

UNIVERZITA HRADEC KRÁLOVÉ

Fakulta informatiky a managementu

MongoDB

Karolína Tomášová, první ročník

V Hradci Králové
dne 19.5.2025

Seminární práce z předmětu NoSQL databáze

Obsah

Úvod	5
Architektura.....	6
1.1 Schéma a popis architektury	6
1.2 Specifika konfigurace	6
1.2.1 CAP teorém	6
1.2.2 Cluster.....	8
1.2.3 Uzly	9
1.2.4 Sharding.....	10
1.2.5 Replikace	10
1.2.6 Perzistence dat	10
1.2.7 Distribuce dat.....	11
1.2.8 Zabezpečení	12
2 Funkční řešení.....	14
2.1 Struktura	14
2.1.1 Docker-compose.yml.....	15
2.2 Instalace	19
2.2.1 Windows – předpoklady pro WSL	19
2.2.2 Instalace Pythonu a potřebných knihoven	19
2.2.3 Inicializace clusteru a import dat.....	19
3 Případy užití a případové studie	20
3.1 Účely užití MongoDB	20
3.1.1 Zdravotnické aplikace a elektronické zdravotní záznamy.....	20
3.1.2 Analýza a reporting dat s nutností rychlých dotazů nad velkými objemy...	20
3.1.3 3. Vývoj prototypů a agilní vývoj:.....	20
3.2 Účel projektu	20
3.3 Proč byla zvolena MongoDB	21
3.4 Případové studie	21
3.4.1 Novo & MongoDB Atlas.....	21
3.4.2 Enpal: Řízení IoT dat v oblasti udržitelné energetiky	22
3.4.3 Vainu: Dynamická práce s firemními daty.....	22
4 Výhody a nevýhody	23
4.1 Výhody a nevýhody MongoDB jako NoSQL databáze	23

Seminární práce z předmětu NoSQL databáze

4.2	Výhody a nevýhody vypracovaného řešení.....	23
5	Další specifika	24
6	Data.....	24
6.1	Data sety	26
6.1.1	Patients	26
6.1.2	Procedures	26
6.1.3	Medication	27
6.2	Grafická vizualizace dat	28
7	Dotazy	29
7.1	Agregační funkce.....	29
7.1.1	Počet předepsaných léků na každého pacienta	29
7.1.2	Průměrná cena zákroku podle kódu.....	30
7.1.3	Kolik pacientů podstoupilo zákroky v každém roce?.....	30
7.1.4	Celkové náklady na léky podle pojišťovny (PAYER)	31
7.1.5	Počet unikátních léků, které byly předepsány	32
7.1.6	Průměrné HEALTHCARE_EXPENSES pacientů podle pohlaví (GENDER) 32	
7.2	Embadded dokumenty	34
7.2.1	Vypsát seznam všech pacientů s polem fullName.....	34
7.2.2	Odebrání zbytečných polí (např. interní ENCOUNTER)	34
7.2.3	Zákroky, které měly "Suspected COVID-19" jako důvod	35
7.2.4	Vyhledání pacientů podle rozsahu data narození	36
7.2.5	Top 5 nejčastěji předepisovaných léků podle názvu	37
7.2.6	Vložení embedded dokumentu do dokumentu patients.....	38
7.3	Práce s daty – Insert, Update, Delete, Merge	39
7.3.1	Vložení nového pacienta	39
7.3.2	Aktualizace příjmení pacienta	40
7.3.3	Smazání pacienta s konkrétním ID	40
7.3.4	Zvýšení počtu výdejů léku o 1	41
7.3.5	Přidání nového pole k dokumentům s hodnotou NULL.....	41
7.3.6	Vytvoření nové kolekce s daty původní kolekce procedures a patients	42
7.4	Indexy a výkon	43
7.4.1	Vytvoření složeného indexu pro pacienty a diagnózu.....	43

Seminární práce z předmětu NoSQL databáze

7.4.2	Textový index na diagnózy.....	43
7.4.3	TTL index pro automatické mazání záznamů o medikaci.....	43
7.4.4	Analýza vhodnosti shard klíče.....	43
7.4.5	Použití hint() pro vynucení indexu	44
7.4.6	Vyhodnocení dotazu pomocí explain().....	45
7.5	Cluster a konfigurace.....	50
7.5.1	Stav shardingu v clusteru.....	50
7.5.2	Rozložení dat mezi shardy.....	57
7.5.3	Stav balanceru.....	59
7.5.4	Informace o běžících routerech	60
7.5.5	Ověření chunků a jejich rozložení	60
7.5.6	Stav všech mongos routerů v clusteru	62
Závěr		63
Zdroje		64
Přílohy		66

Seminární práce z předmětu NoSQL databáze

Úvod

Tato práce se zaměřuje na praktické využití databázového systému MongoDB při zpracování zdravotnických záznamů – konkrétně dat o pacientech, jejich medikaci a lékařských zákrocích. MongoDB bylo zvoleno zejména díky své flexibilitě při práci s dokumentově orientovanými daty, podpoře horizontálního škálování, vysoké dostupnosti prostřednictvím replikací a možnosti shardingu, což z něj činí ideální nástroj pro moderní aplikace pracující s rozsáhlými a variabilními datovými sadami.

Součástí práce je kompletní návrh a implementace databázového clusteru, jehož architektura je v dokumentu detailně popsána. Přehledně je zde zachycen proces inicializace prostředí, konfigurace jednotlivých komponent – shardů, replik, konfiguračních serverů a routerů – a také postup spuštění databáze pomocí shell skriptu.

Do systému jsou následně importována synteticky vygenerovaná zdravotnická data ve formátu CSV, přičemž je kladen důraz na jejich základní předzpracování – odstranění duplicitních záznamů, vyčištění chybějících hodnot a převod do vhodné struktury. Tyto kroky jsou realizovány v jazyce Python s využitím knihoven jako pandas, matplotlib a seaborn, které zároveň slouží k základní vizualizaci klíčových vlastností dat.

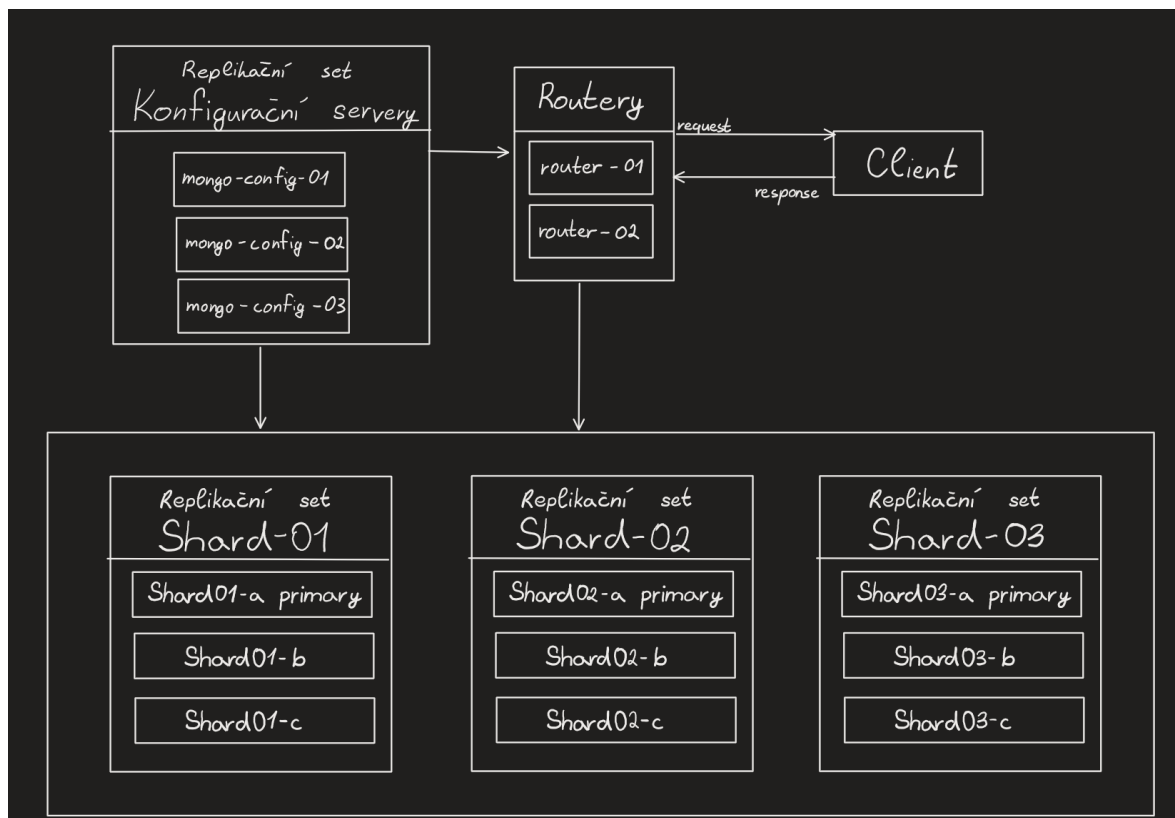
Hlavním přínosem práce je propojení teoretických znalostí s praktickými zkušenostmi a vytvoření projektu, který může sloužit jako srozumitelný úvod a inspirace pro ty, kteří se chtějí seznámit s databázovým systémem MongoDB na reálném a praktickém příkladu.

Seminární práce z předmětu NoSQL databáze

Architektura

1.1 Schéma a popis architektury

Vypracované řešení je postaveno jako distribuovaný MongoDB cluster, který se dělí do několika typů kontejnerů. Toto celé pak společně tvoří plno hodnotovou škálovatelnou databázi. Architektura byla vytvořena na základě požadavků projektu, což konkrétně bylo: zajištění shardingu a replika setů, a aby architektura měla minimálně 3 uzly. Dalším aspektem, který byl při tvorbě zvažován bylo jednoduché spuštění pomocí inicializačního skriptu.



1.2 Specifika konfigurace

V této kapitole budou představeny a podrobně vysvětleny jednotlivé specifikace konfigurace.

1.2.1 CAP teorém

Vypracovaná databáze prioritizuje v rámci CAP teorému variantu **CP** neboli konzistenci (Consistency) a odolnost vůči výpadkům (Partition Tolerance). Tato volba vyplývá ze způsobu konfigurace replikace a nastavení parametrů zápisu.

Write concern: <https://www.mongodb.com/docs/manual/reference/write-concern/>

Konzistence je v databázi zajištěna prostřednictvím atributu „writeConcern“, který definuje, kolik replik musí potvrdit operaci zápisu, aby byla považována za úspěšnou. Na konkrétním

Seminární práce z předmětu NoSQL databáze

případě, kdy se používá model PSS, se tak zápis musí dostat na primární a minimálně jeden sekundární uzel. Konkrétní nastavení je vidět na přiloženém obrázku.

```
> db.adminCommand({ getDefaultRWConcern: 1 })
< {
  defaultReadConcern: { level: 'local' },
  defaultWriteConcern: { w: 'majority', wtimeout: 0 },
  defaultWriteConcernSource: 'implicit',
  defaultReadConcernSource: 'implicit',
  localUpdateWallClockTime: 2025-05-18T11:33:52.367Z,
  ok: 1,
  '$clusterTime': {
    clusterTime: Timestamp({ t: 1747568032, i: 3 }),
    signature: {
      hash: Binary.createFromBase64('NankcH0Bltf0xcFEYswFdY9t40s=', 0),
      keyId: Long('7505715543173758999')
    }
  },
  operationTime: Timestamp({ t: 1747568032, i: 3 })
}
```

Atribut „wtimeout: 0“ navíc zajišťuje, že databáze čeká neomezeně dlouho, dokud není writeConcern splněn – tedy nevrátí výsledek, dokud není dosaženo konzistence.

Odolnost k výpadkům je realizována pomocí replikačních sad. V návrhu databázi je použit model PSS – jeden primární a dva sekundární uzly. Pokud dojde k výpadku primárního uzlu (např. z důvodu výpadku sítě či hardwarového selhání), ostatní členové repliky automaticky zahájí volební proces a zvolí nový primární uzel. Typicky dojde k zotavení clusteru během 5–10 sekund. Původní primární uzel se po návratu stává sekundárním. Pokud se synchronizuje a zůstává v dobrém stavu, může být v budoucnu zvolen znovu jako primární. Tuto možnost dostane, pokud nastane nová volební situace – např. výpadek primárního uzlu.

```
JS init-shard01.js x
scripts > JS init-shard01.js > ...
1  #!/bin/bash
2
3  mongosh <<EOF
4  var config = {
5    "_id": "rs-shard-01",
6    "version": 1,
7    "members": [
8      {
9        "_id": 0,
10       "host": "shard01-a:27017",
11       "priority": 1
12     },
13     {
14       "_id": 1,
15       "host": "shard01-b:27017",
16       "priority": 0.5
17     },
18     {
19       "_id": 2,
20       "host": "shard01-c:27017",
21       "priority": 0.5
22     }
23   ]
24 };
25 rs.initiate(config, { force: true });
26 EOF
27
```

Seminární práce z předmětu NoSQL databáze

Dostupnost dat (Availability) není v tomto řešení upřednostněna, protože byla kladena vyšší priorita na konzistenci dat. Vzhledem k tomu, že se jedná o zaměstnanecké záznamy, je konzistence považována za důležitější než okamžitá dostupnost dat.

1.2.2 Cluster

Ve vypracované databázi je použit jeden cluster, který se skládá ze tří shardů, přičemž každý shard tvoří replikační sadu (replica set) ve struktuře PSS (jeden primární a dva sekundární uzly).

Volba jednoho clusteru byla záměrná, protože je schopen efektivně obsloužit všechny datasety prostřednictvím shardování (detailněji popsáno v následující kapitole). Použití více clusterů by v tomto případě přineslo zbytečnou složitost při správě a komunikaci mezi clustery, což není vzhledem k velikosti datasetů (cca 150 000 záznamů) potřebné ani výhodné.

Pro zprovoznění clusteru byly využity následující konfigurace:

1.2.2.1 Konfigurace serveru

Konfigurační servery představují klíčovou součást MongoDB clusteru, jelikož ukládají metadata, což jsou informace o tom, jak jsou data v rámci clusteru rozložena. Metadata například určují, na kterém shardu se konkrétní data nacházejí a jak je třeba směřovat dotazy nebo zápisy.

Pro zajištění vysoké dostupnosti a odolnosti vůči výpadkům jsou ve vypracované databázi nasazeny celkem 3 konfigurační servery. Jeden funguje jako primární uzel, který zajišťuje zápis změn do metadat. Zbylé 2 uzly fungují jako sekundární, které primárně replikují zapsaná data z primárního uzlu a v případě výpadku mohou primární uzel zastoupit. Konkrétní konfigurace je vypracována v souboru init-configserver.js a její obsah je vidět na přiloženém obrázku:



```
JS init-configserver.js X
scripts > JS init-configserver.js > ...
1  #!/bin/bash
2
3  mongosh <<EOF
4  var config = {
5    "_id": "rs-config-server",
6    "configsvr": true,
7    "version": 1,
8    "members": [
9      {
10       "_id": 0,
11       "host": "configsvr01:27017",
12       "priority": 1
13     },
14     {
15       "_id": 1,
16       "host": "configsvr02:27017",
17       "priority": 0.5
18     },
19     {
20       "_id": 2,
21       "host": "configsvr03:27017",
22       "priority": 0.5
23     }
24   ]
25 };
26 rs.initiate(config, { force: true });
27 EOF
28
```


Seminární práce z předmětu NoSQL databáze

1.2.2.2 Konfigurace shardů

Po inicializaci jednotlivých replikačních sad pro každý shard je potřeba tyto shardy zaregistrovat do samotného MongoDB clusteru. K tomu slouží mongos, což je query router, který přijímá klientské požadavky a směřuje je na příslušné shardy v clusteru.

