Query Agregasi menggunakan pymongo

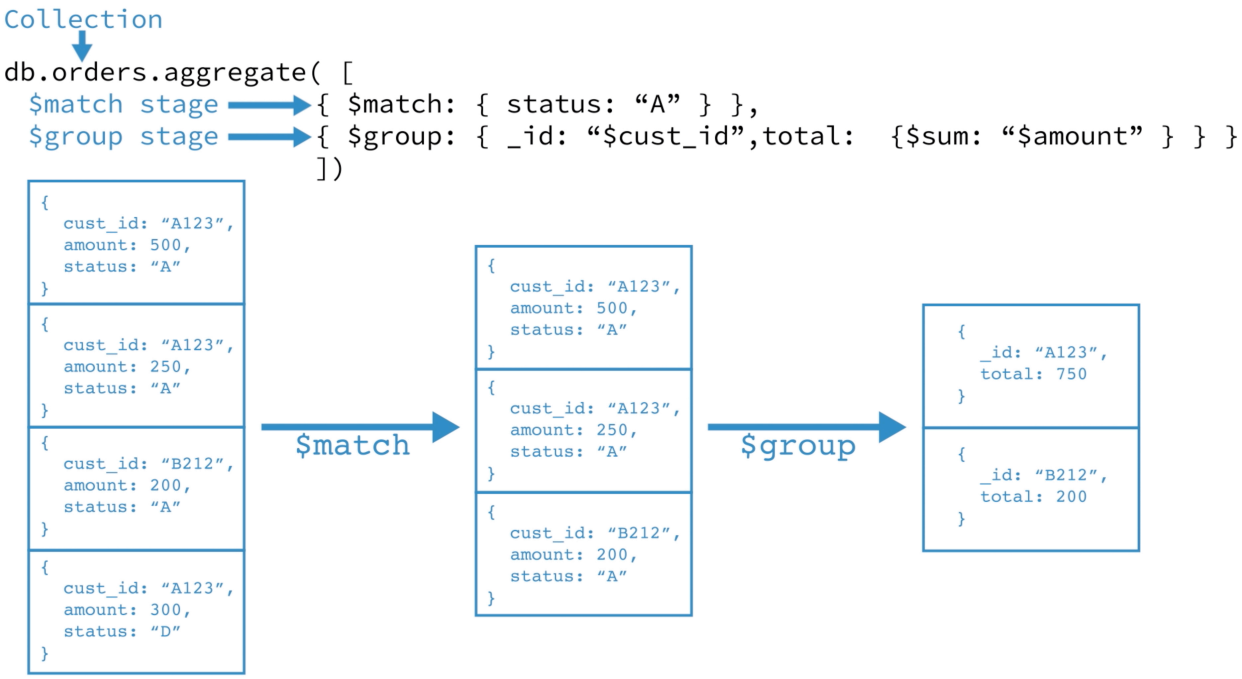
Operasi agregasi memproses catatan data dan mengembalikan hasil yang dihitung. Operasi agregat mengelompokkan nilai dari beberapa dokumen secara bersama, dan dapat melakukan berbagai operasi pada data yang dikelompokkan untuk mengembalikan hasil tunggal. MongoDB menyediakan tiga cara untuk melakukan agregasi: **the aggregation pipeline**, **the map-reduce function**, dan **single purpose aggregation methods**.

Dalam pembahasan ini kita hanya akan membahas dua cara agregasi yaitu: Aggregation Pipeline dan single purpose aggregation methods.

Aggregation Pipeline

Aggregation pipeline adalah framework untuk model data agregasi menggunakan konsep aliran pemrosesan data. Dokumen diproses melalu beberapa langkah pemrosesan sehingga data tersebut ditransformasi menjadi hasil agregasi. Sebagai contoh pada collection `orders` pada contoh berikut yaitu menghitung total *amount* untuk setiap *cust_id* yang memiliki $status =$" A ". perintah ini setara dengan berikut:

```
select cust_id, sum(amount) as total from orders where status = "A" group by cust_id
```



pada contoh ini, dokumen terlebih dahulu diproses pada stage `$match` yaitu kriteria data yang akan diproses selanjutnya masuk kedalam stage `$group` dimana data dikelompokkan berdasar *cust_id* selanjutnya dilakukan operasi akumulasi **sum** untuk menjumlahkan data field *amount* untuk setiap *cust_id* .

