

MongoDB

OBSAH

T1) Základy MongoDB, Modelovanie

T2) CRUD, Kurzory

T3) Agregácia, Indexy

T4) Replikácia a Sharding

Ciele

- Pochopiť kľúčové vlastnosti MongoDB
- Zvládnuť techniky a operácie na riešenie dopytovania a základných úloh MongoDB

T1) Základy MongoDB, Modelovanie

1a) MongoDB

1b) Inštalácia a jemný úvod – cmd MongoCompass vs mongosh.exe

1c) Mongo Shell

1d) Dokumentový model

1e) Modelovanie dát

1a) MongoDB

[MongoDB z hľadiska SQL](#)

[Úvod do MongoDB a dátového modelu, štruktúra dokumentu](#)

MongoDB (humongous [hju: 'mʌŋgəs] - obrovský)

MongoDB je systém na manažovanie **dokument**-orientovaných/dokumentových databáz, ktorý je viacplatformový (Linux, Windows, Mac), bezplatný, open-source a zaraďuje sa medzi NoSQL databázy.

MongoDB bol navrhovaný tak, aby poskytol vysoký výkon a automatickú škálovateľnosť. MongoDB dve najzákladnejšie požiadavky pri práci s big datami, **dostupnosť** a **rýchlosť**, dosahuje na úkor takých vlastností relačných DB, ako **konzistentnosť** a **transakcia**.

MongoDB	Db	kolekcia	dokument	kľúč/pole	vnorenie, odkazovanie	find
RDB	Db	tabuľka	Riadok	stĺpec	JOIN	select

Na rozdiel od relačných DB, ktoré využívajú relácie a tabuľky, MongoDB je založená na dokumentoch s **dynamickými schémami**. Práve táto vlastnosť, že dokumenty môžu mať premenlivú štruktúru, premenlivý počet polí/fieldov/key, uľahčuje integráciu rôznorodých dát.

Dokumenty pozostávajú z **key-value** párov, kľúč-hodnota. Na *key* sa odvoláva aj ako *field/pole*, ale my pojem *pole* necháme na pomenovanie polí, array (**field-pole, array-pole**).

Z dokumentov sa tvoria pomenované kolekcie - MongoDB ukladá dokumenty do kolekcií - databázy obsahujú kolekcie. Kolekcie zodpovedajú tabuľkám v relačných databázach. Na rozdiel od tabuľky, kolekcia nevyžaduje, aby jej dokumenty mali rovnakú schému. Na pridanie dokumentov do kolekcie MongoDB používa metódu `insert*()`. Ak sa pokúsime pridať dokumenty do kolekcie, ktorá neexistuje, MongoDB automaticky vytvorí kolekciu.

1b) Inštalácia a jemný úvod <https://www.mongodb.com/docs/manual/tutorial/install-mongodb-on-windows/>

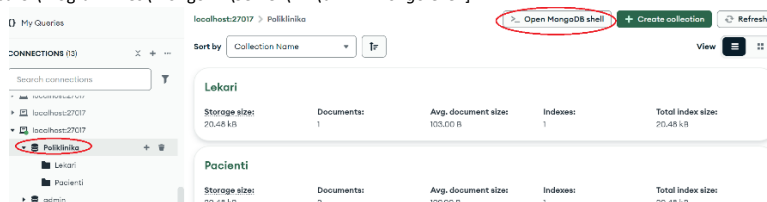
cmd: `mongod.exe` a potom `mongosh.exe` vs Mongo Compass

MongoDB Compass Connect => dole klepni na **_MONGOSH** (jeden príkaz) – Ctrl+L (vyčistí shell)

[V Mongo 4 server MongoDB a klient sa spúšťajú v dvoch príkazových riadkoch pomocou `mongod.exe` a `mongo.exe`.

Win + R -- cmd -- cd C:\Program Files\MongoDB\Server\4.4\bin -- `mongod.exe`

Win + R -- cmd -- cd C:\Program Files\MongoDB\Server\4.4\bin -- `mongo.exe`]



Zoznam všetkých DB získame príkazom

```
> show dbs
```

respektívne príkazom

```
> show databases
```

```
> show databases
blog 0.078GB
local 0.078GB
test 0.078GB
```

```
version() // uz nebudeme pisat znak: >
```

Poznamenáme, že príkazy sú case-sensitive (v Mongo Shell aj názvy DB objektov).

