

# MongoDB

# 0.start

---

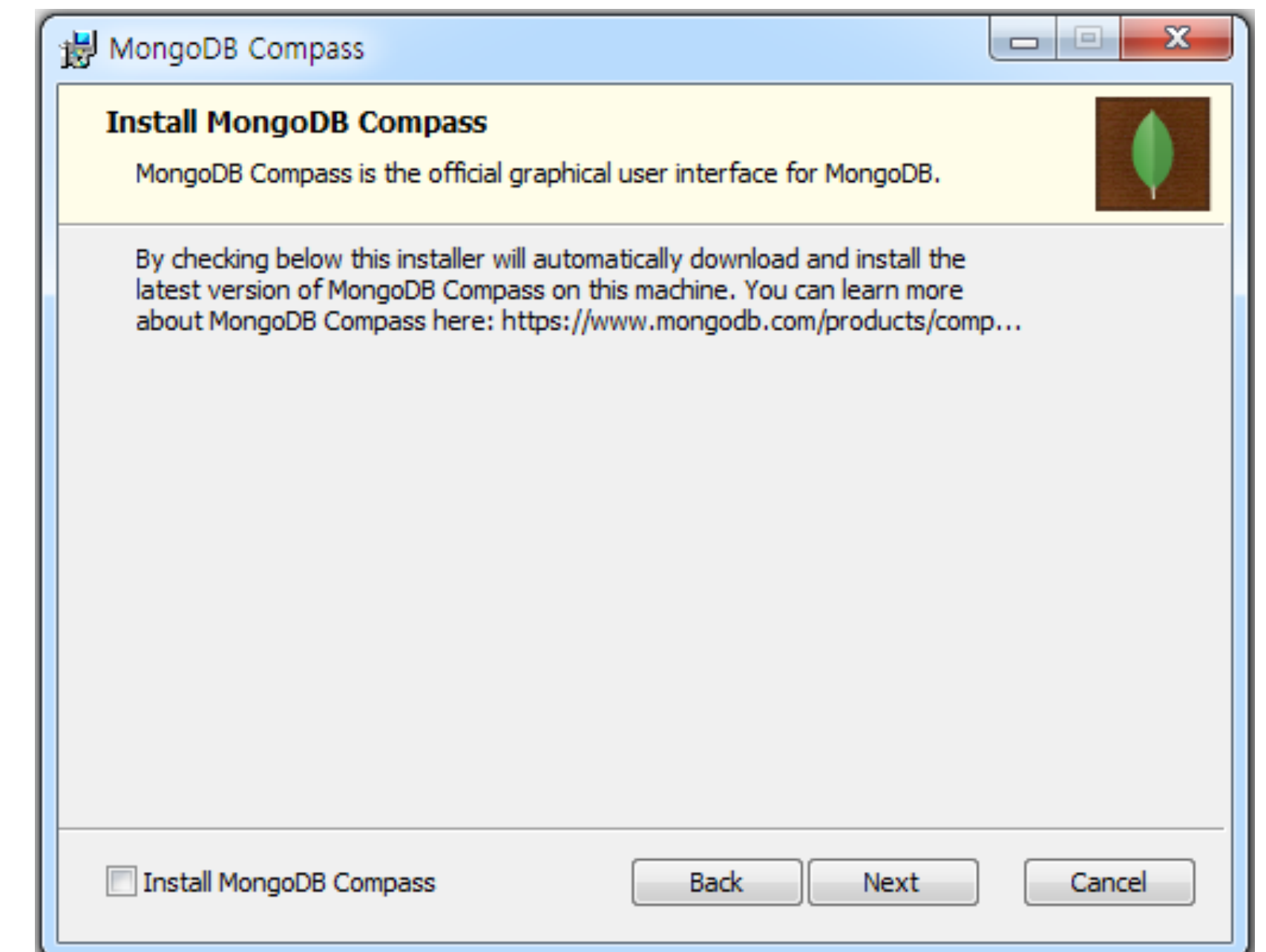
window

<https://www.mongodb.com>

Software → Community Server → Version, OS, Package 선택

\* [ ] Install MongoDB Compass 해제 !!

c:\data\db 폴더 생성



# 0.start

---

window

## Path 설정

컴퓨터 → 속성 → 고급 시스템 설정 → 환경 변수 → 시스템 변수 → Path

### 1. Server

cmd → mongod [--dbpath]

### 2. Client

cmd → mongo

# 0.start

---

window

service 생성

```
mongod.exe  
--dbpath "c:\data\db"  
--serviceName MongoDB  
--serviceDisplayName MongoDB
```

\* 한 줄로 입력할 것!

