

데이터 모델링

관계형 데이터 모델링

관계형 데이터베이스

- 사전 정의된 데이터 항목들의 모음
- 열과 행으로 이루어진 테이블 집합
- 엔티티 간의 관계를 표현

데이터 모델링

model? : 객체, 시스템, 개념에 대한 구조나 작업 또는 현상에 대한 설명을 보여주기 위한 패턴, 계획, 또는 설명

Data model? : 데이터의 관계, 접근과 그 흐름에 필요한 처리 과정에 관한 추상화된 모델

Data modeling? : 개념을 논리적&물리적 데이터 모델로 구성

관계형 데이터 모델링은?

개념을 관계형 데이터베이스에 맞도록
논리적 물리적 데이터 모델로 모델링

관계형 데이터 모델링의 순서

1. 문제정의 및 도메인 파악
2. 개념적 데이터 모델링(ERD)
3. 논리적 데이터 모델링(data scheme)
4. 물리적 데이터 모델링(시스템 / DBMS 최적화)

문제정의 및 도메인 파악

- 어떤 문제를 해결하기 위해 데이터를 이용하는가?
- 문제 해결 형태는 무엇인가?
- 어떻게 데이터를 생성할 것인가?
- 데이터 간의 관계는 어떻게 되는가?
- 데이터는 어떻게 변화하는가?

한번 예시를 들어볼까요?

- 어떤 문제를 해결하기 위해

SNS상에서 자사에 관련된 게시글과 댓글을 통해 브랜드 영향도를 분석
자사의 브랜딩과 마케팅 전략에 이용

- 문제 해결 형태는 무엇인가?

게시글과 댓글을 긍부정에 대한 구분과 0~100사이의 스코어링
각 게시글과 댓글에 대한 분석결과를 통계분석

- 어떻게 데이터를 생성할 것인가?

SNS search 기능을 활용하여 자사에 관련된 키워드로 search하여
해당 검색결과를 크롤링한다.

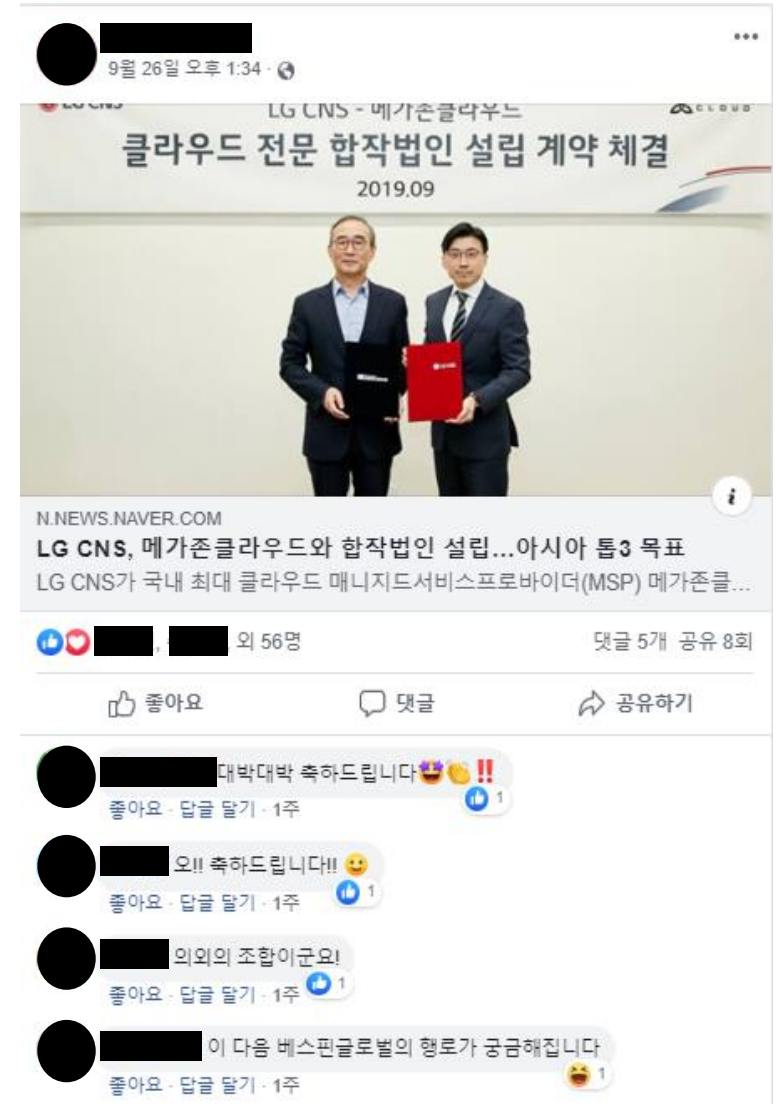
- 데이터 간의 관계는 어떻게 되는가?

한 게시글에 여러 개의 댓글이 달리고 댓글을 하나의 게시글에 달린다.(1:N)
게시글과 댓글은 sns유저가 작성한다.(생성관계)

Sns유저끼리는 서로 여러 명의 친구를 맺는다.(N:N / 네트워크)

- 데이터는 어떻게 변화하는가?

각 게시글과 댓글은 분석을 위해 토큰화 백터화 되어 처리된다.



9월 26일 오후 1:34 · 🌐

LG CNS - 메가존클라우드

클라우드 전문 합작법인 설립 계약 체결

2019.09

N.NEWS.NAVER.COM

LG CNS, 메가존클라우드와 합작법인 설립...아시아 톱3 목표

LG CNS가 국내 최대 클라우드 매니지드서비스프로바이더(MSP) 메가존클...

👍❤️ [redacted] 외 56명

댓글 5개 공유 8회

👍 좋아요

💬 댓글

🔗 공유하기

대박대박 축하드립니다 🎉🥳!!

좋아요 · 답글 달기 · 1주

오!! 축하드립니다!! 😊

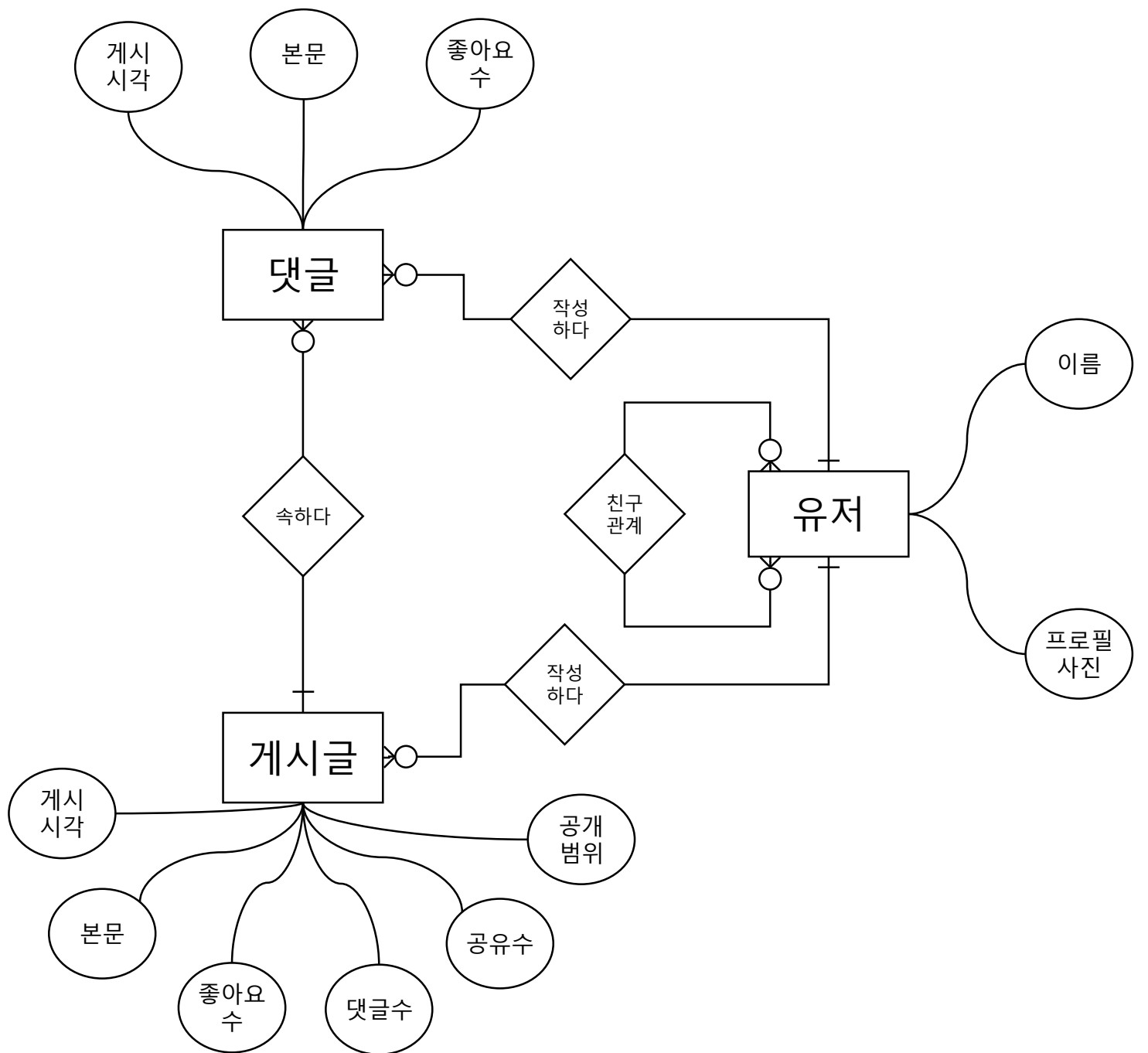
좋아요 · 답글 달기 · 1주

의외의 조합이군요!

