



AGREGACE A MongoDB

Lukáš Matějů

8.4.2024 | DPB



ČÁST I.: OPAKOVÁNÍ



OPAKOVÁNÍ

- MongoDB
 - dokumentová databáze
 - funkce / vlastnosti
 - volné schéma
 - ad-hoc dotazy
 - indexování
 - transakce
 - replikace
 - sharding
 - agregace
 - souborový systém
 - zabezpečení
 - výhody?
 - omezení?



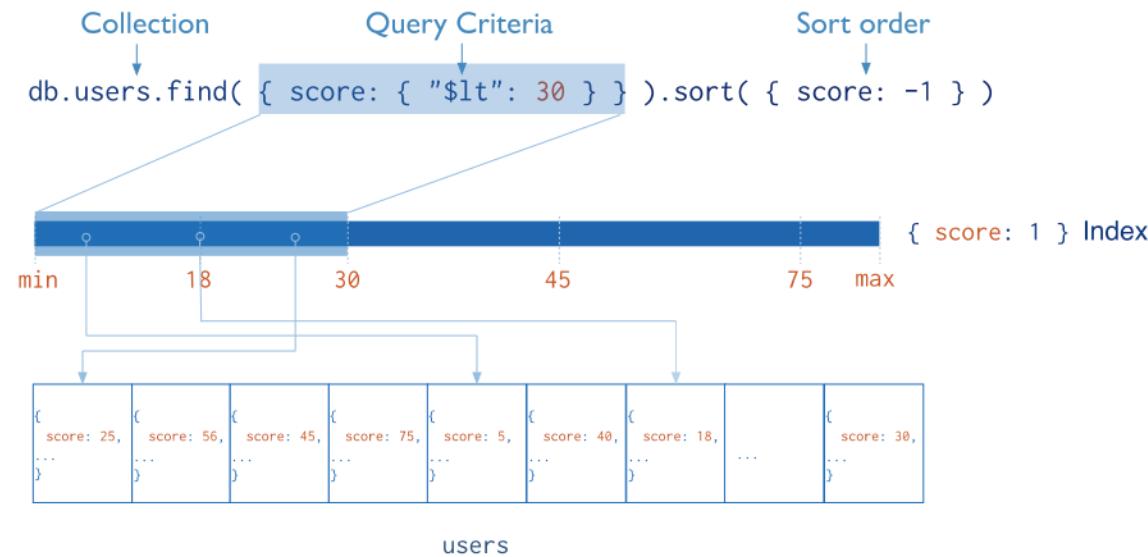
OPAKOVÁNÍ

- **indexování**
 - vylepšuje rychlosť vyhľadávania
 - bez indexovania se provádí sken celé kolekcie
 - vhodný index výrazne omezuje dokumenty, ktoré je potreba skenovať
- **index**
 - speciálna datová struktura definovaná na úrovni kolekcií
 - ukladá hodnotu specifického pole v seřazené formě
 - snadné procházení a porovnávání
 - obsahuje také pointer na celý dokument pre snadný prístup
 - možnosť definovať nad libovolným polem ale i nad jejich kombináciami
 - v základe index _id
 - **základ pre sharding**
 - jen jeden môže byť použit



OPAKOVÁNÍ

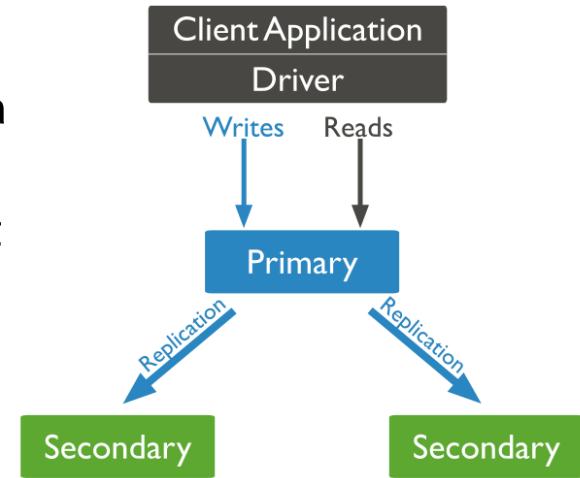
- indexování



<https://docs.mongodb.com/manual/indexes/>

OPAKOVÁNÍ

- replikace
 - master – slave architektura
- sada replik (replica sets)
 - skupina mongod procesů obsahující stejná data
 - datové uzly (primární, sekundární), volitelně arbiter
 - poskytují redundanci dat a vysokou dostupnost
 - vyšší odolnost vůči výpadkům a ztrátám serverů
 - primární uzel
 - obstarává veškeré operace zápisu
 - zaznamenává operace do operačního logu
 - sekundární uzly
 - obsahují stejná data jako primární uzel
 - replikují operační log a aplikují operace na svá data
 - aplikováno asynchronně

