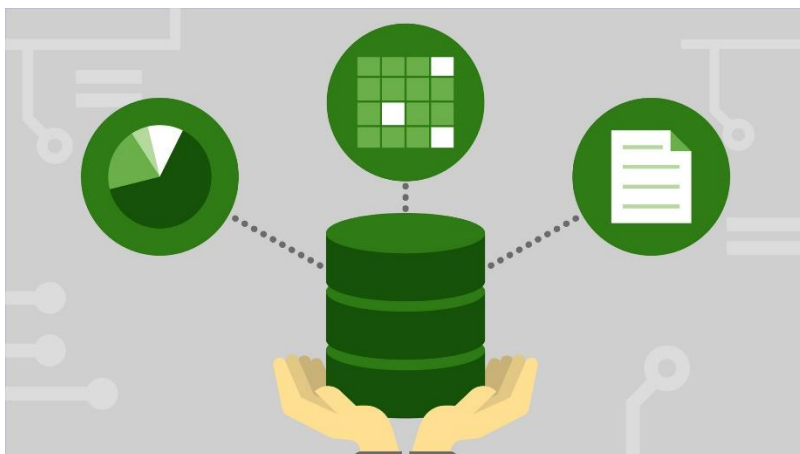


به نام خدا



دانشگاه تهران  
پردیس دانشکده‌های فنی  
دانشکده برق و کامپیوتر



## آزمایشگاه پایگاه داده

دستورکار شماره ۷

شماره دانشجویی

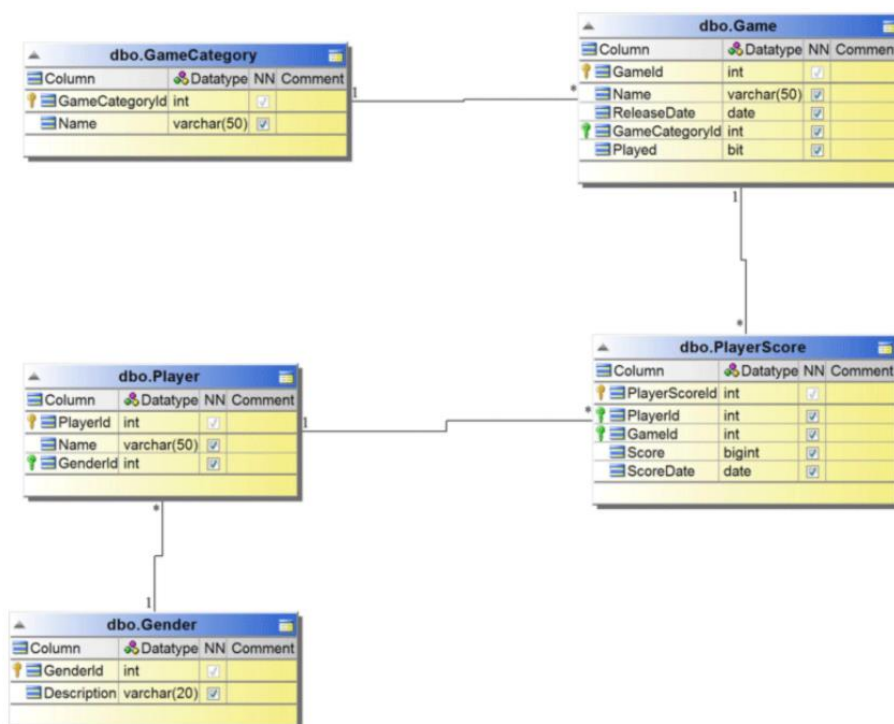
۸۱۰۱۹۶۶۱۵

دی ۹۹

محیا قینی

## گزارش فعالیت‌های انجام شده

در این آزمایشگاه به آشنایی و کار با پایگاه داده غیر رابطه ای مانگو پرداختیم. بدین جهت ابتدا طبق دستورالعمل داده شده در لینک صورت آزمایشگاه، مانگو را نصب کردیم. برای آشنایی با محیط مانگو از روی دستور کار لینک داده شده پایگاه داده retrogames را پیاده سازی کردیم. در ادامه توضیح مختصر و نتیجه هر مرحله آمده است. نتایج هر قسمت از mongoDB Comapass دیده شد.



شمایی از پایگاه داده مورد طراحی

پایگاه داده بایستی شامل موجودیت‌های بالا باشد، اما ما اینجا تنها بازی و بازیکنان را پیاده سازی می‌کنیم.

- پیاده سازی دیتابیس در خط فرمان
- ساخت موجودیت بازی

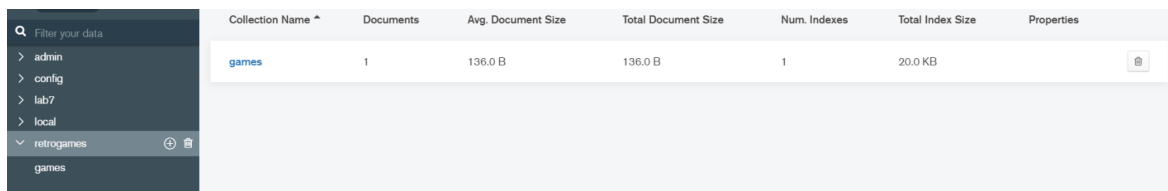
```

> use retrogames
switched to db retrogames
>
game1 = {
  name: "Invaders 2013",
  release_date: new Date(2013, 03, 02),
  categories: ["space", "shooter", "remake"],
  played: false
}
> db.games.save(game1)
writeResult({ "ninserted" : 1 })
> db.games.findOne()
{ "_id" : ObjectId("5ff41aef9cb951ca67cd16c"), "name" : "Invaders 2013", "release_date" : ISODate("2013-04-01T19:30:00Z"), "categories" : [ "space", "shooter", "remake" ], "played" : false }