좋아요 · 답글 달기 · 1주

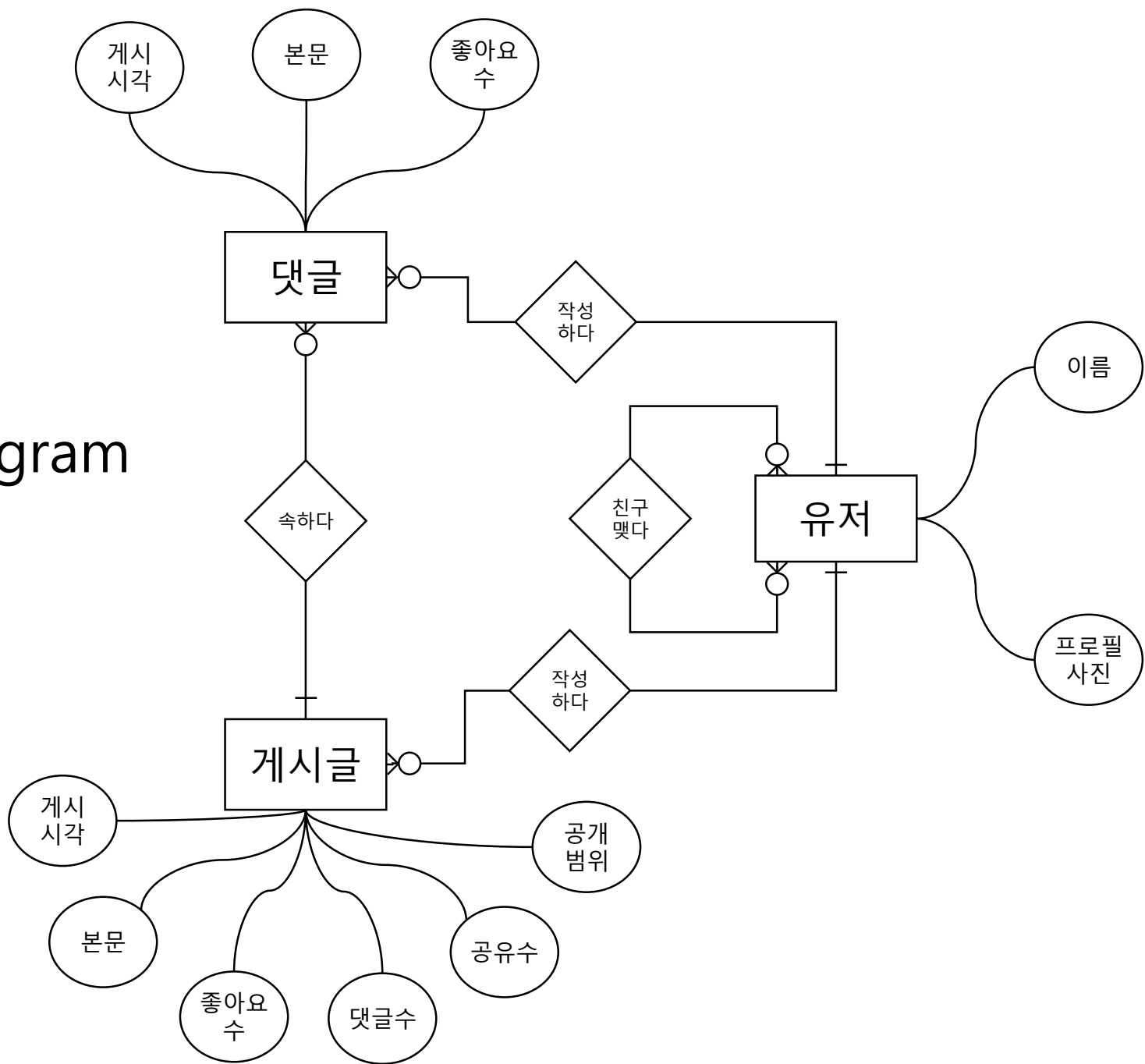
이 다음 베스핀글로벌의 행로가 궁금해집니다 🤔

좋아요 · 답글 달기 · 1주



ERD : Entity Relationship Diagram

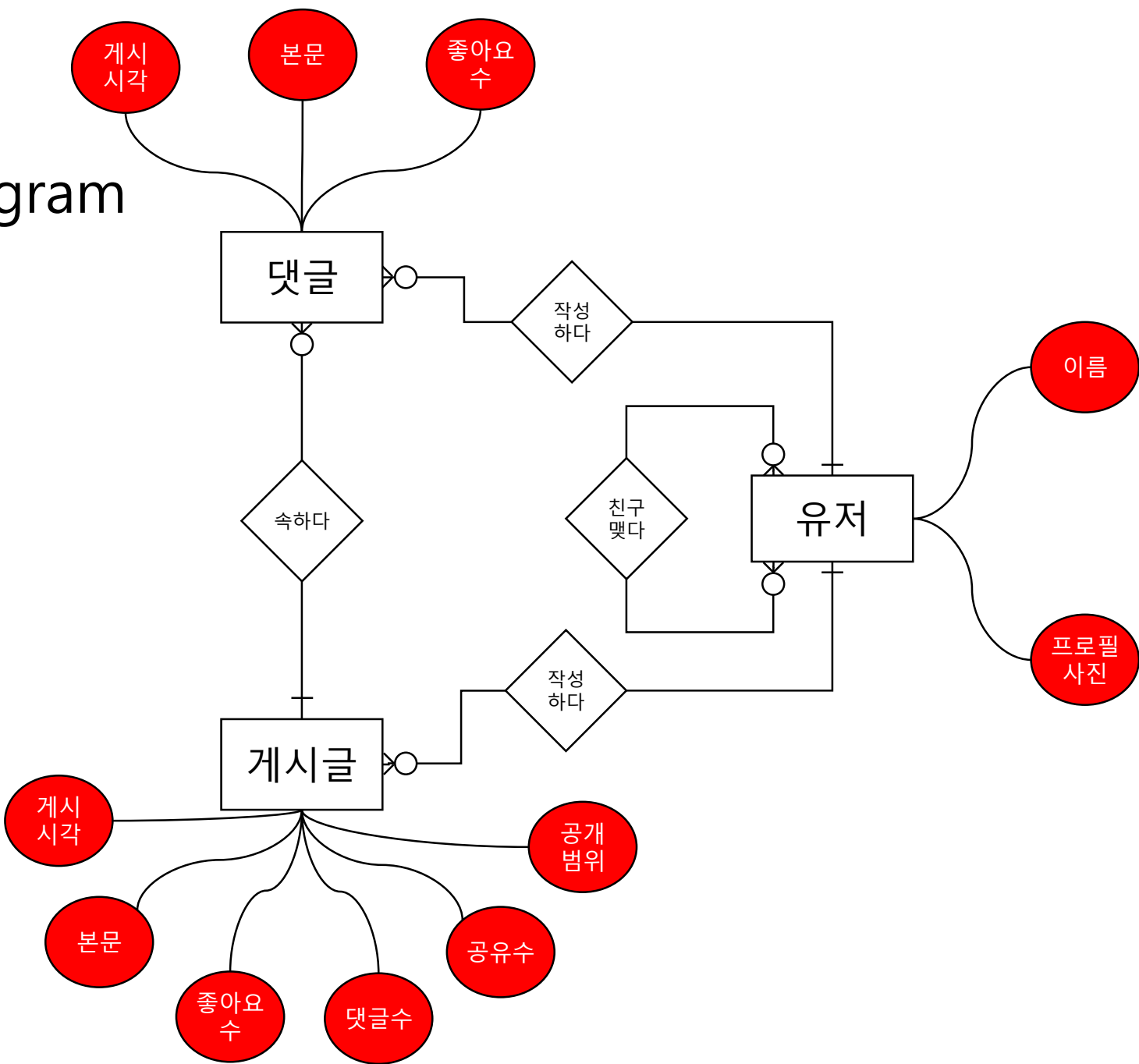
엔티티 간의 관계를 그림으로 표현



ERD : Entity Relationship Diagram

엔티티 간의 관계를 그림으로 표현

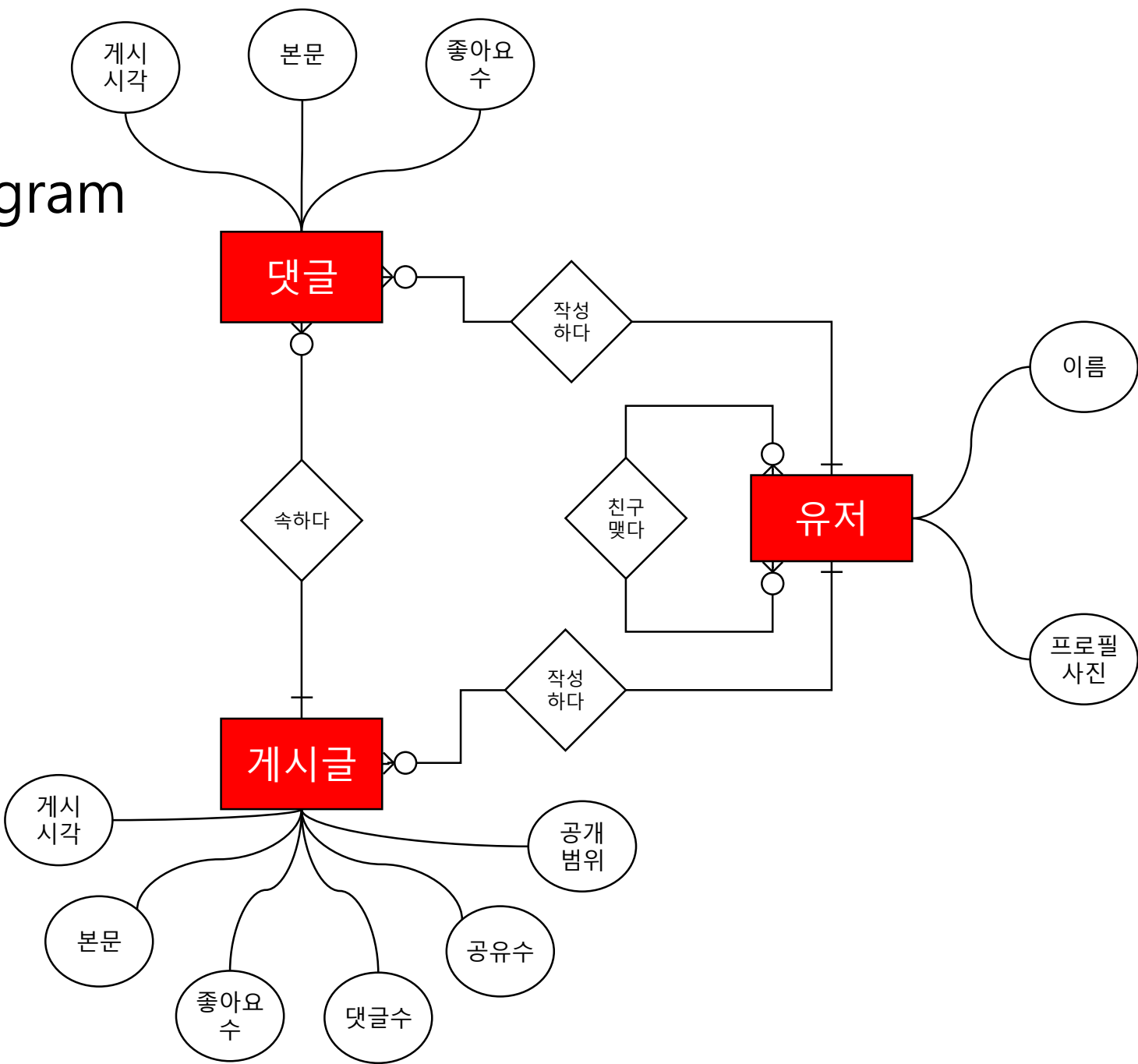
정보



ERD : Entity Relationship Diagram

엔티티 간의 관계를 그림으로 표현

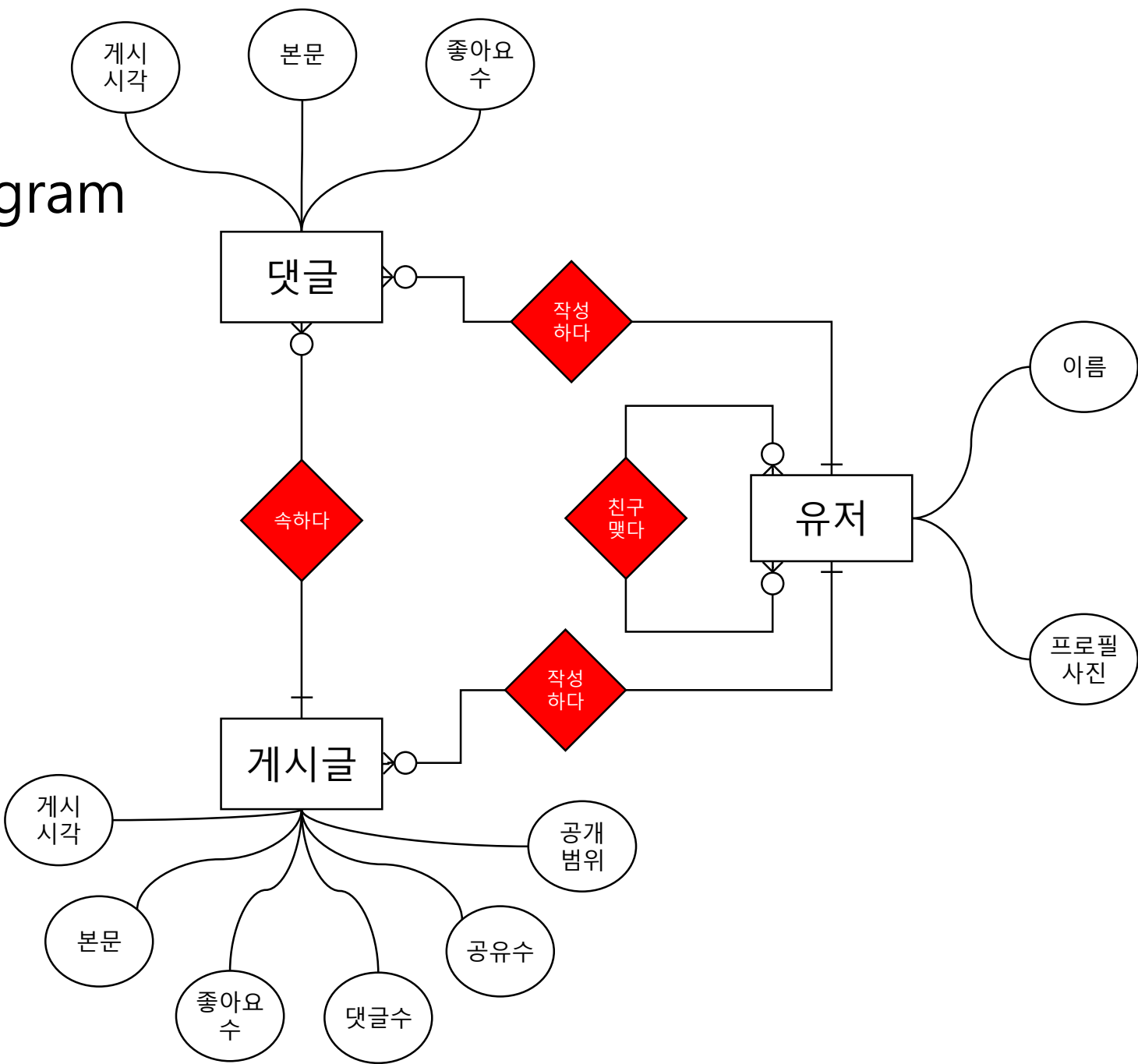
정보그룹



ERD : Entity Relationship Diagram

엔티티 간의 관계를 그림으로 표현

정보그룹관계



유저

9월 26일 오후 1:34

본문

게시시간

공개범위

N.NEWS.NAVER.COM

LG CNS, 메가존클라우드와 합작법인 설립...아시아 톱3 목표

LG CNS가 국내 최대 클라우드 매니지드서비스프로바이더(MSP) 메가존클...

외 56명

댓글 5개

공유 8회

좋아요 수

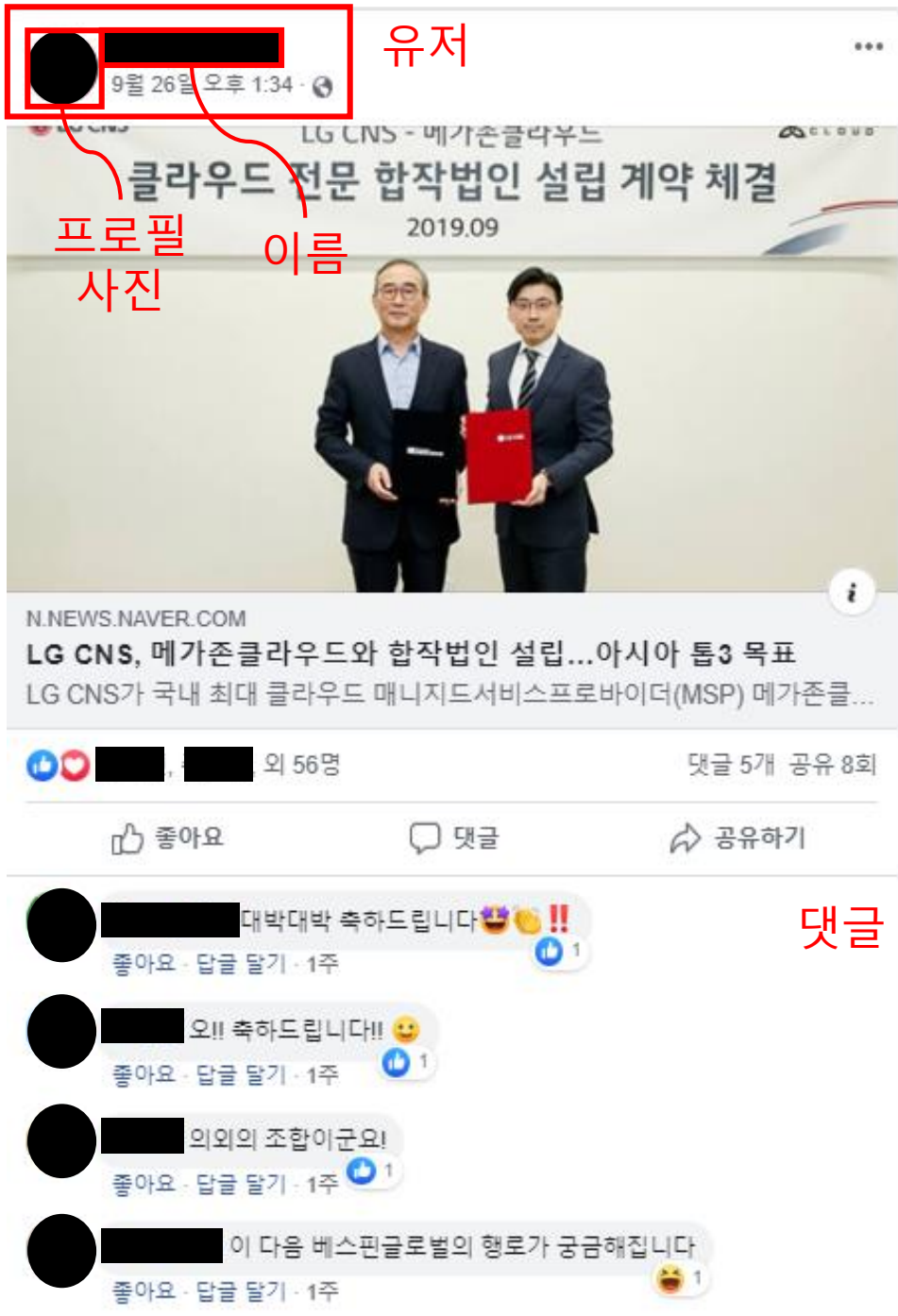
댓글 수

공유 수

댓글


게시글

유저	게시시간	공개범위
본문		
좋아요 수	댓글 수	공유 수
댓글		



9월 26일 오후 1:34 · 🌐


LG CNS - 메가존클라우드
클라우드 전문 합작법인 설립 계약 체결
2019.09



N.NEWS.NAVER.COM
LG CNS, 메가존클라우드와 합작법인 설립...아시아 톱3 목표
LG CNS가 국내 최대 클라우드 매니지드서비스프로바이더(MSP) 메가존클...

👍❤️ [redacted] 외 56명 💬 5개 ➦ 8회

👍 좋아요 💬 댓글 ➦ 공유하기



댓글 달기

1주

대박대박 축하드립니다 🎉👏!!

1

👍

오!! 축하드립니다!! 🙌


의외의 조합이군요!

이 다음 베스핀글로벌의 행로가 궁금해집니다



9월 26일 오후 1:34 · 🌐

LG CNS - 메가존클라우드
클라우드 전문 합작법인 설립 계약 체결
2019.09



N.NEWS.NAVER.COM
LG CNS, 메가존클라우드와 합작법인 설립...아시아 톱3 목표
LG CNS가 국내 최대 클라우드 매니지드서비스프로바이더(MSP) 메가존클...

