33. Dokumentové databáze- koncept, srovnání s key-value úložišti, pojem dokument, výhody a nevýhody. MongoDB- charakteristika a architektura.

Dokumentové databáze- Koncept, srovnání s key-value úložišti a pojem dokument

- označovány také jako
  - dokumentově orientované databáze
  - úložiště dokumentů
- v současnosti asi nejpoužívanější typ NoSQL databází
- v principu podobné key-value úložištím
  - zachován princip key-value (klíč-hodnota)
    - key jednoznačným identifikátorem value
  - ale value obsahují strukturovaná nebo částečně strukturovaná data
    - tzv. dokumenty
  - samotná data mohou být indexována a dotazována
    - indexy nad atributy dat
    - dotazy na strukturu dat i na prvky v této struktuře
    - je možné získat jen požadované části dokumentů
- základní stavební prvky dokumentových databází
  - zapouzdřují a kódují data v definovaném formátu (kódování)
    - implementace se liší databázi od databáze
  - používaná kódování
    - textová forma
      - XML, YAML, JSON
    - binární forma
      - BSON, PDF, MS Office dokumenty
  - identifikovány jednoznačným identifikátorem (klíčem)
    - typicky řetězec, URI nebo cesta
    - slouží pro přístup k dokumentům
      - ale i pro vkládání
    - často indexovány
      - rychlejší přístup k dokumentům

- základní stavební prvky dokumentových databází
  - koncepčně odpovídají objektům v OOP
  - volné schéma
    - dokumenty mohou být velmi komplexní
    - mohou obsahovat vnořené (embedded) dokumenty
    - nemusí obsahovat stejné oddíly, atributy, části nebo klíče
    - podobně jako objekty
    - vysoká míra flexibility

https://expert.data.blog/category/bigdata/

  - obecně data patřící k sobě ukládána do jednoho dokumentu
    - na rozdíl od relačních databází
    - usnadňuje přístup a práci s daty
  - k dokumentům často přidružena a uložena metadata

Ukázka v XML

```xml
<artist>
    <artistname>Iron Maiden</artistname>
    <albums>
     <album>
        <albumname>The Book of Souls</albumname>
        <datereleased>2015</datereleased>
        <genre>Hard Rock</genre>
     </album>
     <album>
        <albumname>Killers</albumname>
        <datereleased>1981</datereleased>
        <genre>Hard Rock</genre>
     </album>
     <album>
        <albumname>Powerslave</albumname>
        <datereleased>1984</datereleased>
        <genre>Hard Rock</genre>
     </album>
     <album>
        <albumname>Somewhere in Time</albumname>
        <datereleased>1986</datereleased>
        <genre>Hard Rock</genre>
     </album>
    </albums>
</artist>
```

Ukázka v JSON:

```json
{
    "_id" : 1,
    "artistName" : "Iron Maiden",
    "albums" : [
        {
            "albumname" : "The Book of Souls",
            "datereleased" : 2015,
            "genre" : "Hard Rock"
        }, {
            "albumname" : "Killers",
            "datereleased" : 1981,
            "genre" : "Hard Rock"
        }, {
            "albumname" : "Powerslave",
            "datereleased" : 1984,
            "genre" : "Hard Rock"
        }, {
            "albumname" : "Somewhere in Time",
            "datereleased" : 1986,
            "genre" : "Hard Rock"
        }
    ]
}
```

- ukázka volného schématu ve formátu JSON


```
_id: ObjectId("5f8ef175c43ece2db0230f85")
title: "Post One"
body: "Body of post one"
category: "News"
likes: 4
tags: Array
  0: "news"
  1: "events"
user: Object
  name: "John Doe"
  status: "author"
date: "Date()"
```

```
_id: ObjectId("5f8ef175c43ece2db0230f86")
title: "Post Two"
body: "Body of post two"
tags: "news"
date: "Date()"
```

```
_id: ObjectId("5f8ef175c43ece2db0230f87")
source: "id1"
title: "Post Three"
views: "80"
```

```
_id: ObjectId("5f8ef175c43ece2db0230f88")
title: "Post Four"
category: "Entertainment"
```

```
_id: ObjectId("5f8fe759c43ece2db0230f8a")
title: "Post Five"
likes: 81
user: Object
  name: "John Doe"
  status: "author"
date: "Date()"
```

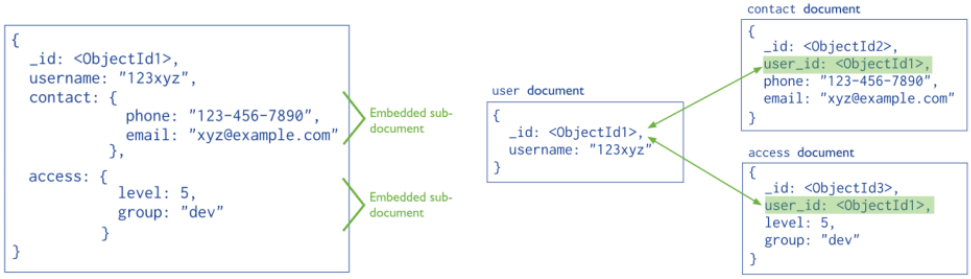
```
_id: ObjectId("5f9026800cbc092824d7e420")
source: "id2"
title: "Unknown"
user: Object
  name: "Jane Doe"
  gender: "female"
```


