7. MongoDB Aggregation 집계연산

. 기존의 find로는 원하는 데이터로 가공하는데 어려움

. 빅데이터를 다루려면 새로운 데이터 가공 방식이 필요

. mongodb aggregation을 사용하면 documents를 grouping, filtering 등 다양한 연산을 적용할 수 있음

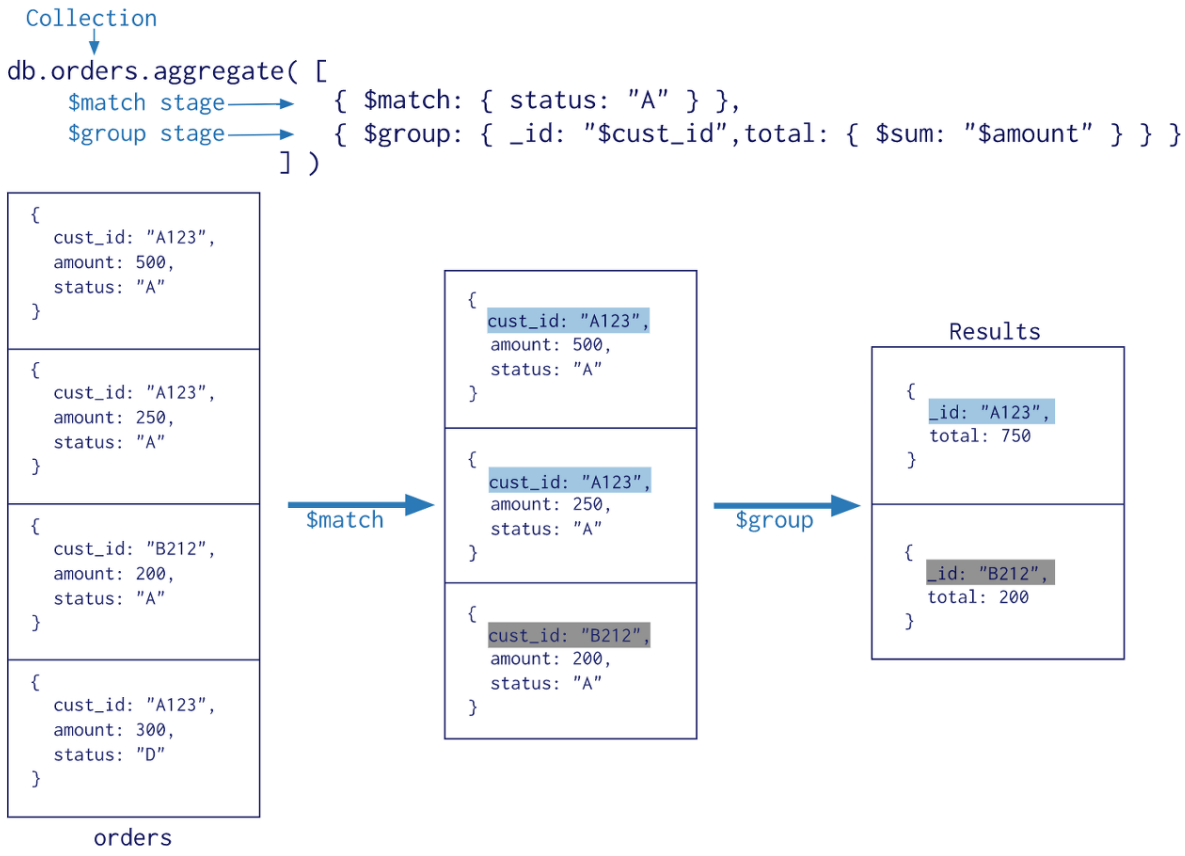
. MongoDB 2.4 버전까지의 aggregate( ) 명령은 하나의 단일 도큐먼트를 반환, aggreage( ) 명령의 결과는 16MB를 초과불가