👍❤️ [redacted] 외 56명 💬 댓글 5개 ➦ 공유 8회

👍 좋아요 💬 댓글 ➦ 공유하기

👤 [redacted] 🎉 대박대박 축하드립니다 🥳🥳!! 👍 1

좋아요 · 댓글 달기 · 1주

👤 [redacted] 😄 오!! 축하드립니다!! 👍 1

좋아요 · 댓글 달기 · 1주

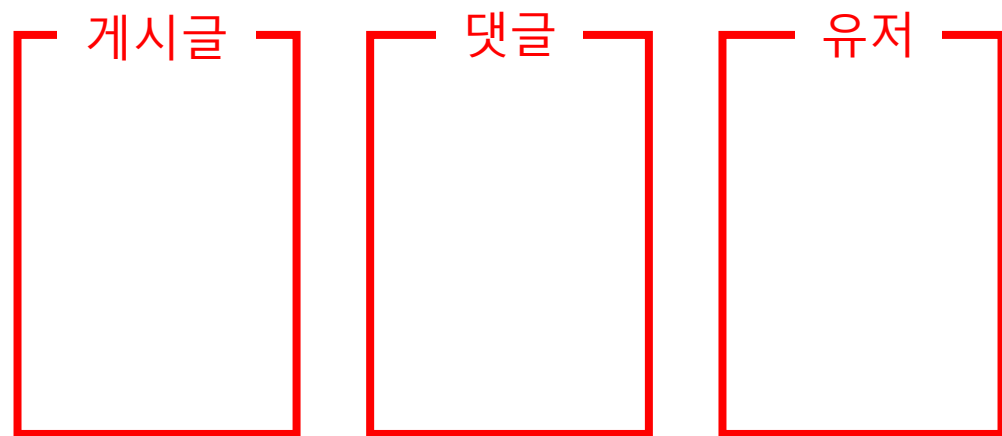
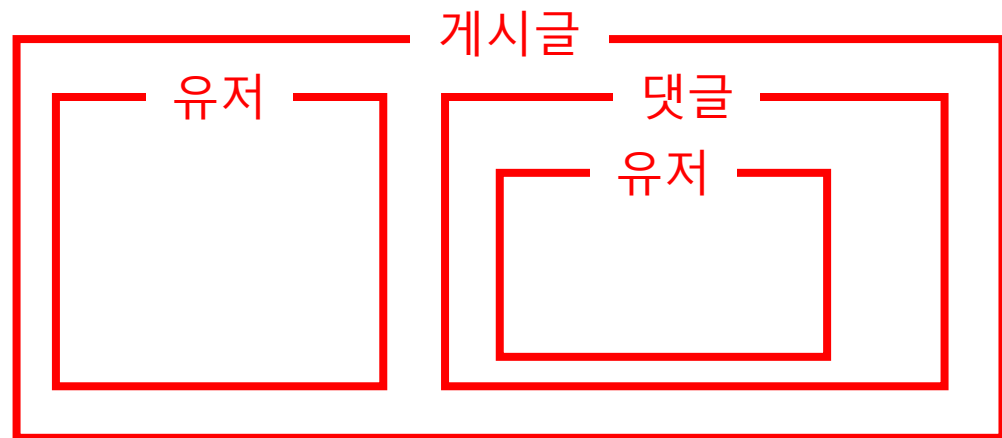
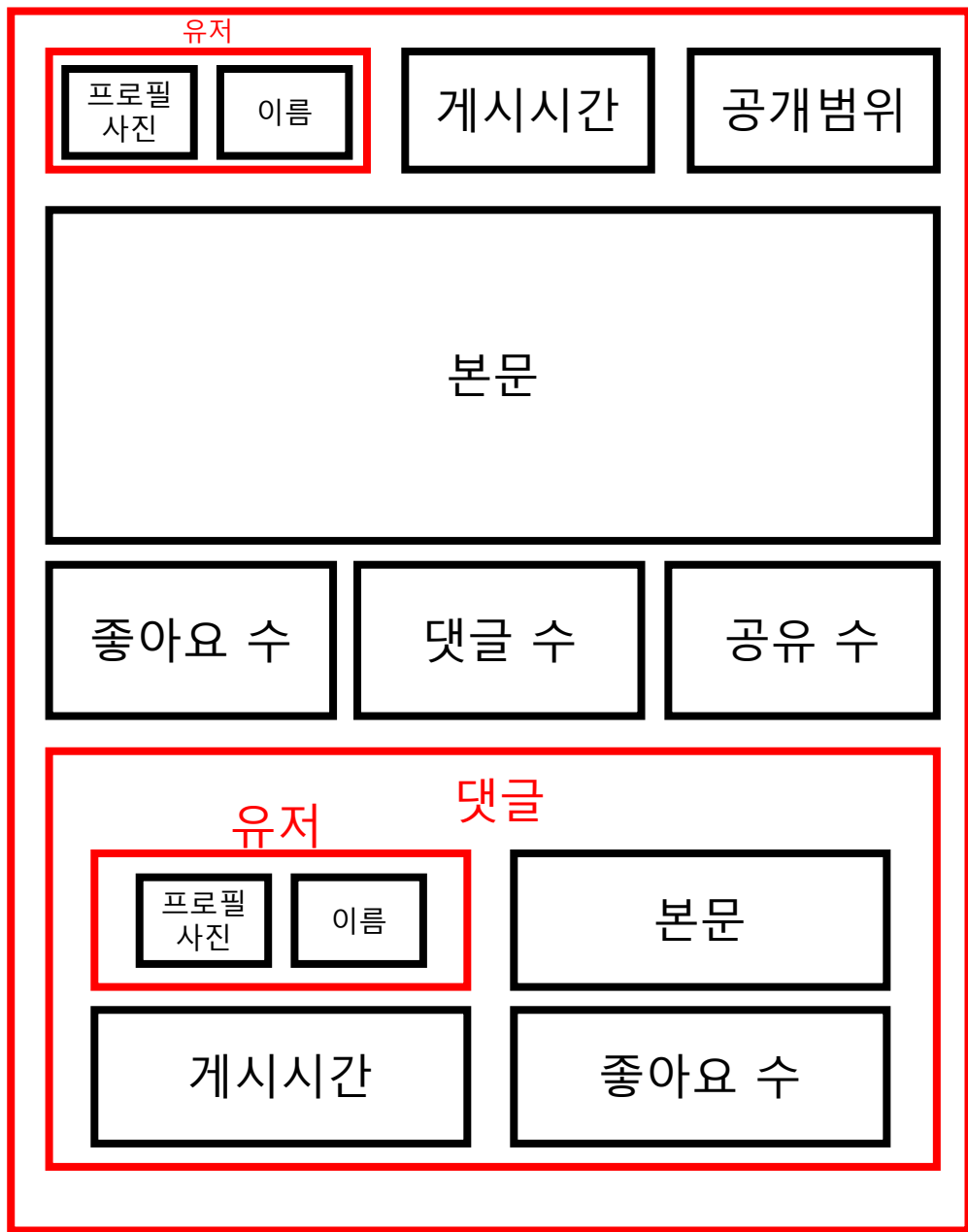
👤 [redacted] 🙌 의외의 조합이군요! 👍 1

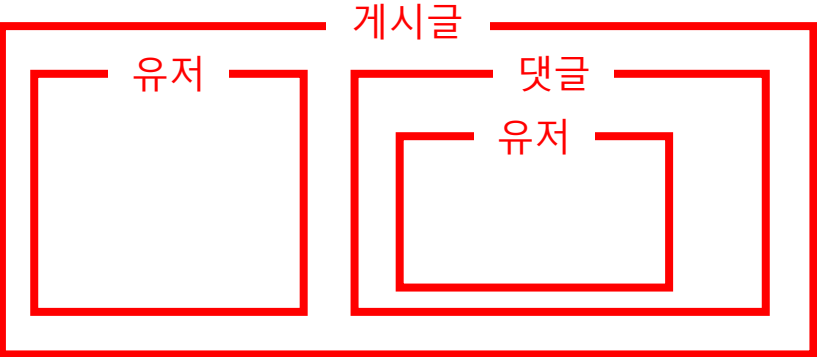
좋아요 · 댓글 달기 · 1주

👤 [redacted] 🤔 이 다음 베스핀글로벌의 행로가 궁금해집니다 😄 1

좋아요 · 댓글 달기 · 1주







게시 시간	본문	좋아요 수	댓글 수	공유 수	공개 범위	게시자(유저)		댓글				
						프로필 사진	이름	게시 시간	본문	좋아요 수	게시자(유저)	
											프로필 사진	이름

게시 시간	본문	좋아요 수	댓글 수	공유 수	공개 범위	게시자 프로필사진	게시자 이름	댓글 게시시간	댓글 본문	댓글 좋아요 수	댓글 게시자 프로필사진	댓글 게시자 이름
글1 시간	본문1	3	2	1	전체	유저1 이미지	유저1 이름	댓글1 시간	댓글1	2	유저2 이미지	유저2 이름
글1 시간	본문1	3	2	1	전체	유저1 이미지	유저1 이름	댓글2 시간	댓글2	1	유저3 이미지	유저3 이름
글2 시간	본문2	1	1	5	전체	유저3 이미지	유저3 이름	댓글3 시간	댓글3	2	유저1 이미지	유저1 이름
글3 시간	본문3	1	0	0	전체	유저2 이미지	유저2 이름					

게시 시간	본문	좋아요 수	댓글 수	공유 수	공개 범위	게시자 프로필사진	게시자 이름	댓글 게시시간	댓글 본문	댓글 좋아요 수	댓글 게시자 프로필사진	댓글 게시자 이름
글1 시간	본문1	3	2	1	전체	유저1 이미지	유저1 이름	댓글1 시간	댓글1	2	유저2 이미지	유저2 이름
글1 시간	본문1	3	2	1	전체	유저1 이미지	유저1 이름	댓글2 시간	댓글2	1	유저3 이미지	유저3 이름
글2 시간	본문2	1	1	5	전체	유저3 이미지	유저3 이름	댓글3 시간	댓글3	2	유저1 이미지	유저1 이름
글3 시간	본문3	1	0	0	전체	유저2 이미지	유저2 이름					

게시글

아이디	게시 시간	본문	좋아요 수	댓글 수	공유 수	공개 범위	게시자
t_1	글1 시간	본문1	3	2	1	전체	u_1
t_2	글2 시간	본문2	1	1	5	전체	u_3
t_3	글3 시간	본문3	1	0	0	전체	u_2

유저

아이디	프로필사진	이름
u_1	유저1 이미지	유저1 이름
u_2	유저2 이미지	유저2 이름
u_3	유저3 이미지	유저3 이름

댓글

아이디	게시 시간	본문	좋아요 수	게시자	게시글
c_1	댓글1 시간	댓글1	2	u_2	t_1
c_2	댓글2 시간	댓글2	1	u_3	t_1
c_3	댓글3 시간	댓글3	2	u_1	t_2

게시 시간	본문	좋아요 수	댓글 수	공유 수	공개 범위	게시자 프로필사진	게시자 이름	댓글 게시시간	댓글 본문	댓글 좋아요 수	댓글 게시자 프로필사진	댓글 게시자 이름
글1 시간	본문1	3	2	1	전체	유저1 이미지	유저1 이름	댓글1 시간	댓글1	2	유저2 이미지	유저2 이름
글1 시간	본문1	3	2	1	전체	유저1 이미지	유저1 이름	댓글2 시간	댓글2	1	유저3 이미지	유저3 이름
글2 시간	본문2	1	1	5	전체	유저3 이미지	유저3 이름	댓글3 시간	댓글3	2	유저1 이미지	유저1 이름
글3 시간	본문3	1	0	0	전체	유저2 이미지	유저2 이름					

게시글

유저

댓글

아이디	게시 시간	본문	좋아요 수	댓글 수	공유 수	공개 범위	게시자	아이디	프로필사진	이름	아이디	게시 시간	본문	좋아요 수	게시자	게시글
t_1	글1 시간	본문1	3	2	1	전체	u_1	u_1	유저1 이미지	유저1 이름	c_1	댓글1 시간	댓글1	2	u_2	t_1
t_2	글2 시간	본문2	1	1	5	전체	u_3	u_2	유저2 이미지	유저2 이름	c_2	댓글2 시간	댓글2	1	u_3	t_1
t_3	글3 시간	본문3	1	0	0	전체	u_2	u_3	유저3 이미지	유저3 이름	c_3	댓글3 시간	댓글3	2	u_1	t_2

SELECT * FROM 게시글 LEFT JOIN 댓글 ON 게시글.아이디 = 댓글.게시글

아이디	게시 시간	본문	좋아요 수	댓글 수	공유 수	공개 범위	댓글 아이디	댓글 게시시간	댓글 본문	댓글 좋아요 수	댓글 게시자 프로필사진	댓글 게시자 이름
t_1	글1 시간	본문1	3	2	1	전체	c_1	댓글1 시간	댓글1	2	유저2 이미지	유저2 이름
t_1	글1 시간	본문1	3	2	1	전체	c_2	댓글2 시간	댓글2	1	유저3 이미지	유저3 이름
t_2	글2 시간	본문2	1	1	5	전체	c_3	댓글3 시간	댓글3	2	유저1 이미지	유저1 이름
t_3	글3 시간	본문3	1	0	0	전체						

게시 시간	본문	좋아요 수	댓글 수	공유 수	공개 범위	게시자 프로필사진	게시자 이름	댓글 게시시간	댓글 본문	댓글 좋아요 수	댓글 게시자 프로필사진	댓글 게시자 이름
글1 시간	본문1	3	2	1	전체	유저1 이미지	유저1 이름	댓글1 시간	댓글1	2	유저2 이미지	유저2 이름
글1 시간	본문1	3	2	1	전체	유저1 이미지	유저1 이름	댓글2 시간	댓글2	1	유저3 이미지	유저3 이름
글2 시간	본문2	1	1	5	전체	유저3 이미지	유저3 이름	댓글3 시간	댓글3	2	유저1 이미지	유저1 이름
글3 시간	본문3	1	0	0	전체	유저2 이미지	유저2 이름					

게시글

유저

댓글

아이디	게시 시간	본문	좋아요 수	댓글 수	공유 수	공개 범위	게시자	아이디	프로필사진	이름	아이디	게시 시간	본문	좋아요 수	게시자	게시글
t_1	글1 시간	본문1	3	2	1	전체	u_1	u_1	유저1 이미지	유저1 이름	c_1	댓글1 시간	댓글1	2	u_2	t_1
t_2	글2 시간	본문2	1	1	5	전체	u_3	u_2	유저2 이미지	유저2 이름	c_2	댓글2 시간	댓글2	1	u_3	t_1
t_3	글3 시간	본문3	1	0	0	전체	u_2	u_3	유저3 이미지	유저3 이름	c_3	댓글3 시간	댓글3	2	u_1	t_2

SELECT * FROM 게시글 LEFT JOIN 유저 ON 게시글.게시자 = 유저.아이디

아이디	게시 시간	본문	좋아요 수	댓글 수	공유 수	공개 범위	아이디	프로필사진	이름
t_1	글1 시간	본문1	3	2	1	전체	u_1	유저1 이미지	유저1 이름
t_2	글2 시간	본문2	1	1	5	전체	u_3	유저3 이미지	유저3 이름
t_3	글3 시간	본문3	1	0	0	전체	u_2	유저2 이미지	유저2 이름

게시 시간	본문	좋아요 수	댓글 수	공유 수	공개 범위	게시자 프로필사진	게시자 이름	댓글 게시시간	댓글 본문	댓글 좋아요 수	댓글 게시자 프로필사진	댓글 게시자 이름
글1 시간	본문1	3	2	1	전체	유저1 이미지	유저1 이름	댓글1 시간	댓글1	2	유저2 이미지	유저2 이름
글1 시간	본문1	3	2	1	전체	유저1 이미지	유저1 이름	댓글2 시간	댓글2	1	유저3 이미지	유저3 이름
글2 시간	본문2	1	1	5	전체	유저3 이미지	유저3 이름	댓글3 시간	댓글3	2	유저1 이미지	유저1 이름
글3 시간	본문3	1	0	0	전체	유저2 이미지	유저2 이름					