Selain stage `$match` dan `$group` masih ada beberapa stage lain yang bisa digunakan. Berikut ini diberikan stage-stage yang dapat digunakan pada agregasi.

- **$project** – digunakan untuk memilih field-field spesifik dari collection untuk ditampilkan pada hasil query.
- **$match** – operasi filter digunakan untuk mengurangi jumlah data yang akan diproses pada stage berikutnya.
- **$group** – proses agregasi.
- **$sort** – untuk mengurutkan dokumen pada output.
- **$skip** – dengan ini, dimungkinkan untuk melompat maju dalam daftar dokumen untuk jumlah dokumen tertentu.
- **$limit** – menentukan jumlah dokumen yang akan ditampilkan. berbeda dengan **$skip**, perintah ini menampilkan dari posisi awal ke posisi yang dinyatakan pada nilai **$limit**.
- **$unwind** – digunakan untuk melepas dokumen yang menggunakan array. Saat menggunakan sebuah array, data adalah jenis pra-bergabung dan operasi ini akan dibatalkan dengan ini untuk memiliki dokumen individual lagi. Maka dengan tahap ini kami akan menambah jumlah dokumen untuk tahap selanjutnya.

Tabel berikut ditunjukkan daftar operasi akumulasi pada stage **$group** yang dapat digunakan.

Name	Description
`$addToSet`	Returns an array of *unique* expression values for each group. Order of the array elements is undefined.
`$avg`	Returns an average of numerical values. Ignores non-numeric values.
`$first`	Returns a value from the first document for each group. Order is only defined if the documents are in a defined order.
`$last`	Returns a value from the last document for each group. Order is only defined if the documents are in a defined order.
`$max`	Returns the highest expression value for each group.
`$mergeObjects`	Returns a document created by combining the input documents for each group.
`$min`	Returns the lowest expression value for each group.
`$push`	Returns an array of expression values for each group.
`$stdDevPop`	Returns the population standard deviation of the input values.
`$stdDevSamp`	Returns the sample standard deviation of the input values.
`$sum`	Returns a sum of numerical values. Ignores non-numeric values.

Misalnya kita memiliki collection dengan nama **sales** dengan isi dokumen seperti yang ditunjukkan pada tabel berikut:

_id	item	price	quantity	date
1.0	abc	10	2	2014-03-01 08:00:00.000
2.0	jkl	20	1	2014-03-01 09:00:00.000
3.0	xyz	5	10	2014-03-15 09:00:00.000
4.0	xyz	5	20	2014-04-04 11:21:39.736
5.0	abc	10	10	2014-04-04 21:23:13.331
6.0	def	7.5	5	2015-06-04 05:08:13.000
7.0	def	7.5	10	2015-09-10 08:43:00.000
8.0	abc	10	5	2016-02-06 20:20:13.000

Jumlah dokumen dapat diperoleh dengan menggunakan perintah berikut:

```
hasil = db.sales.aggregate([{"$group":{"_id":None,"count":{"$sum": 1}}}])
DataFrame(hasil)
```

pada perintah ini, dokumen dikelompokkan berdasarkan field `_id` kemudian menjumlahkan angka 1 yang disimpan pada variable `count`. output perintah ini akan menghitung jumlah seluruh baris karena tidak ada nilai yang sama pada field `_id`.