Registrace shardů do clusteru se provádí pomocí příkazů `sh.addShard()`, které přidávají každý shard podle jeho replikačního setu a jednotlivých členů (uzlů). Tato konfigurace se provádí v souboru `init-router.js` a její obsah je zobrazen na přiloženém obrázku:

```
JS init-router.js X
scripts > JS init-router.js
1 sh.addShard("rs-shard-01/shard01-a:27017")
2 sh.addShard("rs-shard-01/shard01-b:27017")
3 sh.addShard("rs-shard-01/shard01-c:27017")
4 sh.addShard("rs-shard-02/shard02-a:27017")
5 sh.addShard("rs-shard-02/shard02-b:27017")
6 sh.addShard("rs-shard-02/shard02-c:27017")
7 sh.addShard("rs-shard-03/shard03-a:27017")
8 sh.addShard("rs-shard-03/shard03-b:27017")
9 sh.addShard("rs-shard-03/shard03-c:27017")
10
11
```

1.2.3 Uzly

V databázi je použito celkem 9 uzlů, rozdělených do tří replikačních setů. Každý replikační set tvoří jeden shard, který obsahuje 3 uzly rozděleny podle principu PSS – jeden primární a dva sekundární uzly.

Použití minimálně tří uzlů v rámci každého replikačního setu je zvoleno z důvodu zajištění vysoké dostupnosti a odolnosti vůči výpadkům. Pokud by jeden uzel selhal, sekundární uzly dokážou převzít roli primárního a systém pokračuje v provozu bez výpadků či ztráty dat. Primární uzel je zodpovědný za zápisy, zatímco sekundární uzly udržují aktuální kopie dat a slouží jako záloha. Také umožňují provádět čtení s nižší zátěží primárního. Replikační sada a konkrétní uzly jsou specifikovány v souboru `init-shard0x.js`. Celkem jsou tyto soubory tři – pro každý shard jeden.

```
JS init-shard01.js X
scripts > JS init-shard01.js > ...
1 #!/bin/bash
2
3 mongosh <<EOF
4 var config = {
5   "_id": "rs-shard-01",
6   "version": 1,
7   "members": [
8     {
9       "_id": 0,
10      "host": "shard01-a:27017",
11      "priority": 1
12     },
13     {
14       "_id": 1,
15       "host": "shard01-b:27017",
16       "priority": 0.5
17     },
18     {
19       "_id": 2,
20       "host": "shard01-c:27017",
21       "priority": 0.5
22     }
23   ]
24 };
25 rs.initiate(config, { force: true });
26 EOF
27
```

Seminární práce z předmětu NoSQL databáze

1.2.4 Sharding

V databázi jsou použity celkem tři shardy, kdy je každý z nich tvořen replikační sadou, která obsahuje tři uzly.

Volba tří shardů byla vybrána kvůli 3 rozdílným datasetům a zároveň odpovídá i velikosti a charakteru zpracovaných dat. Vyšší počet nebyl zvolen z důvodu jejich dostatečné efektivity v problematice rozdělení dat a škálovatelnosti systému bez zbytečné komplikace správy. Zároveň pro danou databázi poskytují dostatečnou kapacitu a redundanci, což zaručuje vysokou dostupnost a rychlou odezvu systému. Sharding umožňuje rozdělit data do menších, lépe zpracovatelných částí, což napomáhá ke zlepšení výkonu při zápisu a čtení. Zároveň to také snižuje zátěž na jednotlivé servery.

Pokud by bylo v budoucnu potřeba databázi rozšířit, tak je možné přidat další shardy.

Konfigurace jednotlivých shardů je vidět na obrázku z předchozí kapitoly

1.2.5 Replikace

V rámci databázového řešení jsou využívány celkem tři replikační sady – každá replikační sada tvoří jeden shard.

Každá replikační sada obsahuje tři uzly (jeden primární a dva sekundární), čímž je dosaženo požadované vysoké dostupnosti, odolnosti vůči výpadkům a zajištění konzistence dat. Tato konfigurace odpovídá tzv. PSS struktuře (Primary–Secondary–Secondary).

Replikace umožňuje:

- Automatické převzetí role primárního uzlu při jeho výpadku (failover)
- Zajištění záloh bez přerušení provozu
- Rozložení čtecí zátěže mezi více uzlů
- Udržení aktuálních kopií dat pro případ havárie

Počet tří replikačních sad odpovídá rozdělení dat do tří shardů. Každý shard má vlastní nezávislou replikační logiku, což zvyšuje spolehlivost celého systému. Tato úroveň replikace je pro daný objem dat zcela dostačující a zároveň umožňuje jednoduché škálování do budoucna.

Konkrétní konfigurace replikace je uvedena v souborech init-shard01.js, init-shard02.js a init-shard03.js.

1.2.6 Perzistence dat

Ve vypracovaném řešení je perzistence dat zajištěna na několika úrovních, přičemž některé z nich jsou konfigurovány přímo v MongoDB.

1.2.6.1 Replikace

Perzistence dat je v tomto řešení podpořena především replikací dat napříč více uzly, což bylo více popsáno v předchozích kapitolách.

Seminární práce z předmětu NoSQL databáze

1.2.6.2 Zápís a čtení

Zápís dat je primárně směřován na primární uzly jednotlivých replikačních sad. Sekundární uzly pak slouží jako záloha, protože průběžně replikují všechna data z primárního uzlu.

V případě dotazů – čtení může být systém nakonfigurován tak, aby využíval i sekundární uzly, čímž se snižuje zátěž na primární uzel a zvyšuje se výkonost systému. Toto se dá nastavit pomocí změny Read preference, ovšem v tomto řešení bude čtení probíhat pouze z primárního uzlu, protože databáze není natolik rozsáhlá, aby využití sekundárních uzlů přineslo znatelné zvýšení výkonu.

MongoDB využívá operační paměť (RAM) jako cache pro často přístupná data, čímž urychluje čtení. Pro trvalé uložení jsou data zapisována do souborového systému na disku.

1.2.6.3 Ukládání a načítání dat

MongoDB implicitně využívá journaling, což znamená, že každý zápis je nejprve zaznamenán do interního logu před tím, než je trvale uložen na disk. Tento mechanismus chrání data při neočekávaném výpadku napájení nebo systému. V databázi je journaling povolen ve výchozím nastavení databáze.

1.2.7 Distribuce dat

V tomto projektu je využita architektura sharodavného clusteru MongoDB, která umožňuje horizontální škálování a eektivní rozdělení zátěže při práci s rozsáhlými daty. Databázová vrstva je vavržena jako sharded cluster se třemi shardy, jedním konfiguračním serverem a jedním routerem.

Konkrétní obrázky jednotlivých skriptů jsou přiloženy v příslušných předchozích kapitolách.

1.2.7.1 Sharding

Distribuce dat je realizována pomocí shardingu – každé kolekci je přiřazen tzv. shard key, podle kterého MongoDB automaticky rozděluje data mezi jednotlivé shard uzly. V tomto řešení byl využit shard key patient_id, který je obsažen napříč všemi třemi datovými sadami.

Sharding byl inicializován pomocí skriptů:

- Init-configserver.js – inicializace konfigurace clusteru
- Init-shard01.js, Init-shard02.js, Init-shard03.js – inicializace jednotlivých shardů
- Init- router.js – připojení shardů přes mongos router

Tyto skripty jsou volány v rámci shell skriptu mongoInit.sh, který provádí kompletní spuštění a propojení shardované infrastruktury.

1.2.7.2 Automatická distribuce data

Po úspěšném spuštění clusteru jsou data importována pomocí Python skriptů umístěch ve složce scripts/data-import. Hlavním skriptem je import-all.py, který zajišťuje načtení tří

Seminární práce z předmětu NoSQL databáze

základních CSV datasetů (patients.csv, medications.csv, procedures.csv) a jejich zápis do příslušných kolekcí v MongoDB.

MongoDB následně automaticky rozdělí data do shardů na základě zvoleného shard key. Jelikož se jedná o horizontálně škálovaný systém, MongoDB při vkládání dokumentů určuje, do kterého shardu má dokument patřit, a tento zápis provede.

1.2.7.3 Replikace dat

Každý shard v našem řešení je zároveň replikovaným setem (replica set), který zajišťuje vysokou dostupnost a odolnost vůči výpadkům. Replikace probíhá mezi jednotlivými instancemi v rámci jednoho shardu – primární uzel zpracovává zápisy, sekundární uzly slouží pro čtení a zálohu.

1.2.7.4 Dotazy a směrování

MongoDB používá komponentu **mongos** jako router, který směřuje dotazy na příslušné shardy. Díky tomu aplikace nemusí řešit, kde se konkrétní dokument fyzicky nachází – systém zajišťuje transparentní přístup k datům.

Příklad: pokud uživatel provede dotaz na pacienta s ID 12345, mongos vyhledá, do kterého shardu dokument podle shard key spadá, a pošle dotaz pouze tam. Tímto řešením je zajištěna škálovatelnost a výkon.

1.2.7.5 Počty záznamů na jednotlivých uzlech

Konkrétní výpis z příkazu sh.status(), kde je vidět, kolik záznamů je na konkrétním uzlu, je přiložen v kapitole dotazy. Z důvodu jeho velikosti zde není znovu přiložen

1.2.8 Zabezpečení

Zajištění zabezpečení databáze za provedeno pomocí autentizace a autorizace, což se řadí mezi základní bezpečnostní mechanismy pro kontrolu přístupu k databázovým zdrojům.

1.2.8.1 Autentizace

Autentizace je vyřešena pomocí zabezpečení na základě keyfile, což je doporučený způsob zabezpečení komunikace mezi jednotlivými uzly clusteru. Keyfile zajišťuje, že pouze autorizované instance MongoDB se mohou připojit do clusteru a spolu komunikovat.

Kromě tohoto jsou definováni uživatelé, v našem případě pouze administrátor, který se musí přihlásit pomocí přihlašovacího jména a hesla. Tím je zajištěno, že pouze autorizovaní uživatelé mají přístup k databázi

1.2.8.2 Autorizace

Autorizace v MongoDB pracuje na základě rolí, které definují oprávnění uživatelů k jednotlivým databázím a kolekcím. Vypracované řešení by šlo rozšířit o další role, které by měly přístup pouze na potřebné operace jako je například čtení či zápis. Tímto způsobem by se minimalizovalo riziko neautorizovaného přístupu nebo změny dat.

Seminární práce z předmětu NoSQL databáze

Konfigurace autentizace je vyřešena v souboru auth.js, Její obsah je přiložen na snímku obrazovky.



```
JS auth.js 1 x
scripts > JS auth.js > ...
1  #!/bin/bash
2
3  mongosh <<EOF
4  use admin;
5  db.createUser({user: "admin", pwd: "heslo_123", roles:[{role: "root", db: "admin"}]});
6  exit;
7  EOF
```

Použití keyfile autentizace je nezbytné pro zabezpečení clusteru v MongoDB a je to také předpoklad pro provoz shardovaného clusteru. Tento mechanismus zajišťuje bezpečnou komunikaci mezi jednotlivými komponentami systému. Dále je autentizace a autorizace nezbytná k ochraně dat před neoprávněným přístupem, což je kritické zejména v produkčním prostředí.

Celkově tato kombinace zabezpečení splňuje požadavky na bezpečný provoz MongoDB clusteru a umožňuje řídit přístupy uživatelů na základě jejich rolí.

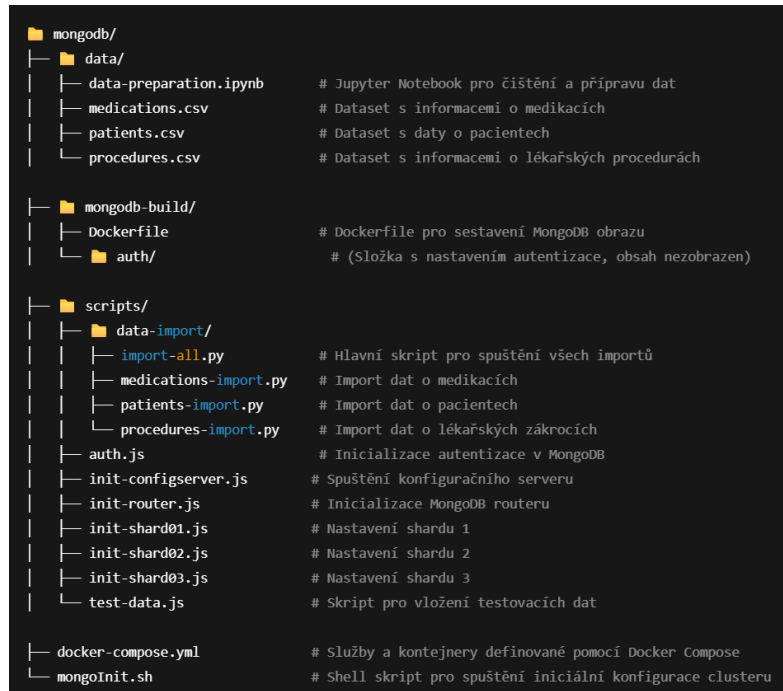
Seminární práce z předmětu NoSQL databáze

2 Funkční řešení

Tato kapitola v první části popisuje adresářovou strukturu a jednotlivé obsahy skriptů. V druhé části je podrobně popsáno, jak vytvořenou databázi spustit na zařízení.

2.1 Struktura

Na následujícím obrázku je zobrazena struktura databázového projektu. Jednotlivé soubory jsou popsány níže.



Projekt se skládá ze 2 hlavních složek (mongodb-build/ a scripts) a docker-compose.yml, který bude blíže vysvětlen v následující kapitole.

Mongodb-build obsahuje Dockerfile soubor a auth složku, ve které je umístěn vygenerovaný keyfile, který slouží k autentizaci. Na přiloženém obrázku je zobrazen Dockerfile, který slouží pro vytvoření vlastního MongoDB image.

```
1 FROM mongo:8.0.9
2 COPY /auth/mongodb-keyfile /data
3 RUN chmod 400 /data/mongodb-keyfile
4 RUN chown 999:999 /data/mongodb-keyfile
```

- FROM mongo:8.0.9 – použije image MongoDB ve verzi 8.0.9 jako základ
- COPY /auth/mongodb-keyfile /data – tento řádek zkopíruje keyfile ze složky /auth/ a vloží ho do složky /data uvnitř Docker image
- RUN chmod 400 /data/mongodb-keyfile – nastaví oprávnění souboru mongodb-keyfile tak, aby ho mohl číst pouze vlastník.

Seminární práce z předmětu NoSQL databáze

- RUN chown 999:999 /data/mongodb-keyfile – změni vlastníka na uživatele s UID 999, ze kterého se stane výchozí uživatel, pod kterým poběží MongoDB v oficiálním Docker image. Tím se zajišťuje že MongoDB bude mít přístup ke keyfilu při startu a že soubor bude vlastněn správným uživatelem

Složka s názvem scripts/ obsahuje jednotlivé skripty a python skripty pro importování dat. Skripty, které jsou potřeba pro inicializaci jednotlivých komponent MongoDB clusteru:

- auth.js – skript pro vytvoření uživatelů a nastavení autentizace/autorizace v databázi.
- init-configserver.js – inicializace config serveru
- init-router.js – nastavení mongos routeru, který směřuje dotazy mezi jednotlivými shardy.
- init-shard01.js, init-shard02.js, init-shard03.js – skripty pro nastavení jednotlivých shardů.
- test-data.js – skript na naplnění databáze testovacími daty

Pro import dat jsou využívány celkem čtyři Python skripty – tři z nich slouží pro načtení jednotlivých datasetů a jeden hlavní skript, který zajišťuje jejich provolání. Tento hlavní skript je zároveň používán při inicializaci databáze.

2.1.1 Docker-compose.yml

Docker file se skládá z několika částí. První z nich jsou routery (mongos), které fungují jako vstupní brána do clusteru. Klienti prostřednictvím routy pokládají dotazy a jejím úkolem je dotazy nasměrovat na správný shard na základě shard klíče. Důležité je, že routy nepřechovávají žádná data.