게시글

유저

댓글

아이디	게시 시간	본문	좋아요 수	댓글 수	공유 수	공개 범위	게시자	아이디	프로필사진	이름	아이디	게시 시간	본문	좋아요 수	게시자	게시글
t_1	글1 시간	본문1	3	2	1	전체	u_1	u_1	유저1 이미지	유저1 이름	c_1	댓글1 시간	댓글1	2	u_2	t_1
t_2	글2 시간	본문2	1	1	5	전체	u_3	u_2	유저2 이미지	유저2 이름	c_2	댓글2 시간	댓글2	1	u_3	t_1
t_3	글3 시간	본문3	1	0	0	전체	u_2	u_3	유저3 이미지	유저3 이름	c_3	댓글3 시간	댓글3	2	u_1	t_2

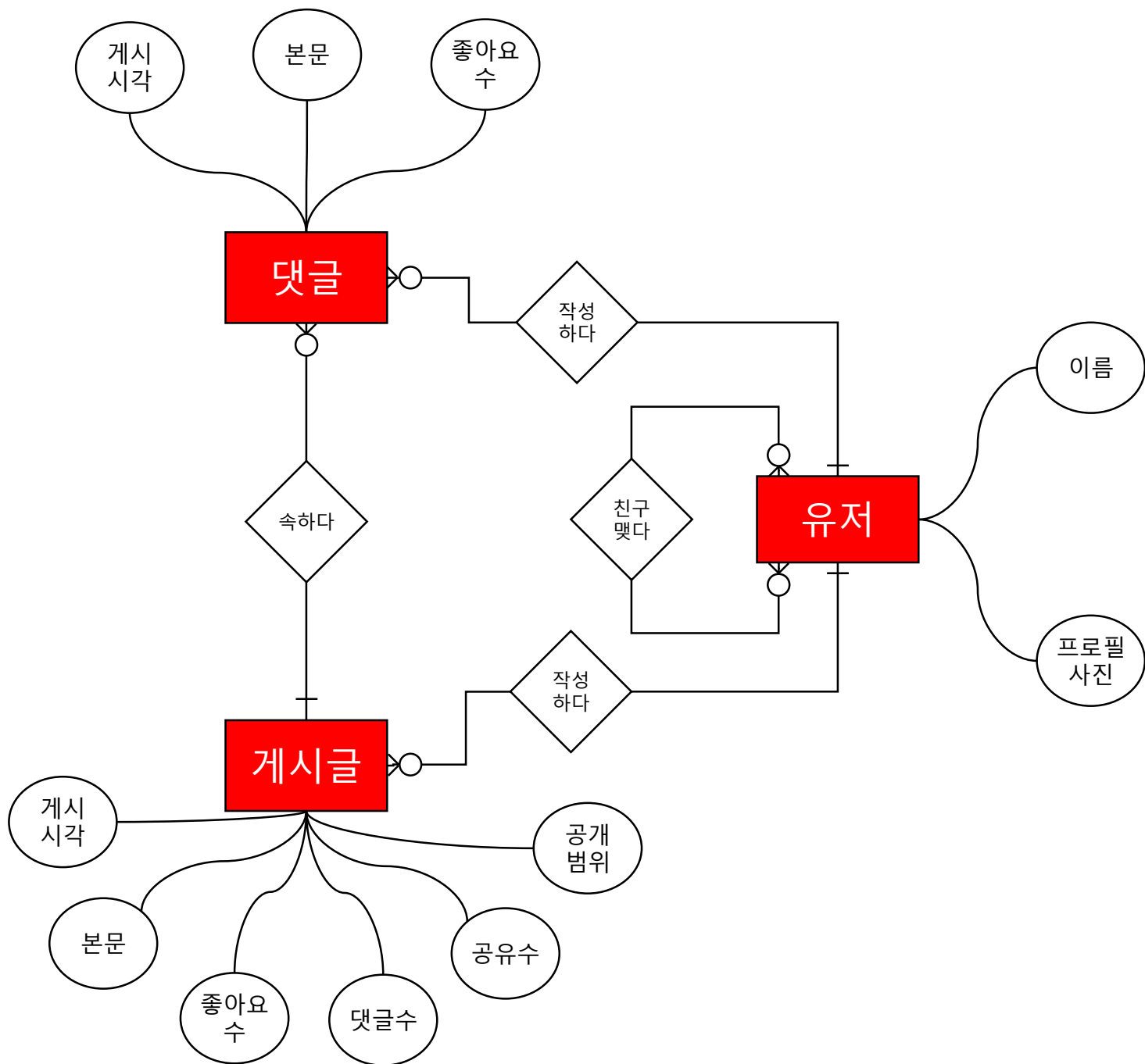
SELECT 아이디, 게시시간, 본문, 좋아요 수, 프로필사진, 이름 FROM 댓글 LEFT JOIN 유저 ON 댓글.게시자 = 유저.아이디

아이디	게시 시간	본문	좋아요 수	프로필사진	이름
c_1	댓글1 시간	댓글1	2	유저1 이미지	유저1 이름
c_2	댓글2 시간	댓글2	1	유저3 이미지	유저3 이름
c_3	댓글3 시간	댓글3	2	유저2 이미지	유저2 이름

