

# *Sistemi Informativi Evoluti e Big Data*



## MongoDB - Esercitazione

**Prof. Devis Bianchini**

**Università degli Studi di Brescia**

**Dipartimento di Ingegneria dell'Informazione**

# Sommario e strumenti

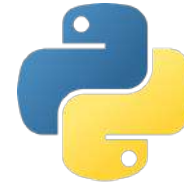
- Richiami sul modello dei dati in MongoDB
- Inserimento di documenti nel database
- Interrogazione in MongoDB
- Modifica e cancellazione di documenti
- Il meccanismo della validazione

<https://www.mongodb.com/docs/manual/>

MondoDB Community Edition



Python - PyMongo



Robo 3T (formerly Robomongo)

<http://mongodb-tools.com/tool/robomongo/>



Studio 3T

# MongoDB – Modello dei dati e setup

MongoDB	RDBMS (e.g., MySQL, Postgres)
Database	Database
Collezioni	Tabelle
Documenti/oggetti	Record/righe
Query ritornano un puntatore a documenti	Query ritornano record



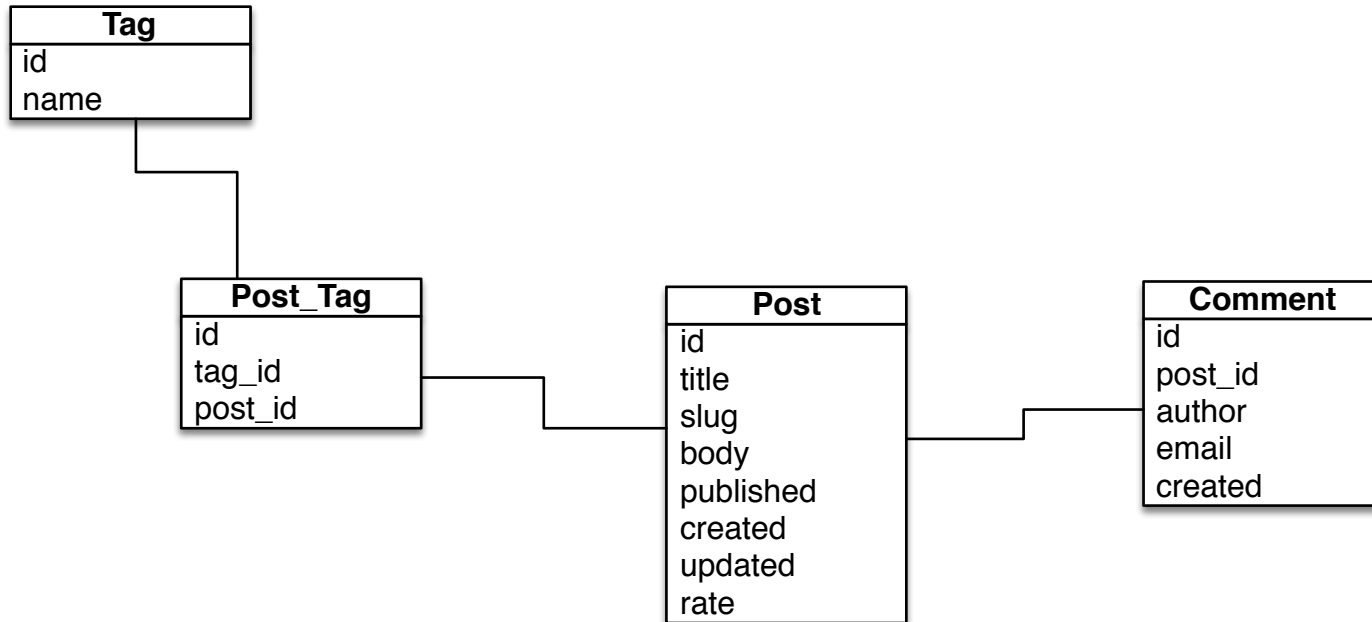
```
import pymongo
from pymongo import MongoClient
```

```
myclient = MongoClient("mongodb://localhost:27017/")
# myclient = MongoClient() # per le impostazioni di default
# myclient = MongoClient("localhost", 27017)
```

```
mydb = myclient["mydatabase"]
# mydb = myclient.mydatabase
```

```
mycol = mydb["post"]
# mycol = mydb.post
```