Ve vypracovaném řešení se nachází 2 routy kvůli zajištění dostupnosti databáze v případě, že by jeden spadnul a pro rozložení zátěže

```
3  ## Router
4  router01:
5    build:
6      context: mongodb-build
7    image: mongo
8    container_name: router-01
9    command: mongos --port 27017 --configdb rs-config-server/configsvr01:27017,configsvr02:27017,configsvr03:27017 --bind_ip_all --keyfile /data/mongodb-keyfile
10   ports:
11     - 27117:27017
12   restart: always
13   volumes:
14     - ./scripts:/scripts
15     - mongodb_cluster_router01_db:/data/db
16     - mongodb_cluster_router01_config:/data/configdb
17  router02:
18    build:
19      context: mongodb-build
20    image: mongo
21    container_name: router-02
22    command: mongos --port 27017 --configdb rs-config-server/configsvr01:27017,configsvr02:27017,configsvr03:27017 --bind_ip_all --keyfile /data/mongodb-keyfile
23   volumes:
24     - ./scripts:/scripts
25     - mongodb_cluster_router02_db:/data/db
26     - mongodb_cluster_router02_config:/data/configdb
27   ports:
28     - 27118:27017
29   restart: always
30   links:
31     - router01
```

- command: mongos ... – spustí se mongos proces

Seminární práce z předmětu NoSQL databáze

- --configdb rs-config-server/configsvr01:27017,... – určuje, kde jsou konfigurační servery, které uchovávají metadata
- --keyFile /data/mongodb-keyfile – parametr, který řeší autentizaci mezi nody v clusteru, kdy aby to fungovalo správně, tak každý node v clusteru musí používat stejný keyfile. Pokud by tomu tak nebylo, tak neexistuje možnost, jak ověřit, že uzly k sobě patří
- --bind_ip_all – router bude naslouchat na všech dostupných IP adresách, což je užitečné v kontejnerech

Další důležitou komponentou jsou konfigurační servery, které ukládají metadata o clusteru – kde jsou data uložena, jaké jsou shard klíče a další. Důležité je zmínit, že ve vypracovaném řešení tvoří replikační sadu ve formátu 1 primární a 2 sekundární.

```
33 ## Config Servers
34 configsvr01:
35   build:
36     context: mongodb-build
37   image: mongo
38   container_name: mongo-config-01
39   command: mongod --port 27017 --configsvr --replSet rs-config-server --keyFile /data/mongodb-keyfile
40   volumes:
41     - ./scripts:/scripts
42     - mongodb_cluster_configsvr01_db:/data/db
43     - mongodb_cluster_configsvr01_config:/data/configdb
44   ports:
45     - 27119:27017
46   restart: always
47   links:
48     - shard01-a
49     - shard02-a
50     - shard03-a
51     - configsvr02
52     - configsvr03
53 configsvr02:
54   build:
55     context: mongodb-build
56   image: mongo
57   container_name: mongo-config-02
58   command: mongod --port 27017 --configsvr --replSet rs-config-server --keyFile /data/mongodb-keyfile
59   volumes:
60     - ./scripts:/scripts
61     - mongodb_cluster_configsvr02_db:/data/db
62     - mongodb_cluster_configsvr02_config:/data/configdb
63   ports:
64     - 27120:27017
65   restart: always
66 configsvr03:
67   build:
68     context: mongodb-build
69   image: mongo
70   container_name: mongo-config-03
71   command: mongod --port 27017 --configsvr --replSet rs-config-server --keyFile /data/mongodb-keyfile
72   volumes:
73     - ./scripts:/scripts
74     - mongodb_cluster_configsvr03_db:/data/db
75     - mongodb_cluster_configsvr03_config:/data/configdb
76   ports:
77     - 27121:27017
78   restart: always
79
```

Příkaz ze řádku 39, 58 a 70 se skládá z následujícího:

- --port 27017: běží na standardním MongoDB portu.
- --configsvr: aktivuje speciální režim pro config servery.
- --replSet rs-config-server: označuje replikační sadu, ke které node patří.
- --keyFile: soubor se sdíleným tajným klíčem pro autentizaci mezi nody

Seminární práce z předmětu NoSQL databáze

Připojené volumes řádek 40, 59 a 72 obsahují:

- ./scripts:/scripts – složka se skripty pro inicializaci
- /data/db – úložiště databázových dat MongoDB.
- /data/configdb – interní konfigurace clusteru (metadata, stav replikace atd.).

Každá konfigurační server má jiný **host port** pro přístup zvenčí (např. 27119, 27120, 27121), interně běží na 27017.

V neposlední řadě jsou zde umístěny shardy, které uchovávají samotná data. Každý shard je tvořen replikovanou sadou, což pomáhá se zvýšením dostupnosti a odolnosti vůči výpadkům.

```
81  ## Shards
82  ## Shards 01
83
84  shard01-a:
85    build:
86      context: mongodb-build
87    image: mongo
88    container_name: shard-01-node-a
89    command: mongod --port 27017 --shardsvr --replSet rs-shard-01 --keyFile /data/mongodb-keyfile
90    volumes:
91      - ./scripts:/scripts
92      - mongodb_cluster_shard01_a_db:/data/db
93      - mongodb_cluster_shard01_a_config:/data/configdb
94    ports:
95      - 27122:27017
96    restart: always
97    links:
98      - shard01-b
99      - shard01-c
100  shard01-b:
101    build:
102      context: mongodb-build
103    image: mongo
104    container_name: shard-01-node-b
105    command: mongod --port 27017 --shardsvr --replSet rs-shard-01 --keyFile /data/mongodb-keyfile
106    volumes:
107      - ./scripts:/scripts
108      - mongodb_cluster_shard01_b_db:/data/db
109      - mongodb_cluster_shard01_b_config:/data/configdb
110    ports:
111      - 27123:27017
112    restart: always
113  shard01-c:
114    build:
115      context: mongodb-build
116    image: mongo
117    container_name: shard-01-node-c
118    command: mongod --port 27017 --shardsvr --replSet rs-shard-01 --keyFile /data/mongodb-keyfile
119    volumes:
120      - ./scripts:/scripts
121      - mongodb_cluster_shard01_c_db:/data/db
122      - mongodb_cluster_shard01_c_config:/data/configdb
123    ports:
124      - 27124:27017
125    restart: always
```

Obsahuje 3 repliky, což je vidět na řádku 84, 100 a 113:

- shard01-a – hlavní (primary) uzel
- shard01-b – sekundární uzel (secondary).
- shard01-c – další sekundární uzel

Seminární práce z předmětu NoSQL databáze

Příkaz na řádce 89, 105 a 118 se skládá z:

- --port 27017 – běží na standardním MongoDB portu.
- --shardsvr – aktivuje režim shard serveru – umožňuje přijímat a ukládat shardovaná data
- --replSet rs-shard-01 – název replikové sady
- --keyFile – zabezpečení a ověřování mezi nody

Odkazy, které jsou v kódu uvedené jako links, už nejsou nutné, protože Docker DNS je automaticky propojí, ale někdy mohou být užitečné pro stabilitu při startu v CI/CD prostředí

Poslední, co v docker-compose.yml je uvedeno jsou volumes. Ty slouží jako trvalé uložení dat mimo kontejnery, což zaručuje, že po restartu kontejneru zůstanou data zachována. Zároveň slouží pro oddělení dat od aplikační logiky v kontejneru a jsou nedílnou součástí pro správné fungování MongoDB, protože data, konfigurace a replikace potřebují stabilní uložení.

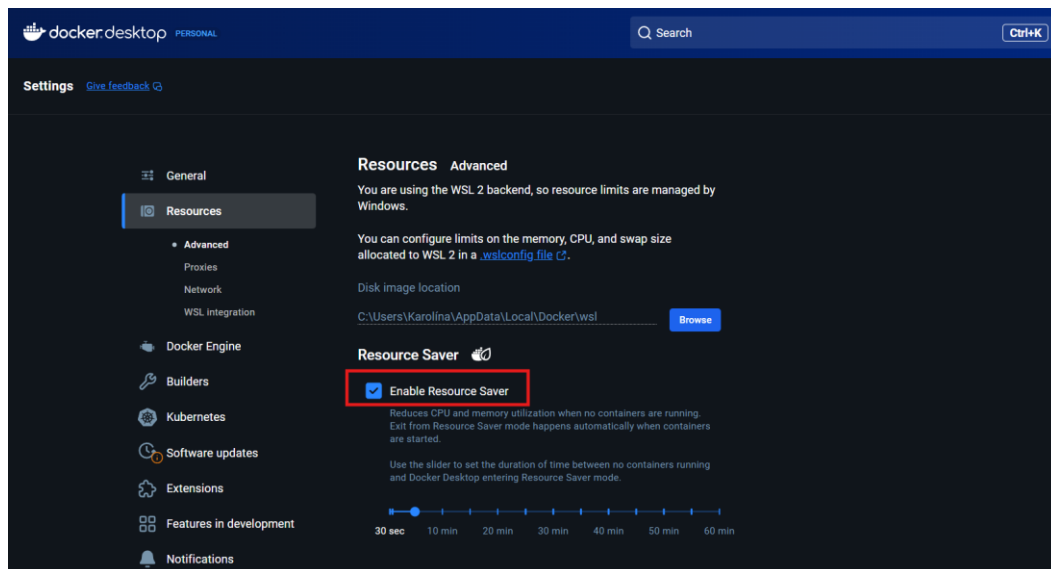
```
215 volumes:
216     mongodb_cluster_router01_db:
217     mongodb_cluster_router01_config:
218
219     mongodb_cluster_router02_db:
220     mongodb_cluster_router02_config:
221
222     mongodb_cluster_configsvr01_db:
223     mongodb_cluster_configsvr01_config:
224
225     mongodb_cluster_configsvr02_db:
226     mongodb_cluster_configsvr02_config:
227
228     mongodb_cluster_configsvr03_db:
229     mongodb_cluster_configsvr03_config:
230
231     mongodb_cluster_shard01_a_db:
232     mongodb_cluster_shard01_a_config:
233
234     mongodb_cluster_shard01_b_db:
235     mongodb_cluster_shard01_b_config:
236
237     mongodb_cluster_shard01_c_db:
238     mongodb_cluster_shard01_c_config:
239
240     mongodb_cluster_shard02_a_db:
241     mongodb_cluster_shard02_a_config:
242
243     mongodb_cluster_shard02_b_db:
244     mongodb_cluster_shard02_b_config:
245
246     mongodb_cluster_shard02_c_db:
247     mongodb_cluster_shard02_c_config:
248
249     mongodb_cluster_shard03_a_db:
250     mongodb_cluster_shard03_a_config:
251
252     mongodb_cluster_shard03_b_db:
253     mongodb_cluster_shard03_b_config:
254
255     mongodb_cluster_shard03_c_db:
256     mongodb_cluster_shard03_c_config:
257
```

Seminární práce z předmětu NoSQL databáze

2.2 Instalace

2.2.1 Windows – předpoklady pro WSL

Pro spuštění připraveného skriptu pro inicializaci a celkového zprovoznění řešení je potřeba mít na systému Windows funkční prostředí Windows Subsystem for Linux – WSL, ideálně s distribucí Ubuntu. Následně je potřeba, aby si uživatel v Docker Desktop nastavení povolil integraci s touto distribucí



2.2.2 Instalace Pythonu a potřebných knihoven

Ve WSL je potřeba mít nainstalovaný Python 3. To lze provést pomocí příkazu:

```
sudo apt install python3 python3-pip -y
```

Dále je potřebné nainstalovat potřebné knihovny pro import dat: Pandas, Pymnogo, matplotlib, seaborn. Na což lze využít příkaz

```
python3 -m pip install pandas matplotlib seaborn
```

2.2.3 Inicializace clusteru a import dat

Inicializace a importování dat je zajištěno mongoInit.sh souborem. Pro jeho spuštění je potřebné se nacházet ve WSL a přejít do složky mongod. Pro spuštění WSL lze použít příkaz:

```
wsl -d Ubuntu
```

Po přesunu do hlavní složky mongod stačí pustit příkaz

```
/mongodb$ ./mongoInit.sh
```

Po jeho úspěšném doběhnutí se v terminálu objeví tato zpráva

```
All data imported successfully.  
MongoDB cluster is fully initialized and sharded. Ready to use!
```

Seminární práce z předmětu NoSQL databáze

3 Případy užití a případové studie

3.1 Účely užití MongoDB

MongoDB je dokumentově orientovaná NoSQL databáze, která je vhodná pro ukládání a práci s velkými objemy a často se měnícími daty. Díky flexibilnímu schématu je ideální pro aplikace, kde struktura dat není pevně daná. Konkrétní případy užití jsou představeny níže

3.1.1 Zdravotnické aplikace a elektronické zdravotní záznamy

V oblasti zdravotnictví je běžnou praxí, že se pracuje s komplexními, různorodými a rychle se měnícími daty o pacientech. Zdravotnická zařízení mají potřebu uchovávat o pacientech informace jako jsou například diagnózy, výsledky testů, anamnézy, lékařské zprávy, předpisy léků a mnohé další. V tomto případě je volba MongoDB opodstatněná tím, že data mohou obsahovat různé datové struktury a není potřeba aplikovat složitou migraci schématu. To umožňuje přidávání nových typů informací a podporuje rychlý vývoj nových zdravotnických aplikací. Zařízení MongoDB ocení i v rychlosti ve vyhledávání a agregaci dat, což může napomoci lékařům například při rozhodovacím procesu.

3.1.2 Analýza a reporting dat s nutností rychlých dotazů nad velkými objemy

MongoDB podporuje výkonné dotazování a agregace nad velkými datovými soubory, což je klíčové pro analytické nástroje a reporting. Díky vestavěnému map-reduce mechanismu a agregacím lze snadno vytvářet složité statistiky, reporty a vizualizace. To ocení zejména firmy, které zpracovávají obrovské množství dat a potřebují rychlé reakce na dotazy bez velkých zpoždění.

3.1.3 3. Vývoj prototypů a agilní vývoj:

Při vývoji nových aplikací, zejména v dynamických a inovativních prostředích, je často potřeba rychle experimentovat a měnit strukturu dat podle aktuálních potřeb. MongoDB díky absenci pevného schématu umožňuje vývojářům flexibilně přidávat nové typy dat a atributy, aniž by bylo nutné zdlouhavě měnit databázovou strukturu. Tento přístup podporuje agilní metodiky vývoje, kde se aplikace vyvíjí iterativně a postupně podle zpětné vazby uživatelů.

3.2 Účel projektu

Projekt lze zařadit do oblasti **elektronických zdravotních záznamů (EHR – Electronic Health Records)**, tedy digitální správy a archivace zdravotnických údajů pacientů. Tento systém slouží jako **základní úložiště zdravotnických dat**, které mohou být dále analyzována nebo vizualizována, případně napojena na další systémy zdravotnické péče (např. plánování zákroků, doporučovací systémy apod.).

MongoDB zde plní úlohu **robustního a škálovatelného backendu** s možností připojení analytických nástrojů skrze Python a další jazyky, čímž se otevírá cesta pro další datovou analýzu a vývoj inteligentních nástrojů nad uloženými záznamy.

Seminární práce z předmětu NoSQL databáze

3.3 Proč byla zvolena MongoDB

Pro zpracování a analýzu syntetických zdravotních dat byla zvolena NoSQL database MongoDB. Hlavním důvodem byla potřeba flexibilního a škálovatelného úložiště, které umožňuje snadno pracovat s heterogenními daty v podobě dokumentů JSON. Data ze Synthea jsou složitá a obsahují různé typy záznamů, jako jsou pacienti, procedury, diagnózy a vztahy mezi nimi, které není možné jednoduše uložit do pevné relační struktury.

Řešení slouží k efektivnímu ukládání, vyhledávání a analýze zdravotních dat za účelem podpory simulací a výzkumu v oblasti zdravotnictví. Jedná se o případ užití, kdy je klíčová flexibilita modelu dat, rychlé dotazy a možnost snadné integrace s analytickými nástroji a vizualizacemi.

Před vypracování projektu byly zváženy i jiné database, ale vždy se našlo něco, co oproti MongoDB byla nevýhda:

Cassandra je optimalizovaná pro velmi rychlé zápisy a rozsáhlé distribuované systémy, ale nemá tak flexibilní model dokumentů a složité dotazy jako MongoDB.

Neo4j je grafová databáze, ideální pro aplikace, kde jsou důležité vztahy a propojení mezi daty (například sociální sítě nebo doporučovací systémy). Pro potřebu ukládání heterogenních a flexibilních dokumentů zdravotních dat však není tak vhodná.

