MongoDB Aggregation 집계연산

집계 연산

aggregation pipeline

데이터 처리 파이프라인의 컨셉으로 다큐먼트가 다수의 스테이지로 구성된 파이프라인에 입력되어 최종적으로 다큐먼먼트를 생성하는 방식이다.

맵리듀스에 비해 디테일한 구성은 할 수 없지만 대부분의 처리가 가능하며 native 엔진(c++)을 사용하기 때문에 성능이 뛰어나다.

the map-reduce function

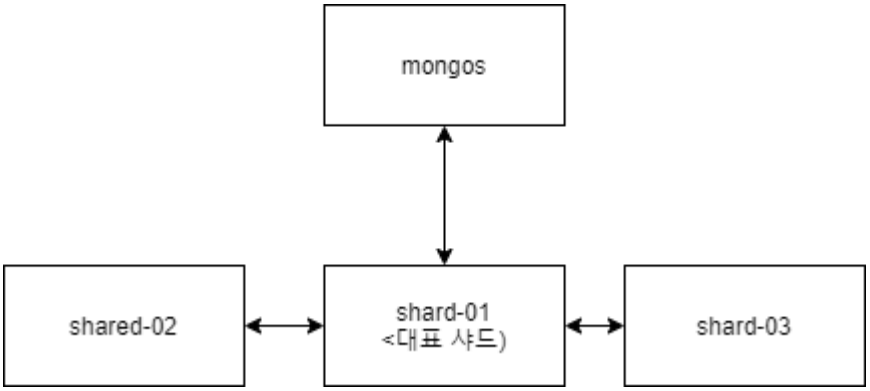
맵리듀스는 javascript 를 이용하여 사용자 정의 기능을 사용할 수 있지만 javascript 자체가 mongodb 에 내장된 native 엔진(c+++)과 바로 호환되지 않기 때문에 성능에 단점이 있다.

single purpose aggregation methods.

단일 collection 을 대상으로 간단한 aggregation 을 메소드로 제공하는 방식으로 기능자체도 단순하며 대상도 많지 않다.

Agrregation

작동방식



mongos 는 aggregation 요청을 받을 경우 $match 와 shard 키 등 조건을 확인한 후 일부 일반샤드 혹은 대표샤드에 전달한다.

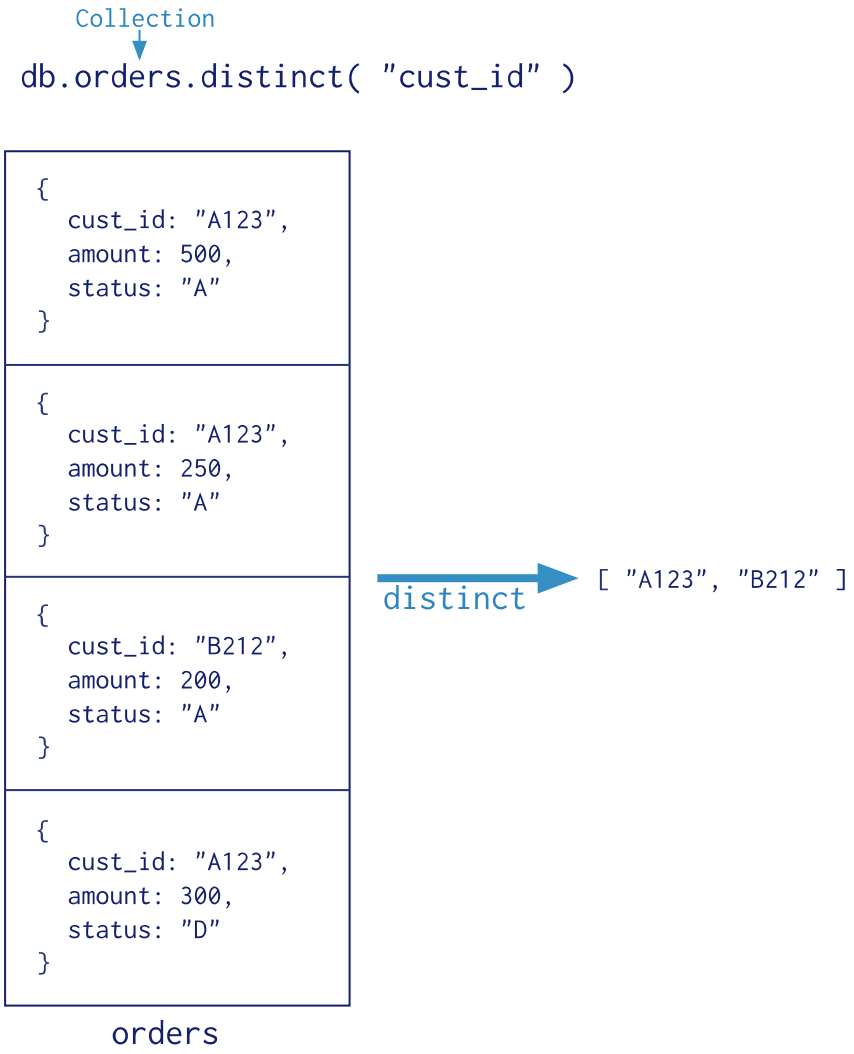
- 일반샤드는
  - aggregation 의 일부 결과를 대표샤드서버에 전달한다.
- 대표샤드는
  - 결과를 취합하여 mongos 로 전달 한다.
  - $sort, $group 등