# Esempio – in MySQL...



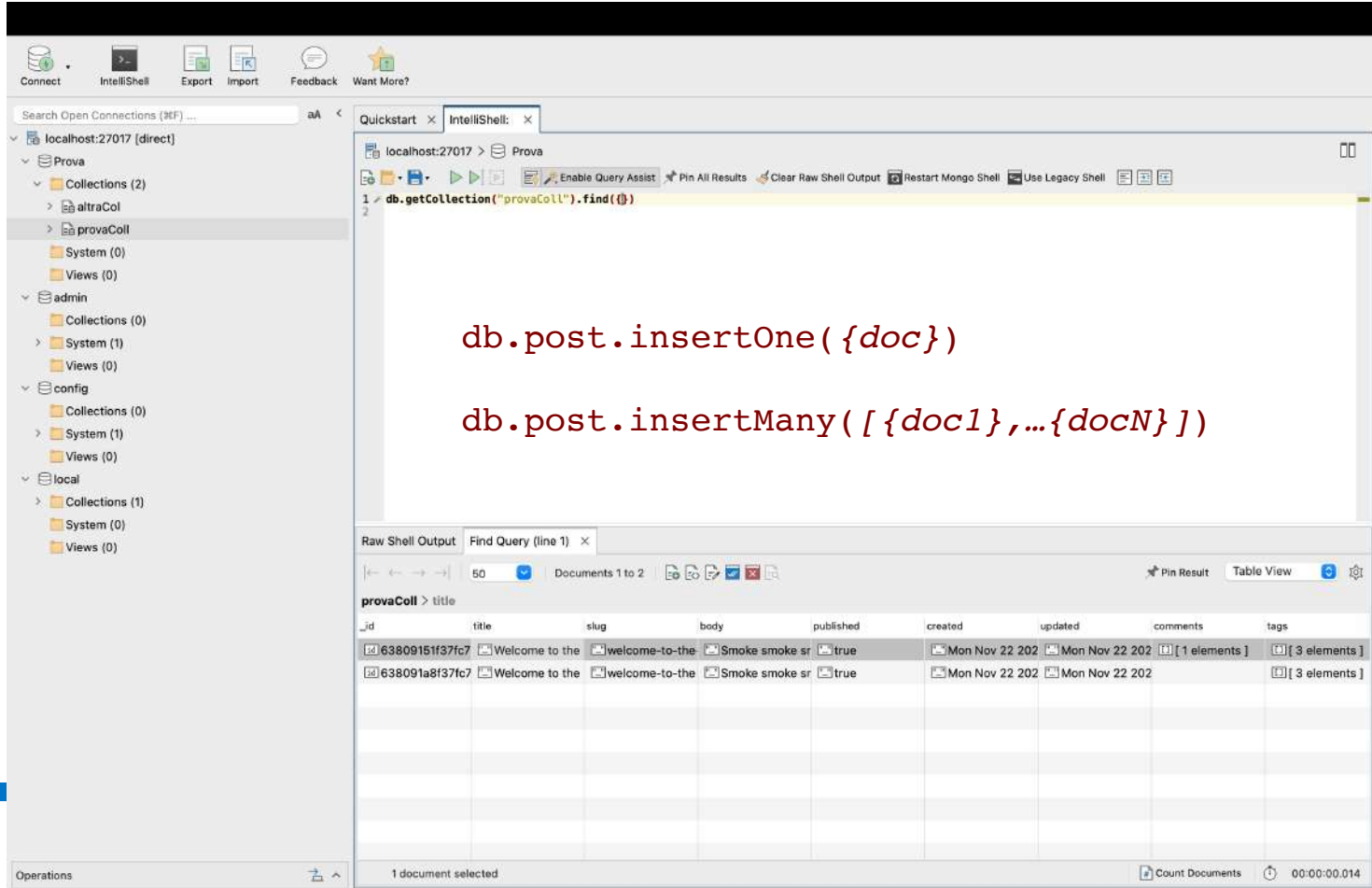
# Esempio – ...e in MongoDB

Post	
id	
title	
slug	
body	
published	
created	
updated	
Comment	
id	
post_id	
author	
email	
created	
Tag	
id	
name	
rate	

```
{ "title": "Welcome to the Marlboro country",
  "slug": "welcome-to-the-marlboro-country",
  "body": "Smoke smoke smoke...",
  "published": "true",
  "created": "Mon Nov 22 2021 22:37:22",
  "updated": "Mon Nov 22 2021 22:37:22",
  "comments" : [
    {
      "author": "Devis",
      "email": "devis.bianchini@unibs.it",
      "body": "We have not to smoke too much...",
      "created": "Mon Nov 22 2021 22:38:45"
    }
  ],
  "tags": ["databases", "MongoDB", "awesome"],
  "rate": 5
}
```



# Inserire documenti in MongoDB (I)



The screenshot shows the MongoDB Compass application. On the left, the 'Prova' database is expanded, showing collections 'ultraCol' and 'provaColl'. The 'provaColl' collection is selected. The main panel displays the IntelliShell with the following code:

```
1 db.getCollection("provaColl").find({})
2
```

Below the code, the 'Raw Shell Output' tab shows the results of the query, displaying a table with columns: `_id`, `title`, `slug`, `body`, `published`, `created`, `updated`, `comments`, and `tags`. The table contains two rows of data:

_id	title	slug	body	published	created	updated	comments	tags
63809151f37fc7	Welcome to the	welcome-to-the	Smoke smoke sr	true	Mon Nov 22 202	Mon Nov 22 202	[1 elements]	[3 elements]
638091a8f37fc7	Welcome to the	welcome-to-the	Smoke smoke sr	true	Mon Nov 22 202	Mon Nov 22 202	[1 elements]	[3 elements]

The bottom status bar indicates '1 document selected' and 'Count Documents 00:00:00.014'.

# Inserire documenti in MongoDB (II)

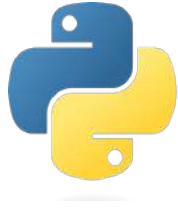