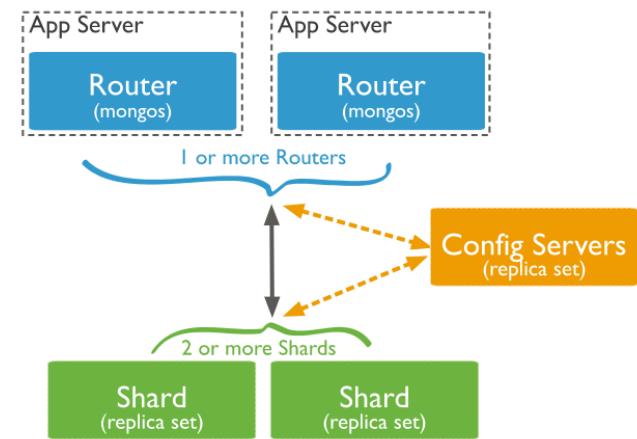


<https://docs.mongodb.com/manual/replication/>



OPAKOVÁNÍ

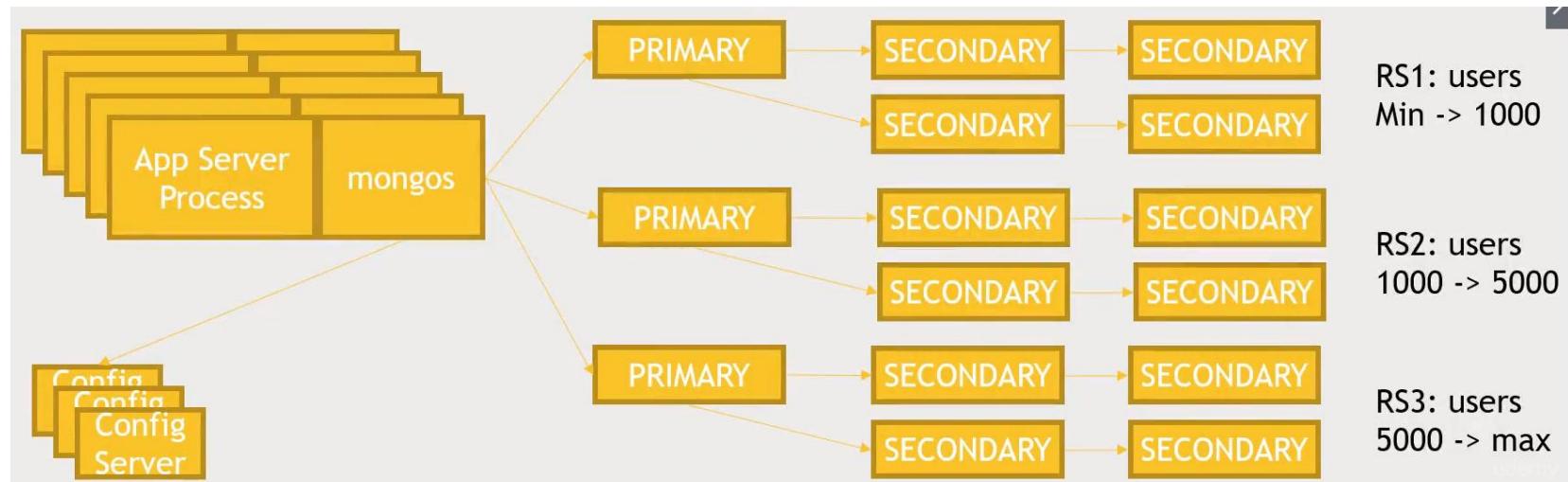
- sharding
 - horizontální škálování na více než 1 server
 - shardy – více sad replik (replica sets)
 - každá sada odpovídá za rozsah hodnot
 - musí být definovaný index na kolekci
 - slouží k určení rozsahu hodnot a vyvážení zátěže
 - služba mongos
 - v serveru komunikujícím s databází
 - komunikuje se třemi konfiguračními servery
 - poskytuje informaci, jak jsou data rozdělena
 - vybere odpovídající sadu replik



<https://docs.mongodb.com/manual/sharding/>



OPAKOVÁNÍ





ČÁST II.: AGREGACE A MongoDB



AGREGACE

- dávkové zpracování dokumentů vracející kompaktní výsledek i po provedení celé řady operací
 - první fáze – shluknutí hodnot z různých dokumentů
 - druhá fáze – provedení operací na shluknutých datech a vrácení výsledku
- tři možnosti v MongoDB
 - agregační roura (pipeline)
 - dokumenty jsou zpracovávány postupně v krocích až do konečného výsledku
 - map-reduce (dnes již deprecated)
 - map – mapování
 - zpracování dokumentů do objektů odpovídajících vstupním dokumentům
 - reduce – redukce
 - zkombinování výstupů z mapování
 - jednoúčelová agregace
 - pro dokumenty v jedné kolekci





ČÁST III.: AGREGAČNÍ ROURA



AGREGAČNÍ ROURA

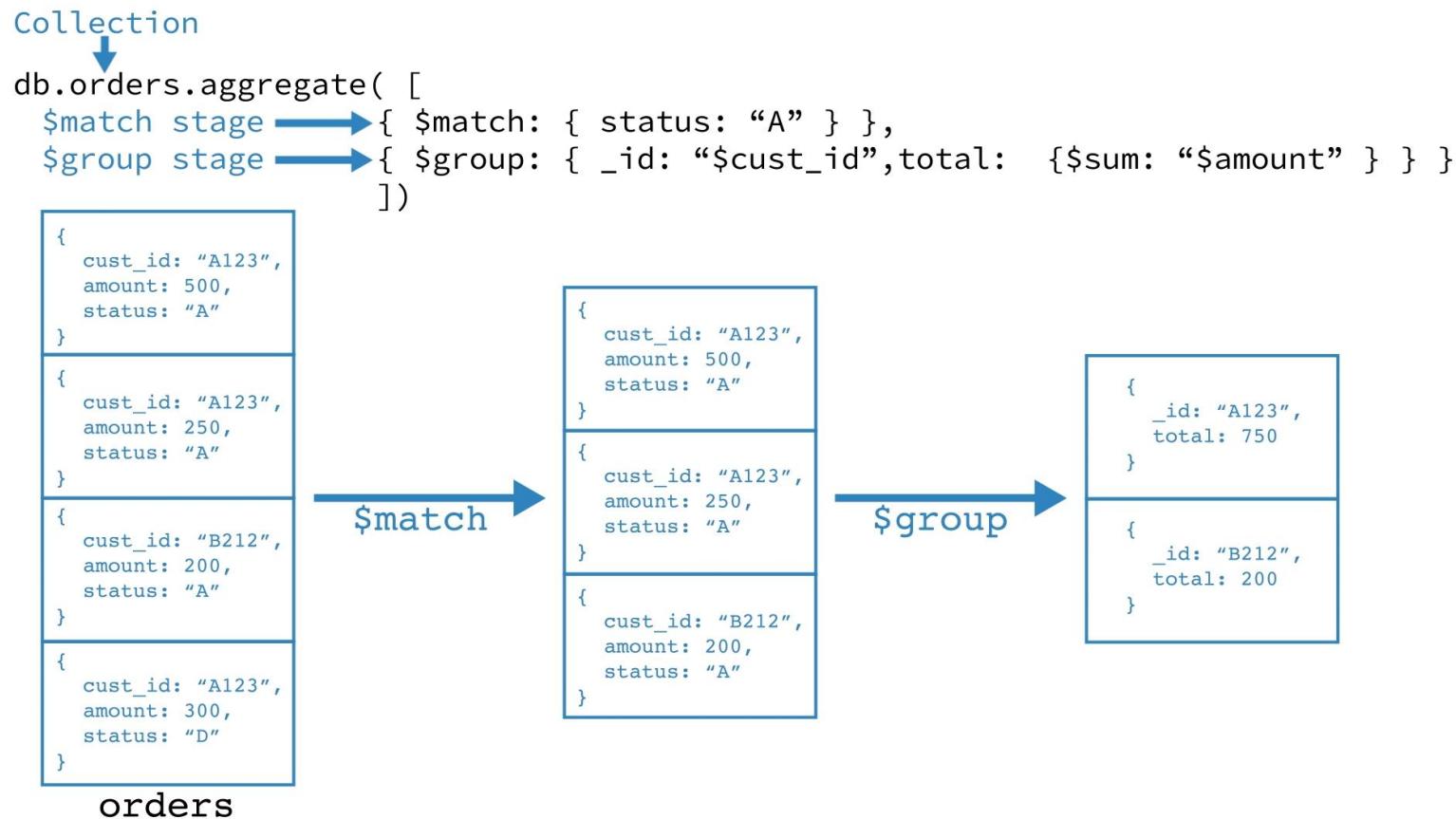
- založena na principu roury na zpracování dat
- dokumenty vstupují do vícefázové roury, která je transformuje na agregovaný výsledek
 - každá fáze transformuje dokumenty pro další fázi roury
 - ne každá fáze vytváří pro každý vstupní dokument výstupní dokument
 - některé např. dokumenty filtroují
 - 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/>



AGREGAČNÍ ROURA



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



AGREGAČNÍ ROURA

```
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*





AGREGAČNÍ ROURA

- 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 aggregaci dat v MongoDB



UKÁZKA

- ukázka použití agregační roury z manuálu MongoDB
<https://docs.mongodb.com/manual/tutorial/aggregation-with-user-preference-data/>
- datový model
 - databáze hypotetického sportovního klubu
 - kolekce uživatelů
 - jméno jako _id
 - vstup do klubu
 - oblíbené sporty