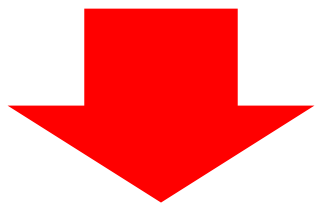
정보그룹



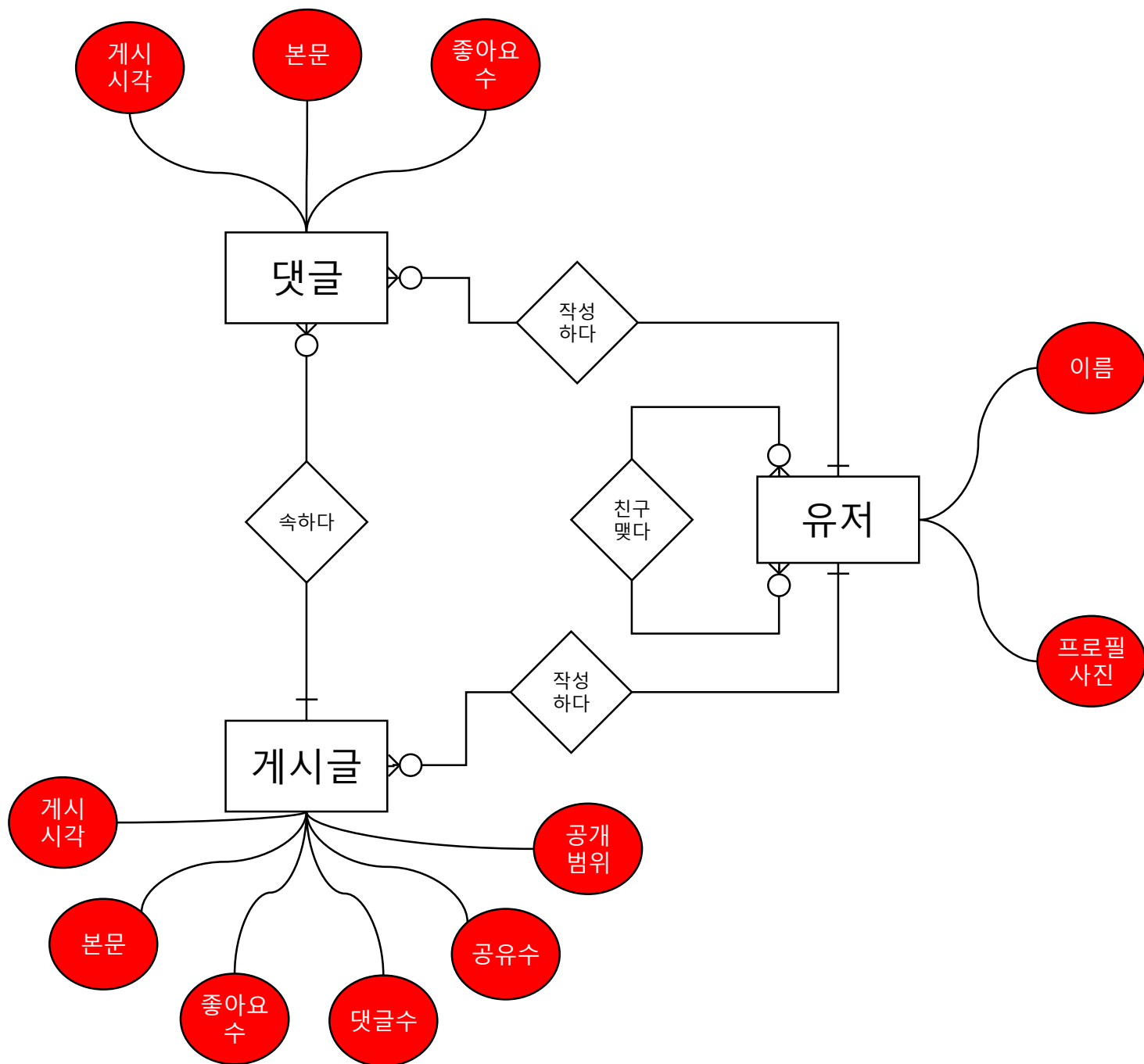
Entity



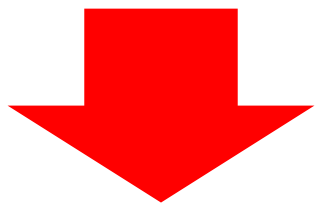
정보



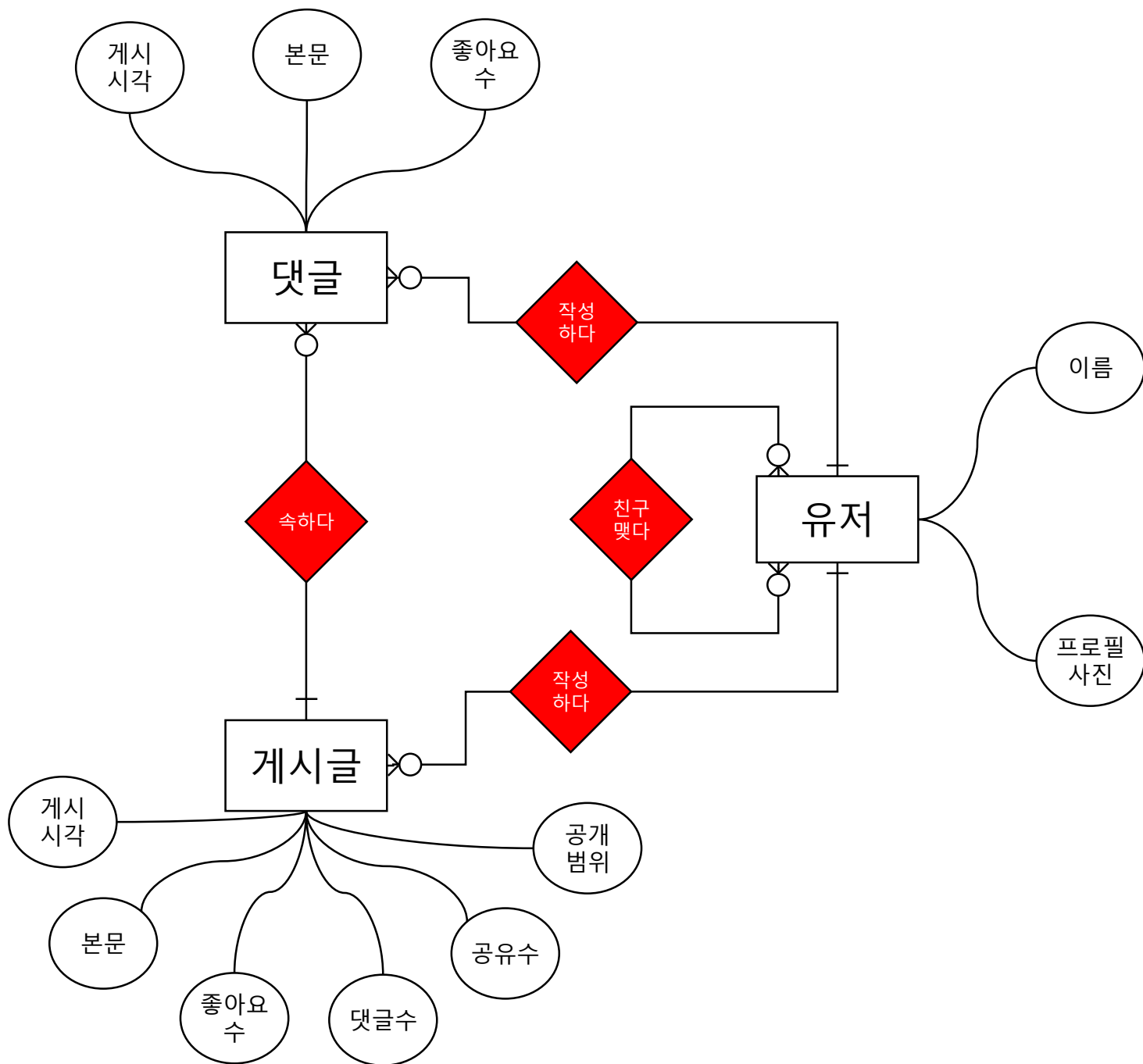
Attribute



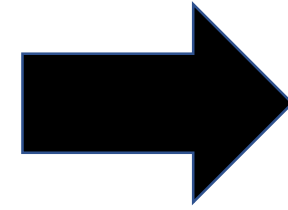
관계



Relation

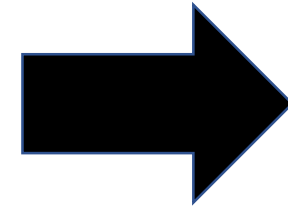


□ Entity



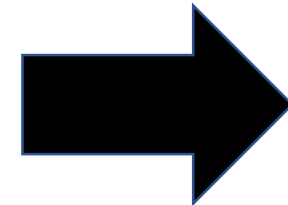
Table

○ Attribute

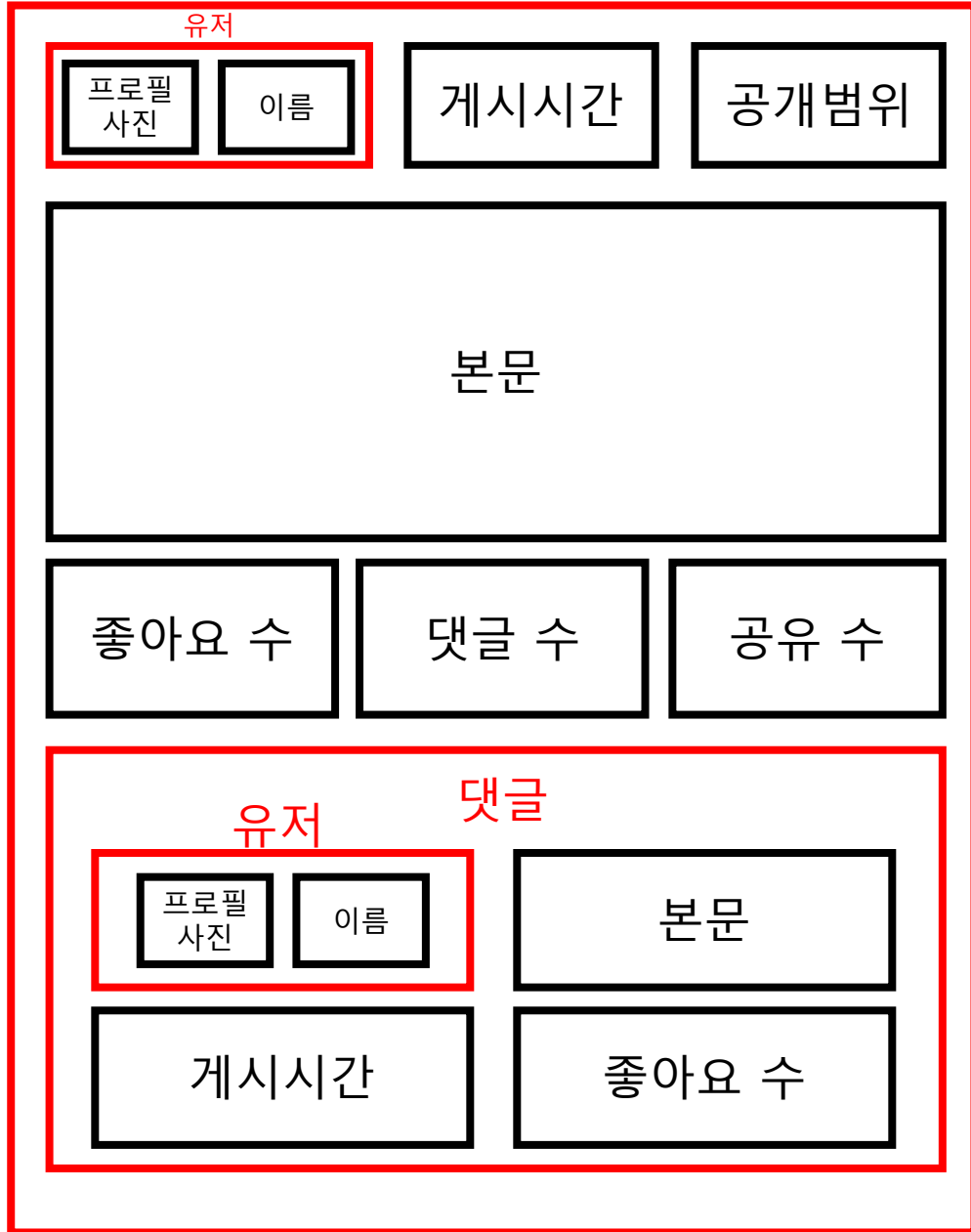


Column

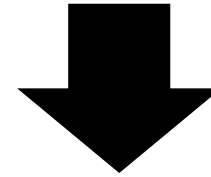
◇ Relationship



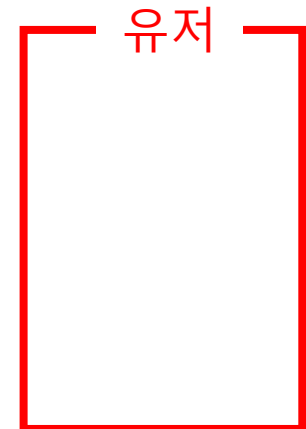
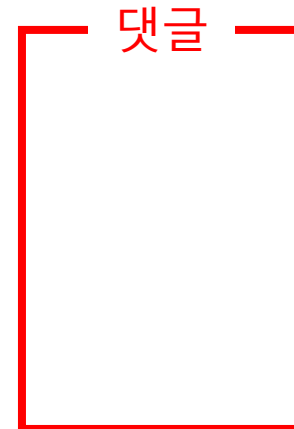
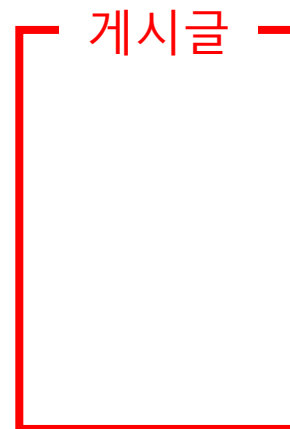
PK,FK

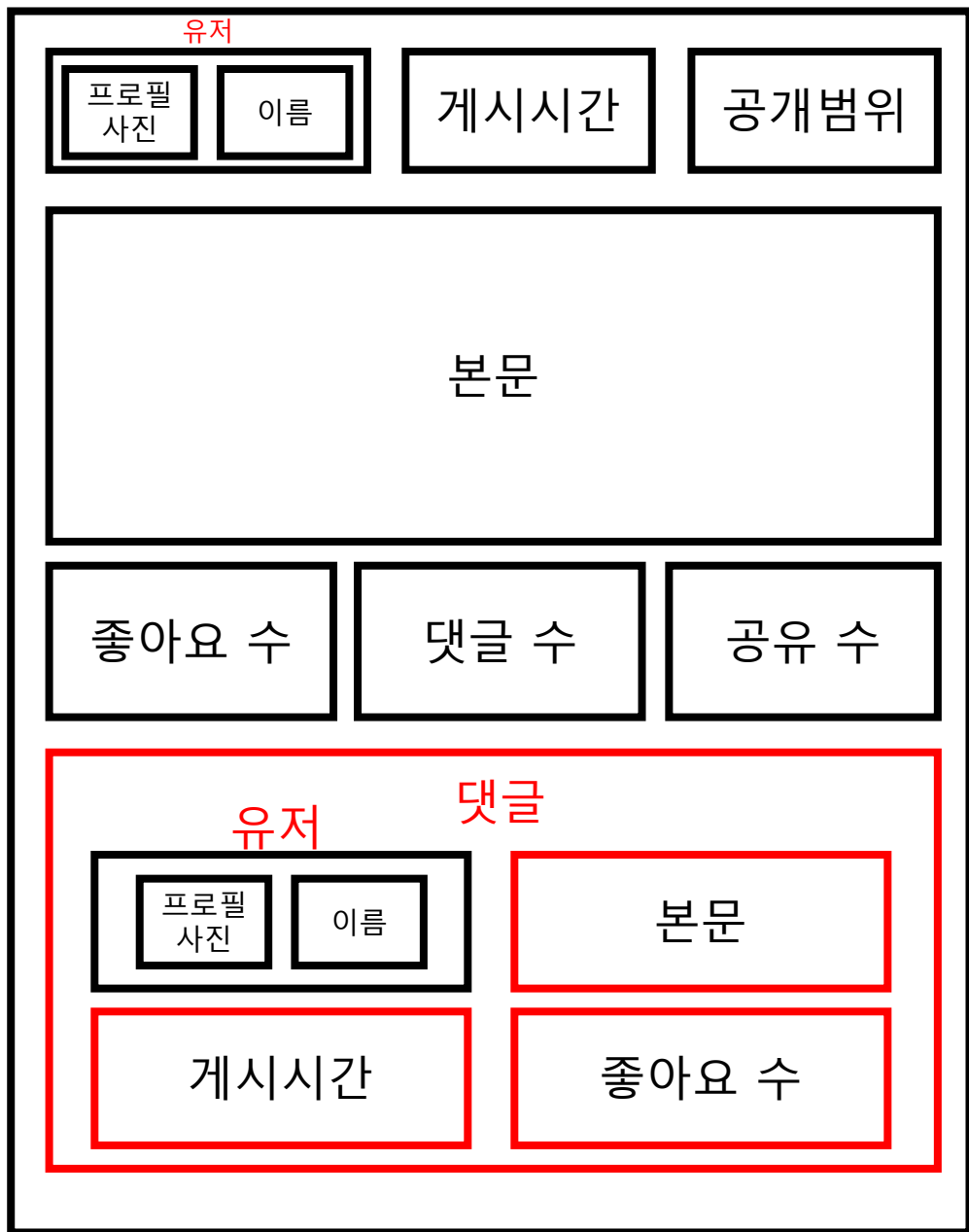


하위 정보를 포함하는 형태의 정보그룹

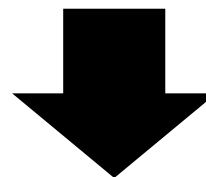


Entity로 정의하여 추출

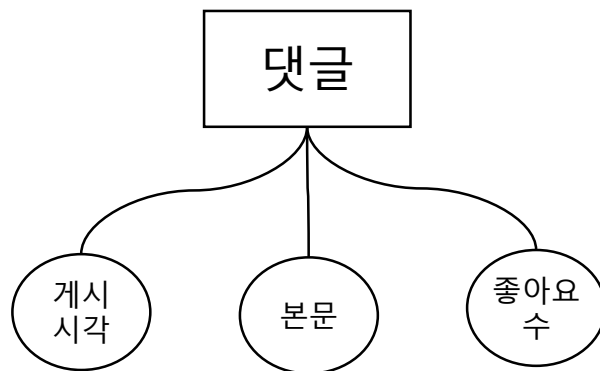




Entity 하위의 정보 항목(단 Entity는 제외)



Attribute로 정의하여 추출



댓글

게시 시간	본문	좋아요 수
댓글1 시간	댓글1	2
댓글2 시간	댓글2	1
댓글3 시간	댓글3	2

Identifier

레코드를 특정하기 위한 attribute 예) 주민번호, 사번, 학번 등

이름은 identifier가 될 수 있는가? -> 될 수 없다. Why? 동명이인이 있기 때문

user_num	User_id	email	Name	Level	Job	city
1	U_10001	alpha@mail.com	Jeff	3	Programmer	Seoul
2	U_10002	Beta@mail.com	David	4	Programmer	Seoul
3	U_10003	gamma@mail.com	Jeff	4	System Engineer	Seoul

Identifier

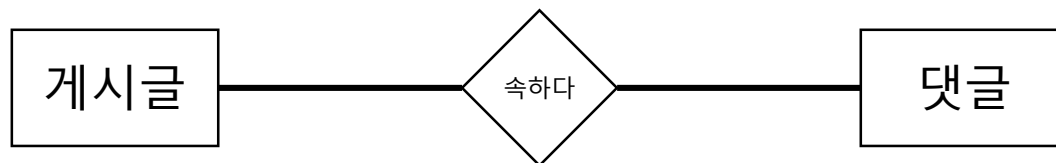
레코드를 특정하기 위한 attribute 예) 주민번호, 사번, 학번 등

이름은 identifier가 될 수 있는가? -> 될 수 없다. Why? 동명이인이 있기 때문

Primary key Alternate key Candidate key

user_num	User_id	email	Name	Level	Job	city
1	U_10001	alpha@mail.com	Jeff	3	Programmer	Seoul
2	U_10002	Beta@mail.com	David	4	Programmer	Seoul
3	U_10003	gamma@mail.com	Jeff	4	System Engineer	Seoul

relationship



게시글

아이디	게시 시간	본문	좋아요 수	댓글 수	공유 수	공개 범위	게시자
t_1	글1 시간	본문1	3	2	1	전체	u_1
t_2	글2 시간	본문2	1	1	5	전체	u_3
t_3	글3 시간	본문3	1	0	0	전체	u_2

댓글

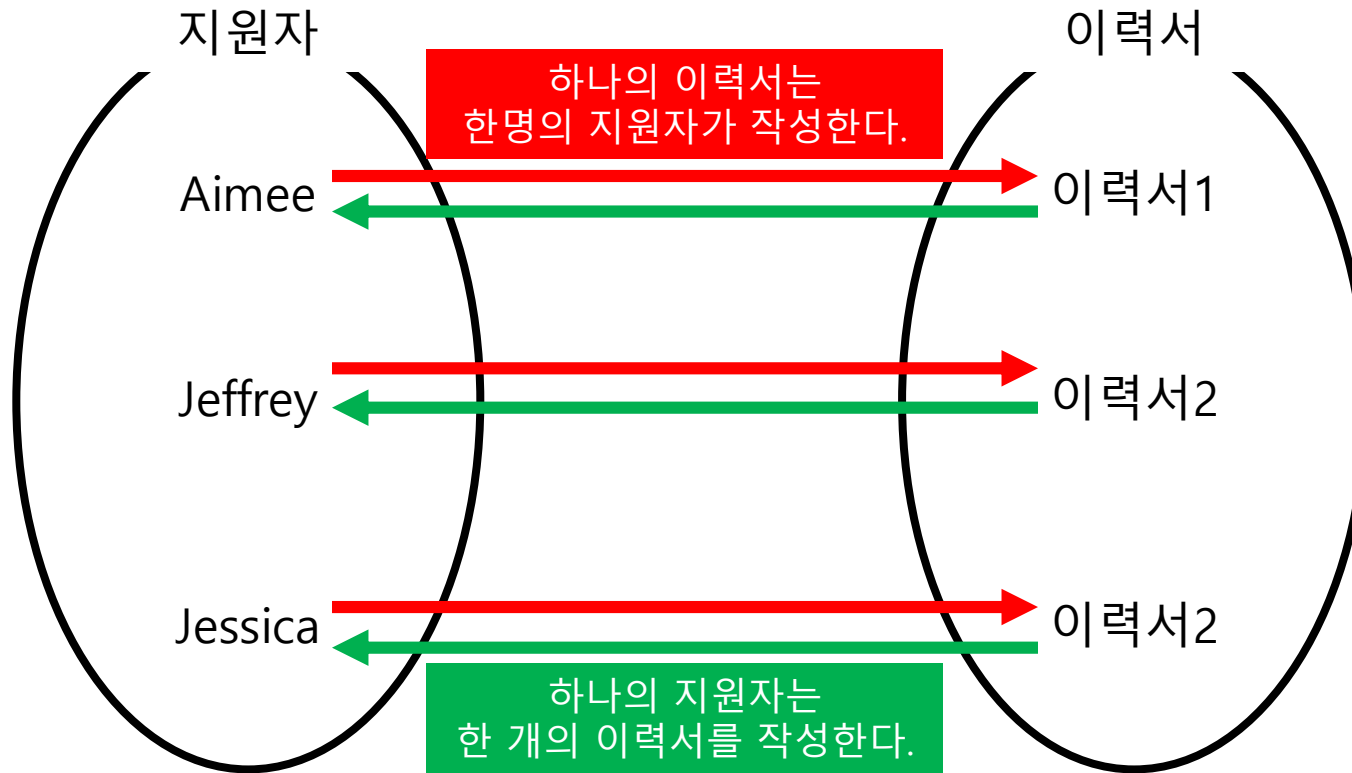
아이디	게시 시간	본문	좋아요 수	게시자	게시글
c_1	댓글1 시간	댓글1	2	u_2	t_1
c_2	댓글2 시간	댓글2	1	u_3	t_1
c_3	댓글3 시간	댓글3	2	u_1	t_2