Charakteristiky

- podporují standardní operace s daty (dokumenty)
  - operace CRUD
  - implementace se liší databázi od databáze
  - vytvoření (vložení) [creation]
  - čtení (dotazování, vyhledávání) [retrieval]
    - kromě vyhledávání podle klíče také podpora dotazovacího jazyka
    - vyhledávání v závislosti na obsahu (nebo metadatech)
  - aktualizace [update]
    - i jen části dokumentu
  - smazání [deletion]
  - mohou podporovat transakce
    - ACID
    - není ale pravidlem

https://medium.com/@hau12a1/golang-http-crud-l-the-create-part-ae42c962c557

  - obecně se vyhýbají vazbám mezi dokumenty
    - případně dvě varianty řešení
      - embedded dokumenty
      - reference

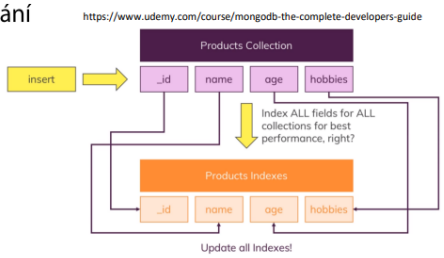


INDEXOVÁNÍ

- vylepšuje rychlost vyhledávání
  - bez indexování se provádí sken celé kolekce
  - vhodný index výrazně omezuje dokumenty, které je potřeba skenovat
- index
  - speciální datová struktura definovaná na úrovni kolekcí
  - ukládá hodnotu specifického pole v seřazené formě
    - snadné procházení a porovnávání
    - obsahuje také pointer na celý dokument pro snadný přístup
  - možnost definovat nad libovolným polem ale i nad jejich kombinací
    - v základu index _id
  - využití i pro rychlé řazení
  - základ pro sharding
    - jen jeden může být použit

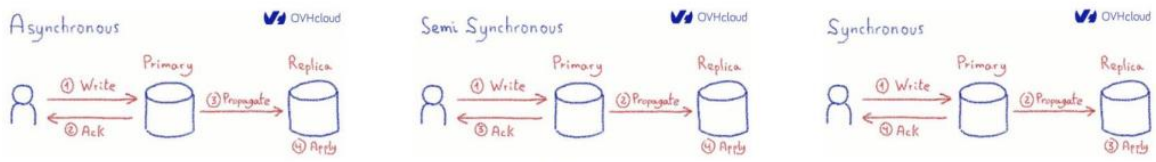
INDEXOVÁNÍ

- vylepšuje rychlost vyhledávání
- proč tedy nedefinovat indexy nad všemi poli?
  - teoreticky vylepší veškeré vyhledávání
  - v praxi ale index není zadarmo
  - cenou je vkládání / aktualizace
    - při každém vložení je potřeba vložit prvek i do seřazeného indexu
    - potřeba vložit do všech indexů
    - pomalé?
  - indexy je potřeba řádně promyslet



- indexování je pomalejší, když dotaz vrací velkou část kolekce
  - způsobeno přechodem index – pointer – dokument
  - naopak při kompletním skenu už jsou dokumenty načteny v paměti

- u dokumentových databází nejčastěji typu master-slave
  - obvyklá vazba mezi originálem a kopiemi
  - master zaznamenává změny, které předává slaves
  - slaves potvrdí přijetí změn, čímž umožní další aktualizace
  - asynchronní (eventuálně k.), semi synchronní, synchronní (striktně k.)

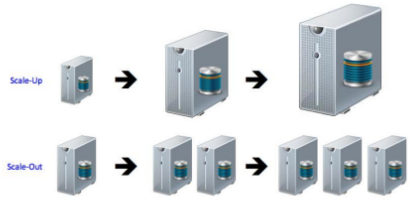


https://www.ovh.com/blog/database-replication-101/

- zajišťuje vysokou dostupnost

Škálování

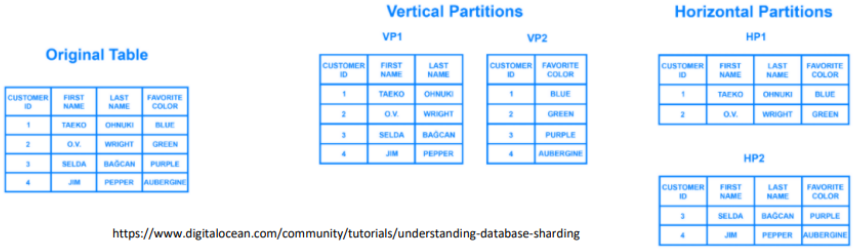
- u dokumentových databází nejčastěji horizontální škálování (ven)
  - přidání (nebo ubrání) prvků
  - ➢ přidání výpočetních uzlů
- přesun k distribuovanému paralelnímu zpracování
  - rozdělení dat mezi uzly
  - ➢ horizontální sharding
- zvýšení kapacity
  - nové komponenty jsou levné
    - základní HW
  - ➢ distribuované clustery
- cloudové služby řeší za uživatele
- ➢ big data