```
{  
  _id : "jane",  
  joined : ISODate("2011-03-02"),  
  likes : ["golf", "racquetball"]  
}  
{  
  _id : "joe",  
  joined : ISODate("2012-07-02"),  
  likes : ["tennis", "golf", "swimming"]  
}
```



UKÁZKA

- normalizace a seřazení dokumentů

```
db.users.aggregate(  
  [  
    { $project : { name:{$toUpperCase:"$_id"} , _id:0 } },  
    { $sort : { name : 1 } }  
  ]  
)
```

- vrací uživatelská jména ve velkých písmenech a v abecedním pořadí
- operátor *\$project*
 - vytváří nové pole *name*
 - převádí *_id* na velká písmena (pomocí operátoru *\$toUpperCase*) a ukládá do pole *name*
 - potlačuje pole *_id* (předáváno automaticky, pokud není potlačeno)
- operátor *\$sort*
 - řadí výsledky podle pole *name*

```
{  
  "name" : "JANE"  
},  
,  
{  
  "name" : "JILL"  
},  
,  
{  
  "name" : "JOE"  
}
```



UKÁZKA

- uživatelé seřazení podle data přidání
 - vhodné např. pro obnovení členství
 - operátor *\$project*
 - vytváří nová pole *month_joined* a *name*
 - potlačuje pole *_id*
 - operátor *\$month* konvertuje hodnoty pole *joined* na integer reprezentaci měsíce
 - přiřazuje tuto reprezentaci do pole *month_joined*
 - operátor *\$sort*
 - řadí výsledky podle pole *month_joined*

```
db.users.aggregate(  
[  
  { $project :  
    {  
      month_joined : { $month : "$joined" },  
      name : "$_id",  
      _id : 0  
    }  
  },  
  { $sort : { month_joined : 1 } }  
]  
)  
  
{  
  "month_joined" : 1,  
  "name" : "harold"  
},  
{  
  "month_joined" : 1,  
  "name" : "kate"  
}  
{  
  "month_joined" : 2,  
  "name" : "jill"  
}
```



UKÁZKA

- počet nových členů za jednotlivé měsíce

```
db.users.aggregate(  
  [  
    { $project : { month_joined : { $month : "$joined" } } } ,  
    { $group : { _id : {month_joined:"$month_joined"} , number : { $sum : 1 } } },  
    { $sort : { "_id.month_joined" : 1 } }  
  ]  
)
```

- operátor *\$project*
 - vytváří nové pole *month_joined*
 - operátor *\$month* konvertuje hodnoty pole *joined* na integer reprezentaci měsíce
 - přiřazuje tuto reprezentaci do pole *month_joined*
- operátor *\$group*
 - sbírá všechny dokumenty se stejnou hodnotou *month_joined* a jejich počet
 - pro každou unikátní hodnotu vytváří dokument (pro každý měsíc) se dvěma polí
 - _id* – obsahuje vnořený dokument s polem *month_joined* a jeho hodnotou
 - number* – generované pole, které je inkrementované operátorem *\$sum* za každý dokument s odpovídající hodnotou *month_joined*



UKÁZKA

- počet nových členů za jednotlivé měsíce

```
db.users.aggregate(  
  [  
    { $project : { month_joined : { $month : "$joined" } } },  
    { $group : { _id : {month_joined:"$month_joined"}, number : { $sum : 1 } } },  
    { $sort : { "_id.month_joined" : 1 } }  
  ]  
)
```

```
{  
  "_id" : {  
    "month_joined" : 1  
  },  
  "number" : 3  
},  
,  
{  
  "_id" : {  
    "month_joined" : 2  
  },  
  "number" : 9  
},  
,  
{  
  "_id" : {  
    "month_joined" : 3  
  },  
  "number" : 5  
}
```

- operátor `$sort`
 - řadí dokumenty vytvořené fází `$group` podle pole `month_joined`



UKÁZKA

- 5 nejoblíbenějších sportů
 - užitečné pro plánování rozvoje
 - roura pracuje se všemi dokumenty v kolekci *users* a zpracovává je následujícími operacemi
 - operátor *\$unwind*
 - separuje každou hodnotu v poli *likes*
 - pro každou hodnotu vytváří novou verzi dokumentu

```
{  
  _id : "jane",  
  joined : ISODate("2011-03-02"),  
  likes : ["golf", "racquetball"]  
}
```

```
db.users.aggregate(  
[  
  { $unwind : "$likes" },  
  { $group : { _id : "$likes" , number : { $sum : 1 } } },  
  { $sort : { number : -1 } },  

```

```
{  
  _id : "jane",  
  joined : ISODate("2011-03-02"),  
  likes : "golf"  
}  
  
{  
  _id : "jane",  
  joined : ISODate("2011-03-02"),  
  likes : "racquetball"  
}
```



UKÁZKA

- 5 nejoblíbenějších sportů
 - operátor `$group`
 - sbírá všechny dokumenty se stejnou hodnotou pole `likes` a počítá jejich množství
 - na základě této informace vytváří nový dokument se dvěma poli
 - `_id` – obsahuje hodnotu pole `likes`
 - `number` – generované pole inkrementované operátorem `$sum` za každý dokument s odpovídající hodnotou `likes`
 - operátor `$sort`
 - řadí sestupně dokumenty podle pole `number`
 - operátor `$limit`
 - vybírá jen prvních 5 dokumentů

```
db.users.aggregate(  
[  
  { $unwind : "$likes" },  
  { $group : { _id : "$likes" , number : { $sum : 1 } } },  
  { $sort : { number : -1 } },  
  { $limit : 5 }  
]
```



UKÁZKA

- 5 nejoblíbenějších sportů

```
db.users.aggregate(  
  [  
    { $unwind : "$likes" },  
    { $group : { _id : "$likes" , number : { $sum : 1 } } },  
    { $sort : { number : -1 } },  
    { $limit : 5 }  
  ]  
)
```