. MongoDB 2.6 버전부터는 aggreage( )의 결과가 커서로 반환되도록 개선

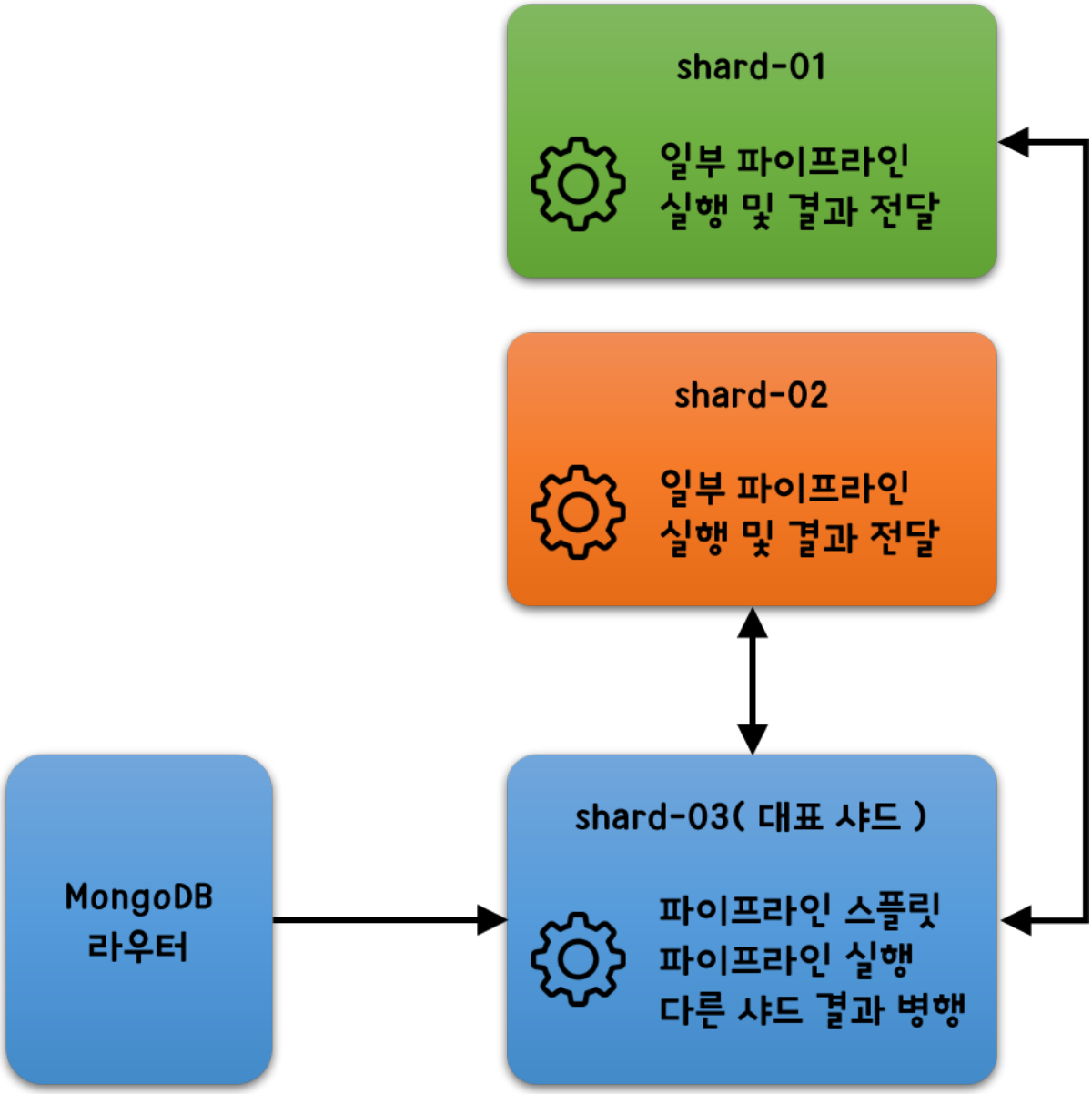
. 16MB 용량 제한이 없어졌을 뿐만 아니라 Aggregation( ) 명령에 커서 옵션을 설정할 수 있게 지원

작동 방식

. aggregate는 일종의 리눅스와 비슷한 pipeline을 가지면서 각 단계별로 진행을 마친 후에 result을 가져오게 된다. 예를 들어, $project, $match, $group, $sort 가 있을 때, 아래와 같은 순서로 pipeline을 진행한 후에 결과값을 가져오게 된다.



collection > $project > $match > $group > $sort> $skip> $limit > $unwind> $out



. MongoDB 라우터(Mongos)는 사용자로부터 Aggregation 쿼리를 전달받으면 우선 요청된 Aggregation 쿼리의 파이프라인 선두에 "$match" 스테이지가 있는지와 $match 스테이지의 검색 조건이 샤드 키를 포함하는지를 비교

. 만약 검색 조건이 필요로 하는 도큐먼트가 단일 샤드에만 있다면 MongoDB 라우터는 해당 샤드로만 Aggregation 쿼리를 전송

. 만약 그렇지 않다면 MongoDB 라우터는 Aggregation 쿼리를 대표 샤드로 전달

. 대표 샤드는 Aggregation 쿼리를 전달받으면 필요한 나머지 샤드로 쿼리(Aggregation 파이프라인에서 필요한 일부 스테이지만)를 전송

. 요청을 전달받은 샤드들이 쿼리 결과를 반환하면 대표 샤드는 그 결과를 병합하고 정렬하는 작업을 수행해서 MongoDB라우터로 최종 결과를 전달

단일 목적의 Aggregation

. SQL과 비교한 Aggregation 주요 명령

SQL Terms, Functions, and Concepts	MongoDB Aggregation Operators
WHERE	`$match`
GROUP BY	`$group`
HAVING	`$match`
SELECT	`$project`
ORDER BY	`$sort`
LIMIT	`$limit`
SUM()	`$sum`
COUNT()	`$sum`
join	`$lookup`
	New in version 3.2.

