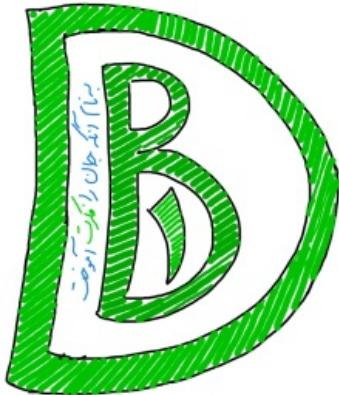


# به نام آنکه جان را فکرت آموخت



## بخش یازدهم: پایگاهداده‌های غیرابطه‌ای (NoSQL)

مرتضی امینی

نیمسال اول ۹۷-۹۸

□ پیدایش پایگاه‌های داده‌ی NoSQL

□ تفاوت پایگاه‌های داده‌ی رابطه‌ای و غیر رابطه‌ای (NoSQL)

□ انواع مدل‌های داده‌ی NoSQL

□ کلید-مقدار (Key-Value Store)

□ سندگرا (Document Based)

□ ستونی (Column Based)

□ مبتنی بر گراف (Graph Based)



### □ پیدایش پایگاه‌های داده‌ی NoSQL

- تفاوت پایگاه‌های داده‌ی رابطه‌ای و غیر رابطه‌ای (NoSQL)
- انواع مدل‌های داده‌ی NoSQL
  - کلید-مقدار (Key-Value Store)
  - سند‌گرا (Document Based)
  - ستونی (Column Based)
  - مبتنی بر گراف (Graph Based)



# گسترش برنامه‌های کاربردی جدید

بخش یازدهم: پایگاه داده‌های NoSQL

۴

□ ظهر و گسترش برنامه‌های استفاده کننده از فناوری‌های جدید



□ اینترنت

□ ابر

□ تلفن‌های هوشمند

□ رسانه‌های اجتماعی

□ داده‌های حجمی

□ اینترنت اشیاء

□ ...



# ویژگی مشترک برنامه‌های کاربردی جدید

بخش یازدهم: پایگاه داده‌های NoSQL

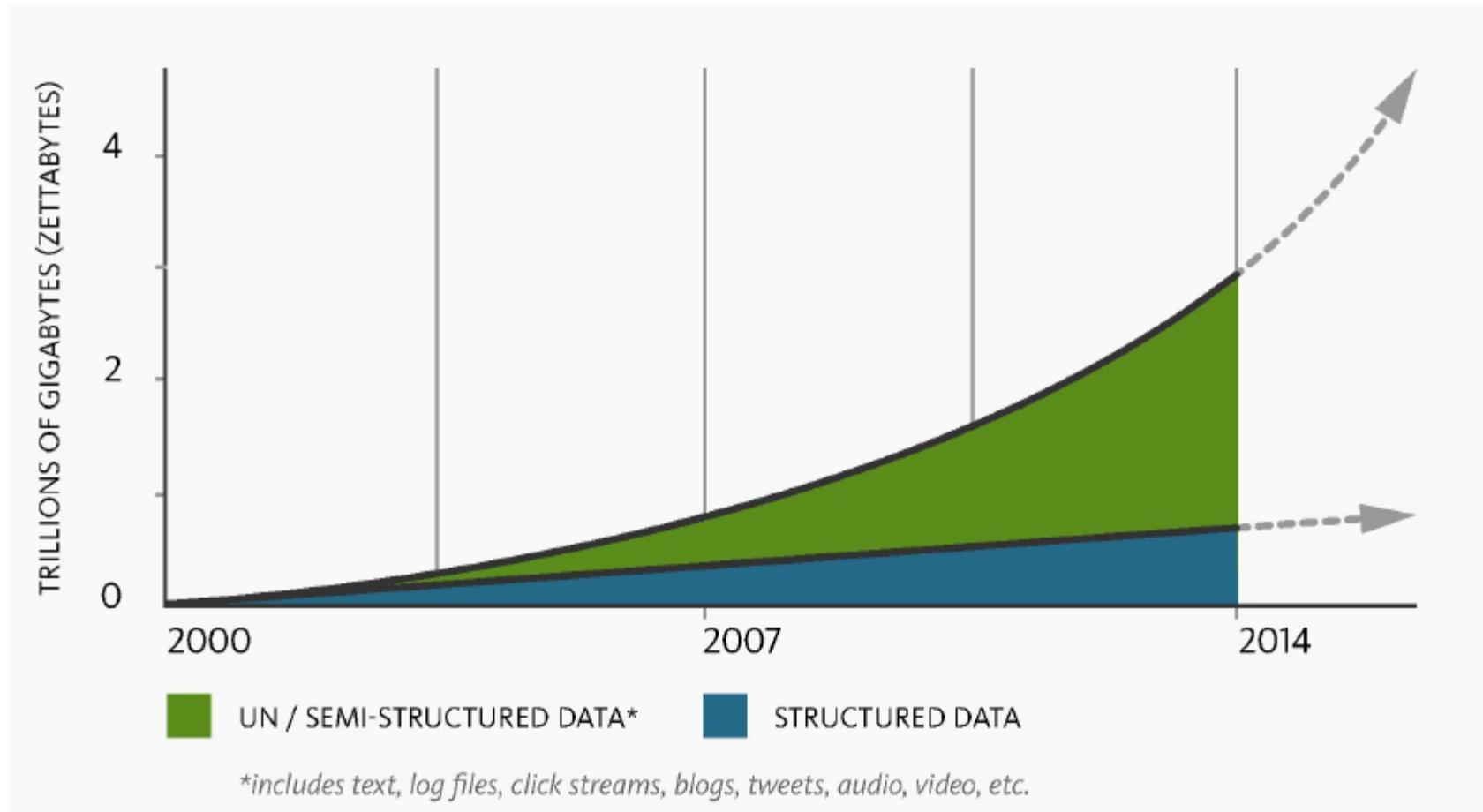
۵

- پشتیبانی همزمان از تعداد زیادی کاربر توزیع شده (دها هزار یا میلیون‌ها)
- پاسخ‌دهی برخط به کاربران و سرویس‌دهی شبانه‌روزی
- حجم بالای داده
- سرعت بالای تولید داده (نیاز به به روزرسانی مکرر)
- پشتیبانی از انواع داده (ساختیافته، ناساختیافته و نیمه‌ساختیافته)

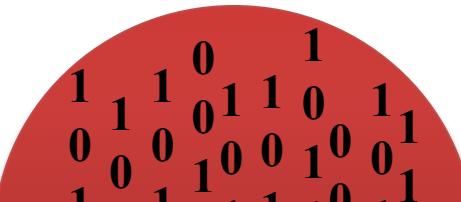


## بخش بیازدهم: پایگاه داده‌های NoSQL

۶



NoSQL = Not Only SQL



1 1 1 0 1  
0 1 0 1 1 0 1  
0 0 1 0 0 1 0 1  
1 1 0 1 0 1 0 1



0 0 1 0 1 0 0 1  
0 0 1 0 0 0 1 1  
0 0 0 0 0 0 0 1  
0 0 0 0 0 0 0 1

- مطرح شدن مفهوم NoSQL در سال ۱۹۹۸ به عنوان پایگاه داده‌ی رابطه‌ای مبتنی بر فایل که استفاده از SQL را حذف کرده بود. در حال حاضر به پایگاه‌داده‌های غیررابطه‌ای گفته می‌شود.
- رقابت با پایگاه‌های داده‌ی رابطه‌ای از سال ۲۰۰۹
- دلایل ایجاد پایگاه داده‌های NoSQL:
- گسترش برنامه‌های کاربردی جدید / افزایش حجم داده / افزایش کاربران



## بخش یازدهم: پایگاه داده‌های NoSQL

۸

□ پیداپیش پایگاه‌های داده‌ی NoSQL

## □ تفاوت پایگاه‌های داده‌ی رابطه‌ای و غیررابطه‌ای (NoSQL)

□ انواع مدل‌های داده‌ی NoSQL

□ کلید-مقدار (Key-Value Store)

□ سندگرا (Document Based)

□ ستونی (Column Based)

□ مبتنی بر گراف (Graph Based)



# یادآوری: ویژگی‌های پایگاه‌های داده‌ی رابطه‌ای

بخش بیانیه: پایگاه داده‌های NoSQL

۹

## □ در پایگاه‌های داده‌ی رابطه‌ای

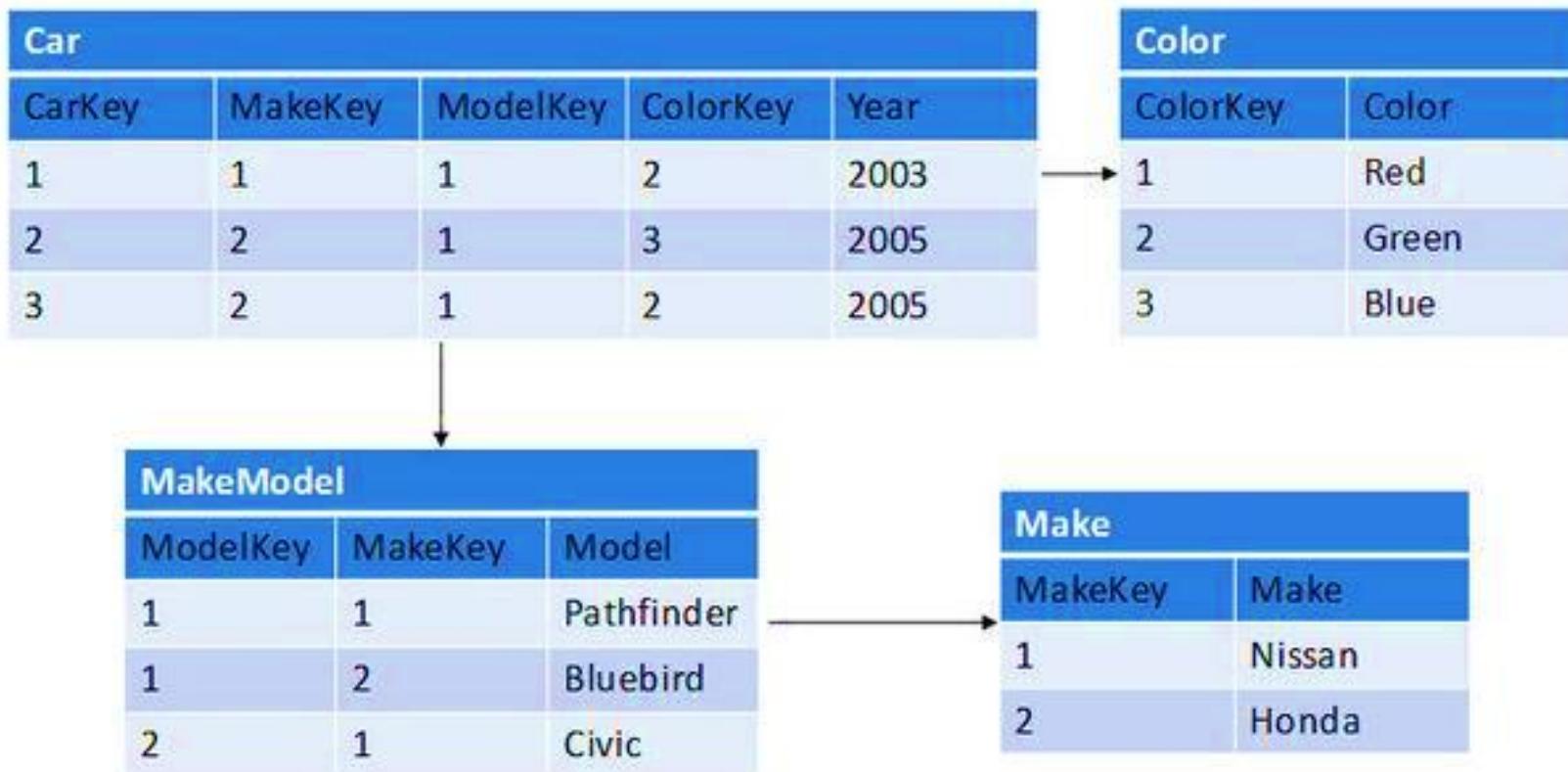
- داده‌ها به صورت رابطه‌هایی (جدول‌هایی) مدل می‌شود.
- اشیاء سطرهای جدول هستند.
- جدول شامل اشیاء مشابهی است.
- ارتباط بین جداول با استفاده از کلیدهای خارجی است.