```
import datetime
myPost = {"title": "Welcome to the marlboro country",
          "slug": "welcome-to-marlboro",
          "body": "Smoke smoke smoke...",
          "published": True,
          "created": datetime.datetime.now().strftime("%a %b %d %Y %H:%M:%S"),
          "updated": datetime.datetime.now().strftime("%a %b %d %Y %H:%M:%S"),
          "comments": [
              {
                  "author": "Devis",
                  "email": "devis.bianchini@unibs.it",
                  "body": "We have not to smoke too much...",
                  "created": datetime.datetime.now().strftime("%a %b %d %Y %H:%M:%S")
              }
          ],
          "tags": ["databases", "MongoDB", "awesome"],
          "rate": 5
        }
```

```
insertedDoc = mycol.insert_one(myPost)
insertedDoc.inserted_id
```

```
ObjectId('63834bff3ad739f508d34662')
```

# Inserire documenti in MongoDB (III)



```
newPosts = [{"title": "Message number 2",
  "slug": "msg-2",
  "body": "Nothing else to add...",
  "published": False,
  "created": datetime.datetime.now().strftime("%a %b %d %Y %H:%M:%S"),
  "updated": datetime.datetime.now().strftime("%a %b %d %Y %H:%M:%S"),
  "tags": ["databases", "MongoDB", "awesome"],
  "rate": 1},
  {"title": "Message number 3",
  "slug": "msg-3",
  "body": "Something else, maybe...",
  "published": True,
  "created": datetime.datetime.now().strftime("%a %b %d %Y %H:%M:%S"),
  "updated": datetime.datetime.now().strftime("%a %b %d %Y %H:%M:%S"),
  "tags": ["databases", "MongoDB", "awesome"],
  "rate": 3
}]
```

```
results = mycol.insert_many(newPosts)
results.inserted_ids
```

```
[ObjectId('63834c5e3ad739f508d34663'), ObjectId('63834c5e3ad739f508d34664')]
```





