**MongoDB**

**설치하기 )**

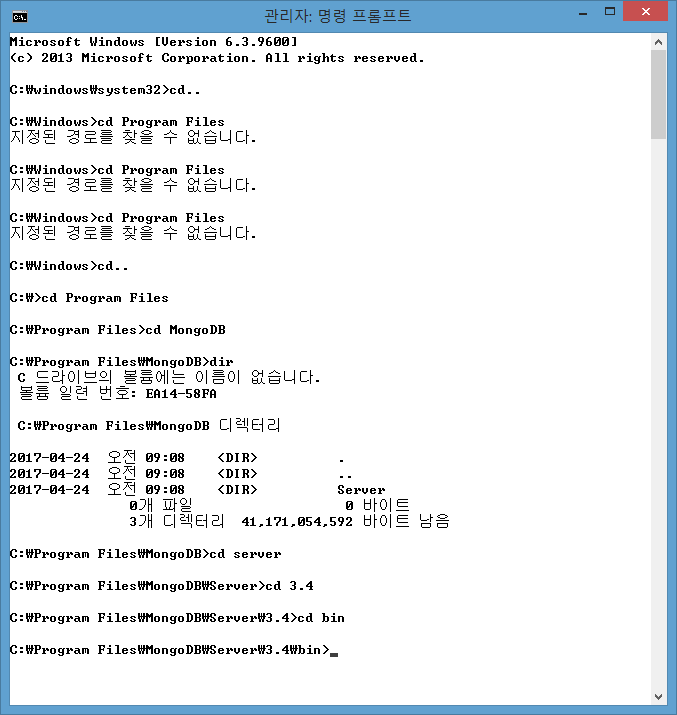
* **노트북은 와이파이땜에 잘 안될 수 있음.**
* **\\192.168.1.40**

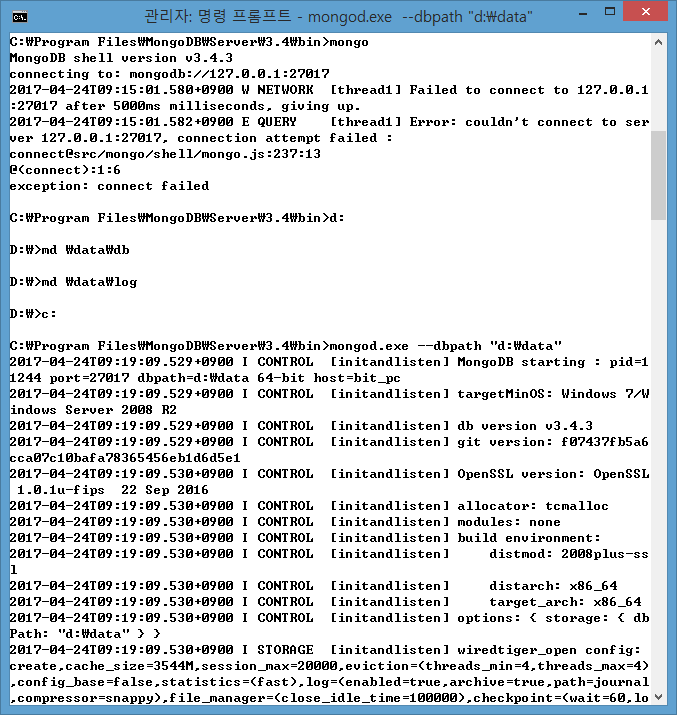
1) 몽고디비 설치(complete로 설치)

