11. Aggregation Framework

1. MongoDB 빅데이터 추출 방법
2. Aggregation Framework 개념
3. Aggregation Framework 사용
4. MapReduce vs Aggregation

1. MongoDB 빅데이터 추출 방법

- **빅데이터 추출 방법**
  - Aggregation Framework 함수를 이용한 데이터 추출
  - MongoDB의 MapReduce 기능을 이용한 데이터 추출
  - MongoDB와 Hadoop의 Map-Reduce를 연동한 빅 데이터의 추출

mongoDB

2. Aggregation Framework 개념

- **Aggregation Framework 함수**
  - Aggregation Framework는 데이터 추출에 최적화 되어 만들어진 기능 (v 2.1 지원)
  - Aggregation은 실행하면 내부적으로 MongoDB의 Map/Reduce를 사용하며 빠른 성능을 보장.
  - 실시간 Aggregation은 SQL의 Group 함수와 유사.
  - MongoDB Map/Reduce를 이용한 JavaScript로 생성
  - JavaScript는 외부 데이터 처리에 제한적이며 MongoDB내의 데이터 만 처리 가능
  - Aggregation Framework 함수
    - $project, $match, $group. $sort, $skip 6개의 연결 연산자로 구성
    - 관계형 데이터베이스의 SELECT, WHERE, GROUP BY, ORDER BY 등과 같은 형태로 문장 작성 가능

1. Aggregation Framework 개념

- 분석 대상 데이터

```
> db.yield_historical.in.find().pretty()
{
  "_id" : ISODate("1990-01-02T00:00:00Z"),
  "dayOfWeek" : "TUESDAY",
  "bc3Year" : 7.9,
  "bc5Year" : 7.87,
  "bc10Year" : 7.94, ← 집계 대상 Field
  "bc20Year" : null,
  "bc1Month" : null,
```

```
{
  "_id" : ISODate("1997-01-03T00:00:00Z"),
  "dayOfWeek" : "WEDNESDAY",
  "bc3Year" : 7.96,
  "bc5Year" : 7.92,
  "bc10Year" : 7.99,        ← 집계 대상 Field
  "bc20Year" : null,
  "bc1Month" : null,
  "bc2Year" : 7.94,
  "bc3Month" : 7.89,
  "bc30Year" : 8.04,
  "bc1Year" : 7.85,
  "bc7Year" : 8.04,
  "bc6Month" : 7.94
```

```
> db.yield_historical.out.find()
{ "_id" : 1990, "value" : 8.552400000000002 }
{ "_id" : 1991, "value" : 7.8623600000000025 }
{ "_id" : 1992, "value" : 7.008844621513946 }
{ "_id" : 1993, "value" : 5.866279999999999 }
{ "_id" : 1994, "value" : 7.085180722891565 }
{ "_id" : 1995, "value" : 6.573920000000002 }
{ "_id" : 1996, "value" : 6.443531746031742 }
{ "_id" : 1997, "value" : 6.353959999999992 }
{ "_id" : 1998, "value" : 5.262879999999994 }
{ "_id" : 1999, "value" : 5.646135458167332 }
.....

.....
```

1. Aggregation Framework 개념

Sample

```
db.yield_histor
{ $project :
  { _id  : 1,   bc
{ $match :
  {bc10Year : { $gte : 0.00, $lte : 9.99 } },
{ $group :
  { _id   :  { $year : "$_id" },
    value :  { $avg : "$bc10Year" } }  } ,
{ $sort :
  {  _id : 1  } } )
```

```
SELECT      year, avg(bc10Year)
FROM        yield_historical.in
WHERE       bc10Year >= 0.00
             and bc10Year <= 9.99
GROUP BY    year
ORDER BY    id asc;
```

