

# Baze de date NoSQL

Seminar 7

# Dezavantaje ale modelului relațional

- Modelul relațional are următoarele dezavantaje:
  - Nu este proiectat pentru a fi scalabil pe orizontală
  - Nepotrivirea structurilor de date *in-memory* folosite de către programatori în programarea aplicațiilor și modelul relațional (**impedance mismatch**)
  - Schema rigidă



# Baze de date NoSQL

- Bazele de date NoSQL (Not only SQL) au apărut ca răspuns la necesitatea de a stoca, gestiona și prelucra volume foarte mari de date
- Bazele de date din categoria NoSQL prezintă următoarele avantaje:
  - Sunt proiectate pentru a fi scalabile pe orizontală și pentru a funcționa într-un mediu distribuit
  - Schema flexibilă
  - Sunt în general open source
- Bazele de date NoSQL sunt relativ noi, diferă foarte mult între ele și evoluează foarte rapid (sunt diferențe semnificative de la o versiune la alta)
- Implementările NoSQL sunt în general specializate în rezolvarea anumitor tipuri de probleme

# Baze de date NoSQL - MongoDB

- MongoDB este o implementare NoSQL open source distribuită din categoria document
- MongoDB permite scalabilitate orizontală și disponibilitate ridicată cu ajutorul sharding-ului automat și al replicării datelor
- Datele sunt stocate într-un format similar cu JSON, numit BSON (Binary JSON)
- JSON (JavaScript Object Notation) este un format text de reprezentare a datelor
- JSON este ușor de citit și se folosește la transmiterea de date între aplicații
- Cele două structuri care stau la baza formatului JSON sunt obiectul (colecția de perechi cheie-valoare) și array-ul



# Baze de date NoSQL - MongoDB

- Exemplu de document JSON:

```
{“nume” : “Pop”,  
  “prenume” : “Bogdan”,  
  “adrese de email” : [ “popb@gmail.com” , “popbogdan@ymail.com” ],  
  “adresă” :  
    {“stradă” : “Macului”,  
      “număr” : 53,  
      “localitate” : “Sibiu”  
    }  
}
```

# Baze de date NoSQL - MongoDB

- Limbajul de interogare al MongoDB (MQL) este bazat pe JavaScript și este foarte expresiv (conține o mulțime de operatori)
- MongoDB este **case sensitive**
- Cea mai importantă unealtă de interacțiune cu o instanță MongoDB este consola Mongo (Mongo Shell) care este un interpretor JavaScript interactiv
- Mongo Shell este inclus în toate distribuțiile MongoDB
- MongoDB Compass este o interfață utilizator grafică pentru MongoDB
- MongoDB oferă posibilitatea de autentificare la nivelul serverului, dar nu creează în mod automat la instalare un utilizator pe server (acesta trebuie creat în mod explicit)



# Baze de date NoSQL - MongoDB

- MongoDB poate fi instalat pe:
  - Windows
  - Linux
  - macOS
- Cea mai nouă versiune stabilă este MongoDB 4.4.6 și poate fi descărcată de la următoarea adresă:

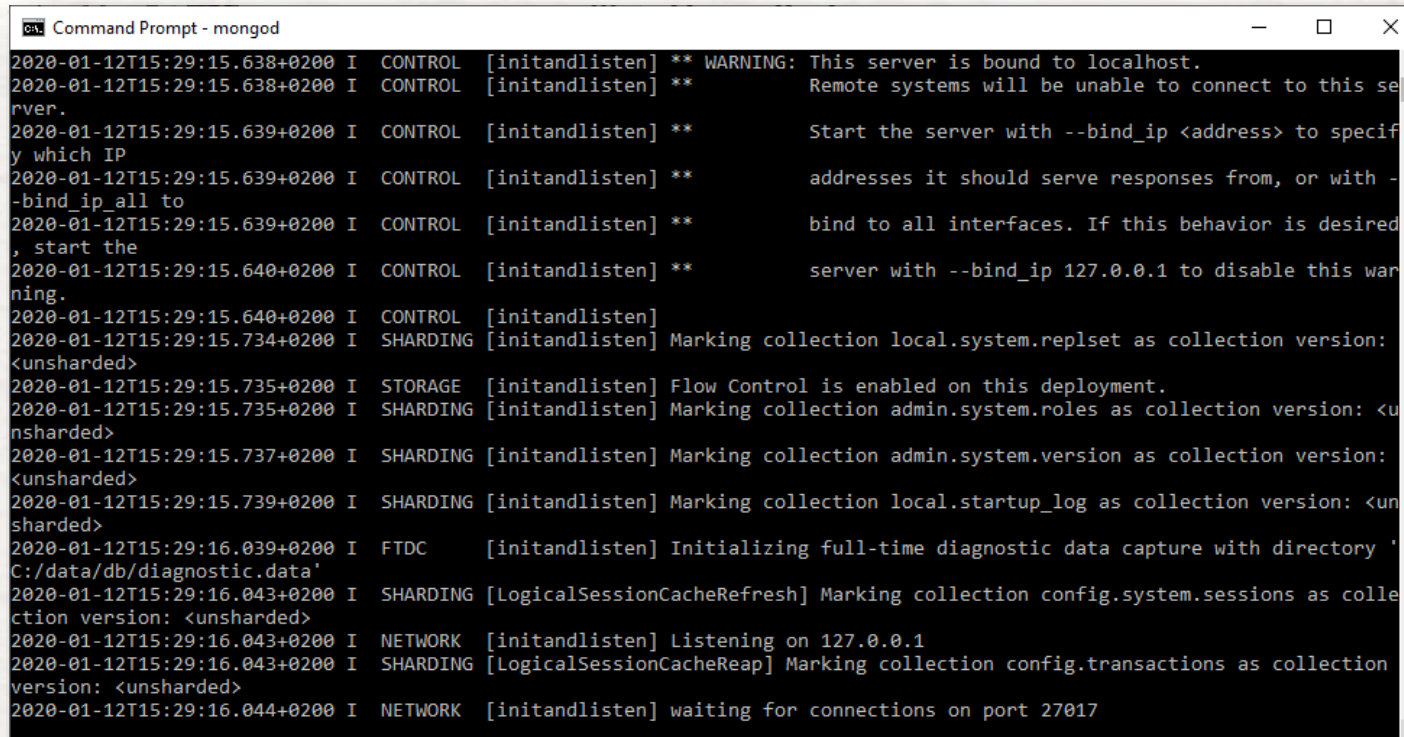
<https://www.mongodb.com/download-center/community>

- Cursuri gratuite oferite de MongoDB, Inc. sunt disponibile la următoarea adresă:

<https://university.mongodb.com/>

# Baze de date NoSQL - MongoDB

- Pentru a porni serverul MongoDB, deschidem o fereastră Command Prompt, scriem **mongod** și apăsăm Enter:

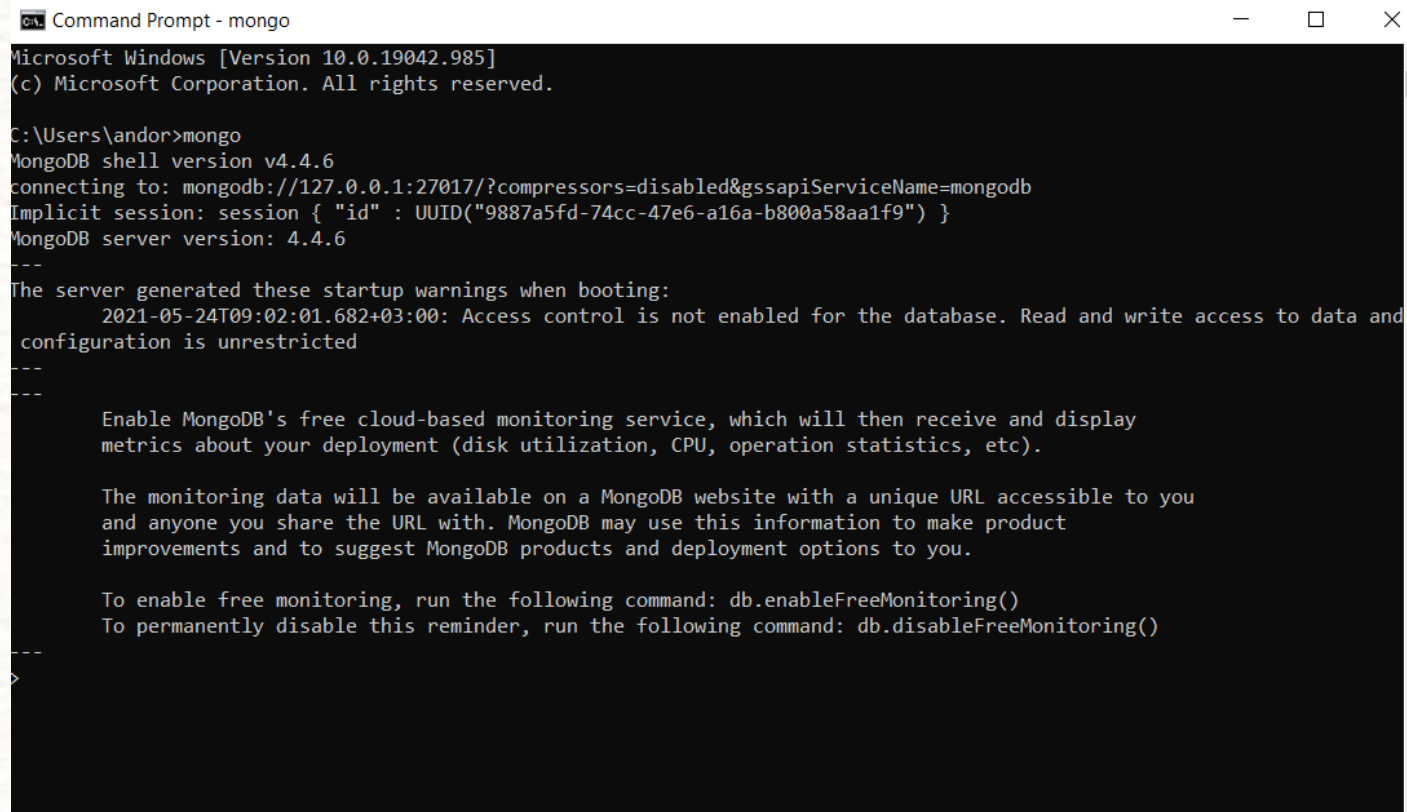


```
Command Prompt - mongod
2020-01-12T15:29:15.638+0200 I CONTROL [initandlisten] ** WARNING: This server is bound to localhost.
2020-01-12T15:29:15.638+0200 I CONTROL [initandlisten] ** Remote systems will be unable to connect to this server.
2020-01-12T15:29:15.639+0200 I CONTROL [initandlisten] ** Start the server with --bind_ip <address> to specify which IP
2020-01-12T15:29:15.639+0200 I CONTROL [initandlisten] ** addresses it should serve responses from, or with --bind_ip_all to
2020-01-12T15:29:15.639+0200 I CONTROL [initandlisten] ** bind to all interfaces. If this behavior is desired, start the
2020-01-12T15:29:15.640+0200 I CONTROL [initandlisten] ** server with --bind_ip 127.0.0.1 to disable this warning.
2020-01-12T15:29:15.640+0200 I CONTROL [initandlisten]
2020-01-12T15:29:15.734+0200 I SHARDING [initandlisten] Marking collection local.system.replset as collection version: <unsharded>
2020-01-12T15:29:15.735+0200 I STORAGE [initandlisten] Flow Control is enabled on this deployment.
2020-01-12T15:29:15.735+0200 I SHARDING [initandlisten] Marking collection admin.system.roles as collection version: <unsharded>
2020-01-12T15:29:15.737+0200 I SHARDING [initandlisten] Marking collection admin.system.version as collection version: <unsharded>
2020-01-12T15:29:15.739+0200 I SHARDING [initandlisten] Marking collection local.startup_log as collection version: <unsharded>
2020-01-12T15:29:16.039+0200 I FTDC [initandlisten] Initializing full-time diagnostic data capture with directory 'C:/data/db/diagnostic.data'
2020-01-12T15:29:16.043+0200 I SHARDING [LogicalSessionCacheRefresh] Marking collection config.system.sessions as collection version: <unsharded>
2020-01-12T15:29:16.043+0200 I NETWORK [initandlisten] Listening on 127.0.0.1
2020-01-12T15:29:16.043+0200 I SHARDING [LogicalSessionCacheReap] Marking collection config.transactions as collection version: <unsharded>
2020-01-12T15:29:16.044+0200 I NETWORK [initandlisten] waiting for connections on port 27017
```



