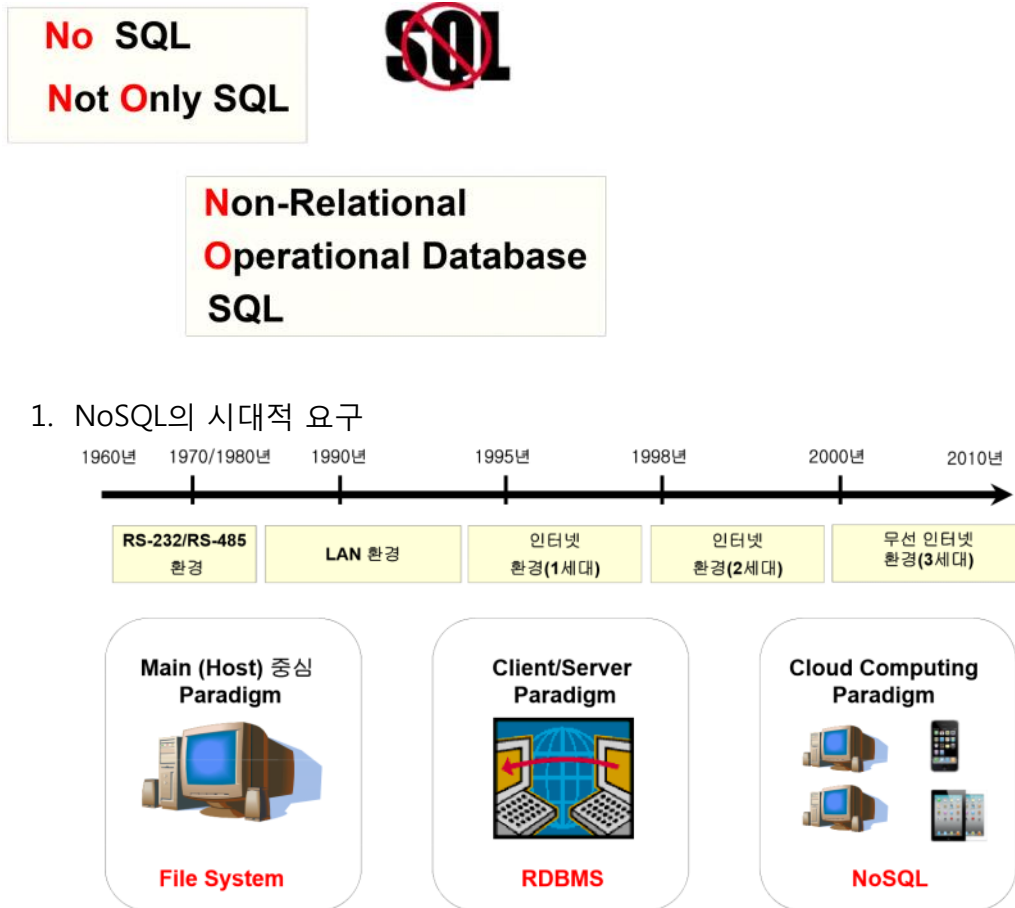


MongDB

2018년 7월 26일 목요일 오전 6:13



1. NoSQL의 시대적 요구

2. 관계형 DB의 단점

- 많은 시스템의 자원을 요구
- 여러 서버에 설치할 경우 라이선스비용이 필요함.
- 기업에서 필요로 하는 기능만 선별하여 사용할 수 없다.
(대부분의 벤더들은 폐쇄적인 마케팅을 하기 때문에 기업의 요구를 들어줄 수 없다.)
- 제품 자체가 상당한 고가의 비용을 지불해야 한다.(구축비용 + 라이선스비용 + 유지보수)

3. NoSQL의 장점

1. 클라우드 컴퓨팅 환경에 적합하다.

- 1) Open Source이다.
- 2) 하드웨어 확장에 유연한 대처가 가능하다.
- 3) 무엇보다 RDBMS에 비해 저렴한 비용으로 분산 처리와 병렬 처리가 가능하다.

2. 유연한 데이터 모델이다

- 1) 비정형 데이터 구조 설계로 설계 비용 감소
- 2) 관계형 데이터베이스의 Relationship과 Join 구조를 Linking과 Embedded로 구현하여 성능이 빠르다.