```

دستور ساخت دیتابیس retrogames و موجودیت بازی

با دستور اول (use retrogames) پایگاه داده ای که باید مانگو از آن در ادامه استفاده کند را مشخص کردیم. عملاً مانند set as default عمل می کند. حال اگر این پایگاه داده موجود نباشد آن را ایجاد می کند. در مرحله بعد یک نمونه از موجودیت game میسازیم با نام game1 و با فرمت JSON خصیصه های مربوط به آن را تولید می کنیم. سپس با دستور db.games.save(game1) نمونه ساخته شده را در کالکشن games از پایگاه داده مورد استفاده و default ذخیره می کنیم. اگر کالکشن موجود نبود آن را می سازد.



Collection Name	Documents	Avg. Document Size	Total Document Size	Num. Indexes	Total Index Size	Properties
games	1	136.0 B	136.0 B	1	20.0 KB	

نتیجه ساخت پایگاه داده و کالکشن در mongoDB Compass

```
_id: ObjectId("5ff41aecf9cb951ca67cd16c")
name: "Invaders 2013"
release_date: 2013-04-01T19:30:00.000+00:00
> categories: Array
played: false
```

نتیجه اضافه شدن نمونه ای از games

- بازیابی داده های کالکشن

اگر بخواهیم تمام داده های یک کالکشن را بازیابی کنیم کافی است بر روی کالکشن find() را صدا کنیم.

```
> db.games.find()
{ "_id" : ObjectId("5ff41aecf9cb951ca67cd16c"), "name" : "Invaders 2013", "release_date" : ISODate("2013-04-01T19:30:00Z"), "categories" : [ "space", "shooter", "remake" ], "played" : false }
```

بازیابی تمام داده های games

اگر بخواهیم تعداد خاصی (n) از داده های ابتدای کالکشن را بازیابی کنیم کافی ست find().limit(n) را بر روی کالکشن صدا کنیم.

```
> db.games.find().limit(100)
{ "_id" : ObjectId("5ff41aecf9cb951ca67cd16c"), "name" : "Invaders 2013", "release_date" : ISODate("2013-04-01T19:30:00Z"), "categories" : [ "space", "shooter", "remake" ], "played" : false }
```

بازیابی ۱۰۰ داده اول games (تنها یک داده ذخیره شده بود).

اگر بخواهیم داده با ویژگی خاصی را بازیابی کنیم کافی ست شرایط را با فرمت مناسب مانگو درون find() ذکر کنیم.

```

> db.games.findOne({ name: "Invaders 2013"})
{
  "_id" : ObjectId("5ff41aecf9cb951ca67cd16c"),
  "name" : "Invaders 2013",
  "release_date" : ISODate("2013-04-01T19:30:00Z"),
  "categories" : [
    "space",
    "shooter",
    "remake"
  ],
  "played" : false
}

```

بازیابی داده با نام Invaders 2013

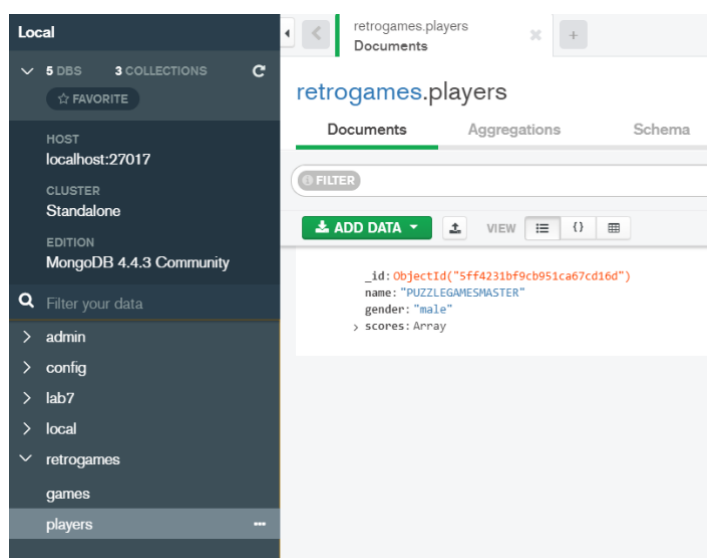
• ساخت موجودیت players

```

> player1 =
...   { name: "PUZZLEGAMESMASTER",
...     gender: "male",
...     scores: [
...       { game_id: new ObjectId("5ff41aecf9cb951ca67cd16c"),
...         game_name: "Invaders 2013",
...         score: 10500,
...         score_date: new Date(2013, 03, 02)
...       }
...     ]
...   }
...
...   "name" : "PUZZLEGAMESMASTER",
...   "gender" : "male",
...   "scores" : [
...     {
...       "game_id" : ObjectId("5ff41aecf9cb951ca67cd16c"),
...       "game_name" : "Invaders 2013",
...       "score" : 10500,
...       "score_date" : ISODate("2013-04-01T19:30:00Z")
...     }
...   ]
... }
> db.players.save(player1)
WriteResult({ "nInserted" : 1 })

```

دستور ساخت موجودیت players و اضافه کردن داده به آن



حاصل ساخت کاکشن players و اضافه کردن یک نمونه به آن

بالتر توضیح داده شد.

