Univerzita Hradec Králové
Fakulta informatiky a managementu

NoSQL Semestrální projekt - MongoDB

Radim Kopřiva

V 16.05.2025

Obsah

Kapitola 1

Úvod

Tato semestrální práce se zabývá návrhem a implementací distribuované NoSQL databázové architektury využívající MongoDB (verze 8.0.9[1]).

1.1 O čem tato práce je

Tato práce představuje praktickou implementaci databázového řešení za použití MongoDB. Především tyto body:

- **Architektura systému** – popis návrhu distribuované databáze
- **Funkční řešení** – implementační detaily
- **Případy užití a Případové studie** – scénáře, pro které je řešení vhodné
- **Výhody a nevýhody MongoDB** – zhodnocení předností a limitů MongoDB

Tato dokumentace se zaměřuje na praktickou implementaci a neobsahuje podrobné teoretické základy použitých technologií.

[1] https://www.mongodb.com/products/updates/version-release

Kapitola 2

Architektura

Tato kapitola detailně popisuje architekturu a implementaci distribuované databáze s využitím MongoDB. Nasazení bylo prováděno a testováno pomocí Docekr Desktop [1]

2.1 Schéma a popis architektury

2.1.1 Návrhová architektura

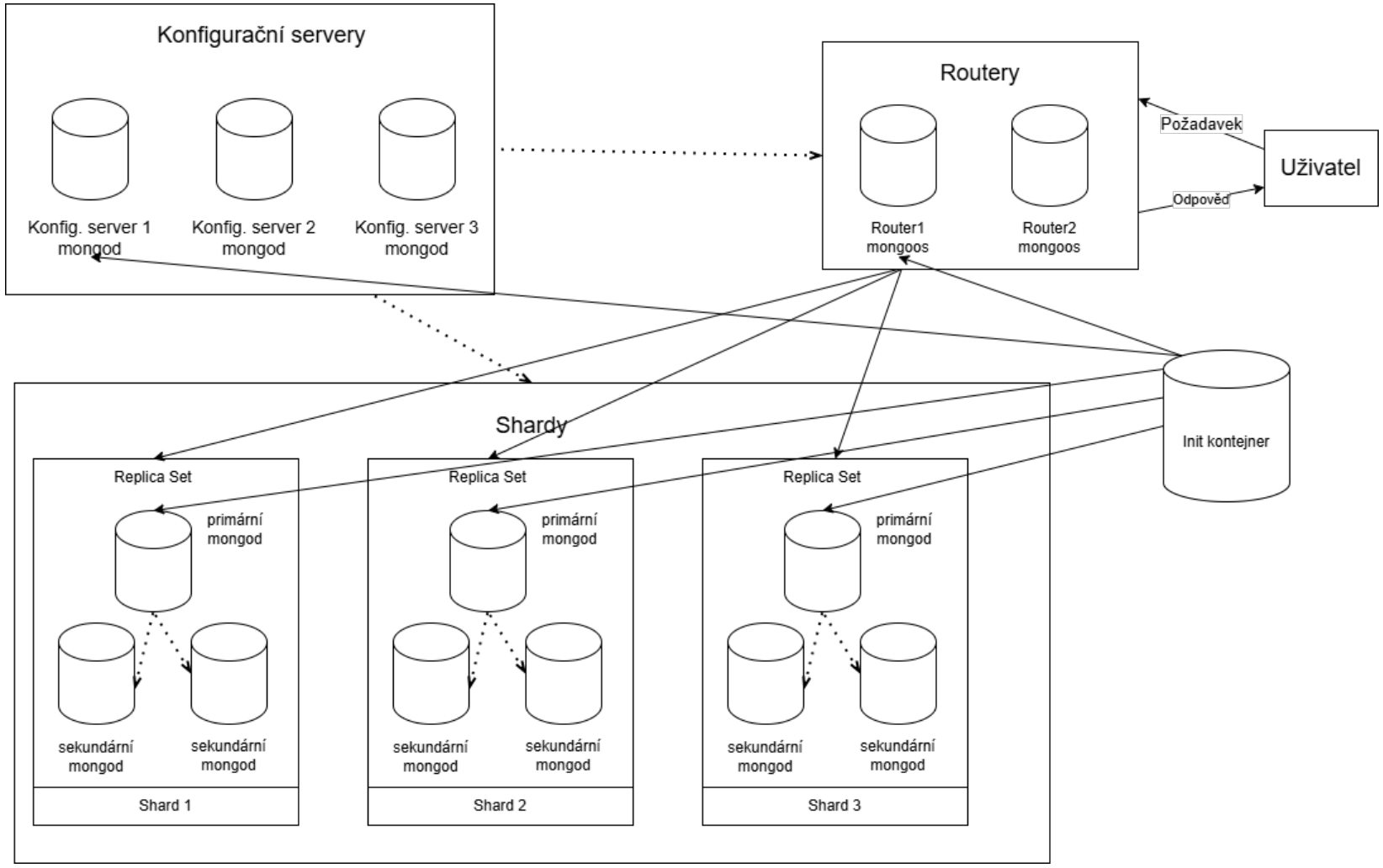
Architektura řešení (viz 7.6) je navržena jako sharded MongoDB cluster který se skládá z následujících komponentů:

- 2 mongos routery (router01, router02) - vstupní body pro klienty
- 3 konfigurační servery (configsvr01-03) - metadata clusteru
- 3 shardy (logické části dat) - každý jako replikační set s 3 nodami
  - Shard 01: shard01-a, shard01-b, shard01-c
  - Shard 02: shard02-a, shard02-b, shard02-c
  - Shard 03: shard03-a, shard03-b, shard03-c
- Inicializační kontejner – pro nastavení clusteru a importu dat

Základní architektura vychází z ověřeného open-source řešení MongoDB Cluster with Docker Compose[1], které bylo lehce upraveno pro potřeby tohoto projektu. Klíčové převzaté komponenty zahrnují:

- Kostru docker-compose.yml – výchozí strukturu služeb
- Autentizační mechanismus – implementaci keyFile autentizace
- Inicializační skripty – základní logiku pro nastartování clusteru

[1] https://github.com/minhhungit/mongodb-cluster-docker-compose/tree/Feature/Auth/with-keyfile-auth



Obrázek 2.1: Schema architektury

2.1.2 Rozdíly

Hlavní rozdíly oproti uvedené vzorové architektuře jsou ve vlastním inicializačním kontejneru, skriptu pro jednoduchou inicializaci s importem dat a aktualizace využívané verze MongoDb z 6.0.1 na 8.0.9. Implementované úpravy byly provedeny pro splnění požadavků zadání, a to maximální automatizaci spuštění a udržení aktuální verze databázového systému.

Inicializační kontejner zajišťuje plnou automatizaci spuštění clusteru tím, že po úspěšném nastartování všech ostatních služeb spouští hlavní inicializační skript, který zajišťuje správné nastavení clusteru - nejprve spouští převzaté konfigurační skripty v daném pořadí, poté nastavuje autentizaci pomocí klíčového souboru a nakonec konfiguruje databázi do které následně importuje data.