3. Big Data 처리에 효과적이다

- 1) Memory Mapping 기능을 통해 Read/Write가 빠르다.
- 2) 전형적인 OS와 Hardware에 구축할 수 있다.
- 3) 기존 RDB와 동일하게 데이터 처리가 가능하다.

4. NoSQL의 종류 및 랭킹

- a. <http://www.nosql-database.org> 참고
- b. <https://db-engines.com/en/ranking> 참고

5. 데이터를 저장 및 관리 방법에 따른 유형

1. Key-Value Database

- 1) Amazon's Dynamo Paper
- 2) Data Model : Collection of K-V pairs
- 3) 제품유형 : Riak, Voldemort, Tokyo*

3. Document Database

- 1) Lotus Notes
- 2) Data Model : Collection of K-V collection
- 3) 제품유형 : Mongo DB, Couch DB

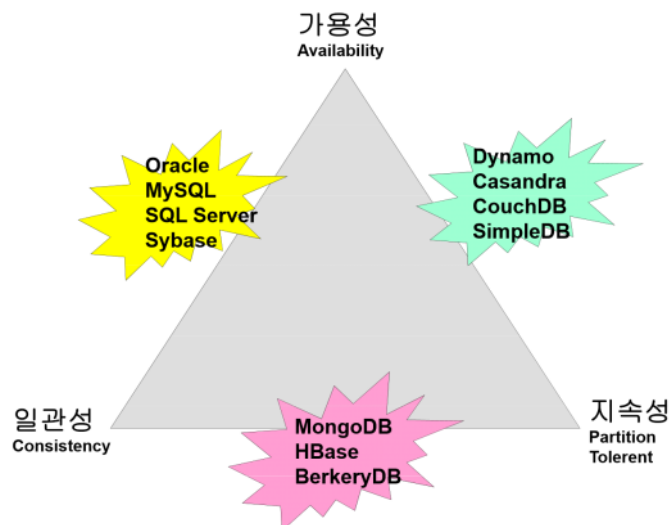
2. BigTable Database

- 1) Google's BigTable paper
- 2) Data Model : Column Families
- 3) 제품유형 : Hbase, Casandra, Hypertable

4. Graph Database

- 1) Euler & Graph Theory
- 2) Data Model : nodes, rels, K-V on both
- 3) 제품유형 : AllegroGraph, Sones

6. CAP 이론



일관성 : 동시에 많은 사용자들이 데이터를 참조했을 때 모든 사용자는 동일한 시점의 동일한 데이터를 참조해야 한다.

가용성 : 많은 사용자들이 동시에 read/write를 할 수 있어야 한다. 여러대의 서버에 데이터가 저장되어 있을 때

한쪽 노드의 데이터에 문제가 생겼다 하더라도 다른 노드에 있는 데이터를 지속적으로 사용 가능하게 해야 한다.

지속성 : 수많은 데이터를 여러 서버에 분산 저장할 수 있어야 한다.

기존의 관계형 DB 기술과 NoSQL을 연동해서 사용하는 정보시스템을 하이브리드 시스템이라고 한다.