#### • بروزرسانی داده

برای بروزرسانی داده ها از دستور update() بر روی کالکشن استفاده می کنیم. این دستور دو قسمت دارد. در قسمت اول مشخص می کند داده با چه خصوصیاتی باید بروز شود، در بخش دوم می گوید چه تغییری باید بکند. در اینجا ما برای هر بازی مشخص می کنیم تا الان بازی شده است یا نه، که ابتدا نه است. حال که یک بازیکن داریم که بازی کرده است باید خصیصه بازی شده را برای بازی مربوطه به روز کنیم. بنابراین با دستور زیر می گوئیم بازی با شناسه مربوطه را خصیصه بازی شدن را برایش true کن.

```
> db.games.update(
...   { _id: new ObjectId("5ff41aecf9cb951ca67cd16c") },
...   { $set: { played: true } }
... )
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.players.update(
```

دستور بروزرسانی نمونه ای از games

```
_id: ObjectId("5ff41aecf9cb951ca67cd16c")
name: "Invaders 2013"
release_date: 2013-04-01T19:30:00.000+00:00
> categories: Array
played: true
```

#### حاصل بروزرسانی

اگر بخواهیم در قالب اضافه کردن یک داده به آرایه مربوط به یک خصیصه داده آن را بروزرسانی کنیم می توانیم در دستور بروزرسانی نوع عملیات را push قرار دهیم. مثلاً اگر بازیکن دوباره بازی کرد و خواستیم امتیاز جدید آن را ذخیره کنیم می توانیم طبق زیر عمل کنیم. (دستور داده شده اشکال syntax ای داشت هم آن دستور و هم فرمت درست شده در پایین آمده اند.)

```
> db.players.update(
...   { _id: new ObjectId("5ff4231bf9cb951ca67cd16d") },
...   { $push: { scores: { game_id: new ObjectId("5ff41aecf9cb951ca67cd16c"),
...     game_name: «Invaders 2013»,
...     score: 30250,
...     score_date: new Date(2013, 03, 03)
...   } }
... )
uncaught exception: SyntaxError: illegal character :
(she11):4:19
> db.players.update( { _id: new ObjectId("5ff4231bf9cb951ca67cd16d") }, { $push: { scores: { game_id: new ObjectId("5ff41aecf9cb951ca67cd16c"),
...   game_name: "Invaders 2013", score: 30250, score_date: new Date(2013, 03, 03) } } })
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
>
```

#### دستور بروزرسانی امتیازهای player

```
_id: ObjectId("5ff4231bf9cb951ca67cd16d")
name: "PUZZLEGAMESMASTER"
gender: "male"
> scores: Array
> 0: Object
> 1: Object
```

#### حاصل بروزرسانی

در نهایت با همان دستور یافتن نمونه ای خاص از داده که بالاتر آمد، بازیکن را می یابیم تا مطمئن شویم نمونه روزرسانی شده اند.

```
> db.players.findOne({ name: "PUZZLEGAMESMASTER"})
{
  "_id" : ObjectId("5ff4231bf9cb951ca67cd16d"),
  "name" : "PUZZLEGAMESMASTER",
  "gender" : "male",
  "scores" : [
    {
      "game_id" : ObjectId("5ff41aecf9cb951ca67cd16c"),
      "game_name" : "Invaders 2013",
      "score" : 10500,
      "score_date" : ISODate("2013-04-01T19:30:00Z")
    },
    {
      "game_id" : ObjectId("5ff41aecf9cb951ca67cd16c"),
      "game_name" : "Invaders 2013",
      "score" : 30250,
      "score_date" : ISODate("2013-04-02T19:30:00Z")
    }
  ]
}
```

بررسی نمونه بازیکن بعد از روزرسانی




















#### ○ پیاده سازی دیتابیس در نرم افزار Studio 3T

حال تمام مراحل بالا را اما درون نرم افزار انجام میدهم. طبق خواسته آزمایشگاه ده بازی و ده بازیکن اضافه میکنیم. دستورات پیاده سازی دقیقاً همان هستند بنابراین توضیحی نمی دهم مگر جایی که نیاز باشد.

#### ● ساخت دیتابیس retrogames2 و کالکشن بازی ها

```
1 use retrogames2
2 game1 =
3   { name: "Invaders 2013",
4     release_date: new Date(2013, 03, 02),
5     categories: ["space", "shooter", "remake"],
6     played: false
7   }
8
9   db.games.save(game1)
10
11 game2 =
12   { name: "game2",
13     release_date: new Date(2014, 03, 02),
14     categories: ["space", "adventure", "remake"],
15     played: false
16   }
17
18   db.games.save(game2)
19
20 game3 =
21   { name: "game3",
22     release_date: new Date(2015, 05, 25),
23     categories: ["puzzle", "thinking"],
24     played: false
25   }
26
27   db.games.save(game3)
28
29 game4 =
30   { name: "game4",
31     release_date: new Date(2020, 08, 10),
32     categories: ["remake"],
33     played: false
34   }
```

