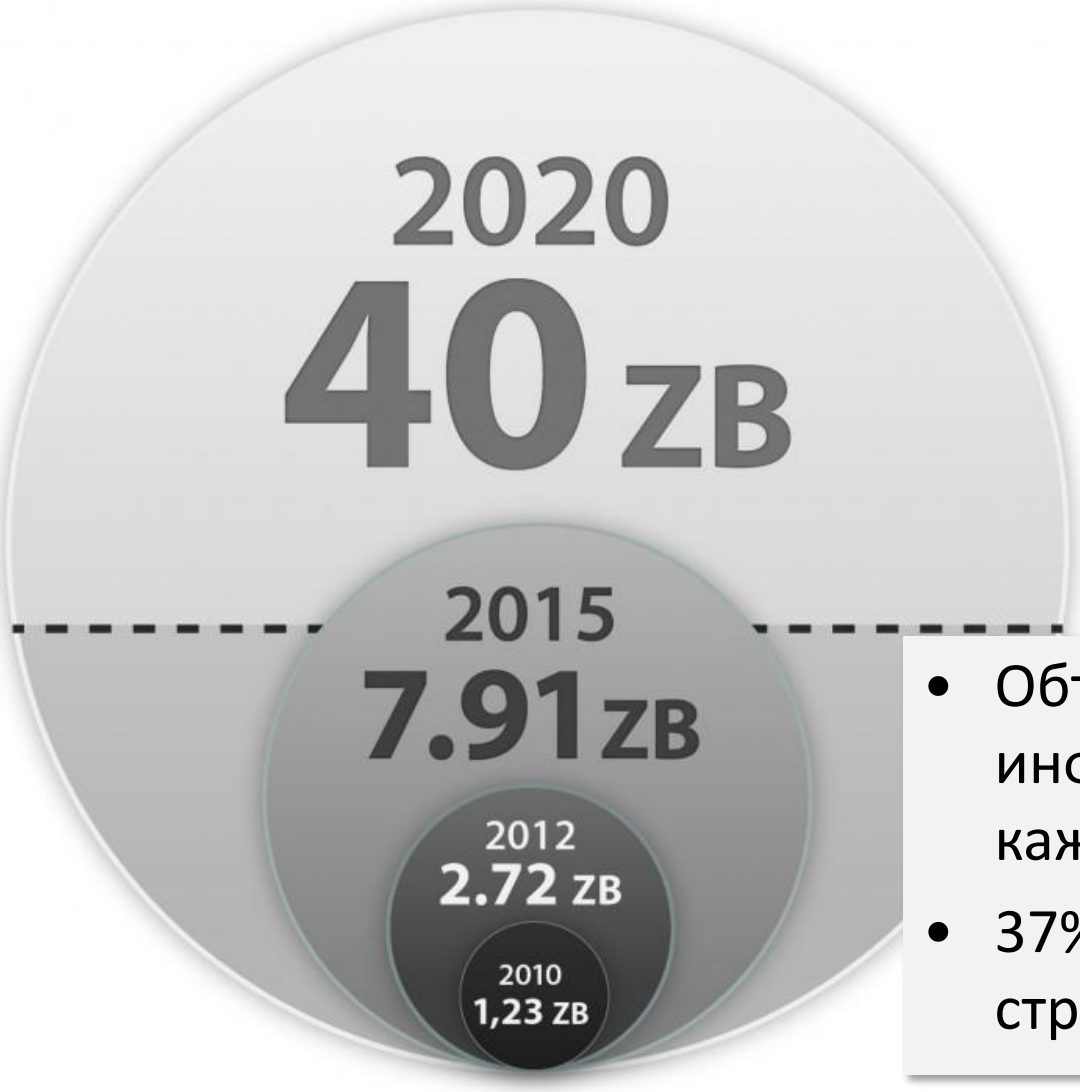
2015 г.



# Проблема больших данных

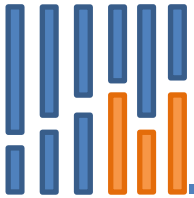
- Нью-Йоркская фондовая биржа генерирует около терабайта данных в день
- Объем хранилища социальной сети Facebook каждый день увеличивается на 500 терабайт
- Проект Internet Archive уже хранит 2 петабайта данных и прирастает 20 терабайтами в месяц
- Эксперименты на Большом адронном коллайдере генерируют около 50 терабайт данных в сутки

# Цифровая вселенная




5,247 GB
of data for every
person on the planet

- Объем хранимой информации удваивается каждые два года
- 37% данных нуждаются в структуризации и анализе