Ktorá DB je aktuálna zistíme príkazom

```
db
```

Aktuálnu databázu zastupuje `db`. Zoznam všetkých kolekcií aktuálnej databázy získame pomocou

```
show collections
```

Zoznam všetkých dokumentov v kolekcií *kolekciaNazov* vráti `find` (vs `findOne`)

```
db.kolekciaNazov.find( )
```

Novú databázu vytvoríme, resp. prepne sa na novú databázu, pomocou `use`. Po vykonaní

[pre vykonanie viac riadkov je lepšie použiť Mongo Shell - `mongosh.exe`, *hustejší výpis - do riadku viac atributov/kľúčov*, pozri nižšie (preto príkazy ukončujeme bodkočiarkou)]

```
use dbmaz1; //db.dropDatabase(); //db.repairDatabase( )
show databases;
```

v obdržanom zozname nenájde dbmaz1. Aby sme ju uvideli, musíme do nej vložiť kolekciu s dokumentami ({ meno : "Fero", vaha : 82 } je dokument s dvomi kľúčmi, atribútmi).

Kvôli kopírovaniu viac riadkov, ukončíme príkazy bodkočiarkou.

```
db.tab1.drop();
db.tab1.insertOne( { meno : "Fero", vaha : 82 } );
db.tab1.insertMany([ { meno : "Jano", vaha : 88 }, { meno : "Stevo", vaha : 88 } ]);
show databases;

db.tab1.find( ); // zoznam dokumentov;
db.tab1.findOne( );
db.tab1.findOne( {meno : "Jano"} );

show collections; // ⇔ // show tables; //// vratia stĺpcovy zoznam;
db.getCollectionNames( ); // vrati pole!!;
```

V skutočnosti DB a kolekcia sa vytvorí po insert . Ako zistíme počet dokumentov v kolekcii ukážeme nižšie.

1c) Mongo Shell (mongosh.exe; Compass a Robo 3T majú GUI)

Mongo Shell je interaktívne JavaScript rozhranie na dotazovanie a aktualizáciu dát pre MongoDB.

A) [Prehľad JavaScript](#) metód, funkcií.

Príkaz, definujúci premennú, sa nevytlačí, ak ho deklarujeme pomocou var - porovnaj dt a y.

```
11*11;

dt = new Date("2022/3/31");
dt.getFullYear(); // js
dt.toString(); //

var y=Math.cos(Math.PI); y;

// funguje v mongosh.exe:
function faktMDB (n) {
  if (n <= 1) return 1;
  return n * faktMDB(n - 1);
}
faktMDB(5);
```

B) [Mongo Shell metódy](#)

- [DB](#)
 - copyDatabase, cloneDatabase, cloneCollection
 - dropDatabase, createCollection
 - getCollection, getCollectionNames, getCollectionInfos, getName, runCommand, ...

- [Kolekcia](#)
 - insertOne, insertMany
 - find, findOne, distinct, findOneAndUpdate
 - update, updateOne, updateMany, replaceOne
 - drop, dropIndex
 - remove, deleteOne, deleteMany
 - validate
 - aggregate, count, group
 - mapReduce
 - ...
- [Kurzor](#) pre dopyt
 - count, min, max
 - limit, skip, size,
 - forEach, next, hasNext
 - map, sort, toArray, ...
- Ďalších 10 tém, ako Replication, Sharding

Príklad **agregácie count** (pozri aj tretiu prednášku k téme agregácia)

```
db.tab1.countDocuments() ; // kolekcia
// ⇔
db.tab1.find().count();      // kurzor + agregacia
db.tab1.find( { vaha: { $gt: 81 } } ).count();
db.tab1.find( ).sort({meno:-1});
```

1d) Dokumentový model

MongoDB ukladá záznamy o údajoch ako binárne BSON dokumenty.