# یادآوری: ویژگی‌های پایگاه‌های داده‌ای رابطه‌ای

بخش بیانیه: پایگاه داده‌ای NoSQL

۱۰





# یادآوری: ویژگی‌های پایگاه‌های داده‌ای رابطه‌ای

بخش بیانیه: پایگاه داده‌ای NoSQL

۱۱

- مدل داده‌ای رابطه‌ای
- شمای ثابت (مناسب برای داده‌های ساختیافته)
- مناسب برای داده با حجم نه چندان زیاد (مقیاس‌پذیر افقی)
- زمان اجرای غیرخطی برای پرس‌و‌جوابها
- تاکید بر فراهم کردن محدودیت‌ها (Constraint) و برخی قابلیت‌های اضافی
- مناسب برای کاربردهای نیازمند دقیق بالا و سازگاری داده‌ها (مانند برنامه‌های مالی و بانکی)



# یادآوری: خواص تراکنش در پایگاه داده‌های رابطه‌ای

فصل اول - مقدمه

۱۲

## □ فراهم کردن ACID

□ اتمیک بودن (Atomicity): یا کل تراکنش اجرا شود یا هیچ بخشی از آن اجرا نشود.

□ سازگاری (Consistency): هر تراکنش پایگاه داده را از یک حالت سازگار یا معتبر

(منطبق با محدودیتهای جامعیتی تعریف شده) به یک حالت سازگار یا معتبر دیگر ببرد.

□ جدا بودن (Isolation): اگر چند تراکنش به صورت همرونده اجرا شوند، حالت پایگاه داده

مشابه حالتی باشد که این تراکنشها به صورت سریال (پشت سر هم) اجرا شده باشند.

□ ماناگری (Durability): هر گاه که یک تراکنش خاتمه یافته اعلام شود، اثر آن بر روی پایگاه

داده مانا باقی بماند.



# مشخصات پایگاه‌های داده‌ی غیرابطه‌ای (NoSQL)

بخش بازدهم: پایگاه داده‌های NoSQL

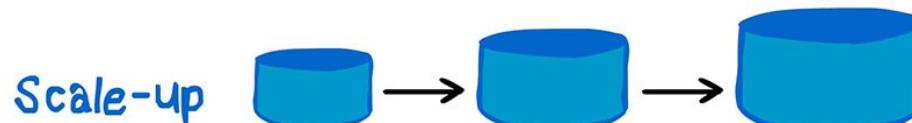
۱۳

□ فاقد شیمی اثبات (انعطاف‌پذیری) و برخی از عملگرهای زمانبر همچون پیوند (Join)

□ مقیاس‌پذیری افقی (Horizontal Scalability)

□ نیازمند افزایش گره‌های پردازشی و ذخیره‌سازی در صورت افزایش حجم داده در ساختاری توزیع شده

□ در مقابل **مقیاس‌پذیری عمودی** که نیازمند افزایش قدرت پردازش و حجم ذخیره‌سازی در گره‌های موجود، در صورت افزایش داده‌ها است.





# مشخصات پایگاه‌های داده‌ی غیرابطه‌ای (NoSQL) (ادامه)

## بخش بازدهم: پایگاه داده‌های NoSQL

۱۴

Original Table

CUSTOMER_ID	FIRST_NAME	LAST_NAME	FAVORITE_COLOR
1	TAEKO	OHNUKI	BLUE
2	O.V.	WRIGHT	GREEN
3	SELDAA	BAĞCAN	PURPLE
4	JIM	PEPPER	AUBERGINE

Vertical Partitions

CUSTOMER_ID	FIRST_NAME	LAST_NAME
1	TAEKO	OHNUKI
2	O.V.	WRIGHT
3	SELDAA	BAĞCAN
4	JIM	PEPPER

VP1

CUSTOMER_ID	FAVORITE_COLOR
1	BLUE
2	GREEN
3	PURPLE
4	AUBERGINE

VP2

Horizontal Partitions

CUSTOMER_ID	FIRST_NAME	LAST_NAME	FAVORITE_COLOR
1	TAEKO	OHNUKI	BLUE
2	O.V.	WRIGHT	GREEN

HP1

CUSTOMER_ID	FIRST_NAME	LAST_NAME	FAVORITE_COLOR
3	SELDAA	BAĞCAN	PURPLE
4	JIM	PEPPER	AUBERGINE

HP2

استفاده از تکنیک پارسازی (Sharding) یا

همان توزیع افقی برای توزیع بار

توزيع افقی: توزیع رکوردهای یک مجموعه

داده روی گره‌های مختلف

هدف اصلی: افزایش کارایی، دسترس‌پذیری و

مقیاس‌پذیری از طریق

توزيع رکوردهای داده‌ای روی پارها در

گره‌های مختلف (Sharding Records)

تکرار پارها روی گره‌های مختلف

(Replicating Shards)



## مشخصات پایگاه‌های داده‌ی غیرابطه‌ای (NoSQL) (ادامه)

بخش بازدهم: پایگاه داده‌های NoSQL

۱۵

□ دسترسی کارا به داده‌ها با توجه به حجم بسیار زیاد داده‌ها با استفاده از یکی از تکنیک‌های:

□ درهم‌سازی کلید با استفاده از یک تابع درهم‌ساز  $h(k)$ : تعیین محل شیء داده با

$$h(k)$$

□ بخش‌بندی بازه مقادیر کلید:  $i$  نمایانگر داده‌های با کلید  $k$  که

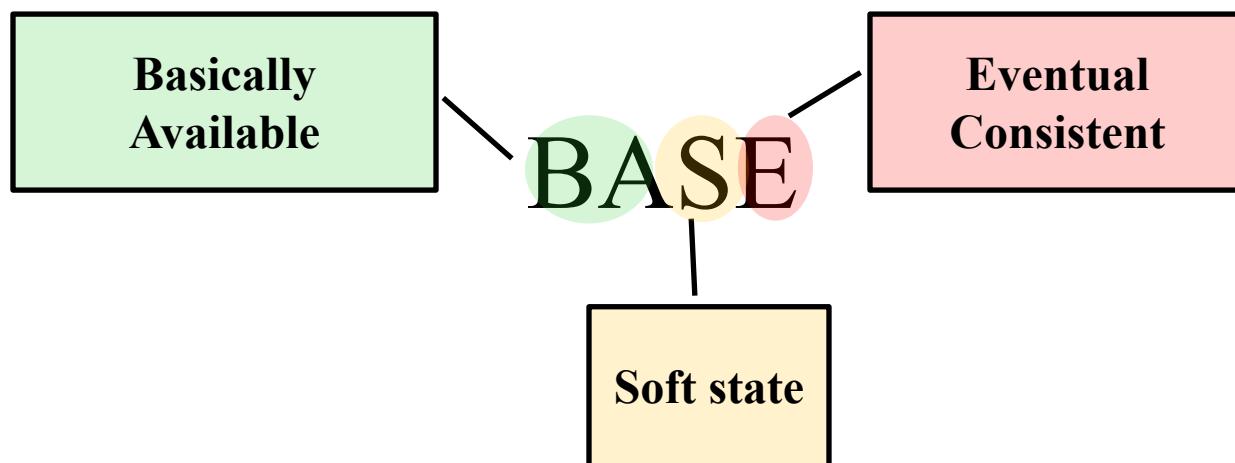


# مشخصات پایگاه‌های داده‌ی غیررابطه‌ای (NoSQL)

بخش بیانیه: پایگاه داده‌های NoSQL

۱۶

- دارای واسط (API) ساده برای ذخیره، بازیابی و بهروزرسانی داده
- فراهم‌سازی خواص **ACID** به جای **BASE**
- جایگزینی قویاً سازگار (Eventually Consistent) با نهایت‌اً سازگار (Strong Consistent)





### □ فراهم کردن BASE

□ **دسترسی اولیه (Basic Availability)**: داده‌ها در اغلب مواقع در دسترس هستند

ولی تضمینی برای دسترسی همیشگی به داده‌ها نیست.