2.2 Specifika konfigurace

2.2.1 CAP teorém

Navržené řešení poskytuje následující vlastnosti podle CAP teorému:

Konzistence (C)

Konzistence v kontextu distribuovaných systémů znamená, že všechny uzly v clusteru zobrazují stejná data ve stejném čase, přičemž každé čtení musí vracet nejnovější úspěšně zapsanou hodnotu (strong consistency).

Tato vlastnost je zajištěna prostřednictvím replikačních sad, které se skládají z primárního uzlu a 2 sekundárních uzlů. Společně se správným využitím `read`[2] a `write`[3] obav, které potvrzují správný zápis nebo čtení, je zaručeno čtení vždy nejnovějších dat.

Dostupnost (A)

Dostupnost v distribuovaných systémech znamená schopnost systému zpracovávat požadavky a vracet smysluplné odpovědi i při částečném selhání některých komponent.

Tato vlastnost je částečně zajištěna pomocí replikačních setů a 2 routerů, díky čemuž je dostupnost zajištěna i při výpadku některých uzlů. Avšak při výpadku primárního uzlu v replikační sadě není dostupnost zajištěna, dokud se nezvolí nový primární uzel.

Odolnost vůči rozdělení (P)

Odolnost vůči rozdělení sítě (Partition Tolerance) představuje schopnost systému pokračovat v provozu i při částečném výpadku síťové komunikace mezi jednotlivými uzly clusteru.

[2] https://www.mongodb.com/docs/manual/reference/read-concern/
[3] https://www.mongodb.com/docs/manual/reference/write-concern/

Tato vlastnost je zajištěna pomocí nezávislých replikačních sadám a distribuované konfiguraci, kde každý shard může operovat samostatně i při ztrátě spojení s ostatními částmi clusteru. Nefunkčním se systém stane až po rozpadu na menšiny, kdy nepůje zvolit primární uzel.

Odůvodnění volby CAP vlastností

Navržené řešení primárně cílí na zajištění silné konzistence dat. Toto zaměření na konzistenci přirozeně vede k určitým omezením v dostupnosti během výpadků nebo síťových rozdělení, která jsou však pro danou aplikaci přijatelná.

2.2.2 Cluster

V rámci řešení je využit jeden distribuovaný MongoDB cluster, který se skládá z 2 routerů, 3 konfiguračních serverů a 3 shardy, přičemž každý shard je tvořen 3-členným replika setem.

Cluster je pouze jeden, neboť v řešení, z kterého jsem vycházel, byl také pouze jeden a usoudil jsem že pro účely tohoto projektu bude stačit.

2.2.3 Uzly

V této implementaci se cluster skládá z 14 uzlů, které jsou následovně rozděleny:

- **2 routery** –
- **3 konfigurační servery**
- **9 datových uzlů** – 3 shardy x 3 repliky

Tato konfigurace uzlů je použita, neboť vychází z referenčního řešení, splňuje zadání a je dostatečně robustní.Architektura využívá dva mongos routery pro možné vyvážení zátěže a umožnění pokračování v práci v případě selhání jednoho z nich. Tři konfigurační servery jsou uspořádány do replikační sady, což umožňuje zachování funkčnosti i při selhání jednoho z nich. Cluster obsahuje tři shardy, přičemž každý shard je tvořen tříčlenným replikačním setem, což umožňuje efektivní rozložení dat a odolnost proti výpadkům jednotlivých uzlů.

2.2.4 Sharding

Navržený cluster využívá tři shardy, přičemž každý shard je implementován jako tříčlenný replikační set. Tato konfigurace byla zvolena i přesto, že celkový objem dat (přibližně 6 MB) by se vešel do jediného základního chunku. Pro účely demonstrace a testování funkčnosti shardovacího mechanismu byla proto velikost chunků explicitně nastavena na 1 MB.

2.2.5 Replikace

V rámci clusteru je implementována replikace na čtyřech místech. Každý ze tří shardů je tvořen tříčlenným replikačním setem sestávajícím z jednoho primárního a dvou sekundárních uzlů. Tato konfigurace také umožňuje automatické převzetí služeb při výpadku primárního uzlu tak, že se jeden ze sekundárních stane novým primárním.

Stejný princip replikace je aplikován i na konfigurační servery, kde tři uzly tvoří replikační sadu zajišťující odolnost metadat celého clusteru.

Zvolená konfigurace s tříčlennými replikačními sadami stačí pro účely projektu, neboť dostatečně demonstruje všechny klíčové replikační mechanismy MongoDB.

2.2.6 Perzistence dat

Perzistence dat v této implementaci zajišťuje MongoDB pomocí shardingu a replikace. Tento přístup zajišťuje jak dlouhodobé uchování dat, tak i jejich dostupnost a odolnost vůči výpadkům.

Způsob perzistence

MongoDB perzistentně ukládá data na disk, a to do složek připojených jako svazky `volumes` v Docker kontejnerech. Každý uzel má vlastní svazky, kde jsou uložena data `/data/db` a konfigurační informace `/data/configdb`. Díky tomu jsou data uložena mimo samotný kontejner, což umožňuje jejich zachování i po restartu nebo odstranění kontejneru.

Práce s primární a sekundární pamětí

MongoDB efektivně využívá primární paměť (RAM) pro cacheování dat, která se často používají. To znamená, že často používané dokumenty jsou drženy v paměti, čímž se výrazně urychluje čtení. Data jsou periodicky zapisována na disk (do sekundární paměti), a to prostřednictvím write-ahead logu (WiredTiger journal), což zajišťuje integritu i v případě pádu systému.

Načítání a ukládání dat

Načítání a ukládání dat probíhá přes MongoDB routery, které směrují dotazy na příslušné shardy. Ukládání je zajištěno pomocí primárních uzlů v každém replikačním setu shardu, přičemž změny jsou automaticky asynchronně replikovány na sekundární uzly.

2.2.7 Distribuce dat

Distribuce dat v implementovaném MongoDB clusteru je zajištěna pomocí shardingu a replikace. Data jsou rozložena mezi tři shardy, z nichž každý tvoří samostatný replikační set se třemi uzly (jeden primární, dva sekundární). To

umožňuje jak horizontální škálování, tak vysokou dostupnost a odolnost vůči výpadkům.

Pro účely demonstrace byla velikost chunků snížena na 1 MB, aby došlo k reálnému rozdělení dat mezi jednotlivé shardy i při relativně malém objemu dat.