1. Aggregation Framework 개념

- **Aggregation Framework** 함수를 이용한 데이터 추출
  - SQL과 매칭

```
$project->SELECT 절
$match->WHERE 절  or HAVING
$group->GROUP BY 절
$sort-> ORDER BY 절
```

mongoDB

2. Aggregation Framework 사용

```
db.order.drop()
db.order.insert({cust_id:"A2012001",
        order_date:new Date("Oct 01,2012"),
        status: "A",
        price:250,
        items:[{itme_name:"Bunny Boot", gty: 5, price:2.5},
                {itme_name:"Sky Ploe", gty: 5, price:2.5}]})

db.order.insert({cust_id:"A2012001",
        order_date:new Date("Sep 15, 2012"),
        status: "A",
        price:1125,
        items:[{itme_name:"Bunny Boot", gty: 15, price:2.5},
                {itme_name:"Sky Ploe", gty: 5, price:2.5}]})

db.order.insert({cust_id:"A2012001",
        order_date:new Date("Nov 10m 2012"),
        status: "A",
        price:200,
        items:[{itme_name:"Bunny Boot", gty: 10, price:2.5},
                {itme_name:"Sky Ploe", gty: 10, price:2.5}]})
```

2. Aggregation Framework 사용

SQL	MongoDB 스키마 문법
SELECT COUNT(*) AS COUNT FROM order;	db.order.aggregate([{$group : {_id : null, count : { $sum :1 }}}])
SELECT SUM(price) as total_price FROM order	db.order.aggregate([{$group:{_id:null, total_price:{$sum:"$price"}}}])
SELECT SUM(price) as total_price FROM order GROUP BY cust_id	db.order.aggregate([{$group:{_id:"$cust_id", total_price:{$sum:"$price"}}}])
SELECT SUM(price) as total_price FROM order GROUP BY cust_id ORDER BY price;	db.order.aggregate([{$group:{_id:"$cust_id", total_price:{$sum:"$price"}}}, {$sort:{total_price :1}}])
SELECT cust_id, ord_date, SUM(price) as total_price FROM order GROUP BY cust_id, ord_date;	db.order.aggregate([{$group:{_id:{cust_id:"$cust_id", ord_date:"$ord_date"}, total_price:{$sum:"$price"}}}])
SELECT cust_id, count(*) from order GROUP BY cust_id HAVING count(*)>1;	db.order.aggregate([{$group:{_id:"$cust_id", count:{$sum:1}}}, {$match : {count :{$gt :1}}}])

2. Aggregation Framework 사용

- **employee aggregate 실습**

```
db.employees.aggregate(
{$match: {$and:[{deptno:10},{sal: {$gte :500, $lte:3000 }}]}},
{$project : {
            _id :0,
            empno:1,
            ename :{$toLower :"$ename"},
            job:{$toUpper:"$job"},
            substr_name : {$substr :["$ename", 1, 2]},
            str_compare:{$strcasecmp : ["$ename","JMJOO"]},
            sal :1,
            deptno:1
}}).pretty()
```

2. Aggregation Framework 사용

```
db.employees.aggregate(
{$match : {deptno:30}},
{$project:{_id:0,
            empno:1,
            stats:{sal :"$sal", comm:"$comm"}
}}).pretty()
```

mongoDB

2. Aggregation Framework 사용

```
db.employees.aggregate(
{$match : {deptno:30}},
{$project:{
            _id:0,
            empno:1,
            ename:1,
            sal:1,
            comm: {$ifNull:["$comm",0]},
            sum_avg_add: {$add :["$sal", {$ifNull:["$comm",0]}]},
            sum_avg_substract: {$subtract : ["$sal",{$ifNull:["$comm",0]}]},
            sum_avg_multiply : {multiply :["$sal", 3]},
            sum_avg_divide : {$divide : ["$sal",2]}
}}).pretty()
```