□ **حالت نرم (Soft-State)**: حالت سیستم در طی زمان (حتی با عدم وجود ورودی)

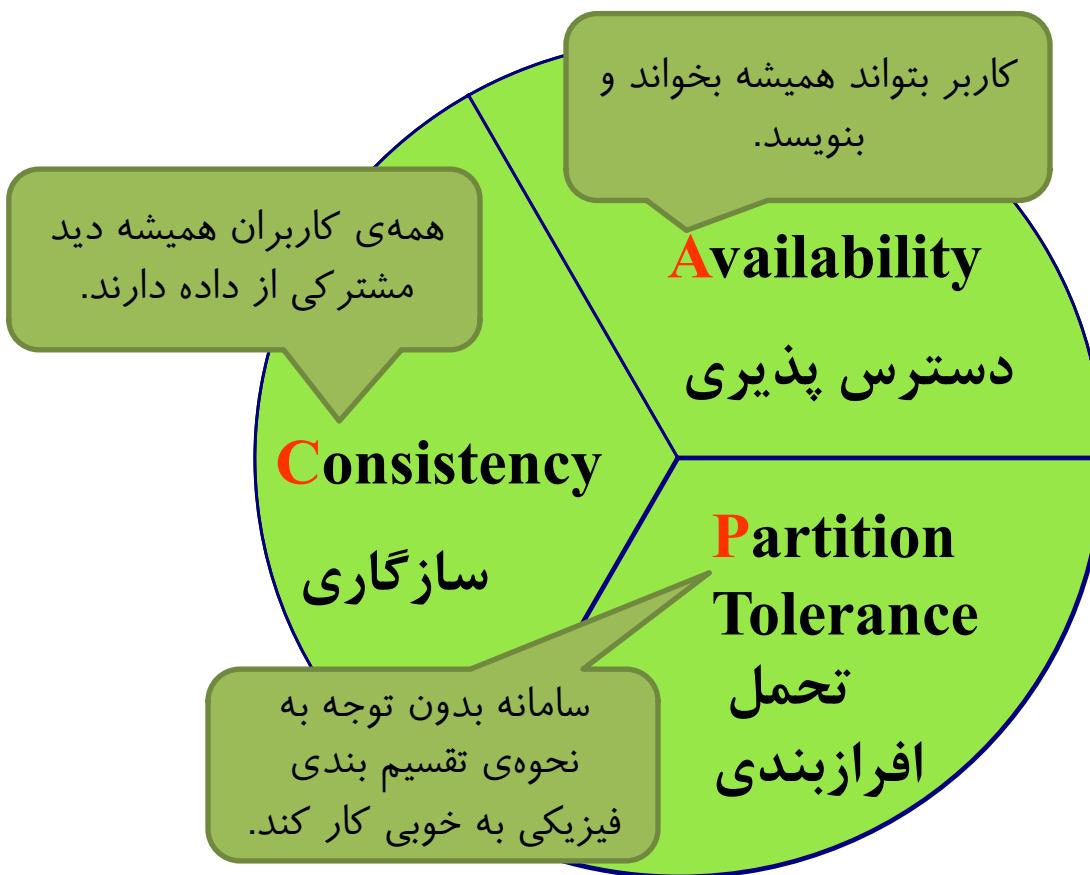
به دلیل سازگارسازی مقادیر توزیع یا تکرار شده، ممکن است تغییر نماید.

□ **نهایتاً سازگار (Eventual Consistency)**: حالت سیستم نهایتاً سازگار خواهد

شد. به عبارتی دیگر سازگاری سیستم در انتهای هر تراکنش چک نمی‌شود چرا که

مقادیر اقلام داده‌ای دیر یا زود منتشر و نسخه‌های مختلف نهایتاً سازگار می‌شوند.

**□ تئوری CAP:** هر پایگاه داده‌ی توزیع شده می‌تواند دو ویژگی از سه ویژگی زیر را دارا باشد.



**AP □**

- Voldemort (Key-value) ▪
- CouchDB (Document) ▪
- Riak (Document) ▪

**CA □**

- Relational databases ▪
- Vertica (column-oriented) ▪
- GreenPlum (Relational) ▪

**CP □**

- BigTable (Column Oriented) ▪
- MongoDB (Document) ▪



□ پیداپیش پایگاه‌های داده‌ی NoSQL

□ تفاوت پایگاه‌های داده‌ی رابطه‌ای و غیررابطه‌ای (NoSQL)

□ انواع پایگاه‌داده‌های NoSQL

□ کلید-مقدار (Key-Value Store)

□ سندگرا (Document Based)

□ ستونی (Column Based)

□ مبتنی بر گراف (Graph Based)

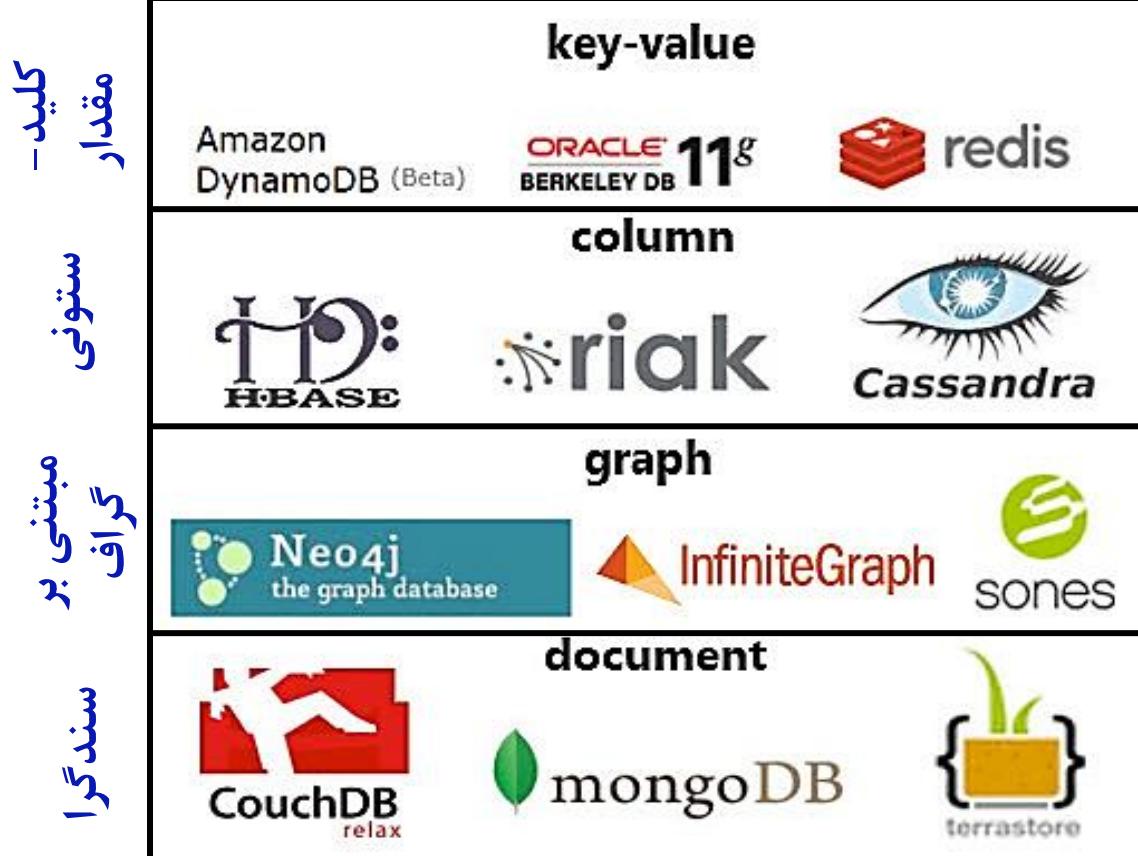


# انواع مدل‌های داده‌ای NoSQL

بخش بیانیه: پایگاه داده‌های NoSQL

۲۰

در حال حاضر بیش از ۱۵۰ نوع پایگاه داده‌ای NoSQL وجود دارد.

