Univerzita Hradec Králové

Fakulta informatiky a managementu

NoSQL databáze dokumentová – MongoDB

Bc. Adéla Leppeltová

2023/24

seminární práce předmětu NoSQL databáze

Hradec Králové 2024

Obsah

Úvod

Semestrální práce je zaměřena na NoSQL databázi MongoDB. Práce se zabývá základními principy databáze MongoDB, jejími výhodami a nevýhodami, možnostmi distribuce dat a replikace, porovnáním s jinými databázemi, a praktickými příklady použití. Dále uvádí popis implementace MongoDB pomocí aplikace Docker.

V práci se čtenář dozví o základních principech MongoDB, možnostech distribuce dat a datových strukturách. Dále je v práci popsána architektura, replikace a typy clusteru. Také se čtenář dozví, jaké jsou výhody a nevýhody vybrané databáze a jaké je její konkrétní použití. V praktické části je popsán soubor docker-compose.yml, vybraný data set a konkrétní příkazy pro nastavení, načtení dat a manipulaci s daty.

Vzhledem k tématu práce by mohlo být ještě popsáno prostředí Mongo Express, které slouží pro jednoduché prohlížení a správu dat. Dále by mohla být součástí ukázka integrace MongoDB s různými platformami, například Google Cloud Platform.

Cílem práce je poskytnout přehled o databázi MongoDB, jejích vlastnostech a praktickém využití.

Teoretická východiska

Základní principy

MongoDB je open-source dokumentově orientovaná NoSQL databáze, která nabízí flexibilitu a vysoký výkon pro ukládání a správu dat. Vznikla v roce 2009 a aktuální verze je 7.0. Základním principem MongoDB je ukládání dat jako dokumenty BSON (binární JSON). Díky tomu umožňuje snadnou manipulaci s daty. MongoDB také využívá replikaci, která databázi dělá vysoce dostupnou a odolnou proti poruchám. Rovněž podporuje sharding, který umožňuje zvýšit výkon a kapacitu databáze.

MongoDB se běžně používá pro aplikace, které vyžadují flexibilitu, vysoký výkon a škálovatelnost. Díky svým funkcím je ideální pro nasazení webových, mobilních či IoT aplikací. (1)

Obecné chování

MongoDB v porovnání s relačními databázemi nabízí flexibilní schéma. To je vhodné pro aplikace, které vyžadují schopnost reagovat na měnící se požadavky na data.

Data v MongoDB jsou ukládána ve formátu BSON (binární JSON). Toto uspořádání umožňuje uložení složitých datových struktur. Na rozdíl od relačních databází, které mohou mít problémy s komplexními vztahy mezi daty.

MongoDB podporuje horizontální škálování pomocí shardingu. Oproti tomu relační databáze jsou lépe uzpůsobeny vertikálnímu škálování – například přidání více RAM nebo úložiště.

V porovnání s ostatními NoSQL databázemi, jako jsou například grafové databáze nebo sloupcové databáze, které se specializují na určité použití, MongoDB poskytuje komplexní a flexibilní řešení pro různé potřeby. (2)

Možnosti distribuce dat, datové struktury

Doporučený typ distribuce dat

Distribuce dat je klíčovým aspektem správy rozsáhlé a složité databáze. S rostoucím objemem dat je obtížné je efektivně ukládat a spravovat. MongoDB distribuci dat nabízí pomocí shardingu.

Sharding je metoda distribuce dat na více serverů (shardů). Používá se pro zpracování velkého množství dat. Zahrnuje rozdělení datové sady na menší, lépe spravovatelné části – shardy. Každý shard je samostatnou nezávislou databází. Když se klient dotazuje databáze, MongoDB přesměruje požadavek na příslušný shard. Rozložením dat na více shardů se zlepšuje odezva databáze. (3)

Datové struktury

MongoDB pracuje s různými typy datových struktur, to poskytuje flexibilitu při práci s daty. Základními datovými strukturami, se kterými MongoDB pracuje jsou databáze, kolekce, pohledy a dokument.

Databáze je fyzický kontejner určený pro jednotlivé kolekce. Je nejhrubším dělením MongoDB. Každá databáze musí mít unikátní název, který je case-insensitive a má maximální délku 64 znaků. Databáze vzniká automaticky při vložení první kolekce do databáze. Automaticky zaniká po odebrání poslední kolekce z databáze. (4)

Kolekce reprezentuje skladiště dokumentů, do kterého můžeme přidávat a odebírat jednotlivé dokumenty. Je založena v rámci jedné databáze a jednotlivé dokumenty v kolekci nemusí být stejné. Kolekce nemají žádnou danou strukturu a mohou být vytvářeny jako kapacitně neomezené nebo omezené. (4)

Pohledy (views) slouží pro vytvoření specifického pohledu na data. Jsou určeny pouze pro čtení. (5)

Dokument představuje objekt a množinou hodnot typu klíč – hodnota. Maximální velikost jednoho dokumentu je omezena na 16 MB. Jednotlivé dokumenty jsou ukládány v datovém typu BSON (binární JSON). Umožňují ukládat strukturu. Podstatnou vlastností dokumentových databází je vytváření a používání vyhledávacích indexů.

Jednotlivé hodnoty dokumentu mohou být těchto datových typů:

- Pole (array) –pole hodnot v dokumentech, speciálním typem je pole s objekty (do jedné hodnoty lze vložit množinu dalších dokumentů – vnořené dokumenty)
- ObjectID – slouží jako identifikátor a je generovaný automaticky
- Textový řetězec (string) – nejčajtěji používaným datovým typem, je zakódován pomocí UTF-8
- Datum (date) – datový typ pro ukládání kompletního datumu, jde o 64bitové celé číslo
- Boolean – hodnota typu pravda/nepravda (true/false)
- Integer – celé číslo, v závislosti na verzi serveru může být 32bitové nebo 64bitové
- Double – reálné číslo s plovoucí tečkou
- Null – prázdná hodnota (6)

Sharding

MongoDB podporuje horizontální škálování prostřednictvím shardingu. Použití shardingu je efektivní způsob, jak zajistit rychlou odezvu při práci s velkými objemy dat. (7) Poskytuje také funkce, jako je indexování, ukládání do mezipaměti a úložiště v paměti, které přispívají k jeho vysokému výkonu.