# 0.start

---

mac

## Homebrew

```
brew tap mongodb/brew
```

```
brew install mongodb-community
```

\* 경로

```
configuration file : /usr/local/etc/mongod.conf
```

```
log directory path : /usr/local/var/log/mongodb
```

```
data directory path : /usr/local/var/mongodb
```

# 0.start

---

docker

## Docker

```
docker run --name mongo -p 27017:27017 -d mongo
```

```
docker start mongo
```

```
docker exec -it mongo bash
```

```
mongo
```

# 0.start

---

nosql

NoSQL?

- 고정되지 않은 테이블 스키마
- 데이터 간의 관계를 정의하지 않는 데이터베이스
- 분산형 구조 (대용량 데이터 저장 용이)

# 0.start

---

nosql

## MongoDB

- 고정되지 않은 테이블 스키마  
필요할 때 마다 필드를 추가/제거 가능 → 개발 속도 향상
- 데이터 간의 관계를 정의하지 않는 데이터베이스  
db > collection > document
- 분산형 구조 (대용량 데이터 저장 용이)  
sharding 지원 (클러스터 데이터 상호 복제)



# 0.start

---

mongo shell

interactive javascript interface

- javascript interpreter 사용
- js program, library, function 활용 가능

# 1.Data Structure

---

database

## Database

- 독립적인 하나의 권한을 가짐
- 각각의 db는 분리된 파일로 저장
- 예약된 db name
  - admin : root db
  - local : 복제되지 않는 db (특정 서버에만 저장하는 collection에 사용)
  - config : shard 정보 저장

# 1.Data Structure

---

collection

## Collection

- document들의 group (rdbms의 table 역할)
- schema를 가지지 않는다 (document들의 field가 각각 다를 수 있다)

# 1.Data Structure

---

document

## Document

- data recode를 BSON (Binary JSON)으로 저장

- field(key) 중복 불가

- 대소문자 구별

```
{  
  name: "sue",  
  age: 26,  
  status: "A",  
  groups: [ "news", "sports" ]  
}
```

← field: value  
← field: value  
← field: value  
← field: value

# 1.Data Structure

---

명령어

show dbs

전체 database 목록

db

현재 database 확인

use dbname

해당 database로 변경

show collections

현재 database의 collection 목록

## 2. CRUD

---

`insert()`

```
db.collection.insertOne(  
    document  
)
```

```
db.collection.insertMany([  
    document,  
    document,  
    ...  
])
```

\* `_id` (primary key) : 명시하지 않으면 자동으로 ObjectId값 생성

## 2. CRUD

---

`find()`

`db.collection.find(query, projection)`

- query : query selector
- projection : 출력할 field 결정. (1: true / 0: false)

## 2. CRUD

---

cursor

```
var cursor = db.collection.find()
```

- `find()`를 통해 리턴되는 `cursor`를 `var` 변수(js)에 저장할 수 있다.
- `hasNext()`, `forEach()`, `toArray()` 등을 사용하여 `cursor` 내부의 `document`들을 사용할 수 있다.



## 2. CRUD

---

update()

`db.collection.updateOne(filter, update, options)`

– document의 **field** 수정

`db.collection.updateMany(filter, update, options)`

`db.collection.replaceOne(filter, update, options)`

– **document**를 수정

## 2. CRUD

---

update()

- filter : 수정할 document를 find()
- update : update operator or aggregation pipeline
- options : 추가적인 기능 (upsert, writeConcern, ...)