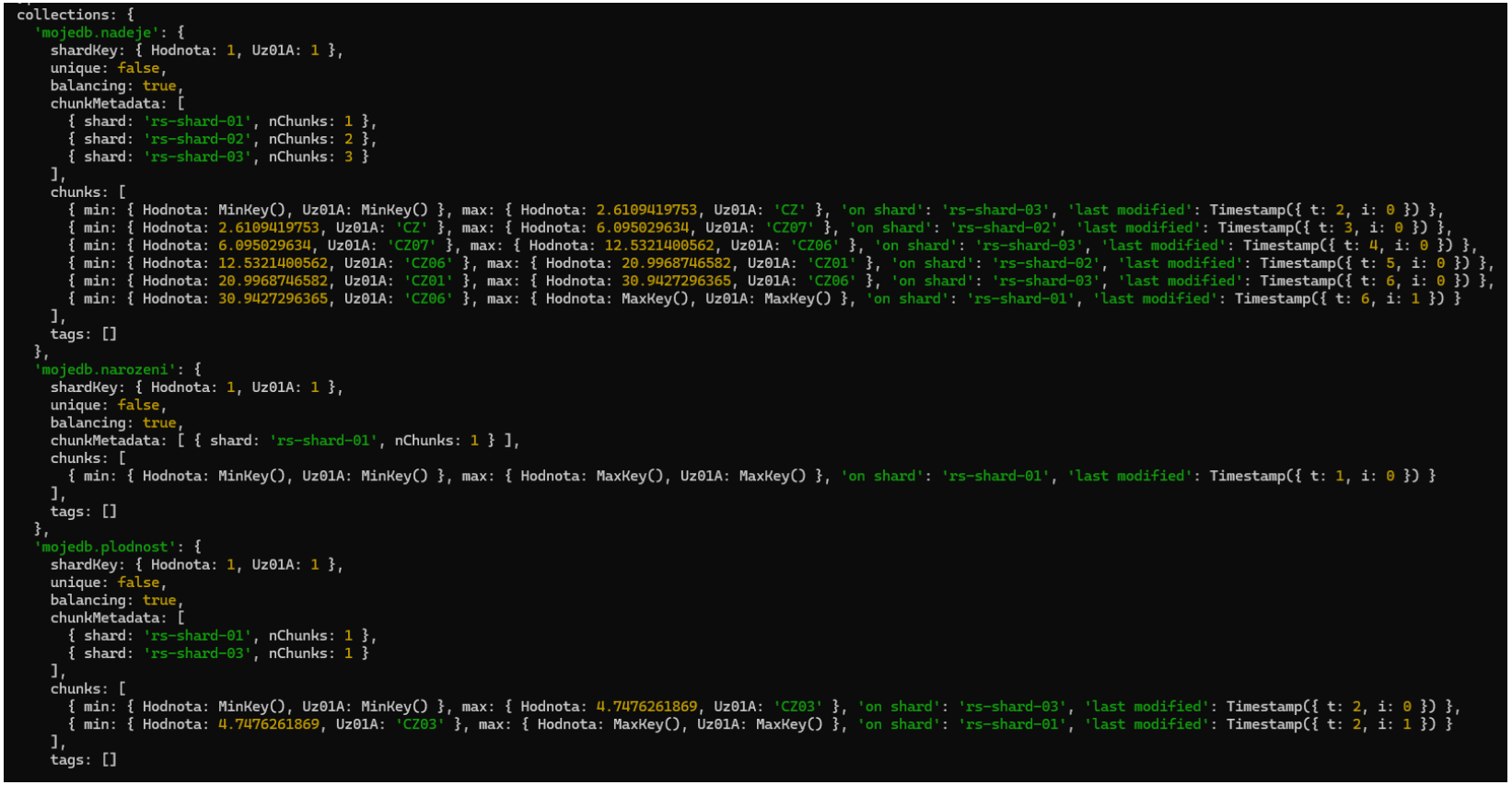
Čtení a zápis probíhají přes dva MongoDB routery, které směrují požadavky na správné shardy. Zápisy probíhají na primární uzly, které poté data replikují na sekundární.

Distribuci dat mezi jednotlivé shardy lze ověřit pomocí administračních nástrojů a konzolových výpisů. Tyto výstupy jsou přiloženy ve formě screenshotů níže, včetně informací o počtu záznamů na jednotlivých uzlech.

Screenshoty a ukázky kódů

Algoritmus 1: Povolení shardingu a nastavení chunksize v MongoDB

Povolení shardingu pro databázi:;
begin
```
sh.enableSharding("mojedb");
```
end
Shardování kolekcí:;
begin
```
sh.shardCollection("mojedb.narozeni", { "Hodnota": 1,
 "Uz01A": 1 });
sh.shardCollection("mojedb.nadeje", { "Hodnota": 1,
 "Uz01A": 1 });
sh.shardCollection("mojedb.plodnost", { "Hodnota": 1,
 "Uz01A": 1 });
```
end
Nastavení chunksize na 1MB:;
begin
```
docker exec router-01 mongosh --username admin
 --password admin --authenticationDatabase admin
 --eval 'db.getSiblingDB("config").settings.updateOne(
{ _id: "chunksize"}, { $set: { value: 1 } }, { upsert:
 true })'
```
end



Obrázek 2.2: Rozložení dat kolekcí

2.2.8 Zabezpečení

Zabezpečení MongoDB clusteru je řešeno pomocí autentizace, autorizace a keyfile ověřování mezi uzly.

Pro interní komunikaci mezi členy clusteru je použit keyfile, který slouží k autentizaci jednotlivých uzlů v rámci replikací a shardingu. Bez správného keyfile se uzel nemůže stát součástí clusteru.

Přístup do databáze je dále chráněn pomocí uživatelské autentizace a při inicializaci clusteru je automaticky vytvořen jeden administrátorský účet, který má následující přihlašovacími údaji:

- **uživatelské jméno**: admin
- **heslo**: admin

Kapitola 3

Funkční řešení

3.1 Struktura

Struktura tohoto projektu je rozdělena do třech hlavních slože:

- Data - složka která obsahuje datasety a python notebook
- Dotazy - složka obsahuje markdown soubor s dotazy
- Funkcni_reseni - složka obsahující funkční řešení projektu

3.1.1 Data

Tato složka obsahuje šest datových souborů ve formátu CSV doplněných o interaktivní Jupyter Python notebook[7]. Notebook slouží jako hlavní analytické prostředí, které:

- Vizualizuje metriky
- Připravuje data pro vložení
- Generuje upravené datasety

Samotné datové soubory se dělí na 3 datasety v původním formátu a 3 upravené, které se používají v databázi.

3.1.2 Funkcni_reseni

Tato složka obsahuje funkční řešení projektu, které se dá rozdělit na tři části:

- Skripty - inicializační skripty
- mongodb-build - skládá se z Dockerfile a key-file
- docker-compose

Skripty

Tato složka obsahuje klíčové skripty pro nastartování a konfiguraci MongoDB clusteru. Hlavním skriptem je `init-cluster.sh`, který zajišťuje celý proces inicializace. Tento skript postupně:

- Pouští převzaté konfigurační skripty z referenčního projektu v předepsaném pořadí
- Inicializuje databázové struktury - vytváří potřebné kolekce
- Nastavuje shardování pro jednotlivé kolekce
- Importuje do databáze data

Skript je navržen tak, aby zajistil automatické nasazení clusteru. Při své činnosti loguje jednotlivé kroky pro potřeby diagnostiky a ověření správného průběhu inicializace.

