

# Big Data 환경의 데이터 모델링 방안 - NoSQL

비투엔컨설팅 / 서동재

2013.09.08



## 주제 : Big Data 환경의 데이터 모델링 방안

- 목표 : 1. 기존 ER모델과 NoSQL 모델에 대한 비교 분석을 통해 정확한 차이점 인지
2. NoSQL 중 카산드라 모델링을 살펴봄으로써 빅데이터 모델링 개념 이해

**NoSQL** : Cassandra 1.2.5

1. ER 모델과 NoSQL 모델은 어떻게 다른가?
2. Cassandra 모델링을 배워보자.
3. ER 모델을 NoSQL 모델로 변경해보자.
4. 실제 사례를 살펴보자.
5. NoSQL 모델링은 누가해야 하나?
6. 마치면서

知彼知己 百戰百勝 (지피지기면 백전백승)

NoSQL 모델링의 기초-실전-사례

NoSQL 모델링의 현재와 미래

## 1. ER 모델과 NoSQL 모델은 어떻게 다른가?

2. Cassandra 모델링을 배워보자.
3. ER 모델을 NoSQL 모델로 변경해보자.
4. 실제 사례를 살펴보자.
5. NoSQL 모델링은 누가해야 하나?
6. 마치면서

1.1 ER 모델이란?

1.2 NoSQL 모델이란?

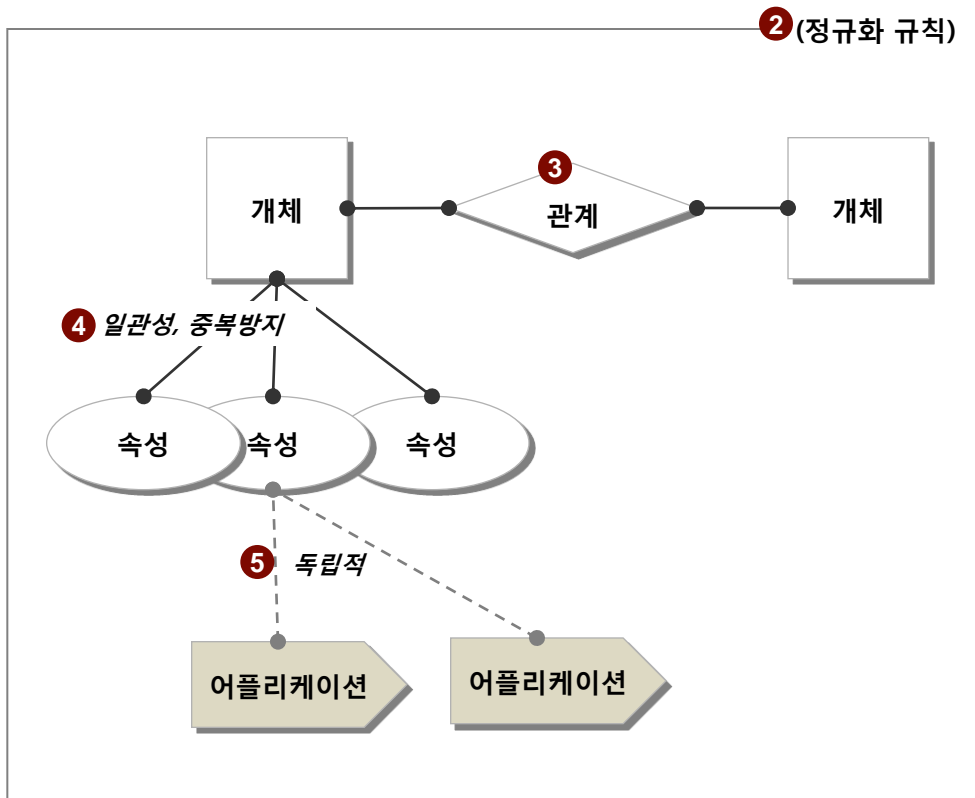
1.3 ER vs NoSQL 모델 차이점과 시사점

# 1.1 ER 모델이란?

ER모델의 정의  
(Entity-Relationship)

개체와 개체 간의 논리 관계의 집합을 표현하는 모델링 기법

## 1 Entity - Relationship 모델



### 1 개체-관계로 구성

- 실체와 실체 간의 관련을 표현하는 도식

### 2 정규화된 모델(Normalized Model)

- 1NF, 2NF, 3NF  
- BCNF...

### 3 조인과 트랜잭션 존재(Joins and Transactions)

- 개체간의 관계를 이용한 조인  
- 데이터 변경을 위한 논리적 작업단위 존재

### 4 관계 모델로 데이터 오류를 상당부분 예방함

- 일관성 체크  
- 중복 방지, 참조 무결성

### 5 어플리케이션에 독립적인 설계

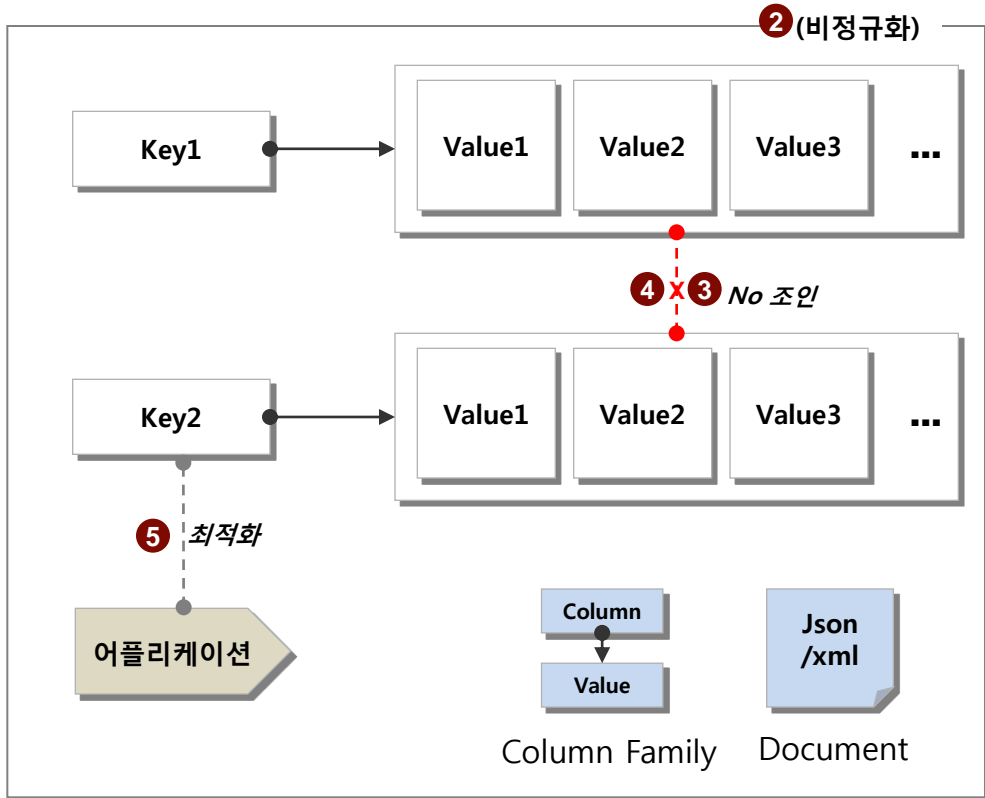
- 어플리케이션 특이성과 독립적인 데이터 구조 설계

# 1.2 NoSQL 모델이란?

NoSQL 정의  
(Not Only SQL)