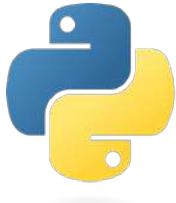
# Query in MongoDB (I)

Anche le query sono espresse sotto forma documentale, utilizzando i metodi `find()` e `find_one()`:

```
db.post.find({})
db.post.find ( {'published':true})
db.post.find ( {'rate':{$gt:4}})
db.post.find({'published':true,'rate':{$gt:4}})
db.post.find({$or:[{'published':true},{'rate':{$gt:4}}]})
Db.post.find({'rate':{$in:[3,5]}})
db.post.find({'comments.author':'Devis'})
db.post.find({'tags':'MongoDB'})
db.post.find({'tags':['MongoDB','databases']})
db.post.find({'tags':{$all:['MongoDB','databases']}})
db.post.find({'tags.0':'databases'})
db.post.find({'tags':{$size:3}})
db.post.find().sort({'rate':1}) or db.post.find().sort({'rate':-1})
db.post.find({'rate': {'$gt':4}}, {'title':1})
db.post.countDocuments( {...});
```



# Query in MongoDB (II)



```
x = mycol.find_one()
```

```
print(x)
```

```
{'_id': ObjectId('63834bff3ad739f508d34662'), 'title': 'Welcome to the marlboro country', 'slug': 'welcome-to-marlboro', 'body': 'Smoke smoke smoke...', 'published': 'true', 'created': 'Sun Nov 27 2022 12:37:06', 'updated': 'Sun Nov 27 2022 12:37:06', 'comments': [{'author': 'Devis', 'email': 'devis.bianchini@unibs.it', 'body': 'We have not to smoke too much...', 'created': 'Sun Nov 27 2022 12:37:06'}], 'tags': ['databases', 'MongoDB', 'awesome'], 'rate': 5}
```

```
for x in mycol.find():
```

```
    print(x)
```

```
{'_id': ObjectId('63834bff3ad739f508d34662'), 'title': 'Welcome to the marlboro country', 'slug': 'welcome-to-marlboro', 'body': 'Smoke smoke smoke...', 'published': 'true', 'created': 'Sun Nov 27 2022 12:37:06', 'updated': 'Sun Nov 27 2022 12:37:06', 'comments': [{'author': 'Devis', 'email': 'devis.bianchini@unibs.it', 'body': 'We have not to smoke too much...', 'created': 'Sun Nov 27 2022 12:37:06'}], 'tags': ['databases', 'MongoDB', 'awesome'], 'rate': 5}
{'_id': ObjectId('63834c5e3ad739f508d34663'), 'title': 'Message number 2', 'slug': 'msg-2', 'body': 'Nothing else to add...', 'published': 'false', 'created': 'Sun Nov 27 2022 12:39:06', 'updated': 'Sun Nov 27 2022 12:39:06', 'tags': ['databases', 'MongoDB', 'awesome'], 'rate': 1}
{'_id': ObjectId('63834c5e3ad739f508d34664'), 'title': 'Message number 3', 'slug': 'msg-3', 'body': 'Something else, maybe...', 'published': 'true', 'created': 'Sun Nov 27 2022 12:39:06', 'updated': 'Sun Nov 27 2022 12:39:06', 'tags': ['databases', 'MongoDB', 'awesome'], 'rate': 3}
```

# Aggiornamento di documenti

- Il meccanismo di aggiornamento di MongoDB permette di aggiornare singoli campi all'interno di un documento

```
db.collection.updateOne(<filter>,[<update>])
```

```
db.collection.updateMany(<filter>,[<update>])
```

```
db.collection.replaceOne(<filter>,[<update>])
```