3.1.3 Docker-compose

Konfigurační soubor docker-compose.yml definuje architekturu MongoDB clusteru s následujícími komponenty:

Routery

Dva `mongos` routery (router01 a router02), které slouží jako vstupní body pro klienty a zajišťují směrování požadavků na příslušné shardy. Routery jsou propojeny se všemi třemi konfiguračními servery (configsvr01-03).

Konfigurační servery

Tři konfigurační servery (configsvr01-03), které tvoří replikační sadu a uchovávají metadata o rozložení dat v clusteru.

Shardy

Tři shardy (shard01-03), přičemž každý shard je implementován jako samostatná replikační sada s jedním primárním a dvěma sekundárními uzly.

Init-container

Speciální inicializační kontejner, který se stará o proces nastartování clusteru. Tento kontejner spouští hlavní inicializační skript init-cluster.sh až poté, co jsou všechny ostatní služby připraveny, čímž zajišťuje správné pořadí inicializace jednotlivých komponent.

Soubor dále definuje vlastní Docker image založenou na MongoDB 8.0, mapování portů pro externí přístup, svazky pro persistentní ukládání dat a síťové propojení mezi jednotlivými komponentami.

3.2 Instalace

Instalaci je doporučené provádět ze složky **Funkcni_reseni** prostřednictvím příkazové řádky. Pro instalaci by pak měl stačit pouze příkaz `docker-compose up`. Doporučeno je však použít následující příkaz:

```
docker-compose up --attach init-container
```

Tímto způsobem se zobrazí logy inicializačního kontejneru, což umožní sledovat průběh inicializace a zjistit, kdy je cluster připraven. Inicializace obvykle trvá přibližně jednu minutu.

Opětovné spuštění kontejneru je možné provést bez logování, protože inicializační skript při pokusu o spuštění nad již inicializovaným clusterem selže. Logování tedy není nutné; je však stále potřeba vyčkat přibližně 30 sekund, než začne cluster opět plně fungovat.

Kapitola 4

Případy užití a případové studie

4.1 Obecné případy užití MongoDB

MongoDB je dokumentově orientovaná NoSQL databáze, která je vhodná zejména pro následující případy užití:

- **Aplikace s dynamickým schématem** — blogy, e-shopy nebo aplikace pro správu uživatelských účtů
- **Mobilní a webové aplikace** — díky snadné integraci a JSON formátu dat.
- **Real-time analytika** -– například sledování uživatelů, streamování dat, monitoring.

4.2 Můj účel

MongoDB jsem si zvolil, neboť umožňuje flexibilní ukládání dokumentů s různými poli, bez nutnosti definovat předem pevné schéma. Každý záznam je možné uložit jako dokument s proměnlivou strukturou a později nad nimi provádět agregace, filtrování či analýzu. MongoDB nám tak umožňuje pracovat s různými formáty vstupních dat bez složitých migrací nebo transformací.

4.3 Jiné databáze

Bylo možné zvolit i jiné databáze, a to Redis, Valkey a Apache Cassandra. Nakonec jsme se rozhodli pro MongoDB, protože lépe odpovídá mému konkrétnímu typu dat a má velmi dobře zpracovaný tutoriál na její zprovoznění.

4.3.1 Redis

Redis je velmi rychlá in-memory databáze, vhodná spíše pro cache a jednoduchá klíč–hodnota data. Nepodporuje složitější datové struktury ani dotazy, které zvolená různorodá a historická data vyžadují.

4.3.2 Valkey

Valkey jakožto fork Redisu sdílí stejné omezení – je optimalizovaný pro rychlost, ne pro flexibilní práci s dokumentově orientovanými nebo polostrukturovanými daty.

4.3.3 Apache Cassandra

Cassandra je vhodná pro masivní škálování, ale má pevnější datový model a omezenou podporu složitých dotazů a agregací. Není vhodná pro náš případ, kde se struktura dat liší mezi záznamy.

4.4 Případové studie

4.4.1 Michelin

- **Odvětví**: Průmyslová výroba a služby
- **Použité produkty**: MongoDB Atlas
- **Use Case**: Content management, katalog produktů

Výchozí stav

Společnost Michelin, globální lídr ve výrobě pneumatik, provozovala vlastní nástroj pro správu hlavních dat s názvem PS9. Ten sloužil jako centrální repozitář produktů a služeb a byl napojen na více než 60 různých aplikací v rámci firmy.

Aplikace PS9 byla provozována jako SaaS řešení u externího poskytovatele. Tento přístup se však ukázal jako neefektivní – vývoj nových funkcí trval až dva roky, i drobné incidenty trvaly více než 20 dní k vyřešení a náklady postupně narůstaly. Problémy vedly k frustraci uživatelů a vývojový tým se musel soustředit na improvizované opravy místo inovací.

Řešení s MongoDB

Michelin se rozhodl modernizovat systém PS9 pomocí MongoDB Atlas, což mu umožnilo využít flexibilní dokumentově orientovaný datový model vhodný pro různé typy dat – strukturovaná i nestrukturovaná. Migrace dat proběhla velmi rychle, během několika hodin, bez problémů. Díky nativnímu konektoru pro Apache Kafka mohl Michelin efektivně zpracovávat události v reálném

čase, například logistiku dopravy pneumatik do 171 zemí. Plná kontrola nad prostředím a snadná škálovatelnost pak umožnily firmě rychle reagovat na aktuální potřeby a zlepšovat výkon aplikace.

Výsledky

Po přechodu na MongoDB Atlas se výrazně zrychlil vývoj nových funkcí – doba implementace se snížila z dvou let na pouhých 20 dní. Výpočty a zpracování dat, které dříve trvaly až 8 hodin, nyní trvají pouze 24 sekund. Podpora systému se zjednodušila díky okamžitému odhalení a opravě chyb, například při výrobě pneumatik. Odstranění improvizovaných oprav a eliminace shadow IT přinesly vyšší kvalitu a spolehlivost dat. Vývojáři tak mohou místo řešení problémů vyvíjet nové funkce, což vede ke zvýšení spokojenosti uživatelů i samotného týmu.[9]

4.4.2 Idealo

- **Odvětví**: Počítačový software a technologie
- **Použité produkty**: MongoDB Professional Services, MongoDB Atlas na AWS
- **Use Case**: Migrace a škálování databáze

Výchozí stav

Idealo je populární německý srovnávač cen, který sjednocuje nabídky od více než 50 000 obchodníků a měsíčně obsluhuje 72 milionů návštěvníků. Za poslední čtyři roky vzrostla návštěvnost šestinásobně a očekává se další růst o 20 % ročně. Původní samosprávná MongoDB databáze běžela on-premises a škálování na zvýšený provoz, zejména v sezónních špičkách, bylo časově i personálně náročné – vyžadovalo 2 až 3 měsíce a vysoké náklady na správu.