Redis je výborný pro rychlé operace v paměti, caching a jednoduché datové struktury, ale není určen pro trvalé ukládání komplexních a rozsáhlých datových struktur.

3.4 Případové studie

3.4.1 Novo & MongoDB Atlas

Novo Nordisk je globální farmaceutická společnost, která se zaměřuje na vývoj a výrobu léků pro léčbu závažných chronických onemocnění, jako je například cukrovka či hemofilie. V průběhu své činnosti pravidelně vytváří rozsáhlé dokumenty zvané Clinical Study Reports (CSR), které jsou určeny pro regulační úřady, aby mohli schvalovat či zamítnout nové léčivo. Tento proces byl dříve velmi časově náročný – zpracování jednoho CSR mohlo trvat až 12 týdnů a vyžadovalo zapojení mnoha odborníků.

Aby Novo Nordisk zefektivnil tento klíčový proces, vyvinul interní nástroj s názvem NovoScribe, který využívá kombinaci generativní umělé inteligence a dokumentové databáze MongoDB Atlas. Díky této technologii se podařilo automatizovat a zrychlit celý proces tvorby CSR na pouhých 10 minut. MongoDB Atlas zde hraje zásadní roli díky své schopnosti pracovat s různorodými nestrukturovanými daty, rychle je ukládat a poskytovat rozhraní pro spolupráci mezi systémy. Výsledkem je nejen významná časová úspora, ale i vyšší kvalita výstupních dokumentů.

Přínosy tohoto řešení jsou obrovské – každý den, o který se podaří dříve uvést nový lék na trh, znamená potenciální nárůst příjmu až o 15 milionů dolarů, a především rychlejší přístup

Seminární práce z předmětu NoSQL databáze

pacientů k léčbě. Řešení od Novo Nordisk je tak ukázkovým příkladem, jak lze MongoDB využít v oblasti zdravotnictví a farmacie pro vysoce důležité a citlivé procesy. Tato případová studie potvrzuje, že MongoDB není pouze technologickým nástrojem, ale může sehrát zásadní roli v inovacích, které mají přímý dopad na lidské zdraví.

3.4.2 Enpal: Řízení IoT dat v oblasti udržitelné energetiky

Společnost Enpal se pohybuje v oblasti instalace solárních panelů a dalších chytrých zařízení pro domácnosti v Německu. Na jejich denním programu bylo zpracovávání dat z více jak 80 000 zařízení v reálném čase, přičemž každé z těchto zařízení generovalo stovky metrik za minutu. Ze začátku byly využívány relační databáze, ale ty často selhávaly kvůli náročnosti na škálování a složitosti správy časových řad

Pro zlepšení zpracovávání dat byla firmou zvolena řešení MongoDB Atlas s využitím time series collections. Díky tomu došlo ke konsolidaci datových toků a vytvoření jednotné pipeline pro agregaci dat. Firma hojně využívá výhod, které nabízí geografický sharding, což rozdělení dat dle zemí, propojený s cloudovou platformou Azure. Podle interních odhadů se společnosti podařilo snížit náklady na provoz databáze až o 60 % a současně zajistit škálovatelnost až ro 100 000 zařízení při nízkých provozních nákladech

Tento případ ilustruje, že MongoDB není jen „startupová databáze“, ale robustní nástroj schopný podporovat velké IoT infrastruktury s důrazem na efektivitu a compliance.

3.4.3 Vainu: Dynamická práce s firemními daty

Vainu je severská technologická firma, která se zaměřuje na B2B databáze a analýzu firemních údajů. Původně firma pracovala s vlastními SQL servery, u kterých se ovšem objevil problém v pružnosti reakce na časté změny v datových schématech a rychle rostoucí objem dat. Vývojáři tak trávili většinu času správou infrastruktury a řešením výpadků než prací, která byla potřeba vykonávat.

Kvůli těmto komplikacím se firma rozhodla přejít na MongoDB Atlas a tím získala plně spravované řešení s automatickým zálohováním, obnovou dat a snadnou škálovatelností. MongoDB umožnilo vývojovému týmu přesunout pozornost z provozních problémů na inovace produktu. Dalším plusem, který firma zaznamenala, bylo zvýšení stability a dostupnosti svých služeb.

Tato případová studie ukazuje, jak MongoDB přispívá k transformaci technologických týmů směrem k vyšší efektivitě a kvalitě vývoje.

Seminární práce z předmětu NoSQL databáze

4 Výhody a nevýhody

4.1 Výhody a nevýhody MongoDB jako NoSQL databáze

MongoDB jako představitel nabízí několik významných výhod. Jednou z hlavních je flexibilita schématu (tzv. *schema-less*), kdy není nutné předem pevně definovat strukturu dokumentu. Tato vlastnost umožňuje rychlé změny ve struktuře dat, což je výhodné zejména při agilním vývoji a častém nasazování nových verzí aplikace.

Další výhodou je podpora horizontálního škálování prostřednictvím shardování, které umožňuje rozdělit data napříč více servery, a tak zvládat větší objemy dat i vyšší zátěž. MongoDB také vyniká vysokým výkonem při čtení i zápisu díky efektivnímu ukládání dat ve formátu BSON (Binary JSON). Významným prvkem je i podpora replikace – tzv. *replica sets* – která zajišťuje vysokou dostupnost databáze a odolnost vůči výpadkům. Také je důležité zmínit jeho jednoduchost v instalaci. V neposlední řadě je důležité mezi jeho výhody zařadit schopnost zpracování dat v reálném čase. To je umožněno prostřednictvím agregačního frameworku MongoDB, který umožňuje analýzu a zpracování velkého množství dat bez časové prodlevy, což je oceněno například při aktivitách jako je reporting, tvorba dashboardů či analytických aplikací.

Mezi nevýhody MongoDB je určitě důležité zmínit jeho vyšší spotřebu paměti. Tento problém vzniká v důsledku ukládání názvů polí všech dokumentů pořád dokola. To je oproti relačním databázím paměťově náročnější, protože ty ukládají záznamy ve sloupcích s pevně danou strukturou. Nevýhoda, která se taktéž vztahuje k dokumentům je omezení jejich velikosti. Jeden dokument může mít maximálně 16 MB, což může způsobovat problém pro aplikace, které pracují s velkými objekty nebo soubory, které by obtížněji rozdělávaly na menší části. Další nevýhodou je konfigurace shardingu, který sice umožňuje škálování databáze do šířky, ale jeho nastavení není vždy jednoduché. Komplikace může nastat při špatném výběru shard klíče, který pak může vést k nerovnoměrnému rozložení dat a způsobit problém s výkoností. Jako poslední nevýhodou oproti relačním databázím je MongoDB náročné z hlediska výpočetních zdrojů jako například CPU, RAM, disků. Pokud je prostředí omezené hardwarem, může být složité dosáhnout požadovaného výkonu.

4.2 Výhody a nevýhody vypracovaného řešení

Mezi výhody je určitě dobré zařadit modularitu a přenositelnost vypracovaného řešení. To je zajištěno pomocí využití Dockeru, díky kterému lze celý MongoDB cluster velmi snadno přenášet a nasazovat prakticky kamkoli. Tímto řešením se předchází zdlouhavým manuálním konfiguracím. S touto výhodou úzce souvisí i automatizované skripty, které s pomocí souboru `docker-compose.yml` jsou schopné inicializovat a konfigurovat cluster, čímž se snižuje časová náročnost a riziko lidské chyby.

Další výhodou je zajištění vysoké dostupnosti pomocí replikačních setů, kdy každý shard i konfigurační server ho má. Pokud by se stalo, že by některý z uzlů vypadl, tak zbytek systému je schopná pokračovat v provozu bez velkého přerušení. Ohledně dostupnosti je

Seminární práce z předmětu NoSQL databáze

určitě důležité ještě zmínit možnost horizontálního škálování pomocí shardů. Pokud by v budoucnosti byla potřeba přidat nový shard, tak nebude potřeba dělat zásadní změny v systému, ale navíc to bude možné provést i bez výpadku stávajících služeb

Jako nevýhoda řešení je složitost architektury, protože pro malé a jednoduché aplikace může být využití celého MongoDB clusteru s více uzly a konfiguračními servery zbytečně náročné. V případě tohoto řešení by bylo lepší zvolit například single-node MongoDB nebo jiná jednodušší řešení. Další velkou nevýhodou je vyšší náročnost na systémové prostředky, což je více rozepsáno v kapitole výhod a nevýhod MongoDB.

Všeobecně v distribuovaném systému může být složitější diagnostikovat chybu, což je tady zařazeno jako další a poslední chyba. Pokud člověk nemá dostatečné znalosti, a to ne jenom o MongoDB, ale i o dockeru, tak pro něj může být náročné najít přesnou lokalitu chyby, jako je například problém s připojení nodu navzájem.

5 Další specifika

Ve vypracovaném řešení nebyly použita žádná specifika, která by se lišila od doporučených.

6 Data

Tato část seminární práce je zaměřena na analýzu tří vybraných data setů, které jsou součástí databáze popsané v předchozích kapitolách. Každý ze 3 data setů obsahuje minimálně 5 000 záznamů a poskytuje cenné informace o pacientech, jejich procedury a léky, které jim byly předepsané. Datasets byly získané pomocí Synthea generátoru, který slouží k vygenerování smysluplných, ale nereálných dat ohledně zdravotního stavu pacientů.

Veškerá analýza a úprava dat byla uskutečněna pomocí jupyter notebook. Toto řešení je uloženo se zbytkem vypracované v databázi v příloze.

Databáze bude pracovat se všemi daty, které jsou dostupné v ukázkových setech, což je konkrétně 93 177 zápisů. Je to primárně kvůli tomu, že sety jsou navzájem provázané pomocí PATIENT_ID a pokud by se začaly náhodně data vybírat, tak by se mohlo stát, že pak příkazy napříč data sety nemuseli správně fungovat, protože by hodnoty neexistovali

Všechna data jsou původně ve formátu **CSV**, ale před vložením do databáze MongoDB jsou převedena do formátu **JSON (BSON)**, což MongoDB interně používá. Mongo databáze poté uchovává každý záznam jako jeden **dokument**, kde každý dokument reprezentuje jednoho zaměstnance (jeden řádek původního CSV).

Alternativní datové struktury (např. **XML**, **XLSX**, nebo přímo **JSON**) nebyly zvoleny, protože:

- **XML** je zbytečně komplikovaný a méně přehledný,

Seminární práce z předmětu NoSQL databáze

- **XLSX** není ideální pro práci v automatizovaných skriptech a vyžaduje specifické knihovny pro načítání,
- Přímý **JSON** není vždy dostupný pro velké data sety a obtížněji se analyzuje ve formě tabulky před importem.

Díky kombinaci CSV → JSON tak využíváme to nejlepší z obou směrů: jednoduchost zpracování i kompatibilitu s MongoDB.

Seminární práce z předmětu NoSQL databáze

6.1 Data sety

6.1.1 Patients

Tento data set poskytuje informace o pacientech. Každý záznam obsahuje osobní údaje jako je datum narození, případně datum úmrtí, trvalé bydliště, pohlaví a specifická id, pomocí kterých jsou k nim přiřazeny další informace jako medikace, pohledávky za péči a další.

6.1.1.1 Analýza datasetu

Data set má celkem 12 352 řádků po očištění dat o chybějící hodnoty, přičemž se tam nevyskytl žádný duplikát, a 20 sloupců. Z důvodu jednoduchosti pro čtení byl ke každému zápisu přidán řádek s věkem, zároveň zde byly odstraněny sloupce jako bylo například pas či zda je osoba řidič či nikoliv. Konkrétní výstup z jupyter notebooku, je zobrazen na obrázku.

```
Počet řádků: 12352
Počet sloupců: 20
Náhled:
```

	PATIENT_ID	BIRTHDATE	DEATHDATE	SSN	FIRST	LAST	RACE	ETHNICITY	GENDER	BIRTHPLACE	ADDRESS	CITY	STATE	COUNTY	ZIP	LAT	LONG	HEALTHCARE_EXPENSES	HEALTHCARE_COVERAGE	AGE
0	102bcb4-e38-48ce-a2bd-dfdd582b271	2017-08-24	NaN	999-68-6630	Jacinto644	Kris249	white	nonhispanic	M	Beverly Massachusetts US	888 Hiddle Ferry Suite 38	Springfield	Massachusetts	Hampden County	1106.0	42.151961	-72.598959	8446.49	1498.08	7
1	06731ba4-d08f-4417-b86e-f2b1e9baa5	2016-08-01	NaN	999-15-5895	Alva958	Krajcik437	white	nonhispanic	F	Boston Massachusetts US	1040 Skiles Trailer	Walpole	Massachusetts	Norfolk County	2081.0	42.177370	-71.281553	89893.40	1845.72	8
2	aed9eba3-dbd4-4389-a781-f72019388548	1992-06-30	NaN	999-27-3385	Jayson808	Fade536	white	nonhispanic	M	Springfield Massachusetts US	1056 Harris Lane Suite 70	Chicopee	Massachusetts	Hampden County	1020.0	42.181642	-72.608842	577445.86	3528.84	32
3	199c586f-a76-4091-9998-eed4c02ee7a	2004-01-09	NaN	999-73-2461	Jimmi993	Harris789	white	nonhispanic	F	Worcester Massachusetts US	201 Mitchell Lodge Unit 67	Pembroke	Massachusetts	Plymouth County	NaN	42.075292	-70.757035	336701.72	2705.64	21
4	353016ea-a0ff-4154-85bb-1cf8b6ced20	1996-11-15	NaN	999-60-7372	Gregorio366	Auer97	white	nonhispanic	M	Patras Achaea GR	1050 Lindgren Extension Apt 38	Boston	Massachusetts	Suffolk County	2135.0	42.352434	-71.028610	484076.34	3043.04	28

```
Typy sloupců:
PATIENT_ID      object
BIRTHDATE       datetime64[ns]
DEATHDATE       object
SSN             object
FIRST           object
LAST            object
RACE            object
ETHNICITY       object
GENDER          object
BIRTHPLACE      object
ADDRESS         object
CITY            object
STATE          object
COUNTY         object
ZIP             float64
LAT             float64
LONG            float64
HEALTHCARE_EXPENSES float64
HEALTHCARE_COVERAGE float64
AGE             int64
dtype: object

Chybějící hodnoty v jednotlivých sloupcích:
...

Nulizováno 0 duplicitních záznamů.
Maximální věk: 115
Minimální věk: 4
```

6.1.2 Procedures

Data set obsahuje informace o provedených procedurách provázaných na pacienty z předchozího souboru. Je zde uvedený jejich datum, kód dané procedury a její popis, pokud bylo potřeba. Zároveň je zde uvedena i jejich cena.

6.1.2.1 Analýza data setu:

Data set má celkem 75 473 řádků po očištění data setů o chybějící hodnoty, přičemž se tam nevyskytl žádný duplikát, a 8 sloupců. Konkrétní výstup z jupyter notebooku, kde byla analýza provedena je na přiloženém obrázku

Seminární práce z předmětu NoSQL databáze

```
Počet řádků: 75473
Počet sloupců: 8
Náhled:
```

	DATE	PATIENT_ID	PROCEDURE_ID	CODE	DESCRIPTION	BASE_COST	REASONCODE	REASONDESCRIPTION
0	2020-03-01	f0f3bc8d-ef38-49ce-a2bd-dfdda982b271	681c380b-3c84-4c55-80a6-db3d9ea12fee	261352009	Face mask (physical object)	516.65	840544004.0	Suspected COVID-19
1	2020-03-13	067318a4-db8f-447f-8b6e-f2f61e9baaa5	1ea74a77-3ad3-4948-a9cc-3084462035d6	261352009	Face mask (physical object)	516.65	840544004.0	Suspected COVID-19
2	2020-04-28	067318a4-db8f-447f-8b6e-f2f61e9baaa5	e03b96de-5604-4989-a2d5-03a63e041eab	117015009	Throat culture (procedure)	2169.16	43878008.0	Streptococcal sore throat (disorder)
3	2020-03-11	ae9efba3-ddc4-43f9-a781-f72019388548	eeab7c2d-71ba-4e04-af16-87a01dce7d54	261352009	Face mask (physical object)	516.65	840544004.0	Suspected COVID-19
4	2020-03-01	199c586f-af16-4091-9998-ee4cf02ee7a	8333efdf-f7bf-43bb-b73f-2b663d14c1ad	261352009	Face mask (physical object)	516.65	840544004.0	Suspected COVID-19

```
Typy sloupců:
DATE          object
PATIENT_ID    object
PROCEDURE_ID  object
CODE          int64
DESCRIPTION   object
BASE_COST     float64
REASONCODE    float64
REASONDESCRIPTION object
dtype: object

Chybějící hodnoty:
DATE          0
PATIENT_ID    0
PROCEDURE_ID  0
CODE          0
DESCRIPTION   0
BASE_COST     0
REASONCODE    0
REASONDESCRIPTION 0
dtype: int64
Součet provedených procedur: 75473
Průměr BASE_COST: 321253054.6036331
Duplikáty (0 záznamů):
```

6.1.3 Medication

Poslední data set, který je zde využit poskytuje informace o užívaných lécích, které jsou opět provázané na konkrétní pacienty. Je zde uvedena jejich cena, kolik na ně přispívají pojišťovny a název samotného léku.