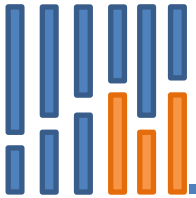


# Технологии обработки больших данных

- Колоночные СУБД
- NoSQL СУБД
- СУБД в оперативной памяти + сжатие
- Параллельные СУБД

# Пример: БД «Поставки»

## Модель «Сущность-связь»



# Реляционная модель: представление

## Поставщики

| Атрибут | Семантика |
|---|---|
| Код_П* | Код поставщика |
| Имя_П | Имя поставщика |
| Город | Город поставщика |

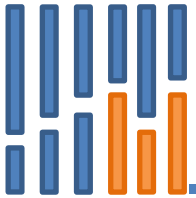
## Детали

| Атрибут | Семантика |
|---|---|
| Код_Д* | Код детали |
| Имя_Д | Имя детали |
| Цвет | Цвет детали |

## Поставки

| Атрибут | Семантика |
|---|---|
| Код_Д* | Код детали |
| Код_П* | Кодпоставщика |
| Кол_шт | Количество штук |
| Цена | Цена за штуку |

* - Первичный ключ



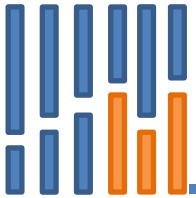
# Реляционная модель: язык запросов

- SQL

  Имена поставщиков, поставляющих хотя бы одну красную деталь

```
SELECT Имя_П
FROM П,ПД,Д
WHERE  П.Код_П = ПД.Код_П
AND ПД.Код_П = Д.Код_Д
AND Д.Цвет = ‘Красный’;
```

# КОЛОНОЧНАЯ МОДЕЛЬ (НА ПРИМЕРЕ СУБД MONETDB)



# Колоночная модель: представление

## Поставщики

| Атрибут | Семантика |
|---|---|
| Код_П* | Код поставщика |
| Имя_П | Имя поставщика |
| Город | Город поставщика |

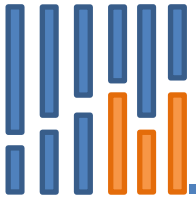
## Детали

| Атрибут | Семантика |
|---|---|
| Код_Д* | Код детали |
| Имя_Д | Имя детали |
| Цвет | Цвет детали |

## Поставки

| Атрибут | Семантика |
|---|---|
| Код_Д* | Код детали |
| Код_П* | Кодпоставщика |
| Кол_шт | Количество штук |
| Цена | Цена за штуку |

* - Первичный ключ



# Колоночная модель: хранение

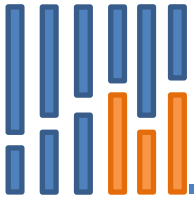
| Код_П | Имя_П | Город_П |
|---|---|---|
| 1 | Петров | Москва |
| 2 | Иванов | Воронеж |
| 3 | Сидоров | Брянск |

Поставщики

| Виртуальный ключ | Код_П |
|---|---|
| 0 | 1 |
| 1 | 2 |
| 2 | 3 |

| Виртуальный ключ | Имя_П |
|---|---|
| 0 | Петров |
| 1 | Иванов |
| 2 | Сидоров |

| Виртуальный ключ | Город_П |
|---|---|
| 0 | Москва |
| 1 | Воронеж |
| 2 | Брянск |



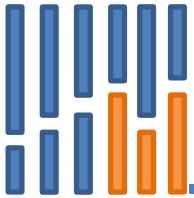
| Код_Д | Имя_Д | Цвет |
|---|---|---|
| 1 | Болт | Красный |
| 2 | Гайка | Синий |
| 3 | Шуруп | Красный |

Детали

| Виртуальный ключ | Код_П |
|---|---|
| 0 | 1 |
| 1 | 2 |
| 2 | 3 |

| Виртуальный ключ | Имя_Д |
|---|---|
| 0 | Болт |
| 1 | Гайка |
| 2 | Шуруп |

| Виртуальный ключ | Цвет |
|---|---|
| 0 | Красный |
| 1 | Синий |
| 2 | Красный |