Řešení s MongoDB

Idealo se rozhodlo migrovat databázi z vlastního datacentra do MongoDB Atlas na platformě AWS. Výhodou byla schopnost MongoDB Atlas snadno škálovat díky shardingové architektuře, vysoká odolnost, flexibilní dokumentový datový model a integrované služby, které zjednodušily práci s daty.

S podporou MongoDB Professional Services proběhla rychlá migrace a nasazení výpočetních, komunikačních a úložných zdrojů. Díky využití AWS služeb, jako je Amazon Elastic Kubernetes Service (EKS) a AWS Cloud Development Kit (CDK), idealo dále zefektivnilo náklady a zjednodušilo škálování aplikací.

Výsledky

Po migraci do MongoDB Atlas došlo k významnému snížení zdrojů potřebných pro provoz služby – počet shardů se snížil z 25 na 12 a velikost každého nodu

byla zmenšena o 66%. Celkově bylo možné snížit nároky na zdroje o 84%. Díky shardingu se zlepšila redundance a snížila se zátěž na jednotlivé části systému.

Platforma nyní zvládá zpracovat až 200 000 dotazů a 60 000 aktualizací za sekundu. Během jedné sezónní špičky zvládla 150 000 dotazů za sekundu po dobu více než 14 hodin v kuse.

Tento výkon umožnil idealo ušetřit náklady na provoz a eliminovat riziko předimenzování datového centra kvůli sezónním výkyvům. Firma kompletně ukončila provoz vlastního datacentra a běží výhradně na AWS. [6]

4.4.3 FAIRTIQ

- **Odvětví**: Transportace a logistika
- **Použité produkty**: MongoDB Atlas
- **Use Case**: Logistická optimalizace

Výchozí stav

FAIRTIQ, švýcarská technologická společnost, která zpřístupňuje veřejnou dopravu bez nutnosti předem kupovat jízdenky, rychle rostla a potřebovala zajistit 100% dostupnost své služby pro více než 1,3 milionu uživatelů v osmi evropských zemích. Původně používala PostgreSQL pro rychlé nasazení, ale rigidní schéma a nutnost databázových migrací s výpadky brzdily agilní rozvoj. Aby si zachovala flexibilitu a škálovatelnost, přešla na MongoDB, které lépe podporuje mikroservisní architekturu a umožňuje nasazování změn bez nutnosti složitých migrací.

Řešení s MongoDB

FAIRTIQ využívá MongoDB Atlas pro svou mikroservisní architekturu s přístupem „jeden mikroservis – jedna databáze“, což usnadňuje správu dat a vývoj. Pro vysokou dostupnost nasadila cluster s jedním primárním a dvěma sekundárními uzly. Postupně rozšířila počet mikroservis a databází na více než 70. Pro potřeby analytických dotazů přidala dedikovaný analytický uzel, aby nedošlo k narušení produkčního provozu. Po incidentu způsobeném nesprávným směrováním náročných analytických dotazů na sekundární uzly tým FAIRTIQ upgradoval cluster, aktualizoval připojovací řetězce, zavedl timeouty a implementoval datovou federaci, která zamezuje lidským chybám v připojení.

Výsledky

FAIRTIQ si osvojilo důležité poznatky o roli sekundárních uzlů, které i přesto, že jsou read-only, zapisují data pro replikaci a mohou být zdrojem výkonnostních problémů. Správné nastavení read preference a zavedení datové federace zvýšilo stabilitu a zabezpečení provozu. Díky MongoDB Atlas dosahuje FAIRTIQ vysoké dostupnosti, flexibility a škálovatelnosti, které podporují rychlý růst a expanzi na nové trhy. [5]

Kapitola 5

Výhody a nevýhody

5.1 Výhody MongoDB

- **Flexibilita schématu**: MongoDB umožňuje ukládat data bez pevného schématu, což usnadňuje rychlý vývoj a úpravy datových struktur bez nutnosti složitých migrací.
- **Škálovatelnost**: Podpora horizontálního škálování pomocí sharding umožňuje snadno rozdělit data na více uzlů a zvládat velké objemy dat i vysokou zátěž.
- **Vysoká dostupnost**: Replikace dat mezi uzly zajišťuje redundanci a minimalizuje výpadky služeb.
- **Bohaté dotazovací možnosti**: MongoDB podporuje komplexní dotazy, agregace a indexování.

5.2 Nevýhody MongoDB

- **Konzistence dat**: V některých konfiguracích může být silná konzistence dat komplikovaná, což může být nevýhoda u transakčně náročných aplikací.
- **Složitější správa**: Při velkém nasazení je potřeba dbát na správné nastavení replikace, sharding, monitorování výkonu a bezpečnosti.
- **Méně standardizované dotazy**: Na rozdíl od SQL nemusí být dotazy standardizované, což může komplikovat přenositelnost mezi systémy.

5.3 Výhody mého řešení

- **Automatizované nasazení**: Jednoduché spuštění a inicializace clusteru pomocí `docker-compose`

- **Dostupnost**: Replikační sady s automatickým failoverem zajišťují provozuschopnost i při výpadku jednotlivých uzlů.

5.4 Nevýhody mého řešení

- **Závislost na Dockeru**: Celé řešení je vázáno na Docker ekosystém.
- **Nadměrná složitost**: Navržená architektura se 3 shardy a replikačními sadami je příliš komplexní vzhledem k aktuálnímu objemu dat (6 MB dat – 69 tisíc záznamů)

Kapitola 6

Další specifikace

Mé řešení v zásadě následuje doporučenou konfiguraci MongoDB clusteru[1] bez výraznějších modifikací. Jediné dvě úpravy oproti standardnímu nastavení jsou:

- Úprava velikosti chunků na 1 MB pro demonstraci shardování. Při základním nastavení byl pro všechny data použit pouze jeden chunk.
- Implementace speciálního inicializačního kontejneru, který zajišťuje konfiguraci clusteru.

[1] Použitá referenční konfigurace: https://github.com/minhhungit/mongodb-cluster-docker-compose/tree/Feature/Auth/with-keyfile-auth[1]

Kapitola 7

Data

Datasety jsem vybíral z Národního katalogu otevřených dat[1], a to:

- Narození - vybrané souhrnné údaje: https://data.gov.cz/datov%C3%A1-sada?iri=https%3A%2F%2Fdata.gov.cz%2Fzdroj%2Fdatov%C3%A9-sady%2F00025593%2F5948c8558e7dcc7b75e89fe13c26dcb0
- Naděje dožití - ČR a regiony soudržnosti: https://data.gov.cz/datov%C3%A1-sada?iri=https%3A%2F%2Fdata.gov.cz%2Fzdroj%2Fdatov%C3%A9-sady%2F00025593%2Fbfe32ca003b860395f09b506d07ff665
- Míry plodnosti žen podle věku: https://data.gov.cz/datov%C3%A1-sada?iri=https%3A%2F%2Fdata.gov.cz%2Fzdroj%2Fdatov%C3%A9-sady%2F00025593%2Fb5094f4457e365b7540df88999d2ccb8

7.1 Typy dat

Formát dat je csv a databáze pracuje s následujícími typy dat:

- Číselné hodnoty (float, int) – např. hodnoty, roky, některé indikátory.
- Řetězce (string) – např. názvy regionů, ukazatele, zkratky regionů.