بخشی از دستورات ذخیره ۱۰ بازی جدید

_id ObjectId	name String	release_date Date	categories Array	played Boolean	
1 5ff43e6f77d11e978eaf2d20	"Invaders 2013"	2013-04-01T19:30:00.000+00:00	[ ] 3 elements	false	  
2 5ff4489b77d11e978eaf2d21	"game2"	2014-04-01T19:30:00.000+00:00	[ ] 3 elements	false	  
3 5ff448ed77d11e978eaf2d22	"game3"	2015-06-24T19:30:00.000+00:00	[ ] 2 elements	false	  
4 5ff4499877d11e978eaf2d23	"game4"	2017-06-24T19:30:00.000+00:00	[ ] 2 elements	false	  
5 5ff449a877d11e978eaf2d24	"game5"	2013-04-26T19:30:00.000+00:00	[ ] 2 elements	false	  
6 5ff449b077d11e978eaf2d25	"game6"	2019-04-01T19:30:00.000+00:00	[ ] 3 elements	false	  
7 5ff449f077d11e978eaf2d26	"game7"	2018-04-01T19:30:00.000+00:00	[ ] 3 elements	false	  
8 5ff449fa77d11e978eaf2d27	"game8"	2014-12-01T20:30:00.000+00:00	[ ] 2 elements	false	  
9 5ff44a0477d11e978eaf2d28	"game9"	2007-05-01T19:30:00.000+00:00	[ ] 2 elements	false	  
10 5ff44a0477d11e978eaf2d29	"game10"	2013-04-28T19:30:00.000+00:00	[ ] 1 elements	false	  

حاصل اضافه کردن ۱۰ بازی جدید

- بازیابی داده های کالکشن

```
1 db.games.find().limit(5)
```

Document × Text × Document × Text × Document × Text × Document × Text × Document × Text × Document

50 Documents 1 to 5

games > name

_id	name	release_date	categories	played
5ff43e6f77d11e978eaf2d20	Invaders 2013	2013-04-01T19:30:00.000+00:00	[ ] [ 3 elements ]	false
5ff4489b77d11e978eaf2d21	game2	2014-04-01T19:30:00.000+00:00	[ ] [ 3 elements ]	false
5ff448ed77d11e978eaf2d22	game3	2015-06-24T19:30:00.000+00:00	[ ] [ 2 elements ]	false
5ff4499877d11e978eaf2d23	game4	2017-06-24T19:30:00.000+00:00	[ ] [ 2 elements ]	false
5ff449a877d11e978eaf2d24	game5	2013-04-26T19:30:00.000+00:00	[ ] [ 2 elements ]	false

دستور و خروجی حاصل از بازیابی ۵ داده اول بازی ها

```
1 db.games.findOne({ name: "game7"})
```

Text × Document × Text × Document × Text × Document × Text × Document × Text × Document

50 Documents 1 to 1

Result > name

_id	name	release_date	categories	played
5ff449f077d11e978eaf2d26	game7	2018-04-01T19:30:00.000+00:00	[ ] [ 3 elements ]	false

دستور و خروجی بازیابی داده با خصوصیت خاص (نام = game7)

- ساخت موجودیت players و بروزرسانی داده

در این قسمت موازی با اضافه کردن یک بازیکن جدید بسته به بازی که کرده بود، خصیصه played آن بازی را هم بروز کردم. این قسمت کد کامل آورده شده است تا کامل ببینیم چه بازی هایی باید بروز شوند.

```

1 player1 =
2   { name: "PUZZLEGAMESMASTER",
3     gender: "male",
4     scores: [
5       { game_id: new ObjectId("5ff43e6f77d11e978eaf2d20"),
6         game_name: "Invaders 2013",
7         score: 10500,
8         score_date: new Date(2013, 03, 02)
9       }
10    ]
11  }
12 db.players.save(player1)
13
14 db.games.update(
15   { _id: new ObjectId("5ff43e6f77d11e978eaf2d20") },
16   { $set: { played: true } }
17 )
18
19 player2 =
20   { name: "player2",
21     gender: "female",
22     scores: [
23       { game_id: new ObjectId("5ff4489b77d11e978eaf2d21"),
24         game_name: "game2",
25         score: 1050,
26         score_date: new Date(2020, 10, 16)
27       }
28    ]
29  }
30 db.players.save(player2)
31
32 db.games.update(
33   { _id: new ObjectId("5ff4489b77d11e978eaf2d21") },
34   { $set: { played: true } }
35 )

```

```

37 player3 =
38   { name: "player3",
39     gender: "male",
40     scores: [
41       { game_id: new ObjectId("5ff448ed77d11e978eaf2d22"),
42         game_name: "game3",
43         score: 10539,
44         score_date: new Date(2020, 10, 16)
45       }
46    ]
47  }
48 db.players.save(player3)
49
50 db.games.update(
51   { _id: new ObjectId("5ff448ed77d11e978eaf2d22") },
52   { $set: { played: true } }
53 )
54
55 player4 =
56   { name: "player4",
57     gender: "female",
58     scores: [
59       { game_id: new ObjectId("5ff449a877d11e978eaf2d24"),
60         game_name: "game5",
61         score: 10536,
62         score_date: new Date(2020, 10, 16)
63       }
64    ]
65  }
66 db.players.save(player4)
67
68 db.games.update(
69   { _id: new ObjectId("5ff449a877d11e978eaf2d24") },
70   { $set: { played: true } }
71 )
72