RDB에 적합하지 않으면 RDB에 얽매이지 말고 용도에 맞는 데이터 저장구조를 사용

## 1 NoSQL 모델



### 1 Key / Value로 구성

- Key/Value
- Key/Column Family
- Document / Graph 등등

### 2 비정규화

- 정규화가 의미 없음

### 3 No 조인, No 트랜잭션

- 관계가 없으므로 참조 무결성도 조인도 없음
- No 조인, No 트랜잭션






### 4 데이터 무결성/정합성 보장 어려움

- 관계가 없으므로 참조 무결성도 조인도 없음
- 필요하다면 어플리케이션에서 처리하게 설계

### 5 어플리케이션에 최적화된 설계

- 데이터는 동일한데 어플리케이션이 완전히 바뀌는 경우

# 1.3 ER 모델 vs NoSQL 모델 차이점과 시사점

관점	ER 모델링	NoSQL 모델링	비고
설계 목적	<ul style="list-style-type: none"> <li>정형화된 업무의 데이터 무결성/정합성/일관성 있는 <b>저장구조 설계</b></li> </ul>	<ul style="list-style-type: none"> <li>대용량 데이터에서 빠른 응답 성능과 확장성, 가용성 좋은 <b>저장구조 설계</b></li> </ul>	저장구조에 대한 설계
구성 요소 	<ul style="list-style-type: none"> <li>개체, 관계, 속성</li> </ul>	<ul style="list-style-type: none"> <li>Key, Value, Column</li> </ul>	관계를 제외하고 대응되는 부분 존재
정규화 	<ul style="list-style-type: none"> <li>1NF, 2NF, 3NF, BCNF</li> </ul>	<ul style="list-style-type: none"> <li>정규화 개념 없음</li> </ul>	
조인과 트랜잭션 	<ul style="list-style-type: none"> <li>조인 존재</li> <li>트랜잭션 존재</li> </ul>	<ul style="list-style-type: none"> <li>No 조인</li> <li>No 트랜잭션</li> </ul>	
참조무결성 	<ul style="list-style-type: none"> <li>참조무결성 존재</li> </ul>	<ul style="list-style-type: none"> <li>참조무결성 없음</li> </ul>	
어플리케이션 	<ul style="list-style-type: none"> <li>어플리케이션 특이성에 독립적으로 설계</li> </ul>	<ul style="list-style-type: none"> <li>어플리케이션에 최적화된 설계</li> </ul>	

**1. ER모델링과 NoSQL모델링은 서로 많이 다르고 새로운 관점으로 해석해야 한다.**

**2. 하지만 기본적으로 저장구조에 대한 설계이므로 비슷한 부분이 존재한다.**

1. ER 모델과 NoSQL 모델은 어떻게 다른가?

## 2. Cassandra 모델링을 배워보자.

3. ER 모델을 NoSQL 모델로 변경해보자.

4. 실제 사례를 살펴보자.

5. NoSQL 모델링은 누가해야 하나?

6. 마치면서

2.1 Cassandra 개요

2.2 Cassandra 기본 개념

2.3 Column Family 유형

2.4 복합키(Composite Key)의 특징

2.5 Cassandra Simple 모델링









2.6 Column Family 생성 및 변경 테스트

## 2. Cassandra 모델링을 배워보자

### 2.1 Cassandra 개요

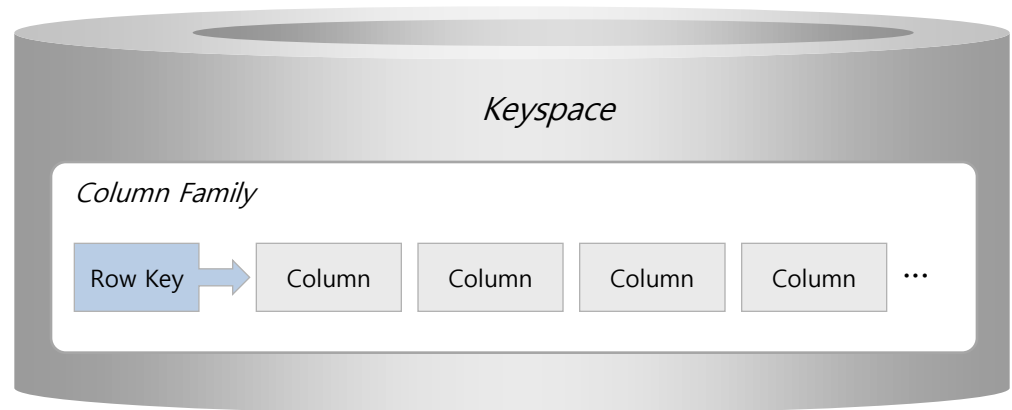
#### 1) Cassandra 란?

- 컬럼패밀리 형태의 NoSQL 데이터베이스
- Twitter, eBay, Cisco 등 천여 개가 넘는 조직에서 사용되는 Big Data 인프라

구분	NoSQL
Key/Value	 
ColumnFamily	 
Document	  
Graph	

#### 2) Cassandra 데이터 구조

- Key, Column, Column Family 구조를 가짐
- Keyspace는 논리적으로 Column Family를 묶어 주는 개념





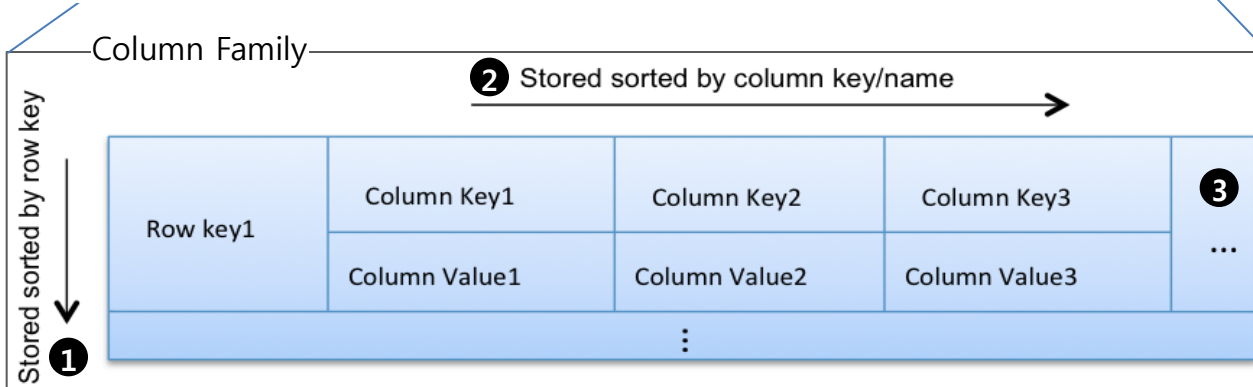
## 2. Cassandra 모델링을 배워보자

### 2.2 Cassandra 기본 개념

#### 1) Cassandra Elements

ER 모델	Cassandra 모델	비고
Database	Keyspace	
Table	Column Family(CF)	
Primary Key	Row Key	
Column Name	Column Name/Key	Name/Value/Timestamp 로 구성
Column Value	Column Value	