7.2 Nakládání s daty v databázi

Pro každou datovou sadu je v databázi vytvořena samostatná kolekce, nad kterou je definováno validační schéma. Tato schémata vycházejí ze společného základu, který reflektuje sdílené atributy napříč datasety (například `Ukazatel`, `Roky`, `Hodnota` apod.). Pro každou kolekci je pak toto základní schéma upraveno podle specifik dané datové sady. Kolekce jsem zvolil, neboť umožňují jednodušší správu dat, jasné oddělení různých typů údajů a přehlednější validaci pomocí

[1] https://data.gov.cz/datov%C3%A9-sady

specifických schémat pro každou kolekci. Alternativní přístup, jako je uložení všech záznamů do jediné kolekce, by vedl k méně přehlednému schématu a obtížnějšímu ladění validace.

7.3 Informace o datech

Datasety mají celkovou velikost přibližně 6 MB a obsahují dohromady přibližně 69 000 záznamů. Neobsahují žádné prázdné hodnoty.

Při zpracování dat došlo k několika úpravám. Byl odstraněn sloupec `CasR`, který byl redundantní, neboť obsahoval stejné hodnoty jako sloupec `Roky`. V datasetech `narozeni` a `plodnost` byl navíc vygenerován nový sloupec `Uz01A` na základě hodnot ve sloupci `Uz012`, a to za účelem jednoduššího propojování jednotlivých datových sad.

Podrobnější informace o zpracování a úpravě dat jsou uvedeny v Python notebooku.

7.3.1 Vizualizace informací

	Ukazatel	IndicatorType	ČR, regiony	Uz01A	Roky	Pohlaví	POHZM	Věk (roky)	VEK1UT	Hodnota
count	43884	43884	43884	43884	43884.000000	43884	43884.000000	43884.000000	4.388400e+04	43884.000000
unique	1	1	9	9	NaN	2	NaN	NaN	NaN	NaN
top	Naděje dožití (ex)	5411UT	Česko	CZ	NaN	ženy	NaN	NaN	NaN	NaN
freq	43884	43884	4876	4876	NaN	21942	NaN	NaN	NaN	NaN
mean	NaN	NaN	NaN	NaN	2012.000000	NaN	1.500000	52.500000	4.096758e+14	31.258511
std	NaN	NaN	NaN	NaN	6.633325	NaN	0.500006	30.598551	3.885930e+12	24.311403
min	NaN	NaN	NaN	NaN	2001.000000	NaN	1.000000	0.000000	4.000006e+14	1.282733
25%	NaN	NaN	NaN	NaN	2006.000000	NaN	1.000000	26.000000	4.100266e+14	7.823007
50%	NaN	NaN	NaN	NaN	2012.000000	NaN	1.500000	52.500000	4.100531e+14	27.087884
75%	NaN	NaN	NaN	NaN	2018.000000	NaN	2.000000	79.000000	4.100796e+14	52.159219
max	NaN	NaN	NaN	NaN	2023.000000	NaN	2.000000	105.000000	4.201058e+14	84.157403

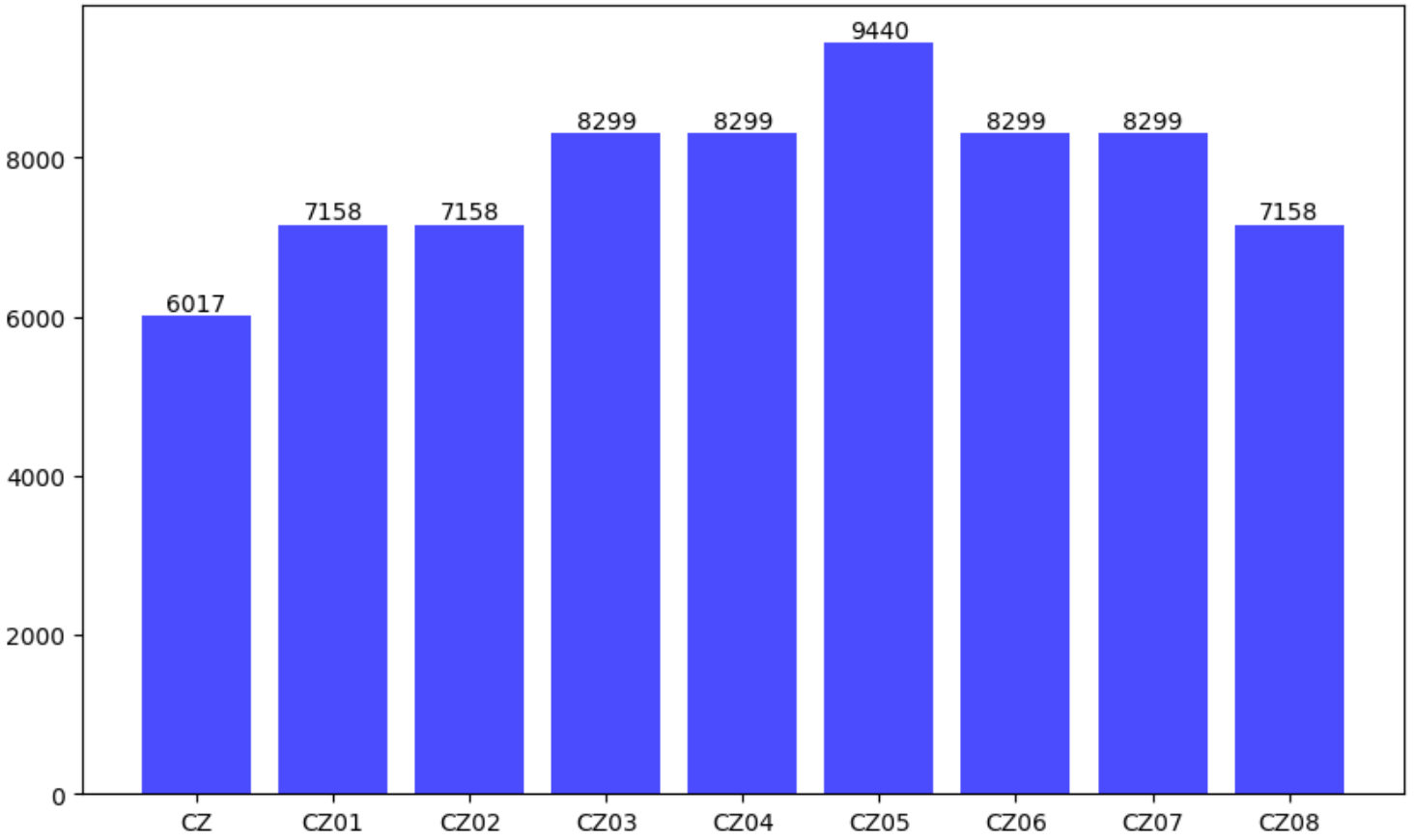
Obrázek 7.1: Informace o datasetu naděje