https://medium.com/faun/scalability-248019b918ed

Sharding

- databázový vzor pro horizontální škálování na více serverů
- rozdělení záznamů na části (partitions, shards) umístěné na různých serverech
  - např. u relačních databází rozdělení tabulky podle řádků, ne sloupců
  - např. u dokumentových databází rozdělení podle dokumentů, ne atributů

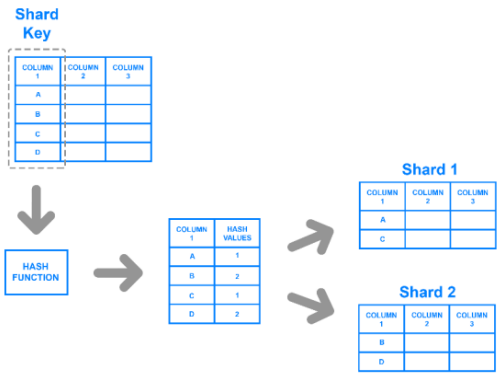


https://www.digitalocean.com/community/tutorials/understanding-database-sharding

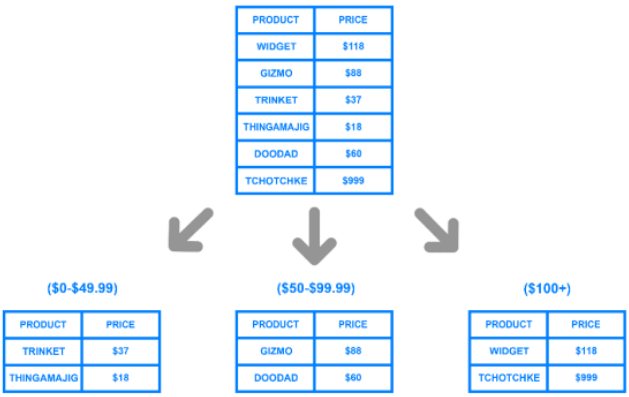
  - data mezi shardy nejsou sdílena

Architektury:

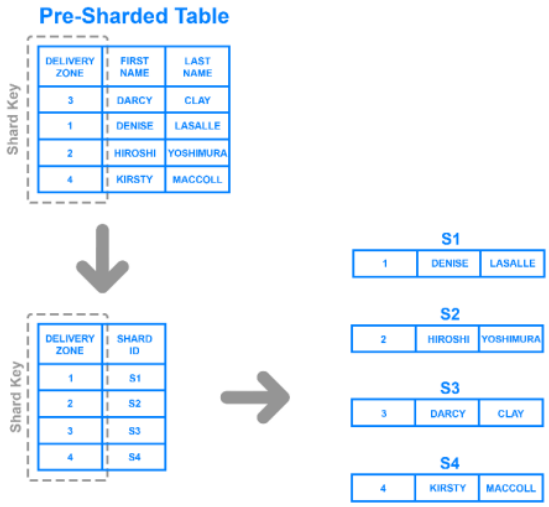
- Založena na klíči (hashi)



- Založena na rozsahu



- Založena na adresářích



Použití

- široké možnosti použití (ukládání)
  - webové aplikace
    - blogovací platformy, analytická data, nastavení uživatelů, e-reklamy, …
  - data generovaná uživateli
    - chaty, tweety, příspěvky, hodnocení, komentáře, …
  - katalogy
    - uživatelské účty, produkty, preference, …
  - počítačové hry
    - herní statistiky, žebříčky, vestavěné chaty, splněné úkoly, integrace social media, …
  - networking
    - data ze senzorů, logy, real-time analýza, …
  - …

výhody a nevýhody (převzato ze sekce MongoDB)

Výhody:

- flexibilní databáze
  - není pevně dané schéma
- ad-hoc dotazování
- vestavěna agregace
- horizontální škálování & sharding
- vysoká dostupnost
  - minimalizace nedostupnosti pomocí replikace
- rychlost
  - pokud jsou data v paměti
  - závislá na správně zvolených indexech
    - snadný přístup k dokumentům pomocí indexování
- práce s geospatial daty

Nevýhody:

- join není podporován
  - možnost doprogramovat, i tak ale pomalejší
- rychlost
  - může být výrazně pomalejší než u relačních databází
  - špatně zvolené indexy & data na disku
  - horizontální škálování & sharding
- paměťová náročnost
  - data v paměti vs. indexy v paměti
- komplexní transakce
- omezená velikost dokumentu (16 MB)
- omezené vnořování dokumentů (max 100)

MongoDB – charakteristika, architektura.