대표샤드와 primary 샤드 차이

대표샤드는 하나의 aggregation 요청별 mongos 에서 선택하는 샤드이다. 의미상 primary 샤드와는 별개이다. 단 $out, $lookup, $graphLookup 은 기능 특성상 primary 샤드에서만 실행되기 때문에 pipeline 에 해당 stage 가 있다면 대표샤드는 primary 샤드가 된다.

> $out 에 명시된 collection 은 샤딩을 지원하지 않고 이러한 collection 은 primary 샤드에만 존재할 수 있다. $lookup, $graphLookup 에 서 드리븐되는 collection 은 unsharded collection 만 지원한다.

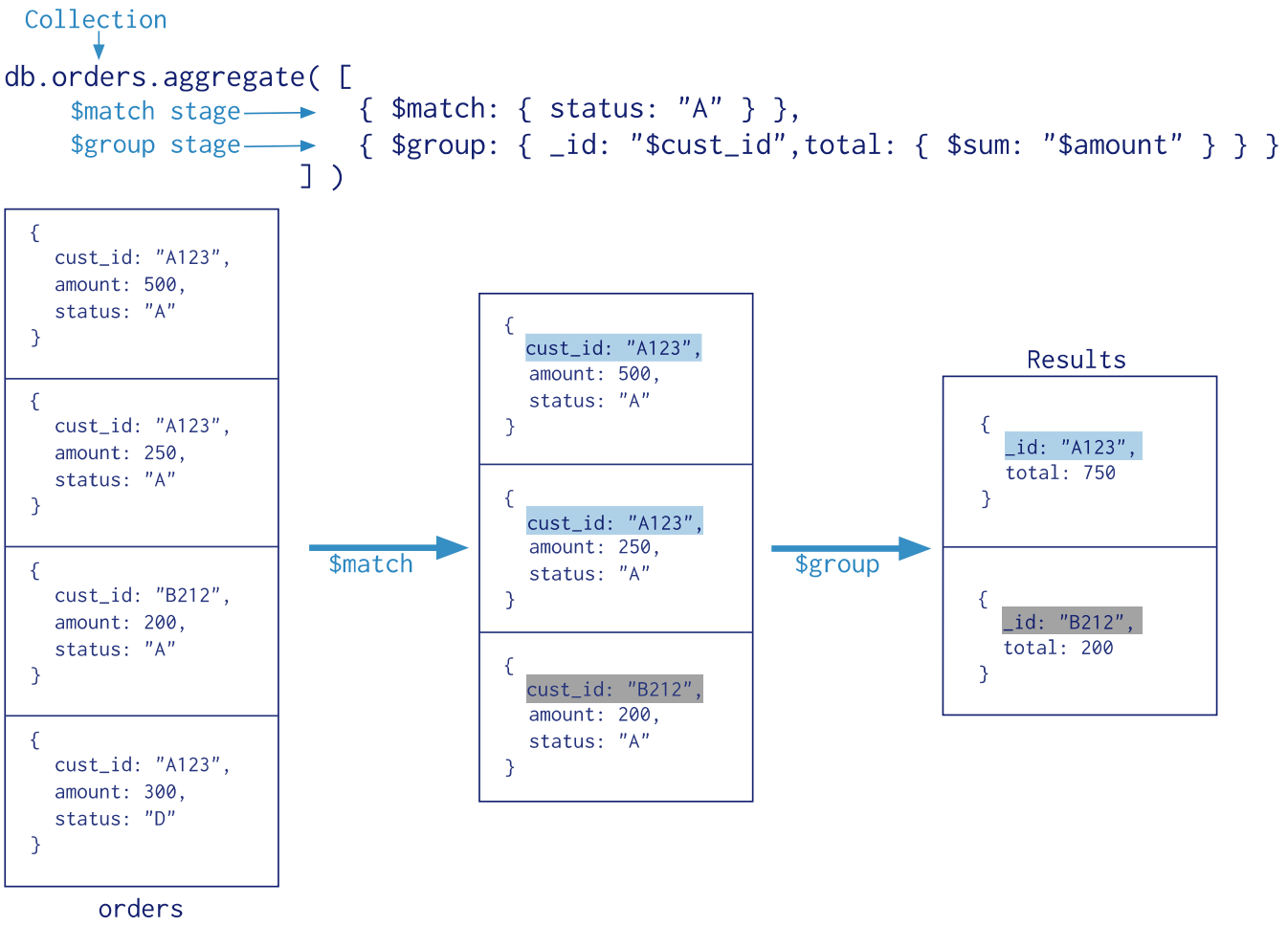
단일 목적의 Agrregation



aggregation pipeline 보다 단순하지만 자주 사용될 수 있는 기능을 method 형태로 지원한 것이다.

- db.collection.estimatedDocumentCount()
  - count() 를 랩핑한 함수로 estimated 시간을 지정할 수 있지만 query 를 제공하지 않는다.
- db.collection.count()
  - query 에 부합되는 다큐먼트의 합을 반환한다.
- db.collection.distinct()
  - query 에 부합되는 distinct 결과를 array 형태로 반환한다.

범용 Agrregation



단순한 목적이 아닌 사용자의 요청을 복수의 stage 를 이용하여 pipeline 처럼 동작하는 것을 말한다.

```
db.collection.aggregate( pipeline, options )
```

pipeline