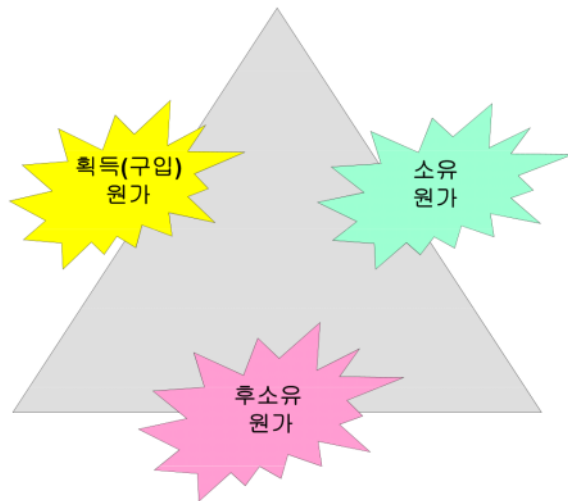
7. NoSQL 과 RDBMS의 기술적 비교

	Dynamo	Hbase	MongoDB	RDBMS (Oracle, MySQL 등)
기능	Key/Value 데이터 저장 기술 -Amazon co.	ColumnFamily 데이터 저장기술 -Apache 재단	Document 데이터 저장 기술 -10gen co.	테이블 데이터 저장 -Oracle co, MS, IBM 등
성능	빠른 쓰기/읽기 성능 우수 -In Memory 기술 적용 -모니터링 툴 제공 -Map/Reduce 사용 가능 -Hash Key 및 Secondary 인덱스 기능 제공	빠른 쓰기/읽기 성능 우수 -In Memory 기술 적용 -Hadoop Sub-System 활용가능 -Hadoop Map/Reduce 사용 가능 -Row Key 인덱스 만 제공	빠른 쓰기/읽기 성능 우수 -In Memory 기술 적용 -다양한 INDEX 기능 제공 -Map/Reduce 제공 -다양한 데이터 추출함수 제공 -모니터링 툴 기본 제공	트랜잭션 위주 데이터 처리 우수 -다양한 INDEX 제공 -다양한 함수 제공
확장성	.Scale Out 가능 -다수의 노드 확장 용이	.Scale Out 가능 -다수의 노드 확장 용이	.Scale Out 가능 -다수의 노드 확장 용이	-Scale Up 가능 -Scale Up도 가능하지만 추가 라이선스로 시간/비용증가
안정성	.복제 기능 제공 -다수의 노드로 복제 용이 -빠른 패치 제공	.복제 기능 제공 -다수의 노드로 복제 용이	.복제 기능 제공 -다수의 노드로 복제 용이 -빠른 패치 제공	.오랜 세월 시스템 안정성 확보 -다수의 노드로 복제 가능 -빠른 패치 제공
단점	.데이터 scan 처리량 1mb제한 .사용자 Interface 불편 .트랜잭션 처리가 매우 낮음 (Auto-Commit 만 지원)	.비영리 단체 Apache 재단 운영 .Hadoop System 연계 가능하지만 여러 프로젝트에서 만들어짐 .트랜잭션 처리가 매우 낮음 (Auto-Commit 만 지원) .기술지원 업체가 없음	.트랜잭션 처리가 낮음 (Auto-Commit/Rollback가능)	-Scale Out 가능하지만 고가비용 -대용량 데이터 처리에 적정하지 않는 메모리 구조 -조인으로 인한 성능 지연유발 -분석/설계 시 시간 및 비용 증가 -유지보수 비용 증가 -고 사양의 시스템 요구 -지속성 보장이 안됨

8. MongoDB EcoSystem Benifit

			
General Purpose	Document Database	Open Source	Low Cost

총 소요비용(TCO-Total Coast of OwnerShip)



기업이 DBMS를 최초 선택을 한 후 분석을 하고 이 기술을 이용해서 시스템을 구축하고 향후 몇년간 이 시스템을 유지하기 위해서는 비용이 발생한다.

이 과정(선택-구축-유지)에서 발생하는 모든 비용을 TCO라고 한다.

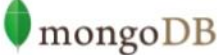

TCO는 크게 3가지 항목으로 나뉜다.(획득원가, 소유원가, 후소유원가)

이 세가지를 다시 2가지 유형으로 나눌 수 있다.



TCO Framework

선행 비용(UpFront Cost)	진행 비용(Ongoing Cost)
초기 개발자 비용 (Application Coding 및 Data Store)	진행 개발자 비용 (사용자 요구에 따라 변경 등)
초기 관리자 비용 (설치, 환경설정, Shard/Replication 등)	진행 관리자 비용 (Data 유지보수, Health Check, Backup & Recovery 등)
Software Licenses	Software Maintenance & Support
Server Hardware	Server Maintenance & Support
Server Storage	Storage Maintenance & Support
	기타 개발 비용

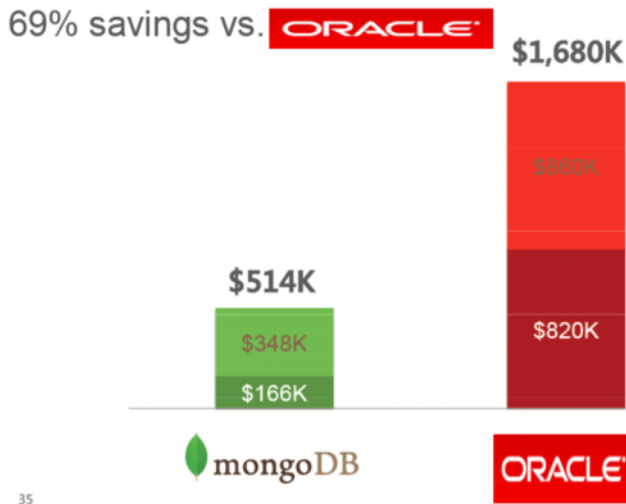
UpFront Cost(Small Project)