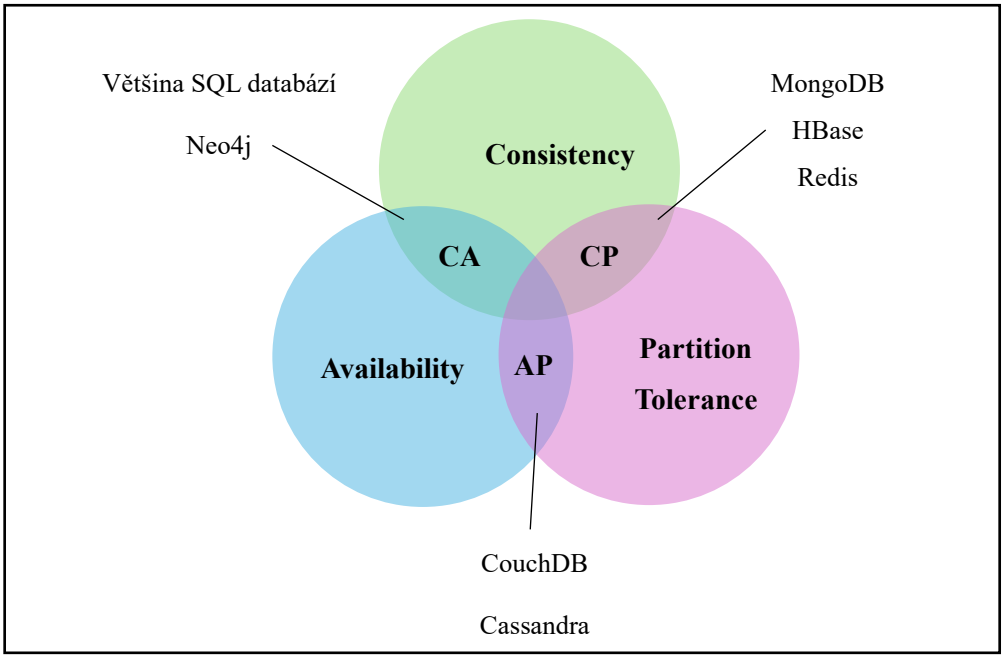
Replikace

Replikace v MongoDB je proces, který zajišťuje vysokou dostupnost a odolnost proti poruchám dat v databázi. Tento proces umožňuje duplikovat data mezi několika servery, které tvoří replikační skupinu. (8)

CAP teorém

Zkratka CAP se skládá z následujících vlastností:

- **Consistency** (konzistence) – stejná data jsou k dispozici v určitém čase na všech uzlech distribuovaného systému
- **Availability** (dostupnost) – každý klient, který odešle dotaz, dostane informaci o tom, zda byla operace úspěšná nebo ne
- **Partition tolerance** (tolerance k rozdělení) – pokud dojde k výpadku části sítě, musí být systém stále schopný odpovídat na dotazy



Obrázek 1 – CAP teorém (vlastní zdroj)

Garance a kombinace

MongoDB splňuje garance CP, zajišťuje konzistenci a toleranci k rozdělení na úkor dostupnosti.

Pro zajištění konzistence dat pracuje MongoDB na více serverech. Jeden z nich je primární (master) a ostatní jsou záložní (slave). Data jsou rozdělena do jednotlivých replikačních setů, které jsou uchovávány současně na několika serverech.

Pokud nastane výpadek primárního serveru, zvolí se nejaktuálnější ze záložních serverů, který se stane hlavním. Pomocí tohoto jsou data zálohovaná a při výpadku je systém rychle schopen opět reagovat. (9)

Vhodná a nevhodná řešení

Garance CP je dostačující pro transakční aplikace, protože prioritou je zachování konzistence dat. Pro tyto aplikace je důležité mít silnou záruku, že data budou vždy konzistentní. Vhodným příkladem může být zpracování objednávek a rezervací (např. Booking.com) či e-commerce (např. eBay).

Nevhodným řešením jsou naopak aplikace vyžadující vysokou dostupnost i za cenu dočasné ztráty konzistence dat. V takových případech je vhodnější zvolit řešení s garancí AP. Dalším nevhodným příkladem mohou být systémy, které tolerují dočasnou nekonzistenci a dávají přednost rychlosti a dostupnosti. (10)

Architektura

Architektura MongoDB je navržena tak, aby umožňovala správu velkých datových souborů. Je optimalizována pro paralelní práci s daty.

Replikace

V MongoDB se replikace skládá z primárních a sekundárních bloků, které představují objekty, do kterých se ukládají data. Primární blok přijímá všechny zápisy dat a zároveň replikuje data na sekundární bloky. Sekundární bloky sledují změny provedené na primárním bloku a udržují si stejná data jako primární blok. Každý sekundární blok smí mít nadřazený pouze jeden primární blok.

V případě selhání primárního bloku se automaticky zvolí jeden ze sekundárních bloků jako nový primární blok, což umožňuje obnovu dostupnosti dat a pokračování v provozu bez přerušení. (8)

Možnosti clusteru

MongoDB nabízí dva základní typy clusterů – replica set a sharded cluster.

Replica set (sada replik) je skupina serverů, které uchovávají kopie stejných dat – to zajišťuje vysokou odolnost a dostupnost. Například pokud by byla všechna data jen na jednom serveru, který by selhal, všechna data by se ztratila. Kromě odolnosti sada replik pomáhá i s přesměrováním uživatelů na ostatní servery, díky tomu snižuje celkové zatížení clusteru.

Cluster musí mít jeden primární node (uzel) a sadu sekundárních nodů, aby mohl být považován jako replica set. Cluster teoreticky může mít neomezený počet nodů, ale s rostoucím počtem nodů se zvyšuje složitost správy clusteru. Primární node přijímá všechny zápisové operace a sekundární nody replikují data z primárního nodu. Replikace probíhá asynchronně, což umožňuje sadě replik pokračovat i přes případné selhání. V případě selhání primárního nodu se určí nový primární node.

Sharded cluster distribuuje data do více shardů, díky tomu lépe zvládá velký objem dat. Každý shard může být samostatný replica set. Počet shardů lze nastavit podle potřeb a objemu dat. (11)

Sharding prakticky

MongoDB využívá sharding, aby rozložilo data do více úložišť. Disponuje výhodou v jednoduchosti vytvoření nového prostoru.

Klíčovými součástmi sharingu je shard, mongos (query router) a config server. Shard je samostatný server nebo seskupení serverů, které ukládají data. Mongos přijímá dotazy od klientů a posílá se

k odpovídajícím shardům. Obvykle je použito více query routerů, díky tomu je práce rozdělena a tím se zrychluje odezva. Config server uchovává metadata o clusteru, které obsahují mapování dat mezi jednotlivými shardy. (12)

Výhody shardingu

- Vyšší výkon – rozdělením dat na více serverů může sharding pomoci zlepšit výkonnost dotazů a snížit latenci
- Škálovatelnost – sharding umožňuje přidávat další servery (shardy) podle toho, jak data rostou
- Vysoká dostupnost – sharding může pomoci zajistit, že databáze zůstane dostupná i v případě výpadku jednoho z shardů

Problémy shardingu

- Složitost – sharding je komplikovanější na nastavení a správu než řešení s jedním serverem (shardem)
- Distribuce dat – zejména při práci s velkými datovými sadami může být obtížné zajistit rovnoměrné rozložení dat
- Návrh klíče pro sharding – špatně zvolený klíč shardu může vést k nerovnoměrnému rozložení dat a zvýšené latenci

Porovnání s jinými NoSQL a relačními databázemi

Značným rozdílem mezi nerelačními a relačními databázemi je jejich škálovatelnost. Oba typy databází je možné škálovat vertikálně – zvyšovat počet jader procesoru, kapacitu disku či přidávat RAM.