	Ukazatel	IndicatorType	Roky	Oblast	Uz012	Hodnota	Uz01A
count	7728	7728	7728.000000	7728	7728	7728.000000	7728
unique	14	14	NaN	23	23	NaN	9
top	Narození	4355	NaN	Česko	CZ	NaN	CZ05
freq	552	552	NaN	336	336	NaN	1344
mean	NaN	NaN	2011.500000	NaN	NaN	4381.727223	NaN
std	NaN	NaN	6.922634	NaN	NaN	10616.565241	NaN
min	NaN	NaN	2000.000000	NaN	NaN	1.072804	NaN
25%	NaN	NaN	2005.750000	NaN	NaN	30.810030	NaN
50%	NaN	NaN	2011.500000	NaN	NaN	1491.000000	NaN
75%	NaN	NaN	2017.250000	NaN	NaN	5510.000000	NaN
max	NaN	NaN	2023.000000	NaN	NaN	119842.000000	NaN

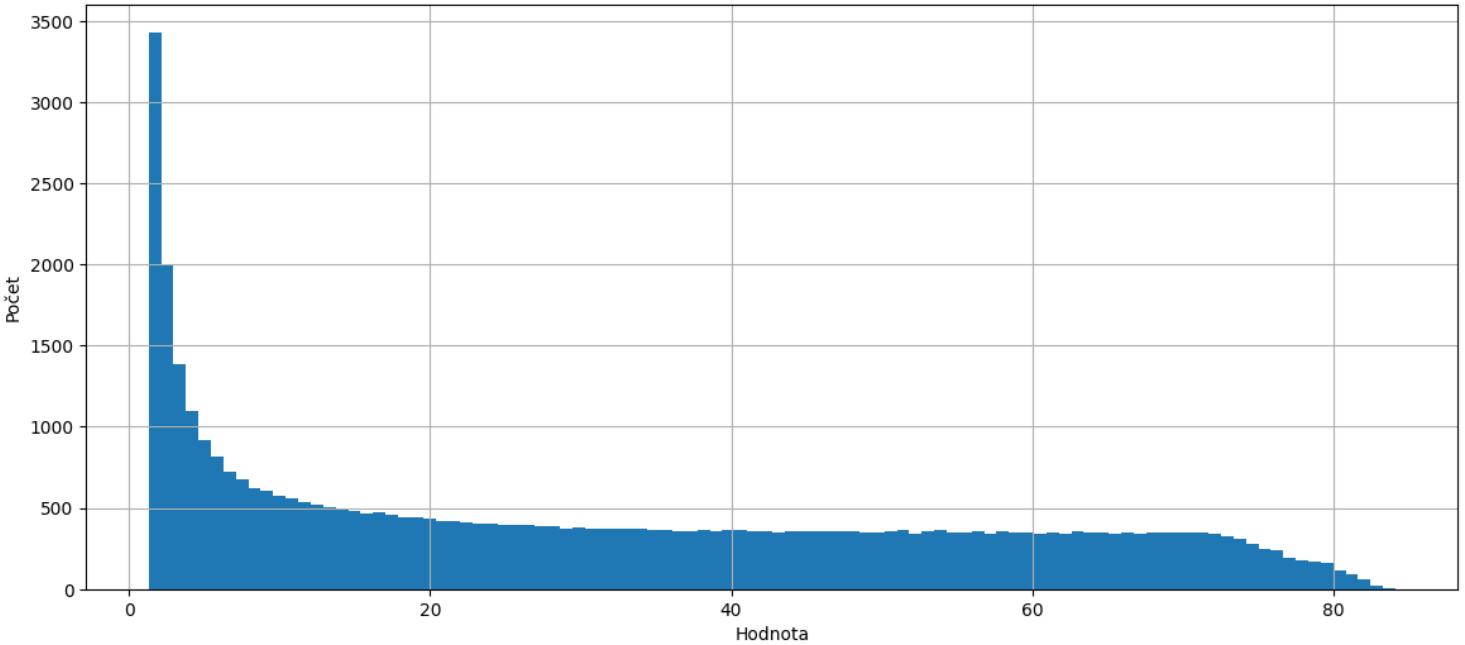
Obrázek 7.2: Informace o datasetu narození

	Ukazatel	IndicatorType	Roky	Oblast	Uz012	Věk (jednoleté skupiny)	VEKZEN1PLOD	Hodnota	Uz01A
count	18515	18515	18515.000000	18515	18515	18515.000000	1.851500e+04	18515.000000	18515
unique	1	1	NaN	23	23	NaN	NaN	NaN	9
top	Míry plodnosti	5406	NaN	Česko	CZ	NaN	NaN	NaN	CZ05
freq	18515	18515	NaN	805	805	NaN	NaN	NaN	3220
mean	NaN	NaN	2012.000000	NaN	NaN	32.000000	4.100326e+14	42.206040	NaN
std	NaN	NaN	6.633429	NaN	NaN	10.099778	1.009979e+10	41.198390	NaN
min	NaN	NaN	2001.000000	NaN	NaN	15.000000	4.100156e+14	0.000000	NaN
25%	NaN	NaN	2006.000000	NaN	NaN	23.000000	4.100236e+14	3.881731	NaN
50%	NaN	NaN	2012.000000	NaN	NaN	32.000000	4.100326e+14	29.616725	NaN
75%	NaN	NaN	2018.000000	NaN	NaN	41.000000	4.100416e+14	74.680895	NaN
max	NaN	NaN	2023.000000	NaN	NaN	49.000000	4.100496e+14	165.510406	NaN