```



```
72
73 player5 =
74     { name: "player5",
75       gender: "male",
76       scores: [
77         { game_id: new ObjectId("5ff449b977d11e978eaf2d25"),
78           game_name: "game6",
79           score: 10500,
80           score_date: new Date(2020, 10, 16)
81         }
82       ]
83     }
84 db.players.save(player5)
85
86 db.games.update(
87   { _id: new ObjectId("5ff449b977d11e978eaf2d25") },
88   { $set: { played: true } }
89 )
90
91 player6 =
92     { name: "player6",
93       gender: "female",
94       scores: [
95         { game_id: new ObjectId("5ff449f077d11e978eaf2d26"),
96           game_name: "game7",
97           score: 4903,
98           score_date: new Date(2020, 10, 16)
99         }
100       ]
101     }
102 db.players.save(player6)
103
104 db.games.update(
105   { _id: new ObjectId("5ff449f077d11e978eaf2d26") },
106   { $set: { played: true } }
107 )
108
109
110 player7 =
111     { name: "player7",
112       gender: "male",
113       scores: [
114         { game_id: new ObjectId("5ff449fa77d11e978eaf2d27"),
115           game_name: "game8",
116           score: 1528,
117           score_date: new Date(2020, 10, 16)
118         }
119       ]
120     }
121 db.players.save(player7)
122
123 db.games.update(
124   { _id: new ObjectId("5ff449fa77d11e978eaf2d27") },
125   { $set: { played: true } }
126 )
127
128 player8 =
129     { name: "player8",
130       gender: "female",
131       scores: [
132         { game_id: new ObjectId("5ff44a0477d11e978eaf2d28"),
133           game_name: "game9",
134           score: 10090,
135           score_date: new Date(2020, 10, 16)
136         }
137       ]
138     }
139 db.players.save(player8)
140
141 db.games.update(
142   { _id: new ObjectId("5ff44a0477d11e978eaf2d28") },
143   { $set: { played: true } }
144 )
```































```

145 player9 =
146   { name: "player9",
147     gender: "male",
148     scores: [
149       { game_id: new ObjectId("5ff44a0477d11e978eaf2d29"),
150         game_name: "game10",
151         score: 1398,
152         score_date: new Date(2020, 10, 16)
153       }
154     ]
155   }
156 db.players.save(player9)
157
158 db.games.update(
159   { _id: new ObjectId("5ff44a0477d11e978eaf2d29") },
160   { $set: { played: true } }
161 )
162
163 player10 =
164   { name: "player10",
165     gender: "female",
166     scores: [
167       { game_id: new ObjectId("5ff43e6f77d11e978eaf2d20"),
168         game_name: "Invaders 2013",
169         score: 10589,
170         score_date: new Date(2020, 10, 16)
171       }
172     ]
173   }
174 db.players.save(player10)































```

دستورات اضافه کردن و بروزرسانی

همانطور که دیده می شود همه بازی ها انجام شدند جز game4.

_id ObjectId	name String	gender String	scores Array	
1 5ff49a9b77d11e978eaf2d3c	"PUZZLEGAMESMASTER"	"male"	[ ] 1 elements	  
2 5ff49a9b77d11e978eaf2d3d	"player2"	"female"	[ ] 1 elements	  
3 5ff49a9b77d11e978eaf2d3e	"player3"	"male"	[ ] 1 elements	  
4 5ff49a9b77d11e978eaf2d3f	"player4"	"female"	[ ] 1 elements	  
5 5ff49a9b77d11e978eaf2d40	"player5"	"male"	[ ] 1 elements	  
6 5ff49a9b77d11e978eaf2d41	"player6"	"female"	[ ] 1 elements	  
7 5ff49a9b77d11e978eaf2d42	"player7"	"male"	[ ] 1 elements	  
8 5ff49a9b77d11e978eaf2d43	"player8"	"female"	[ ] 1 elements	  
9 5ff49a9b77d11e978eaf2d44	"player9"	"male"	[ ] 1 elements	  
10 5ff49a9b77d11e978eaf2d45	"player10"	"female"	[ ] 1 elements	  

نتیجه اضافه کردن ۱۰ کاربر جدید

_id ObjectId	name String	release_date Date	categories Array	played Boolean	
1 5ff43e6f77d11e978eaf2d20	"Invaders 2013"	2013-04-01T19:30:00.000+00:00	[ ] 3 elements	true	  
2 5ff4489b77d11e978eaf2d21	"game2"	2014-04-01T19:30:00.000+00:00	[ ] 3 elements	true	  
3 5ff448ed77d11e978eaf2d22	"game3"	2015-06-24T19:30:00.000+00:00	[ ] 2 elements	true	  
4 5ff4499b77d11e978eaf2d23	"game4"	2017-06-24T19:30:00.000+00:00	[ ] 2 elements	false	  
5 5ff449a877d11e978eaf2d24	"game5"	2013-04-26T19:30:00.000+00:00	[ ] 2 elements	true	  
6 5ff449b977d11e978eaf2d25	"game6"	2019-04-01T19:30:00.000+00:00	[ ] 3 elements	true	  
7 5ff449f077d11e978eaf2d26	"game7"	2018-04-01T19:30:00.000+00:00	[ ] 3 elements	true	  
8 5ff449fa77d11e978eaf2d27	"game8"	2014-12-01T20:30:00.000+00:00	[ ] 2 elements	true	  
9 5ff44a0477d11e978eaf2d28	"game9"	2007-05-01T19:30:00.000+00:00	[ ] 2 elements	true	  
10 5ff44a0477d11e978eaf2d29	"game10"	2013-04-28T19:30:00.000+00:00	[ ] 1 elements	true	  

نتیجه بروزرسانی بازی ها (played برای همه true است به جز game4)