Детали

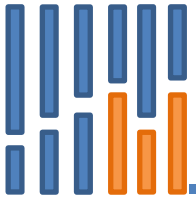
| Код_П | Код_Д | Кол_шт | Цена |
|---|---|---|---|
| 1 | 1 | 5 | 100 |
| 2 | 3 | 6 | 500 |
| 3 | 2 | 40 | 300 |

| Виртуальный ключ | Код_П |
|---|---|
| 0 | 1 |
| 1 | 2 |
| 2 | 3 |

| Виртуальный ключ | Кол_шт |
|---|---|
| 0 | 5 |
| 1 | 6 |
| 2 | 40 |

| Виртуальный ключ | Код_Д |
|---|---|
| 0 | 1 |
| 1 | 3 |
| 2 | 2 |

| Виртуальный ключ | Цена |
|---|---|
| 0 | 100 |
| 1 | 500 |
| 2 | 300 |



# Колоночная модель: язык запросов

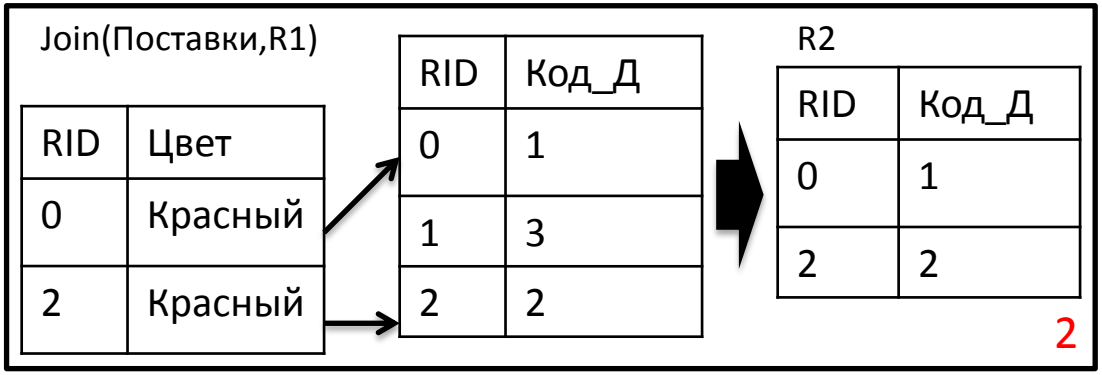
- SQL

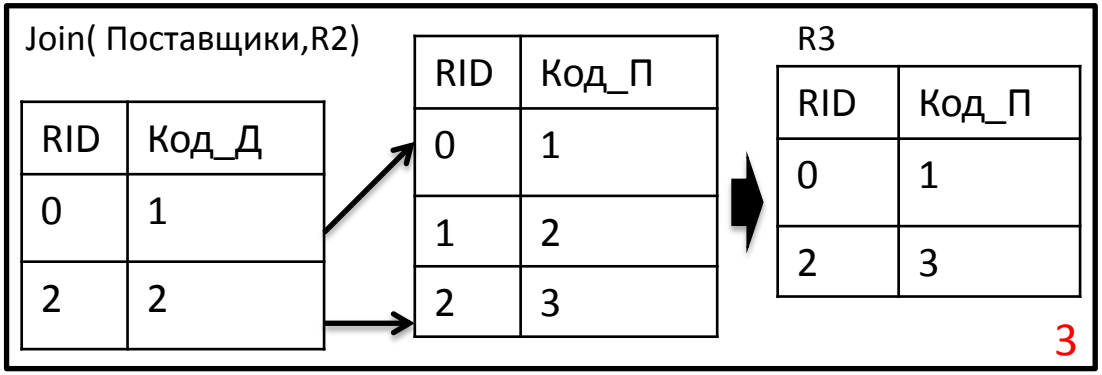
  Имена поставщиков, поставляющих хотя бы одну красную деталь

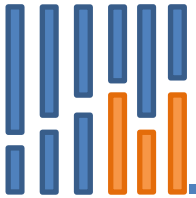
```
SELECT Имя_П
FROM П,ПД,Д
WHERE  П.Код_П = ПД.Код_П
AND ПД.Код_П = Д.Код_Д
AND Д.Цвет = ‘Красный’;
```

# Схема выполнения запроса





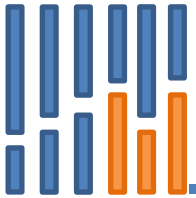





# Модели NoSQL

- Ключ-значение
- Ключ-документ
- Столбцовая модель
- Графовая модель

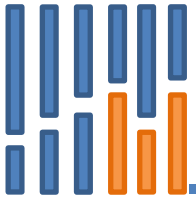
# МОДЕЛЬ КЛЮЧ-ЗНАЧЕНИЕ (НА ПРИМЕРЕ СУБД REDIS)



# Модель ключ-значение: представление

<Тип ключа> : <Идентификатор>

| | |
|---|---|
| Поставщики | П:<Код_П> |
| Детали | Д:<Код_Д> |
| Поставки | ПД:<Код_Д>:<Код_П> |