SELECT * FROM 게시글 LEFT JOIN 댓글 ON 게시글.아이디 = 댓글.게시글

아이디	게시 시간	본문	좋아요 수	댓글 수	공유 수	공개 범위	댓글 아이디	댓글 게시 시간	댓글 본문	댓글 좋아요 수	댓글 게시자 프로필 사진	댓글 게시자 이름
t_1	글1 시간	본문1	3	2	1	전체	c_1	댓글1 시간	댓글1	2	유저2 이미지	유저2 이름
t_1	글1 시간	본문1	3	2	1	전체	c_2	댓글2 시간	댓글2	1	유저3 이미지	유저3 이름
t_2	글2 시간	본문2	1	1	5	전체	c_3	댓글3 시간	댓글3	2	유저1 이미지	유저1 이름
t_3	글3 시간	본문3	1	0	0	전체						

cardinality

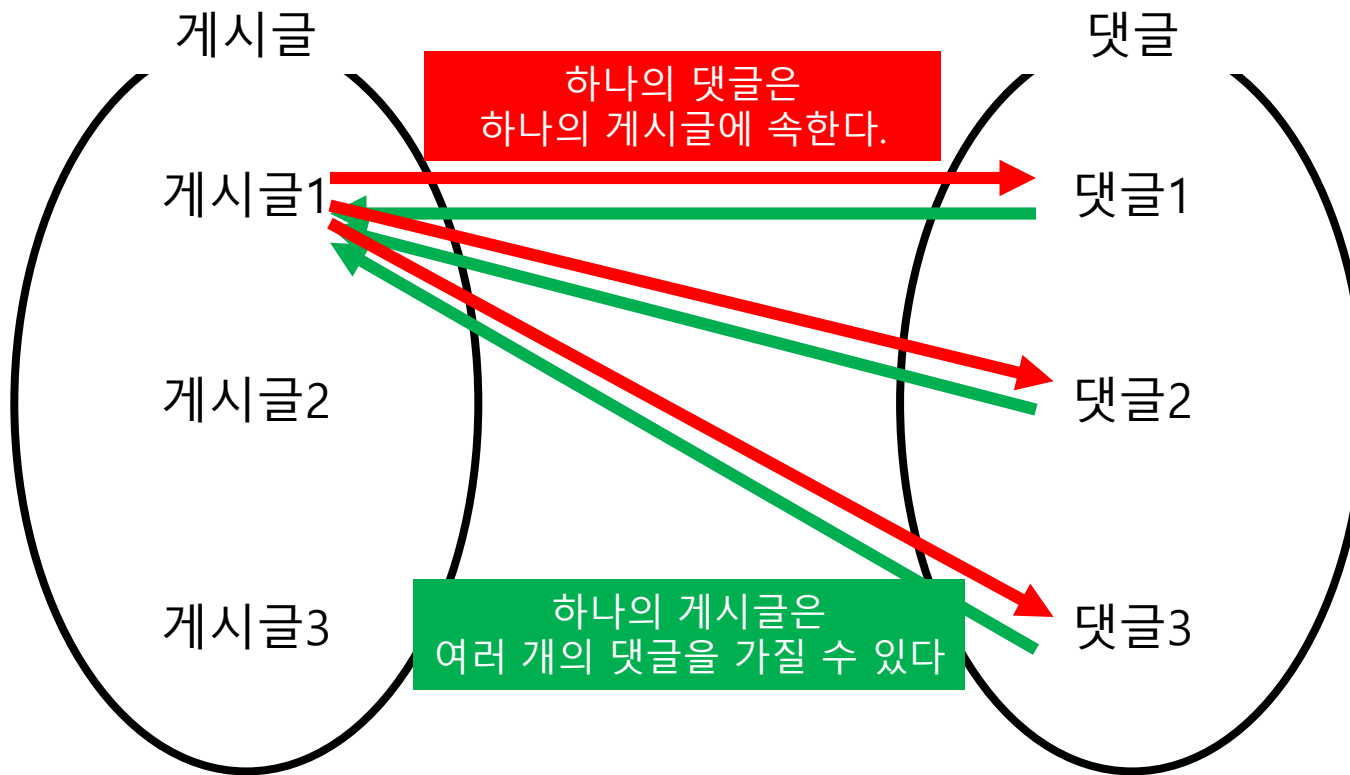
- A entity 하나에 B entity 여러 개가 연결될 수 있을까?
- B entity 하나에 A entity 여러 개가 연결될 수 있을까?



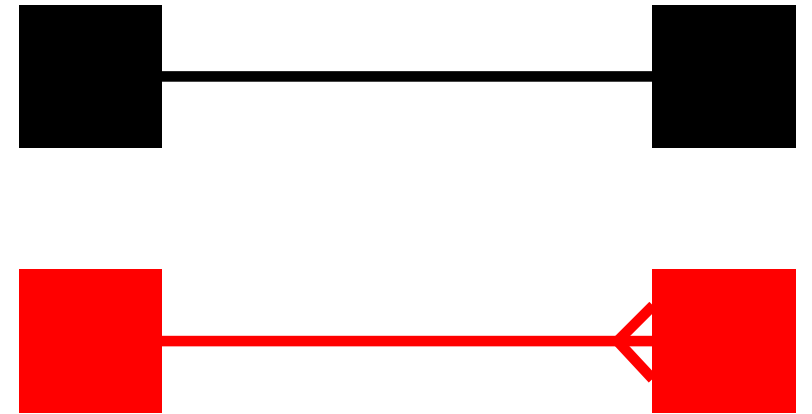
1 : 1 관계

cardinality

- A entity 하나에 B entity 여러 개가 연결될 수 있을까?
- B entity 하나에 A entity 여러 개가 연결될 수 있을까?

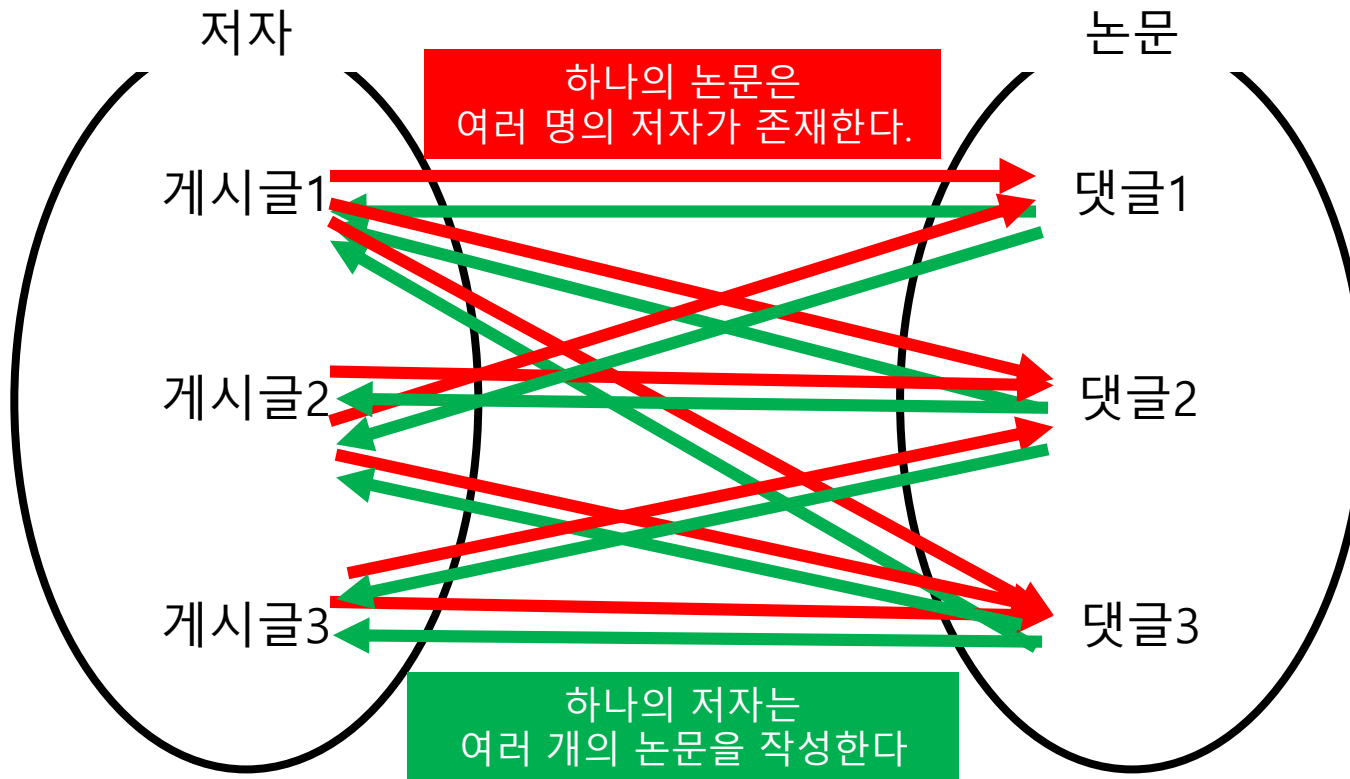


1 : N 관계

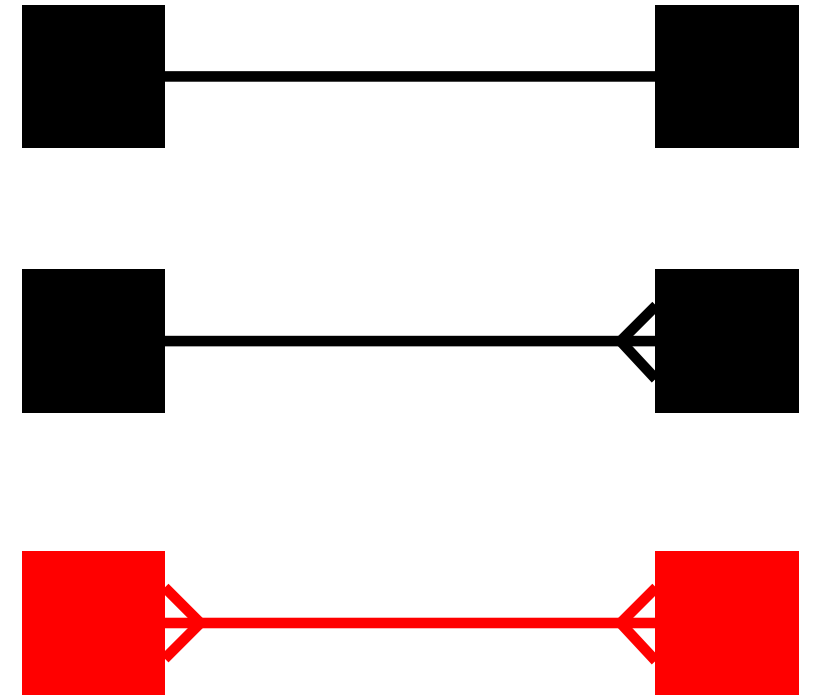


cardinality

- A entity 하나에 B entity 여러 개가 연결될 수 있을까?
- B entity 하나에 A entity 여러 개가 연결될 수 있을까?

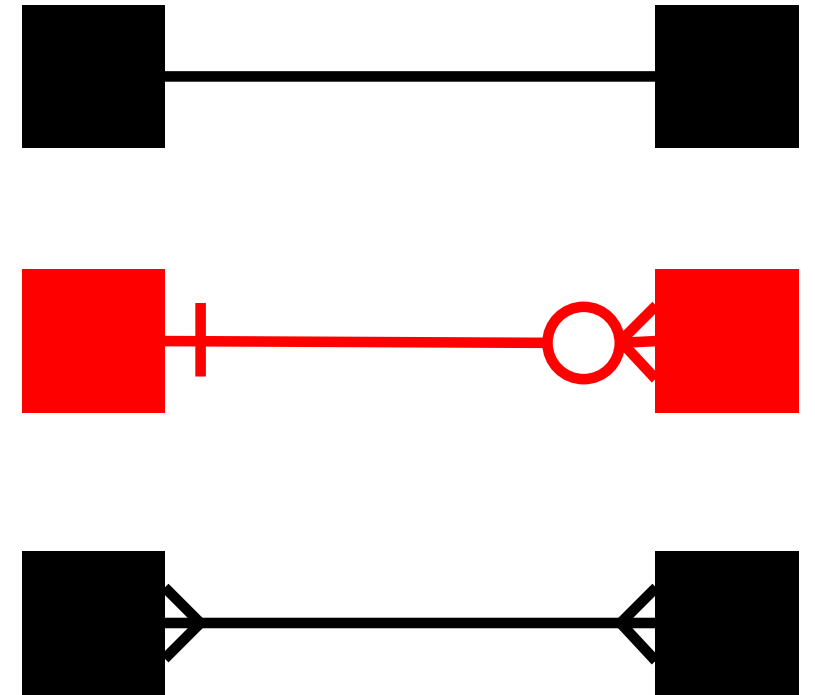
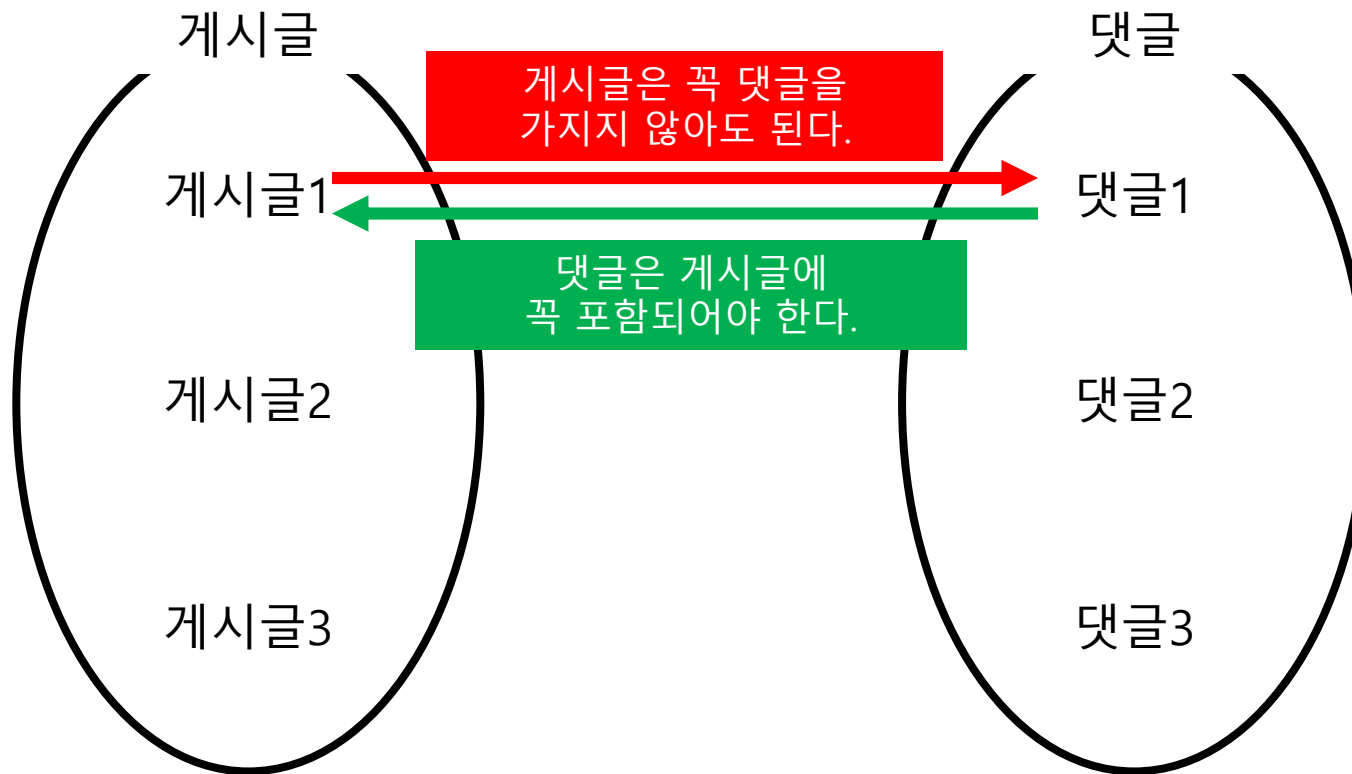


N : M 관계



optionality

- A entity 는 꼭 B entity 와 관계를 맺어야 하는가, 안 맺어도 되는가?
- B entity 는 꼭 A entity 와 관계를 맺어야 하는가, 안 맺어도 되는가?