حال به امتیازات player5 یک امتیاز اضافه میکنیم.

```

1 db.players.update(
2   { _id: new ObjectId("5ff49a9077d11e978eaf2d40") },
3   { $push: { scores: { game_id: new ObjectId("5ff449b977d11e978eaf2d25"),
4     game_name: "game6",
5     score: 21569,
6     score_date: new Date(2021, 01, 01)
7   }
8 }
9 })

```

اضافه کردن امتیازی جدید به امتیازهای player5

_id ObjectId	name String	gender String	scores Array
1 5ff49a9077d11e978eaf2d3c	"PUZZLEGAMESMASTER"	"male"	[ ] 1 elements
2 5ff49a9077d11e978eaf2d3d	"player2"	"female"	[ ] 1 elements
3 5ff49a9077d11e978eaf2d3e	"player3"	"male"	[ ] 1 elements
4 5ff49a9077d11e978eaf2d3f	"player4"	"female"	[ ] 1 elements
5 5ff49a9077d11e978eaf2d40	"player5"	"male"	[ ] 2 elements
6 5ff49a9077d11e978eaf2d41	"player6"	"female"	[ ] 1 elements
7 5ff49a9077d11e978eaf2d42	"player7"	"male"	[ ] 1 elements
8 5ff49a9077d11e978eaf2d43	"player8"	"female"	[ ] 1 elements
9 5ff49a9077d11e978eaf2d44	"player9"	"male"	[ ] 1 elements
10 5ff49a9177d11e978eaf2d45	"player10"	"female"	[ ] 1 elements

نتیجه اضافه کردن امتیاز جدید

```

1 db.players.findOne({ name: "player5"})

```

Document x

← ← → → | 50 | Documents 1 to 1 | 🔍

```

1 {
2   "_id" : ObjectId("5ff49a9077d11e978eaf2d40"),
3   "name" : "player5",
4   "gender" : "male",
5   "scores" : [
6     {
7       "game_id" : ObjectId("5ff449b977d11e978eaf2d25"),
8       "game_name" : "game6",
9       "score" : 10500.0,
10      "score_date" : ISODate("2020-11-15T20:30:00.000+0000")
11    },
12    {
13      "game_id" : ObjectId("5ff449b977d11e978eaf2d25"),
14      "game_name" : "game6",
15      "score" : 21569.0,
16      "score_date" : ISODate("2021-01-31T20:30:00.000+0000")
17    }
18  ]
19 }
20

```

بازیابی player5 و دیدن دو امتیاز او (بروزرسانی شده)

```
1 db.players.findOne({ name: "player10"})
```

```
1 {
2   "_id" : ObjectId("5ff49a9177d11e978eaf2d45"),
3   "name" : "player10",
4   "gender" : "female",
5   "scores" : [
6     {
7       "game_id" : ObjectId("5ff43e6f77d11e978eaf2d20"),
8       "game_name" : "Invaders 2013",
9       "score" : 10589.0,
10      "score_date" : ISODate("2020-11-15T20:30:00.000+0000")
11    }
12  ]
13 }
```

بازیابی player10 و دیدن اطلاعات او

○ فراخوانی API و ذخیره در مانگو و پیاده سازی چند پرس و جو در پایتون

در این قسمت ابتدا از API ای که داده شده است تعدادی توییت (طبق موافقت استاد ۱۵۰ به جای ۱۰۰۰) استخراج میکنیم و با کد پایتونی که در ادامه توضیح داده خواهد شد، به یک دیتابیس مانگو اضافه کردیم.

● فراخوانی API در پست من و دیدن فیلدهای خروجی

```
1 {
2   "errorcode": "error",
3   "errortitle": "تایید نبودن",
4   "success": true,
5   "hasmore": true,
6   "items": [
7     {
8       "id": "280599551",
9       "sendtime": "2021-01-05T19:43:03Z",
10      "sendtimePersian": "1399/10/16 23:13",
11      "parentSendTime": "2021-01-05T07:43:20Z",
12      "parentSendTimePersian": "1399/10/16 11:11",
13      "parentid": "280496740",
14      "parentUsername": "taze varad",
15      "parentSenderId": "tazevarad1399",
16      "parentSenderProfileImage": "60161610-b0c7-4242-992f-e35ab207c734",
17      "parentContent": "یه هفته اول سال از میانه‌های جوی دو بودیم اما الان فاصله اشتغال میگیره این مهم بایده حد اکثر 10 تا 15 هزار تومان بکشد... این مهم من چقدر بیدار پامین من تخیل و فکر... حد فروز رو که افرو 90 درصد آلوده اونجا شد مهم دو نفر... جزا... نه",
18      "senderName": "taze varad",
19      "senderUsername": "tazevarad1399",
20      "senderProfileImage": "60161610-b0c7-4242-992f-e35ab207c734",
21      "content": "آبشده... فضا به فضا مختلف میانه ارد و موافق تومان گیره... مهم اما قدر فرق می کنه",
22      "type": "tweet",
23      "finalUpdateVersion": ""
24    },
25    {
26      "id": "280599551",
27      "sendtime": "2021-01-05T19:43:19Z",
28      "sendtimePersian": "1399/10/16 23:13",
29      "senderName": "Mohan",
30      "senderUsername": "mohammasid110",
31      "senderProfileImage": "default",
32      "content": "آبشده... فضا به فضا مختلف میانه ارد و موافق تومان گیره... مهم اما قدر فرق می کنه",
33      "type": "tweet",
34      "scorePostDate": "160875842206",
35      "finalUpdateVersion": ""
36    },
37    {
38      "id": "280599550",
39      "sendtime": "2021-01-05T19:43:14Z",
40      "sendtimePersian": "1399/10/16 23:13",
```

حاصل فراخوانی API

همانطور که در خروجی میبینیم در فیلد items چند توثیت وجود دارد. پس با هر بار فراخوانی چند توثیت استخراج می شود.