# Baze de date NoSQL - MongoDB

- Pentru a ne conecta la server cu ajutorul consolei Mongo, deschidem o altă fereastră Command Prompt și scriem **mongo**:



```
Command Prompt - mongo
Microsoft Windows [Version 10.0.19042.985]
(c) Microsoft Corporation. All rights reserved.

C:\Users\andor>mongo
MongoDB shell version v4.4.6
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("9887a5fd-74cc-47e6-a16a-b800a58aa1f9") }
MongoDB server version: 4.4.6
---
The server generated these startup warnings when booting:
  2021-05-24T09:02:01.682+03:00: Access control is not enabled for the database. Read and write access to data and
configuration is unrestricted
---
---
  Enable MongoDB's free cloud-based monitoring service, which will then receive and display
metrics about your deployment (disk utilization, CPU, operation statistics, etc).

  The monitoring data will be available on a MongoDB website with a unique URL accessible to you
and anyone you share the URL with. MongoDB may use this information to make product
improvements and to suggest MongoDB products and deployment options to you.

  To enable free monitoring, run the following command: db.enableFreeMonitoring()
  To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
---
>
```

# Baze de date NoSQL - MongoDB

- Pentru a vedea toate bazele de date disponibile pe serverul MongoDB, executăm în Mongo Shell comanda **show dbs**:

```
> show dbs
admin          0.000GB
collations     0.001GB
local          0.000GB
m034           0.011GB
m034-compass   0.002GB
m201           0.113GB
startups       0.033GB
test           0.000GB
```

- Pentru a ne conecta la o anumită bază de date, executăm comanda:  
**use database\_name**
- În mod automat, suntem conectați la baza de date 'test'



# Baze de date NoSQL - MongoDB

- Pentru a vizualiza toate colecțiile din baza de date la care ne-am conectat, executăm comanda **show collections**:

```
> show collections
companies
people
peopleSectors
```

- Colecția de documente corespunde unui tabel dintr-o bază de date relațională, iar un document corespunde unei înregistrări dintr-un tabel relațional
- Totuși, colecția nu are o structură rigidă, iar documentul are o structură mai bogată decât o înregistrare, deoarece permite stocarea unor structuri de date complexe ca valori pentru chei