2) 환경 변수 지정

3) db, log 폴더 만들기

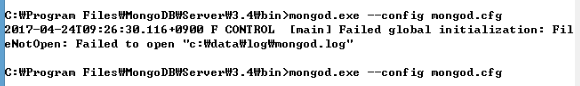
4) 서비스 등록 및 실행



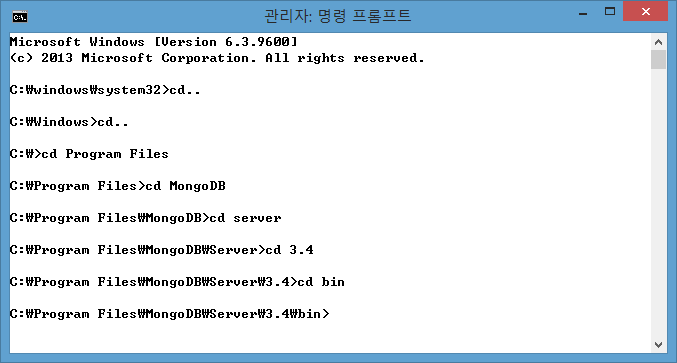


D드라이브에 생겼는지 확인

* ctrl+c 해서 종료 하면 돌아옴.
* C:\Program Files\MongoDB\Server\3.4\bin 에 “mongod.dfg” 복사

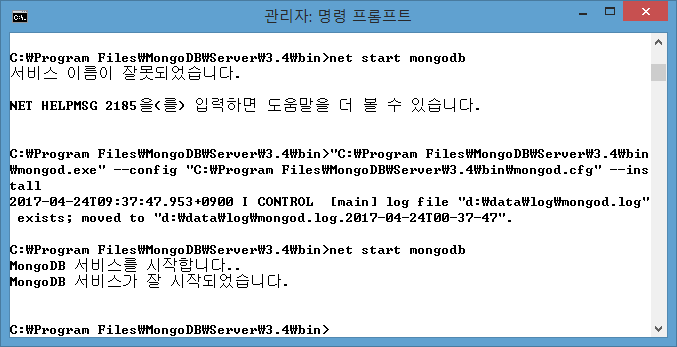


후 ctrl+c 끄고 다시 cmd 관리자권한으로실행



"C:\Program Files\MongoDB\Server\3.4\bin\mongod.exe" --config "C:\Program Files\MongoDB\Server\3.4\bin\mongod.cfg" --install

입력

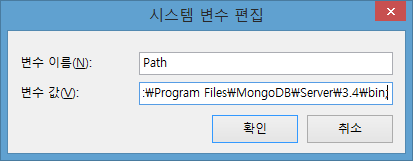


다됐네 ~ 끄면 됨.

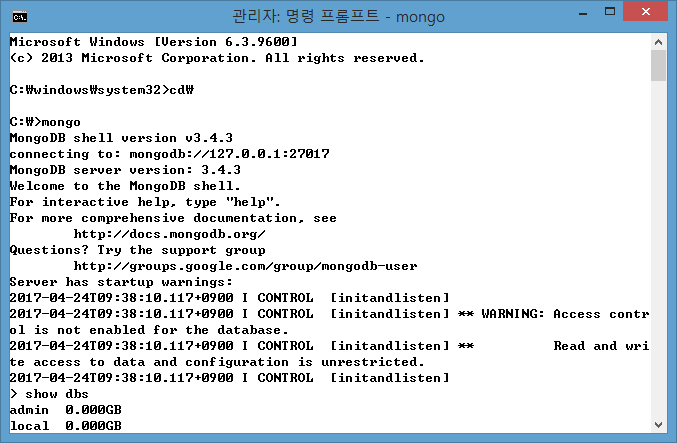
**시스템 환경변수 편집** 들어감.

환경변수 - 아래의 path 클릭 후 편집 -

;후 추가 경로 (C:\Program Files\MongoDB\Server\3.4\bin) → 빈파일 있는곳. 경로 복사



cmd 관리자 권한으로 들어가기



**RDBMS vs NoSQL**

RDBMS

* 스키마 : 데이터베이스 사용에 있어서의 규칙
* RDBMS 의 특징 : 제약사항이 존재한다. 효율적으로 공간 사용 가능. 무결성으로 인해 확실한 데이터가 들어간다는 확신 o(데이터의무결성을 보장), 보완성 뛰어남.