## 2. CRUD

---

`delete()`

`db.collection.deleteOne(filter, options)`

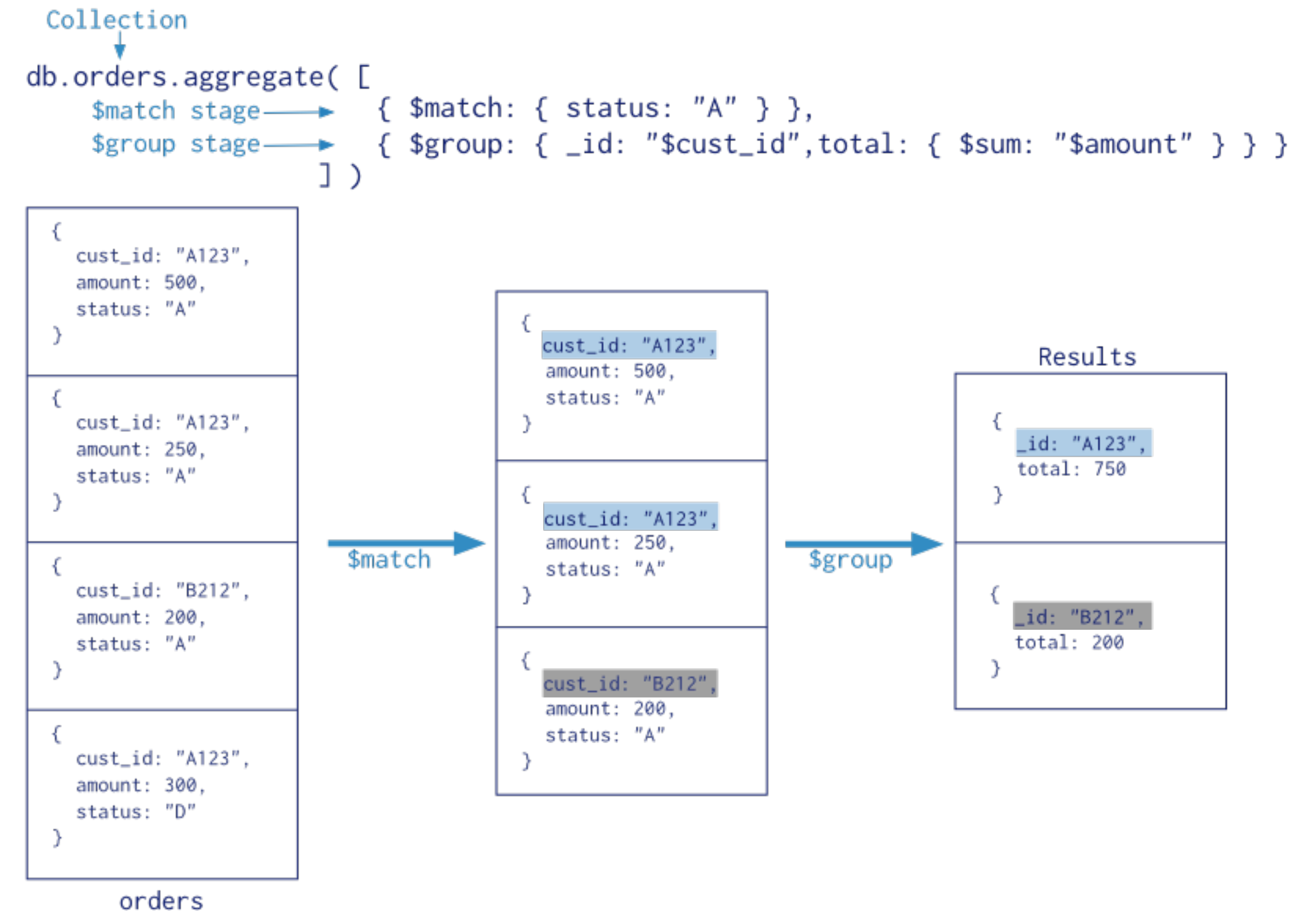
`db.collection.deleteMany(filter, options)`

- `filter` : 삭제할 document를 `find()`
- `options` : 추가적인 기능 (`writeConcern`, `collation`)

# 3. aggregation

## pipeline

- collection 이 각 stage를 거치면서 document 처리 및 집계
- 일부 처리는 shard에 대응 (각 shard에서 처리)
- \* pipeline : 이전 단계의 연산 결과를 다음 단계에서 사용
- \* stage 순서 중요!



# 3.aggregation

---

sql ↔ nosql

SQL

WHERE

GROUP BY

HAVING

SELECT

ORDER BY

LIMIT

SUM

COUNT

NoSQL

\$match

\$group

\$match

\$project

\$sort

\$limit

\$sum

\$sum

# 3.aggregation

---

aggregation

```
db.score.aggregate(  
  score collection 에서  
  {$match:{"test":"midterm"}},  
  test가 midterm 인 document들을 뽑고,  
  {$project:{"kor":1}},  
  kor만 출력시켜서, {_id:"..", kor:""}  
  {$group:{"_id":"test","average":{"$avg":"$kor"}}}  
  집계해서 {_id:"test", average:n}으로 출력한다.  
)
```

\* \$field : 해당 field 참조 시 사용 (= \$\$current.field)

# 4.map reduce

---

map reduce

- aggregation framework가 처리하지 못하는 복잡한 집계 작업에 사용
- javascript function을 사용하여 복잡한 작업 처리
- shard에 대응 → 분산 처리 가능

# 4.map reduce

map reduce

순서 : query → map → reduce → out

- map : data mapping (grouping)

- reduce : 집계 연산 실행

- query : 입력될 document

- out : collection or document 출력

```
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" }
)
```