Relační databáze lze horizontálně škálovat velmi obtížně. Oproti nerelačním databázím je komplikované měnit jejich architekturu za provozu. Nerelační databáze lze horizontálně škálovat snadno pomocí přidání dalších samostatných serverů.

Nerelační databáze jsou více flexibilní i v možnostech přidání libovolného počtu a typu parametrů. Nejsou vázány na neměnné schéma řádků a atributů. Relační databáze jsou velmi konzistentní vzhledem ke svému pevnému schématu. Tyto databáze jsou vhodné pro případy, kdy záznamy mají stejný typ a počet hodnot. (13)

Dalším rozdílem je jazyk dotazování. Relační databáze využívají SQL (Structured Query Language) a MongoDB vyvinulo vlastní dotazovací jazyk MQL (MongoDB Query Language), který je přizpůsobený pro snadnou práci.

Relační databáze zaručují integritu vlastnostmi ACID (atomičnost, konzistence, izolovanost a trvanlivost). Nerelační databáze splňují CAP teorém (konzistence, dostupnost a tolerance k rozdělení), kterým nabízí kompromis mezi těmito vlastnostmi. (14)

V porovnání s jinými NoSQL databázemi má MongoDB flexibilní schéma, které umožňuje jednoduchou práci s datovými strukturami. Jiné NoSQL databáze mohou mít pevnější schéma, tím ale mohou být efektivnější pro konkrétní použití, například grafové databáze.

V rámci CAP teorému MongoDB splňuje garance CP. Jiné nerelační databáze například Cassandra a CouchDB splňují kombinaci AP – poskytují dostupnost a toleranci k rozdělení na úkor konzistence. Garance CA splňuje většina relačních databází a nerelační databáze Neo4j. (9)

Perzistence

Perzistence dat v MongoDB znamená ukládání dat a jejich obnovení v případě výpadku či havárii. MongoDB řeší perzistenci dat úložištěm (storage engine) WiredTiger. Od verze 3.2 je výchozím úložištěm. Je vhodný pro většinu úloh a doporučuje se pro nová nasazení. WiredTiger poskytuje mimo jiné souběžnost na úrovni dokumentu, kontrolní body (checkpointing) a kompresi.

Souběžnost na úrovni dokumentů (document-level concurrency) umožňuje více uživatelům upravovat různé dokumenty současně.

Kontrolní body (checkpoints) fungují jako body obnovy, WiredTiger od verze 3.6 tyto kontrolní body vytváří v intervalech 60 sekund.

WiredTiger podporuje několik možností komprese pro data i indexy. Komprese snižuje nároky na úložiště a může zlepšit výkon snížením množství dat, která je třeba číst nebo zapisovat na disk. (15)

Zabezpečení

Významnou součástí databáze je zabezpečení, velké množství dat je třeba chránit. MongoDB nabízí funkci Encryption at Rest, která byla zavedena od verze 3.6.8. Tato funkce umožňuje uživatelům šifrovat data uložená v databázi na úrovni souborů pomocí integrovaného algoritmu šifrování. Data jsou tedy šifrována, když jsou uložena na disku. To poskytuje další vrstvu ochrany dat.

Pro šifrování dat putujících mezi jednotlivými uzly lze využít síťové šifrování pomocí TLS (Transport Layer Security) nebo SSL (Secure Sockets Layer). Pro zabezpečenou komunikaci mezi klienty a serverem je možné aplikovat TLS a SSL je možné využít pro bezpečný přenos dat mezi klienty a serverem MongoDB. Pro zprovoznění šifrování je nutné na všech uzlech databázové sítě vytvořit certifikát podepsaný certifikační autoritou.

Obě funkce šifrování přenosové komunikace a Encryption at Rest jsou důležité pro zajištění celkové bezpečnosti dat v MongoDB. (16)

Cloud

MongoDB nabízí cloudovou službu MongoDB Atlas, která umožňuje vysoký výkon, je dobře škálovatelná a bezpečná. Značnou výhodou je, že není třeba se zabývat zprovozněním MongoDB serveru, vše řeší samotná služba. MongoDB Atlas využívá servery ve více než 70 oblastech napříč Amazon Web Services, Google Cloud Platform a Microsoft Azure.

Cloudové platformy poskytují širokou škálu služeb a nástrojů, které lze integrovat s MongoDB, jako jsou služby pro analýzu dat, strojové učení, IoT a další. To umožňuje vývojářům vytvářet komplexní aplikace a služby s využitím MongoDB jako datového úložiště a dalších cloudových služeb pro zpracování a analýzu dat. (17)

Výhody a nevýhody

MongoDB má spoustu výhod, ale i nevýhod. Podrobněji jsou popsány v této části práce.

Výhody

Nejvýznamnější výhodou MongoDB je flexibilita. Nevyžaduje schéma (je schema-less) a díky tomu MongoDB umožňuje ukládání dat s různými strukturami v jednom dokumentu. MongoDB ukládá data ve formátu dokumentů (BSON). To velmi usnadňuje práci s daty na rozdíl od klasických relačních databází.

MongoDB nabízí sharding, díky kterému je v databázi zajištěno rovnoměrné rozdělení dat a tím je zajištěno efektivní získávání dat i v případně velkého objemu dat. Další výhodou MongoDB je replikace, která zabezpečuje, že jsou data odolná a dostupná vůči výpadkům. (18)

MongoDB má velkou aktivní komunitu uživatelů a vývojářů s rozsáhlou dokumentací, návody a možnostmi podpory. Kromě toho nabízí i vlastní MongoDB univerzitu, kde se uživatelé mohou vzdělávat. Pro komerční účely nabízí enterprise verzi, kde poskytuje lepší zabezpečení i podporu. (19)

Nevýhody

Protože je MongoDB flexibilní databází a nabízí možnosti replikace a shardingu, může vyžadovat docházet k značnému nadbytku dat. A proto v porovnání s některými relačními databázemi MongoDB vyžaduje větší úložiště.

MongoDB je také nevýhodné v případě dat s pevně definovaným a neměnným schématem, v tomto případě je vhodnější použít relační databázi.