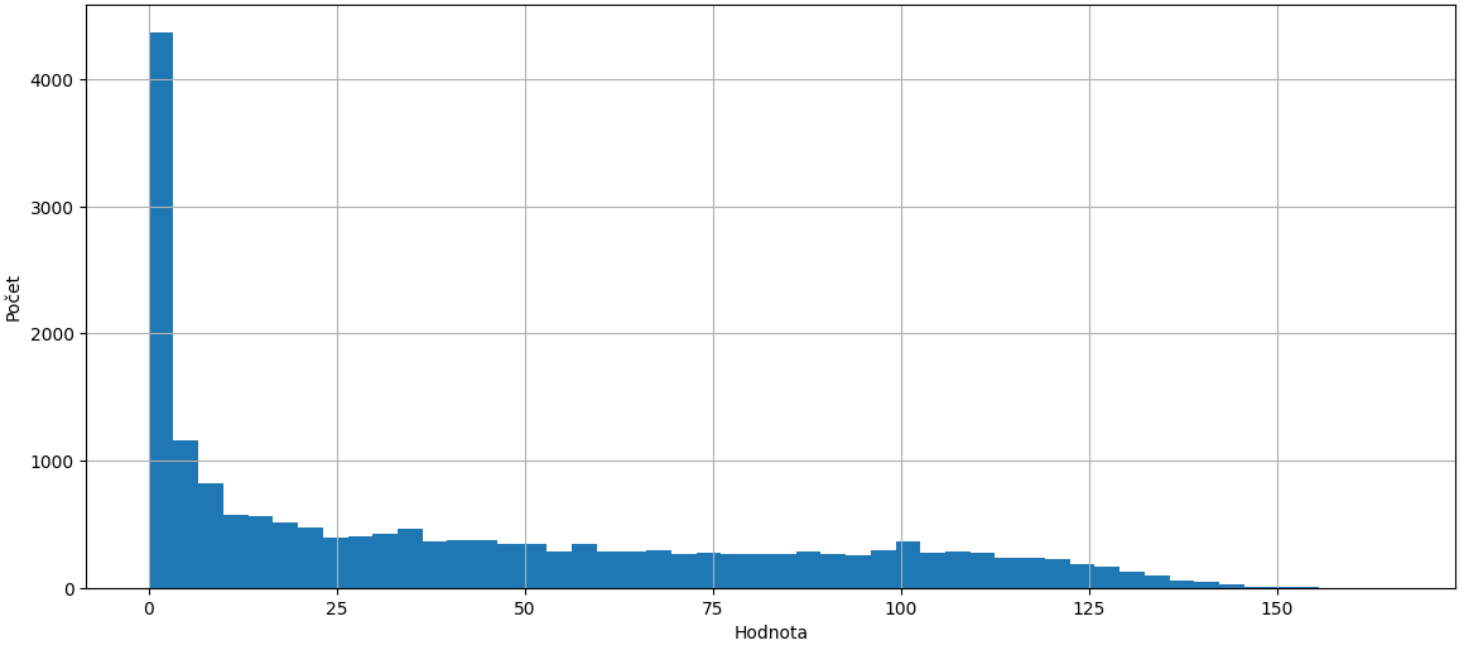
Obrázek 7.3: Informace o datasetu plodnost



Obrázek 7.4: Počet celkových záznamů pro regiony



Obrázek 7.5: Histogram o naději dožití



Obrázek 7.6: Histogram o plodnosti

Kapitola 8

Dotazy

Dotazy včetně jejich popisu jsou uvedeny v přiloženém `markdown` souboru (`markdown.md` a dělí se na pět kategorií:

- Dotazy pracující s daty (insert, update, delete, merge)
- Agregační funkce
- Konfigurační dotazy
- Nested (embedded) dokumenty
- Indexy

Kapitola 9

Závěr

Semestrální práce představuje funkční implementaci distribuovaného MongoDB clusteru s podporou shardování a replikace. Řešení by mělo splňovat všechny stanovené požadavky a poskytlo praktické zkušenosti s návrhem složitějšího databázového systému.

Řešení sice neobsahuje věci navíc, ale stejně si myslím, že silnější stránkou je automatická konfigurace pomocí inicializačního kontejneru a skriptu, který by však mohl být zároveň rozšířen o detekování opětovného spouštění clusteru.

Touto práce jsem si dokázal schopnost implementovat distribuovaný databázový systém využívající MongoDB.

Zdroje

[1] *Demo MongoDB (6.0.1) Sharded Cluster with Docker Compose.* [cit. 18.05.2025]. Dostupné z: https://github.com/minhhungit/mongodb-cluster-docker-compose/tree/Feature/Auth/with-keyfile-auth.

[2] *Docker Compose.* [cit. 19.05.2025]. Dostupné z: https://docs.docker.com/compose/.

[3] *Docker-desktop.* [cit. 19.05.2025]. Dostupné z: https://www.docker.com/products/docker-desktop/.

[4] *Docker.* [cit. 19.05.2025]. Dostupné z: https://www.docker.com/.

[5] *FAIRTIQ secures high availability and flexible performance with MongoDB.* [cit. 19.05.2025]. Dostupné z: https://www.mongodb.com/solutions/customer-case-studies/fairtiq.

[6] *Idealo manages 6x more traffic during seasonal peaks with MongoDB.* [cit. 19.05.2025]. Dostupné z: https://www.mongodb.com/solutions/customer-case-studies/idealo.

[7] *Jupyter Notebook.* [cit. 19.05.2025]. Dostupné z: https://jupyter.org/.

[8] *Everything you need to learn Markdown.* [cit. 19.05.2025]. Dostupné z: https://www.markdownguide.org/.

[9] *Michelin supercharges its data management application with MongoDB.* [cit. 19.05.2025]. Dostupné z: https://www.mongodb.com/solutions/customer-case-studies/michelin.

[10] *MongoDB.* [cit. 19.05.2025]. Dostupné z: https://www.mongodb.com/.

[11] *Naděje dožití - ČR a regiony soudržnosti.* [cit. 19.05.2025]. Dostupné z: https://data.gov.cz/datov%C3%A1-sada?iri=https%3A%2F%2Fdata.gov.cz%2Fzdroj%2Fdatov%C3%A9-sady%2F00025593%2Fbfe32ca003b860395f09b506d07ff665.

[12] *Narození - vybrané souhrnné údaje.* [cit. 19.05.2025]. Dostupné z: https://data.gov.cz/datov%C3%A1-sada?iri=https%3A%2F%2Fdata.gov.cz%2Fzdroj%2Fdatov%C3%A9-sady%2F00025593%2F5948c8558e7dcc7b75e89fe13c26dcb0.

[13] *Míry plodnosti žen podle věku*. [cit. 19.05.2025]. Dostupné z: https://data.gov.cz/datov%C3%A1-sada?iri=https%3A%2F%2Fdata.gov.cz%2Fzdroj%2Fdatov%C3%A9-sady%2F00025593%2Fb5094f4457e365b7540df88999d2ccb8.

Kapitola 10

Přílohy

Data

Složka `Data` obsahuje kompletní datové soubory použité v projektu:

- 6 datasetů ve formátech CSV
- Interaktivní Python notebook

Dotazy

Složka `Dotazy` obsahuje:

- Soubor `dotazy.md` s kompletní sadou dotazů

Funkční řešení

Složka `Funkční_řešení` obsahuje:

- Soubor `docker-compose.yml`
- Složku `scripts` obsahující:
  - `init-cluster.sh` - hlavní inicializační skript
  - Konfigurační skripty pro jednotlivé komponenty, využívané `init-cluster.sh`
- Složku `mongodb-build` obsahující:
  - `Dockerfile`
  - Podsložku s autentizačním key-file
- Jednoduchý návod pro spuštění v souboru `README.md`