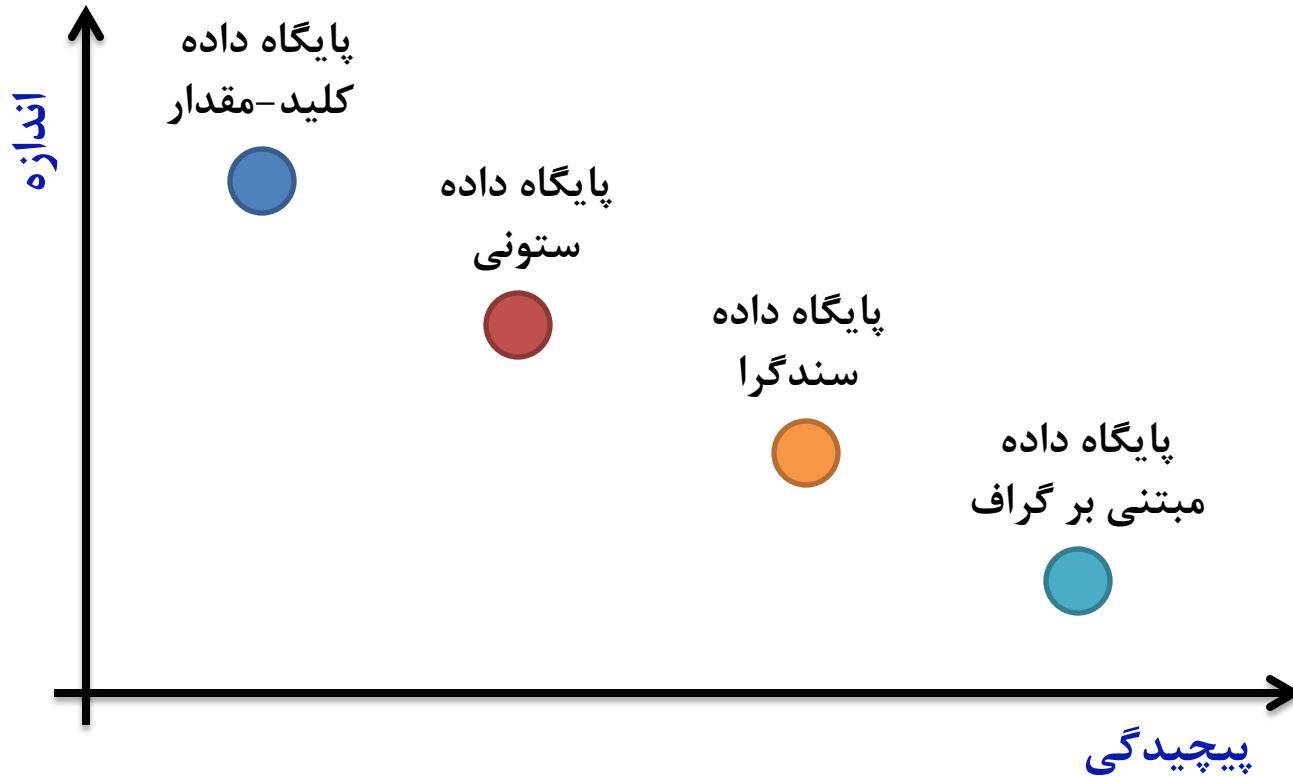




# انواع پایگاه داده‌های NoSQL

بخش بیازدهم: پایگاه داده‌های NoSQL

۲۱



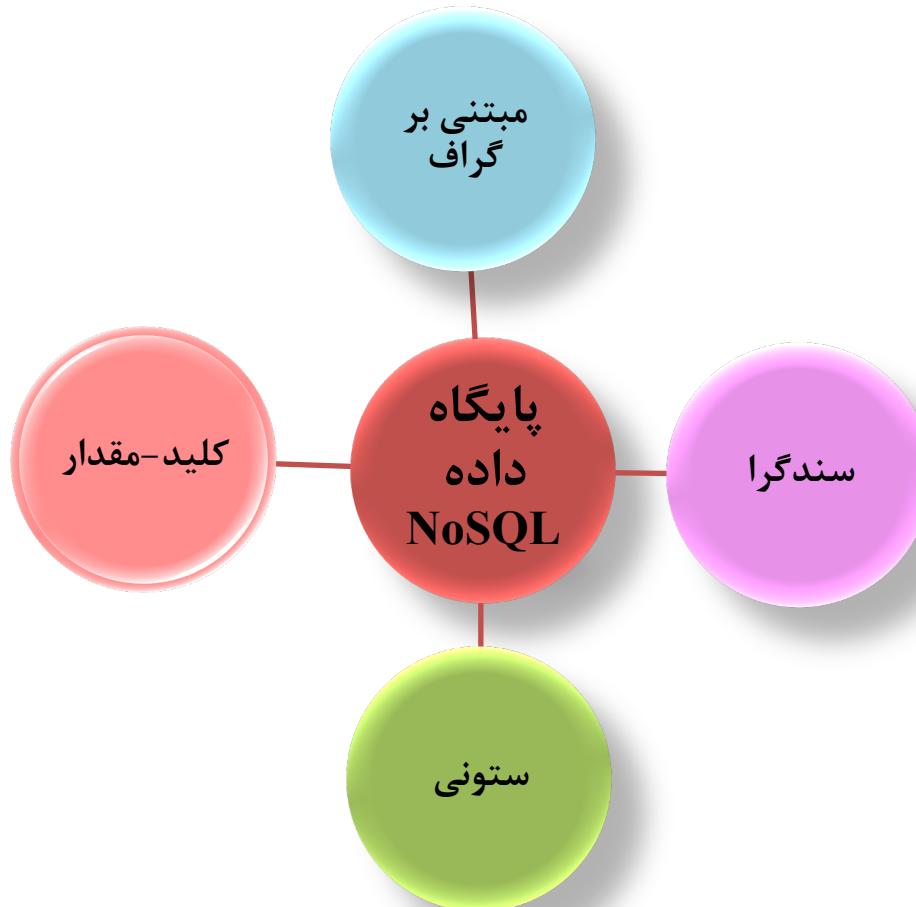


# شرکت‌های استفاده کننده از NoSQL

## بخش یازدهم: پایگاه داده‌های NoSQL

۲۲

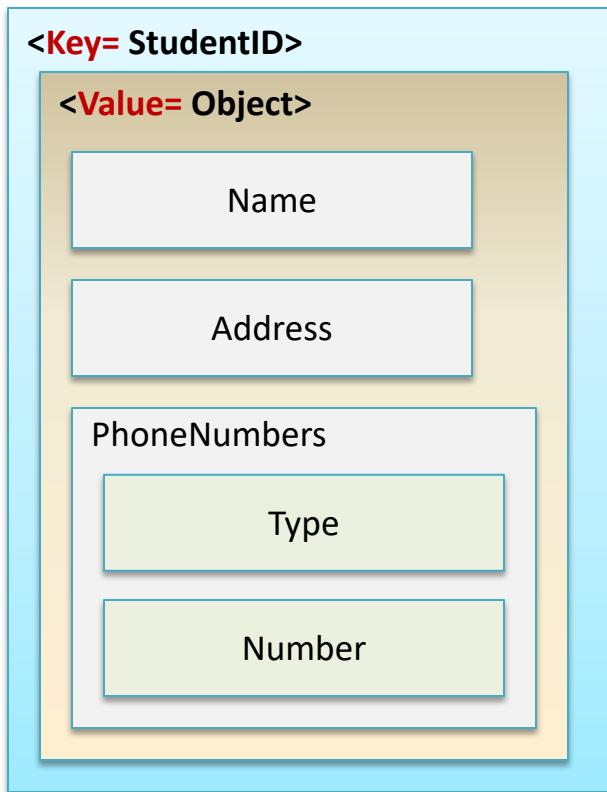
Company Name	NoSQL Name	NoSQL Storage Type
Adobe	HBase	Column
Amazon	Dynamo   SimpleDB	Key-Value   Document
BestBuy	Riak	Key-Value
eBay	Cassandra   MongoDB	Column   Document
Facebook	Cassandra   Neo4j	Column   Graph
Google	BigTable	Column
LinkedIn	Voldemort	Key-Value
LotsOfWords	CouchDB	Document
MongoHQ	MongoDB	Document
Mozilla	HBase   Riak	Column   Key-Value
Netflix	SimpleDB   HBase   Cassandra	Document   Column   Column
Twitter	Cassandra	Column





ذخیره‌سازی مجموعه‌ای از عناصر کلید-مقدار به صورت  $(k, v)$  که در آن  $k$  یک کلید و  $v$  یک مقدار (یک رشته از بایتها) است.

مقدار  $v$  می‌تواند ساختاری خود-توصیف داشته باشد (مانند JSON یا XML). در این صورت باید ساختار داده را به صورت رشته‌ای از بایتها **سریال‌سازی** و ذخیره نمود.



**عملگرها:**

بازیابی  $v := \text{get}(k)$

درج  $\text{put}(k, v)$

حذف  $\text{delete}(k)$

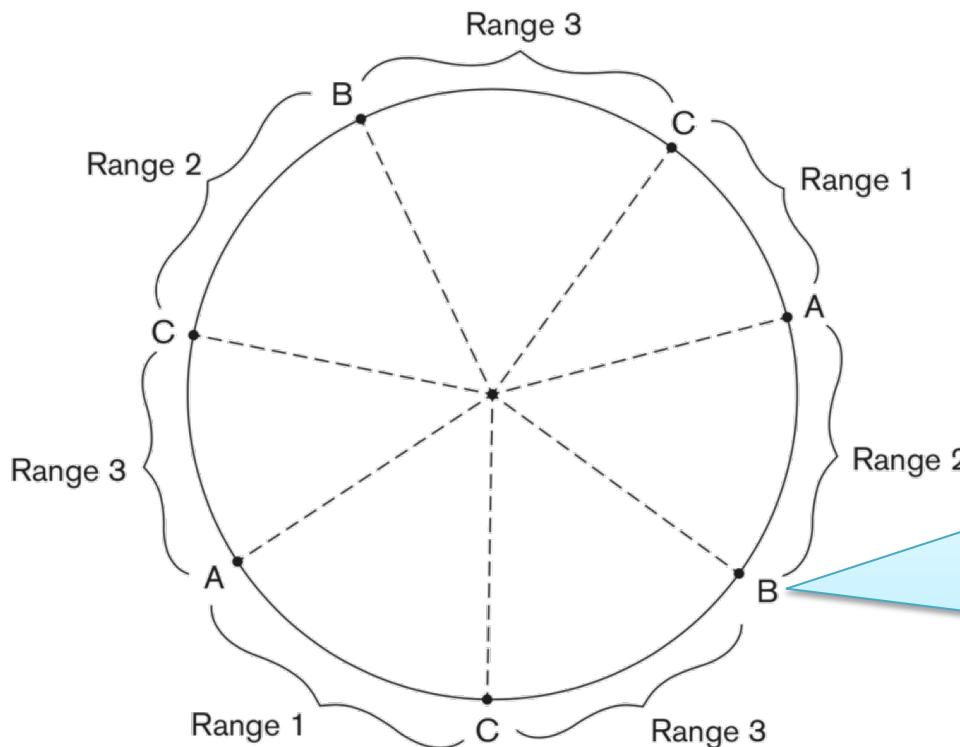


# توزیع و تکرار داده‌ها در پایگاه داده کلید-مقدار

## بخش بیانیه: پایگاه داده‌های NoSQL

۲۵

- ❑ توزیع داده‌ها در پایگاه داده کلید-مقدار با استفاده از  $h(k)$  (مقدار درهمسازی شده کلید) انجام می‌شود.
- ❑ بسته به ظرفیت گره‌ها، چند نقطه به صورت شبه تصادفی روی حلقه برای گره‌ها در نظر گرفته می‌شود.
- ❑ بسته به اینکه  $h(k)$  در چه بازه‌ای بیفت، گره مربوطه بر روی حلقه (در جهت عقربه‌های ساعت) برای ذخیره‌سازی  $(k, v)$  استفاده می‌شود.



❑ در صورت نیاز به **تکرار داده**، کلید-مقدار مورد نظر بر روی گره مربوط به خود و چند گره متوالی بعدی بر روی حلقه (در جهت عقربه‌های ساعت) ذخیره می‌شود.

کلید-مقدارهایی که  $h(k)$  آنها نقطه‌ای در بازه Range 2 در حلقه باشد، بر روی گره B ذخیره می‌شوند. تکرار این داده روی گره‌های C و A می‌تواند ذخیره شود.



# مقیاس پذیری افقی در پایگاه داده کلید-مقدار

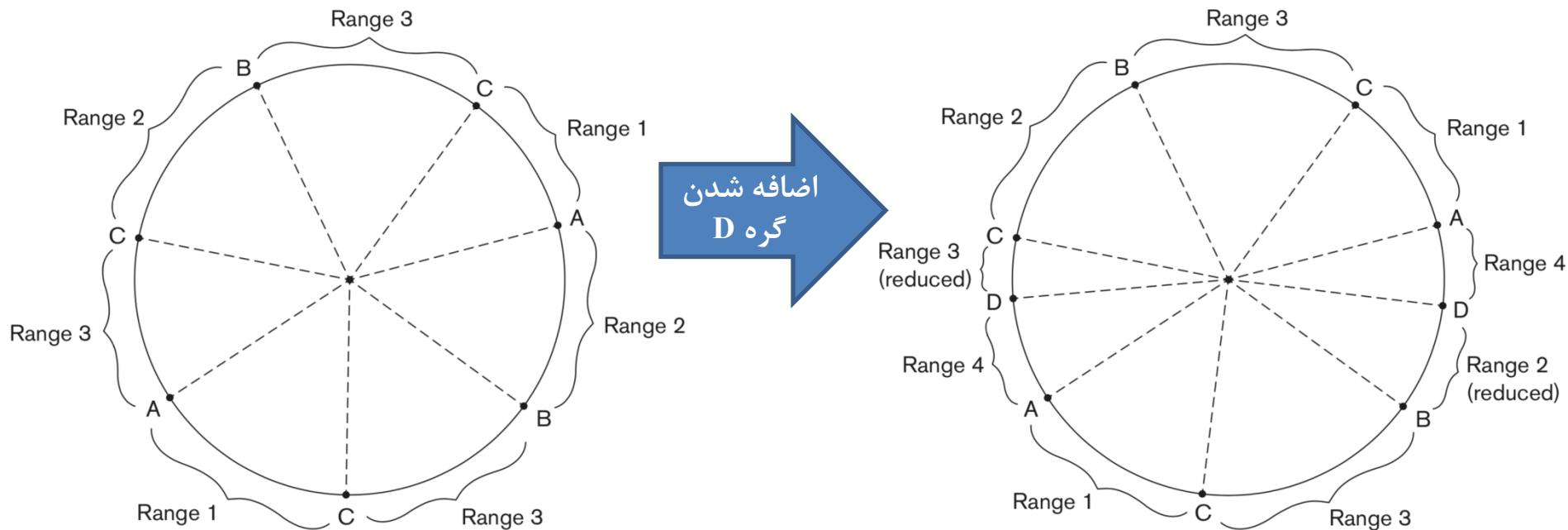
## بخش یازدهم: پایگاه داده‌های NoSQL

۲۶

با اضافه شدن یک گره جدید (مانند D) چند نقطه روی حلقه (بسته به ظرفیت گره جدید) اضافه می‌شود.