- Dokumenty ve formátu BSON (binární JSON)
- Hlavní funkce:
  - volné schéma
  - ad-hoc dotazy
    - dotazy neznámé v době vytvoření databáze
  - indexování
  - replikace
    - vysoká dostupnost
  - sharding
  - agregace

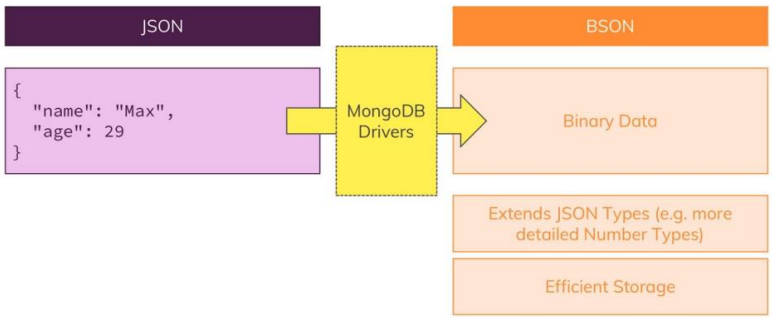
- v databázi data ukládána v kolekcích dokumentů
  - kolekce
    - seznam dokumentů
    - odpovídá relaci (tabulce)
  - dokument
    - obsahuje data
    - reprezentován pomocí vnořených objektů / map
    - ve formátu BSON
      - binární JSON
      - přidané datové typy
    - odpovídá záznamu v relaci (řádku)
  - pohled
    - pouze ke čtení
    - zdrojem je kolekce nebo jiný pohled

BSON dokument:

Looks like JSON. Example:

```
{
        "_id" : ObjectID("7b33e366ae32223aee34fd3"),
        "title" : "A blog post about MongoDB",
        "content" : "This is a blog post about MongoDB",
        "comments": [
                {
                        "name" : "Frank",
                        "email" : fkane@sundog-soft.com,
                        "content" : "This is the best article ever written!"
                        "rating" : 1
                }
        ]
}
```

JSON VS. BSON

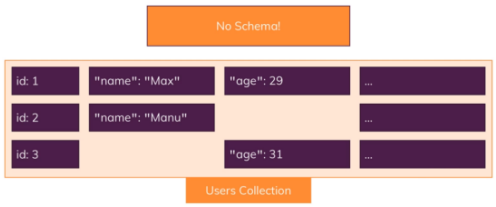


DOKUMENT

- maximum 16 MB
- volné schéma
- pole _id
  - primární klíč
  - automaticky přidán
  - ObjectID
    - 12 bytů
    - unikátní, rostoucí
- pole comments
  - obsahuje pole dalších vnořených dokumentů

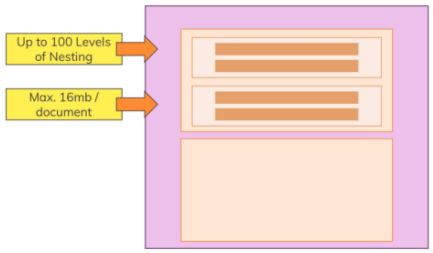
https://www.udemy.com/course/the-ultimate-hands-on-hadoop-tame-your-big-data

https://www.udemy.com/course/mongodb-the-complete-developers-guide



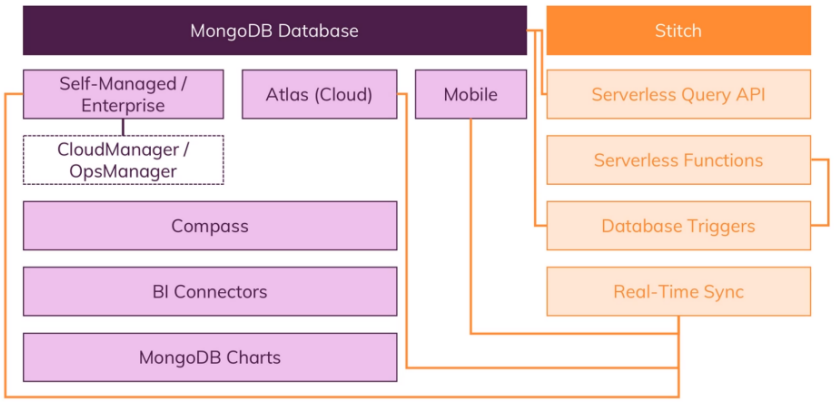
DOKUMENTY

- data patřící k sobě uložena v jednom dokumentu
  - na rozdíl od relačních databází
    - data v různých relacích propojena přes cizí klíče
    - přístup k datům přes náročné join dotazy
  - ➢ usnadňuje přístup a práci s daty
  - ➢ vazby mezi kolekcemi ideálně nejsou
    - možné ale jsou
    - ale je nutné je sloučit manuálně
      - dotaz na první dokument v první kolekci
      - dotaz na základě prvního dokumentu na druhou kolekci
- podpora vnořených dokumentů
  - embedded dokumenty
  - až 100 úrovní