#### 2) Column Family 구조

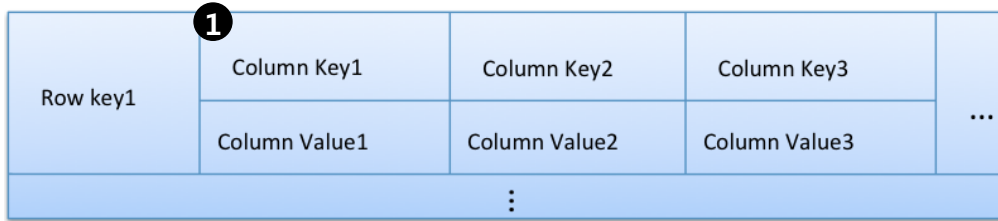


- ❶ Row Key 로 정렬하여 저장
- ❷ Column Key로 정렬하여 저장
- ❸ Column의 수는 제한이 없음

## 2. Cassandra 모델링을 배워보자

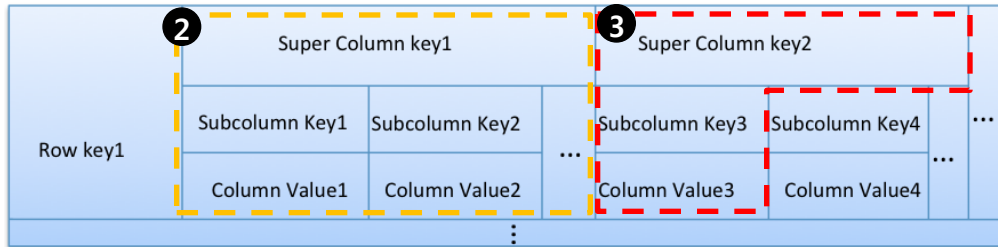
### 2.3 Column Family 유형

#### 1) 기본형(CF)



① 컬럼의 중첩이 없는 단순형(기본형)

#### 2) 슈퍼컬럼패밀리(SCF)

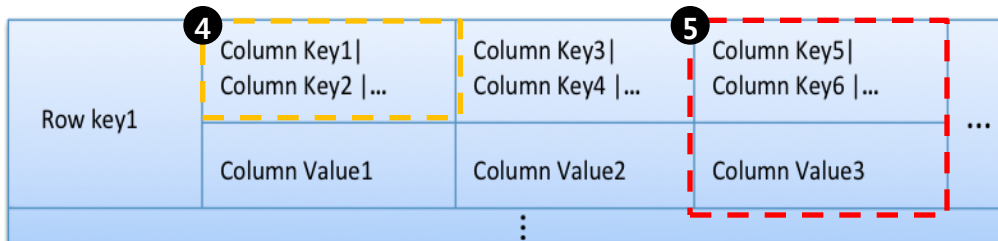


② Column Family안에 Column Family가 들어가는 형태(중첩된 형태)

③ Super Column의 한 서브컬럼만 읽을 수 없으며, 하위 서브컬럼을 같이 읽어야 함

- 서브컬럼이 수천 개가 넘는다면 권장하지 않음 (성능 문제 발생)

#### 3) 복합키(Composite Key) → 가장 중요 뒷장에 상세하게 설명



④ Userid:lastupdate 같은 형태로 복합키를 구성하고 값을 저장하는 방식

⑤ 복합키별로 값의 접근 가능

- 슈퍼컬럼패밀리의 대안이 될 수 있음

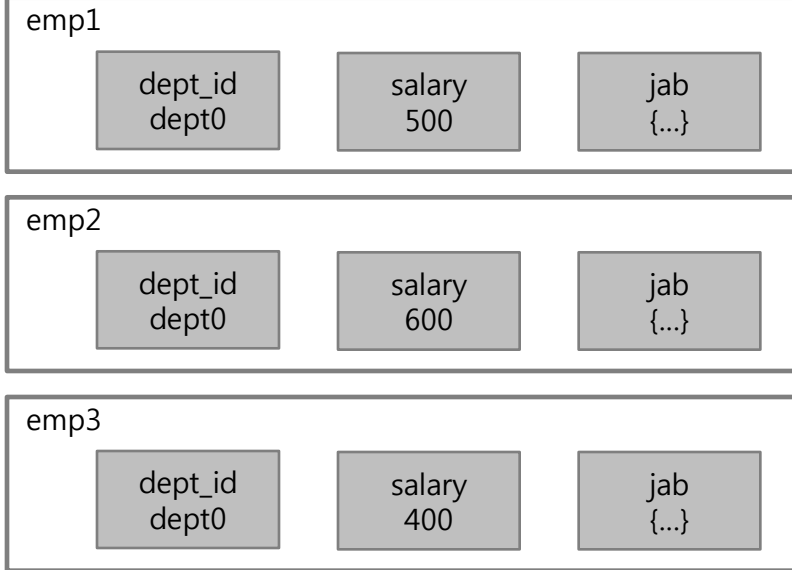
## 2. Cassandra 모델링을 배워보자

### 2.4 복합키(Composite Key)의 특징

#### Single-Key 테이블

생성 CQL)

```
create table emp (  
  emp_id varchar PRIMARY KEY,  
  salary bigint,  
  job varchar  
);
```

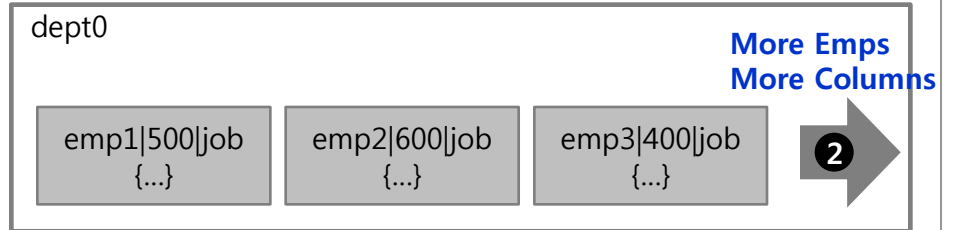


1 More Emps  
More Rows

#### Composite Key 테이블

생성 CQL)

```
create table emp (  
  dept_id varchar,  
  emp_id varchar,  
  salary bigint,  
  job varchar,  
  PRIMARY KEY (dept_id, emp_id)  
);
```



- 1 Single-Key 테이블은 직원이 추가될 때 row로 증가한다.
- 2 Composite Key로 구성할 경우 직원이 추가되면 컬럼으로 증가한다.