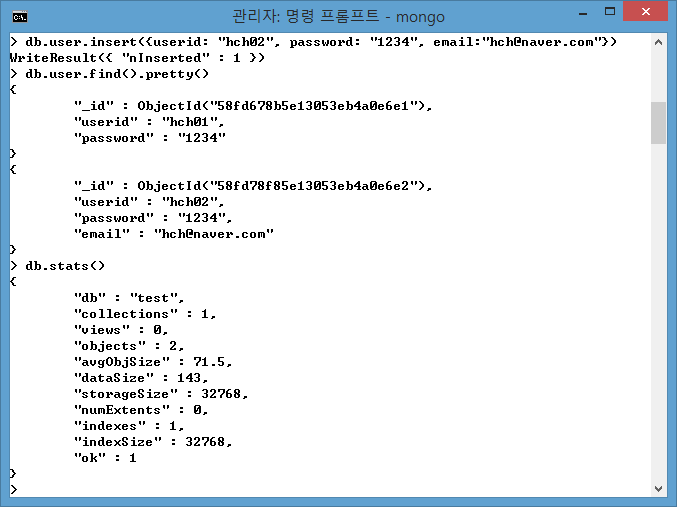
NOSQL

* 제약조건 없고 무더기로 넣는 느낌. 데이터를 빠르게 처리하려 등장.
* 샤딩& 리플리카셋

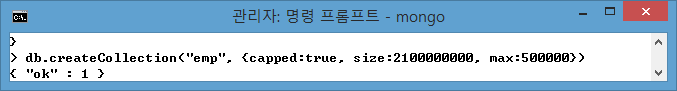
Sharding : 데이터 분할 저장 / Replica sets : 데이터 복제세트(백업, 복구)

실습)

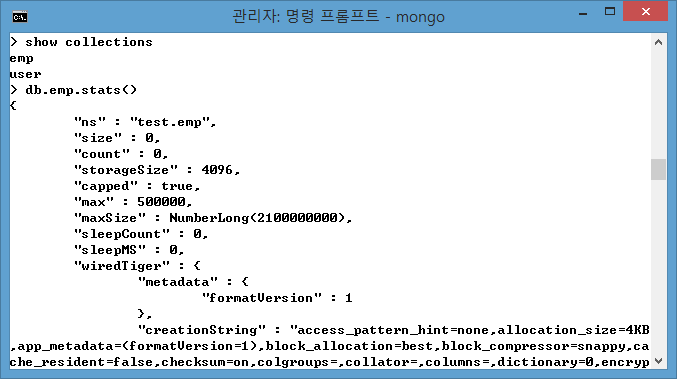
\_id: 키





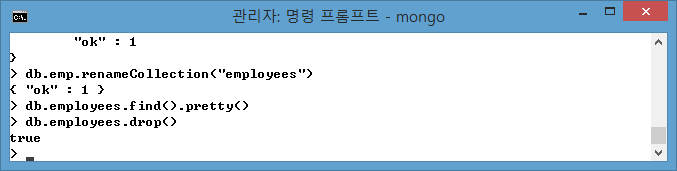


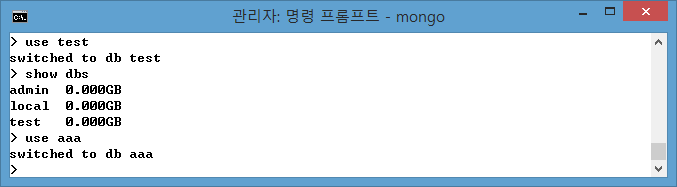
* capped: 데이터의 유지x. 21억바이트 다 채워지면 전의 데이터 다 지우고 처음부터 다시. ex: 로그. false면 이어서 붙임. 데이터 유지
* max: 하나의 데이터의 총 크기가 데이터하나의 50만 바이트.
* 사이즈 : emp라는 콜랙션의 크기.