대상 collection 데이터를 처리하는 stage 의 조합이며 위에서 부터 순차적으로 실행되어 다음 stage 에 전달 된다.

options

option	**desc**
allowDiskUse	각 stage 는 필요에 따라 메모리를 사용하지만 100mb 까지만 가능하며 초과할 경우 operation 이 실패된다. 이를 대비해 디스크 사용 여부를 설정할 수 있다.
cursor	결과로 반환되는 커서의 배치 사이즈 조정
maxTimeMS	실행될 수 있는 최대 시간
readConcern	readConcern , default local
bypassDocumentValidation	$out 등 결과를 다른 collection 에 저장할 때 유효성 체크에 대한 설정

option	desc
collation	collation 설정

파이프라인 스테이지와 표현식

pipeline 구성은 다음과 같다.

- stages
  - https://docs.mongodb.com/manual/meta/aggregation-quick-reference/#stages
- expression
  - https://docs.mongodb.com/manual/meta/aggregation-quick-reference/#expressions
- operator expressions
  - https://docs.mongodb.com/manual/meta/aggregation-quick-reference/#operator-expressions

stages

aggregation 의 pipeline 에서 사용되는 데이터 처리의 단위로 array 형태로 기술되며 순차적으로 실행된다. $out, $merge, $geoNear 를 제외하고는 복수로 사용될 수 있다.