→ Key 구조에 따라 row 또는 column으로 데이터 증가

## 2. Cassandra 모델링을 배워보자

### 2.4 복합키(Composite Key)의 특징 - 예시

#### Tweets 컬럼패밀리

tweet_id	author	body
1742	홍길동	NoSQL 시대 열린다.
1765	이순신	빅데이터 전문가가 뜬다.
1778	강감찬	그래도 RDB가 최고다.

```
CREATE TABLE tweets (
  tweet_id uuid PRIMARY KEY,
  author varchar,
  body varchar );
```

#### timeline 컬럼패밀리

user_id	tweet_id	author	body
김철수	1765	이순신	빅데이터 전문가가 뜬다.
김철수	1742	홍길동	NoSQL 시대 열린다.
이영희	1778	강감찬	그래도 RDB가 최고다.
이영희	1742	홍길동	NoSQL 시대 열린다.

```
CREATE TABLE timeline (
  user_id varchar,
  tweet_id uuid,
  author varchar,
  body varchar,
  PRIMARY KEY (user_id, tweet_id)
);
```

#### Timeline Physical Layout **② Clustered by tweet\_id**

김철수	[1765,author] : 이순신	[1765, body] : 빅데이터 전문가가 뜬다.	[1742,author] : 홍길동	[1742, body] : NoSQL 시대 열린다.
이영희	[1778,author] : 강감찬	[1778, body] : 그래도 RDB가 최고다.	[1742,author] : 홍길동	[1742, body] : NoSQL 시대 열린다.

#### ① 복합키 구성

- Row Key + Remaining Key

#### ② Remaining Key로 Clustered

- 두 번째로 지정한 컬럼 값으로 모여서 저장됨

## 2. Cassandra 모델링을 배워보자

### 2.5 Cassandra Simple 모델링

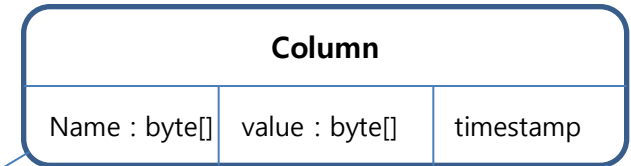
#### Step 1 • Column Family 설계 기본원칙

- Column Family는 하나의 Row Key를 가져야 한다(PK와 유사, CF 설계에서 가장 중요)
- Column을 구성해야 한다.(Column은 항상 Name, Value, timestamps로 구성)

#### Step 2 • 예시로 Contact(주소록)을 설계한다면..

- Row Key는 cnct\_id 로 유일하게 생성

Name	Value	Timestamp
• Email	: gdhong@b2en.com	: 1320405627458000
• Address	: Gyeonggi-do	: 1320405627458000
• Name	: Gildong Hong	: 1320405627458000



#### Step 3 • Cassandra의 Contact의 구조는 다음과 같다.

[Row Key]	<b>Email</b>	<b>Address</b>	<b>Name</b>
c001	gdhong@b2en.com <i>20405627458000</i>	Gyeonggi-do <i>1320405627458000</i>	Gildong Hong <i>1320405627458000</i>
[Row Key]	<b>Email</b>	<b>Address</b>	<b>Name</b>
c002	sslee@b2en.com <i>20405627458000</i>	Seoul <i>1320405627458000</i>	Sunsin Lee <i>1320405627458000</i>
...	...	...	...

## 2. Cassandra 모델링을 배워보자

### 2.6 Column Family 생성 및 변경 테스트

#### 1) 생성

```
CREATE COLUMNFAMILY Contact (  
  cnct_id  varchar PRIMARY KEY,  
  address  varchar,  
  email    varchar,  
  name     varchar  
);
```

#### 2) 입력 ①

```
INSERT INTO Contact (cnct_id , email)  
VALUES ('c001', 'gdhong@b2en.com');
```

```
INSERT INTO Contact (cnct_id , address)  
VALUES ('c001', 'Gyeonggi-do');
```

```
INSERT INTO Contact (cnct_id , name)  
VALUES ('c001', 'Gildong Hong');
```

```
SELECT * FROM Contact ;
```

cnct_id	address	email	name
c001	Gyeonggi-do	gdhong@b2en.com	Gildong Hong

#### ① Row Key를 기준으로 데이터 입력

- 기본형 컬럼패밀리를 생성하고 cnct\_id 를 Row Key로 생성
- 데이터는 Row Key를 기준으로 입력됨

#### 3) 변경 및 삭제 ②

```
-- INSERT를 이용한 값 변경  
INSERT INTO Contact (cnct_id , name)  
VALUES ('c001', 'HI~ Gildong');
```

```
SELECT * FROM Contact ;
```

cnct_id	address	email	name
c001	Gyeonggi-do	gdhong@b2en.com	HI~ Gildong

```
-- UPDATE를 이용한 값 변경  
UPDATE Contact  
SET name = 'HongGildong'  
WHERE cnct_id = 'c001';
```

```
SELECT * FROM Contact ;
```

cnct_id	address	email	name
c001	Gyeonggi-do	gdhong@b2en.com	HongGildong

```
DELETE name FROM Contact WHERE cnct_id = 'c001';
```

```
SELECT * FROM Contact ;
```