```
{  
  "_id" : "golf",  
  "number" : 33  
},  
{  
  "_id" : "racquetball",  
  "number" : 31  
},  
{  
  "_id" : "swimming",  
  "number" : 24  
},  
{  
  "_id" : "handball",  
  "number" : 19  
},  
{  
  "_id" : "tennis",  
  "number" : 18  
}
```

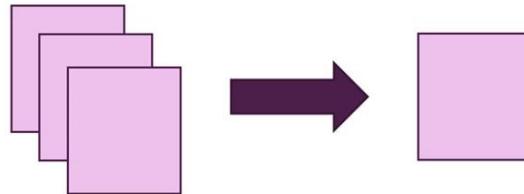


UKÁZKA

- \$group vs. \$project

\$group

n:1



Sum, Count, Average, Build Array

\$project

1:1



Include/ Exclude Fields, Transform
Fields (within a Single Document)

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



ČÁST IV.: MAP-REDUCE



MAP-REDUCE

- paradigmata paralelního zpracování dat
 - pro převedení velkého množství dat na užitečné agregované výsledky
 - zpracování je rozdělené na dvě základní operace
 - map – data jsou rozdělena na chunky a mohou být paralelně zpracována
 - reduce – slučuje výsledky z operace map do finálního výsledku
- paralelní výpočty na obrovských datech v clusterech
 - data zpracovávána souborovým systémem nebo databází
 - strukturovaná / nestrukturovaná data
- primární použití
 - data, která jsou větší než paměť jednoho stroje
 - analýza grafů, strojové učení, klasifikace, klasterování dokumentů, ...
- big data

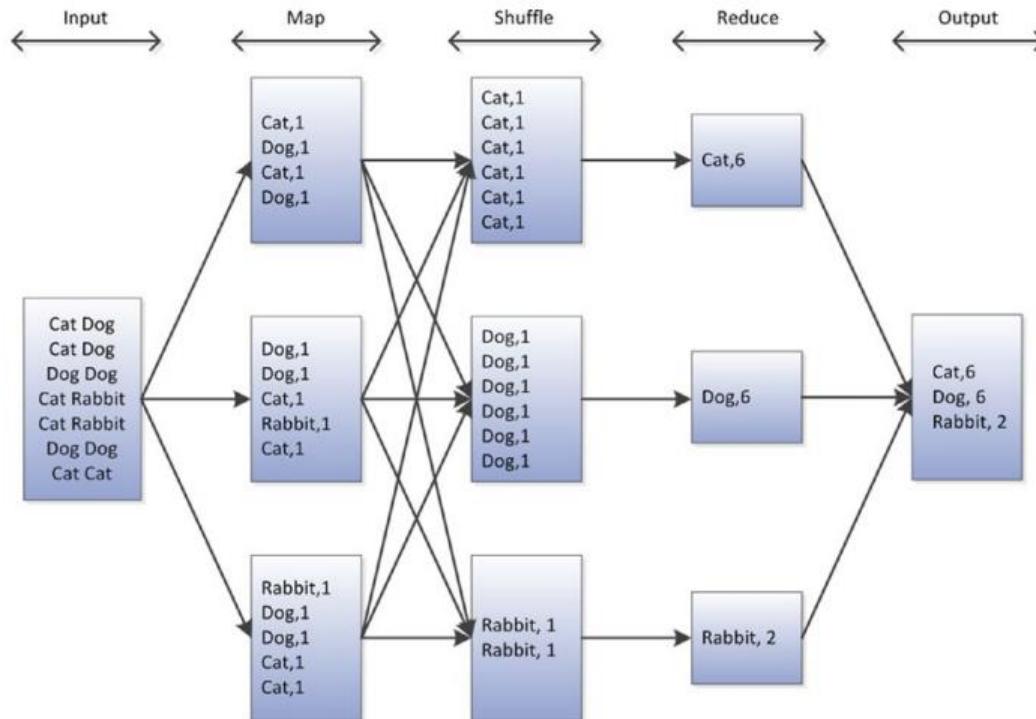


MAP-REDUCE

- princip paralelního zpracování
 - 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
 - sám master může také provádět funkci *map*
 - po obdržení dostatečného množství výsledků master připravuje data
 - odstranění duplicit, setřídění a přerozdělení dat
 - obecně operace proveditelné jen nad kompletní sadou výsledků ze všech uzlů
 - master rozesílá funkci *reduce* všem ostatním uzlům
 - uzly paralelně provádí kód funkce *reduce* a vrací agregované výsledky
 - master výsledky sloučí (a případně upraví) a vrací klientovi



MAP-REDUCE



HARRISON, Guy. *Next generation databases: NoSQL, NewSQL, and Big Data*. ISBN 978-1484213308.



MAP-REDUCE

- map-reduce v MongoDB
 - využívá JavaScript funkcí pro operace *map*, *reduce* a volitelně *finalize*
 - spouštěny procesem *mongod*
 - vstupem jsou dokumenty z jedné kolekce
 - map-reduce má schopnost data setřídit a omezit ještě před *map* operací
 - operace *map*
 - mapuje hodnoty ke klíči
 - pokud pro daný klíč existuje více hodnot, mapovány budou všechny
 - funkce *emit(klíč, hodnota)* vrací výstupní dokumenty spojující klíče s hodnotami
 - operace *reduce*
 - bere klíč a agreguje k němu namapované hodnoty
 - jako vstup bere výstup z operace *map*, který zpracovává
 - volitelně operace *finalize*
 - dále zpracovává výstup z operace *reduce*



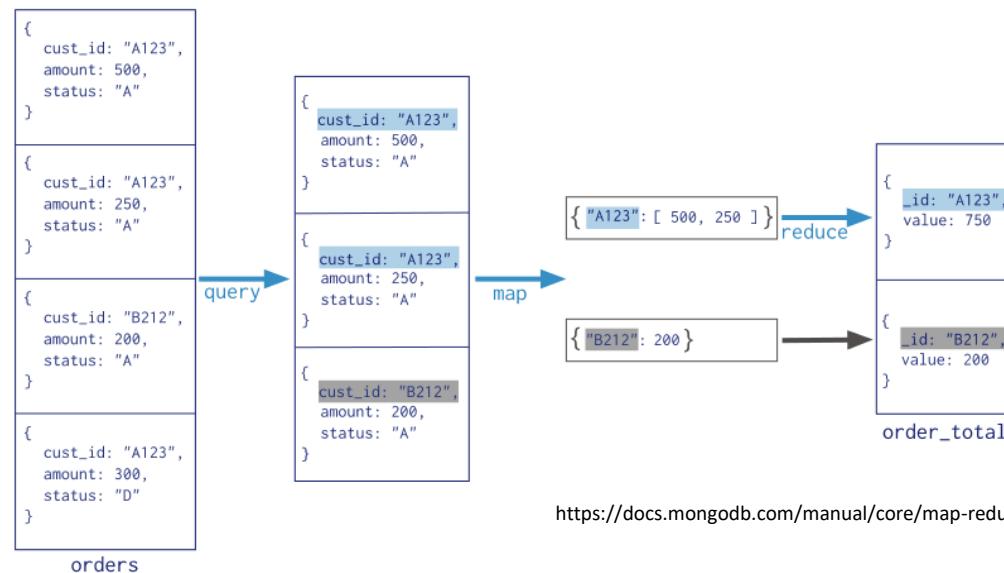
MAP-REDUCE

- map-reduce v MongoDB
 - výstupem je dokument
 - musí dodržet velikostní limit BSON dokument formátu
 - výstupem je kolekce
 - existuje možnost provést další map-reduce nad původní kolekcí a výsledky zpracovat s předchozími výsledky
 - sloučit
 - nahradit
 - redukovat
 - incremental map-reduce
 - vstupem i výstupem může být shardovaná kolekce
 - u vstupu se o vše stará *mongos*
 - rozesílá map-reduce na všechny shardy paralelně a čeká na jejich dokončení



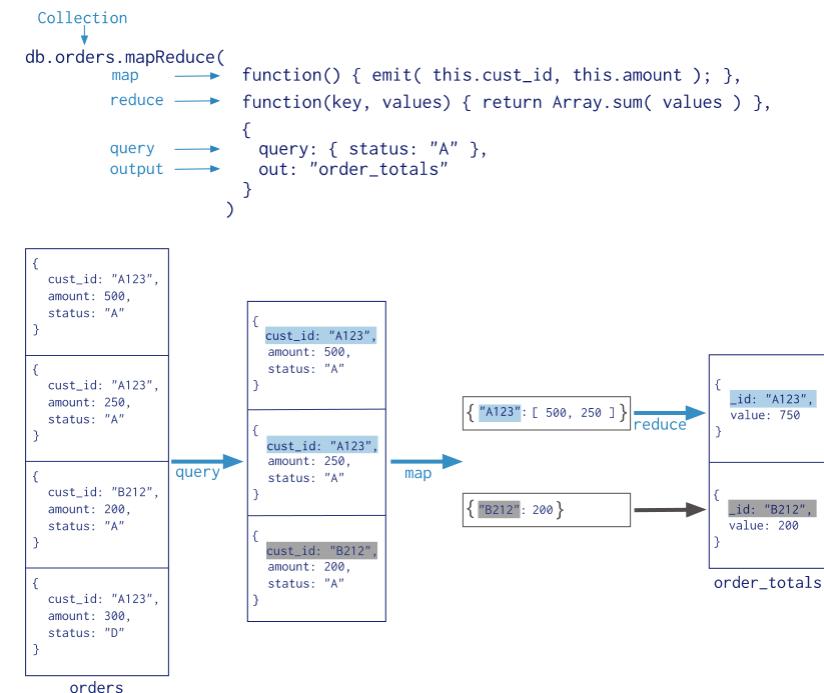
MAP-REDUCE