□ بازه‌هایی از مقادیر درهم‌سازی شده کلیدها به گره جدید منتب می‌شود.

□ بخشی از کلید-مقدارها از گره‌های موجود به گره جدید منتقل می‌شوند.





# ویژگیهای پایگاه داده‌های کلید-مقدار

بخش بیانیه: پایگاه داده‌های NoSQL

۲۷

مزایا

سرعت بالا

بسیار مقیاس‌پذیر

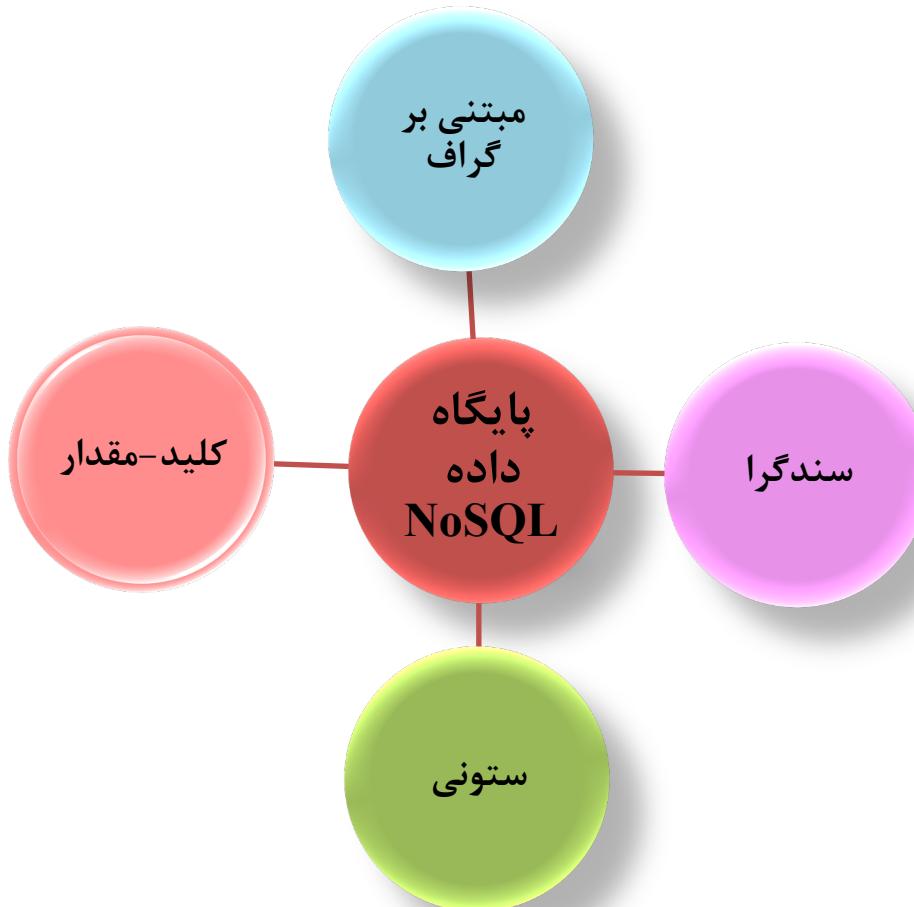
مدل ساده

توزیع پذیر افقی

معایب

بسیاری از ساختارهای داده را نمی‌توان با زوج کلید-مقدار مدل کرد.

برای پرس‌وجوهای پیچیده و حاوی عملگر تجمعی، مناسب نیست.





- داده به شکل سند خود-توصیف‌گر (self-descriptive) ذخیره می‌شود.
- به شکل استاندارد XML، JSON و ...
- فاقد شیمای از پیش‌تعریف شده است.
- انعطاف‌پذیر
- کاهش فضای ذخیره‌سازی
- ذخیره قسمت‌های غیرتنهی و مهم



```
{  
  "firstName": "Ali",  
  "lastName": "Mohseni",  
  "address": {  
    "street": "21 2nd street",  
    "city": "tehran"  
  },  
  "phoneNumbers": [  
    {  
      "type": "home",  
      "number": "02177665544"  
    },  
    {  
      "type": "mobile",  
      "number": "091234567890"  
    }  
  ]  
}
```

JSON

```
<?xml version="1.0"?>  
<student>  
  <firstName> Ali </firstName>  
  <lastName> Mohseni </lastName>  
  <address>  
    <street> 21 2nd street </street>  
    <city> tehran </city>  
  </address>  
  <phoneNumbers>  
    <phone>  
      <type> home </type>  
      <number> 02177665544 </number>  
    </phone>  
    <phone>  
      <type> mobile </type>  
      <number> 91234567890 </number>  
    </phone>  
  </phoneNumbers>  
</student>
```

XML



- سند (document): داده خودتوصیف‌گر
- مجموعه (collection): مجموعه‌ای از اسناد مشابه
- ممکن است دارای شماهای مختلفی باشند.
- اسناد به عنوان مقدار در پایگاه داده‌های کلید-مقدار می‌توانند ذخیره شوند.
- به شکلی که مقدار آن ساختار مشخصی دارد و بر روی آن می‌توان جستجو کرد.



# پایگاه داده سندگرا – MongoDB

بخش یازدهم: پایگاه داده‌های NoSQL

۳۲

مستندات در قالب BSON (Binary JSON) ذخیره می‌شوند.

BSON دارای نوع داده‌های بیشتر و کارایی بهتر نسبت به JSON است.

در هر مجموعه (collection)، تعدادی سند (document) ذخیره می‌شود.

هر سند دارای شناسه منحصر به فردی (id) به عنوان کلید است که به صورت خودکار بر روی آن

شاخص درهم‌سازی (مشابه آنچه در پایگاه داده کلید-مقدار بیان شد) ایجاد می‌شود.

در صورت عدم درج شناسه برای یک سند، سیستم به صورت خودکار رشته‌ای را به عنوان کلید برای

آن تولید و ثبت می‌کند.



# پایگاه داده سندگرا – عملیات CRUD در MongoDB

بخش بیازدهم: پایگاه داده‌های NoSQL

۳۳

## □ ایجاد پایگاه داده و ایجاد یک مجموعه

```
use myNewDB
```

```
db.createCollection(<name>: {options})
```



```
db.createCollection("project", { capped : true, size : 1310720, max : 500 } )
```

```
db.createCollection("worker", { capped : true, size : 5242880, max : 2000 } )
```

□ اگر پایگاه داده myNewDB وجود نداشته باشد، ابتدا پایگاه داده را ایجاد می‌نماید و سپس مجموعه‌ها را ایجاد می‌نماید.

□ با پارامتر capped می‌توان بر روی اندازه (size) مجموعه و حداقل تعداد اسناد (max) در مجموعه، محدودیت تعریف کرد.



# پایگاه داده سندگرا – عملیات CRUD در MongoDB

بخش بیازدهم: پایگاه داده‌های NoSQL

۳۴

## □ درج سند (تک یا گروهی) در مجموعه

**db.collection.insertOne( document, {writeConcern: optionsDocument} )**

**db.collection.insertMany( [document1, document2, ...], {writeConcern: optionsDocument} )**

**db.project.insertOne( {\_id: "P1", Pname: "ProductX", Plocation: "Bellaire"} )**

**db.worker.insertMany( [{\_id: "W1", Ename: "John Smith", Ehours: "32.5"},  
                          {\_id: "W1", Ename: "Joyce English", Ehours: "20.0"}] )**

□ در این دو دستور در صورت عدم وجود مجموعه، ابتدا مجموعه ایجاد و سپس سند(ها) در آن درج می‌شوند. لذا نیازی به تعریف صریح یک مجموعه با دستور `createCollection` نیست مگر آنکه طراح بخواهد پارامترهای مربوط به مجموعه را به دلخواه تنظیم نماید.





## پایگاه داده سندگرا – طراحی در پایگاه داده سندگرا

بخش بیازدهم: پایگاه داده‌های NoSQL

۳۵

### □ طراحی سند غیرنرمال با درج زیرسندها در سند

```
{  
    _id: "P1",  
    Pname: "ProductX",  
    Plocation: "Bellaire",  
    Workers: [  
        { Ename: "John Smith",  
         Hours: 32.5  
        },  
        { Ename: "Joyce English",  
         Hours: 20.0  
        }  
    ]  
};
```



## پایگاه داده سندگرا – طراحی در پایگاه داده سندگرا

بخش بیازدهم: پایگاه داده‌های NoSQL

۳۶

□ طراحی سند غیرنرمال با آرایه‌ای از ارجاعات به اسناد

```
{  
    _id: "P1",  
    Pname: "ProductX",  
    Plocation: "Bellaire",  
    WorkerIds: [ "W1", "W2" ]  
}  
{  
    _id: "W1",  
    Ename: "John Smith",  
    Hours: 32.5  
}  
{  
    _id: "W2",  
    Ename: "Joyce English",  
    Hours: 20.0  
}
```



## پایگاه داده سندگرا – طراحی در پایگاه داده سندگرا

بخش یازدهم: پایگاه داده‌های NoSQL

۳۷

### □ طراحی اسناد نرم‌مال

```
{  
    _id: "P1",  
    Pname: "ProductX",  
    Plocation: "Bellaire"  
}  
  
{  
    _id: "W1",  
    Ename: "John Smith",  
    ProjectId: "P1",  
    Hours: 32.5  
}  
  
{  
    _id: "W2",  
    Ename: "Joyce English",  
    ProjectId: "P1",  
    Hours: 20.0  
}
```



# پایگاه داده سندگرا – عملیات CRUD در MongoDB

بخش بازدهم: پایگاه داده‌های NoSQL

۳۸

## □ بازیابی سند(ها) از مجموعه

`db.collection.find( query, projection )`

query فیلتر یا شرط بازیابی است. در صورت عدم استفاده، کل اسناد درون مجموعه بازیابی می‌شوند.

projection پرتوگیری روی اسناد منطبق با شرط را توصیف می‌کند. در صورت عدم استفاده، کل فیلدها برگردانده می‌شوند. 0 بیانگر عدم نمایش صفت و 1 بیانگر نمایش آن است. در عبارت پرتو نمی‌توان هم از 0 استفاده کرد و هم از 1 مگراینکه بخواهیم `id` را در خروجی نداشته باشیم.

اطلاعات پروژه‌های P1 و P2



`db.project.find( { _id: { $in: ["P1", "P2"] } } )`

نام (و نه شناسه) کارکنانی که بیشتر از ۲۰ و کمتر از ۳۰ ساعت در پروژه‌ها کار کرده‌اند.