cnct_id	address	email	name
c001	Gyeonggi-do	gdhong@b2en.com	

#### ② 데이터의 변경/삭제/조회도 Row Key로만 가능

- Row Key를 이용하여 Insert/update를 통해 값을 변경할 수 있음
- 삭제/조회도 Row Key를 이용함

1. ER 모델과 NoSQL 모델은 어떻게 다른가?
2. Cassandra 모델링을 배워보자.

## 3. ER 모델을 NoSQL 모델로 변경해보자.

4. 실제 사례를 살펴보자.
5. NoSQL 모델링은 누가해야 하나?
6. 마치면서

3.1 ER 모델을 NoSQL 모델로 변경

3.2 Key가 아닌 조건을 사용할 경우

3.3 Join이 필요할 경우

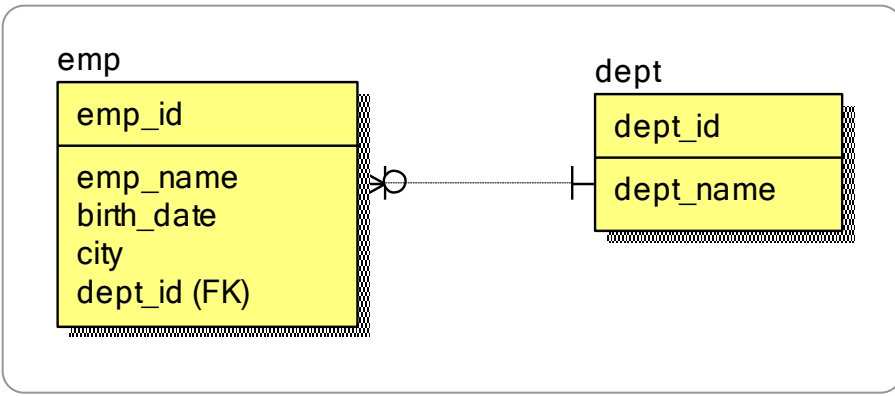
3.4 Group By를 할 경우

3.5 Advanced 모델링

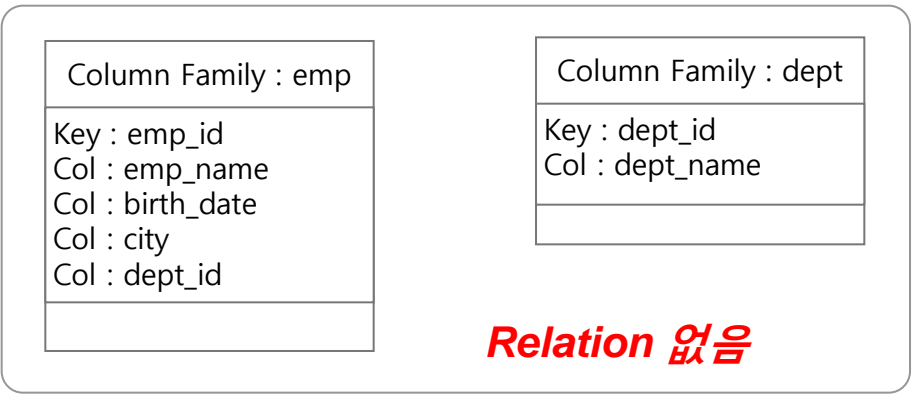
### 3. ER 모델을 NoSQL 모델로 변경해 보자.

#### 3.1 ER 모델을 NoSQL 모델로 변경

ER 모델



NoSQL 모델



Emp 테이블

emp_id	emp_name	Birth_date	City	Dept_id
5	홍길동	1970/01/01	서울	12
7	강감찬	1980/02/02	부산	13

Column Family : Emp

5	emp_name	Birth_date	City	Dept_id
	홍길동	1970/01/01	서울	12
7	emp_name	Birth_date	City	Dept_id
	강감찬	1980/02/02	부산	13

Dept 테이블

Dept_id	dept_name
12	총무부
13	영업부

Column Family : Dept

12	dept_name
	총무부
13	dept_name
	영업부

**No Join !!**

**NoSQL 모델은 Relation도 없고 Join도 없다.**



### 3. ER 모델을 NoSQL 모델로 변경해 보자.

#### 3.2 Key가 아닌 조건을 사용할 경우

```
SELECT *
FROM emp
WHERE birth_date = '1970/01/01'
```

#### Point

- ❶ Secondary Index 생성(0.8.4버전 이상)
  - Key 또는 인덱스 컬럼만 조회 가능
- ❷ 검색 컬럼을 Key로 인덱스 테이블 생성
  - birth\_date을 Row Key 구성한 컬럼페밀리 생성
  - Secondary Index보다 성능 우수함

#### ❷ Column Family : Birth\_Emp

1970/01/01	5:emp_name	5:City	5:Dept_id	33Emp_name	33:...
	홍길동	서울	12	...	...

Column Family : Emp

5	emp_name	Birth_date	City	Dept_id
	홍길동	1970/01/01	서울	12
7	emp_name	Birth_date	City	Dept_id
	강감찬	1980/02/02	부산	13
⋮				
33	emp_name	Birth_date	City	Dept_id
	이순신	1970/01/01	대전	10
⋮				

#### ❶ Secondary Index 생성

```
create index emp_idx01 on emp (birth_date);
```

```
SELECT * FROM Birth_Emp
WHERE birth_date = '1970/01/01';

emp_id | emp_name | dept_id
-----+-----+-----
5      | 홍길동   | 12
33     | 이순신   | 10
```

### 3. ER 모델을 NoSQL 모델로 변경해 보자.

#### 3.3 조인이 필요한 경우

```
SELECT *  
FROM emp e, dept d  
WHERE e.dept_id = d.dept_id
```

#### Point

- 1 Application에서 조인 처리로 성능 저하
  - 조인해야 하는 테이블 수가 많을 수록 IO수 증가
- 2 Denormalization
  - Dept\_id + emp\_id를 복합키로한 컬럼패밀리 생성
  - 한번의 쿼리를 수행하기 때문에 성능 향상

Column Family : Emp

```
select emp_id, emp_name, birth_date, city, $dept_id  
from emp  
where emp_id="사원번호" ← Query 1회 수행
```



1 Application에서 조인 처리

Column Family : Dept

```
select dept_nm  
from dept  
where dept_id=$dept_id ← Query 2회 수행
```

emp 테이블에서 dept\_id값을 읽어온 후에, dept 테이블에서 앞서 읽어온 dept\_id값을 키로 해서 다시 dept\_nm을 찾아와야 함

#### 2 Column Family : Dept\_Emp

12	5:Emp_name	5:City	5:birth_date	5:dept_name	130:...
	홍길동	서울	1970/01/01	총무부	...
13	7:Emp_name	7:City	7:birth_date	7:dept_name	51:...
	강감찬	부산	1980/02/02	영업부	...

```
SELECT * FROM Dept_Emp;  
  
dept_id | emp_id | emp_name | city | ...  
-----+-----+-----+-----+  
12 | 5 | 홍길동 | 서울 |  
12 | 130 | 유관순 | 서울 |  
... | ... | ... | ... |  
13 | 7 | 강감찬 | 부산 |  
13 | 51 | 김유신 | 대전 |
```

### 3. ER 모델을 NoSQL 모델로 변경해 보자.

#### 3.4 Group by를 할 경우

```
SELECT city, count(*)
FROM emp
GROUP BY city
```

#### Point

- ❶ 복합키를 이용한 집계
  - city + dept\_name + emp\_id로 복합키 구성
- ❷ Key의 정렬화를 이용한 설계
  - Row Key와 나머지 컬럼들의 보조정렬이 결합하여 다차원 인덱스 효과

Column Family : City\_Emp

서울	총무부 : 05	총무부 : 19	...
	-	-	-
부산	총무부 : 07	총무부 : 59	...
	-	-	-

❶ 복합키 : city + dept\_name + emp\_id로 구성

city : dept_name : emp_id	value
서울 : 총무부 : 05	---
서울 : 총무부 : 19	---
서울 : 총무부 : 22	---
서울 : 영업부 : 33	---
서울 : 영업부 : 56	---
부산 : 총무부 : 07	---
부산 : 총무부 : 59	---
부산 : 영업부 : 77	---