- [Dokumenty](#)

- [BSON Typy](#)

[Dokumenty](#)

- Document Format
- Document Structure
- Field Names
- Field Value Limit
- Document Limitations
- The _id Field
- Dot Notation
- Additional Resources

Štruktúra dokumentu

```
{
  field1: value1,
  field2: { fieldA: valueA, fieldB: valueB}, // vnoreny dokument
  field3: [val1, val2],                     // pole hodnot
  ...
  fieldN: valueN
}
```

Jediná kolekcia **Autori** s poľom vnorených dokumentov kníh [{}, {}] namiesto dvoch kolekcií *Spisovatelia* a *Knihy*. Pre kolekciu **Novinari**, ktorí týždenne môžu publikovať viac článkov, sa tento model s objemným poľom sa menej hodí – radšej ďalšia kolekcia **Clanky**.

3

Kolekcia **Autori** s dvomi dokumentami, kde vnorené dokumenty tvoria pole.

```

use knihy;
db.dropDatabase();
use knihy;

db.Autori.insertMany(
[
  {
    meno: "Sano",
    adresa: "KE",
    dat_nar: "1980",
    knihy: [
      { nazov: "Financie",
        zaner: "Ekonomika",
        rok: 2005,
        cena: 45},
      { nazov: "Algoritmy",
        zaner: "PC",
        rok: 2010,
        cena: 40}]
  },
  {
    meno: "Imro",
    adresa: "AL",
    dat_nar: "1990",
    knihy: [
      { nazov: "RDBS",
        zaner: "PC",
        rok: 2005,
        cena: 45},
      { nazov: "NoSQL",
        zaner: "PC",
        rok: 2010,
        cena: 45},
      { nazov: " C# ",
        zaner: "PC",
        rok: 2016,
        cena: 50}]
  }
]
);

```

```
db.Autori.findOne();
```

```
db.Autori.find();
```

Validácia

Splnenie požiadaviek

```
db.Autori.validate( {
  $and: [
    {dat_nar: {$lte: 2016}},
    {$or: [
      {meno: { $type: "string" }},
      {adresa: { $type: "string" }}
    ]},
    {full:true}
  ]
});

...
},
"valid" :
"errors"
"ok" : 1
```

BSON Typy

MongoDB je založená na BSON/JSON dokumentoch s rozšíreným počtom typov a dynamickými schémami. **JSON** (pozri koniec prednášky o XML) ponúka iba šesť typov

numeric, string, boolean, array, object, null

BSON poskytuje ďalšie typy ako **int, long, double, date** alebo **javascript** a **regex**.

1e) Modelovanie dát

MongoDB **nepodporuje JOIN**. **Dáta** sú v ňom

- buď denormalizované, uložené spolu so súvisiacimi dátami ako **vnorené** dokumenty
- alebo normalizované, uložené v samostatných dokumentoch rôznych kolekcií s **odkazmi**.

Návrh dátových modelov

Dva základné dátové **modely** v MongoDB súvisia s (de)normalizovanými dátami

- modely s vnorenými dokumentami (denormalizované, 1 kolekcia)
- modely s referenciami medzi dokumentami (normalizované dátové modely, 2/viac kolekcií) .

Ako modelovať vzťah 1:N?

Pri navrhovaní schém v MongoDB je potrebné, aby sme upresnili svoj 1:N vzťah:

ide o 1:**málo**, 1:**veľa** alebo 1:**obrovské množstvo** ?

Podľa toho, o ktorý prípad ide, použijeme inú - inú schému.

Existujú tri základné spôsoby a dve vyspelejšie návrhové schémy

- 1) Modelovanie 1:málo
 - a. vložené *pole dokumentov* (1 kolekcia) resp.
 - b. vložené *pole odkazov* (2 kol.)
- 2) Modelovanie 1:veľa
 - a. (2 kolekcie) pole odkazov na potomka* alebo
 - b. vložené/vnorené **podmnožinové** dokumenty** + odkazy
- 3) Modelovanie 1:obrovské množstvo
- 4) Denormalizácia
- 5) Modelovanie stromovej štruktúry

*) budujeme databázu novinárov a ich článkov s dvomi **kolekciami** *Novinar*, *Clanok*
 - dokument v kol. *Novinar* má atribút *idClanok*: [1,2, ...] s odkazmi na dokument v kolekcii *Clanok*
 vs **lepšie** (chceme sa vyhnúť poliam s rastúcim počtom prvkov)
 - *Novinar* nemá atribút *idClanok*, ale každý článok má atribút *idNovinar*