`db.worker.find( { Ehours: { $gt: 20, $lt: 30 } }, { _id:0, Ename:1 } )`





# پایگاه داده سندگرا – عملیات CRUD در MongoDB

بخش بیازدهم: پایگاه داده‌های NoSQL

۳۹

## □ بازیابی سند(ها) از مجموعه

اطلاعات پروژه با نام ProductZ و در شهر Tehran



```
db.project.find( { Pname: "ProductZ", Pcity: "Tehran" } )
```

نام پروژه‌ای که کارکنانی با نام Davoud با ساعت کار بیشتر از ۲۰ در آنها کار می‌کنند.



```
db.worker.find( {Workers: { $elementMatch:  
    { Ename: "Davoud",  
      Ehours: {$gt: 20} }  
  }  
})
```



# پایگاه داده سندگرا – عملیات CRUD در MongoDB

بخش بیازدهم: پایگاه داده‌های NoSQL

۴۰

## مرتبسازی و محدودسازی

**db.collection.find(...).sort( {field1: asc-dsc, field2: asc-dsc, ...} )**

□ مرتبسازی اسناد خروجی بر اساس فیلدهای field1، field2 و ... انجام می‌شود. asc-dsc می‌تواند یکی از مقادیر 1 (به معنای صعودی) و -1 (به معنای نزولی) را داشته باشد.

**db.collection.find(...).limit(N)**

□ N سند اول حاوی شرط find را برمی‌گرداند.

**db.collection.find(...).skip(N)**

□ همه اسناد حاوی شرط find به جز N سند اول را برمی‌گردانند.

لیست پنج نفر از کارکنان با بالاترین حقوق، به ترتیب الفبا



**db.worker.find().sort({Esalary: -1, Ename: 1}).limit(5)**



# پایگاه داده سندگرا – عملیات CRUD در MongoDB

بخش بیازدهم: پایگاه داده‌های NoSQL

۴۱

## ❑ حذف سند (تک یا گروهی) از مجموعه

**db.collection.deleteOne( filter, {writeConcern: optionsDocument} )**

**db.collection.deleteMany( filter, {writeConcern: optionsDocument} )**

اولین سندی که در مجموعه project نام پروژه آن ProductX است، را حذف کن.



**db.project.deleteOne( {Pname: "ProductX"} )**

همه اسنادی که در مجموعه project نام آنها Unknown است، را حذف کن.



**db.project.deleteMany( {Pname: "Unknown"} )**



# پایگاه داده سندگرا – عملیات CRUD در MongoDB

بخش بیازدهم: پایگاه داده های NoSQL

۴۲

بروزرسانی و یا جایگزینی سند (تک یا گروهی) در مجموعه

**db.collection.updateOne( filter, update, options )**

**db.collection.updateMany(filter, update, options )**

**db.collection.replaceOne(filter, replacement, options )**

محل پروژهای که نام آن ProductX را به London تغییر بده.

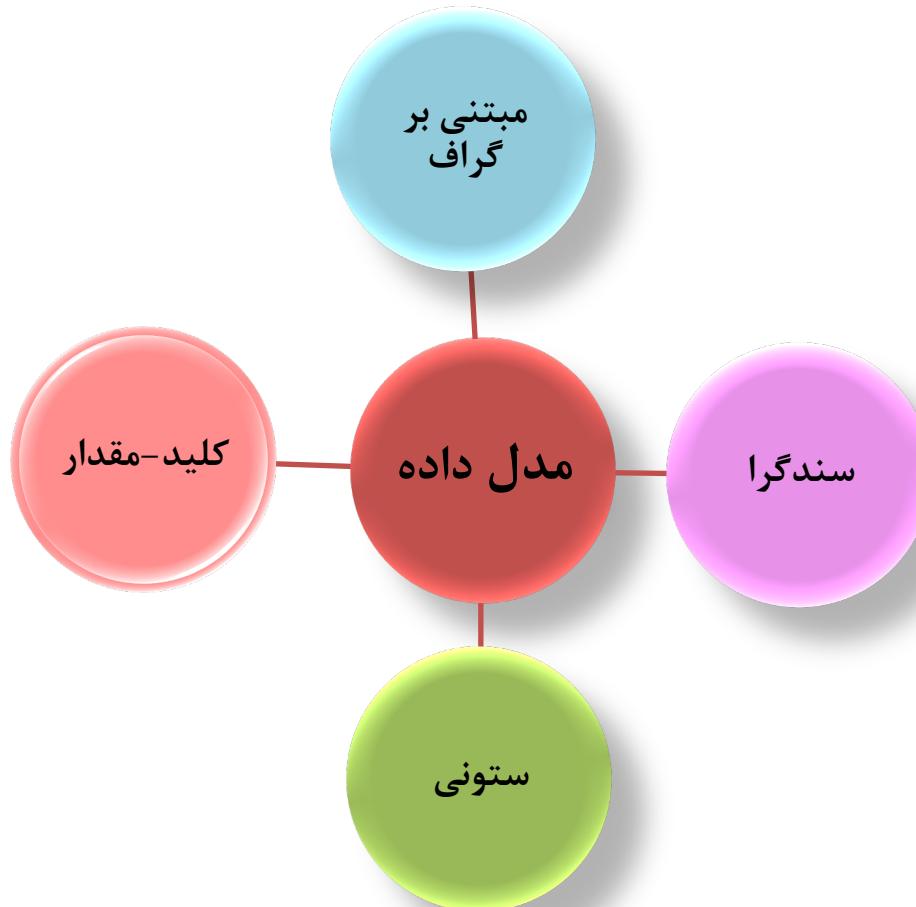


**db.project.updateOne( {Pname: "ProductX"}, { \$set: {Plocation: "London"} } )**

اطلاعات کارمند با شناسه W1 را با کارمندی با نام Jack Pit و تعداد ساعت ۴۲ جایگزین کن.

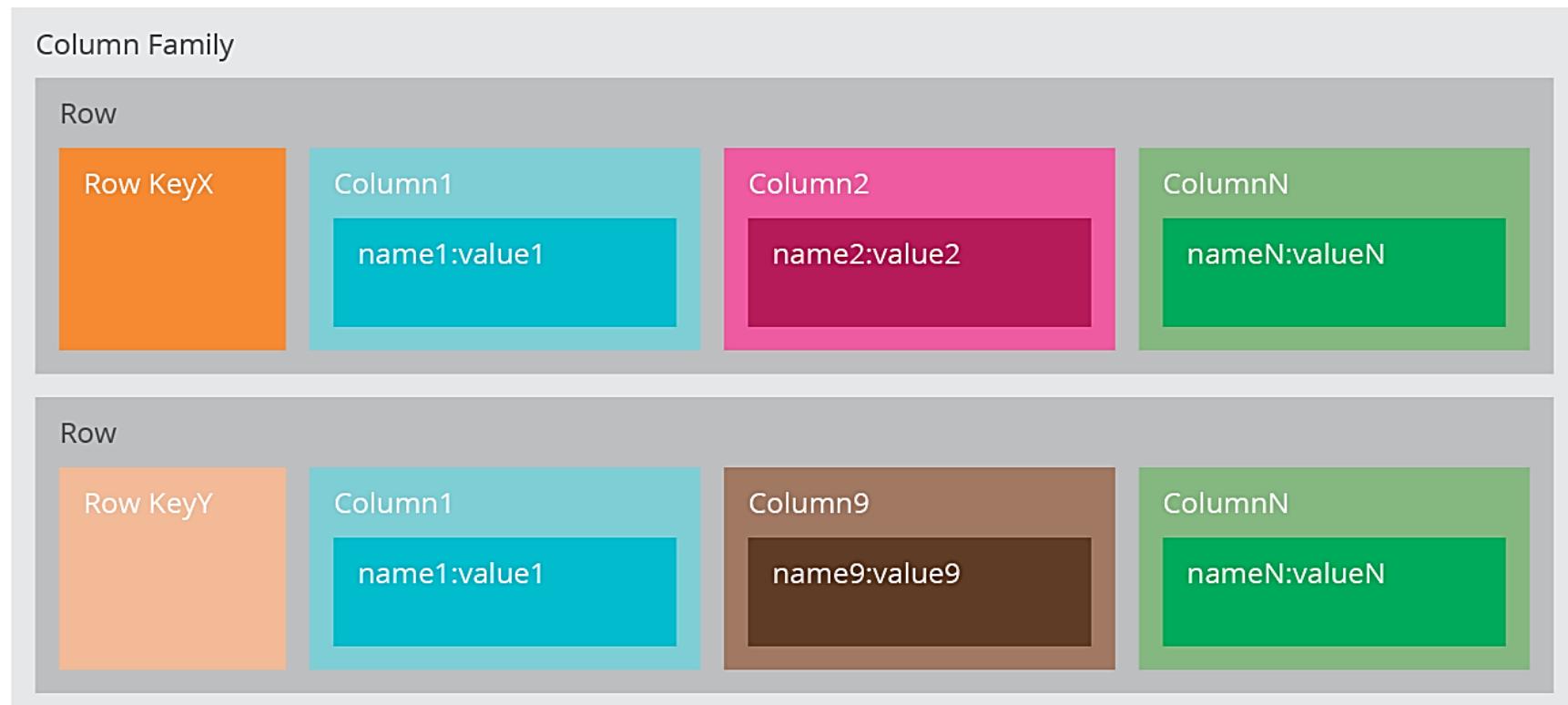


**db.worker.replaceOne( {\_id: "W1"},  
{\_id: "W1", Ename: "Jack Pit", Ehours: "42.0" } )**





□ مدل داده‌ای آن شامل تعدادی **جدول**، هر جدول شامل تعدادی **خانواده ستون** – **Column Family** (با تعريف و ساختار ثابت)، هر خانواده ستون دارای تعدادی **ستون** – **Column** (با ساختار متغير) است. هر جدول شامل تعدادی **سطر** – **Row** است که هر یک دارای کلید یکتایی است.





□ خانواده ستون (Column Family): گروهی از ستون‌های مرتبط که با هم مورد دسترسی قرار می‌گیرند و لذا در یک فایل ذخیره می‌شوند.

- هر جدول یک یا چند خانواده ستون دارد.
- خانواده ستون‌ها در هنگام تعریف جدول تعریف می‌شوند و ثابت و غیرقابل تغییر هستند.
- به هر خانواده ستون تعدادی ستون می‌تواند منتب شود که تعریف ثابت و مشخصی ندارند.
- بازیابی اطلاعات مندرج در خانواده ستون‌ها می‌تواند با کارایی بالایی انجام شود، لذا ساختار مناسبی برای انجام کارهای آماری و تحلیلی فراهم می‌نماید.

در یک سیستم فروشگاه اینترنتی می‌توان خانواده ستون‌های زیر را در نظر گرفت:



- اطلاعات پروفایل مشتری
- اطلاعات سفارشات

□ **ستون (Column):** پایه‌ای ترین عنصر این مدل که به یک خانواده ستون منتب می‌گردد.

□ ساختار از پیش‌تعریف شده‌ای ندارد و هنگام درج در خانواده ستون، ساختار آن نیز توصیف می‌شود.

□ شامل سه‌تایی **نام، مقدار و مهرزمانی** است. با وجود مهر زمانی، می‌توان **نسخه‌های مختلفی** از مقدار یک ستون را نگهداری کرد. تعداد نسخ قابل نگهداری را معمولاً می‌توان تنظیم کرد.