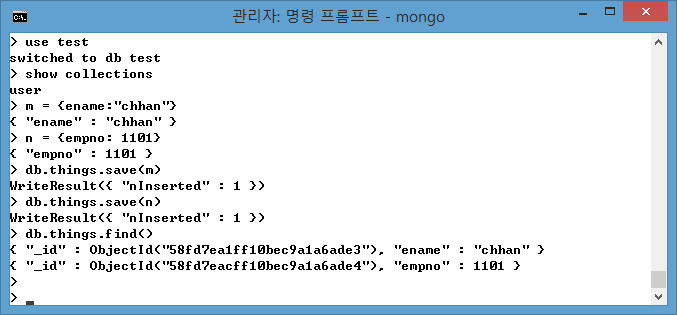
db.emp.stats() :상태보기

show collection : 테이블 보기

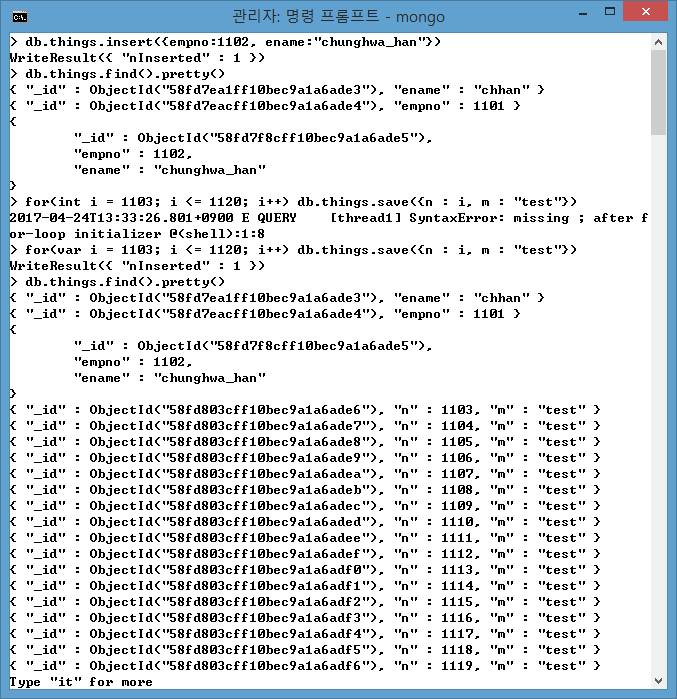


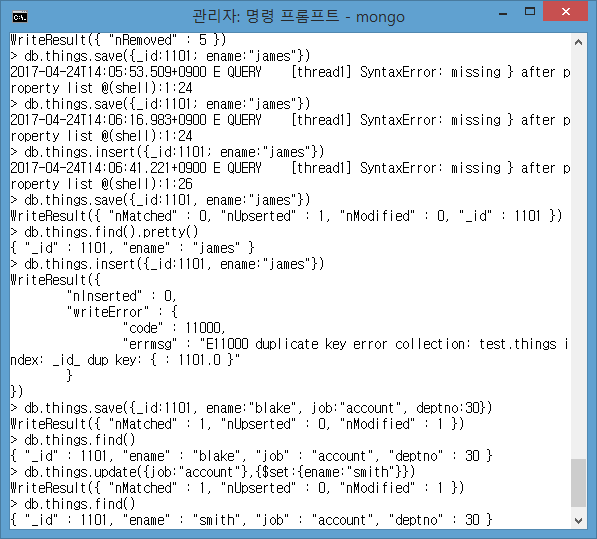


use aaa : 없으면 새로 만들어짐. drop 콜렉션 지우기



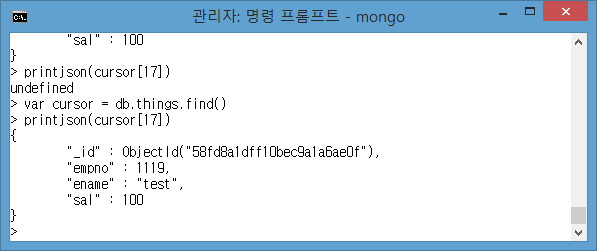
|  |
| --- |
| RDBMS  RDBMS는 테이블을 만들 때 컬럼도 만든다  CREATE TABLE EMP(  )  // NoSQL은 컬랙션(RDBMS의 TABLE) 만 만든다.  -> db.createCollection("emp")  //RDBMS는 지정된 컬럼에만 데이터 삽입  INSERT INTO EMP(EMPNO, ENAME) VALUES(110, "MHSO")  //NoSQL은 컬럼(필드)를 마음대로 지정가능함.  //NoSQL의 데이터 -> document라고 한다. (RDBMS의 row)  db.emp.insert({empno:1101, ename:"MHSO"})  UPDATE THINGS SET deptno=30, ENAME=”HI” WHERE EMPNO=1104  -> db.things.update({empno:1104}, {$set: {deptno:30, ename=”HI}})  \*insert, save, update의 차이점   * insert : document 삽입   단, \_id필드가 중복되어 있는 경우에는 삽입이 안된다.   * save : document 삽입 및 수정   삽입 : \_id 필드가 없는 경우 document 삽입  수정 : \_id 필드가 있는 경우 해당 document를 수정한다.   * update : Field 수정   지정한 조건에 맞는 필드만 수정을 하게 된다. |