```
Collection
↓
db.orders.mapReduce(
  map → function() { emit( this.cust_id, this.amount ); },
  reduce → function(key, values) { return Array.sum( values ) },
  query → { query: { status: "A" } },
  output → { out: "order_totals" }
)
```



MAP-REDUCE

- ukázkový příklad
 - nejprve jsou vybrány jen dokumenty se *status* = „A“
 - řádek query
 - fáze *map*
 - mapuje hodnoty *amount* ke klíči *cust_id*
 - vrací páry klíč – hodnota
 - fáze *reduce*
 - aplikována u klíčů s více hodnotami
 - agreguje data na jednu hodnotu
 - vrácena kolekce *order_totals*
 - řádek output



UKÁZKA

- ukázka použití map-reduce z manuálu MongoDB
<https://docs.mongodb.com/manual/tutorial/map-reduce-examples/>
- datový model
 - kolekce objednávek
 - zákazník
 - datum objednání
 - cena
 - pole objednaných položek
 - zboží
 - množství
 - cena za kus
 - status

```
db.orders.insertMany([
  { _id: 1, cust_id: "Ant O. Knee", ord_date: new Date("2020-03-01"), price: 25, items: [ { sku: "oranges", qty: 5, price: 2.5 }, { sku: "apples", qty: 5, price: 2.5 } ], status: "A" },
  { _id: 2, cust_id: "Ant O. Knee", ord_date: new Date("2020-03-08"), price: 70, items: [ { sku: "oranges", qty: 8, price: 2.5 }, { sku: "chocolates", qty: 5, price: 10 } ], status: "R" },
  { _id: 3, cust_id: "Busby Bee", ord_date: new Date("2020-03-08"), price: 50, items: [ { sku: "oranges", qty: 10, price: 2.5 }, { sku: "pears", qty: 10, price: 2.5 } ], status: "A" },
  { _id: 4, cust_id: "Busby Bee", ord_date: new Date("2020-03-18"), price: 25, items: [ { sku: "oranges", qty: 10, price: 2.5 } ], status: "A" },
  { _id: 5, cust_id: "Busby Bee", ord_date: new Date("2020-03-19"), price: 50, items: [ { sku: "chocolates", qty: 5, price: 10 } ], status: "A" },
  { _id: 6, cust_id: "Cam Elot", ord_date: new Date("2020-03-19"), price: 35, items: [ { sku: "carrots", qty: 10, price: 1.0 }, { sku: "apples", qty: 10, price: 2.5 } ], status: "A" },
  { _id: 7, cust_id: "Cam Elot", ord_date: new Date("2020-03-20"), price: 25, items: [ { sku: "oranges", qty: 10, price: 2.5 } ], status: "A" },
  { _id: 8, cust_id: "Don Quis", ord_date: new Date("2020-03-20"), price: 75, items: [ { sku: "chocolates", qty: 5, price: 10 }, { sku: "apples", qty: 10, price: 2.5 } ], status: "A" },
  { _id: 9, cust_id: "Don Quis", ord_date: new Date("2020-03-20"), price: 55, items: [ { sku: "carrots", qty: 5, price: 1.0 }, { sku: "apples", qty: 10, price: 2.5 }, { sku: "oranges", qty: 10, price: 2.5 } ], status: "A" },
  { _id: 10, cust_id: "Don Quis", ord_date: new Date("2020-03-23"), price: 25, items: [ { sku: "oranges", qty: 10, price: 2.5 } ], status: "A" }
])
```



UKÁZKA

- celková útrata jednotlivých zákazníků
 - map-reduce operace nad kolekcí *orders*, která seskupí objednávky podle pole *cust_id* a vypočítá sumu pole *price* pro každé *cust_id*
- 1) definování *map* funkce, která zpracuje každý vstupní dokument

```
var mapFunction1 = function() {  
    emit(this.cust_id, this.price);  
};
```

- *this* odpovídá současně zpracovávanému dokumentu
- funkce mapuje hodnoty *price* na klíč *cust_id* pro každý dokument
- vrací *cust_id* a *price* páry



UKÁZKA

- celková útrata jednotlivých zákazníků
- 2) definování odpovídající *reduce* funkce se dvěma parametry *keyCustomerId* a *valuesPrices*

```
var reduceFunction1 = function(keyCustomerId, valuesPrices) {  
    return Array.sum(valuesPrices);  
};
```

- *valuesPrices* je pole, jehož elementy jsou hodnoty *price* získané z *map* funkce a sdružené podle *keyCustomerId*
- funkce redukuje pole *valuesPrices* na sumu jeho elementů





UKÁZKA

- celková útrata jednotlivých zákazníků
- 3) provedení map-reduce operace na všech dokumentech kolekce *orders*
 - zavoláním funkcí *mapFunction1* a *reduceFunction1*

```
db.orders.mapReduce(  
    mapFunction1,  
    reduceFunction1,  
    { out: "map_reduce_example" }  
)
```

- výstup je uložen do kolekce *map_reduce_example*
- jestli kolekce již existuje, její obsah je nahrazen





UKÁZKA

- celková útrata jednotlivých zákazníků

4) ověření výsledků

- dotaz na kolekci *map_reduce_example*

```
db.map_reduce_example.find().sort( { _id: 1 } )
```

- výsledky

```
{ "_id" : "Ant O. Knee", "value" : 95 }
{ "_id" : "Busby Bee", "value" : 125 }
{ "_id" : "Cam Elot", "value" : 60 }
{ "_id" : "Don Quis", "value" : 155 }
```



UKÁZKA

- celková útrata jednotlivých zákazníků
 - řešení pomocí agregační roury
 - nejsou potřeba vlastní funkce