6.1.3.1 Analýza datasetu

Data set obsahuje celkem 5 352, přičemž se tam nevyskytl žádný duplikát, a 13 sloupců. V tomto datasetu nebyly prázdné hodnoty odebrány, protože tam jsou uvedeny správně. Například ne každý pacient přestal daný lék používat. Konkrétní výstupy jsou přiloženy na obrázku

```
Počet řádků: 5352
Počet sloupců: 13
Náhled:
```

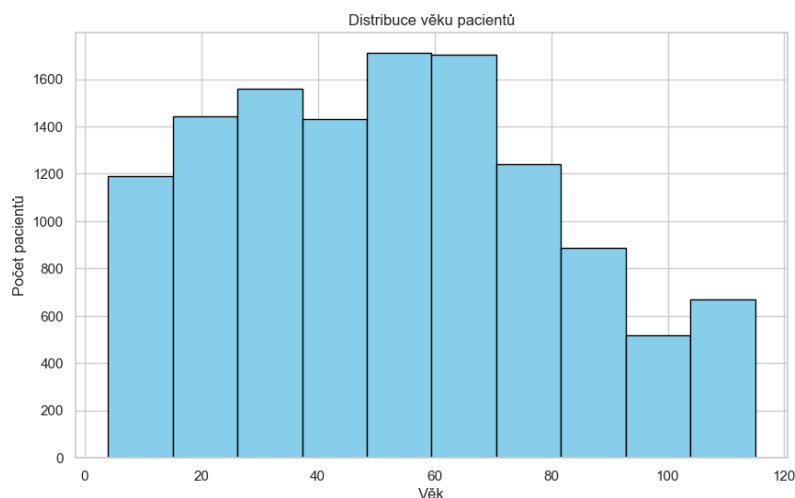
	START	STOP	PATIENT	PAYER	ENCOUNTER	CODE	DESCRIPTION	BASE_COST	PAYER_COVERAGE	DISPENSES	TOTALCOST	REASONCODE	REASONDESCRIPTION
0	2014-06-27T07:10:43Z	2015-08-26T12:57:17Z	0fe22cec-1a19-99da-b67f-7c364c4d3e	b046940f-1664-3047-bca7-dfa76bc352a4	23c14b40-6a1e-a422-e0a2-751e44b4ca7	807283	Mirena 52 MG Intrauterine System	10.00	0.00	12	120.00	NaN	NaN
1	2019-08-05T15:17:17Z	2020-07-30T12:15:17Z	0fe22cec-1a19-99da-b67f-7c364c4d3e	b046940f-1664-3047-bca7-dfa76bc352a4	70d96778-4ec3-edc8-423e-c6018962884	757594	Jolvette 28 Day Pack	229.32	0.00	12	2751.84	NaN	NaN
2	2020-07-30T12:15:17Z	2021-07-25T12:15:17Z	0fe22cec-1a19-99da-b67f-7c364c4d3e	b046940f-1664-3047-bca7-dfa76bc352a4	4aa5f00f-829c-bb8-2b9e-6139e49e6151	757594	Jolvette 28 Day Pack	764.40	0.00	12	9172.80	NaN	NaN
3	2021-10-24T10:41:12Z	NaN	0fe22cec-1a19-99da-b67f-7c364c4d3e	b046940f-1664-3047-bca7-dfa76bc352a4	3b81c72e-3c1b-6259-21f5-fa49a4a0eb7f	310325	ferrous sulfate 325 MG Oral Tablet	0.15	0.15	43	6.45	NaN	NaN
4	2021-11-07T12:23:13Z	2021-11-07T12:23:13Z	0fe22cec-1a19-99da-b67f-7c364c4d3e	b046940f-1664-3047-bca7-dfa76bc352a4	0ffa9f6b-d03e-7795-6aba-30a28d7b2378	1535362	sodium fluoride 0.0272 MG/MG Oral Gel	129.94	129.94	1	129.94	66383009.0	Gingivitis (disorder)

```
Typy sloupců:
START          object
STOP           object
PATIENT        object
PAYER          object
ENCOUNTER      object
CODE          int64
DESCRIPTION    object
BASE_COST     float64
PAYER_COVERAGE float64
DISPENSES     int64
TOTALCOST     float64
REASONCODE    float64
REASONDESCRIPTION object
dtype: object

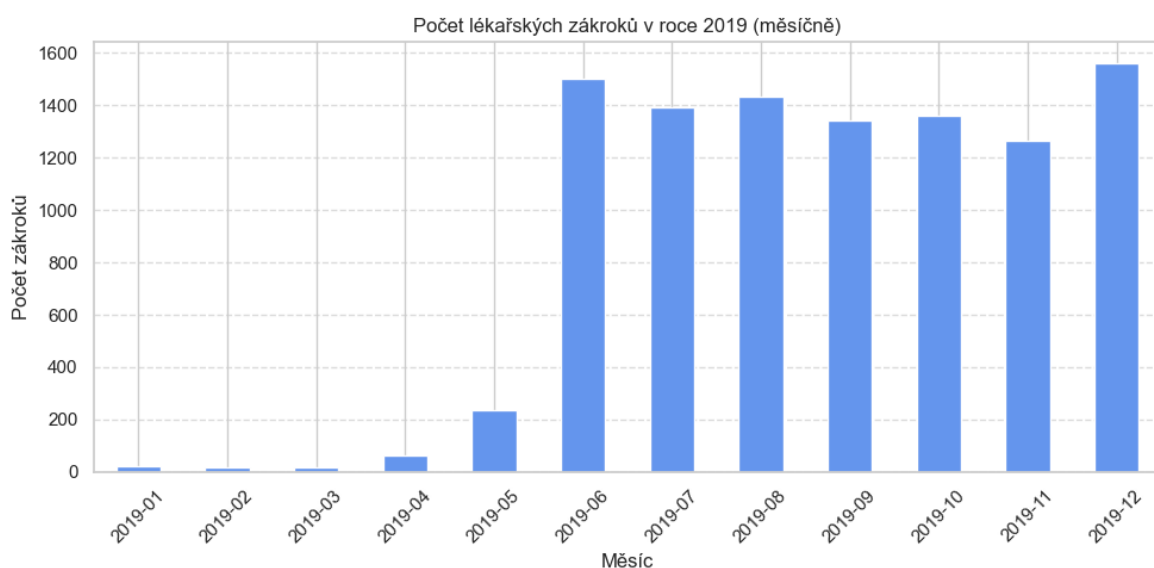
Chybějící hodnoty:
START          0
STOP           366
PATIENT        0
PAYER          0
ENCOUNTER      0
CODE          0
DESCRIPTION    0
BASE_COST     0
...
REASONCODE    1058
REASONDESCRIPTION 1058
dtype: int64
Duplikáty (0 záznamů):
```

Seminární práce z předmětu NoSQL databáze

6.2 Grafická vizualizace dat



Tento sloupkový graf znázorňuje počet zaznamenaných lékařských zákroků za jednotlivé měsíce roku 2019. Data byla agregována podle kalendářních měsíců a poté vizualizována formou bar chartu, který umožňuje jasně identifikovat případné sezónní výkyvy nebo období vyšší zdravotní zátěže.



Graf zobrazuje histogram, který ukazuje rozložení věku pacientů v daném souboru dat. Osa X reprezentuje věkové intervaly (vytvořené po deseti letech), zatímco osa Y znázorňuje počet pacientů spadajících do daného věkového rozpětí.

Z grafu lze vyčíst, které věkové skupiny jsou v datovém souboru zastoupeny nejvíce. Například pokud histogram vykazuje výrazný vrchol v dětském věku (0–10 let), může to naznačovat zaměření datasetu na pediatrii nebo na období zvýšeného výskytu dětských infekcí (např. COVID-19 u dětí). Histogram je užitečný nástroj pro rychlé pochopení demografického profilu pacientů, což může být klíčové pro epidemiologické nebo zdravotnické analýzy.

Seminární práce z předmětu NoSQL databáze

7 Dotazy

7.1 Agregční funkce

7.1.1 Počet předepsaných léků na každého pacienta

```
> db.medications.aggregate([
  { $group: { _id: "$PATIENT_ID", totalPrescriptions: { $sum: "$DISPENSES" } } },
  { $sort: { totalPrescriptions: -1 } }
])
```

Tento příkaz využívá aggregate pipeline, která kombinuje několik operací. Nejprve seskupí záznamy z kolekce medications podle pacienta (\$group) a spočítá celkový počet vydaných dávek léků pomocí \$sum. Nakonec pomocí \$sort seřadí pacienty podle počtu předepsaných dávek v sestupném pořadí. Tento dotaz pomáhá identifikovat pacienty s největším počtem medikací.

Výstup:

```
< {
  _id: 'd45d6e91-94e0-fa36-ca99-0693baf517d6',
  totalPrescriptions: 2027
}
{
  _id: '488bff1c-ccdc-f85f-a347-8bddf6c33568',
  totalPrescriptions: 1131
}
{
  _id: '5bc6b903-8fcd-c8ea-8be6-bf2351e1f78a',
  totalPrescriptions: 1010
}
```

Seminární práce z předmětu NoSQL databáze

7.1.2 Průměrná cena zákroku podle kódu

```
> db.procedures.aggregate([
  { $group: { _id: "$CODE", avgCost: { $avg: "$BASE_COST" } } },
  { $sort: { avgCost: -1 } }
])
```

Tento dotaz seskupuje zákroky podle jejich kódu a počítá průměrné náklady (\$avg) na každý typ zákroku. Následně jsou výsledky seřazeny sestupně podle ceny, což umožňuje zjistit, které procedury jsou v průměru nejdražší.

Výstup:

```
< {
  _id: 88039007,
  avgCost: 113117.27
}
{
  _id: 232717009,
  avgCost: 51693.25626984127
}
{
  _id: 429609002,
  avgCost: 38018.49333333333
}
```

7.1.3 Kolik pacientů podstoupilo zákroky v každém roce?

```
> db.procedures.aggregate([
  {
    $project: {
      year: { $substr: ["$DATE", 0, 4] },
      PATIENT_ID: 1
    }
  },
  { $group: { _id: "$year", patients: { $addToSet: "$PATIENT_ID" } } },
  { $project: { year: "$_id", count: { $size: "$patients" }, _id: 0 } },
  { $sort: { year: 1 } }
])
```

Tento dotaz nejdříve extrahuje rok z pole DATE pomocí \$substr. Následně seskupí záznamy podle roku a pomocí \$addToSet vytvoří seznam unikátních pacientů. Počet pacientů na daný rok je spočítán pomocí \$size. Výsledek ukazuje, kolik různých pacientů podstoupilo zákroky v jednotlivých letech.

Seminární práce z předmětu NoSQL databáze

Výstup:

```
< {
  year: '2017',
  count: 40
}
{
  year: '2018',
  count: 46
}
{
  year: '2019',
  count: 1521
}
{
  year: '2020',
  count: 9242
}
```

7.1.4 Celkové náklady na léky podle pojišťovny (PAYER)

```
> db.medications.aggregate([
  { $group: { _id: "$PAYER", totalCost: { $sum: "$TOTALCOST" } } },
  { $sort: { totalCost: -1 } }
])
```

Dotaz seskupuje záznamy podle identifikátoru pojišťovny a spočítá celkové náklady na předepsané léky. Výsledek pomáhá analyzovat výdaje jednotlivých pojišťoven v systému.

Výstup:

Seminární práce z předmětu NoSQL databáze

```
< {
  _id: 'a735bf55-83e9-331a-899d-a82a60b9f60c',
  totalCost: 721966.86
}
{
  _id: '26aab0cd-6aba-3e1b-ac5b-05c8867e762c',
  totalCost: 358158.33999999997
}
{
  _id: 'df166300-5a78-3502-a46a-832842197811',
  totalCost: 189414.4
}
```

7.1.5 Počet unikátních léků, které byly předepsány

```
> db.medications.aggregate([
  { $group: { _id: "$DESCRIPTION" } },
  { $count: "uniqueMedications" }
])
```

Tento dotaz seskupí záznamy podle názvu léku (DESCRIPTION), což v MongoDB efektivně vytvoří množinu unikátních hodnot. Pomocí \$count spočítá, kolik různých medikací bylo v systému zaznamenáno.

Výstup:

```
< {
  uniqueMedications: 139
}
```

7.1.6 Průměrné HEALTHCARE_EXPENSES pacientů podle pohlaví (GENDER)

```
> db.patients.aggregate([
  {
    $group: {
      _id: "$GENDER",
      avgHealthcareExpenses: { $avg: "$HEALTHCARE_EXPENSES" }
    }
  }
])
```

Tento agregát spočítá průměrné náklady na zdravotní péči (HEALTHCARE_EXPENSES) seskupené podle pohlaví (GENDER).

Seminární práce z předmětu NoSQL databáze

Použití operátoru \$group umožňuje seskupit data podle zadaného pole a \$avg vypočítá průměrnou hodnotu ve skupině.

Agregační pipeline v MongoDB umožňuje efektivní zpracování a analýzu dat přímo na serveru.

Tento přístup je vhodný pro získání přehledů a statistik nad velkými datasetem.

Výstup:

```
< {
  _id: 'M',
  avgHealthcareExpenses: 867090.744332568
}
{
  _id: 'F',
  avgHealthcareExpenses: 780804.3847257316
}
{
  _id: null,
  avgHealthcareExpenses: null
}
```

Seminární práce z předmětu NoSQL databáze

7.2 Embadded dokumenty

7.2.1 Vypsát seznam všech pacientů s polem fullName

```
> db.patients.aggregate([
  {
    $project: {
      fullName: { $concat: ["$FIRST", " ", "$LAST"] },
      BIRTHDATE: 1
    }
  }
])
```

Tento dotaz vytváří nové pole fullName spojením křestního jména a příjmení pomocí \$concat. Projektuje zároveň BIRTHDATE, což je užitečné pro čitelnější výstup nebo export.

Výstup:

```
< {
  _id: ObjectId('68470163b11a9c37c2f2ac18'),
  BIRTHDATE: '2017-08-24',
  fullName: 'Jacinto644 Kris249'
}
{
  _id: ObjectId('68470163b11a9c37c2f2ac19'),
  BIRTHDATE: '2016-08-01',
  fullName: 'Alva958 Krajcik437'
}
```

7.2.2 Odebrání zbytečných polí (např. interní ENCOUNTER)

```
> db.medications.aggregate([
  { $unset: ["ENCOUNTER"] }
])
```

Používá operátor \$unset, který odstraní specifikované pole ze všech dokumentů. Tento dotaz například čistí výstup pro export nebo reporting tím, že odstraňuje nepodstatná pole.