- Sono disponibili diversi operatori, di cui vale la pena citare `$set`, `$unset`, `$addField`

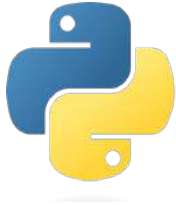


# Aggiornamento di documenti - MongoDB

```
localhost:27017 > mydatabase

76
77 db.post.updateMany({"published":true},[
78   {
79     $set : {
80       "published":false
81     }
82   }
83 ])
84
85 db.post.updateMany({},[
86   {
87     $addField : {
88       "newRate":{"multiply":["$rate",10]}
89     }
90   }
91 ])
92
93 db.post.updateMany({},[
94   {
95     $unset : ["newRate"]
96   }
97 ])
```

# Aggiornamento di documenti - PyMongo



```
mycol.update_many({"published":False},
                  [{
                    "$set":
                      {"published":True}
                  }])
```

```
<pymongo.results.UpdateResult at 0x1078c9310>
```

```
mycol.update_many({},
                  [{
                    "$addFields":
                      {"newRate":{"$multiply":["$rate",10]}}
                  }])
```

```
<pymongo.results.UpdateResult at 0x107b130a0>
```

```
mycol.update_many({},
                  [{
                    "$unset": ["newRate"]
                  }])
```

```
<pymongo.results.UpdateResult at 0x1078c92e0>
```

# Cancellazione di documenti

- Il meccanismo di rimozione di documenti in MongoDB si appoggia sui seguenti metodi (principali):

```
db.collection.deleteOne(<filter>)
```

```
db.collection.deleteMany(<filter>)
```

```
db.collection.remove(<filter>)
```



```
mycol.delete_one({"published":True})
```

```
<pymongo.results.DeleteResult at 0x107b13370>
```

```
mycol.delete_many({"slug":"msg-3"})
```

```
<pymongo.results.DeleteResult at 0x111d9b370>
```

# Validazione in MongoDB (I)

- Il meccanismo di validazione permette di imporre delle regole (diversamente rigide) sul salvataggio di documenti JSON nel database
- La validazione avviene attraverso delle regole espresse in JSON
- Le regole di validazione sono associate ad una collezione
- Le regole di validazione sono applicate quando:
  - un documento viene aggiunto alla collezione
  - un documento della collezione viene modificato



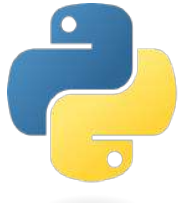
# Validazione in MongoDB (I)

```
db.createCollection("validatedPost",{
  validator:{
    $jsonSchema:{
      bsonType: "object",
      title: "Post Object Validation",
      required:["title","body","published","rate"],
      properties:{
        name:{
          bsonType:"string",
          description:"Una descrizione del campo opzionale"
        },
        body:{
          bsonType:"string",
          description:"Una descrizione del campo opzionale"
        },
        published:{
          bsonType:"bool",
          description:"Una descrizione del campo opzionale"
        },
        rate:{
          bsonType:"int",
          minimum:0,
          maximum:5,
          description:"Una descrizione del campo opzionale"
        }
      }
    }
  }
})
```





# Validazione in MongoDB (II)



```
mydb.create_collection("validatedPost", validator={
    "$jsonSchema":{
        "bsonType": "object",
        "title": "Post Object Validation",
        "required":["title","body","published","rate"],
        "properties":{
            "name":{
                "bsonType":"string",
                "description":"Una descrizione del campo opzionale"
            },
            "body":{
                "bsonType":"string",
                "description":"Una descrizione del campo opzionale"
            },
            "published":{
                "bsonType":"bool",
                "description":"Una descrizione del campo opzionale"
            },
            "rate":{
                "bsonType":"int",
                "minimum":0,
                "maximum":5,
                "description":"Una descrizione del campo opzionale"
            },
        },
    },
})
```