		
Software	MongoDB Enterprise	Oracle Enterprise (Scale Out-Add on)
Server HW	3 Server (8 Core/32GB ram/per Server)	3 Server (8 Core/32GB ram/per Server)
Storage HW	3 one-TB SSDs	3 TB SAN
Initial Dev. Effort	\$120K • 12 man-months • \$120K annual salary	\$240K • 24 man-months • \$120K annual salary
Initial Admin. Effort	\$10K • 1 man-month • \$120K annual salary	\$20K • 2 man-months • \$120K annual salary
Software Licenses	\$0 • (Ongoing Cost for Subscription)	\$423K • \$17.6K/core • 75% discount
Server HW	\$12K • 3 servers; 8/cores & 32GB RAM/server • \$4K/server	\$12K • 3 servers; 8/cores & 32GB RAM/server • \$4K/server
Storage HW	\$24K • 2 One-TB SSDs/server (1 mirrored SSD) • \$4K/SSD	\$125K • 3 TB SAN • \$125K
Total Upfront	\$166K	\$820K

OnGoing Cost(Small Project) : 유지 보수 비용

		
Software	MongoDB Enterprise	Oracle Enterprise (Scale Out-Add on)
Server HW	3 Server (8 Core/32GB ram/per Server)	3 Server (8 Core/32GB ram/per Server)
Storage HW	3 one-TB SSDs	3 TB SAN
Ongoing Dev. Effort	\$60K • 0.5 developers • \$120K annual salary	\$120K • 1 developer • \$120K annual salary
Ongoing Admin. Effort	\$30K • 0.25 DBAs • \$120K annual salary	\$60K • 0.5 DBAs • \$120K annual salary
SW Maint. & Support	\$23K • 3 servers • \$7.5K/server/year	\$93K • 22% of license fees
HW Maint. & Support	\$4K • 10% of HW cost	\$14K • 10% of HW cost
Total Ongoing per Year	\$116K per Year	\$287K per Year
Total Ongoing over 3 Years	\$348K	\$860K

결론 (Small Project : 3 Year TCO)



35

9. 사례

- a. Disney Interactive Media Group
 - i. 다양한 Game, Media Data 관리 시스템에 적용
 - ii. MySQL 바이너리 데이터 저장 한계 및 성능 문제 - MongoDB로 전환
 - iii. ReplicaSets & Auto Sharding 유연성과 확장성 활용
- b. Music Television -MTV
 - i. 비디오/오디오 Content Management System에 적용
 - ii. MySQL을 NoSQL로 전환
 - iii. MTV의 계층적 데이터 구조에 적합한 데이터 모델 활용
 - iv. 쉬운 Query와 Index를 이용한 빠른 검색 기능 활용
- c. Business Media Company - Forbes
 - i. 원고 자동 수집 및 발생 시스템에 적용
 - ii. Oracle DB를 NoSQL로 전환 - MongoDB 활용
 - iii. 전형적인 Static Data관리에서 Dynamic Data관리로 전환하면서 발생하는 재설계 및 구축 비용 절감 목적으로 활용
- d. 인터넷 기반 사진 정보 및 개인 출판 서비스 사이트 - Shutterfly
 - i. Oracle DB를 NoSQL로 전환(20TB)
 - ii. 100만명 고객/60억개의 이미지/초당 10,000개 트랜잭션 처리에서 발생하는 구축/관리 비용 및 성능 문제가 이슈
- e. 위치 기반 Social Network 사이트 - foursquare
 - i. RDB 기반의 시스템 확장 비용 및 관리 문제가 이슈
 - ii. GeoSpatial Index 기능 활용
 - iii. ReplicaSets & Auto Sharding System 활용
- f. 미국 국가 기록원 - A The National Archives
 - i. 콘텐츠 정보 관리 시스템에 적용
 - ii. SQL Server에서 NoSQL로 전환(117GB)
 - iii. 쉽고 간편한 GridFs 기능 활용

10. MongoDB 주요 특징

- a. Humongos라는 회사의 제품명이었으며 10gen으로 회사명으로 변경하였다가 현재는 MongoDB inc로 변경되었다.
- b. **JSON Type**의 데이터 저장 구조를 제공한다.
(Standard ECMA-262 3rd Edition-1999을 근거로 하는 **JavaScript형태의 데이터 표현** 방식을 근거로 한다.[European Computer Manufactures Association])
- c. **Sharding(분산)/Replica(복제)** 기능을 제공한다.