Relační databáze mohou být také výhodnější v možnostech dotazovacích prostředků, MongoDB je totiž méně efektivní v dotazech nad daty – například v rozsáhlých analytických funkcích. (18)

Nevýhodou MongoDB může být i znalostní bariéra, přechod na tento databázový systém může například vyžadovat školení (vzdělání) zaměstnanců, což většinou znamená i větší náklady.

Případy užití

Databáze MongoDB je vhodná systémy, pro které je třeba rychlý tok dat. Jsou jimi například sociální sítě, e-shopy či diskusní fóra. Díky shardingu je MongoDB efektivní ve zpracování velmi velkého množství dat, distribuuje data do více shardů.

MongoDB díky flexibilní architektuře bez schémat, může ukládat různé typy dat s různou strukturou. To umožňuje jednoduše ukládat data, aniž by bylo nutné předem definovat logiku. Schéma lze měnit podle toho, jak se aplikace vyvíjí. To je výhodné pro rychlý vývoj.

Tato databáze je vhodná i pro IoT (internet věcí). MongoDB je vhodné pro zpracování a analýzu dat v reálném čase. Zařízení (např. senzorová data či logy) mají architekturu řízenou událostmi v reálném čase, vyžadují rychlý příjem dat a nepřetržitý provoz. (20)

MongoDB je také vhodné pro cloudová úložiště, která je nezbytné efektivně prohledávat. MongoDB nabízí integraci s různými cloudovými platformami – například Google Cloud.

Vybraná databáze není vhodná pro systémy, které vyžadují integritu a bezpečnost dat. Pro tyto případy je vhodnější použít relační databáze. Jedná se například o aplikace ve finančních službách, kde všichni uživatelé potřebují vidět stejná data ve stejnou dobu.

MongoDB také není vhodné pro aplikaci, která vyžaduje datové schéma, které se často nemění. V těchto případech může být flexibilita MongoDB i nevýhodou, a proto vhodnější použít relační databáze. (18)

Případové studie

V této části seminární práce jsou uvedeny aplikace, které MongoDB využívají.

Airbnb

Airbnb používá MongoDB pro ukládání uživatelských profilů, rezervací, recenzí a dat souvisejících s pronájmem ubytování po celém světě. MongoDB umožňuje flexibilně ukládat různé typy dat – například obrázky, popisy a hodnocení ubytování. Také poskytuje vysokou dostupnost a efektivní zpracování velkého množství dat.

Twitter

Twitter používá MongoDB pro ukládání a zpracování velkého objemu dat, jako například jsou uživatelské profily, příspěvky a analýza chování uživatelů. MongoDB umožňuje Twitteru reagovat na požadavky uživatelů a poskytovat personalizovaný obsah či doporučení na základě jejich aktivity.

Shutterstock

Shutterstock je platforma, která poskytuje možnost zakoupit a použít fotografie, videa a hudbu pro vlastní potřeby. MongoDB využívá pro správu rozsáhlé knihovny digitálních souborů. Také umožňuje vyhledávání, filtrování a nákup těchto souborů.

Adobe Cloud

Společnost Adobe Inc. využívá MongoDB pro cloudové služby, které uživatelům poskytují možnosti pro správu obsahu a digitální marketing. MongoDB umožňuje efektivně zpracovávat velké množství dat v reálném čase. Zajišťuje také dostupnost a odolnost dat po celém světě.

AstraZeneca

AstraZeneca je farmaceutická společnost, využívá MongoDB pro výzkum a vývoj léčiv. Poskytuje společnosti efektivní práci s komplexními daty v reálném čase například vyhledávání léků či sledování výsledků testování léků. MongoDB této farmaceutické společnosti pomáhá zkrátit čas a náklady spojené s uvedením nových léků na trh.

Praktické použití

Instalace/Konkrétní použití

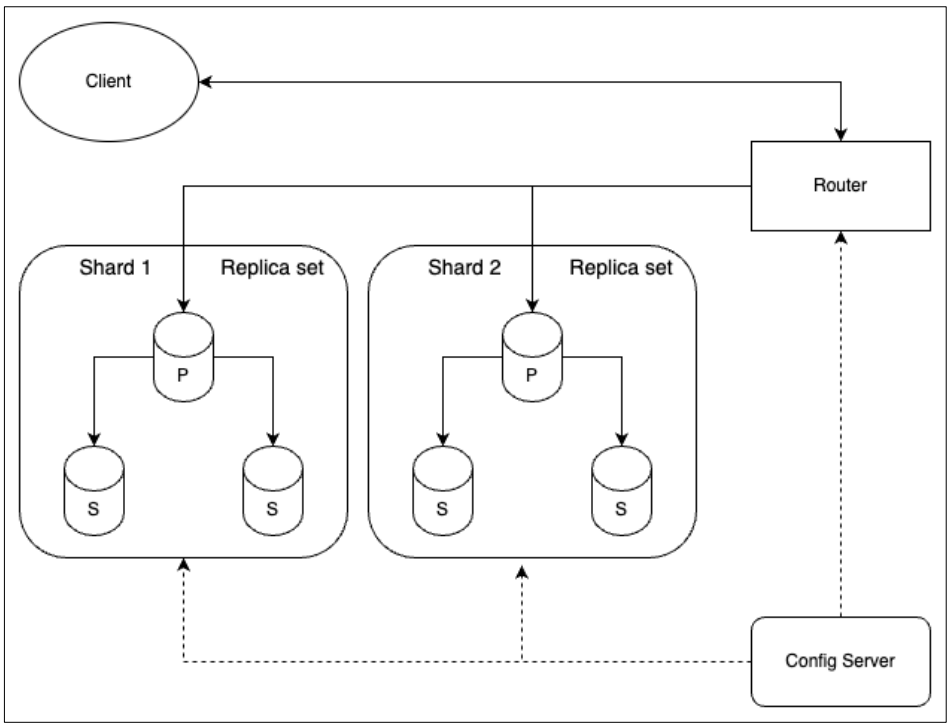
Popis docker-compose.yml

Níže je popsaný soubor docker-compose.yml, který nastavuje router, config server a shardy. Protože je příkazů mnoho a často opakují, je soubor popsán stručně.