□ هر ستون از ترکیب نام جدول، کلید سطر، نام خانواده ستون و نام ستون مشخص می‌شود.

(**tableName.rowKey.columnFamily:columnName**)

□ در فرآیند بازیابی، اخیرترین نسخه‌های مقدار ستون (مقادیر با بالاترین مهرزمانی) بازیابی می‌شوند. تعداد نسخه‌های اخیری که بازیابی می‌شوند معمولاً قابل تنظیم است.

person.info

```
{ name: "firstName", value: "Ali",
  timestamp: 12345678998 }
```

ستون توصیف کننده «نام» با مقدار «علی»

در جدول person و خانواده ستون info

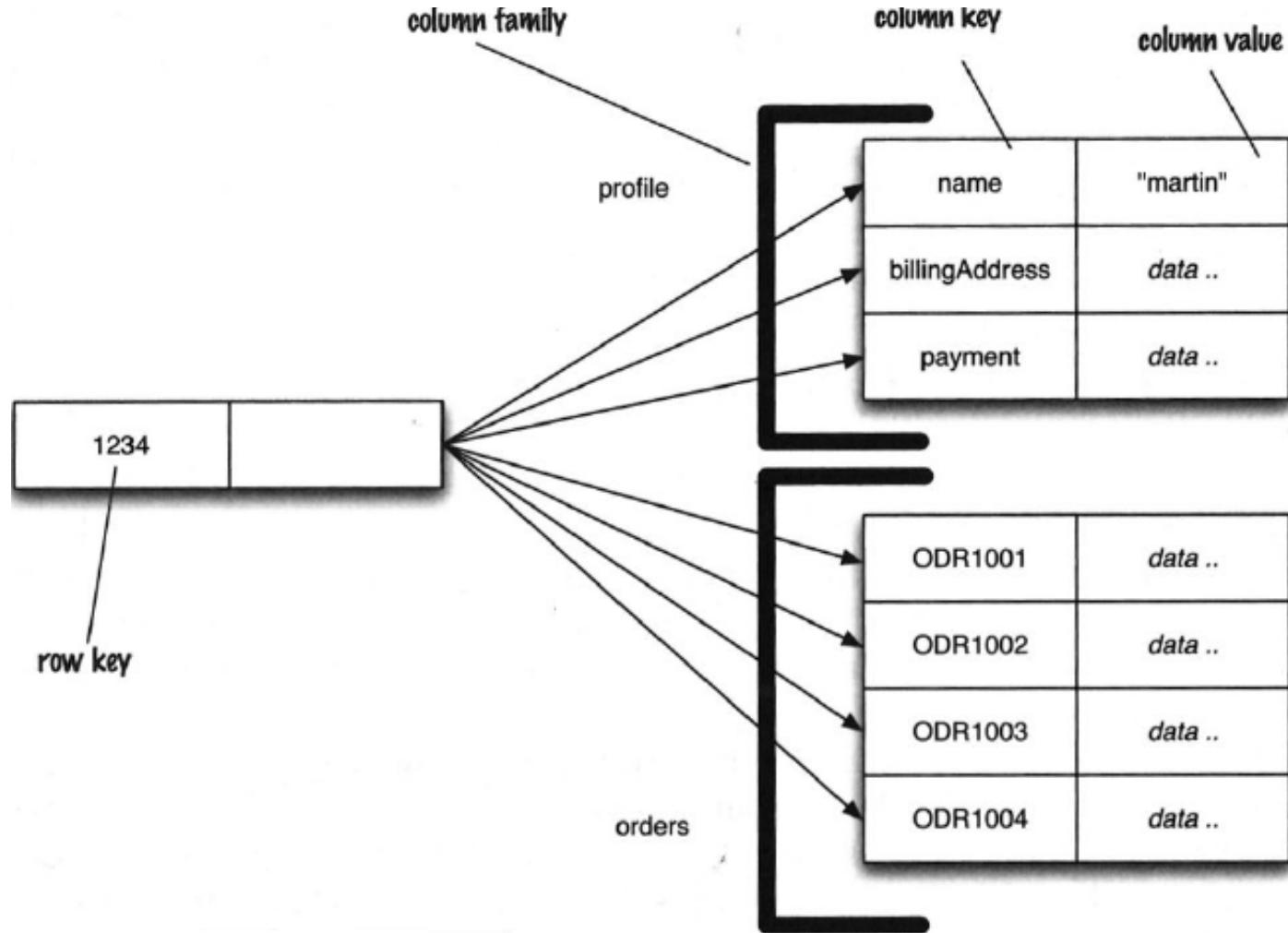




□ **سطر (Row):** داده‌ها در جدول‌ها در قالب تعدادی سطر خود توصیف-کننده ذخیره می‌شوند.

□ هر سطر مجموعه‌ای از ستون‌ها است که دارای **کلید سطر (Row Key)** مشترکی هستند.

		Column Family 1			Column Family 2	
Row Key 1		Column Name 1	Column Name 2	Column Name 3	...	
		Column Value 1	Column Value 2	Column Value 3	...	
Row Key 2		Column Name 4	Column Name 5	...	...	
		Column Value 6	Column Value 6	...	...	
⋮		⋮	⋮	⋮	⋮	⋮





## پایگاه داده ستونی – عملیات CRUD در Hbase

بخش بیازدهم: پایگاه داده های NoSQL

۴۹

□ ایجاد جدول: با معرفی نام جدول و نام خانواده ستون های آن که ثابت و غیرقابل تغییر است.

**create** tableName, columnFamily1, columnFamily2, ...

□ درج داده: ثبت نام و مقدار ستون به همراه مهر زمانی در یک سطر

**put** tableName, rowKey, columnFamily:columnName, value, timeStamp

□ بازیابی داده یک سطر از یک جدول با استفاده از کلید سطر

**get** tableName, rowKey

□ بازیابی کل داده های یک جدول: بازیابی کل سطرهای یک جدول

**scan** tableName



# پایگاه داده ستونی – عملیات CRUD در Hbase

بخش بیازدهم: پایگاه داده‌های NoSQL

۵۰



```
create 'EMP', 'Name', 'Address', 'Details'  
put 'EMP', 'row1', 'Name:Fname', 'Javad'  
put 'EMP', 'row1', 'Name:Lname', 'Amiri'  
put 'EMP', 'row1', 'Name:Nickname', 'JA'  
put 'EMP', 'row1', 'Details:Job', 'Computer Engineer'  
put 'EMP', 'row1', 'Details:Review', 'Good'  
  
-----  
put 'EMP', 'row2', 'Name:Fname', 'Pooneh'  
put 'EMP', 'row2', 'Name:Lname', 'Moradi'  
put 'EMP', 'row2', 'Details:Job', 'DBA'  
  
-----  
put 'EMP', 'row3', 'Name:Fname', 'Amir'  
put 'EMP', 'row3', 'Name:Mname', 'R.'  
put 'EMP', 'row3', 'Name:Lname', 'Zamani'  
put 'EMP', 'row3', 'Address:ZipCode', '87652-12763'  
put 'EMP', 'row3', 'Details:Job', 'CEO'  
put 'EMP', 'row3', 'Details:Salary', '10,000,000'
```



# پایگاه داده ستونی – مدیریت توزیع و ذخیره‌سازی در Hbase

بخش بیانیه: پایگاه داده‌های NoSQL

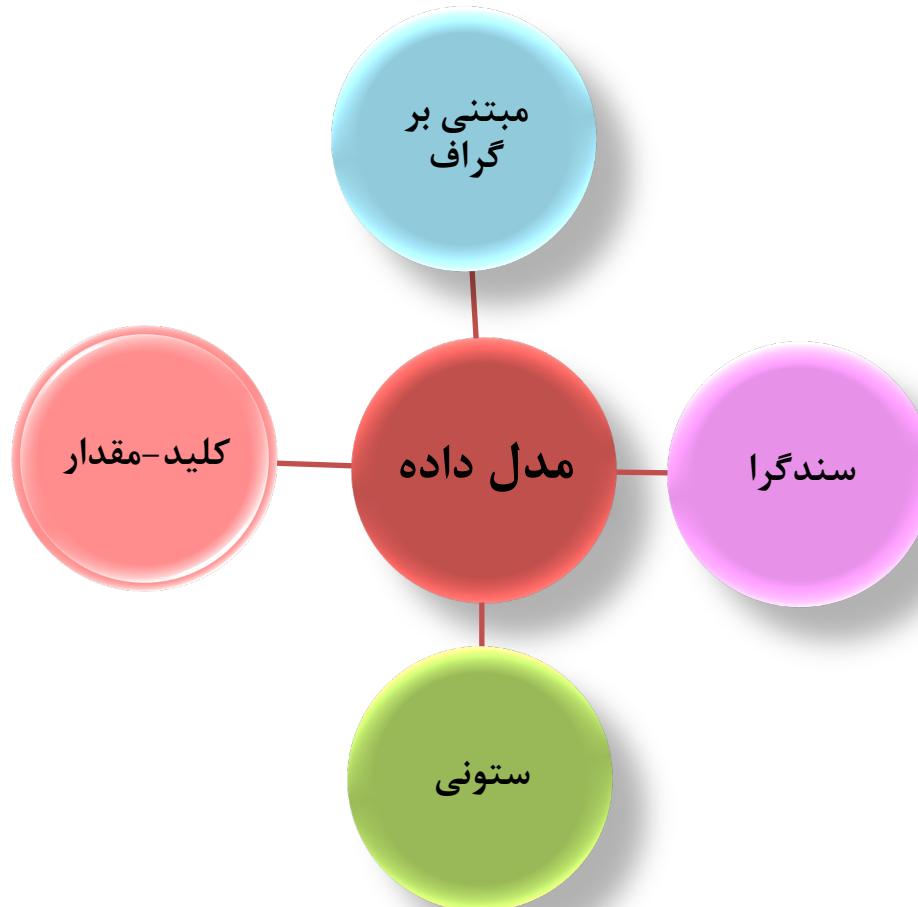
۵۱

- **ناحیه (Region):** هر جدول به چند ناحیه (Region) تقسیم می‌شود که در هر ناحیه سطرهای یک بازه از مقادیر کلید سطر ذخیره می‌شوند.
- **انباره (Store):** هر ناحیه شامل تعدادی انباره (Store) است. هر خانواده ستون به یکی از انباره‌ها نگاشت می‌شود.
- **کارگزار ناحیه (Region Server):** ناحیه‌ها برای ذخیره‌سازی داده‌ها به کارگزارهای ناحیه که همان گره‌های ذخیره‌سازی هستند، منتب می‌شوند.
- **یک کارگزار اصلی (Master Server):** مسئول پایش کارگزارن ناحیه، تقسیم جدول به تعدادی ناحیه و انتساب ناحیه‌ها به کارگزارن ناحیه است.
- Hbase از سیستم فایل توزیع شده هدوپ (HDFS) و سیستم Apache Zookeeper برای مدیریت توزیع، تکرار و همزمانی (synchronization) داده‌ها در کارگزارهای ناحیه استفاده می‌کند.