- d. MapReduce(분산/병렬처리)기능을 제공한다.
- e. CRUD 위주의 다중 트랜잭션 처리도 가능하다.
 - i. 동시 트랜잭션에서는 적합하지 않다.
- f. Memory Mapping기술을 기반으로 Big Data처리에 탁월한 성능을 제공한다.

11. 설치

- a. <https://www.mongodb.com/>

Server - mongod.exe

Router - mongos.exe

Client - mongo.exe

MonitoringTools - mongostat.exe, mongotop.exe

ImportExportTools - mongodump.exe, mongorestore.exe, mongoexport.exe, mongoimport.exe

MiscellaneousTools - bsondump.exe, mongofiles.exe, mongooplog.exe, mongoperf.exe

- b. MongoDB 설정

mongod.exe를 통해 db path를 설정한다.

"C:\Program Files\MongoDB\Server\3.4\bin\mongod.exe" --dbpath d:\test\mongodb\data
db path를 설정할때 공백이 있을경우 "" 로 감싸준다.

"C:\Program Files\MongoDB\Server\3.4\bin\mongod.exe" --dbpath "d:\test\mongodb\data"

mongoDB 데몬을 실행한다.

"C:\Program Files\MongoDB\Server\3.4\bin\mongod.exe"

로그에 다음과 waiting for connections on port 27017 같은 구문이 떠서 대기상태이라면 새 커맨드 프롬프트를 켜서 클라이언트를 실행하도록

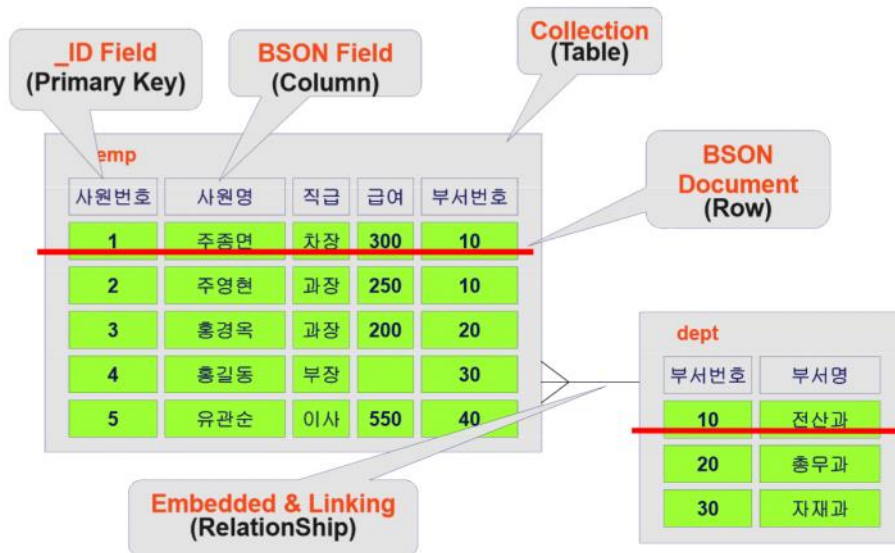
"C:\Program Files\MongoDB\Server\3.4\bin\mongo.exe"

DBMS 종류	실행 파일	Client 툴
Mysql	Mysqld.exe	Mysql.exe
Oracle	Oracle.exe	Sqlplus.exe
MongoDB	Mongod.exe	Mongo.exe

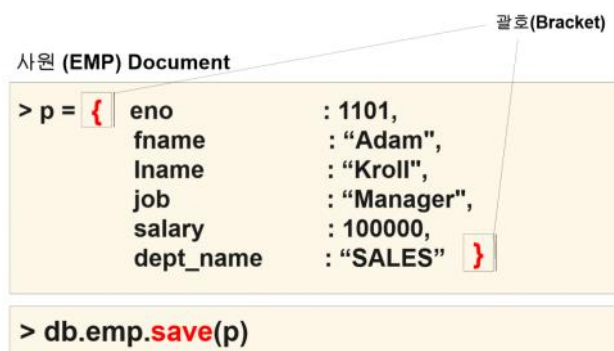
12. 실행

- a. 샘플 테이블(관계형DB와의 용어 구별)