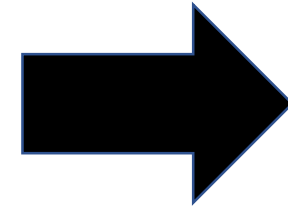


Mapping Rule

개념적 모델링

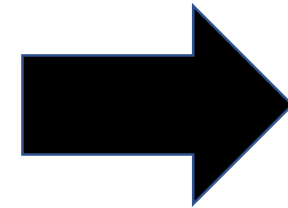
논리적 모델링

 Entity



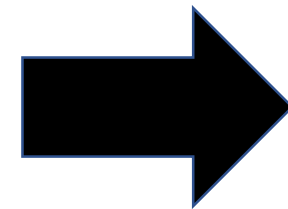
Table

 Attribute

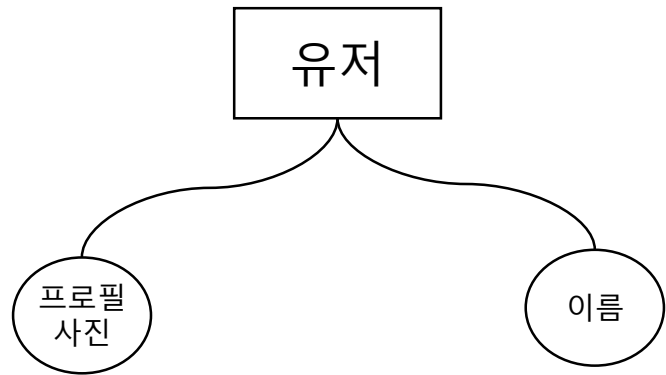
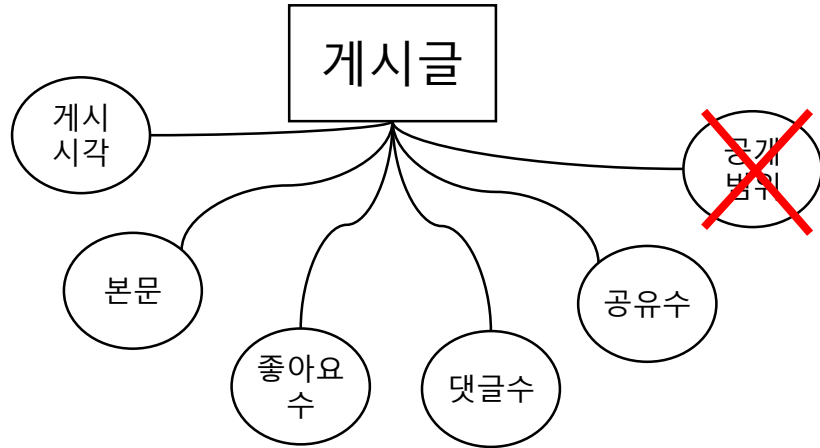
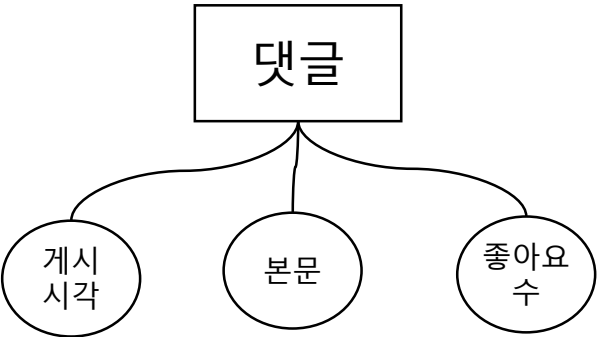


Column

 Relationship



PK,FK



댓글

게시 시간	본문	좋아요 수
댓글1 시간	댓글1	2
댓글2 시간	댓글2	1
댓글3 시간	댓글3	2

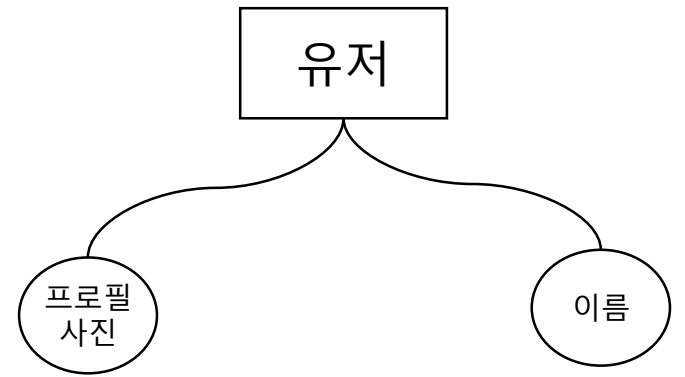
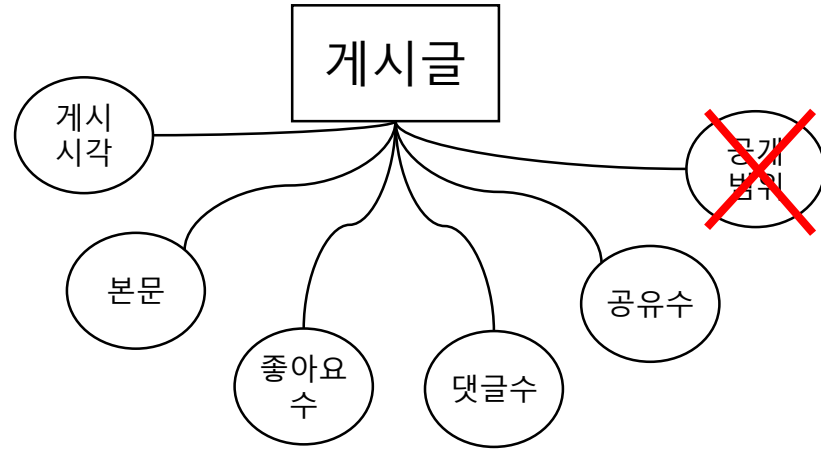
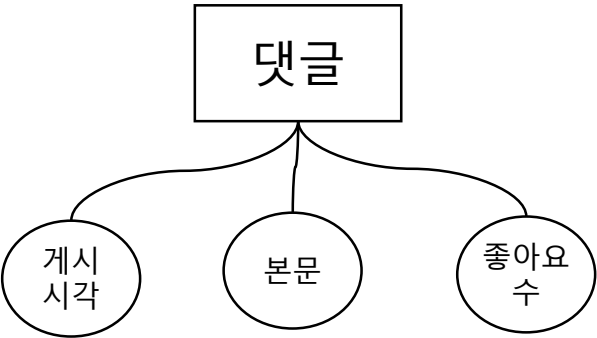
게시글

게시 시간	본문	좋아요 수	댓글 수	공유 수
글1 시간	본문1	3	2	1
글2 시간	본문2	1	1	5
글3 시간	본문3	1	0	0

유저

프로필 사진	이름
유저1 이미지	유저1 이름
유저2 이미지	유저2 이름
유저3 이미지	유저3 이름

비즈니스 도메인에 맞춰서 attribute 선택
Entity를 테이블로 치환



댓글

아이디	게시 시간	본문	좋아요 수
c_1	댓글1 시간	댓글1	2
c_2	댓글2 시간	댓글2	1
c_3	댓글3 시간	댓글3	2

게시글

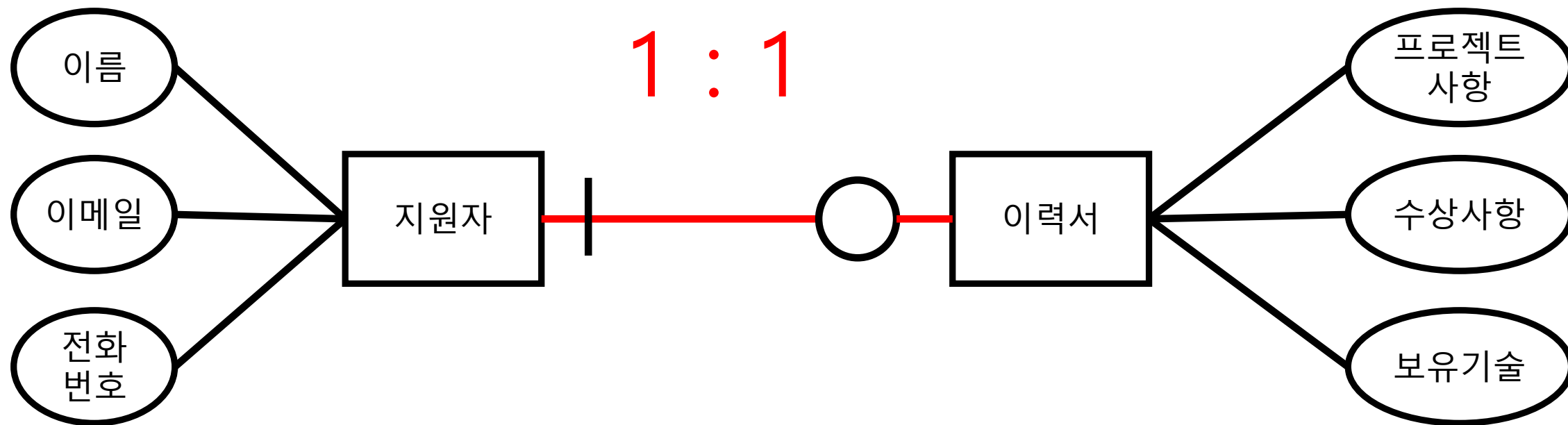
아이디	게시 시간	본문	좋아요 수	댓글 수	공유 수	공개 범위
t_1	글1 시간	본문1	3	2	1	전체
t_2	글2 시간	본문2	1	1	5	전체
t_3	글3 시간	본문3	1	0	0	전체

유저

아이디	프로필 사진	이름
u_1	유저1 이미지	유저1 이름
u_2	유저2 이미지	유저2 이름
u_3	유저3 이미지	유저3 이름

PK를 지정하거나 없으면 ID를 추가

Relationship



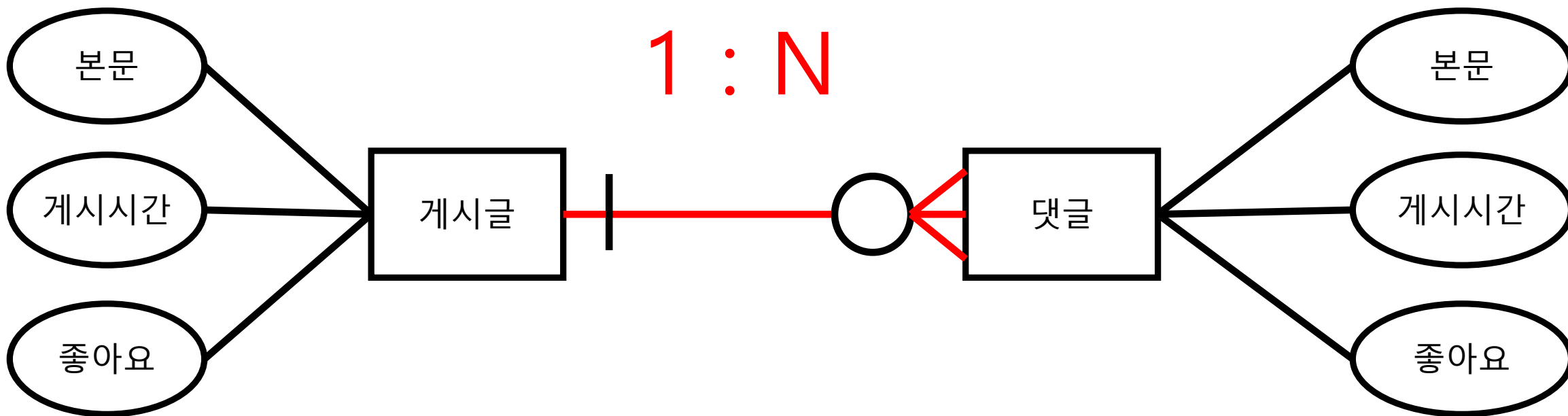
지원자

ID	이름	이메일	전화번호
1	tom	tom@...	60-242 ...
2	mike	mike@...	60-472 ...
3	sarah	sarah@...	60-982 ...

이력서

지원자ID	프로젝트사항	수상사항	보유기술
2	project1 ...	수상1...	자바,파이썬
1	project2 ...	수상2...	파이썬,C
3	project3 ...	수상3...	자바,스칼라

Relationship



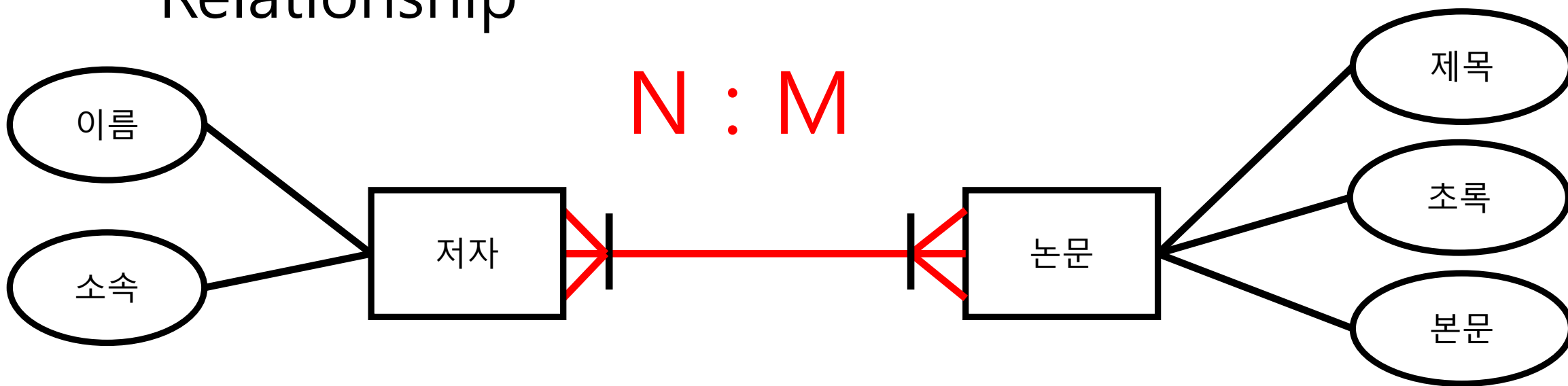
게시글

ID	본문	게시시간	좋아요
1	본문1	2019-08...	2
2	본문2	2019-09...	2
3	본문3	2019-09...	3

댓글

ID	본문	게시시간	좋아요	게시글ID
1	댓글1	2019-08...	1	1
2	댓글2	2019-08...	1	1
3	댓글3	2019-09...	3	3
4	댓글4	2019-09...	1	3

Relationship



저자

ID	이름	소속
1	conner	cambridge
2	david	deepmind
3	kim	modulabs
4	daniel	facebook

작성

저자ID	논문ID
1	3
1	4
2	1
3	2
3	3
4	3

논문

ID	제목	초록	본문
1	논문1	논문1초록	논문1본문
2	논문2	논문2초록	논문2본문
3	논문3	논문3초록	논문3본문
4	논문4	논문4초록	논문4본문

정규화

- DB 테이블의 중복을 제거하여 데이터 업데이트 시 좀 더 효율적인 작업이 가능하게 하기 위해 테이블을 나누는 과정
- 제 1~6 정규화까지 할 수 있음
- 실제 비즈니스에서는 제 3 정규화까지 진행함
(성능적인 이슈 때문)

제 1 정규화

- table의 모든 column 값은 atomic 해야 한다.

id	게시 시간	본문	해쉬태그	좋아요 수	댓글 수	공유 수	게시자 프로필사진	게시자 이름	댓글 게시시간	댓글 본문	댓글 좋아요 수	댓글 게시자 프로필사진	댓글 게시자 이름
1	글1 시간	본문1	클라우드, iot	3	2	1	유저1 이미지	유저1 이름	댓글1 시간	댓글1	2	유저2 이미지	유저2 이름
1	글1 시간	본문1	클라우드,iot	3	2	1	유저1 이미지	유저1 이름	댓글2 시간	댓글2	1	유저3 이미지	유저3 이름
2	글2 시간	본문2	클라우드, LG	1	0	5	유저1 이미지	유저1 이름					
3	글3 시간	본문3	LG	1	1	0	유저2 이미지	유저2 이름	댓글3 시간	댓글3	2	유저3 이미지	유저3 이름



id	게시 시간	본문	해쉬 태그1	해쉬 태그2	좋아요 수	댓글 수	공유 수	게시자 프로필사진	게시자 이름	댓글 게시시간	댓글 본문	댓글 좋아요 수	댓글 게시자 프로필사진	댓글 게시자 이름
1	글1 시간	본문1	클라우드	iot	3	2	1	유저1 이미지	유저1 이름	댓글1 시간	댓글1	2	유저2 이미지	유저2 이름
1	글1 시간	본문1	클라우드	iot	3	2	1	유저1 이미지	유저1 이름	댓글2 시간	댓글2	1	유저3 이미지	유저3 이름
2	글2 시간	본문2	클라우드	LG	1	0	5	유저1 이미지	유저1 이름					
3	글3 시간	본문3	LG	NULL	1	1	0	유저2 이미지	유저2 이름	댓글3 시간	댓글3	2	유저3 이미지	유저3 이름

- 태그가 추가되면?