. $sum: 개수를 count 하거나 value을 더해준다.

. $avg: value의 평균값을 구해준다.

. $min: 최소값을 구하기

. $max: 최대값을 구하기

. $push: array값을 result document에 더해준다.

. $addtoset: 특정값을 result document에 더해준다. (위 두 expression은 본 collection에 영향을 주지 않는다)

. $first ($last): sort을 할 때 첫번째 값이나 마지막값을 구하기 위함

범용 Aggregation

```
mongo> db.collection.aggregate( pipeline, options );
```

. MongoDB의 Aggregation 프레임워크가 인식할 수 있도록 미리 정의된 스테이지로만 구성

. 두번째 파라미터는 다음과 같이 다양한 설정을 가질 수 있다.

Aa 옵션	설명
explain	Aggregation 명령의 실행 계획을확인할 수 있는 옵션이다. 기본값은 false이며, Aggregate( ) 명령이 어떤 실행 계획으로 실행될지 확인하고자 한다면 explain 필드를 true로 설정하면된다.
allowDiskUse	MongoDB의 Aggregate( ) 명령은 기본적으로 정렬을 위해서 100MB의 메모리까지 사용할 수 있다. 하지만 100MB 이상의 데이터를 정렬해야 하는 경우라면 Aggregate( ) 명령은 실패하게 된다. 이런 경우에는 allowDiskUse 옵션을 true로 설정해서 Aggregate( ) 처리가 디스크를 이용해서 정렬을 처리할 수 있게 한다. allowDiskUse 옵션을 true로 설정하면 MongoDB 서버는 MongoDB의 데이터 디렉터리 하위에 "_tmp"라는 디렉터리를 만들어서 임시 가공용 데이터 파일을 저장한다.
cusor	Aggregate( ) 명령의 결과로 반환되는 커서의 배치 사이즈를 설정할 수 있다.
maxTimeMS	Aggregate( ) 명령이 실행될 최대 시간을 설정한다.
readConcern	Aggregate( ) 명령이 도큐먼트의 개수를 확인할 때, 사용할 readConcern 옵션을 설정한다. 아무런 readConcern 옵션도 설정하지 않으면 "local" readConcern이 사용된다.
bypassDocumentValidation	Aggregate( ) 명령의 결과를 다른 컬렉션으로 저장하는 경우에 저장되는 컬렉션의 도큐먼트 유효성체크를 무시할 것인지 설정할 수 있다.
collation	Aggregate( ) 명령이 필요한 도큐먼트를 검색하 쿼리에서 사용할 콜레이션을 설정할 수 있다.

파이프라인 스테이지와 표현식

. 파이프라인(pipeline) 이란, 이전 단계의 연산결과를 다음 단계연산에 이용하는 것을 의미

. 파이프라인은 stage의 연속 묶음, aggregate구문은 순차적으로 stage 구성

. db.collection.aggregate 메서드는 파이프라인 단계를 Array의 형태 도출

. Document는 파이프라인 Array의 순서대로 가공되며,$out 및 $geoNear 를 제외한 모든 단계는 파이프라인에 여러 번 나타날 수 있음

파이프라인 최적화

. 스테이지 별 연산을 순차적으로 수행하기 때문에 순서를 잘 정하여 연산하는 것이 중요

. aggregate에 있어서 가장 기본적인 필터링은 $match

. 각 document 내부 항목들에 대한 필터링을 거쳐서 데이터베이스 조회 결과물들을 한 번 추림(Filtering)

. 보통 통계를 위한 데이터 처리에 있어서 필요한 것은 같은 분류의 데이터끼리 합치는 것($group)(Grouping)

. 새로운 document를 생성하는만큼, 고유한 _id를 필수적으로 지정

. $facet를 통해 특정 stage 내에서 여러 개의 연산자를 처리(MongoDB 3.4버젼에서 추가)

. 특정 쿼리에 도움이 되는 index를 사전에 생성함으로 인해 query 실행 시 기존보다 훨씬 빠른 순회(Indexing)

출처

https://secretartbook.tistory.com/22?category=433672

https://gonewbie.github.io/2019/12/16/how-to-optimize-mongodb-query-aggregate/https://junho94.tistory.com/11