https://www.udemy.com/course/mongodb-the-complete-developers-guide

MongoDB EKOSYSTÉM



PŘÍSTUPOVÉ METODY

- MongoDB shell
  - interaktivní JS interface k MongoDB
  - dotazy, updaty
  - administrativní operace
  - ➢ kompletní obsluha
- MongoDB Compass
  - GUI nadstavba
- ➢ v praxi shell užitečnější
  - práce na dálku
  - terminálová obsluha rychlejší…

Vlastnosti/Funkce

- volné schéma
- ad-hoc dotazy
- indexování
- transakce
- replikace
- sharding
- agregace
- souborový systém
- zabezpečení

1. Volné schéma

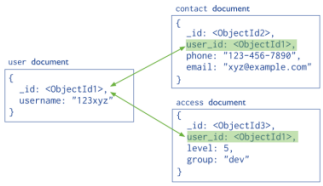
- Žádné schéma není nutné
  - ➢ každé pole v každém dokumentu může být jiné…
    - i datové typy se mohou lišit
  - není nutný jediný klíč jako v jiných databázích
    - indexy mohou být vytvořeny na různá pole
  - ➢ flexibilita
    - ad-hoc dotazy
    - i tak je dobré si návrh dobře promyslet…
      - které indexy budou potřeba pro dotazy, co budu používat?
      - vazby mezi daty?
    - join stále neefektivní
      - NoSQL databáze…

Dva druhy vazeb – Reference a emmbeded dokumenty

- embedded dokumenty
  - nenormalizovaný model
  - veškerá relevantní data v jednom dokumentu
  - kdy použít?
    - vazby 1:1
    - vazby 1:N
  - výhody
    - lepší výkon pro operace čtení
    - přístup k relevantním datům v rámci jedné db operace
    - update dat v rámci jedné atomické operace zápisu

reference

- potřeba normalizovaného datového modelu
- relevantní data v různých dokumentech (kolekcích)
- kdy použít?
  - vnořené dokumenty tvoří velké množství duplikátů a zároveň neurychlují přístup k datům
  - vazby M:N
  - velké hierarchické datasety
- join kolekcí pomocí
  - $lookup
    - left outer join
    - agregace
  - $graphLookup
    - rekurzivní vyhledávání



Když je potřeba pevné schéma – využití **validátoru** (pokud struktura nesedí vyhodí vyjímku)

2. Ad-hoc dotazy

AD-HOC DOTAZY

dotazy neznámé při návrhu databáze

- vytvořené až při potřebě získat určitou informaci

podpora

- hledání podle pole dokumentu
- rozsahové dotazy
- hledání podle regulárních výrazů

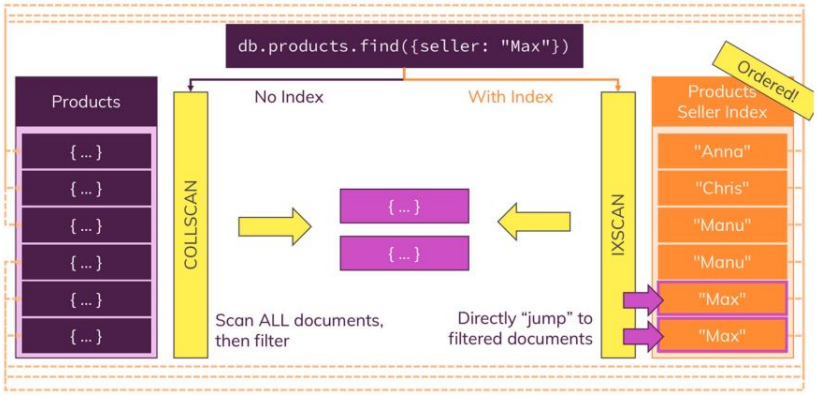
dotazy mohou vracet

- specifická pole dokumentu
- náhodný vzorek výsledků o dané velikosti
- a také obsahovat uživatelem definované JS funkce

3.