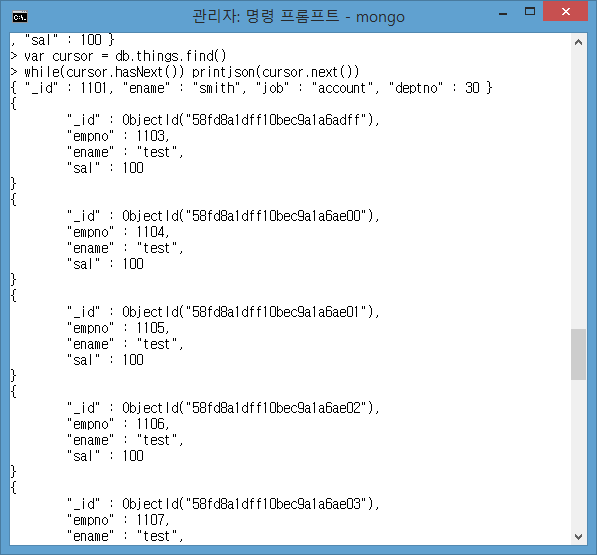


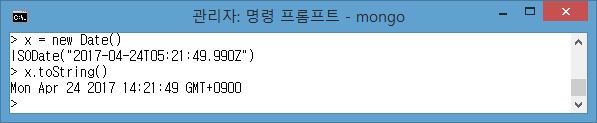


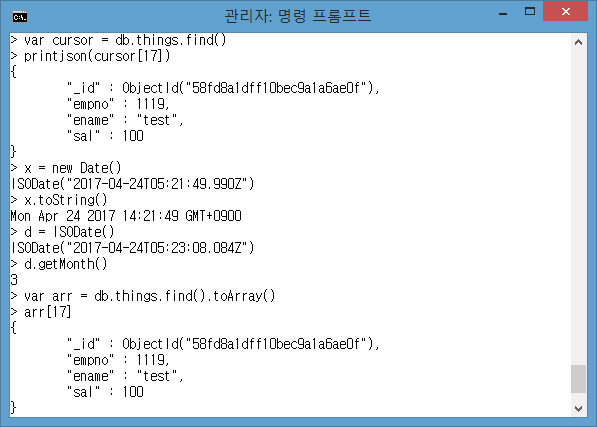
find를 배열 처럼 쓸 수 있음. cursor

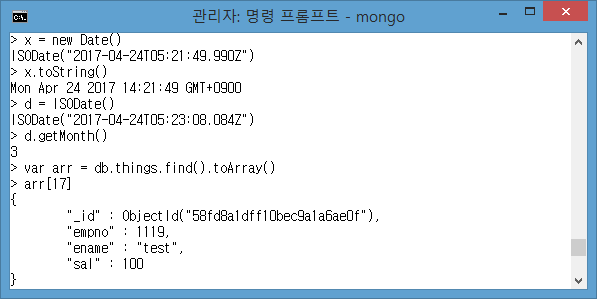


17은 인덱스

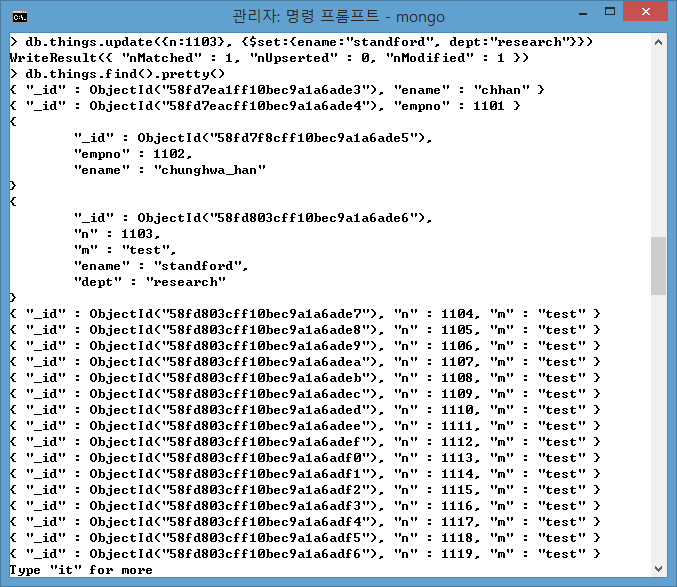






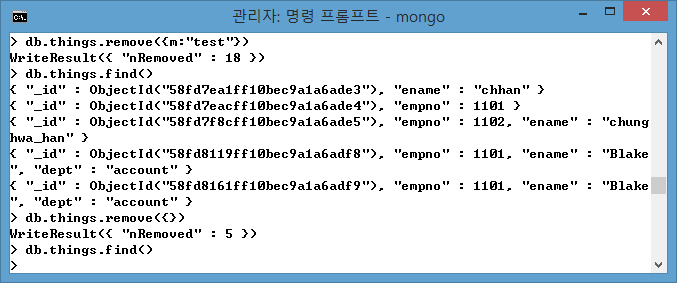