대표적인 stage

stage	desc
$project	필요한 필드만 선택하거나 새로운 필드를 만들 수 있다.
$match	다큐먼트를 조건에 맞게 필터링 한다.
$unwind	array 구성된 필드가 있다면 풀어서 재구성한다.
$group	그룹핑을 수행한다.
$sample	임의의 다큐먼트를 샘플링 한다.
$addFields	새로운 필드를 추가 한다.
$replaceRoot	특정 필드를 최상위 다큐먼트로 만든다.
$count	다큐먼트의 개수를 구한다.

expression

field paths

입력된 다큐먼트의 필드를 접근하는 방식이다. 일반적으로 "$"를 붙인다.

```
{ user: "chlee", address: {city: "seoul"} }
$user
$address.city
```

system variables

"$$" 로 시작하는 시스템 변수, BSON data 를 사용자 변수로 지정해서 사용할 수 있다.

literal, expression objects

일반적으로 표현식에서 $ 로 시작하는 경우 field path 로 인식해 해석하려 하지만 $literal 표현식을 사용하는 경우 $로 시작해도 문자열로 처리하게 된다.

```
// $price 필드의 값과 $1 필드의 값을 비교한다.
$eq: [ "$price", "$1" ]
// $price 필드의 값이 $1 인지 비교한다.
$eq: [ "$price", { $literal: "$1" } ]
```

expression operators

아래 참조

expression operators

aggregation pipeline 에서는 연산을 위해 표현식 연산자를 제공하며 자세한 것은 메뉴얼을 참고한다.

파이프라인 최적화

스테이지 순서 최적화

```
// $match 에서 필요한 maxtime, minTime, avgTime 을
// $addFields , $project 에서 정의하고 있다.
{ $addFields: {
    maxTime: { $max: "$times" },
    minTime: { $min: "$times" }
} },
{ $project: {
    _id: 1, name: 1, times: 1, maxTime: 1, minTime: 1,
    avgTime: { $avg: ["$maxTime", "$minTime"] }
} },
{ $match: {
    name: "Joe Schmoe",
    maxTime: { $lt: 20 },
    minTime: { $gt: 5 },
    avgTime: { $gt: 7 }
} }

// name: "Joe Schmoe" 를 선행 처리해서 $addFields 의 처리량을 줄일 수 있다.
{ $match: { name: "Joe Schmoe" } },
{ $addFields: {
    maxTime: { $max: "$times" },
    minTime: { $min: "$times" }
} },
// $addFields 에서 maxTime, minTime 가 추가되어
// $match 를 통해 $project 의 처리량을 줄 일 수 있다.
{ $match: { maxTime: { $lt: 20 }, minTime: { $gt: 5 } } },
```

```
{ $project: {
    _id: 1, name: 1, times: 1, maxTime: 1, minTime: 1,
    avgTime: { $avg: ["$maxTime", "$minTime"] }
} },
{ $match: { avgTime: { $gt: 7 } } }
```

```
{ $sort: { age : -1 } },
{ $match: { status: 'A' } }

// $match 로 $sort 의 처리량을 줄 일 수 있다.
{ $match: { status: 'A' } },
{ $sort: { age : -1 } }
```

스테이지 결합 최적화

```
{ $limit: 100 },
{ $limit: 10 }

// $limist : 100 은 의미 없다.
{ $limit: 10 }
```

```
{ $skip: 5 },
{ $skip: 2 }

// 결합하여 7 로 기술한다.
{ $skip: 7 }
```

인덱스 사용

pipeline 의 각 stage 는 collection 의 index 를 사용할 수 있다. 하지만 상위 stage 에서 데이터의 변형이 있을 경우에는 index 를 사용할 수 없다. 일반적은 RDBMS 의 index 사용과 유사함.

메모리 사용

aggregation pipeline 의 각 stage 는 경우에 따라 메모리를 사용하지만 100mb 를 넘을 수 없다. 넘을 경우에는 operation 이 실패하기 때문에 이런 경우 allowDiskUse 옵션을 활성화해야 한다.