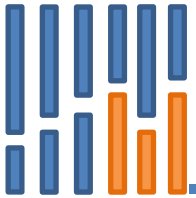


# Ввод данных

Поставщики

| Код_П | Имя_П | Город_П |
|---|---|---|
| 1 | Петров | Москва |
| 2 | Иванов | Воронеж |
| 3 | Сидоров | Брянск |

set П:1 Имя_П ‘Петров’ Город_П ‘Москва’
set П:2 Имя_П ‘Иванов’ Город_П ‘Воронеж’
set П:3 Имя_П ‘Сидоров’ Город_П ‘Брянск’

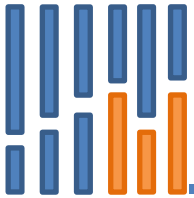


# Ввод данных

Детали

| Код_Д | Имя_Д | Цвет |
|---|---|---|
| 1 | Болт | Красный |
| 2 | Гайка | Синий |
| 3 | Шуруп | Красный |

set Д:1 Имя_Д ‘Болт’ Цвет ‘Красный’
set Д:2 Имя_Д ‘Гайка’ Цвет ‘Синий’
set Д:3 Имя_Д ‘Шуруп’ Цвет ‘Красный’



# Ввод данных

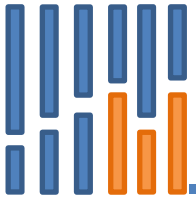
Поставки

| Код_П | Код_Д | Кол_шт | Цена |
|---|---|---|---|
| 1 | 1 | 5 | 100 |
| 2 | 3 | 6 | 500 |
| 3 | 2 | 40 | 300 |

set ПД:1:1 Кол_шт ‘5’ Цена ‘100’

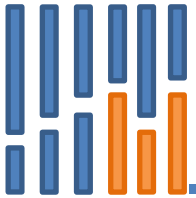
set ПД:2:3 Кол_шт ‘6’ Цена ‘500’

set ПД:3:2 Кол_шт ‘40’ Цена ‘300’



# Операции

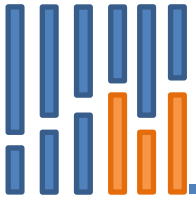
- Set – добавление записей
- Get – получение значения по ключу
- Delete – удаление записи



# Пример запроса

```
parts = keys('Д:*') #находим ключи всех деталей
for part in parts: #Для каждой детали
  if get(part,’Цвет’) = ‘Красный’: #Находим цвет детали, и если он красный
    part_id = part.split(‘:’)[1]] #Находим код детали
    supplies = keys('ПД:%s:*’ & part_id) # Находим поставки с кодом детали
      for supply in supplies: #Для каждой детали
        supplier_id = supply.split(‘:’)[2] #Находим код поставщика в поставке
        print get(‘П:%s’ & supplier_id,’Имя_П’) #Выводим имя поставщика
```

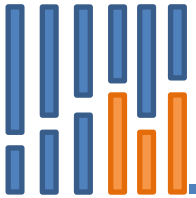
# МОДЕЛЬ КЛЮЧ-ДОКУМЕНТ (НА ПРИМЕРЕ СУБД MONGODB)



# Модель ключ-документ: представление

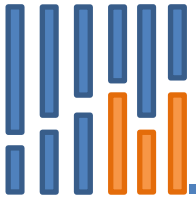
– Коллекции документов

| | |
|---|---|
| Поставщики | Suppliers<br>{_id:<идентификатор>, Имя_П: <Имя>, Город_П: <Город>} |
| Детали | Parts<br>{_id: <идентификатор>, Имя_Д: <Имя>, Цвет: <Цвет>} |
| Поставки | Supplies<br>{SID: <идентификатор поставщика>, PID: <идентификатор детали>, Кол_шт: <кол-во деталей>, Цена: <цена за штуку>} |



# Пример документа

```
{ title: 'Babylon 5',
seasons: [
  {
    season_number: 1,
    episodes:
     [
       {
         episode_number: 1
         title: 'Midnight on the Firing Line'
         reviews:[{...}],
         cast_members: [{...}]
       },
    ...
     ]
  },
  ...
 ]}
```

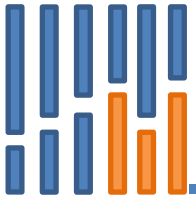


# Ввод данных

Поставщики

| Код_П | Имя_П | Город_П |
|---|---|---|
| 1 | Петров | Москва |
| 2 | Иванов | Воронеж |
| 3 | Сидоров | Брянск |

db.suppliers.instert({_id:1, Имя_П:‘Петров’, Город_П: ‘Москва’})
db.suppliers.instert({_id:2, Имя_П: ‘Иванов’, Город_П: ‘Воронеж’})
db.suppliers.instert({_id:3, Имя_П: ‘Сидоров’, Город_П: ‘Брянск’})