**) sme zvedaví na kvalitu produktu, preto v kolekcii *Produkty*
 - každý produkt obsahuje všetky ohodnotenia, z ktorých sa zobrazuje 10 posledných
 slabšie vs **lepšie**
 - každý produkt obsahuje **10 posledných ohodnotení** + kolekcia všetkých ohodnotení s odkazom na produkt

Základné spôsoby

1) Modelovanie 1:málo - vložené pole dokumentov

- osoby a adresy

- **autori s knihami** (pozri vyššie) <https://docs.mongodb.org/manual/tutorial/model-embedded-one-to-many-relationships-between-documents/#data-modeling-example-one-to-many>
 vložíme knihy do poľa vnútri objektu/dokumentu autor.

- výhodou je, že nemusíme vykonávať ďalší dopyt na získanie vložených údajov
- nevýhodou je, že nie je štandardný spôsob, ako pristupovať k jednotlivým vloženým údajom ako samostatné objekty.

2) Modelovanie 1:veľa – pole odkazov

- náhradné súčiastky **produktu**

- lekári **pacienta** <https://docs.mongodb.org/manual/tutorial/model-referenced-one-to-many-relationships-between-documents/>

- pacienti **lekára** (napr. dva prípady: - pacient má menej lekárov < 100
 - lekár má viac pacientov < 2000) :

```
use Poliklinika;
db.dropDatabase();
use Poliklinika;

db.Pacienti.insertMany(
[
  {
    _id: 'AAAAA',
    rodneCislo: '12345678',
    meno: 'Pac1',
    vaha: 87,
    dat_navstevy: ISODate("2016-02-27T08:31:42.273Z"),
    poplatok: '15'
  },
  {
    _id: 'AAAAB',
```

```
    rodneCislo: '12345679',
    meno: 'Pac2',
    vaha: 70,
    dat_navstevy: ISODate("2016-02-27T08:55:45.214Z"),
    poplatok: '11'
  }
];
```

```
// Lekari;
db.Lekari.insert(
{
  name : 'Imro',
  spec : 'zubar',
  kod: 1234,
  pacienti : ['AAAAA', 'AAAAB'] // pole odkazov, referencii na pacienta
}
);
```

```
db.Pacienti.findOne();
```

```
// Vrátí dokument lekára na základe kódu;
var lekar = db.Lekari.findOne({kod: 1234});
lekar
{
  "_id" : ObjectId("570eb08fff88a5e44573cdeb"),
  "name" : "Imro",
  "spec" : "zubar",
  "kod" : 1234,
  "pacienti" : [
    "AAAAA",
    "AAAAB"
  ]
};
```

```
// Vrátí všetkých pacientov toho lekára;
var lekarove_pac = db.Pacienti.find({_id: { $in : lekar.pacienti } }
).toArray();
lekarove_pac
[
  {
    "_id" : "AAAAA",
    "rodneCislo" : "12345678",
    "meno" : "Pac1",
    "vaha" : 87,
    "dat_navstevy" : ISODate("2016-02-27T08:31:42.273Z"),
    "poplatok" : "15"
  },
  {
    "_id" : "AAAAB",
    "rodneCislo" : "12345679",
```

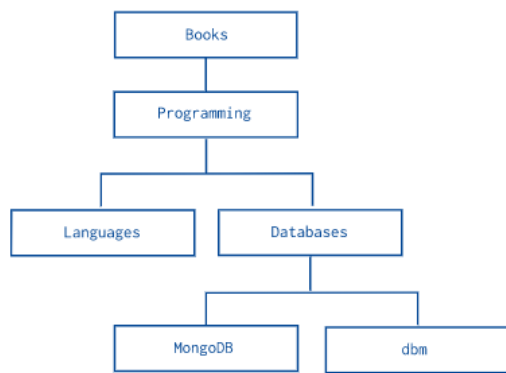
```

    "meno" : "Pac2",
    "vaha" : 70,
    "dat_navstevy" : ISODate("2016-02-27T08:55:45.214Z"),
    "poplatok" : "11"
  }
];

```

- Výhodou tohto modelu je, že Lekari a Pacienti sú samostatné kolekcie a tak dopytovať a modifikovať ich je ľahšie.
- Nevýhodou je, že k získaniu pacientov lekára je nutné vykonať ďalší dopyt.