- **version:** (určuje verzi Docker Compose)
- **services:** (definuje jednotlivé služby, které budou spuštěny)
    - **router01:** (router (mongos), který směruje dotazy na správné shardy)
        - image: (verze Docker image)
        - container_name: (jméno kontejneru)
        - command: (příkaz pro spuštění routeru)
        - ports: (mapuje port)
        - volumes: (definuje úložiště)
    - **configsvr01:** (konfigurační server)
        - image: (verze Docker image)
        - container_name: (jméno kontejneru)
        - command: (příkaz pro spuštění konfiguračního serveru)
        - volumes: (definuje úložiště pro data)
        - ports: (mapuje porty)
        - links: (propojuje kontejner s shardy)
    - **shard01-a, shard01-b, shard01-c:** (první shard, tři uzly v replika setu)
        - image: (verze Docker image)
        - container_name: (jméno kontejneru)
        - command: (příkaz pro spuštění shardu s replika setem)
        - volumes: (definuje úložiště pro data)
        - ports: (mapuje porty)
        - links: (propojuje tento kontejner s dalšími shardy)
    - **shard02-a, shard02-b, shard02-c:** Druhý shard, tři uzly v replika setu.
        - image: (verze Docker image)
        - container_name: (jméno kontejneru)
        - command: (příkaz pro spuštění shardu s replika setem)
        - volumes: (definuje úložiště pro data)
        - ports: (mapuje porty)
        - links: (propojuje tento kontejner s dalšími shardy)
- **volumes:** (definuje úložiště pro ukládání dat a konfigurací, aby data byla perzistentní)

Vlastní řešení

Schéma architektury



Obrázek 2 – Schéma architektury

Cluster

Cluster se skládá z jednoho konfiguračního serveru (config server), dvou shardů (každý se třemi uzly) a jednoho routeru.

Cluster má celkem 7 nodů – tři jsou v prvním shardu, tři v druhém shardu a config server.

Distribuce dat

Data jsou distribuována pomocí shardingu, konkrétně kombinací config serveru, routeru a shardů.

Sharding

Sharding je v projektu řešen pomocí config serveru a shardů, které jsou nastaveny jako replica sety a routeru, který směruje dotazy na správné shardy.

Replikace

V projektu se replikace nachází mezi dvěma shardy. Je také uvnitř každého shardu v podobě replica setu.

Perzistence dat

Data jsou perzistentní díky použití Docker volumes. Každý kontejner má vlastní volume, který zajišťuje uchování dat i při restartu kontejnerů. Jsou definovány v docker-compose.yml.

Zabezpečení

Zabezpečení je v práci řešeno pomocí autentizace (ověření) uživatele, který se může pod jménem a heslem přihlásit.

Případ užití

Toto řešení slouží pro případy, kde je potřeba vysoká dostupnost a škálovatelnost dat. To je vhodné například pro e-commerce platformy, kde je třeba spravovat velké množství uživatelských dat a produktových informací. Dalším vhodným příkladem použití jsou analytické nástroje, kde se analyzují a zpracovávají velké objemy dat z různých zdrojů.

CAP teorém

Consistency

V projektu je konzistence zajištěna v rámci replica setu, kde primární uzel zajišťuje, že všechny zápisy jsou replikovány na sekundární uzly.

Partition Tolerance

V projektu je partition tolerance řešeno pomocí shardingu, data jsou distribuována mezi další nody, což umožňuje pokračovat v práci i při výpadku některých částí systému.

Data

Tento dataset obsahuje informace o uživatelích Amazon Prime – zahrnuje identifikační údaje jako jméno (Name), e-mail (Email Adress), uživatelské jméno (Username), datum narození (Date of Birth), pohlaví (Gender) a lokalitu (Location). Dále obsahuje informace o předplatném (Subscription plan, Membership Start Date, Membership End Date), způsobu platby (Payment Information), obnově předplatného (Renewal Status), frekvenci používání služeb (Usage Frequency), historii nákupů (Purchase History), oblíbených žánrech (Favourite Genres), používaných zařízeních (Device Used), metrikách interakce (Engagement Metrics), hodnoceních (Feedback/Ratings) a počtu interakcí se zákaznickou podporou (Customer Support Interactions).

URL adresa dat

Rozsah hodnot

Dataset má 2500 řádků a 19 sloupců. Podrobnější rozsah hodnot v datasetu je proveden pomocí Pythonu níže.

```python
# Rozsah hodnot
min_value = dataset.min()
max_value = dataset.max()
print("\nMinimální hodnota:")
print(min_value)
print("\nMaximální hodnota:")
print(max_value)
```

Ukázka kódu 1 – Rozsah hodnot v datasetu

```
Minimální hodnota:
User ID                                1
Name                          Aaron Goodman
Email Address          aanderson@example.org
Username                           aanderson
Date of Birth                     1933-04-26
Gender                                Female
Location                           Aaronfort
Membership Start Date             2024-01-01
Membership End Date               2024-12-31
Subscription Plan                     Annual
Payment Information                     Amex
Renewal Status                   Auto-renew
Usage Frequency                    Frequent
Purchase History                      Books
Favorite Genres                      Action
Devices Used                       Smart TV
Engagement Metrics                     High
Feedback/Ratings                        3.0
Customer Support Interactions             0
```

Obrázek 3 – Minimální hodnoty v datasetu (výstup z kódu)

```
Maximální hodnota:
User ID                             2500
Name                     Zachary Vasquez
Email Address       zwilson@example.net
Username                         zwilson
Date of Birth                 2006-04-11
Gender                              Male
Location                   Zimmermanshire
Membership Start Date         2024-04-14
Membership End Date           2025-04-14
Subscription Plan                Monthly
Payment Information                 Visa
Renewal Status                    Manual
Usage Frequency                  Regular
Purchase History             Electronics
Favorite Genres                    Sci-Fi
Devices Used                       Tablet
Engagement Metrics                 Medium
Feedback/Ratings                      5.0
Customer Support Interactions          10
```

Obrázek 4 – Maximální hodnoty v datasetu (výstup z kódu)

Struktura dat

Kontrola typů dat byla provedena pomocí Pythonu, ukázka kódu je uvedena níže.

```python
# Kontrola typů dat
data_types = dataset.dtypes
print("\nTypy dat:")
print(data_types)
```

Ukázka kódu 2 – Datové struktury v datasetu

```
Typy dat:
User ID                              int64
Name                                object
Email Address                       object
Username                            object
Date of Birth                       object
Gender                              object
Location                            object
Membership Start Date               object
Membership End Date                 object
Subscription Plan                   object
Payment Information                 object
Renewal Status                      object
Usage Frequency                     object
Purchase History                    object
Favorite Genres                     object
Devices Used                        object
Engagement Metrics                  object
Feedback/Ratings                   float64
Customer Support Interactions        int64
```

Obrázek 5 – Typy dat v datasetu (výstup z kódu)

Prázdné hodnoty (missing values)

Dataset neobsahuje žádné chybějící hodnoty, kontrola byla provedena v Pythonu, ukázka kódu se nachází níže.

```python
missing_values = dataset.isnull().sum()
print("Chybějící hodnoty:")
print(missing_values)
```