# پایگاه داده مبتنی بر گراف

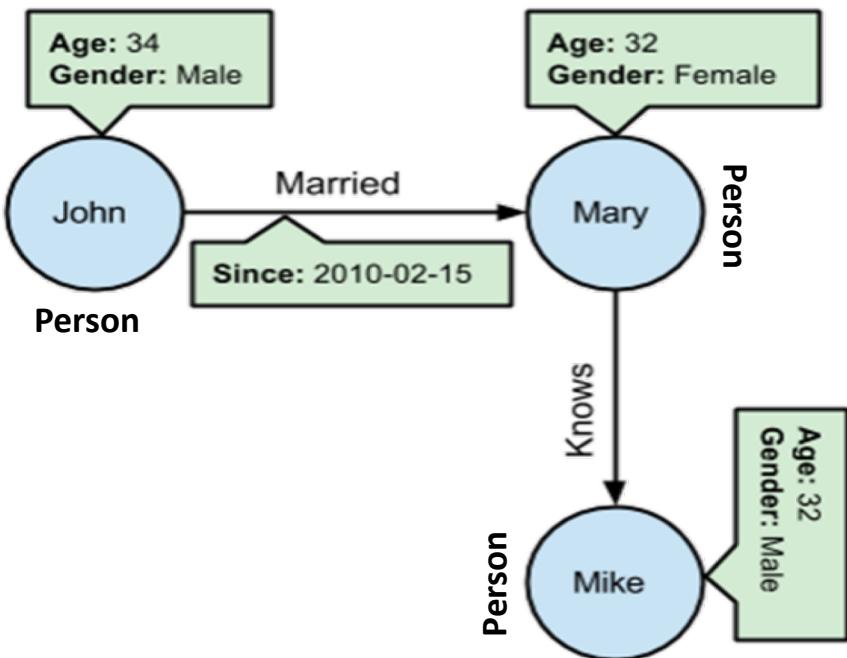
بخش یازدهم: پایگاه داده‌های NoSQL

۵۲





- داده‌ها به شکل گراف ذخیره می‌شوند.
- راس‌ها یا گره‌ها نمایانگر اشیاء و یال‌ها رابطه‌ی بین آن‌ها را مشخص می‌کنند.
- هدف اصلی این نوع پایگاه داده، بهینه‌سازی **ذخیره و بازیابی روابط بین موجودیت‌ها** است.



# مدل داده‌ای مبتنی بر گراف



- مدل داده‌ای مبتنی بر گراف شباهت زیادی با مدل داده‌ای ER/EER دارد با این تفاوت عمدۀ که این مدل داده برای پایگاه داده توزیع شده استفاده می‌شود ولی ER/EER برای طراحی پایگاه داده کاربرد دارد.
- تناظر بین مفاهیم ER/EER با مفاهیم پایگاه داده مبتنی بر گراف Neo4j

ER/EER مفاهیم	Neo4j مفاهیم
نمونه موجودیت	گره (Node)
نوع موجودیت	برچسب گره (Node Label)
نمونه ارتباط	ارتباط (Relationship)
نوع ارتباط	نوع ارتباط (Relationship Type)
صفت	خصوصیت (Property)

ارتباط دوطرفه

ارتباط یکطرفه



تعریف گرهای گراف شامل LOCATION، PROJECT، DEPARTMENT، EMPLOYEE به همراه خصوصیات آنها

```
CREATE (e1: EMPLOYEE, {Empid: '1', Lname: 'Smith', Fname: 'John', Minit: 'B'})  
CREATE (e2: EMPLOYEE, {Empid: '2', Lname: 'Wong', Fname: 'Franklin'})  
CREATE (e3: EMPLOYEE, {Empid: '3', Lname: 'Zelaya', Fname: 'Alicia'})  
CREATE (e4: EMPLOYEE, {Empid: '4', Lname: 'Wallace', Fname: 'Jennifer', Minit: 'S'})  
...  
CREATE (d1: DEPARTMENT, {Dno: '5', Dname: 'Research'})  
CREATE (d2: DEPARTMENT, {Dno: '4', Dname: 'Administration'})  
...  
CREATE (p1: PROJECT, {Pno: '1', Pname: 'ProductX'})  
CREATE (p2: PROJECT, {Pno: '2', Pname: 'ProductY'})  
CREATE (p3: PROJECT, {Pno: '10', Pname: 'Computerization'})  
CREATE (p4: PROJECT, {Pno: '20', Pname: 'Reorganization'})  
...  
CREATE (loc1: LOCATION, {Lname: 'Houston'})  
CREATE (loc2: LOCATION, {Lname: 'Stafford'})  
CREATE (loc3: LOCATION, {Lname: 'Bellaire'})
```





تعریف یالهای (ارتباطات) گراف شامل WorksOn, LocatedIn, Manager, WorksFor به همراه



خصوصیات آنها

CREATE (e1) - [ :WorksFor ] -> (d1)

CREATE (e3) - [ :WorksFor ] -> (d2)

...

CREATE (d1) - [ :Manager ] -> (e2)

CREATE (d2) - [ :Manager ] -> (e4)

...

CREATE (d1) - [ :LocatedIn ] -> (loc1)

CREATE (d1) - [ :LocatedIn ] -> (loc3)

CREATE (d1) - [ :LocatedIn ] -> (loc4)

CREATE (d2) - [ :LocatedIn ] -> (loc2)

...

CREATE (e1) - [ :WorksOn, {Hours: '32.5'} ] -> (p1)

CREATE (e1) - [ :WorksOn, {Hours: '7.5'} ] -> (p2)

CREATE (e2) - [ :WorksOn, {Hours: '10.0'} ] -> (p1)

CREATE (e2) - [ :WorksOn, {Hours: 10.0} ] -> (p2)

CREATE (e2) - [ :WorksOn, {Hours: '10.0'} ] -> (p3)

CREATE (e2) - [ :WorksOn, {Hours: 10.0} ] -> (p4)



### برخی عملگرهای ساده شده در زبان پرس‌وجوی Cypher



Finding nodes and relationships that match a pattern: MATCH <pattern>

Specifying aggregates and other query variables: WITH <specifications>

Specifying conditions on the data to be retrieved: WHERE <condition>

Specifying the data to be returned: RETURN <data>

Ordering the data to be returned: ORDER BY <data>

Limiting the number of returned data items: LIMIT <max number>

Creating nodes: CREATE <node, optional labels and properties>

Creating relationships: CREATE <relationship, relationship type and optional properties>

Deletion: DELETE <nodes or relationships>

Specifying property values and labels: SET <property values and labels>

Removing property values and labels: REMOVE <property values and labels>



# پایگاه داده مبتنی بر گراف Neo4j

۵۸

## بخش بیانی: پایگاه داده های NoSQL

1. MATCH (d : DEPARTMENT {Dno: '5'}) – [ : LocatedIn ] → (loc)  
RETURN d.Dname , loc.Lname
2. MATCH (e: EMPLOYEE {Empid: '2'}) – [ w: WorksOn ] → (p)  
RETURN e.Ename , w.Hours, p.Pname
3. MATCH (e ) – [ w: WorksOn ] → (p: PROJECT {Pno: 2})  
RETURN p.Pname, e.Ename , w.Hours
4. MATCH (e) – [ w: WorksOn ] → (p)  
RETURN e.Ename , w.Hours, p.Pname  
ORDER BY e.Ename
5. MATCH (e) – [ w: WorksOn ] → (p)  
RETURN e.Ename , w.Hours, p.Pname  
ORDER BY e.Ename  
LIMIT 10
6. MATCH (e) – [ w: WorksOn ] → (p)  
WITH e, COUNT(p) AS numOfprojs  
WHERE numOfprojs > 2  
RETURN e.Ename , numOfprojs  
ORDER BY numOfprojs
7. MATCH (e) – [ w: WorksOn ] → (p)  
RETURN e , w, p  
ORDER BY e.Ename  
LIMIT 10
8. MATCH (e: EMPLOYEE {Empid: '2'})  
SET e.Job = 'Engineer'

نمونه هایی از پرس و جوهای Cypher



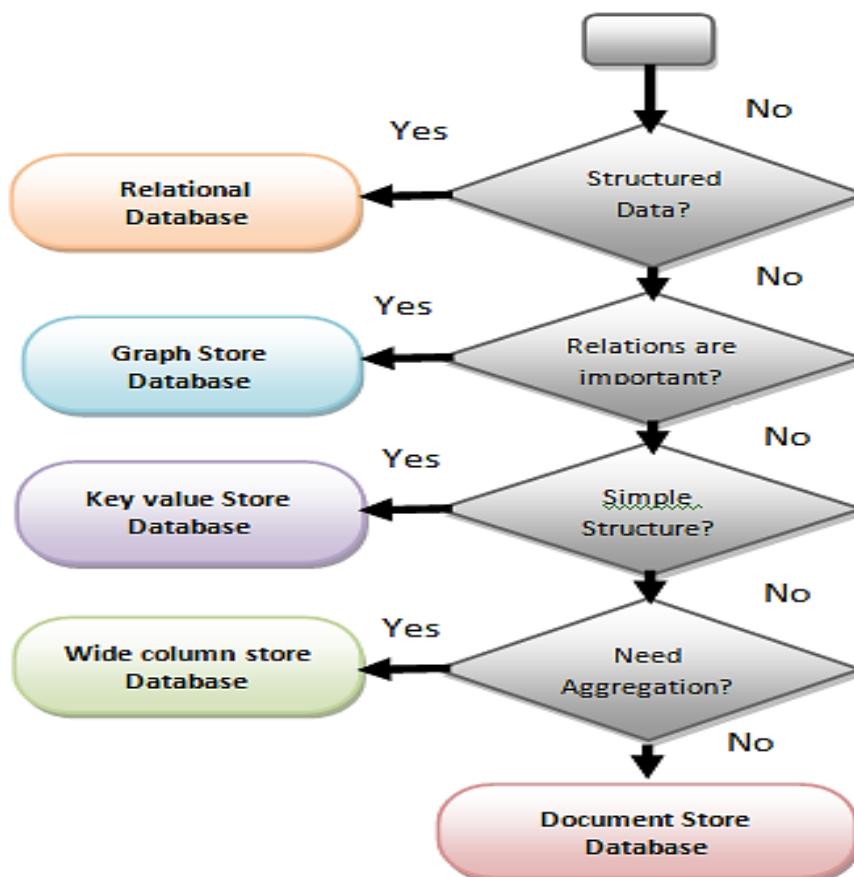


# انتخاب پایگاه داده غیررابطه‌ای

بخش یازدهم: پایگاه داده‌های NoSQL

۵۹

کدام نوع پایگاه داده غیررابطه‌ای مناسب است؟



□ بستگی به نیاز برنامه کاربرد

□ اندازه‌ی داده و پیچیدگی

□ تئوری CAP

□ ساختار داده



# پرسش و پاسخ . . .

[amini@sharif.edu](mailto:amini@sharif.edu)