mongoDB

2. Aggregation Framework 사용

```
db.employees.aggregate(
{$match : {deptno:30}},
{$project:{
                _id:0,
                empno:1,
                ename:1,
                sal:1,
                comm: {$ifNull:["$comm",0]},
                sum_avg_add: {$add :["$sal", {$ifNull:["$comm",0]}]},
                sum_avg_substract: {$subtract : ["$sal",{$ifNull:["$comm",0]}]},
                sum_avg_multiply : {$multiply :["$sal", 3]},
                sum_avg_divide : {$divide : ["$sal",2]}
}},
{$group:
                {_id:0,
                numTotalAmount:{$sum :1},
                addTotalAmount:{$sum:"$sal"},
                avgTotalAmount:{$avg:"$sal"},
                maxAmount:{$max:"$sal"},
                minAmount:{$min:"$sal"},
                firstempno:{$first:"$empno"},
                lastempno :{$last:"$empno"},
                addAmount: {$sum:"$sum_avg_add"},
                subtractAmount:{$sum:"$sum_avg_substract"},
                multiplyAmount: {$sum:"$sum_avg_multiply"},
                divideAmount: {$sum: "$sum_avg_divide"}
}}).pretty()
```

mongoDB

2. Aggregation Framework 사용

```
db.employees.aggregate(
{$group: {_id:"$deptno",
          numTotalAmount:{$sum :1},
          addTotalAmount:{$sum:"$sal"},
          avgTotalAmount:{$avg:"$sal"},
          maxAmount:{$max:"$sal"},
          minAmount:{$min:"$sal"},
          firstempno:{$first:"$empno"},
          lastempno :{$last:"$empno"}
}}).pretty()
```

```
db.employees.aggregate(
{$group:{_id : "$deptno",
          enames :{$addToSet : "$ename"}}
}).pretty()
```

mongoDB

2. Aggregation Framework 사용

```
db.employees.aggregate(
{$match :{deptno : 30}},
{$sort :{empno:1}},
{$project:{_id:0, empno:1, ename:1, deptno:1}}
).pretty()
```

```
db.employees.aggregate(
{$match:{deptno: 30}},
{$project : {
            _id:0,
            empno:1,
            ename:1,
            job:1,
            comm:1,
            condition : {$cond :["$comm", 1,0]},
}}).pretty()
```

mongoDB

2. Aggregation Framework 사용

```
db.employees.aggregate(
{$match:{deptno:30}},
{$project: {
            _id:0,
            empno:1,
            ename:1,
            job:1,
            comm:1,
            boolean_condition :{$cond : ["$comm", 1, 0]},
            ifNumm_condition: {$ifNull:["$comm",0]}
}}).pretty()
```

mongoDB

2. Aggregation Framework 사용

```
db.employees.aggregate(
{$match: {empno:7782}},
{$project:{
        _id:0,
        empno:1,
        ename:1,
        year_hiredate:{$year : "$hiredate"},
        month_hiredate:{$month : "$hiredate"},
        week_hiredate:{$week : "$hiredate"},
        hour_hiredate:{$hour : "$hiredate"},
        minute_hiredate:{$minute : "$hiredate"},
        second_hiredate:{$second : "$hiredate"},
        dayOfYear_hiredate:{$dayOfYear : "$hiredate"},
        dayOfMonth_hiredate:{$dayOfMonth : "$hiredate"},
        dayOfWeek_hiredate:{$dayOfWeek : "$hiredate"}
}}).pretty()
```

mongoDB

3. Aggregation vs Mapreduce

Aggregation Methods 비교

구분	Aggregation	MapReduce
설명	V2.2 부터 지원, 빠른 성능과 효율적인 사용 가능하며pipeline 연산 제공, pipeline 연산은 필요한 만큼 반복적으로 사용	Large Data Sets의 빠른 처리를 위해 map과 reduce 함수 제공. 복잡한 추출 작업을 위해 여러 개의 grouping 함수를 실행 해야함.
결과 출력	BSON 타입의 inline Result 결과로 출력되며, 크기가 제한됨	다양한 Result(inline, New Collection, merge 등) 결과로 출력
Sharding	shared and Non Sharded input Collection 지원	Shared and Non Sharded input Collection 지원
기타		V,2.4 이전 버전에서 Java Script 코드는 Single Thread로 실행

mongoDB