Ukázka kódu 3 – Chybějící hodnoty v datasetu

```
Chybějící hodnoty:
User ID                            0
Name                               0
Email Address                      0
Username                           0
Date of Birth                      0
Gender                             0
Location                           0
Membership Start Date              0
Membership End Date                0
Subscription Plan                  0
Payment Information                0
Renewal Status                     0
Usage Frequency                    0
Purchase History                   0
Favorite Genres                    0
Devices Used                       0
Engagement Metrics                 0
Feedback/Ratings                   0
Customer Support Interactions      0
```

Obrázek 6 – Chybějící hodnoty (výstup z kódu)

Duplicitní hodnoty

Kontrola a odstranění duplicitní hodnot byla provedena v Pythonu, ukázka kódu je níže. Počet řádků po odstranění duplicit zůstal stejný, v datasetu se tedy nenachází duplicity.

```python
# Odstranění duplicitních záznamů
dataset = dataset.drop_duplicates()
print("Počet řádků po odstranění duplicit: ", len(dataset))
```

Ukázka kódu 4 – Duplicitní hodnoty v datasetu

```
Počet řádků po odstranění duplicit:  2500
```

Obrázek 7– Počet řádků po odstranění duplicit (výstup z kódu)

Kontrola validity dat

Pomocí Pythonu byla provedena kontrola validity některých dat – konkrétně proměnné 'Membership Start Date', 'Feedback/Ratings' a 'Gender'. Ukázka kódu je uvedena níže.

```python
# Kontrola validity dat pro proměnnou 'Membership Start Date'
print("Minimální datum:", dataset['Membership Start Date'].min())
print("Maximální datum:", dataset['Membership Start Date'].max())

# Kontrola negativních hodnot proměnné 'Feedback/Ratings'
negative_feedback = dataset[dataset['Feedback/Ratings'] < 0]
print("Počet záporných hodnot proměnné 'Feedback/Ratings':", len(negative_feedback))

# Kontrola validity dat pro proměnnou 'Gender'
unique_genders = dataset['Gender'].unique()
print("Hodnoty proměnné 'Gender':", unique_genders)
```

Ukázka kódu 5 – Kontrola validity dat v datasetu

```
Minimální datum: 2024-01-01
Maximální datum: 2024-04-14
Počet záporných hodnot proměnné 'Feedback/Ratings': 0
Hodnoty proměnné 'Gender': ['Male' 'Female']
```

Obrázek 8 – Kontrola validity dat (výstup z kódu)

Příklady k procvičování

Příklad 1 – Nastavení DB

```
// Spuštění databáze
docker-compose up -d

// Připojení ke kontejneru „configsvr01“ a spuštění mongosh
docker-compose exec configsvr01 mongosh

// Definice konfigurace pro replica set
var config = {
  _id: "rs-config-server",
  configsvr: true,
  version: 1,
  members: [
    {
      _id: 0,
      host: "configsvr01:27017",
      priority: 1,
    },
  ],
};
rs.initiate(config, { force: true });
```

Příklad 2 – Nastavení shardu

```
// Definice konfigurace pro shard
var config = {
  _id: "rs-shard-01",
  version: 1,
  members: [
    {
      _id: 0,
      host: "shard01-a:27017",
      priority: 1,
    },
    {
      _id: 1,
      host: "shard01-b:27017",
      priority: 0.5,
    },
    {
      _id: 2,
      host: "shard01-c:27017",
      priority: 0.5,
    },
  ],
};

// Nastavení konfigurace pro shard
rs.initiate(config, { force: true });
```

Příklad 3 – Autorizace

```
// Vstup do Mongo Shell
mongosh

// Použití admin databáze
use admin;

// Vytvoření uživatele
db.createUser({user: "admin",
      pwd: "password",
      roles:[{role: "root", db: "admin"}]});

// Zapnutí shardingu pro databázi
sh.enableSharding("database");

db.adminCommand({
  shardCollection: "database.collection",
  key: { oemNumber: "hashed", zipCode: 1, supplierId: 1 },
});
Exit;
```

Příklad 4 – Příprava a načtení dat

```
// Převedení dat z .csv na json
python3 convert-data-to-json.py

// Kopírování do docker kontejneru
docker cp amazon_prime_users.csv router-01:/amazon_prime_users.json

// Otevření příkazové řádky routeru01
docker exec -it router-01 bash

// Import dat do kolekce users
mongoimport --db data --collection users --file /amazon_prime_users.json -
-jsonArray

mongosh

use data;

// Kontrola dat
db.users.find();
```

Příklad 5 – Dotazy na data

```
// Hledání všech uživatelů
db.users.find({});

// Hledání všech uživatelů, kteří mají oblíbený žánr "Sci-Fi"
db.users.find({ "Favorite_Genres": "Sci-Fi" });

// Hledání všech uživatelů, kteří mají skóre hodnocení vyšší než 4.5
db.users.find({ "Feedback/Ratings": { $gt: 4.5 } });

// Hledání uživatele podle User_ID
db.users.find({ "User_ID": 10 });

// Hledání všech uživatelů, kteří používají zařízení "Smart TV"
db.users.find({ "Devices_Used": "Smart TV" });
```

Příklad 6 – Vkládání nových dat

```
// Vložení nového uživatele pomocí příkazu insertOne
db.users.insertOne({
  "User_ID": 118,
  "Name": "John Doe",
  "Email_Address": "johndoe@example.com",
  "Username": "johndoe",
  "Date_of_Birth": "1990-01-01",
  "Gender": "Male",
  "Location": "New York",
  "Membership_Start_Date": "2024-05-01",
  "Membership_End_Date": "2025-05-01",
  "Subscription_Plan": "Annual",
  "Payment_Information": "Visa",
  "Renewal_Status": "Manual",
  "Usage_Frequency": "Regular",
  "Purchase_History": "Books",
  "Favorite_Genres": "Fiction",
  "Devices_Used": "Smart TV",
  "Engagement_Metrics": "Medium",
  "Feedback/Ratings": 4.7,
  "Customer_Support_Interactions": 2
});
```

```
// Vložení více uživatelů najednou pomocí příkazu insertMany
db.users.insertMany([
  {
    "User_ID": 119,
    "Name": "Jane Smith",
    "Email_Address": "janesmith@example.com",
    "Username": "janesmith",
    "Date_of_Birth": "1985-05-15",
    "Gender": "Female",
    "Location": "Los Angeles",
    "Membership_Start_Date": "2024-05-01",
    "Membership_End_Date": "2025-05-01",
    "Subscription_Plan": "Monthly",
    "Payment_Information": "Mastercard",
    "Renewal_Status": "Auto-renew",
    "Usage_Frequency": "Frequent",
    "Purchase_History": "Electronics",
    "Favorite_Genres": "Drama",
    "Devices_Used": "Smartphone",
    "Engagement_Metrics": "High",
    "Feedback/Ratings": 4.8,
    "Customer_Support_Interactions": 1
  },
  {
    "User_ID": 120,
    "Name": "Alice Johnson",
    "Email_Address": "alicejohnson@example.com",
    "Username": "alicejohnson",
    "Date_of_Birth": "1975-03-10",
    "Gender": "Female",
    "Location": "Chicago",
    "Membership_Start_Date": "2024-05-01",
    "Membership_End_Date": "2025-05-01",
    "Subscription_Plan": "Annual",
    "Payment_Information": "Amex",
    "Renewal_Status": "Manual",
    "Usage_Frequency": "Occasional",
    "Purchase_History": "Books",
    "Favorite_Genres": "Horror",
    "Devices_Used": "Tablet",
    "Engagement_Metrics": "Low",
    "Feedback/Ratings": 4.2,
    "Customer_Support_Interactions": 3
  }
]);
```