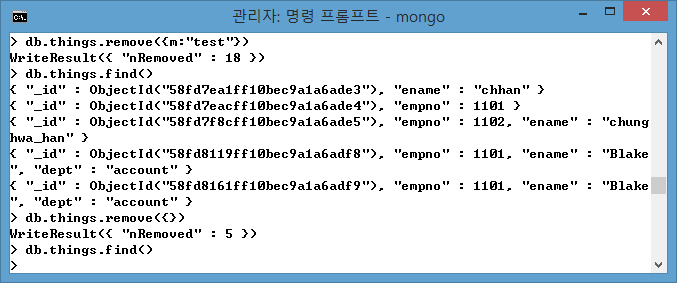
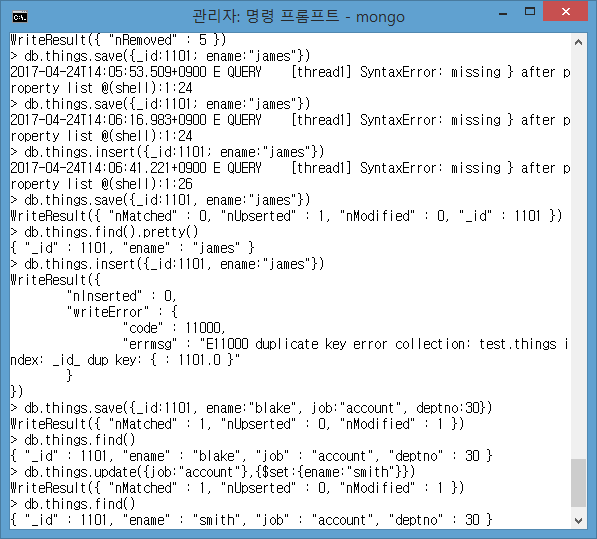
d.getYear()



n:1103은 조건 같은 것.

삭제 관련



* insert : 데이터 삽입(\_id 필드가 중복 X). 도큐먼트 삽입
* save: 데이터 삽입&수정 (도큐먼트 삽입&수정, \_id에 따라서) id값이 같으면 수정(도큐먼트 단위로). id값이 일치하지 않으면 삽입과 같음
* update: 필드 수정
* 
* 