- فراخوانی با کد پایتون و بقیه عملیات مورد نیاز

```
class MongoDB(object):
    def __init__(self, host='localhost', port=27017, database_name=None, collection_name=None):
        try:
            self._connection = MongoClient(host=host, port=port, maxPoolSize=200)
        except Exception as error:
            raise Exception(error)
        self._database = None
        self._collection = None
        if database_name:
            self._database = self._connection[database_name]
        if collection_name:
            self._collection = self._database[collection_name]

    def __readFromUrl(self, url):
        response = requests.get(url)
        if response.status_code != 200:
            print('Failed to get data:', response.status_code)
        data = response.text
        return data

    def __checkExist(self, id):
        cnt = self._collection.count_documents({"id": id})
        return (cnt == 0)

    def insert(self, url, tweetNum):
        count = self._collection.estimated_document_count()
        while(count != tweetNum):
            print("Count:::", count)
            result = json.loads(self.__readFromUrl(url))
            data = result["items"]
            for tweet in data:
                doesNotExist = self.__checkExist(tweet["id"])
                if doesNotExist:
                    self._collection.insert_one(tweet)
                    count += 1
            return "%d added to tweets DB", count

    def findUserWithMostTweets(self):
        sortedCntUsers = list(self._collection.aggregate([
            {"$sortByCount": "$senderUsername"}
        ]))
        return sortedCntUsers[0]

    def getTweets(self, username):
        tweets = list(self._collection.find({"senderUsername": username}))
        for tweet in tweets:
            print(tweet["content"], "\n")
```

کد پیاده سازی شده در پایتون

در کد بالا کلاسی به نام MongoDB داریم که به جهت مدیریت و کار با پایگاه داده پیاده سازی شده است.

در تابع \_\_init\_\_ یک اتصال به پایگاه داده مانگو با host و port ای که روی کامپیوترمان هست ایجاد می کنیم. سپس در این اتصال پایگاه داده مورد نظرمون را پیدا می کنیم و در آن پایگاه داده هم به کالکشن موردنظرمان متصل میشویم.

در تابع `readFromUrl` یک آدرس (API داده شده) میگیریم و یک درخواست از نوع `Get` به آن آدرس میفرستیم و پاسخ آن را دریافت میکنیم. چک میکنیم تا پاسخ مشکلی نداشته باشد و وضعیت 200 باشد. سپس پاسخ را در قالب `text` دریافت میکنیم.

بایستی تویت های دریافتی حتماً چک شوند تا تکراری نباشند. در تابع `checkExist` شناسه یک تویت را میگیریم. سپس با کمک تابع `count_documents` تعداد داده ها با این شناسه را پیدا می کنیم. این تعداد بایستی صفر باشد وگرنه تکراری است ونباید مجدداً ذخیره شود.

در تابع `insert` ابتدا سازی که پایگاه داده، دارد را حساب میکنیم. این به ما کمک می کند که اگر خطای 503 رخ داد و اتصال قطع شد، بدانیم تا الان چند داده در پایگاه داده داریم و چند تا دیگر باید بگیریم تا به تعداد موردنظر برسیم. دقت شود از تابع `estimated_document_count()` استفاده کرده ایم. زیرا تابع `count` در پایتون با مشکل روبرو می شود. حال تا زمانی که تعداد داده های درون پایگاه داده با تعداد درخواستی یکسان نیست، از تویت میگیریم. سپس از خروجی `JSON` به دست آمده با کمک `json.loads` داده های (تویت های) درون فیلد `items` را جدا میکنیم و چون یک دسته داده (مجموعه ای از تویت ها) است در یک حلقه `for` بعد از چک کردن عدم تکراری بودن با تابع بالا، درون کالکشن با دستور `insert_one()` ذخیره می کنیم و تعداد داده های درون پایگاه داده را هم به روزرسانی می کنیم.

```
url = 'https://www.sahamyab.com/guest/twiter/list?v=0.1'
mongo_db = MongoDB(database_name='SahamTweets', collection_name='tweets')
mongo_db.insert(url, 150)
```