Příklad 7 – Aktualizace existujících dat

```
// Aktualizace e-mailu uživatele s User_ID 1
db.users.updateOne(
  { "User_ID": 1 },
  { $set: { "Email_Address": "newemail@example.com" } }
);

// Přidání pole do dat uživatele s User_ID 2
db.users.updateOne(
  { "User_ID": 2 },
  { $set: { "New_Field": "New Value" } }
);

// Zvýšení hodnoty „Feedback/Ratings“ uživatele s User_ID 3 o 0.5
db.users.updateOne(
  { "User_ID": 3 },
  { $inc: { "Feedback/Ratings": 0.5 } }
);

// Nastavení „Renewal_Status“ na „Manual“ pro všechny uživatele s
„Subscription_Plan“ = „Annual“
db.users.updateMany(
  { "Subscription_Plan": "Annual" },
  { $set: { "Renewal_Status": "Manual" } }
);

// Aktualizace lokace pro všechny uživatele v konkrétním městě
db.users.updateMany(
  { "Location": "Chicago" },
  { $set: { "Location": "New Chicago" } }
);
```

Příklad 8 – Mazání dat

```
// Smazání uživatele s User_ID 4
db.users.deleteOne({ "User_ID": 4 });

// Smazání všech uživatelů s „Feesback/Ratings“ menším než 3.0
db.users.deleteMany({ "Feedback/Ratings": { $lt: 3.0 } });

// Smazání uživatele podle e-mailu
db.users.deleteOne({ "Email_Address": "janedoe@example.com" });

// Smazání všech uživatelů s „Favourite_Genres“ = „Horror“
db.users.deleteMany({ "Favorite_Genres": "Horror" });

// Smazání všech uživatelů, kteří používají „Smartphone“
db.users.deleteMany({ "Devices_Used": { $exists: "Smartphone" } });
```

Příklad 9 – Agregační funkce

```
// Seskupit uživatele podle oblíbeného žánru a spočítat počet uživatelů
db.users.aggregate([
  { $group: { _id: "$Favorite_Genres", count: { $sum: 1 } } }
]);

// Seskupit uživatele podle zařízení a spočítat průměrnou hodnotu
„Feedback/Ratings“
db.users.aggregate([
  { $group: { _id: "$Devices_Used", avgRating: { $avg: "$Feedback/Ratings" } } }
]);

// Seskupit uživatele podle lokace a spočítat celkový počet interakcí s podporou
db.users.aggregate([
  { $group: { _id: "$Location", totalInteractions: { $sum:
"$Customer_Support_Interactions" } } }
]);

// Seskupit uživatele podle předplatného plánu a spočítat průměrnou frekvenci
používání
db.users.aggregate([
  { $group: { _id: "$Subscription_Plan", avgFrequency: { $avg:
"$Usage_Frequency" } } }
]);

// Seskupit uživatele podle pohlaví a spočítat celkový počet uživatelů pro každé
pohlaví
db.users.aggregate([
  { $group: { _id: "$Gender", count: { $sum: 1 } } }]);
```

Příklad 10 – Operace s kolekcemi

```
// Vytvoření nové kolekce

db.createCollection("archived_users");


// Zobrazení všech kolekcí

db.getCollectionNames();


// Zkopírování dat z kolekce users do archived_users

db.users.aggregate([{ $match: {} }, { $out: "archived_users" }]);


// Smazání všech dat v kolekci archived_users

db.archived_users.deleteMany({});


// Zrušení kolekce archived_users

db.archived_users.drop();
```

Závěr

MongoDB díky dokumentově orientovanému modelu a podpoře škálování pomocí shardingu je vhodným řešením pro aplikace, které vyžadují flexibilitu a vysoký výkon. Přestože má své nevýhody, jako jsou například vyšší požadavky na úložiště, má řadu výhod, díky kterým se MongoDB často stává preferovanou platformou v mnoha organizacích.

Tato seminární práce poskytla komplexní pohled na databázi MongoDB, její vlastnosti a praktické využití. Praktická část ukázala popis souboru docker-compose.yml, nastavení databáze a konkrétní příkazy pro manipulaci s daty.

Zdroje

1. **IBM.** What is MongoDB? *IBM.* [Online] [Citace: 14. duben 2024.] https://www.ibm.com/topics/mongodb.

2. **MongoDB, Inc.** Relational vs. Non-Relational Databases. *MongoDB, Inc.* [Online] [Citace: 5. květen 2024.] https://www.mongodb.com/resources/compare/relational-vs-non-relational-databases.

3. **Thombare, Dayanand.** Data Distribution: Sharding and Partitioning in MongoDB. *Medium.* [Online] 28. leden 2024. [Citace: 25. duben 2024.] https://medium.com/@dayanandthombare/data-distribution-sharding-and-partitioning-in-mongodb-3e4cd8edd955.

4. **MongoDB, Inc.** Databases and Collections. *MongoDB, Inc.* [Online] [Citace: 26. duben 2024.] https://www.mongodb.com/docs/manual/core/databases-and-collections/.

5. **GeeksforGeeks.** MongoDB Views. *GeeksforGeeks.* [Online] 22. leden 2024. [Citace: 26. duben 2024.] https://www.geeksforgeeks.org/mongodb-views/.

6. —. DataTypes in MongoDB. *GeeksforGeeks.* [Online] 9. červen 2022. [Citace: 26. duben 2024.] https://www.geeksforgeeks.org/datatypes-in-mongodb/.

7. **MongoDB, Inc.** Sharding. *MongoDB.* [Online] MongoDB, Inc., 2023. [Citace: 10. duben 2024.] https://www.mongodb.com/docs/manual/sharding/.

8. —. Replication. *MongoDB.* [Online] [Citace: 20. duben 2024.] https://www.mongodb.com/docs/manual/replication/.