- i. 관계형에서의 pk와 mongoDB에서의 _id는 완전히 다른 성격이다. 하지만 여기서는 편의상 같이 묶어서 보기로 한다.
- ii. 참조관계인 Relationship도 mongoDB에서는 Embedded 또는 Linking이라 부르지만 마찬가지로 서로 다른 성격이기 때문에 여기서는 편의상 묶어서 보기로 한다.



b. JSON(Java Script Object Notation)



c. Collection 생성과 삭제

```
db.createCollection("emp", {capped:false, size:8192});
```

```
show collections
```

```
db.emp.validate()
```

```
db.emp.insert ({eno:1101, fname:"JIMMY"})
```

```
db.emp.insert ({eno:1102, fname:"ADAM", lname:"KROLL"})
```

```
db.emp.insert ({eno:1103, fname:"SMITH", job:"CLERK"})
```

```
db.emp.find().sort({eno:-1})
```

```
db.emp.update ({eno:1101}, {$set:{fname:"JOO"}})
```

```
db.emp.update ({eno:1102}, {$set:{job:"CHIEF"}})
```

```
db.emp.update ({eno:1103}, {$set:{lname:"STANFORD"}})
```

```
db.emp.find().sort({eno:-1}).pretty()
```

```
db.emp.remove ({eno:1101})
```

```
db.emp.find().pretty()
```

d. SQL & MongoDB Query 비교

SQL Statement	Mongo Query Statement
CREATE TABLE emp	...

SQL Statement	Mongo Query Statement
CREATE TABLE emp (empno Number, ename Number)	db.createCollection("emp")
INSERT INTO emp VALUES(3,5)	db.emp.insert({empno:3, ename:5})
SELECT * FROM emp	db.emp.find()
SELECT empno, ename FROM emp	db.emp.find({}, {empno:1, ename:1})
SELECT * FROM emp WHERE empno=3	db.emp.find({empno:3})
SELECT empno, ename FROM emp WHERE empno=3	db.emp.find({empno:3}, {empno:1, ename:1})
SELECT * FROM emp WHERE empno=3 ORDER BY ename	db.emp.find({empno:3}).sort({ename:1})
SELECT * FROM emp WHERE empno > 3	db.emp.find({empno:{\$gt:3}})
SELECT * FROM emp WHERE empno != 3	db.emp.find({empno:{\$ne:3}})
SELECT * FROM emp WHERE ename LIKE "%Joe%"	db.emp.find({ename:/Joe/})
SELECT * FROM emp WHERE ename LIKE "Joe%"	db.emp.find({ename:/^Joe/})
SELECT * FROM emp WHERE empno>1 AND empno <=4	db.emp.find({empno:{\$gt:1,\$lte:4}})
SELECT * FROM emp ORDER BY ename DESC	db.emp.find().sort({ename:-1})
SELECT * FROM emp WHERE empno=1 and ename='Joe'	db.emp.find({empno:1,ename:'Joe'})
SELECT * FROM emp WHERE empno=1 or empno=3	db.emp.find({ \$or : [{ empno : 1 } , { empno : 3 }] })
SELECT * FROM emp WHERE rownum = 1	db.emp.findOne()
SELECT empno FROM emp o, dept d WHERE d.deptno=o.deptno AND d.deptno=10	o = db.emp.findOne({empno:1}); name = db.dept.findOne({deptno:o.deptno})
SELECT DISTINCT ename FROM emp	db.emp.distinct('ename')
SELECT COUNT(*) FROM emp	db.emp.count()
SELECT COUNT(*) FROM emp where deptno > 10	db.emp.find({deptno: {\$gt: 10}}).count()
SELECT COUNT(sal) from emp	db.emp.find({sal: {\$exists: true}}).count()
CREATE INDEX i_emp_ename ON emp(ename)	db.emp.ensureIndex({ename:1})
CREATE INDEX i_emp_no ON emp(deptno ASC, ename DESC)	db.emp.ensureIndex({deptno:1,ename:-1})
UPDATE emp SET ename='test' WHERE empno=1	db.emp.update({empno:1}, {\$set:{ename:'test'}})
DELETE FROM emp WHERE deptno=10	db.emp.remove({deptno:10});