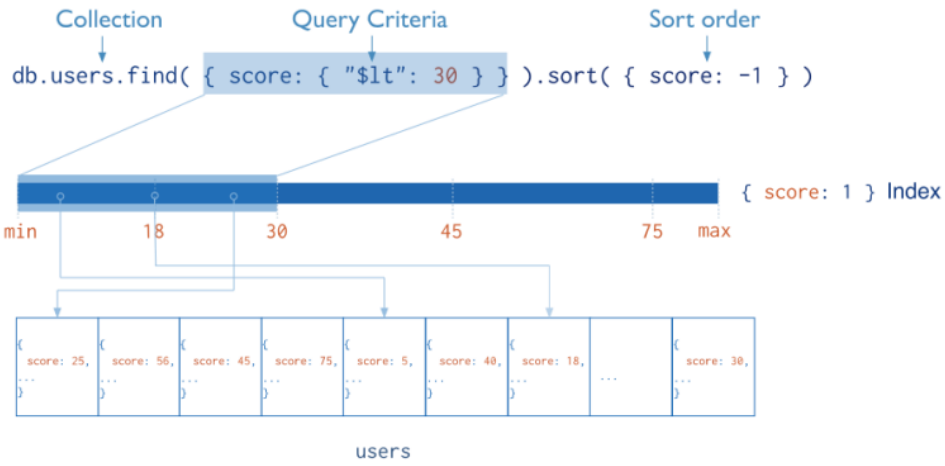
3. Indexování

- vylepšuje rychlost vyhledávání
  - bez indexování se provádí sken celé kolekce
  - vhodný index výrazně omezuje dokumenty, které je potřeba skenovat
- index
  - speciální datová struktura definovaná na úrovni kolekcí
  - ukládá hodnotu specifického pole v seřazené formě
    - snadné procházení a porovnávání
    - obsahuje také pointer na celý dokument pro snadný přístup
  - možnost definovat nad libovolným polem ale i nad jejich kombinací
    - v základu index _id
  - využití i pro rychlé řazení
  - základ pro sharding
    - jen jeden může být použit



- v praxi ale index není zadarmo
- cenou je vkládání / aktualizace
  - při každém vložení je potřeba vložit prvek i do seřazeného indexu
  - potřeba vložit do všech indexů
  - pomalé?
- indexy je potřeba řádně promyslet

- indexování je pomalejší, když dotaz vrací velkou část kolekce
  - způsobeno přechodem index – pointer – dokument
  - naopak při kompletním skenu už jsou dokumenty načteny v paměti



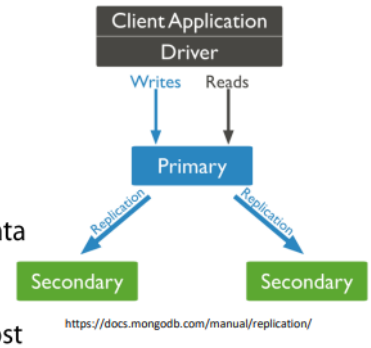
- vytvoření / aktualizace indexu dočasně uzamkne kolekci
  - možnost ovlivnit parametrem background
- typy indexů
  - single field
    - index nad jedním polem
  - compound
    - index nad více poli
  - multikey
    - index pro prvky v poli (array)
  - text
    - index pro vyhledávání textových řetězců
  - geospatial
  - hashed

3. Transakce

- operace nad jedním dokumentem atomické
  - vazby mezi daty často zachyceny pomocí vnořených dokumentů místo normalizovaných dokumentů a kolekcí
  - ➢ multi-dokumentové transakce často nejsou potřeba
    - v určitých případech ale chybí
- ➢ podpora multi-dokumentových transakcí od verze 4.0
  - transakce mohou být provedeny nad
    - databázemi
    - shardy
    - kolekcemi
    - dokumenty
    - operacemi

4. Replikace

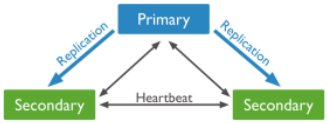
- MongoDB
  - master – slave architektura
    - 1 master -> CP
  - primární – sekundární uzly
  - asynchronní
  - sada replik (replica sets)
    - skupina mongod procesů obsahující stejná data
      - datové uzly (primární a sekundární)
      - volitelně arbiter (max 1)
    - poskytují redundanci dat a vysokou dostupnost
      - vyšší odolnost vůči výpadkům a ztrátám serverů
      - v určitých případech zrychlení čtení
      - možnost kopií pro specifické účely
        - ochrana proti lidským chybám, disaster recovery, logování, záloha, …



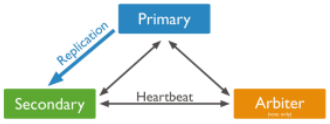
https://docs.mongodb.com/manual/replication/

REPLIKACE

- primární uzel
  - existuje jen jeden
  - obstarává veškeré operace zápisu
  - zaznamenává operace do operačního logu
    - oplog