```
db.orders.aggregate([
  { $group: { _id: "$cust_id", value: { $sum: "$price" } } },
  { $out: "agg_alternative_1" }
])
```

- fáze `$group` sdružuje data podle `cust_id` a počítá hodnotu pole `value` jako sumu příslušných hodnot `price`
- pole `value` obsahuje celkovou utracenou částku pro každé `cust_id`
- fáze `$out` zapisuje výstup do kolekce `agg_alternative_1`

UKÁZKA

- počet objednávek a množství pro jednotlivé druhy produktů
 - map-reduce operace nad kolekcí *orders*
 - pracuje s objednávkami novějšími než 1. 3. 2020
 - shlukuje objednávky na základě pole *items.sku*
 - počítá počet objednávek pro každé *sku*
 - počítá celkové objednané množství pro každé *sku*
 - počítá průměrné množství *sku* na objednávku
 - slučuje výsledky do výstupní kolekce
 - existující dokumenty s klíčem shodným s novými výsledky jsou přepsány
 - nové výsledky, jejichž klíč se v existujících dokumentech nevyskytuje, jsou vloženy

```
db.orders.insertMany([  
  { _id: 1, cust_id: "Ant O. Knee", ord_date: new Date("2020-03-01"), price: 25, items: [ { sku: "oranges", qty: 5, price: 2.5 }, { sku: "apples", qty: 5, price: 2.5 } ], status: "A" },  
  { _id: 2, cust_id: "Ant O. Knee", ord_date: new Date("2020-03-08"), price: 70, items: [ { sku: "oranges", qty: 8, price: 2.5 }, { sku: "chocolates", qty: 5, price: 10 } ], status: "A" },  
  { _id: 3, cust_id: "Busby Bee", ord_date: new Date("2020-03-08"), price: 50, items: [ { sku: "oranges", qty: 10, price: 2.5 }, { sku: "pears", qty: 10, price: 2.5 } ], status: "A" },  
  { _id: 4, cust_id: "Busby Bee", ord_date: new Date("2020-03-18"), price: 25, items: [ { sku: "oranges", qty: 10, price: 2.5 } ], status: "A" },  
  { _id: 5, cust_id: "Busby Bee", ord_date: new Date("2020-03-19"), price: 50, items: [ { sku: "chocolates", qty: 5, price: 10 } ], status: "A" },  
  { _id: 6, cust_id: "Cam Elot", ord_date: new Date("2020-03-19"), price: 35, items: [ { sku: "carrots", qty: 10, price: 1.0 }, { sku: "apples", qty: 10, price: 2.5 } ], status: "A" },  
  { _id: 7, cust_id: "Cam Elot", ord_date: new Date("2020-03-20"), price: 25, items: [ { sku: "oranges", qty: 10, price: 2.5 } ], status: "A" },  
  { _id: 8, cust_id: "Don Quis", ord_date: new Date("2020-03-20"), price: 75, items: [ { sku: "chocolates", qty: 5, price: 10 }, { sku: "apples", qty: 10, price: 2.5 } ], status: "A" },  
  { _id: 9, cust_id: "Don Quis", ord_date: new Date("2020-03-20"), price: 55, items: [ { sku: "carrots", qty: 5, price: 1.0 }, { sku: "apples", qty: 10, price: 2.5 }, { sku: "oranges", qty: 10, price: 2.5 } ], status: "A" },  
  { _id: 10, cust_id: "Don Quis", ord_date: new Date("2020-03-23"), price: 25, items: [ { sku: "oranges", qty: 10, price: 2.5 } ], status: "A" }])
```



UKÁZKA

- počet objednávek a množství pro jednotlivé druhy produktů
 - 1) definování *map* funkce, která zpracovává každý vstupní dokument

```
var mapFunction2 = function() {  
    for (var idx = 0; idx < this.items.length; idx++) {  
        var key = this.items[idx].sku;  
        var value = { count: 1, qty: this.items[idx].qty };  
  
        emit(key, value);  
    }  
};
```

- *this* odpovídá současně zpracovávanému dokumentu
- pro každý *item* funkce přidruží *sku* nový objekt *value*
 - *value* obsahuje *count 1* a *qty* odpovídající množství daného zboží v daném dokumentu
- vrací *sku* a *value* páry



UKÁZKA

- počet objednávek a množství pro jednotlivé druhy produktů
 - 2) definování odpovídající *reduce* funkce se dvěma parametry *keySKU* a *countObjVals*

```
var reduceFunction2 = function(keySKU, countObjVals) {  
    reducedVal = { count: 0, qty: 0 };  
  
    for (var idx = 0; idx < countObjVals.length; idx++) {  
        reducedVal.count += countObjVals[idx].count;  
        reducedVal.qty += countObjVals[idx].qty;  
    }  
  
    return reducedVal;  
};
```

- *countObjVals* je pole, jehož elementy jsou objekty mapované na sdružené hodnoty *keySKU* získané z *map* funkce
- funkce redukuje pole *countObjVals* na objekt *reducedVal*, který obsahuje pole *count* a *qty*
 - *count* je součtem jednotlivých hodnot *count*
 - *qty* je součtem jednotlivých hodnot *qty*



UKÁZKA

- počet objednávek a množství pro jednotlivé druhy produktů
 - 3) definování *finalize* funkce se dvěma parametry *key* a *reducedVal*

```
var finalizeFunction2 = function (key, reducedVal) {  
    reducedVal.avg = reducedVal.qty/reducedVal.count;  
    return reducedVal;  
};
```

- funkce modifikuje objekt *reducedVal*
- přidává a počítá pole *avg*
- vrací upravený objekt



UKÁZKA

- počet objednávek a množství pro jednotlivé druhy produktů
- 4) provedení map-reduce operace na všech dokumentech kolekce *orders*
 - zavoláním funkcí *mapFunction2*, *reduceFunction2* a *finalizeFunction2*

```
db.orders.mapReduce(  
    mapFunction2,  
    reduceFunction2,  
    {  
        out: { merge: "map_reduce_example2" },  
        query: { ord_date: { $gte: new Date("2020-03-01") } },  
        finalize: finalizeFunction2  
    }  
);
```

- operace využívá operaci *query* na výběr objednávek novějších než 1. 3. 2020
- výstup je uložen do kolekce *map_reduce_example2*
- pokud kolekce *map_reduce_example2* již existuje, dojde ke sloučení
 - pokud existující dokument má stejný klíč jako nový výsledek, dojde k jeho přepsání
 - pokud neexistuje dokument se stejným klíčem, dojde k jeho vložení





UKÁZKA

- počet objednávek a množství pro jednotlivé druhy produktů

5) ověření výsledků

- dotaz na kolekci *map_reduce_example2*

```
db.map_reduce_example2.find().sort( { _id: 1 } )
```

- výsledky

```
{ "_id" : "apples", "value" : { "count" : 3, "qty" : 30, "avg" : 10 } }
{ "_id" : "carrots", "value" : { "count" : 2, "qty" : 15, "avg" : 7.5 } }
{ "_id" : "chocolates", "value" : { "count" : 3, "qty" : 15, "avg" : 5 } }
{ "_id" : "oranges", "value" : { "count" : 6, "qty" : 58, "avg" : 9.666666666666666 }
{ "_id" : "pears", "value" : { "count" : 1, "qty" : 10, "avg" : 10 } }
```



UKÁZKA

- počet objednávek a množství pro jednotlivé druhy produktů
 - řešení pomocí agregační roury