- NULL로 가득가득?

id	게시 시간	본문	해쉬태그	좋아요 수	댓글 수	공유 수	게시자 프로필사진	게시자 이름	댓글 게시시간	댓글 본문	댓글 좋아요 수	댓글 게시자 프로필사진	댓글 게시자 이름
1	글1 시간	본문1	클라우드, iot	3	2	1	유저1 이미지	유저1 이름	댓글1 시간	댓글1	2	유저2 이미지	유저2 이름
1	글1 시간	본문1	클라우드,iot	3	2	1	유저1 이미지	유저1 이름	댓글2 시간	댓글2	1	유저3 이미지	유저3 이름
2	글2 시간	본문2	클라우드, LG	1	0	5	유저1` 이미지	유저1 이름					
3	글3 시간	본문3	LG	1	1	0	유저2 이미지	유저2 이름	댓글3 시간	댓글3	2	유저3 이미지	유저3 이름

topic

id	게시 시간	본문	좋아요 수	댓글 수	공유 수	게시자 프로필사진	게시자 이름	댓글 게시시간	댓글 본문	댓글 좋아요 수	댓글 게시자 프로필사진	댓글 게시자 이름
1	글1 시간	본문1	3	2	1	유저1 이미지	유저1 이름	댓글1 시간	댓글1	2	유저2 이미지	유저2 이름
1	글1 시간	본문1	3	2	1	유저1 이미지	유저1 이름	댓글2 시간	댓글2	1	유저3 이미지	유저3 이름
2	글2 시간	본문2	1	0	5	유저1 이미지	유저1 이름					
3	글3 시간	본문3	1	1	0	유저2 이미지	유저2 이름	댓글3 시간	댓글3	2	유저3 이미지	유저3 이름

tag

id	tag
1	클라우드
2	iot
3	LG

topic_tag_relation

topic_id	tag_id
1	1
1	2
2	1
2	3
3	3

제 2 정규화

- partial dependencies를 제거해야 한다.

topic

id	게시 시간	본문	좋아요 수	댓글 수	공유 수	게시자 프로필사진	게시자 이름	댓글 게시시간	댓글 본문	댓글 좋아요 수	댓글 게시자 프로필사진	댓글 게시자 이름
1	글1 시간	본문1	3	2	1	유저1 이미지	유저1 이름	댓글1 시간	댓글1	2	유저2 이미지	유저2 이름
1	글1 시간	본문1	3	2	1	유저1 이미지	유저1 이름	댓글2 시간	댓글2	1	유저3 이미지	유저3 이름
2	글2 시간	본문2	1	0	5	유저1 이미지	유저1 이름					
3	글3 시간	본문3	1	1	0	유저2 이미지	유저2 이름	댓글3 시간	댓글3	2	유저3 이미지	유저3 이름

topic_tag_relation

topic_id	tag_id
1	1
1	2
2	1
2	3
3	3

tag

id	tag
1	클라우드
2	iot
3	LG

topic

id	게시 시간	본문	좋아요 수	댓글 수	공유 수	게시자 프로필사진	게시자 이름	댓글 게시시간	댓글 본문	댓글 좋아요 수	댓글 게시자 프로필사진	댓글 게시자 이름
1	글1 시간	본문1	3	2	1	유저1 이미지	유저1 이름	댓글1 시간	댓글1	2	유저2 이미지	유저2 이름
1	글1 시간	본문1	3	2	1	유저1 이미지	유저1 이름	댓글2 시간	댓글2	1	유저3 이미지	유저3 이름
2	글2 시간	본문2	1	0	5	유저1 이미지	유저1 이름					
3	글3 시간	본문3	1	1	0	유저2 이미지	유저2 이름	댓글3 시간	댓글3	2	유저3 이미지	유저3 이름

topic

id	게시 시간	본문	좋아요 수	댓글 수	공유 수	게시자 프로필사진	게시자 이름
1	글1 시간	본문1	3	2	1	유저1 이미지	유저1 이름
2	글2 시간	본문2	1	0	5	유저1 이미지	유저1 이름
3	글3 시간	본문3	1	1	0	유저2 이미지	유저2 이름

comment

id	게시시간	본문	좋아요 수	게시자 프로필사진	게시자 이름	topic id
1	댓글1 시간	댓글1	2	유저2 이미지	유저2 이름	1
2	댓글2 시간	댓글2	1	유저3 이미지	유저3 이름	1
3	댓글3 시간	댓글3	2	유저3 이미지	유저3 이름	3

topic_tag_relation

topic_id	tag_id
1	1
1	2
2	1
2	3
3	3

tag

id	tag
1	클라우드
2	iot
3	LG

제 3 정규화

- transitive dependencies를 제거해야 한다.

topic

id	게시 시간	본문	좋아요 수	댓글 수	공유 수	게시자 프로필사진	게시자 이름
1	글1 시간	본문1	3	2	1	유저1 이미지	유저1 이름
2	글2 시간	본문2	1	0	5	유저1 이미지	유저1 이름
3	글3 시간	본문3	1	1	0	유저2 이미지	유저2 이름

comment

id	댓글 게시시간	본문	댓글 좋아요 수	댓글 게시자 프로필사진	댓글 게시자 이름	topic id
1	댓글1 시간	댓글1	2	유저2 이미지	유저2 이름	1
2	댓글2 시간	댓글2	1	유저3 이미지	유저3 이름	1
3	댓글3 시간	댓글3	2	유저3 이미지	유저3 이름	3

topic_tag_relation

topic_id	tag_id
1	1
1	2
2	1
2	3
3	3

tag

id	tag
1	클라우드
2	iot
3	LG

topic

id	게시 시간	본문	좋아요 수	댓글 수	공유 수	author id
1	글1 시간	본문1	3	2	1	1
2	글2 시간	본문2	1	0	5	1
3	글3 시간	본문3	1	1	0	2

topic_author

id	프로필사진	이름
1	유저1 이미지	유저1 이름
2	유저2 이미지	유저2 이름

topic_tag_relation

topic_id	tag_id
1	1
1	2
2	1
2	3
3	3

comment

id	게시시간	본문	좋아요 수	게시자 프로필사진	게시자 이름	topic id
1	댓글1 시간	댓글1	2	유저2 이미지	유저2 이름	1
2	댓글2 시간	댓글2	1	유저3 이미지	유저3 이름	1
3	댓글3 시간	댓글3	2	유저3 이미지	유저3 이름	3

tag

id	tag
1	클라우드
2	iot
3	LG

topic

id	게시 시간	본문	좋아요 수	댓글 수	공유 수	author id
1	글1 시간	본문1	3	2	1	1
2	글2 시간	본문2	1	0	5	1
3	글3 시간	본문3	1	1	0	2

comment

id	게시시간	본문	좋아요 수	author id	topic id
1	댓글1 시간	댓글1	2	1	1
2	댓글2 시간	댓글2	1	2	1
3	댓글3 시간	댓글3	2	2	3

topic_author

id	프로필사진	이름
1	유저1 이미지	유저1 이름
2	유저2 이미지	유저2 이름

comment_author

id	프로필사진	이름
1	유저2 이미지	유저2 이름
2	유저3 이미지	유저3 이름

topic_tag_relation

topic_id	tag_id
1	1
1	2
2	1
2	3
3	3

tag

id	tag
1	클라우드
2	iot
3	LG

다 됐습니다.

~~다 됐습니다.~~

이러면 망합니다.

topic

id	게시 시간	본문	좋아요 수	댓글 수	공유 수	author id
1	글1 시간	본문1	3	2	1	1
2	글2 시간	본문2	1	0	5	1
3	글3 시간	본문3	1	1	0	2

comment

id	게시시간	본문	좋아요 수	author id	topic id
1	댓글1 시간	댓글1	2	1	1
2	댓글2 시간	댓글2	1	2	1
3	댓글3 시간	댓글3	2	2	3

topic_author

id	프로필사진	이름
1	유저1 이미지	유저1 이름
2	유저2 이미지	유저2 이름

comment_author

id	프로필사진	이름
1	유저2 이미지	유저2 이름
2	유저3 이미지	유저3 이름

topic_tag_relation

topic_id	tag_id
1	1
1	2
2	1
2	3
3	3

tag

id	tag
1	클라우드
2	iot
3	LG

글 쓰는 사람이 댓글도 쓰고 댓글 쓰는 사람이 글도 쓰는 거 아닌가?

합쳐야 합니다.

topic

id	게시 시간	본문	좋아요 수	댓글 수	공유 수	author id
1	글1 시간	본문1	3	2	1	1
2	글2 시간	본문2	1	0	5	1
3	글3 시간	본문3	1	1	0	2

comment

id	게시시간	본문	좋아요 수	author id	topic id
1	댓글1 시간	댓글1	2	2	1
2	댓글2 시간	댓글2	1	3	1
3	댓글3 시간	댓글3	2	3	3

user

id	프로필사진	이름
1	유저1 이미지	유저1 이름
2	유저2 이미지	유저2 이름
3	유저3 이미지	유저3 이름

topic_tag_relation

topic_id	tag_id
1	1
1	2
2	1
2	3
3	3

tag

id	tag
1	클라우드
2	iot
3	LG

정말로

다 됐습니다.

정말로

~~다 됐습니다.~~

는 이제 성능을 신경 써야 합니다

SELECT

SELECT

데이터가 늘어날수록 속도가 떨어집니다.

SELECT

데이터가 늘어날수록 속도가 떨어집니다.

레코드 하나를 찾는데 드는 시간 = t 초

레코드 만개를 찾는데 드는 시간 = $10,000t$ 초

레코드 십억개를 찾는데 드는 시간 = $1,000,000,000t$ 초

JOIN

JOIN

그냥 느립니다. 이걸 그냥 비싸요

index
cache

그래도 정 안된다면

denormalization

index

검색속도를 높이기 위해
DB table에 일종의 목차를
만들어 두는 것

but. 용량을 더 쓰게 되고,
쓰기속도가 느려진다.

=> 손익분기점을 잘 살펴야 한다.

Chapter 5	
네트워크 운영체제를 앤서블을 통해서 관리하기	217
5-1. NX-OS를 다루기	218
5-2. VyOS를 다루기	246
5-3. Cumulus를 다루기	278
Chapter 6	
플레이북을 효율적으로 작성하기	323
6-1. 실패 환경 개선하기	324
6-2. 플레이북을 동적으로 구성하기	354
6-3. 지난 코드를 효율적으로 변환하기	390
Chapter 7	
재사용이 가능한 플레이북 만들기	435
7-1. 플레이북을 재사용하기 위한 형식	436
7-2. 플레이북을 재사용하도록 구성하기	478
부록 1 플레이북을 최적화하기	503
부록 2 CMOB 구성관리 (데이터베이스)	507
부록 3 플레이북의 요소별로 제공하는 옵션	515
찾아보기	520

어디에 index를 걸어야 할까?

일단은 case by case

운영을 하면서 slow query를 찾아야 한다.

많이 사용되는 where절에 등장하는 조건칼럼

random access 보단 sequential access

어디에 index를 걸어야 할까?

일단은 case by case

운영을 하면서 slow query를 찾아야 한다.

많이 사용되는 where절에 등장하는 조건칼럼

random access 보단 sequential access

왜만하면 DBA한테 맡겨라

cache

왜 캐시를 쓰는가?

메모리 : 1초



I/O : 10000초



평균 10000배 차이

로그인을 빠르게 해야 한다면



결론은.....



결론은.....



caffé **bene**
gelato & waffle

DB를 늘리면 되잖아요



DB를 늘리면 되잖아요

여친 : 오빠 $20 + 20$ 은 뭐게 ㅎㅎ?

헬창 : 60이지!

여친 : ?? 오빠 수학못해? 40이잖아....
수학 안배웠어?

왜 이렇게 무식해 ㅠㅠ

헬창 : 그럼 바벨에 봉무게는
조상님이 들어주냐?

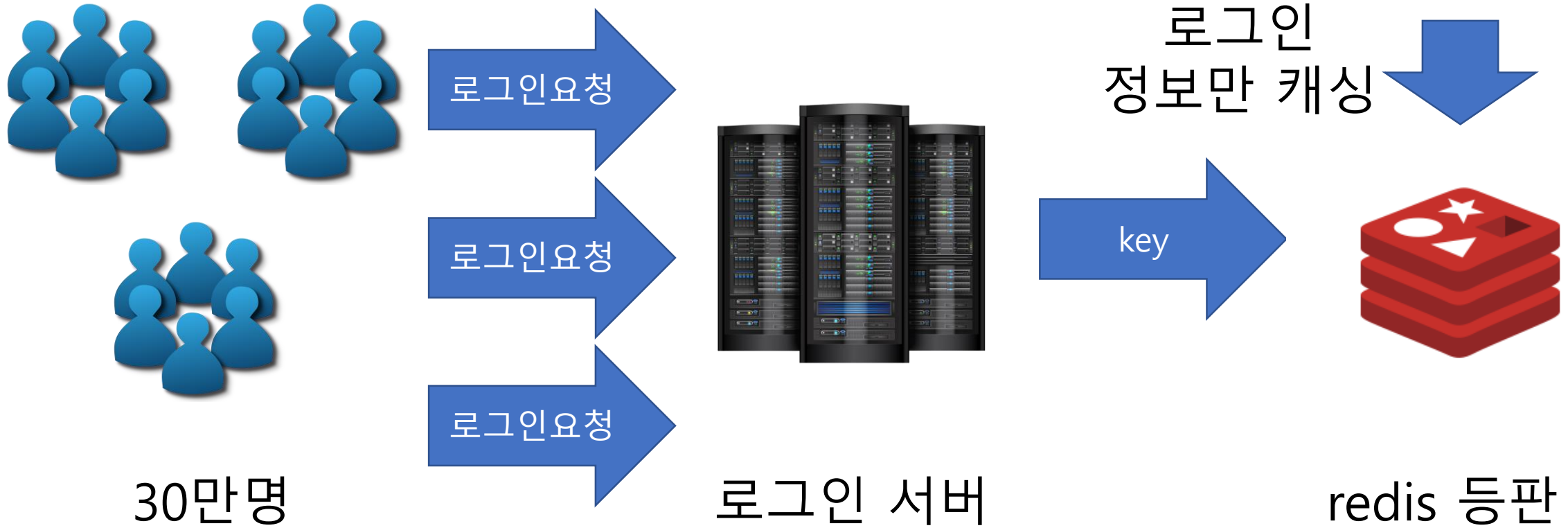
기적의 헬창논리이다. 그에게 있어서
단순덧셈에 바벨무게는 항상 포함된다.

그럼 replication하고
운영은 조상님이 해주냐?



비용은 어떻게 감당할건가?

cache를 써봅시다.



denormalization

- 중복을 다시 허용
- 2중 JOIN 3중 JOIN을 한번의 조인으로 처리하도록 테이블 만듦
- counting을 미리 하는 column을 만듦
- column과 tuple에 맞춰서 테이블을 다시 재구성

denormalization

- 중복을 다시 허용
- 2중 JOIN 3중 JOIN을 한번의 조인으로 처리하도록 테이블 만듦
- counting을 미리 하는 column을 만듦
- column과 tuple에 맞춰서 테이블을 다시 재구성

왜만하면 DBA한테 맡겨라

자 그럼 이제 같이 실습해봅시다.