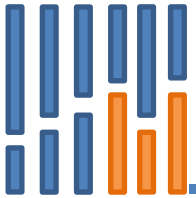


# Ввод данных

Детали

| Код_Д | Имя_Д | Цвет |
|---|---|---|
| 1 | Болт | Красный |
| 2 | Гайка | Синий |
| 3 | Шуруп | Красный |

```
db.parts.insert({_id: 1, Имя_Д: ‘Болт’, Цвет: ‘Красный’})
db.parts.insert({_id: 2, Имя_Д: ‘Гайка’, Цвет: ‘Синий’})
db.parts.insert({_id: 3, Имя_Д : ‘Шуруп’, Цвет: ‘Красный’})
```



# Ввод данных

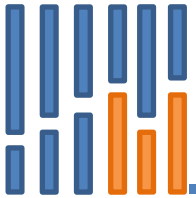
Поставки

| Код_П | Код_Д | Кол_шт | Цена |
|---|---|---|---|
| 1 | 1 | 5 | 100 |
| 2 | 3 | 6 | 500 |
| 3 | 2 | 40 | 300 |

db.parts.insert({SID: 1, PID: 1, Кол_шт: 5, Цена: 100})

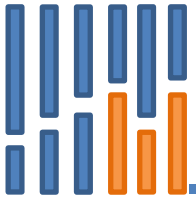
db.parts.insert({SID: 2, PID: 3, Кол_шт: 6, Цена: 500})

db.parts.insert({SID: 3, PID: 2, Кол_шт: 40, Цена: 300})



# Операции

- Insert – добавление документа
- Find – получение документа по условию
- Update – обновление полей документа
- Delete – удаление документа