Diagram annotations: A blue circle labeled '2' points to the first four rows. A blue arrow labeled '서울 거주 총무부 직원' points to the first two rows. Another blue arrow labeled '서울 거주 직원' points to the first four rows.

```
SELECT count(*) FROM City_Emp
WHERE city = '서울';

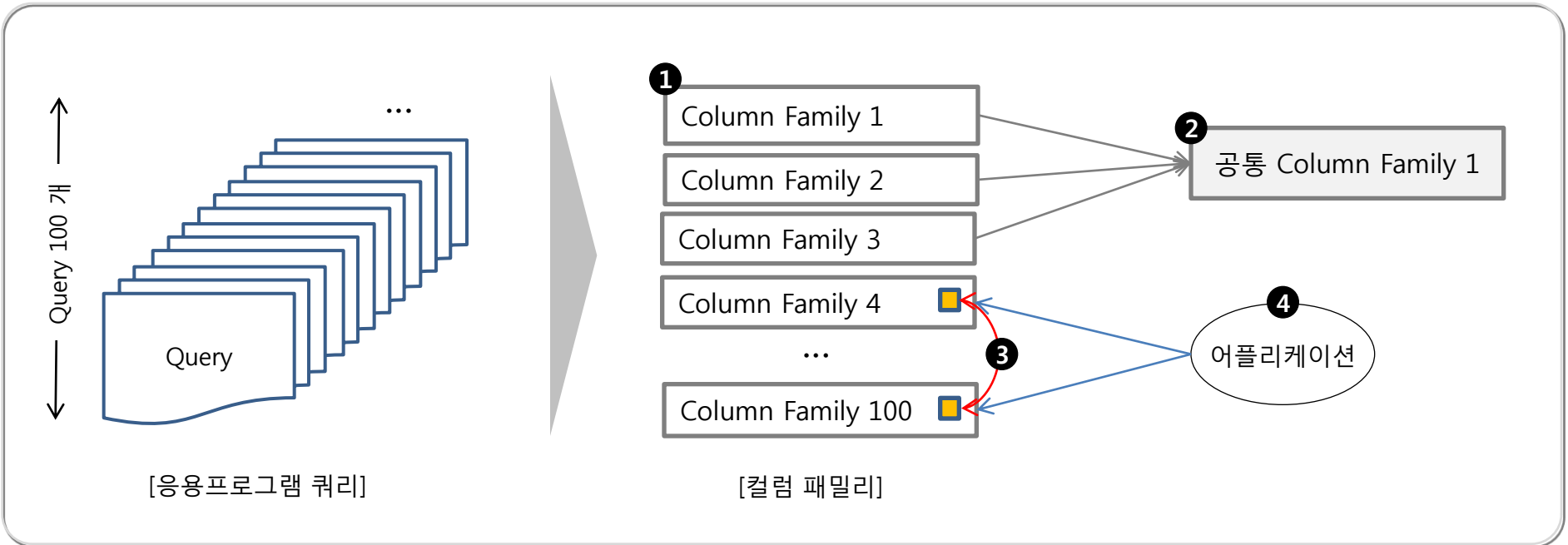
count
-----
5

SELECT count(*) FROM City_Emp
WHERE city = '서울' AND dept_name = '총무부';

count
-----
3
```

### 3. ER 모델을 NoSQL 모델로 변경해 보자.

#### 3.5 Advanced 모델링



**문제점** 1 스토리지 용량 증가  
- 데이터를 중복해서 저장하는 만큼 스토리지 용량이 증가 발생

**문제점** 3 데이터 불일치 발생  
- 동일한 값에 대해 중복해서 저장하다 보면 컬럼패밀리간에 데이터 불일치 발생

**해결책** 2 공통 쿼리 패스의 컬럼패밀리 설계  
- 응용프로그램에서 공통 쿼리 패스를 생각하고, 쿼리 패스에 맞는 컬럼 패밀리 생성 필요

**해결책** 4 시그널을 통해 값 일치 작업 수행  
- 일관성을 어플리케이션을 통해 보장함  
- signal, de-signal, signal again(ebay 사례)

1. ER 모델과 NoSQL 모델은 어떻게 다른가?
2. Cassandra 모델링을 배워보자.
3. ER 모델을 NoSQL 모델로 변경해보자.

## 4. 실제 사례를 살펴보자.

5. NoSQL 모델링은 누가해야 하나?
6. 마치면서

4.1 국내사례 : C메신저

4.2 해외사례 : 이베이(eBay)

## 4. 실제 사례를 살펴보자.

### 4.1 국내사례 : C메신저

#### 메시지박스(SCF) ❶

Row Key (사용자ID)	SuperColumn : [timestamp]   [송신자] ❶					...
	메시지ID	수신자	송신자	채팅방ID	timestamp	
U00000001	-	-	-	-	-	
....						

❶ Super Column Family 구조로 되어 있으며 성능상의 이유로 Composite Key 변경할 계획임

❷ Super Column은 메시지가 도착한 시간 | 송신자로 배열하게 설계된 구조임  
사용자별로 최근 도착한 메시지에 보낸사람 별로 보여줌

#### Cassandra를 선택한 이유

- 다수의 분산된 서버에서 확장하기 쉽기 때문(확장성)
- 빠른 Write 성능을 기반으로 대량의 데이터를 다루는데 효과적임

#### Cassandra를 활용한 영역

- 메신저 친구와 대화 메시지 저장
- 대화 메시지를 제외한 기존 정보성은 MySQL에 ER모델로 관리

#### 환경 및 특징

- 기존의 슈퍼컬럼패밀리로 설계된 구조를 Composite Key 구조로 변경하고자 함
- NoSQL에 대한 모델링은 개발자가 수행함

## 4. 실제 사례를 살펴보자.

### 4.2 해외사례 : 이베이(eBay)

ItemLike ①

itemid1	userid1	userid2	userid3	...
	timeuuid1	timeuuid2	timeuuid3	
....				

- ① 아이템에 대해 사용자들이 마음에 들었는지 저장 (일반형 CF로 설계)

UserLike ②

userid1	Timeuid1 itemid1	Timeuid2 itemid2	...
	-null-	-null-	
....			

- ② 사용자가 마음에 들어 한 아이템을 시간 순으로 저장 (Composite Key 로 설계)

#### Cassandra를 선택한 이유

- Multi-datacenter (active-active)
- 가용성, 확장성
- 쓰기 성능
- 하둡 지원

#### Cassandra를 활용한 영역

- eBay 제품 및 항목에 대한 선호
- 사용자 및 상품에 대한 직관적 그래프
- 많은 시계열 사용 사례
  - 모바일 알림 로깅 및 추적
  - 사기 탐지에 대한 추적 등등

#### 환경 및 특징

- 이베이에서는 MongoDB, Hbase, Cassandra를 두루 사용하고 있으며 솔루션 별로 적합한 업무에 사용
- 어플리케이션에서 Signal을 통해 일관성을 유지하게 설계