Seminární práce z předmětu NoSQL databáze

Výstup:

```
< {
  _id: ObjectId('684701645bf4fd540162971b'),
  MEDICATION_START: '2016-01-23T19:37:03Z',
  MEDICATION_STOP: '2016-01-23T19:37:03Z',
  PATIENT_ID: '446d8007-ae35-2994-77cd-c44d9f4286a6',
  PAYER: 'df166300-5a78-3502-a46a-832842197811',
  MEDICATION_CODE: 1535362,
  DESCRIPTION: 'sodium fluoride 0.0272 MG/MG Oral Gel',
  BASE_COST: 129.94,
  PAYER_COVERAGE: 79.94,
  DISPENSES: 1,
  TOTALCOST: 129.94,
  REASONCODE: 66383009,
  REASONDESCRIPTION: 'Gingivitis (disorder)'
}
```

7.2.3 Zákroky, které měly "Suspected COVID-19" jako důvod

```
> db.procedures.aggregate([
  { $match: { REASONDESCRIPTION: "Suspected COVID-19" } },
  {
    $project: {
      DATE: 1,
      PATIENT_ID: 1,
      CODE: 1,
      DESCRIPTION: 1,
      COST: "$BASE_COST"
    }
  }
])
```

Filtrování všech procedur podle specifického důvodu zákroku. Projektování slouží k přejmenování pole BASE_COST na COST pro lepší čitelnost ve výstupu.

Výstup:

Seminární práce z předmětu NoSQL databáze

```
< {
  _id: ObjectId('6847016557f199d1cdce6f88'),
  DATE: '2020-03-02',
  PATIENT_ID: '353016ea-a0ff-4154-85bb-1cf8b6cedf20',
  CODE: 261352009,
  DESCRIPTION: 'Face mask (physical object)',
  COST: 516.65
}
{
  _id: ObjectId('6847016557f199d1cdce6f9e'),
  DATE: '2020-02-25',
  PATIENT_ID: 'b9fd2dd8-181b-494b-ab15-e9f286d668d9',
  CODE: 261352009,
  DESCRIPTION: 'Face mask (physical object)',
  COST: 516.65
}
```

7.2.4 Vyhledání pacientů podle rozsahu data narození

```
> db.patients.aggregate([
  {
    $match: {
      BIRTHDATE: { $gte: "1970-01-01", $lte: "1980-12-31" }
    }
  },
  { $sort: { BIRTHDATE: 1 } }
])
```

Vyhledá pacienty narozené v 70. letech. Dotaz používá \$match pro filtrování na základě rozsahu hodnot a \$sort pro seřazení výstupu podle data narození.

Seminární práce z předmětu NoSQL databáze

Výstup:

```
< {
  _id: ObjectId('68470163b11a9c37c2f2ad05'),
  PATIENT_ID: '72ee7748-bc1e-4e8a-9c26-3aee16bb93f7',
  BIRTHDATE: '1970-01-02',
  DEATHDATE: NaN,
  SSN: '999-94-8786',
  FIRST: 'Bell723',
  LAST: 'Little434',
  RACE: 'white',
  ETHNICITY: 'nonhispanic',
  GENDER: 'F',
  BIRTHPLACE: 'Boston Massachusetts US',
  ADDRESS: '120 Hahn Parade',
  CITY: 'Framingham',
  STATE: 'Massachusetts',
  COUNTY: 'Middlesex County',
  ZIP: 1701,
  LAT: 42.331358277560746,
  LON: -71.46958429301526,
  HEALTHCARE_EXPENSES: 1194956.43,
  HEALTHCARE_COVERAGE: 11600.6,
  AGE: 55
}
```

7.2.5 Top 5 nejčastěji předepisovaných léků podle názvu

```
> db.medications.aggregate([
  {
    $group: {
      _id: "$DESCRIPTION",
      totalPrescriptions: { $sum: 1 }
    }
  },
  { $sort: { totalPrescriptions: -1 } },
  { $limit: 5 },
  {
    $project: {
      _id: 0,
      medication: "$_id",
      totalPrescriptions: 1
    }
  }
])
```

Seminární práce z předmětu NoSQL databáze

Tento dotaz vrací 5 nejčastěji předepisovaných léků na základě počtu záznamů v kolekci medications. Dotaz využívá několik fází agregace, čímž se řadí mezi netriviální MongoDB dotazy.

Výstup:

```
< {
  totalPrescriptions: 601,
  medication: 'insulin isophane human 70 UNT/ML / insulin regular human 30 UNT/ML Injectable Suspension [Humulin]'
}
{
  totalPrescriptions: 461,
  medication: '1 ML Epoetin Alfa 4000 UNT/ML Injection [Epogen]'
}
{
  totalPrescriptions: 456,
  medication: 'lisinopril 10 MG Oral Tablet'
}
{
  totalPrescriptions: 356,
  medication: 'sodium fluoride 0.0272 MG/MG Oral Gel'
}
{
  totalPrescriptions: 302,
  medication: 'Hydrochlorothiazide 25 MG Oral Tablet'
}
```

7.2.6 Vložení embedded dokumentu do dokumentu patients

```
> db.patients.updateOne(
  { PATIENT_ID: "f0f3bc8d-ef38-49ce-a2bd-dfdda982b271" },
  {
    $set: {
      CONTACT: {
        phone: "123-456-7890",
        email: "patient@example.com",
        address: {
          street: "888 Hickley Ferry Suite 38",
          city: "Springfield",
          state: "Massachusetts",
          zip: "1106"
        }
      }
    }
  }
)
```

Seminární práce z předmětu NoSQL databáze

Tento příkaz přidá do dokumentu pacienta vložený (embedded) dokument CONTACT, který obsahuje informace o telefonu, emailu a adrese. Adresa je zároveň složený embedded dokument uvnitř CONTACT, což umožňuje strukturovanější uložení dat.

Embedded dokumenty zlepšují čitelnost a organizaci dat v MongoDB, protože všechny relevantní informace jsou seskupeny uvnitř jednoho záznamu. Takové uspořádání usnadňuje dotazy a aktualizace souvisejících dat bez potřeby složitých join operací.

Výstup:

```
_id: ObjectId('68470eba0b9d75d8379c87e0')
LAST : "Nováková"
BIRTHDATE : "1982-06-21"
PATIENT_ID : "f0f3bc8d-ef38-49ce-a2bd-dfdda982b271"
▼ CONTACT : Object
  phone : "123-456-7890"
  email : "patient@example.com"
  ▼ address : Object
    street : "888 Hickle Ferry Suite 38"
    city : "Springfield"
    state : "Massachusetts"
    zip : "1106"
FIRST : "Jana"
```

7.3 Práce s daty – Insert, Update, Delete, Merge

7.3.1 Vložení nového pacienta

```
> db.patients.insertOne({
  FIRST: "Jana",
  LAST: "Nováková",
  BIRTHDATE: "1982-06-21",
  PATIENT_ID: "custom-uuid-001"
})
```

Používá insertOne, který vloží nový dokument do kolekce patients. Tento pacient bude možné později spojit s dalšími daty (např. procedurami nebo léky). Zároveň je zde uváženo validační schéma – pokud bude porušeno, tak příkaz neprojde.

```
> db.patients.insertOne({ FIRST: "Jan",
  LAST: "Novak"
});
✖ ► MongoServerError: Document failed validation
```

Seminární práce z předmětu NoSQL databáze

Výstup:

```
< {
  acknowledged: true,
  insertedId: ObjectId('68470e170b9d75d8379c87df')
}
```

Výstup:

7.3.2 Aktualizace příjmení pacienta

```
> db.patients.updateOne(
  { PATIENT_ID: "f0f3bc8d-ef38-49ce-a2bd-dfdda982b271" },
  { $set: { LAST: "Novotná" } }
)
```

Používá updateOne k úpravě příjmení záznamu pacienta se zadaným PATIENT_ID. Pomocí \$set se změní pouze jedno pole.

Výstup:

```
< {
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

7.3.3 Smazání pacienta s konkrétním ID

```
> db.patients.deleteOne({ PATIENT_ID: "f0f3bc8d-ef38-49ce-a2bd-dfdda982b271" })
```

Pomocí deleteOne se odstraní první nalezený dokument odpovídající filtru – v tomto případě konkrétní pacient.

Výstup:

```
< {
  acknowledged: true,
  deletedCount: 1
}
```


Seminární práce z předmětu NoSQL databáze

7.3.4 Zvýšení počtu výdejů léku o 1

```
> db.medications.updateMany(  
  { DESCRIPTION: "Mirena 52 MG Intrauterine System" },  
  { $inc: { DISPENSES: 1 } }  
)
```

Tento dotaz zvýší hodnotu DISPENSES o 1 pro všechny dokumenty, kde byl předepsán daný lék. Používá operátor \$inc.

Výstup:

```
< {  
  acknowledged: true,  
  insertedId: null,  
  matchedCount: 5,  
  modifiedCount: 5,  
  upsertedCount: 0  
}
```

7.3.5 Přidání nového pole k dokumentům s hodnotou NULL

```
> db.procedures.updateMany({}, { $set: { doctor: null } })
```

U všech dokumentů v kolekci procedures se přidá nové pole doctor, aby mohlo být později doplněno. Vhodné pro refaktoring schématu.

Výstup:

```
< {  
  acknowledged: true,  
  insertedId: null,  
  matchedCount: 75473,  
  modifiedCount: 75473,  
  upsertedCount: 0  
}
```

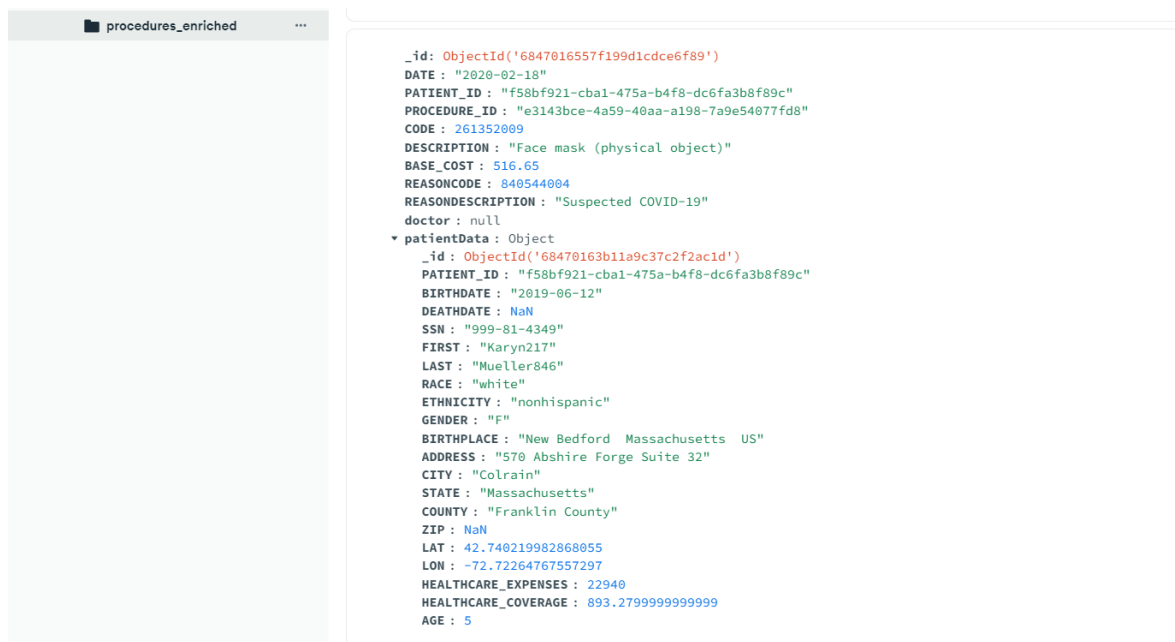
Seminární práce z předmětu NoSQL databáze

7.3.6 Vytvoření nové kolekce s daty původní kolekce procedures a patients

```
> db.procedures.aggregate([
  {
    $lookup: {
      from: "patients",
      localField: "PATIENT_ID",
      foreignField: "PATIENT_ID",
      as: "patientData"
    }
  },
  { $unwind: "$patientData" },
  {
    $out: "procedures_enriched"
  }
])
```

Tento dotaz pomocí \$lookup propojí kolekce procedures a patients podle PATIENT_ID, přidá data o pacientech k jednotlivým zákrokům. \$unwind rozbalí pole patientData na samostatné objekty. Nakonec \$out uloží výsledky do nové kolekce procedures_enriched, kterou přepíše. Takto získáme jednu kolekci s kompletními informacemi o zákrocích i pacientech pro snadnější analýzu.

Výstup:



The screenshot displays the MongoDB Compass interface. On the left, a sidebar shows the 'procedures_enriched' collection. The main area on the right shows a detailed view of a document from this collection. The document contains the following fields:

- `_id`: ObjectId('6847016557f199d1cdce6f89')
- `DATE`: "2020-02-18"
- `PATIENT_ID`: "f58bf921-cba1-475a-b4f8-dc6fa3b8f89c"
- `PROCEDURE_ID`: "e3143bce-4a59-40aa-a198-7a9e54077fd8"
- `CODE`: 261352009
- `DESCRIPTION`: "Face mask (physical object)"
- `BASE_COST`: 516.65
- `REASONCODE`: 840544004
- `REASONDESCRIPTION`: "Suspected COVID-19"
- `doctor`: null
- `patientData`: Object
 - `_id`: ObjectId('68470163b11a9c37c2f2ac1d')
 - `PATIENT_ID`: "f58bf921-cba1-475a-b4f8-dc6fa3b8f89c"
 - `BIRTHDATE`: "2019-06-12"
 - `DEATHDATE`: NaN
 - `SSN`: "999-81-4349"
 - `FIRST`: "Karyn217"
 - `LAST`: "Mueller846"
 - `RACE`: "white"
 - `ETHNICITY`: "nonhispanic"
 - `GENDER`: "F"
 - `BIRTHPLACE`: "New Bedford Massachusetts US"
 - `ADDRESS`: "570 Abshire Forge Suite 32"
 - `CITY`: "Colrain"
 - `STATE`: "Massachusetts"
 - `COUNTY`: "Franklin County"
 - `ZIP`: NaN
 - `LAT`: 42.740219982868055
 - `LON`: -72.72264767557297
 - `HEALTHCARE_EXPENSES`: 22940
 - `HEALTHCARE_COVERAGE`: 893.2799999999999
 - `AGE`: 5

Seminární práce z předmětu NoSQL databáze

7.4 Indexy a výkon

7.4.1 Vytvoření složeného indexu pro pacienty a diagnózu

```
> db.patients.createIndex({ SEX: 1, DIAGNOSE: 1 })
```

Index zrychlí dotazy hledající všechny zákroky konkrétního pacienta. Vhodné např. pro generování zdravotní historie.

Výstup:

```
< SEX_1_DIAGNOSE_1
```

7.4.2 Textový index na diagnózy

```
> db.patients.createIndex({ DIAGNOSE: "text" })
```

Umožňuje fulltextové hledání podle diagnóz – např. vyhledat pacienty, jejichž diagnóza obsahuje „diabetes“ nebo „hypertenze“. Textový index se hodí pro volně psané diagnózy.

Výstup:

```
< DIAGNOSE_text
```

7.4.3 TTL index pro automatické mazání záznamů o medikaci

```
> db.medications.createIndex(  
  { TIMESTAMP: 1 },  
  { expireAfterSeconds: 2592000 }  
)
```

TTL index zajistí, že záznam o medikaci (např. podání léku) bude automaticky odstraněn po 30 dnech. Vhodné pro testovací nebo dočasná data (např. klinické zkoušky).

Výstup:

```
< TIMESTAMP_1
```

7.4.4 Analýza vhodnosti shard klíče

```
db.adminCommand({  
  analyzeShardKey: "medical_records.patients",  
  key: { PATIENT_ID: "hashed" }  
})
```

Příkaz analyzeShardKey slouží k analýze vhodnosti konkrétního pole jako shard klíče v MongoDB. V tomto případě zkoumáme hashovaný klíč PATIENT_ID v kolekci patients. Analýza vyhodnocuje rozložení hodnot a jejich jedinečnost, což pomáhá zjistit, zda je klíč vhodný pro rovnoměrné rozdělení dat mezi shardy. Hashování zajišťuje, že data budou

Seminární práce z předmětu NoSQL databáze

distribuuována rovnoměrně i při nerovnoměrném výskytu hodnot. Výsledkem je doporučení, jestli lze daný shard klíč použít pro efektivní škálování databáze.

Výstup:

```
{
  keyCharacteristics: {
    numDocsTotal: Long('12357'),
    numOrphanDocs: Long('0'),
    avgDocSizeBytes: Long('497'),
    numDocsSampled: Long('12357'),
    isUnique: false,
    numDistinctValues: Long('12354'),
    mostCommonValues: [ [Object], [Object], [Object], [Object], [Object] ],
    monotonicity: { recordIdCorrelationCoefficient: 0, type: 'not monotonic' },
    note: 'Due to performance reasons, the analyzeShardKey command does not filter out orphan documents
when calculating metrics about the characteristics of the shard key. Therefore, if "numOrphanDocs" is
large relative to "numDocsTotal", you may want to rerun the command at some other time to get more
accurate "numDistinctValues" and "mostCommonValues" metrics.'
  },
  readDistribution: {
    sampleSize: {
      total: Long('0'),
      find: Long('0'),
      aggregate: Long('0'),
      count: Long('0'),
      distinct: Long('0')
    }
  },
  writeDistribution: {
    sampleSize: {
      total: Long('0'),
      update: Long('0'),
      delete: Long('0'),
      findAndModify: Long('0')
    }
  },
  ok: 1,
  '$clusterTime': {
    clusterTime: Timestamp({ t: 1749490971, i: 1 }),
    signature: {
      hash: Binary.createFromBase64('1xcf9aBixH6TZTnZwXUMpJFK2pk=', 0),
      keyId: Long('7513975967874809857')
    }
  },
  operationTime: Timestamp({ t: 1749490971, i: 1 })
}
```

7.4.5 Použití hint() pro vynucení indexu

```
db.patients.find({ PATIENT_ID: "f0f3bc8d-ef38-49ce-a2bd-dfdda982b271" }).hint("PATIENT_ID_hashed")
```

Příkaz find() s hint() vynucuje použití konkrétního indexu při vyhledávání dokumentů v kolekci.

Například dotaz db.patients.find({PATIENT_ID:"12345"}).hint("PATIENT_ID_hashed") využívá hashovaný index na poli PATIENT_ID. Tento způsob zajišťuje efektivnější vyhledávání, protože MongoDB přímo použije zvolený index místo automatického výběru. Použití hintu je užitečné zejména při ladění výkonu dotazů.

Seminární práce z předmětu NoSQL databáze

Výstup:

```
< {
  _id: ObjectId('68470eba0b9d75d8379c87e0'),
  LAST: 'Nováková',
  BIRTHDATE: '1982-06-21',
  PATIENT_ID: 'f0f3bc8d-ef38-49ce-a2bd-dfdda982b271'
}
```

7.4.6 Vyhodnocení dotazu pomocí explain()

```
db.patients.find({ PATIENT_ID: "f0f3bc8d-ef38-49ce-a2bd-dfdda982b271" }).explain("executionStats")
```

Tento příkaz v MongoDB vyhledává v kolekci **patients** záznam pacienta podle jeho unikátního **PATIENT_ID**. Pomocí funkce **explain("executionStats")** získáváme detailní statistiky o vykonání dotazu, včetně počtu prohledaných indexů, dokumentů a času potřebného k vyhledání. Díky tomu lze analyzovat výkon dotazu a optimalizovat databázi.

Výstup:

```
{
  "queryPlanner":{
    "winningPlan":{
      "stage":"SINGLE_SHARD",
      "shards":[
        {
          "explainVersion":"1",
          "shardName":"rs-shard-01",
          "connectionString":"rs-shard-01/shard01-a:27017, shard01-b:27017, shard01-c:27017",
          "serverInfo":{
            "host":"aebcbc3b7227",
            "port":27017,
            "version":"8.0.8",
            "gitVersion":"7f52660c14217ed2c8d3240f823a2291a4fe6abd"
          },
          "namespace":"medical_records.patients",
          "parsedQuery":{
            "PATIENT_ID":{
              "$eq":"f0f3bc8d-ef38-49ce-a2bd-dfdda982b271"
            }
          },
          "indexFilterSet":false,
          "queryHash":"471EE59E",
          "planCacheShapeHash":"471EE59E",
          "planCacheKey":"9F1ED66B",
          "optimizationTimeMillis":0,

```

Seminární práce z předmětu NoSQL databáze

```
"maxIndexedOrSolutionsReached":false,
"maxIndexedAndSolutionsReached":false,
"maxScansToExplodeReached":false,
"prunedSimilarIndexes":false,
"winningPlan":{
  "isCached":false,
  "stage":"FETCH",
  "inputStage":{
    "stage":"IXSCAN",
    "keyPattern":{
      "PATIENT_ID":1
    },
    "indexName":"PATIENT_ID_1",
    "isMultiKey":false,
    "multiKeyPaths":{
      "PATIENT_ID":[

    ]
    },
    "isUnique":false,
    "isSparse":false,
    "isPartial":false,
    "indexVersion":2,
    "direction":"forward",
    "indexBounds":{
      "PATIENT_ID":[
        ["f0f3bc8d-ef38-49ce-a2bd-dfdda982b271\","
\ "f0f3bc8d-ef38-49ce-a2bd-dfdda982b271\"]]
      ]
    }
  }
},
"rejectedPlans":[
  {
    "isCached":false,
    "stage":"FETCH",
    "filter":{
      "PATIENT_ID":{
        "$eq":"f0f3bc8d-ef38-49ce-a2bd-dfdda982b271"
      }
    },
    "inputStage":{
      "stage":"IXSCAN",
      "keyPattern":{
        "PATIENT_ID":"hashed"
      },
      "indexName":"PATIENT_ID_hashed",
      "isMultiKey":false,
```

Seminární práce z předmětu NoSQL databáze

```
        "isUnique":false,
        "isSparse":false,
        "isPartial":false,
        "indexVersion":2,
        "direction":"forward",
        "indexBounds":{"PATIENT_ID":[
            "[-346542154798968182, -346542154798968182]"
        ]}
    }
}
]
},
"executionStats":{"nReturned":1,
"executionTimeMillis":0,
"totalKeysExamined":1,
"totalDocsExamined":1,
"executionStages":{"stage":"SINGLE_SHARD",
"nReturned":1,
"executionTimeMillis":0,
"totalKeysExamined":1,
"totalDocsExamined":1,
"totalChildMillis":0,
"shards":[
{
"shardName":"rs-shard-01",
"executionSuccess":true,
"nReturned":1,
"executionTimeMillis":0,
"totalKeysExamined":1,
"totalDocsExamined":1,
"executionStages":{"isCached":false,
"stage":"FETCH",
"nReturned":1,
"executionTimeMillisEstimate":0,
"works":3,
"advanced":1,
"needTime":0,
"needYield":0,
"saveState":0,
"restoreState":0,
```

Seminární práce z předmětu NoSQL databáze

```
"isEOF":1,
"docsExamined":1,
"alreadyHasObj":0,
"inputStage":{"
  "stage":"IXSCAN",
  "nReturned":1,
  "executionTimeMillisEstimate":0,
  "works":2,
  "advanced":1,
  "needTime":0,
  "needYield":0,
  "saveState":0,
  "restoreState":0,
  "isEOF":1,
  "keyPattern":{"
    "PATIENT_ID":1
  },
  "indexName":"PATIENT_ID_1",
  "isMultiKey":false,
  "multiKeyPaths":{"
    "PATIENT_ID":[

    ]
  },
  "isUnique":false,
  "isSparse":false,
  "isPartial":false,
  "indexVersion":2,
  "direction":"forward",
  "indexBounds":{"
    "PATIENT_ID":[
      ["\f0f3bc8d-ef38-49ce-a2bd-dfdda982b271\\"",
        "\f0f3bc8d-ef38-49ce-a2bd-dfdda982b271\"]]
    ]
  },
  "keysExamined":1,
  "seeks":1,
  "dupsTested":0,
  "dupsDropped":0
}
}
}
]
},
"queryShapeHash":"ABD92FA581C43147661FC52A7771CF126F1147D642FC448F4C0036E
EA9637A67",
"serverInfo":{"
```


Seminární práce z předmětu NoSQL databáze

```
"host": "e0be6cc2ea93",
"port": 27017,
"version": "8.0.8",
"gitVersion": "7f52660c14217ed2c8d3240f823a2291a4fe6abd"
},
"serverParameters": {
  "internalQueryFacetBufferSizeBytes": 104857600,
  "internalQueryFacetMaxOutputDocSizeBytes": 104857600,
  "internalLookupStageIntermediateDocumentMaxSizeBytes": 104857600,
  "internalDocumentSourceGroupMaxMemoryBytes": 104857600,
  "internalQueryMaxBlockingSortMemoryUsageBytes": 104857600,
  "internalQueryProhibitBlockingMergeOnMongoS": 0,
  "internalQueryMaxAddToSetBytes": 104857600,
  "internalDocumentSourceSetWindowFieldsMaxMemoryBytes": 104857600,
  "internalQueryFrameworkControl": "trySbeRestricted",
  "internalQueryPlannerIgnoreIndexWithCollationForRegex": 1
},
"command": {
  "find": "patients",
  "filter": {
    "PATIENT_ID": "f0f3bc8d-ef38-49ce-a2bd-dfdda982b271"
  },
  "lsid": {
    "id": "UUID(\"4dbb4985-4db8-4fb4-8da7-05759aac08bd\")"
  },
  "$clusterTime": {
    "clusterTime": "Timestamp({
      \"t\": 1749490967,
      \"i\": 1
    })",
    "signature": {
      "hash": Binary.createFromBase64("sL7y4cvOiNgKa3lBdGcwYJR9PiE=",
        0),
      "keyId": 7513975967874810000
    }
  },
  "$db": "medical_records"
},
"ok": 1,
"$clusterTime": {
  "clusterTime": "Timestamp({
    \"t\": 1749491389,
    \"i\": 3
  })",
  "signature": {
    "hash": Binary.createFromBase64("siepVSx6AIW0C6tL+Hw0liM56o=",
      0),
    "keyId": 7513975967874810000
  }
}
```

Seminární práce z předmětu NoSQL databáze

```
}  
},  
"operationTime":"Timestamp({  
  "t":1749491384,  
  "i":1  
})"  
}
```

7.5 Cluster a konfigurace

7.5.1 Stav shardingu v clusteru

```
> sh.status()
```

Zobrazí přehledný stav shardovaného clusteru MongoDB. Uvidíš informace o jednotlivých shardech, databázích, jejich rozdělení na chunků a nastavení balanceru. Skvělý příkaz pro rychlý přehled o tom, jak jsou data rozložena v clusteru.

Výstup:

```
shardingVersion  
{  
  _id: 1,  
  clusterId: ObjectId('68470154aee0ccb38fdc6405')  
}  
shards  
[  
  {  
    _id: 'rs-shard-01',  
    host: 'rs-shard-01/shard01-a:27017,shard01-b:27017,shard01-c:27017',  
    state: 1,  
    topologyTime: Timestamp({ t: 1749483866, i: 12 }),  
    replSetConfigVersion: -1  
  },  
  {  
    _id: 'rs-shard-02',  
    host: 'rs-shard-02/shard02-a:27017,shard02-b:27017,shard02-c:27017',  
    state: 1,  
    topologyTime: Timestamp({ t: 1749483866, i: 30 }),  
    replSetConfigVersion: -1  
  },  
  {  
    _id: 'rs-shard-03',  
    host: 'rs-shard-03/shard03-a:27017,shard03-b:27017,shard03-c:27017',  
    state: 1,  
    topologyTime: Timestamp({ t: 1749483866, i: 57 }),  
    replSetConfigVersion: -1  
  }  
]
```

Seminární práce z předmětu NoSQL databáze

```
]
active mongoses
[
  {
    '8.0.8': 2
  }
]
autosplit
{
  'Currently enabled': 'yes'
}
balancer
{
  'Currently enabled': 'yes',
  'Failed balancer rounds in last 5 attempts': 0,
  'Currently running': 'no',
  'Migration Results for the last 24 hours': 'No recent migrations'
}
shardedDataDistribution
[
  {
    ns: 'medical_records.medications',
    shards: [
      {
        shardName: 'rs-shard-01',
        numOrphanedDocs: 0,
        numOwnedDocuments: 1730,
        ownedSizeBytes: 851160,
        orphanedSizeBytes: 0
      },
      {
        shardName: 'rs-shard-02',
        numOrphanedDocs: 0,
        numOwnedDocuments: 1974,
        ownedSizeBytes: 975156,
        orphanedSizeBytes: 0
      },
      {
        shardName: 'rs-shard-03',
        numOrphanedDocs: 0,
        numOwnedDocuments: 1292,
        ownedSizeBytes: 639540,
        orphanedSizeBytes: 0
      }
    ]
  },
  {
    ns: 'medical_records.patients',
```

Seminární práce z předmětu NoSQL databáze

```
shards: [  
  {  
    shardName: 'rs-shard-01',  
    numOrphanedDocs: 0,  
    numOwnedDocuments: 4078,  
    ownedSizeBytes: 2026766,  
    orphanedSizeBytes: 0  
  },  
  {  
    shardName: 'rs-shard-02',  
    numOrphanedDocs: 0,  
    numOwnedDocuments: 4095,  
    ownedSizeBytes: 2035215,  
    orphanedSizeBytes: 0  
  },  
  {  
    shardName: 'rs-shard-03',  
    numOrphanedDocs: 0,  
    numOwnedDocuments: 4184,  
    ownedSizeBytes: 2079448,  
    orphanedSizeBytes: 0  
  }  
]  
,  
{  
  ns: 'medical_records.procedures',  
  shards: [  
    {  
      shardName: 'rs-shard-01',  
      numOrphanedDocs: 0,  
      numOwnedDocuments: 24789,  
      ownedSizeBytes: 7684590,  
      orphanedSizeBytes: 0  
    },  
    {  
      shardName: 'rs-shard-02',  
      numOrphanedDocs: 0,  
      numOwnedDocuments: 25672,  
      ownedSizeBytes: 7932648,  
      orphanedSizeBytes: 0  
    },  
    {  
      shardName: 'rs-shard-03',  
      numOrphanedDocs: 0,  
      numOwnedDocuments: 25012,  
      ownedSizeBytes: 7753720,  
      orphanedSizeBytes: 0  
    }  
  ]  
}
```

Seminární práce z předmětu NoSQL databáze

```
]
},
{
  ns: 'config.system.sessions',
  shards: [
    {
      shardName: 'rs-shard-01',
      numOrphanedDocs: 0,
      numOwnedDocuments: 17,
      ownedSizeBytes: 2244,
      orphanedSizeBytes: 0
    }
  ]
}
]
databases
[
  {
    database: {
      _id: 'config',
      primary: 'config',
      partitioned: true
    },
    collections: {
      'config.system.sessions': {
        shardKey: {
          _id: 1
        },
        unique: false,
        balancing: true,
        chunkMetadata: [
          {
            shard: 'rs-shard-01',
            nChunks: 1
          }
        ],
        chunks: [
          {
            min: {
              _id: MinKey()
            },
            max: {
              _id: MaxKey()
            },
            'on shard': 'rs-shard-01',
            'last modified': Timestamp({ t: 1, i: 0 })
          }
        ],
      }
    }
  },
]
```

Seminární práce z předmětu NoSQL databáze

```
    tags: []
  }
},
{
  database: {
    _id: 'medical_records',
    primary: 'rs-shard-01',
    version: {
      uuid: UUID('45865c4b-f86d-4a7b-b657-54d9a78c35dd'),
      timestamp: Timestamp({ t: 1749483874, i: 3 }),
      lastMod: 1
    }
  },
  collections: {
    'medical_records.medications': {
      shardKey: {
        PATIENT_ID: 'hashed'
      },
      unique: false,
      balancing: true,
      chunkMetadata: [
        {
          shard: 'rs-shard-01',
          nChunks: 1
        },
        {
          shard: 'rs-shard-02',
          nChunks: 1
        },
        {
          shard: 'rs-shard-03',
          nChunks: 1
        }
      ],
      chunks: [
        {
          min: {
            PATIENT_ID: MinKey()
          },
          max: {
            PATIENT_ID: -3074457345618258400
          },
          'on shard': 'rs-shard-03',
          'last modified': Timestamp({ t: 1, i: 0 })
        },
        {
          min: {
```

Seminární práce z předmětu NoSQL databáze