# Baze de date NoSQL - MongoDB

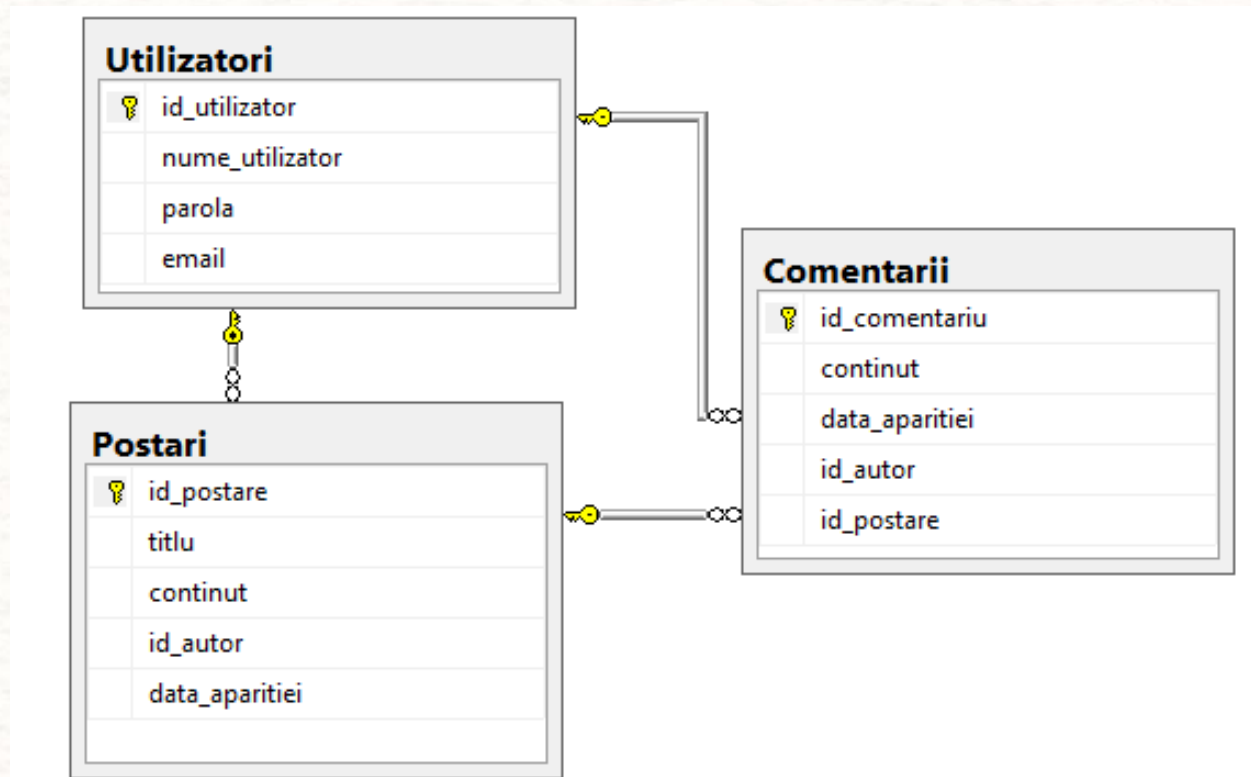
## Problemă

- Să se creeze o bază de date pentru o aplicație web de tip blog care stochează date despre utilizatori și postările acestora. Fiecare utilizator are un nume de utilizator, o parolă și o adresă de email. Fiecare postare are un titlu, un conținut, o dată a apariției, un autor și o listă de comentarii. Fiecare comentariu are un autor, un conținut și o dată a apariției. Un utilizator poate avea mai multe postări, dar postarea poate să aibă un singur autor. La fel, un comentariu poate avea un singur autor, dar un utilizator poate să posteze mai multe comentarii. Fiecare postare poate avea mai multe comentarii, dar fiecare comentariu aparține unei singure postări.



# Baze de date NoSQL - MongoDB

- Structura bazei de date din SQL Server



# Baze de date NoSQL - MongoDB

- În varianta relațională, baza de date ar conține trei tabele: *Utilizatori*, *Postări* și *Comentarii*
- Între *Utilizatori* și *Postări* avem o relație one to many (un utilizator poate avea mai multe postări, dar o postare poate avea un singur autor)
- Între *Postări* și *Comentarii* avem o relație one to many (o postare poate avea mai multe comentarii, dar un comentariu poate aparține unei singure postări)
- Între *Utilizatori* și *Comentarii* avem o relație one to many (un utilizator poate avea mai multe comentarii, dar un comentariu poate avea doar un singur autor)
- Mai avem o constrângere UNIQUE pe coloana “nume\_utilizator”, deoarece în general numele de utilizator este unic



# Baze de date NoSQL - MongoDB

- Script-ul SQL utilizat pentru crearea tabelelor:

```
USE Blog;
```

```
--Crearea tabelului Utilizatori
```

```
CREATE TABLE Utilizatori
```

```
(id_utilizator INT PRIMARY KEY IDENTITY,
```

```
nume_utilizator VARCHAR(50) UNIQUE,
```

```
parola VARCHAR(50),
```

```
email VARCHAR(100)
```

```
);
```

# Baze de date NoSQL - MongoDB

--Crearea tabelului Postari

CREATE TABLE Postari

(

id\_postare INT PRIMARY KEY IDENTITY,

titlu NVARCHAR(200),

continut NVARCHAR(MAX),

id\_autor INT FOREIGN KEY REFERENCES Utilizatori(id\_utilizator),

data\_aparitiei DATETIME

);



# Baze de date NoSQL - MongoDB

--Crearea tabelului Comentarii

```
CREATE TABLE Comentarii
```

```
(
```

```
id_comentariu INT PRIMARY KEY IDENTITY,
```

```
continut NVARCHAR(4000),
```

```
data_aparitiei DATETIME,
```

```
id_autor INT FOREIGN KEY REFERENCES Utilizatori(id_utilizator),
```

```
id_postare INT FOREIGN KEY REFERENCES Postari(id_postare)
```

```
);
```

# Baze de date NoSQL - MongoDB

## Structura bazei de date din MongoDB

- În cazul bazei de date MongoDB, schema este flexibilă și nu trebuie stabilită la crearea colecției de documente
- Fiecare document dintr-o colecție poate să aibă o structură diferită, deci așa-zisa structură a colecției este dată de documentele inserate în colecție
- Fiind vorba de o implementare NoSQL, nu suntem nevoiți să normalizăm datele, ci putem să ne gândim la modul în care vor fi utilizate datele și la cele mai frecvente interogări
- În MongoDB, în cazul unei relații one to many, avem două variante:
  - creăm o singură colecție și încorporăm documentele de pe partea de many în documentul de pe partea de one
  - creăm două colecții diferite și stocăm în fiecare document din colecția de pe partea de many o referință (valoarea câmpului “\_id” care are rol de identificador unic la nivel de colecție pentru fiecare document) la documentul care se află în colecția de pe partea de one