9. **IBM.** What is the CAP theorem? *IBM.* [Online] [Citace: 20. duben 2024.] https://www.ibm.com/topics/cap-theorem.

10. **MongoDB, Inc.** MongoDB Use Cases. *MongoDB.* [Online] [Citace: 20. duben 2024.] https://www.mongodb.com/solutions/use-cases.

11. —. MongoDB Clusters . *MongoDB, Inc.* [Online] [Citace: 5. květen 2024.] https://www.mongodb.com/resources/products/fundamentals/clusters.

12. —. Sharding. *MongoDB.* [Online] [Citace: 25. duben 2024.] https://www.mongodb.com/docs/manual/sharding/.

13. **Můčka, Jan.** Relační databáze vs. nerelační databáze: jaké jsou mezi nimi rozdíly? *MasterDC.* [Online] 27. květen 2021. [Citace: 26. duben 2024.] https://www.master.cz/blog/relacni-databaze-nerelacni-databaze-jake-jsou-rozdily/.

14. **Microsoft.** Relační vs. data NoSQL. *Microsoft Learn.* [Online] 5. březen 2024. [Citace: 27. duben 2024.] https://learn.microsoft.com/cs-cz/dotnet/architecture/cloud-native/relational-vs-nosql-data.

15. **MongoDB, Inc.** WiredTiger Storage Engine. *MongoDB, Inc.* [Online] [Citace: 5. květen 2024.] https://www.mongodb.com/docs/manual/core/wiredtiger/.

16. —. Security. *MongoDB, Inc.* [Online] [Citace: 27. duben 2024.] https://www.mongodb.com/docs/manual/security/.

17. —. What is MongoDB Atlas? *MongoDB, Inc.* [Online] [Citace: 27. duben 2024.] https://www.mongodb.com/docs/atlas/.

18. **GeeksforGeeks.** MongoDB Advantages & Disadvantages. *GeeksforGeeks.* [Online] 1. prosinec 2023. [Citace: 30. duben 2024.] https://www.geeksforgeeks.org/mongodb-advantages-disadvantages/.

19. **MongoDB, Inc.** Enterprise Advanced. The best way to run MongoDB yourself. *MongoDB, Inc.* [Online] [Citace: 30. duben 2024.] https://www.mongodb.com/products/self-managed/enterprise-advanced.

20. —. MongoDB for Internet of Things (IoT). *MongoDB.* [Online] [Citace: 10. duben 2024.] https://www.mongodb.com/solutions/use-cases/internet-of-things.

21. **Kumari, Bharti.** Introduction to MongoDB. *Scaler.* [Online] 11. únor 2024. [Citace: 14. duben 2024.] https://www.scaler.com/topics/mongodb/what-is-mongodb/.

22. **GeeksforGeeks.** DataTypes in MongoDB. *GeeksforGeeks.* [Online] 9. červen 2022. [Citace: 25. duben 2024.] https://www.geeksforgeeks.org/datatypes-in-mongodb/.

Přílohy

Docker-compose.yml

```
version: "3"
services:
  router01:
    image: mongo:6.0.1
    container_name: router-01
    command: mongos --port 27017 --configdb rs-config-server/configsvr01:27017,configsvr02:27017,configsvr03:27017 --bind_ip_all
    ports:
      - 27117:27017
    volumes:
      - ./config:/config
      - mongodb_cluster_router01_db:/data/db
      - mongodb_cluster_router01_config:/data/configdb
  configsvr01:
    image: mongo:6.0.1
    container_name: mongo-config-01
    command: mongod --port 27017 --configsvr --replSet rs-config-server
    volumes:
      - ./config:/config
      - mongodb_cluster_configsvr01_db:/data/db
      - mongodb_cluster_configsvr01_config:/data/configdb
    ports:
      - 27119:27017
    links:
      - shard01-a
      - shard02-a
  shard01-a:
    image: mongo:6.0.1
    container_name: shard-01-node-a
    command: mongod --port 27017 --shardsvr --replSet rs-shard-01
    volumes:
      - ./config:/config
      - mongodb_cluster_shard01_a_db:/data/db
      - mongodb_cluster_shard01_a_config:/data/configdb
    ports:
      - 27122:27017
    links:
      - shard01-b
      - shard01-c
  shard01-b:
    image: mongo:6.0.1
    container_name: shard-01-node-b
    command: mongod --port 27017 --shardsvr --replSet rs-shard-01
    volumes:
      - ./config:/config
      - mongodb_cluster_shard01_b_db:/data/db
      - mongodb_cluster_shard01_b_config:/data/configdb
    ports:
      - 27123:27017
  shard01-c:
    image: mongo:6.0.1
    container_name: shard-01-node-c
    command: mongod --port 27017 --shardsvr --replSet rs-shard-01
    volumes:
      - ./config:/config
      - mongodb_cluster_shard01_c_db:/data/db
      - mongodb_cluster_shard01_c_config:/data/configdb
    ports:
      - 27124:27017
  shard02-a:
    image: mongo:6.0.1
    container_name: shard-02-node-a
    command: mongod --port 27017 --shardsvr --replSet rs-shard-02
    volumes:
      - ./config:/config
      - mongodb_cluster_shard02_a_db:/data/db
      - mongodb_cluster_shard02_a_config:/data/configdb
    ports:
      - 27125:27017
    links:
      - shard02-b
      - shard02-c
  shard02-b:
    image: mongo:6.0.1
    container_name: shard-02-node-b
    command: mongod --port 27017 --shardsvr --replSet rs-shard-02
    volumes:
      - ./config:/config
      - mongodb_cluster_shard02_b_db:/data/db
      - mongodb_cluster_shard02_b_config:/data/configdb
    ports:
      - 27126:27017
  shard02-c:
    image: mongo:6.0.1
    container_name: shard-02-node-c
    command: mongod --port 27017 --shardsvr --replSet rs-shard-02
    volumes:
      - ./config:/config
      - mongodb_cluster_shard02_c_db:/data/db
      - mongodb_cluster_shard02_c_config:/data/configdb
    ports:
      - 27127:27017

volumes:
  mongodb_cluster_router01_db:
  mongodb_cluster_router01_config:
  mongodb_cluster_configsvr01_db:
  mongodb_cluster_configsvr01_config:
  mongodb_cluster_shard01_a_db:
  mongodb_cluster_shard01_a_config:
  mongodb_cluster_shard01_b_db:
  mongodb_cluster_shard01_b_config:
  mongodb_cluster_shard01_c_db:
  mongodb_cluster_shard01_c_config:
  mongodb_cluster_shard02_a_db:
  mongodb_cluster_shard02_a_config:
  mongodb_cluster_shard02_b_db:
  mongodb_cluster_shard02_b_config:
  mongodb_cluster_shard02_c_db:
  mongodb_cluster_shard02_c_config:
```