출처 : *Cassandra at eBay - Jay Patel*

1. ER 모델과 NoSQL 모델은 어떻게 다른가?
2. Cassandra 모델링을 배워보자.
3. ER 모델을 NoSQL 모델로 변경해보자.
4. 실제 사례를 살펴보자.

## 5. NoSQL 모델링은 누가해야 하나?

6. 마치면서

### 5.1 불편한 진실

### 5.2 Big Data 전문가

### 5.3 NoSQL 데이터베이스의 변화

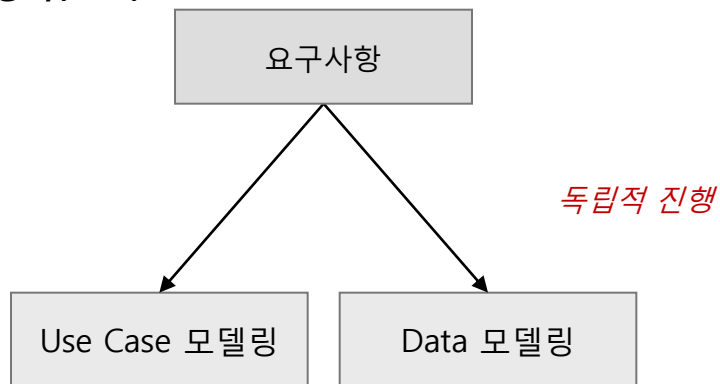
### 5.4 우리가 준비해야 할 역량



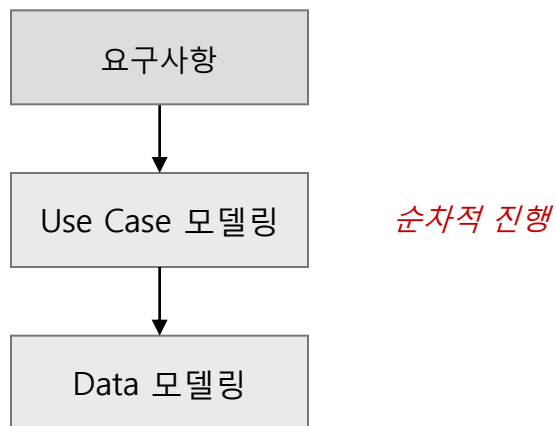
## 5. NoSQL 모델링은 누가해야 하나?

### 5.1 불편한 진실

#### 1 전통적인 방식(RDB)



#### 2 NoSQL 방식



#### 1 전통적인 방식

- 비즈니스 요구사항을 도출하고 시스템이 제공할 기능관점의 Use Case 모델링과 저장구조적 측면에서 Data 모델링을 독립적으로 병행해서 진행

#### 2 NoSQL 방식

- 비즈니스 질의사항을 이해하는 Use Case Specific 관점의 모델링을 수행한 후 Data 모델링을 수행

#### 카산드라 완벽 가이드

데이터 모델링을 먼저하고 쿼리를 작성하는 것이 아니라 쿼리를 모델링한 다음에 쿼리에 맞춰 데이터를 조직화한다는 것이 핵심이다.


- Eben Hewitt -

**그렇다면 NoSQL Data 모델링은 개발자가?**

# 5. NoSQL 모델링은 누가해야 하나?

## 5.2 Big Data 전문가

출처 : 빅데이터의 기술영역과 요구역량

역할 구분	수행 내용	요구지식
현장 전문가	<ul style="list-style-type: none"> <li>현장 분야의 고객 요구사항을 잘 이해하고 있고, <u>고객 입장에서 실제 업무를 수행</u></li> </ul>	<ul style="list-style-type: none"> <li>Visualization</li> <li>Infograph</li> <li>IR &amp; RecSys</li> </ul>
 데이터 분석가	<ul style="list-style-type: none"> <li>현장 전문가로부터 요구사항을 수집하고 이들로부터 <u>요건을 도출하여 데이터 분석과 개발을 위한 상세 설계 수행</u></li> </ul>	<ul style="list-style-type: none"> <li>OLAP Tools</li> <li>SAS</li> <li><b>SQL</b></li> <li><b>RDBMS</b></li> <li>NoSQL</li> <li>Script Language</li> <li>Pig, Hive</li> <li>MapReduce</li> </ul>
개발자	<ul style="list-style-type: none"> <li>데이터 분석가로부터 전달받은 설계 문서에 기반하여 <u>분석 시스템을 구현</u></li> </ul>	<ul style="list-style-type: none"> <li>Log Aggregator</li> <li>NoSQL</li> <li>Hadoop</li> <li>Linux</li> <li>X86</li> <li>Network</li> </ul>
인프라 담당자	<ul style="list-style-type: none"> <li>데이터 분석가와 개발자가 필요로 하는 <u>분석용 데이터를 준비하고 이를 위한 대용량 빅데이터 인프라를 운영</u></li> </ul>	

← Data Scientist<sup>주1)</sup> →

← DevOps →

**빅 데이터에서도 각 영역별 전문분야가 있고 개발자가 모델링을 하는 것은 과도기적인 형태이며 NoSQL 모델링은 데이터 분석가의 역할이다.**

주1) Data Scientist는 다양한 전문 지식을 바탕으로 비즈니스 통찰력을 제시해 주고 의사 결정을 지원하는 전문가

## 5. NoSQL 모델링은 누가해야 하나?

### 5.3 NoSQL 데이터베이스의 변화 – 기존을 인력 활용하려고 한다.

#### Get / Set 문

```
set users['jbellis']['name'] = 'Jonathan Ellis';
set users['jbellis']['birth_year'] = 1976;
set users['jbellis']['phone_number'] = long(1112223333);

get users['jbellis'];
=> (cell=birth_year, value=1976, timestamp=1370470203739000)
=> (cell=name, value=Jonathan Ellis, timestamp=1370470203736000)
=> (cell=phone_number, value=00000000424b2e65,
    timestamp=1370470203743000)
```

#### CQL(Cassandra Query Language)

```
INSERT INTO users (user_id, name, birth_year, phone_numbers)
VALUES ('jbellis', 'Jonathan Ellis', 1976, 1112223333);

SELECT * FROM users;
user_id | birth_year | name           | phone_number
-----+-----+-----+-----
jbellis |      1976 | Jonathan Ellis | 1112223333
```

#### 첫째, 카산드라 CQL 지원

1. 사람들에게 더 많은 관심을 얻기 위해
  - NoSQL에 관심 있는 자들을 자극하기 위해
  - SQL에 관심 있는 자들도 자극하기 위해
2. 가독성, 유지보수성 우수
  - (이미알고 있어) 사용하기 용의함
  - 검증됐으며 간단하고 직관적임

출처 : CQL : SQL for Cassandra - Eric Evans

#### 둘째, 맵리듀스 대신 SQL로 빅데이터 처리

1. 오픈소스 진영 SQL 엔진 지속적 출시
  - 클라우데라의 <임팔라>, 하이브의 <하이브QL>
2. 빅데이터에서 표준 SQL의 활용
  - 고급 하둡 개발자 없이 표준 SQL을 통해 하둡 기술 활용 가능

출처 : 디지털데일리(2013.06.04)

## 5. NoSQL 모델링은 누가해야 하나?

### 5.4 우리가 준비해야 할 역량

#### I. 쿼리와 리포팅은 기본이다.

- 빅 데이터 분석역량 중 91%가 가장 기본이 되는 기술인 쿼리와 리포팅을 선택
- 쿼리와 리포팅은 RDB에서도 중요하고 우리가 계속 연마했던 기술임