# Baze de date NoSQL - MongoDB

- În cazul problemei noastre, există trei relații one to many:
  - O relație one to many între *Utilizatori* și *Postări*
  - O relație one to many între *Utilizatori* și *Comentarii*
  - O relație one to many între *Postări* și *Comentarii*
- În cazul aplicației noastre, datele care sunt accesate împreună sunt grupate astfel:
  - O postare împreună cu toate comentariile
  - Toate postările unui anumit utilizator
- Pentru fiecare postare și fiecare comentariu, este necesar doar numele utilizatorului, nu și celelalte detalii despre acesta
- În aceste condiții, vor exista două colecții:
  - Colecția **utilizatori**
  - Colecția **postari**

# Baze de date NoSQL - MongoDB

- În MongoDB, fiecare document stocat conține în mod implicit câmpul “\_id”
- Câmpul “\_id” reprezintă identificatorul unic al fiecărui document la nivel de colecție (un fel de cheie primară a fiecărui document)
- MongoDB furnizează în mod automat o valoare unică la nivel de colecție pentru câmpul “\_id” în cazul în care la inserarea documentului nu se specifică o valoare pentru acesta
- Vom furniza valoarea numelui de utilizator câmpului “\_id” la inserare în colecția **utilizatori**, deoarece dorim ca numele utilizatorului să fie unic și îl vom folosi ca referință în colecția **postari**
- Pentru că la afișarea fiecărei postări și fiecărui comentariu este necesar doar numele utilizatorului care a creat comentariul sau postarea, putem folosi direct valoarea stocată ca referință spre documentul din colecția **utilizatori**, deoarece aceasta este exact numele utilizatorului



# Baze de date NoSQL - MongoDB

- Exemplu de document din colecția **utilizatori**:

```
{  
  "_id" : "Bob",  
  "parola" : "233567",  
  "email" : "bob@gmail.com"  
}
```

# Baze de date NoSQL - MongoDB

- Exemplu de document din colecția **postari**:

```
{ "_id" : ObjectId("5a53b2c51d2f5a15e12d3109"),  
  "titlu" : "O zi însorită",  
  "autor" : "Bob",  
  "continut" : "Într-o zi de vară târzie, stând pe terasă, am observat....."  
  "data_postarii" : ISODate("2018-01-08T18:04:00Z"),  
  "comentarii": [  
    { "autor" : "Tom",  
      "continut" : "Frumos spus ",  
      "data_com" : ISODate("2018-01-08T19:04:00Z")  
    } ]  
}
```



# Baze de date NoSQL - MongoDB

- În MongoDB nu există constrângeri sau relații între colecții
- Referințele la documente din altă colecție pe care le stocăm într-un document dintr-o colecție nu sunt verificate la nivelul bazei de date
- Nu există suport pentru JOIN-uri, deci dacă avem nevoie de JOIN-uri, acestea vor fi implementate la nivel de aplicație, nu la nivelul bazei de date (o variantă de JOIN există doar în cadrul framework-ului de agregare și este disponibil doar pentru colecții care se află în aceeași bază de date și care nu sunt distribuite)
- Datorită structurii JSON, în partea de valoare a unei perechi cheie-valoare se poate stoca o valoare simplă, un subdocument JSON, un array de valori simple sau de subdocumente JSON
- În cazul aplicației noastre, comentariile unei postări sunt stocate ca subdocumente într-un array asociat cheii (câmpului) “comentarii”

# Baze de date NoSQL - MongoDB

- Exemplu de inserare a unui utilizator în baza de date:

- Varianta SQL Server

```
INSERT INTO Utilizatori (nume_utilizator, parola, email)  
VALUES ('Bob', '233567', 'bob@gmail.com');
```

- Varianta MongoDB

```
db.utilizatori.insert({"_id":"Bob", "parola":"233567", "email":  
"bob@gmail.com"});
```

- Exemplu de inserare a unei postări în baza de date:

- Varianta SQL Server

```
INSERT INTO Postari (titlu, continut, id_autor, data_aparitiei)  
VALUES ('O zi însorită', 'Într-o zi de vară târzie, stând pe terasă, am  
observat.....', 1, GETDATE());
```