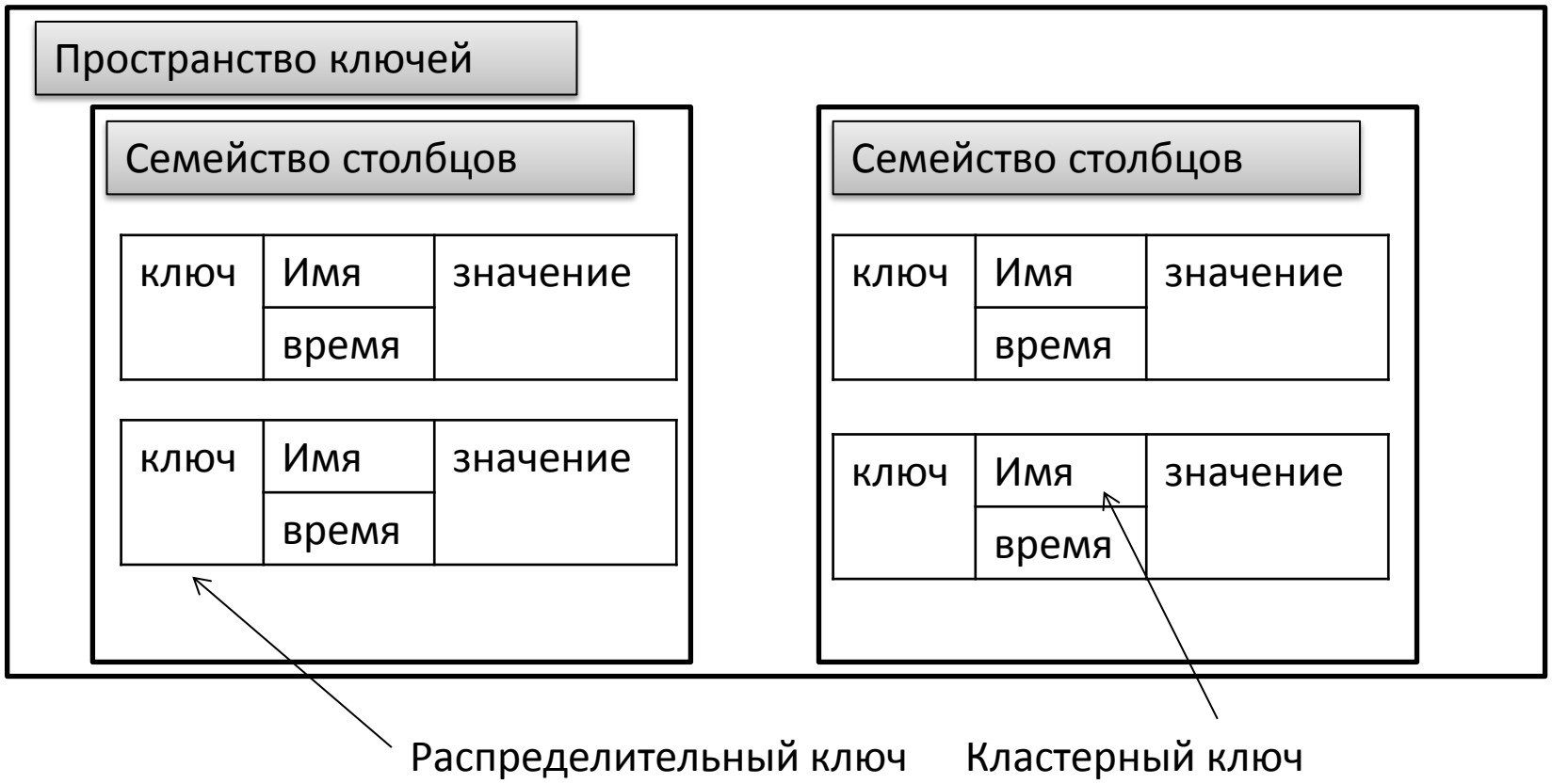


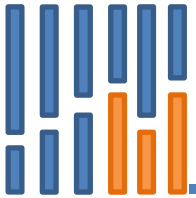
# Пример запроса

```
parts = db.parts.find({‘Цвет’: ‘Красный’}) #находим все детали красного цвета
for part in parts: #Для каждой детали
    supplies = db.supplies.find({‘PID’: part[‘id’]}) # Находим поставки с кодом детали
        for supply in supplies: #Для каждой детали
         supplier = db.suppliers.find_one({‘id’: supply[‘SID’])
            print (supplier[‘Имя_П’]) #Выводим имя поставщика
```

# СТОЛБЦОВАЯ МОДЕЛЬ (НА ПРИМЕРЕ СУБД CASSANDRA)

# Столбцовая модель : представление





# Ввод данных

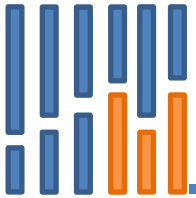
CREATE TABLE Suppliers ( ID int, Name text, City text) PRIMARY KEY (ID,Name)
ID – Распределительный ключ
Name – Кластерный ключ

INSERT INTO Suppliers(ID,Name,City) VALUES (1,’Петров’, ‘Москва’)
INSERT INTO Suppliers (ID,Name,City) VALUES (2, ’Иванов’, ‘Воронеж’)
INSERT INTO Suppliers (ID,Name,City) VALUES (3, ’Сидоров’, ‘Брянск’)

| | Петров:City |
|---|---|
| 1 | Москва |

| | Иванов:City |
|---|---|
| 2 | Воронеж |

| | Сидоров:City |
|---|---|
| 3 | Брянск |



# Ввод данных

CREATE TABLE Parts( ID int, Name text, Color text) PRIMARY KEY (ID,Name)

ID – Распределительный ключ

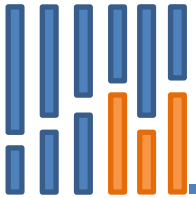
Name – Кластерный ключ

INSERT INTO Parts (ID,Name,Color) VALUES (1,’Болт’, ‘Красный’)
INSERT INTO Parts (ID,Name,Color) VALUES (2, ’Гайка’, ‘Синий’)
INSERT INTO Parts (ID,Name,Color) VALUES (3, ’Шуруп’, ‘Красный’)

| | Болт:Color |
|---|---|
| 1 | Красный |

| | Гайка:Color |
|---|---|
| 2 | Синий |

| | Шуруп:Color |
|---|---|
| 3 | Красный |



# Ввод данных

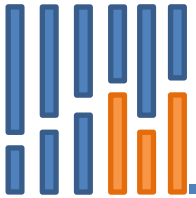
CREATE TABLE Supplies ( SID int, PID int, Amount int, Price int)
PRIMARY KEY (SID,PID)
SID – Распределительный ключ
PID– Кластерный ключ