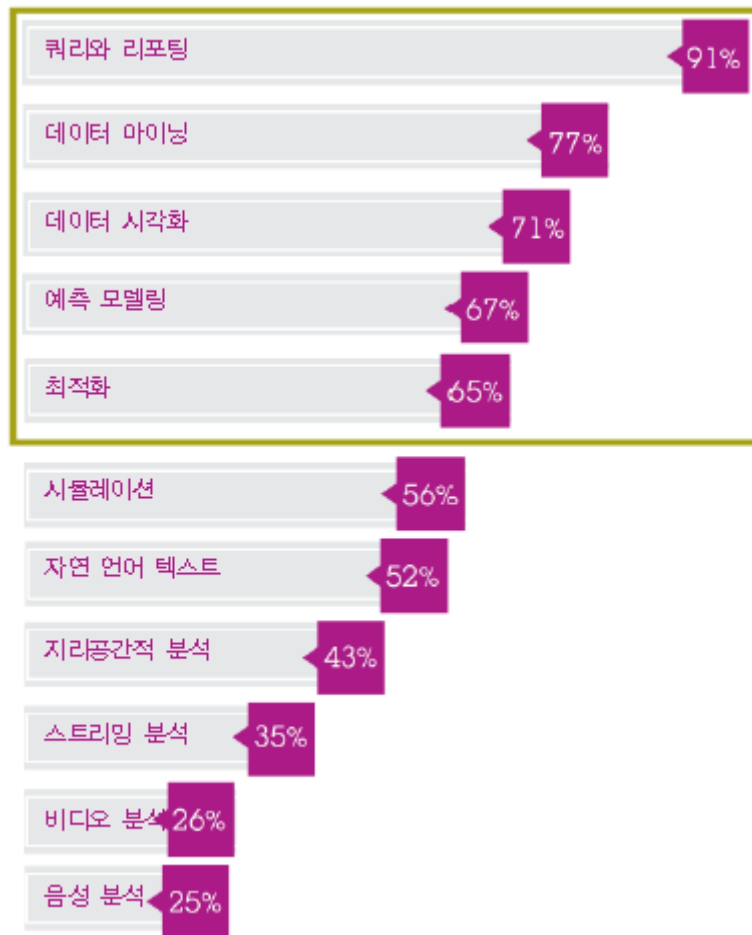
#### II. 새로운 패러다임은 하루 아침에 변하지 않는다.

- 쿼리/리포팅을 비롯해 데이터 마이닝, 데이터 시각화, 최적화 등 과거에도 존재했던 기술임

우리의 현재 가지고 있는 기술을 바탕으로  
빅데이터 기술을 해석하여 내 것으로 만들면

**결론)**  
**걱정하지 마라! 우리는 누구보다 잘 할 수 있다.**

#### 빅 데이터 분석 역량



출처 : 빅데이터를 활용 중인 기업 구성원 대상 설문 자료  
(IBM 2012)

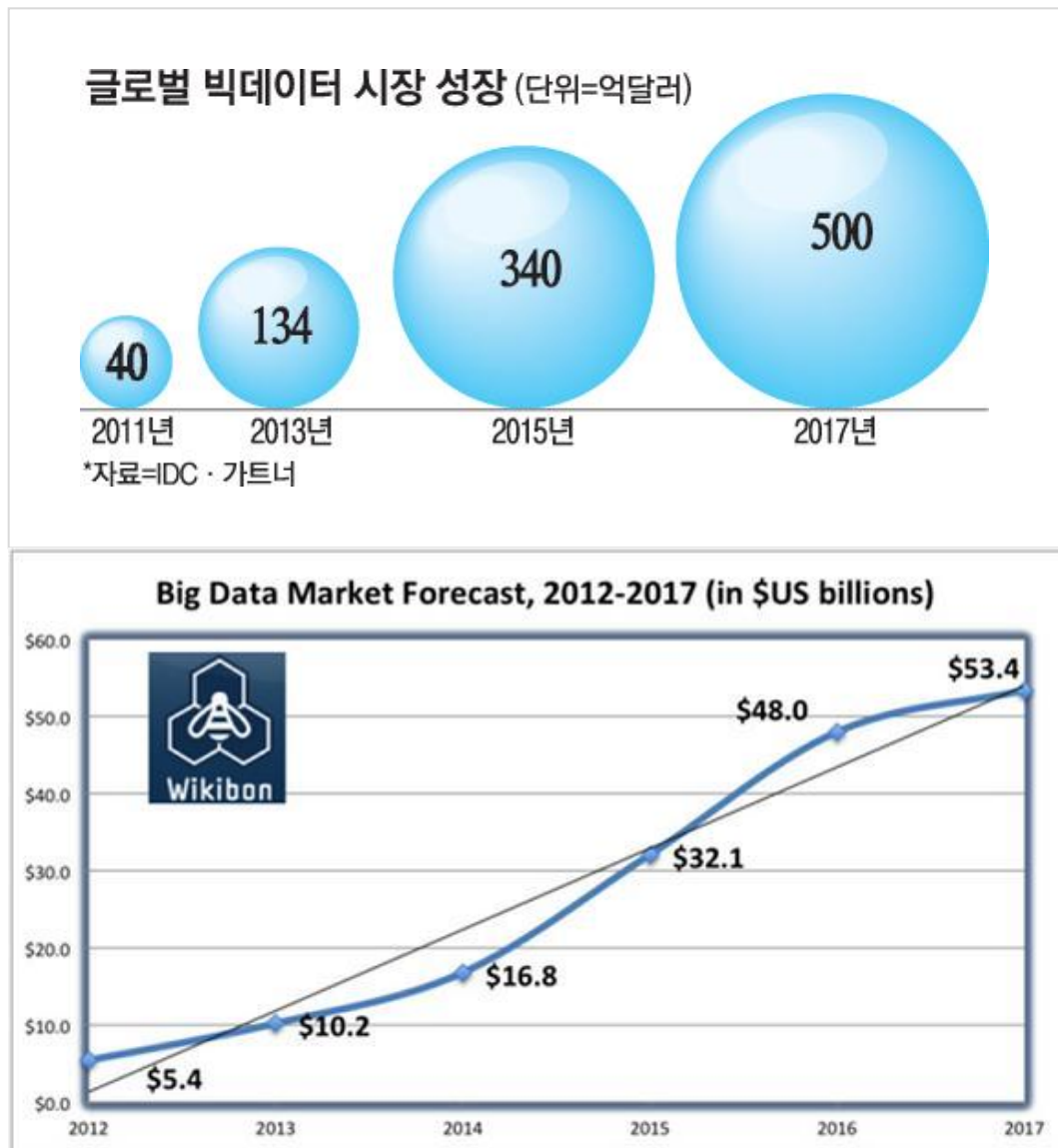
## 6. 마치면서

### 1. 여러 자료를 통해 확인 할 수 있듯이 앞으로 빅데이터 시장은 성장할 것이다.

- RDB시장은 견고한 시장이고 빅데이터가 대체할 시장은 아니다.
- 그러나, 전체 투자비가 한정되어 있고 지속적으로 성장한다면 RDB에게도 위협적인 시장이다.

### 2. 빅데이터는 보험이다.

- 현재 당장 하지 않아도 어려움은 없지만 변화하는 미래를 위해 준비해야 할 기술이다.



---

- Thank you -