Perintah untuk mendapatkan seluruh item yang berbeda pada collection tersebut dapat diperoleh menggunakan query berikut:

```
hasil = db.sales.aggregate( [ { "$group" : { "_id" : "$item" } } ] )
DataFrame(hasil)
```

perintah ini setara dengan sintak mysql berikut:

```
SELECT distict item as _id FROM sales
```

Sedangkan untuk menghitung jumlah setiap item digunakan perintah

```
hasil = db.sales.aggregate( [ { "$group" : { "_id" : "$item","count":{"$sum":1}} } ] )
DataFrame(hasil)
```

perintah ini setara dengan perintah mysql berikut:

```
SELECT item as _id, count(*) as `count` FROM sales group by item
```

*Perlu diingat bahwa setiap perintah query yang tidak melibatkan perintah `$project` harus menyertakan **_id** sebagai output. jika output dari perintah tersebut hanya satu field maka field tersebut harus diberi nama **_id** .*

Pada contoh berikut diberikan perhitungan *total sales amount* , rata-rata *sales quantity* dan jumlah penjualan setiap hari pada tahun 2014.

```
from datetime import datetime
```

```
hasil = db.sales.aggregate([
  {
    "$match" : { "date": { "$gte": datetime.strptime("2014-01-01",'%Y-%m-%d'), "$lt":
datetime.strptime("2015-01-01",'%Y-%m-%d') } }
  },
  {
    "$group" : {
      "_id" : { "$dateToString": { "format": "%Y-%m-%d", "date": "$date" } },
      "totalSaleAmount": { "$sum": { "$multiply": [ "$price", "$quantity" ] } },
      "averageQuantity": { "$avg": "$quantity" },
      "count": { "$sum": 1 }
    }
  },
  {
    "$sort" : { "totalSaleAmount": -1 }
  }
 ])
DataFrame(hasil)
```

Output:

	_id	totalSaleAmount	averageQuantity	count
0	2014-04-04	200	15.0	2
1	2014-03-15	50	10.0	1
2	2014-03-01	40	1.5	2

Perintah ini setara dengan perintah sql berikut:

```
SELECT date,
       Sum(( price * quantity )) AS totalSaleAmount,
       Avg(quantity)             AS averageQuantity,
       Count(*)                  AS Count
FROM   sales
GROUP  BY Date(date)
ORDER  BY totalSaleAmount DESC
```

Single purpose aggregation methods

Aggregation pipeline disediakan agar dapat membuat berbagai macam proses aggregasi dari suatu dokumen. Selain itu mongoDB juga menyediakan fungsi-fungsi tunggal yang digunakan untuk tujuan tertentu, fungsi-fungsi itu adalah:

- db.collection.estimatedDocumentCount()

  Mengembalikan hitungan semua dokumen dalam koleksi atau tampilan. Metode ini membungkus perintah hitungan. Fungsi ini tidak tersedia pada pymongo.

- db.collection.count(*query* , *options*)

Mengembalikan jumlah dokumen yang cocok dengan parameter *query* untuk pada suatu collection atau view. Metode db.collection.count () tidak melakukan operasi `find()` tetapi menghitung dan mengembalikan jumlah hasil yang cocok dengan *query* .

Parameter	Type	Description
`query`	document	The query selection criteria.
`options`	document	Optional. Extra options for modifying the count.

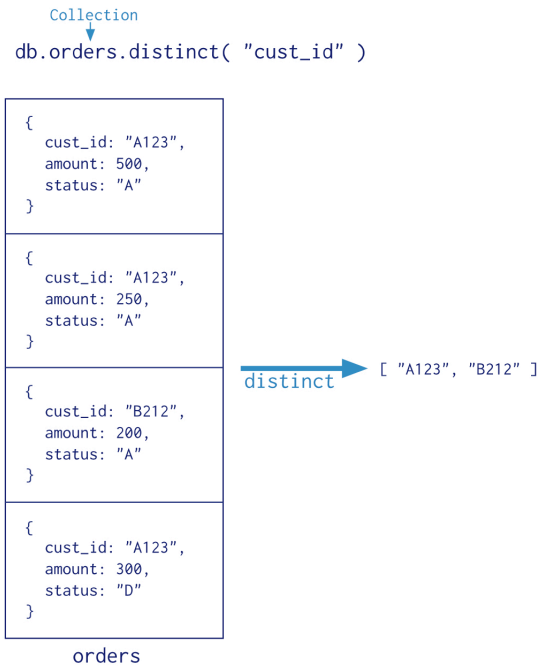
parameter `options` yang dapat digunakan adalah sebagai berikut:

Field	Type	Description
`limit`	integer	Optional. The maximum number of documents to count.
`skip`	integer	Optional. The number of documents to skip before counting.
`hint`	string or document	Optional. An index name hint or specification for the query.
`maxTimeMS`	integer	Optional. The maximum amount of time to allow the query to run.
`readConcern`	string	Optional. Specifies the **read concern** . The default level is **`"local"`** .To use **read concern** level of **`"majority"`** , replica sets must use **WiredTiger storage engine** .
`collation`	document	Optional.Specifies the **collation** to use for the operation. **Collation** allows users to specify language-specific rules for string comparison, such as rules for lettercase and accent marks.

`count()` ekuivalen dengan perintah `db.collection.find(query).count()` .

- db.collection.distinct(field, query, options)

  Menemukan nilai berbeda untuk field tertentu di satu collection atau view dan mengembalikan hasilnya dalam array.



Parameter	Type	Description
`field`	string	The field for which to return distinct values.
`query`	document	A query that specifies the documents from which to retrieve the distinct values.
`options`	document	Optional. A document that specifies the options.

Contoh:

```
db.sales.distinct( "item" )
```

Output:

```
['abc', 'def', 'jkl', 'xyz']
```

Contoh 1

Berapa banyak judul film yang dibintangi oleh masing-masing kategori pada database sakila.

```
from pymongo import MongoClient
from pandas import DataFrame
c = MongoClient()
db = c.sakila

film = db.Sakila_films.aggregate([
    {"$group":{"_id":"$Category","count":{"$sum":1}}}
])
DataFrame(film)
```

output:

	_id	count
0	Action	64
1	Drama	62
2	Sci-Fi	61
3	Children	60
4	Documentary	68
5	Horror	56
6	New	63
7	Comedy	58
8	Animation	66
9	Sports	74
10	Family	69
11	Foreign	73
12	Games	61
13	Classics	57
14	Music	51
15	Travel	57

Contoh 2

Berapa banyak file yang dibintangi oleh BELA WALKEN

```
db.Sakila_films.count_documents({"Actors.First name":"BELA","Actors.Last name":"WALKEN"})
```

Untuk versi pymongo terbaru fungsi `count` sudah tidak direkomendasikan digunakan. diganti dengan count_documents.

Contoh 3

Tuliskan jumlah sewa untuk masing-masing judul film dan urutkan secara descending

```
data = db.Sakila_customers.aggregate([
    {
    "$unwind":"$Rentals"
    },
    {
     "$group":{"_id":"$Rentals.Film Title","jumlah":{"$sum":1}}
    },
    {
        "$sort":{"jumlah":-1}
    }

])
DataFrame(data)
```

output:

	_id	jumlah
0	BUCKET BROTHERHOOD	34
1	ROCKETEER MOTHER	33
2	RIDGEMONT SUBMARINE	32
3	SCALAWAG DUCK	32
4	JUGGLER HARDLY	32
...	...	...
953	TRAFFIC HOBBIT	5
954	BRAVEHEART HUMAN	5
955	HARDLY ROBBERS	4
956	MIXED DOORS	4
957	TRAIN BUNCH	4

Tugas

Buat query untuk mendapat informasi berikut:

1. Jumlah judul film yang bercerita tentang “Crocodile” and a “Shark” pada dokumen Sakila_films.
2. Jumlah judul film untuk masing-masing **Rating** pada dokumen Sakila_films.
3. Jumlah aktor pada masing-masing judul film yang memiliki Rating **PG** pada dokumen Sakila_films.
4. Jumlah peminjaman oleh masing-masing customer yang tinggal di negara **United States** diurutkan secara descending. output berupa First Name, Last Name, Jumlah peminjaman.
5. Rata-rata uang yang digunakan untuk menyewa film yang dikelompokkan berdasarkan negara asal customer.
6. 10 Cusomer dengan jumlah uang yang terbanyak yang digunakan untuk menyewa Film diurutkan secara descanding.