```
db.orders.aggregate( [  
  { $match: { ord_date: { $gte: new Date("2020-03-01") } } },  
  { $unwind: "$items" },  
  { $group: { _id: "$items.sku", qty: { $sum: "$items.qty" }, orders_ids: { $addToSet: "$_id" } } },  
  { $project: { value: { count: { $size: "$orders_ids" }, qty: "$qty", avg: { $divide: [ "$qty", { $size: "$orders_ids" } ] } } } },  
  { $merge: { into: "agg_alternative_3", on: "_id", whenMatched: "replace", whenNotMatched: "insert" } }  
] )  
  
db.orders.insertMany([  
  { _id: 1, cust_id: "Ant O. Knee", ord_date: new Date("2020-03-01"), price: 25, items: [ { sku: "oranges", qty: 5, price: 2.5 }, { sku: "apples", qty: 5, price: 2.5 } ], status: "A" },  
  { _id: 2, cust_id: "Ant O. Knee", ord_date: new Date("2020-03-08"), price: 70, items: [ { sku: "oranges", qty: 8, price: 2.5 }, { sku: "chocolates", qty: 5, price: 10 } ], status: "A" },  

```



UKÁZKA

- ukázka použití map-reduce z manuálu MongoDB
<https://docs.mongodb.com/manual/tutorial/perform-incremental-map-reduce/>
- datový model
 - kolekce připojení uživatelů
 - uživatel, datum připojení a doba připojení

```
db.usersessions.insertMany([
    { userid: "a", start: ISODate('2020-03-03 14:17:00'), length: 95 },
    { userid: "b", start: ISODate('2020-03-03 14:23:00'), length: 110 },
    { userid: "c", start: ISODate('2020-03-03 15:02:00'), length: 120 },
    { userid: "d", start: ISODate('2020-03-03 16:45:00'), length: 45 },
    { userid: "a", start: ISODate('2020-03-04 11:05:00'), length: 105 },
    { userid: "b", start: ISODate('2020-03-04 13:14:00'), length: 120 },
    { userid: "c", start: ISODate('2020-03-04 17:00:00'), length: 130 },
    { userid: "d", start: ISODate('2020-03-04 15:37:00'), length: 65 }
])
```



UKÁZKA

- doba připojení uživatele a počet jeho připojení
- 1) definování *map* funkce, která zpracuje každý vstupní dokument

```
var mapFunction = function() {  
    var key = this.userid;  
    var value = { total_time: this.length, count: 1, avg_time: 0 };  
  
    emit( key, value );  
};
```

- mapuje *userid* na objekt *value*, který obsahuje pole *total_time*, *count* a *avg_time*
- vrací *userid* a *value* páry

UKÁZKA

- doba připojení uživatele a počet jeho připojení
- 2) definování odpovídající *reduce* funkce se dvěma parametry *key* a *values*
 - pro výpočet počtu a doby připojení

```
var reduceFunction = function(key, values) {  
  
    var reducedObject = { total_time: 0, count:0, avg_time:0 };  
  
    values.forEach(function(value) {  
        reducedObject.total_time += value.total_time;  
        reducedObject.count += value.count;  
    });  
  
    return reducedObject;  
};
```

- *key* odpovídá *userid*
- *values* je pole, jehož elementy odpovídají objektům mapovaným na *userid* v *map* fázi
- funkce redukuje *values* na objekt *reducedObject*, který obsahuje pole *total_time* a *count*
 - pole *total_time* je součtem jednotlivých polí *total_time*
 - pole *count* je součtem jednotlivých polí *count*



UKÁZKA

- doba připojení uživatele a počet jeho připojení
- 3) definování *finalize* funkce se dvěma parametry *key* a *reducedValue*

```
var finalizeFunction = function(key, reducedValue) {  
  
    if (reducedValue.count > 0)  
        reducedValue.avg_time = reducedValue.total_time / reducedValue.count;  
  
    return reducedValue;  
};
```

- funkce modifikuje objekt *reducedValue*
- počítá pole *avg_time*
- vrací upravený objekt

UKÁZKA

- doba připojení uživatele a počet jeho připojení
- 4) provedení map-reduce operace na dokumentech kolekce *usersessions*
 - zavoláním funkcí *mapFunction*, *reduceFunction* a *finalizeFunction*

```
db.usersessions.mapReduce(  
    mapFunction,  
    reduceFunction,  
    {  
        out: "session_stats",  
        finalize: finalizeFunction  
    }  
)
```

- výstup je uložen do kolekce *session_stats*
- jestli kolekce již existuje, její obsah je nahrazen



UKÁZKA

- doba připojení uživatele a počet jeho připojení

5) ověření výsledků

- dotaz na kolekci *session_stats*

```
db.session_stats.find().sort( { _id: 1 } )
```

- výsledky

```
{ "_id" : "a", "value" : { "total_time" : 200, "count" : 2, "avg_time" : 100 } }
{ "_id" : "b", "value" : { "total_time" : 230, "count" : 2, "avg_time" : 115 } }
{ "_id" : "c", "value" : { "total_time" : 250, "count" : 2, "avg_time" : 125 } }
{ "_id" : "d", "value" : { "total_time" : 110, "count" : 2, "avg_time" : 55 } }
```



UKÁZKA

- doba připojení uživatele a počet jeho připojení
 - řešení pomocí agregační roury

```
db.usersessions.aggregate([
  { $group: { _id: "$userid", total_time: { $sum: "$length" }, count: { $sum: 1 }, avg_time: { $avg: "$length" } } },
  { $project: { value: { total_time: "$total_time", count: "$count", avg_time: "$avg_time" } } },
  { $merge: {
    into: "session_stats_agg",
    whenMatched: [ { $set: {
      "value.total_time": { $add: [ "$value.total_time", "$$new.value.total_time" ] },
      "value.count": { $add: [ "$value.count", "$$new.value.count" ] },
      "value.avg": { $divide: [ { $add: [ "$value.total_time", "$$new.value.total_time" ] }, { $add: [ "$value.count", "$$new.value.count" ] } ] }
    } } ],
    whenNotMatched: "insert"
  }}
])
db.usersessions.insertMany([
  { userid: "a", start: ISODate('2020-03-03 14:17:00'), length: 95 },
  { userid: "b", start: ISODate('2020-03-03 14:23:00'), length: 110 },
  { userid: "c", start: ISODate('2020-03-03 15:02:00'), length: 120 },
  { userid: "d", start: ISODate('2020-03-03 16:45:00'), length: 45 },
  { userid: "a", start: ISODate('2020-03-04 11:05:00'), length: 105 },
  { userid: "b", start: ISODate('2020-03-04 13:14:00'), length: 120 },
  { userid: "c", start: ISODate('2020-03-04 17:00:00'), length: 130 },
  { userid: "d", start: ISODate('2020-03-04 15:37:00'), length: 65 }
])
```



ČÁST V: JEDNOÚČELOVÁ AGREGACE

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

UKÁZKA

- *db.collection.count()*
 - počet dokumentů v kolekci *orders* s polem *ord_dt* vyšším než 1. 1. 2012