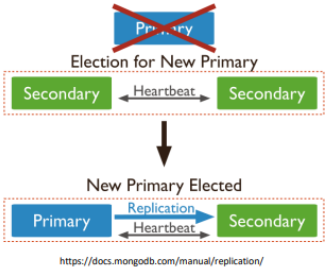
- sekundární uzly
  - obsahují stejná data jako primární uzel
  - replikují operační log a aplikují operace na svá data
    - aplikováno asynchronně
    - ➢ aby data opět odpovídala primárnímu uzlu
- arbiter
  - speciální případ, který neobsahuje data
  - využití při hlasování o novém primárním uzlu



https://docs.mongodb.com/manual/replication/

REPLIKACE

- co se stane při výpadku primárního uzlu?
  - primární uzel nekomunikuje déle jak stanovený limit
    - chybějící reakce na heartbeats (ping) posílané mezi uzly každé 2 sekundy
    - v základu 10 sekund
  - ➢ jeden ze sekundárních uzlů zažádá o zvolení nového primárního uzlu
    - většina sekundárních uzlů se musí shodnout
      - ➢ lichý počet serverů
      - volitelně arbiter
    - v základním nastavení by nemělo trvat déle než 12 sekund
  - i tak je ale porušena dostupnost
    - operace zápisu nejsou možné, dokud není zvolen nový primární uzel
    - čtení je možné, pokud je povoleno ze sekundárních uzlů

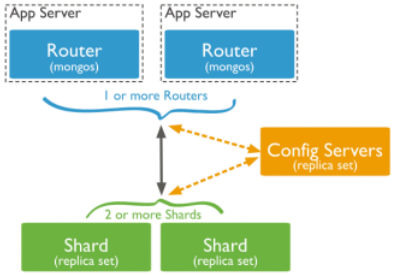


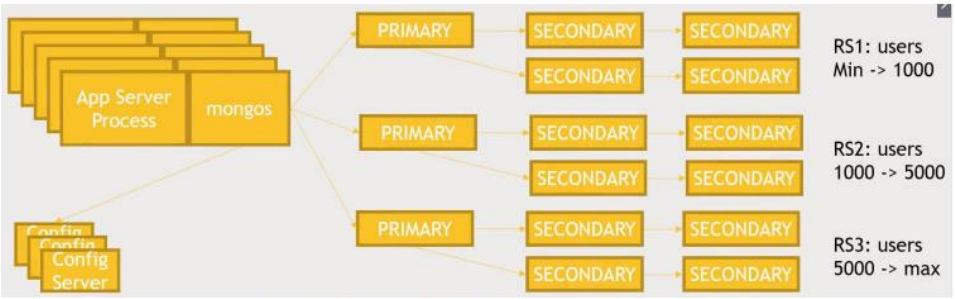
https://docs.mongodb.com/manual/replication/

- operace čtení
  - v základu z primárního uzlu
  - je ale možné i ze sekundárních
    - asynchronní replikace nezaručuje, že budou přečtena stejná data jako z primárního uzlu
    - není možné pro multi-dokumentové transakce

5. Sharding

- shardy – více sad replik (replica sets)
  - každá sada odpovídá za rozsah hodnot
  - ➢ musí být definovaný index na kolekci
    - ➢ slouží k určení rozsahu hodnot a vyvážení zátěže
- služba mongos
  - v serveru komunikujícím s databází
  - komunikuje se 3 konfiguračními servery
    - poskytují informaci, jak jsou data rozdělena
  - vybere odpovídající sadu replik



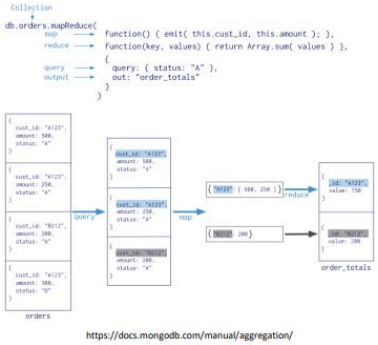


https://www.udemy.com/course/the-ultimate-hands-on-hadoop-tame-your-big-data

- v praxi relativně pomalé a komplikované
  - závisí na správné volbě indexů
  - ➢ Cassandra vhodnější pro horizontální škálování

6. Agregace

- dávkové zpracování dokumentů vracející jeden výsledek i po provedení celé řady operací
- tři možnosti v MongoDB
  - agregační roura (pipeline)
    - dokumenty jsou zpracovávány postupně v krocích až do konečného výsledku
  - jednoúčelová agregace
    - pro dokumenty v jedné kolekci
  - map-reduce (deprecated)
    - dvě fáze
    - map – mapování
      - zpracování dokumentů do objektů odpovídajících vstupním dokumentům
    - reduce – redukce
      - zkombinování výstupů z mapování



https://docs.mongodb.com/manual/aggregation/

Agregační roura:

- založena na principu roury na zpracování dat
- dokumenty vstupují do vícefázové roury, která je transformuje na agregovaný výsledek
  - každá fáze transformuje dokumenty pro další fázi roury
  - ne každá fáze vytváří pro každý vstupní dokument výstupní dokument
    - některé, např., dokumenty filtrují
  - většina fází se může i opakovat
  - po průchodu celou rourou je získán výsledek