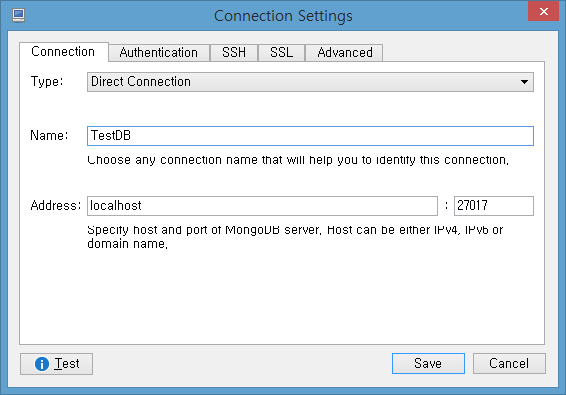
db.things.drop() 후 employees.json 파일 붙여넣기

|  |
| --- |
| 오라클과 MongoDB의 차이  SELECT \* FROM EMPLOYEES WHERE EMPNO=7369  db.employees.find({empno:7369}).forEach(printjson)   * find문법   where select  DB.컬랙션.find({조건}, {나타낼 필드})  employees 컬랙션에서 job이 CLERK인 사람 찾기  db.employees.find({job:”CLERK”}).forEach(printjson)  select ename from employees where empno=7900    1은 true의 의미 . id는 무조건 나옴.  select empno, ename from employees where ename >= “ALLEN” AND ename < “SCOTT”;  \*ename이 allen보다 크고, scott보다 작은 (기준 : 유니코드)    인덱스화 시키기. 검색이 빨라진다.    id: 0이면 안나오게 .  select empno, ename from employees where empno > 7500 and empno <= 7600  급여가 1000 이상 2000 미만인 사람들의 이름, 직업을 출력하시오    select empno from empoyees where empno=7782 or empno=7844    논리키워드는 배열을 이용 하자.  job이 salesman이거나 clerk인 사람들의 이름, 직업, 급여 출력    select count(\*) from employees;  -> db.employees.count()  select count(\*) from employees where empno > 7900  -> db.employees.find({empno:{$gt:7900}}).count()  select distinct deptno from employees;    select ename, job from employees where deptno=10 order by ename desc    select \* from employees where job=’clerk’ and sal <=1000 |

**Tool깔기**

<https://robomongo.org/> 에서 exe.파일로 깔기

create눌러서



|  |
| --- |
| 실습시간시 문제풀이)  1. select \* from employees where job = 'CLERK' and sal >=1000  -> db.employees.find({$and: [{job: "CLERK"}, {sal: {$gte: 1000}}]})  2. select \* from employees where job = 'SALESMAN' and deptno = 30;  -> db.employees.find({$and: [{job:"SALESMAN"}, {deptno:30}]})  3. select empno, ename, sal from employees where sal <= 1000  -> db.employees.find({sal:{$lte:1000}}, {\_id:0, empno: 1, ename: 1, sal: 1})  4. select empno, ename, sal from employees where sal >= 1500 and sal <= 5000  -> db.employees.find({$and: [{sal:{$gte:1500}}, {sal:{$lte:5000}}]}, {\_id:0, empno : 1, ename : 1, sal : 1})  5. select empno, ename, job from employees where deptno = 10 or deptno = 30  -> db.employees.find({$or : [{deptno: 10}, {deptno: 30}]}, {\_id: 0, empno: 1, ename: 1, job: 1})  6. select \*  from employees  where (deptno = 10 or deptno = 30) and sal >= 1500  -> db.employees.find({$and: [{$or: [{deptno: 10}, {deptno: 30}]}, {sal : {$gte:1500}}]})  7.  select \*  from employees  where job != 'SALESMAN'  order by empno asc  -> db.employees.find({job: {$ne: "SALESMAN"}}).sort({empno: 1})  8.  select empno, ename, sal, job  from employees  where (job = 'SALESMAN' or job = 'CLERK') AND sal>=1000 AND sal <= 3500  -> db.employees.find(  {  $and: [  {$or: [  {job: "SALESMAN"}, {job: "CLERK"}  ]  }, {sal: {$gte: 1000}}, {sal: {$lte: 3500}}    ]    }, {\_id: 0, empno: 1, ename: 1, sal: 1, job: 1}  )    ->  db.employees.find({  $and:[{$or:[{job:"SALESMAN"},{job:"CLERK"}]},{sal:{$gte:1000, $lte:3500}}]}) |