Modelovanie stromovej štruktúry



<https://docs.mongodb.com/manual/tutorial/model-tree-structures-with-child-references/>

Odkaz na rodiča

```

db.categories.insertMany( [
  { _id: "MongoDB", parent: "Databases" },
  { _id: "dbm", parent: "Databases" },
  { _id: "Databases", parent: "Programming" },
  { _id: "Languages", parent: "Programming" },
  { _id: "Programming", parent: "Books" },
  { _id: "Books", parent: null }
] )

```

Odkaz na potomka

```

db.categories.insertMany( [
  { _id: "MongoDB", children: [] },
  { _id: "dbm", children: [] },
  { _id: "Databases", children: [ "MongoDB", "dbm" ] },
  { _id: "Languages", children: [] },
  { _id: "Programming", children: [ "Databases", "Languages" ] },
  { _id: "Books", children: [ "Programming" ] }
] )

```

Ďalšie zdroje

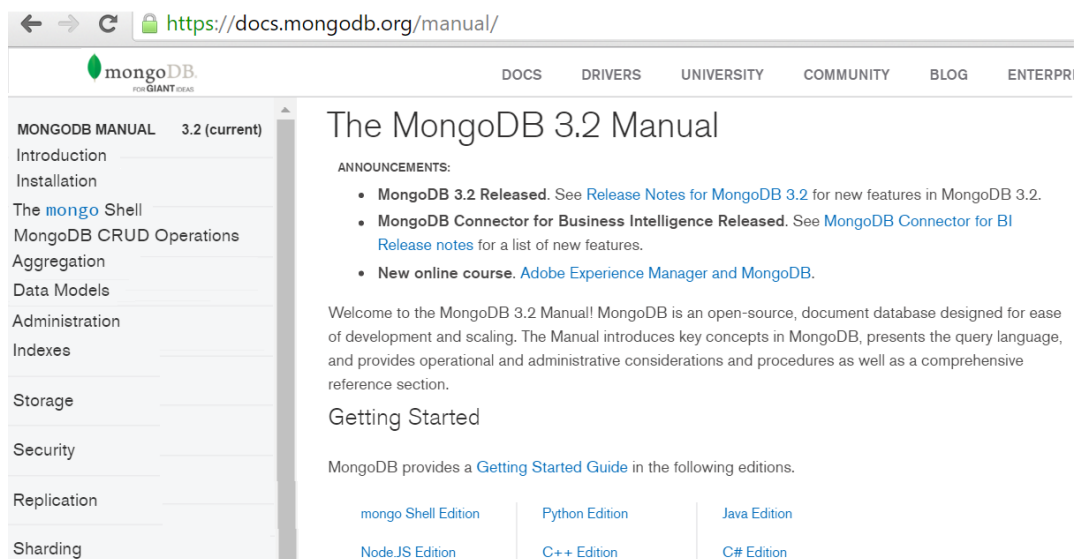
Pripomíname, že pri výklade základ MongoDB postupujeme podľa obsahu

- **T1) Základy a princípy MongoDB, Mongo Shell** – táto prvá prednáška
- **T2) CRUD, Indexy**
- **T3) Agregácia, Kurzory**
- **T4) Replikácia a Sharding**

Všetky učebnice, prednášky o MongoDB čerpajú, aj my, zo zdrojov oficiálnej stránky MongoDB, ktorá ponúka dve členenia výkladu materiálu:

Členenie na základe manuálu	Členenie na báze referencií
-----------------------------	-----------------------------

MongoDB 3.2 Manual Getting Started with MongoDB (MongoDB Shell Edition) Getting Started with MongoDB (C#)	Pojmy a koncepty MongoDB
	Referencie - Prehľad pojmov a konceptov MongoDB. - Komponenty MongoDB balíka - <i>Operátory</i> - Databázové príkazy v tvare <code>db.runCommand({ . . . })</code> - Mongo Shell metódy



[The bios Example Collection](#)

[Tutorial](#)