دستور استفاده از کد بالا

#	tweet	_id ObjectId	id String	sendTime String	sendTimePersian String	senderName String	senderUsername Str
1		5ff4e73b157c427ae22c9478	"206661066"	"2021-01-05T22:24:47Z"	"1399/10/17 01:54"	"fadaiani"	"fadaiani"
2		5ff4e731157c427ae22c9479	"206661052"	"2021-01-05T22:24:35Z"	"1399/10/17 01:54"	"qushchi"	"qushchi"
3		5ff4e731157c427ae22c947a	"206661025"	"2021-01-05T22:24:18Z"	"1399/10/17 01:54"	"tempo"	"tempo"
4		5ff4e731157c427ae22c947b	"206661005"	"2021-01-05T22:24:04Z"	"1399/10/17 01:54"	"sajjaddaliry"	"sajjaddaliry"
5		5ff4e731157c427ae22c947c	"206660995"	"2021-01-05T22:23:58Z"	"1399/10/17 01:53"	"mostafa"	"mostafa229945"
6		5ff4e731157c427ae22c947d	"206660982"	"2021-01-05T22:23:53Z"	"1399/10/17 01:53"	"محسن"	"release"
7		5ff4e731157c427ae22c947e	"206660935"	"2021-01-05T22:23:32Z"	"1399/10/17 01:53"	"بروگر"	"sososos"
8		5ff4e731157c427ae22c947f	"206660870"	"2021-01-05T22:23:00Z"	"1399/10/17 01:53"	"ZamZam"	"zamzan1982"
9		5ff4e731157c427ae22c9480	"206660840"	"2021-01-05T22:22:48Z"	"1399/10/17 01:52"	"ماد"	"enadking"
10		5ff4e731157c427ae22c9481	"206660841"	"2021-01-05T22:22:47Z"	"1399/10/17 01:52"	"ساره ای"	"sahbania"
11		5ff4e7493ae6fc0b09e3d403	"206661114"	"2021-01-05T22:25:12Z"	"1399/10/17 01:55"	"علی داد اف"	"sajad07654321"
12		5ff4e7bc503708c301549b	"206661118"	"2021-01-05T22:26:51Z"	"1399/10/17 01:56"	"مافق ایران"	"fromiran"
13		5ff4e7bc503708c301549c	"206659581"	"2021-01-05T22:12:29Z"	"1399/10/17 01:42"	"Amirkhan"	"zizo04"
14		5ff4e7bc503708c301549d	"206661271"	"2021-01-05T22:26:30Z"	"1399/10/17 01:56"	"مهرداد اری"	"fatemi1369"
15		5ff4e7bc503708c301549e	"206661240"	"2021-01-05T22:26:22Z"	"1399/10/17 01:56"	"mhkhanl"	"mhkhanl"
16		5ff4e7bc503708c301549f	"206661186"	"2021-01-05T22:25:51Z"	"1399/10/17 01:55"	"مزه ای"	"sahbania"
17		5ff4e83d7145b20902e8e518	"206661477"	"2021-01-05T22:28:19Z"	"1399/10/17 01:58"	"علی داد اف"	"sajad07654321"
18		5ff4e83d7145b20902e8e519	"206661458"	"2021-01-05T22:28:09Z"	"1399/10/17 01:58"	"محسن"	"release"

حاصل اجرای کد بالا و ذخیره ۱۵۰ داده

- یافتن کاربر با بیشترین تعداد توییت

هر کاربر username یکتایی دارد. و برای هر توییت username کاربر ثبت می شود. بنابراین اگر ببینیم در داده ها username کدام کاربر بیشتر تکرار شده است متوجه می شویم کدام کاربر بیشتر توییت کرده است.

```
def findUserWithMostTweets(self):
    sortedCntUsers = list(self._collection.aggregate([
        {"$sortByCount": "$senderUsername"}
    ]))
    return sortedCntUsers[0]
```

تابع یافتن کاربر با بیشترین تعداد توییت

در تابع بالا که در کلاس MangoDB پیاده سازی شده است، بر روی کالکشنمان یک دستور aggregation پیاده سازی کرده ایم. بدین شکل که با کمک \$sortByCount داده ها را براساس تعداد خصیصه ای که جلوییش ذکر شده (senderUsername) به صورت نزولی مرتب میکنیم. حال نتیجه را به صورت لیست در sortedCntUsers ذخیره می کنیم. اما چون فقط بیشترین را می خواهیم، عضو اول این آرایه برگردانده می شود.

```
a = mongo_db.findUserWithMostTweets()
print(a)
```

دستور فراخوانی و چاپ نتیجه این قسمت

```
{'_id': 'gharibzahedi', 'count': 5}
```

نتیجه - کاربر با بیشترین تعداد توییت ها

- یافتن توییت های کاربری خاص

```
def getTweets(self, username):
    tweets = list(self._collection.find({"senderUsername" : username}))
    for tweet in tweets:
        print(tweet["content"], "\n")
```

دستور و تابع پیاده سازی شده جهت یافتن توییت های کاربری خاص

در تابع بالا که در کلاس MangoDB پیاده سازی شده است، همانطور که در بالاتر گفته شد با کمک find و نوشتن شرایط مورد نظر که اینجا username کاربر است، داده های مرتبط با آن را می یابیم. نتیجه را در یک لیست ذخیره می کنیم. سپس آن را پیمایش کرده و به ازای هرکدام مقدار content که متن توییت است را چاپ میکنیم.

```
mongo_db.getTweets('gharibzahedi')|
```

دستور فراخوانی تابع

#شخص\_پورس با خریدن عرضه اولیه فردا (سپیدار) برای یکبار که شده اتحاد خودمونو اعلام کنیم، شاد باشید.

#وئجارت با خریدن عرضه اولیه فردا (سپیدار) برای یکبار که شده اتحاد خودمونو اعلام کنیم، شاد باشید.

#تسنا با خریدن عرضه اولیه فردا (سپیدار) برای یکبار که شده اتحاد خودمونو اعلام کنیم، شاد باشید.

#شینا با خریدن عرضه اولیه فردا (سپیدار) برای یکبار که شده اتحاد خودمونو اعلام کنیم، شاد باشید.

#دی با خریدن عرضه اولیه فردا (سپیدار) برای یکبار که شده اتحاد خودمونو اعلام کنیم، شاد باشید.

نتیجه- توییت های کاربر با `username = gharibzahedi`

توییت های بالا یکسان نیستند، # های متفاوتی دارند.