# Baze de date NoSQL - MongoDB

- Varianta MongoDB

```
db.postari.insert( {"titlu":"O zi însorită", "autor":"Bob",  
"continut":"Într-o zi de vară târzie, stând pe terasă, am observat.....",  
"data_postarii" : new Date()} )
```

- Exemplu de inserare a unui comentariu în baza de date:

- Varianta SQL Server

```
INSERT INTO Comentarii (continut, data_aparitiei, id_autor, id_postare)  
VALUES ('Frumos spus ', GETDATE(), 2, 1);
```

- Varianta MongoDB

```
db.postari.update({"titlu":"O zi însorită"},  
{$set:{"comentarii":[{"autor":"Tom", "continut":"Frumos spus",  
"data_com": new Date()}]}})
```

# Baze de date NoSQL - MongoDB

- Exemplu de interogare – returnarea tuturor utilizatorilor din baza de date
  - Varianta SQL Server

```
SELECT * FROM Utilizatori;
```

- Rezultat:

	id_utilizator	nume_utilizator	parola	email
1	1	Bob	233567	bob@gmail.com
2	2	Tom	243567	tom@gmail.com

- Varianta MongoDB

```
db.utilizatori.find().pretty()
```

- Rezultat:

```
{ "_id" : "Bob", "parola" : "233567", "email" : "bob@gmail.com" }  
{ "_id" : "Tom", "parola" : "243567", "email" : "tom@gmail.com" }
```



# Baze de date NoSQL - MongoDB

- Exemplu de interogare: returnarea tuturor postărilor care au titlul “O zi însorită” din baza de date, împreună cu toate comentariile lor

- Varianta SQL Server

```
SELECT * FROM Postari INNER JOIN  
Comentarii ON Postari.id_postare=Comentarii.id_postare  
WHERE titlu='O zi însorită';
```

- Rezultat:

	id_postare	titlu	continut	id_autor	data_aparitiei	id_comentariu	continut	data_aparitiei	id_autor	id_postare
1	1	O zi insorita	Într-o zi ...	1	2018-01-08 22:25:01.860	1	Frumos spus	2018-01-08 22:38:44.070	2	1

# Baze de date NoSQL - MongoDB

- Varianta MongoDB

```
db.postari.find({"titlu":"0 zi însorită"}).pretty()
```

- Rezultat:

```
{
  "_id" : ObjectId("5a53d95e1d2f5a15e12d310b"),
  "titlu" : "0 zi însorită",
  "autor" : "Bob",
  "continut" : "Într-o zi de vară târzie, stând pe terasă, am observat.....",
  "data_postarii" : ISODate("2018-01-08T20:49:34.655Z"),
  "comentarii" : [
    {
      "autor" : "Tom",
      "continut" : "Frumos spus",
      "data_com" : ISODate("2018-01-08T20:51:30.065Z")
    }
  ]
}
```



# Baze de date NoSQL - MongoDB

- Exemplu de actualizare a utilizatorului cu numele “Bob” și modificarea adresei de email în “bobt@gmail.com”

- Varianta SQL Server

```
UPDATE Utilizatori SET email='bobt@gmail.com' WHERE nume_utilizator='Bob';
```

- Rezultat:

	id_utilizator	nume_utilizator	parola	email
1	1	Bob	233567	bobt@gmail.com
2	2	Tom	243567	tom@gmail.com

- Varianta MongoDB

```
db.utilizatori.update({"_id":"Bob"}, {$set:{"email":"bobt@gmail.com"}})
```

- Rezultat:

```
{ "_id" : "Bob", "parola" : "233567", "email" : "bobt@gmail.com" }  
{ "_id" : "Tom", "parola" : "243567", "email" : "tom@gmail.com" }
```

# Baze de date NoSQL - MongoDB

- Exemplu de ștergere a utilizatorului cu numele “Tom” din baza de date

- Varianta SQL Server

```
DELETE FROM Utilizatori WHERE nume_utilizator='Tom';
```

- Varianta MongoDB

```
db.utilizatori.remove({"_id":"Tom"})
```

- În varianta SQL Server, ștergerea va eșua din cauza constrângerii FOREIGN KEY din tabelul *Comentarii* (există un comentariu al utilizatorului Tom)
- În varianta MongoDB, documentul corespunzător va fi șters din colecția **utilizatori**, dar comentariul lăsat de utilizatorul Tom rămâne stocat în documentul postării de care aparține (în MongoDB nu putem defini constrângeri)