```
db.orders.count( { ord_dt: { $gt: new Date('01/01/2012') } } )
```

- odpovídá

```
db.orders.find( { ord_dt: { $gt: new Date('01/01/2012') } } ).count()
```

<https://docs.mongodb.com/manual/aggregation/#single-purpose-aggregation-operations>

- bez filter dotazů využívá metadata
 - nemusí být vždy přesné



UKÁZKA

- *db.collection.estimatedDocumentCount()*
 - počet všech dokumentů v kolekci *orders*

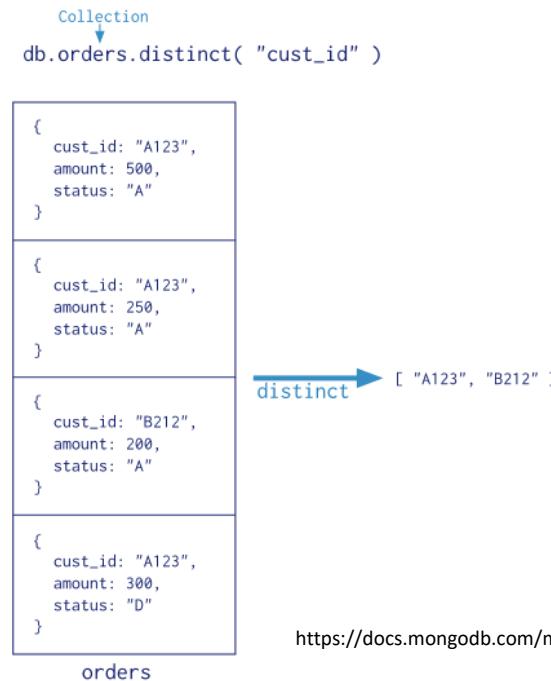
```
db.orders.estimatedDocumentCount({})
```

<https://docs.mongodb.com/manual/aggregation/#single-purpose-aggregation-operations>

- nepodporuje filter dotazy
- využívá metadata pro určení počtu dokumentů
 - nemusí být vždy přesné
 - orphaned dokumenty u shardovaných databází

UKÁZKA

- *db.collection.distinct()*
 - unikátní záznamy v poli *cust_id* v kolekci *orders*



<https://docs.mongodb.com/manual/aggregation/#single-purpose-aggregation-operations>





ČÁST VI.: MongoDB 5.0, 6.0 a 7.0

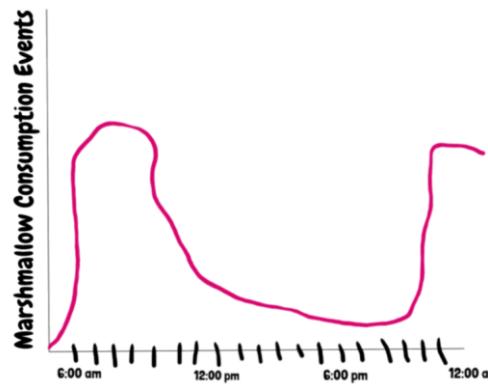


CO JE NOVÉHO?

- nová verze [5.0](#) vydaná 13. 7. 2021 přináší různá vylepšení
 - plná podpora časových řad
 - live resharding
 - kompatibilita do budoucna
 - pomocí verzovaného API
 - zaručení zpětné kompatibility od verze 5.0+
 - vylepšené šifrování
- informace ke [zpětné kompatibilitě](#)
 - výrazné problémy nejsou
 - map-reduce je nyní deprecated
 - mongosh místo mongo CLI

ČASOVÉ ŘADY

- v originále označovány jako
 - time series data nebo také time-stamped data
- soubor pozorování získaný opakovaným měřením v čase
 - při nanesení na graf je jednou z os vždy čas
 - přirozeně setříděný
 - samotné body nenesou důležité informace, ty pochází z celé časové řady
 - až přidání času přináší hlavní výhody



ČASOVÉ ŘADY A MongoDB

- vytvoření kolekce pro časové řady
 - nutné vědět již při vytváření
 - parametr timeseries
 - parametr timeField je vyžadovaný
 - obsahuje čas
 - zbylé parametry volitelné
- vložení měření
 - každý dokument obsahuje jedno měření
 - *insertOne()*
 - *insertMany()*

```
db.createCollection(  
  "weather",  
  {  
    timeseries: {  
      timeField: "timestamp",  
      metaField: "metadata",  
      granularity: "hours"  
    }  
  }  
)
```

```
db.weather.insertMany( [  
  {  
    "metadata": { "sensorId": 5578, "type": "temperature" },  
    "timestamp": ISODate("2021-05-18T00:00:00.000Z"),  
    "temp": 12  
  },  
  {  
    "metadata": { "sensorId": 5578, "type": "temperature" },  
    "timestamp": ISODate("2021-05-18T04:00:00.000Z"),  
    "temp": 11  
  }  
)
```





ČASOVÉ ŘADY A MongoDB

- dotazování

```
db.weather.findOne({  
    "timestamp": ISODate("2021-05-18T00:00:00.000Z")  
})
```

- agregace
 - co zjišťuje daná agregace?

```
db.weather.aggregate( [  
    {  
        $project: {  
            date: {  
                $dateToParts: { date: "$timestamp" }  
            },  
            temp: 1  
        }  
    },  
    {  
        $group: {  
            _id: {  
                date: {  
                    year: "$date.year",  
                    month: "$date.month",  
                    day: "$date.day"  
                }  
            },  
            avgTmp: { $avg: "$temp" }  
        }  
    }  
]
```





ČASOVÉ ŘADY A MongoDB

- agregace
 - co zjišťuje daná agregace?
 - průměrnou teplotu v daný den
- je kolekce kolekcí časových řad?

```
db.runCommand( { listCollections: 1.0 } )
```

- obecně se kolekce časových řad chovají jako každá jiná kolekce

```
{  
  "_id" : {  
    "date" : {  
      "year" : 2021,  
      "month" : 5,  
      "day" : 18  
    }  
  },  
  "avgTmp" : 12.714285714285714  
}  
{  
  "_id" : {  
    "date" : {  
      "year" : 2021,  
      "month" : 5,  
      "day" : 19  
    }  
  },  
  "avgTmp" : 13  
}
```



CO JE NOVÉHO?

- nová verze [6.0](#) vydaná 19. 7. 2022 přináší další [vylepšení](#)
 - rozšířená podpora časových řad
 - nové a vylepšené operátory
 - rozšířená tvorba událostmi řízených architektur
 - vylepšené shardování
 - vylepšená bezpečnost
 - cluster-to-cluster sync
- informace ke [zpětné kompatibilitě](#)
 - výrazné problémy nejsou
 - původní CLI mongo je deprecated
- verze [7.0](#) přináší další drobná vylepšení

A PŘÍŠTĚ?

- dokumentové databáze
 - databáze pro vyhledávání a analýzu textu





Děkuji za pozornost.
Otázky?