INSERT INTO Supplies(SID, PID, Amount, Price) VALUES (1,1,5,100)
INSERT INTO Supplies (SID, PID, Amount, Price) VALUES (2,3, 6, 500)
INSERT INTO Supplies (SID, PID, Amount, Price) VALUES (3,2, 40, 300)
INSERT INTO Supplies (SID, PID, Amount, Price) VALUES (2,2, 2, 50)

| | 1:Amount | 1:Price |
|---|---|---|
| 1 | 5 | 100 |

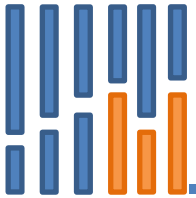
| | 2:Amount | 2:Price |
|---|---|---|
| 3 | 40 | 300 |

| | 3:Amount | 3:Price | 2:Amount | 2:Price |
|---|---|---|---|---|
| 2 | 6 | 500 | 6 | 50 |



# Столбцовая модель: язык запросов

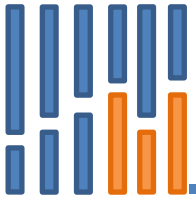
- CQL
  - Ограниченное подмножество операций SQL
  - CREATE TABLE – создание семейства столбцов
  - Нет операции INSERT
  - UPDATE – Обновление или добавление значений
  - DELETE - Удаление значений
  - SELECT
    - Нет соединения
    - Ограниченное множество условий



# Пример запроса

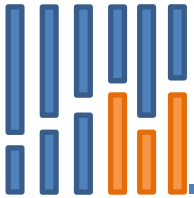
```
part_ids = execute('SELECT Parts.ID FROM Parts WHERE Parts.Color = ‘Красный’') #находим все идентификаторы деталей красного цвета
for part_id in part_ids :
  supplier_ids =client.execute( 'SELECT Supplies.SID FROM Supplies WHERE Supplies.PID=%s’ & part_id)  # Находим коды поставщиков поставки с кодом детали
        for supplier_id in supplier_ids: #Для каждой детали
          supplier_name = execute ('SELECT Supplies.Name FROM Suppliers WHERE Suppliers.PID=%s’ & supplier_id)
          print (supplier_name) #Выводим имя поставщика
```

# ГРАФОВАЯ МОДЕЛЬ (НА ПРИМЕРЕ СУБД NEO4J)



# Графовая модель: представление

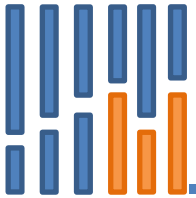
- Представление в виде графа
  - Узлы
    - метка (Тип узла)
    - Атрибуты
  - Отношения (ребра)
    - метка (тип отношения)
    - Атрибуты



# Ввод данных

```
CREATE (Supplier:Suppliers {Id: 1, Name: ‘Петров’, City: ‘Москва’})
CREATE (Supplier:Suppliers {Id:2, Name: ‘Иванов’, City: ‘Воронеж’})
CREATE (Supplier:Suppliers {Id:3, Name: ‘Сидоров’, City: ‘Брянск’})
```

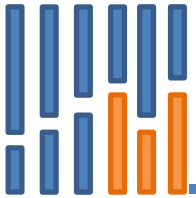




# Ввод данных

```
CREATE (part:Parts {Id:1, Name: ‘Болт’, Color: ‘Красный’})
CREATE (part:Parts {Id:2, Name: ‘Гайка’, City: ‘Синий’})
CREATE (part:Parts {Id:3, Name: ‘Шуруп’, City: ‘Красный’})
```





# Ввод данных

- Создание отношений

**MATCH** (Supplier:Suppliers [Id: 1])

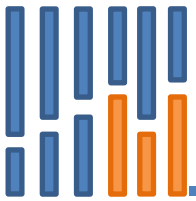
**CREATE**(Supplier) – [:SELLS {amount: 5, Price:100}] -> (Part:Parts [id: 1])