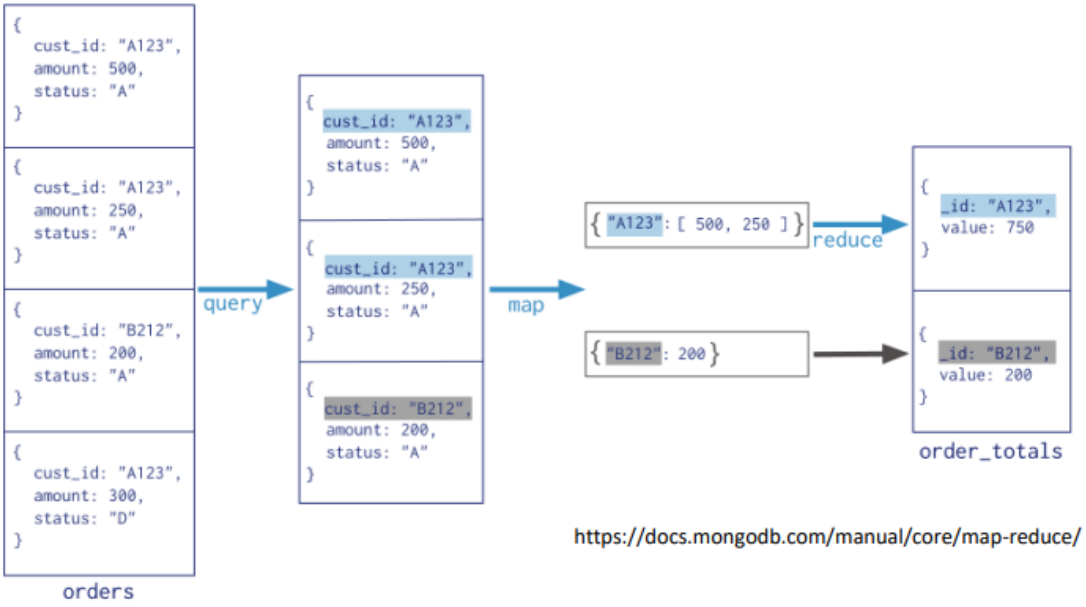
```
db.orders.aggregate([
   { $match: { status: "A" } },
   { $group: { _id: "$cust_id", total: { $sum: "$amount" } } }
])
```

https://docs.mongodb.com/manual/aggregation/

- první fáze
  - fáze *$match* filtruje dokumenty na základě pole *status*
  - do další fáze předává dokumenty, u kterých *status* odpovídá hodnotě *„A“*
- druhá fáze
  - fáze *$group* shlukuje dokumenty na základě pole *cust_id*
  - pro každé unikátní *cust_id* je spočítána suma polí *amount*
- různé fáze
  - filtrování (odpovídá dotazům)
  - transformace dokumentů
  - shlukování a řazení podle specifických polí
  - agregace obsahu polí
  - použití operátorů (např. průměr, zřetězení řetězců, …)
  - (interní optimalizační)
- může využívat indexů v některých fázích pro zrychlení
- použitelná i na shardované kolekce
- preferovaná metoda pro agregaci dat v MongoDB

Map-reduce:

- princip paralelního zpracování
  - cluster databázových serverů
  - jeden z uzlů (master) přijme požadavek map-reduce od klienta
    - v některých modelech se může jednat o libovolný uzel
  - master rozesílá funkci *map* všem ostatním uzlům
  - uzly paralelně provádí kód funkce *map* a vrací masteru výsledky
    - výsledky mohou být i duplicitní
      - určitá odolnost proti výpadkům
    - sám master může také provádět funkci *map*
  - po obdržení dostatečného množství výsledků (nebo vypršení času) provádí master nad daty funkci *reduce*
    - odstraní duplicitní data
    - provádí operace, které je možné provést jen nad kompletní sadou výsledků ze všech uzlů
    - agreguje výsledky
  - možné vrácení výsledků klientovi



https://docs.mongodb.com/manual/core/map-reduce/

Jednoúčelová agregace:

- MongoDB také poskytuje operace
  - *db.collection.count()*
    - vrací počet dokumentů v kolekci (nebo pohledu), které by odpovídaly dotazu *find*
    - počítá výsledky odpovídající danému dotazu
    - deprecated
  - *db.collection.estimatedDocumentCount()*
    - vrací počet dokumentů v kolekci (nebo pohledu)
    - zaobaluje příkaz *count*
  - *db.collection.distinct()*
    - vrací unikátní hodnoty pro dané pole
  - agregují dokumenty z jedné kolekce
  - výhodou je jednoduchý přístup k základním agregačním procesům
  - nevýhodou je nedostatek flexibility a možností ostatních přístupů