```
    PATIENT_ID: -3074457345618258400
  },
  max: {
    PATIENT_ID: 3074457345618258400
  },
  'on_shard': 'rs-shard-01',
  'last modified': Timestamp({ t: 1, i: 1 })
},
{
  min: {
    PATIENT_ID: 3074457345618258400
  },
  max: {
    PATIENT_ID: MaxKey()
  },
  'on_shard': 'rs-shard-02',
  'last modified': Timestamp({ t: 1, i: 2 })
}
],
tags: []
},
'medical_records.patients': {
  shardKey: {
    PATIENT_ID: 'hashed'
  },
  unique: false,
  balancing: true,
  chunkMetadata: [
    {
      shard: 'rs-shard-01',
      nChunks: 1
    },
    {
      shard: 'rs-shard-02',
      nChunks: 1
    },
    {
      shard: 'rs-shard-03',
      nChunks: 1
    }
  ],
  chunks: [
    {
      min: {
        PATIENT_ID: MinKey()
      },
      max: {
        PATIENT_ID: -3074457345618258400
      }
    }
  ]
}
```

Seminární práce z předmětu NoSQL databáze

```
    },
    'on_shard': 'rs-shard-03',
    'last modified': Timestamp({ t: 1, i: 0 })
  },
  {
    min: {
      PATIENT_ID: -3074457345618258400
    },
    max: {
      PATIENT_ID: 3074457345618258400
    },
    'on_shard': 'rs-shard-01',
    'last modified': Timestamp({ t: 1, i: 1 })
  },
  {
    min: {
      PATIENT_ID: 3074457345618258400
    },
    max: {
      PATIENT_ID: MaxKey()
    },
    'on_shard': 'rs-shard-02',
    'last modified': Timestamp({ t: 1, i: 2 })
  }
],
tags: []
},
'medical_records.procedures': {
  shardKey: {
    PATIENT_ID: 'hashed'
  },
  unique: false,
  balancing: true,
  chunkMetadata: [
    {
      shard: 'rs-shard-01',
      nChunks: 1
    },
    {
      shard: 'rs-shard-02',
      nChunks: 1
    },
    {
      shard: 'rs-shard-03',
      nChunks: 1
    }
  ],
  chunks: [
```


Seminární práce z předmětu NoSQL databáze

```
{
  min: {
    PATIENT_ID: MinKey()
  },
  max: {
    PATIENT_ID: -3074457345618258400
  },
  'on shard': 'rs-shard-03',
  'last modified': Timestamp({ t: 1, i: 0 })
},
{
  min: {
    PATIENT_ID: -3074457345618258400
  },
  max: {
    PATIENT_ID: 3074457345618258400
  },
  'on shard': 'rs-shard-01',
  'last modified': Timestamp({ t: 1, i: 1 })
},
{
  min: {
    PATIENT_ID: 3074457345618258400
  },
  max: {
    PATIENT_ID: MaxKey()
  },
  'on shard': 'rs-shard-02',
  'last modified': Timestamp({ t: 1, i: 2 })
}
],
tags: []
}
}
```

7.5.2 Rozložení dat mezi shardy

```
> sh.getShardedDataDistribution()
```

Poskytuje detailní statistiky o tom, jak jsou data rozložena mezi jednotlivými shardy. Pomáhá odhalit nerovnoměrné rozložení dat a zjistit, jestli některý shard není přetížený.

Seminární práce z předmětu NoSQL databáze

Výstup:

```
{
  "ns":"medical_records.patients",
  "shards":[
    {
      "shardName":"rs-shard-01",
      "numOrphanedDocs":0,
      "numOwnedDocuments":4078,
      "ownedSizeBytes":2026766,
      "orphanedSizeBytes":0
    },
    {
      "shardName":"rs-shard-02",
      "numOrphanedDocs":0,
      "numOwnedDocuments":4095,
      "ownedSizeBytes":2035215,
      "orphanedSizeBytes":0
    },
    {
      "shardName":"rs-shard-03",
      "numOrphanedDocs":0,
      "numOwnedDocuments":4184,
      "ownedSizeBytes":2079448,
      "orphanedSizeBytes":0
    }
  ]
}
{
  "ns":"medical_records.medications",
  "shards":[
    {
      "shardName":"rs-shard-01",
      "numOrphanedDocs":0,
      "numOwnedDocuments":1730,
      "ownedSizeBytes":851160,
      "orphanedSizeBytes":0
    },
    {
      "shardName":"rs-shard-02",
      "numOrphanedDocs":0,
      "numOwnedDocuments":1974,
      "ownedSizeBytes":975156,
      "orphanedSizeBytes":0
    },
    {
      "shardName":"rs-shard-03",
      "numOrphanedDocs":0,
      "numOwnedDocuments":1292,
```

Seminární práce z předmětu NoSQL databáze

```
        "ownedSizeBytes":639540,
        "orphanedSizeBytes":0
    }
]
}{
  "ns":"config.system.sessions",
  "shards":[
    {
      "shardName":"rs-shard-01",
      "numOrphanedDocs":0,
      "numOwnedDocuments":17,
      "ownedSizeBytes":2244,
      "orphanedSizeBytes":0
    }
  ]
}{
  "ns":"medical_records.procedures",
  "shards":[
    {
      "shardName":"rs-shard-01",
      "numOrphanedDocs":0,
      "numOwnedDocuments":24789,
      "ownedSizeBytes":7684590,
      "orphanedSizeBytes":0
    },
    {
      "shardName":"rs-shard-02",
      "numOrphanedDocs":0,
      "numOwnedDocuments":25672,
      "ownedSizeBytes":7932648,
      "orphanedSizeBytes":0
    },
    {
      "shardName":"rs-shard-03",
      "numOrphanedDocs":0,
      "numOwnedDocuments":25012,
      "ownedSizeBytes":7753720,
      "orphanedSizeBytes":0
    }
  ]
}
```

7.5.3 Stav balanceru

```
> sh.getBalancerState()
< true
```

Seminární práce z předmětu NoSQL databáze

Vrací aktuální stav balanceru — zda je zapnutý (enabled) nebo vypnutý (disabled). Balancer je proces, který zajišťuje rovnoměrné rozložení chunků mezi shardy.

7.5.4 Informace o běžících routerech

```
> db.adminCommand({ getCmdLineOpts: 1 })
```

Ukáže, s jakými parametry byla spuštěna MongoDB instance. To zahrnuje konfiguraci serveru, jako jsou zapnuté moduly, adresy, porty a další důležité volby při startu.

Výstup:

```
< {
  argv: [
    'mongos',
    '--port',
    '27017',
    '--configdb',
    'rs-config-server/configsvr01:27017,configsvr02:27017,configsvr03:27017',
    '--bind_ip_all',
    '--keyFile',
    '/data/mongodb-keyfile'
  ],
  parsed: {
    net: { bindIp: '*', port: 27017 },
    security: { keyFile: '/data/mongodb-keyfile' },
    sharding: {
      configDB: 'rs-config-server/configsvr01:27017,configsvr02:27017,configsvr03:27017'
    }
  },
  ok: 1,
  '$clusterTime': {
    clusterTime: Timestamp({ t: 1749493270, i: 1 }),
    signature: {
      hash: Binary.createFromBase64('NcrbCkv9oENkkLA1/pZaYpBb+ZU=', 0),
      keyId: Long('7513975967874809857')
    }
  },
  operationTime: Timestamp({ t: 1749493270, i: 1 })
}
```

7.5.5 Ověření chunků a jejich rozložení

```
> use config
< switched to db config
> db.chunks.aggregate([
  { $group: { _id: "$shard", count: { $sum: 1 } } }
])
< {
```

Seminární práce z předmětu NoSQL databáze

Agregační dotaz, který spočítá počet chunků (datových částí) na jednotlivých shardech. Pomáhá ověřit, jak jsou chunk data rozložena napříč shardami.

Výstup:

```
< {  
  _id: 'rs-shard-01',  
  count: 4  
}  
{  
  _id: 'rs-shard-02',  
  count: 3  
}  
{  
  _id: 'rs-shard-03',  
  count: 3  
}
```

Seminární práce z předmětu NoSQL databáze

7.5.6 Stav všech mongos routerů v clusteru

```
> db.adminCommand({ listShards: 1 })
```

Vrátí seznam všech shardů v clusteru včetně jejich ID, adresy a dalších základních informací. Užitečné pro rychlý přehled o dostupných shardech.

Výstup:

```
< {
  shards: [
    {
      _id: 'rs-shard-01',
      host: 'rs-shard-01/shard01-a:27017,shard01-b:27017,shard01-c:27017',
      state: 1,
      topologyTime: Timestamp({ t: 1749483866, i: 12 }),
      replSetConfigVersion: Long('-1')
    },
    {
      _id: 'rs-shard-02',
      host: 'rs-shard-02/shard02-a:27017,shard02-b:27017,shard02-c:27017',
      state: 1,
      topologyTime: Timestamp({ t: 1749483866, i: 30 }),
      replSetConfigVersion: Long('-1')
    },
    {
      _id: 'rs-shard-03',
      host: 'rs-shard-03/shard03-a:27017,shard03-b:27017,shard03-c:27017',
      state: 1,
      topologyTime: Timestamp({ t: 1749483866, i: 57 }),
      replSetConfigVersion: Long('-1')
    }
  ],
  ok: 1,
  '$clusterTime': {
    clusterTime: Timestamp({ t: 1749493063, i: 1 }),
    signature: {
      hash: Binary.createFromBase64('Kt1VEHpKX2sL8jYStUNE7HnL/u0=', 0),
      keyId: Long('7513975967874809857')
    }
  },
  operationTime: Timestamp({ t: 1749493063, i: 1 })
}
```

Seminární práce z předmětu NoSQL databáze

Závěr

Cílem této práce bylo prakticky ověřit možnosti databázového systému MongoDB při zpracování komplexních a variabilních dat v oblasti zdravotnictví. Na základě návrhu a realizace distribuovaného databázového clusteru byly demonstrovány klíčové vlastnosti MongoDB – především sharding, replikace, horizontální škálování a práce s dokumentově orientovanými strukturami.

Pomocí shell skriptů a konfiguračních souborů byla vytvořena robustní architektura, kterou lze snadno spustit a znovu nasadit v prostředí Docker/WSL. Import syntetických dat z CSV souborů a jejich předzpracování pomocí Pythonu navíc ukázaly, jak efektivně propojit MongoDB s nástroji pro analýzu a vizualizaci dat.

Klíčovým přínosem bylo vytvoření sady netriviálních dotazů využívajících pokročilé funkce MongoDB, jako jsou aggregate, lookup, unwind, project, group, sort nebo indexy. Tyto dotazy ilustrují široké možnosti práce s daty – od základního filtrování až po komplexní analýzy a transformace napříč více kolekcemi.

Výsledkem je projekt, který nejen demonstruje praktickou aplikaci MongoDB na konkrétním doménovém problému, ale zároveň poskytuje názorný materiál pro studenty, vývojáře nebo datové analytiky, kteří chtějí získat pevný základ v práci s dokumentově orientovanými databázemi.

Do budoucna by bylo možné projekt dále rozšířit například o webové rozhraní pro přímou práci s daty, automatizované skripty pro import reálných datových zdrojů, nebo hlubší analýzy využívající strojové učení. MongoDB poskytuje k tomu všemu flexibilní a výkonný základ.

Seminární práce z předmětu NoSQL databáze

Zdroje

Aggregation in MongoDB. GeeksforGeeks, [cit. 2025-05-30]. Dostupné z: <https://www.geeksforgeeks.org/aggregation-in-mongodb/>

Built-In Roles—MongoDB Manual. MongoDB, [cit. 2025-05-22]. Dostupné z: <https://www.mongodb.com/docs/manual/reference/built-in-roles/>

Choose a Shard Key—MongoDB Manual. MongoDB, [cit. 2025-05-22]. Dostupné z: <https://www.mongodb.com/docs/manual/core/sharding-choose-a-shard-key/>

Documents—MongoDB Manual. MongoDB, [cit. 2025-05-22]. Dostupné z: <https://www.mongodb.com/docs/manual/core/document/>

GeeksforGeeks. MongoDB – Advantages and Disadvantages, [cit. 30. 5. 2025]. Dostupné z: <https://www.geeksforgeeks.org/mongodb-advantages-disadvantages/>

GEEKSFORGEEKS. Shard Keys in MongoDB. GeeksforGeeks, [cit. 30. 5. 2025]. Dostupné z: <https://www.geeksforgeeks.org/shard-keys-in-mongodb/>

Introduction to MongoDB — MongoDB Manual. MongoDB, [cit. 2025-05-22]. Dostupné z: <https://www.mongodb.com/docs/manual/introduction/>

MINHHUNGIT. mongodb-cluster-docker-compose. GitHub, 2024 [cit. 2025-05-22]. Dostupné z: <https://github.com/minhhungit/mongodb-cluster-docker-compose>

MongoDB Aggregation Framework Tutorials. MongoDB, [cit. 2025-05-30]. Dostupné z: <https://www.mongodb.com/developer/products/mongodb/aggregation-framework/tutorials/>

MongoDB Developer Community. Choosing a shard key. MongoDB Developer Forums, [cit. 30. 5. 2025]. Dostupné z: <https://www.mongodb.com/community/forums/t/choosing-a-shard-key/1105>

MongoDB Documentation. MongoDB, [cit. 2025-05-22]. Dostupné z: <https://www.mongodb.com/docs/>

MongoDB Security Best Practices. Satori Cyber, [cit. 2025-05-30]. Dostupné z: <https://satoricyber.com/mongodb-security/11-mongodb-security-features-and-best-practices/>

Replica Set Members—MongoDB Manual. MongoDB, [cit. 2025-05-22]. Dostupné z: <https://www.mongodb.com/docs/manual/core/replica-set-members/>

Schema Validation—MongoDB Manual. MongoDB, [cit. 2025-05-22]. Dostupné z: <https://www.mongodb.com/docs/manual/core/schema-validation/>

Sharding—MongoDB Manual. MongoDB, [cit. 2025-05-22]. Dostupné z: <https://www.mongodb.com/docs/manual/sharding/>

Seminární práce z předmětu NoSQL databáze

Použité nástroje

Docker Inc. Docker [software]. [cit. 2025-05-30]. Dostupné z: <https://www.docker.com/>

Docker Inc. (2). Docker Compose v2.34.0-desktop.1 [software]. [cit. 2025-05-30]. Dostupné z: <https://docs.docker.com/compose/>

Git SCM. Git [software]. [cit. 2025-05-30]. Dostupné z: <https://git-scm.com/>

GitHub Inc. Git Large File Storage (Git LFS) [software]. [cit. 2025-05-30]. Dostupné z: <https://git-lfs.github.com/>

Microsoft. Microsoft Word [software]. [cit. 2025-05-30]. Dostupné z: <https://www.microsoft.com/cs-cz/microsoft-365/word>

Microsoft. Microsoft OneNote [software]. [cit. 2025-05-30]. Dostupné z: <https://www.microsoft.com/cs-cz/microsoft-365/onenote>

Microsoft. Visual Studio Code [software]. [cit. 2025-05-30]. Dostupné z: <https://code.visualstudio.com>

Microsoft. MongoDB for VS Code Extension [software]. [cit. 2025-05-30]. Dostupné z: <https://marketplace.visualstudio.com/items?itemName=mongodb.mongodb-vscode>

MongoDB Inc. MongoDB Compass [software]. [cit. 2025-05-30]. Dostupné z: <https://www.mongodb.com/products/tools/compass>

Seminární práce z předmětu NoSQL databáze

Přílohy

Inicializace clusteru:

- docker-compose.yml
- mongoInit.sh

Data sety

- patients.csv
- medication.csv
- procedures.csv
- data-preparation.ipynb

Skripty

- auth.js
- import_all.py
- medications-import.py
- patients-import.py
- procedures.import.py
- init-configserver.js
- init-router.js
- init-shard01.js
- init-shard02.js
- init-shard03.js

Ostatní

- příkazy.txt

Obrázky v semestrální práci

- Všechny obrázky použité v semestrální práci byly pořízeny pomocí screenshotu obrazovky buď v samotném řešení databáze nebo v MongoDB Shell