**MATCH** (Supplier:Suppliers [Id: 2])

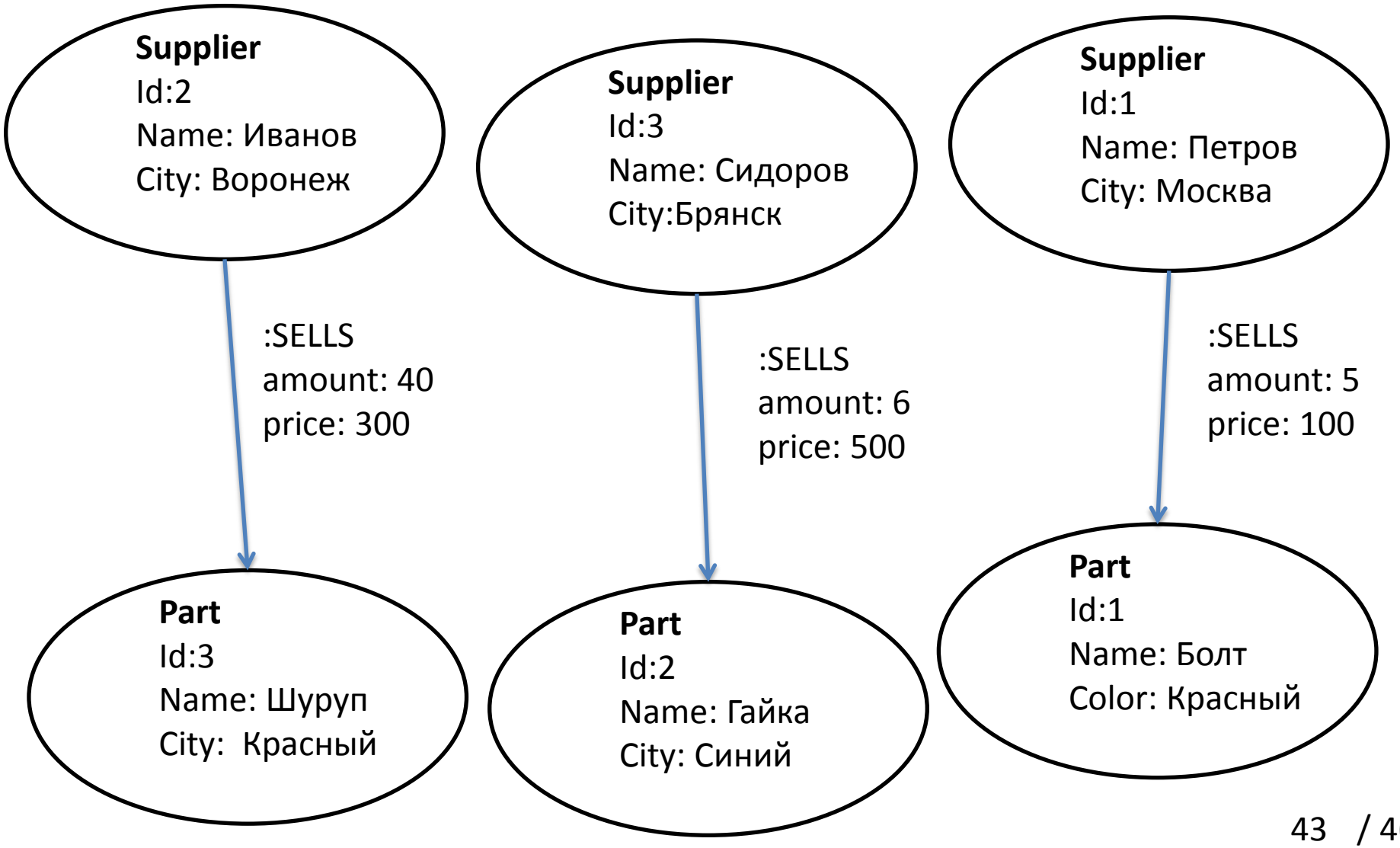
**CREATE**(Supplier) – [:SELLS {amount: 6, Price:500}] -> (Part:Parts [id: 3])

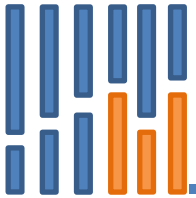
**MATCH** (Supplier:Suppliers [Id: 3])

**CREATE**(Supplier) – [:SELLS {amount: 40, Price:300}] -> (Part:Parts [id: 2])



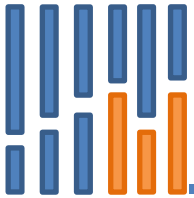
# Графическое представление





# Графовая модель: язык запросов

- Cypher
  - CREATE (node:Label)
  - CREATE (node:Label) –[:REL_TYPE] -> (node2:Label2)
  - MATCH (n)
  - MATCH (n) WHERE cond



# Пример запроса

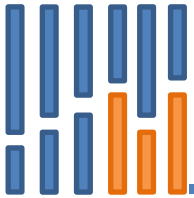
MATCH (Supplier:Suppliers) –[:SELLS] ->
(Part:Parts {Color: ‘Красный’})

RETURN Supplier.Name

# Плюсы и минусы NoSQL

- Эффективное распараллеливание
- Эффективное сжатие колонок
- Эффективное добавление данных

- Примитивное множество операций
- Отсутствует высокоуровневый язык запросов
  - Отсутствует генерация планов
  - Отсутствует оптимизация запроса



Спасибо за внимание!