

40. DBMS ဆိုတဲ့ (၇)

* ဤအားလုံး

JDBC API

Database

[datetime]

(ii) yyyy-MM-dd hh:mm:ss

getDate() \Rightarrow java.sql.Date yyyy-MM-dd

getTimestamp() \Rightarrow java.sql.Timestamp yyyy-MM-dd hh:mm:ss,

41. 외부키(foreign key) 활용

myapp-users

user_id	name	email	pwd	created_at
1	aaa	-	-	-
2	bbb	-	-	-
3	ccc	-	-	-

myapp-projects

project_id	Title	description	start_date	end_date	members
P01	P1	-	-	-	1, 3
P02	P2	-	-	-	1, 2
P03	P3	-	-	-	1, 2, 3

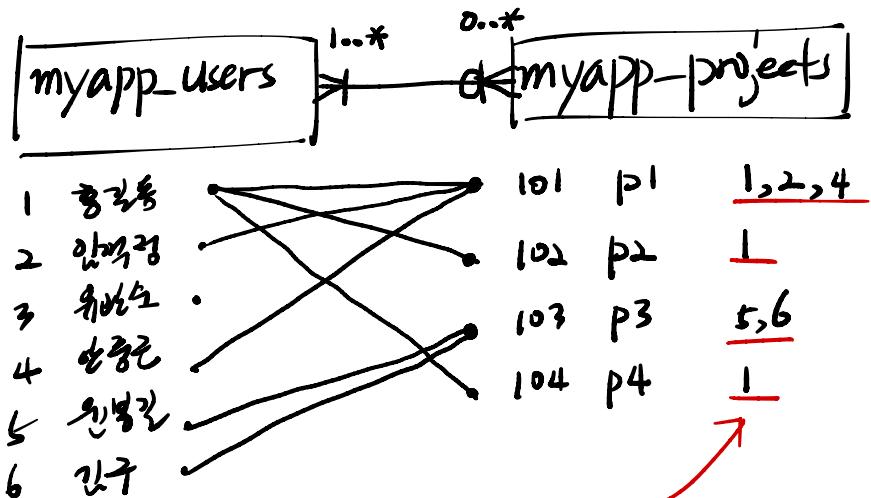
1번은 1번

프로젝트에 참여하는
사람은 반드시 있는
필수적인 조건이다!

↑
필수적이기 때문이다!
↳ 필수인 필수인!

41. 외부키(foreign key) 활용

1) 데이터의 데이터간에 관계를 짜는
(cf 대 cf 관계)



(cf 대 cf)
(cf 대 cf 관계를

외부 키와 함께 관계의 풍부함을

증가시킬 수 있어서 관계를 막을 수 있다

Foreign key 활용
+
데이터의 관계
관계형 데이터베이스로 구현하기 좋다

어디?

* 데이터를 관리하는 틀

- 1 흐름
- 2 일정
- 3 주제

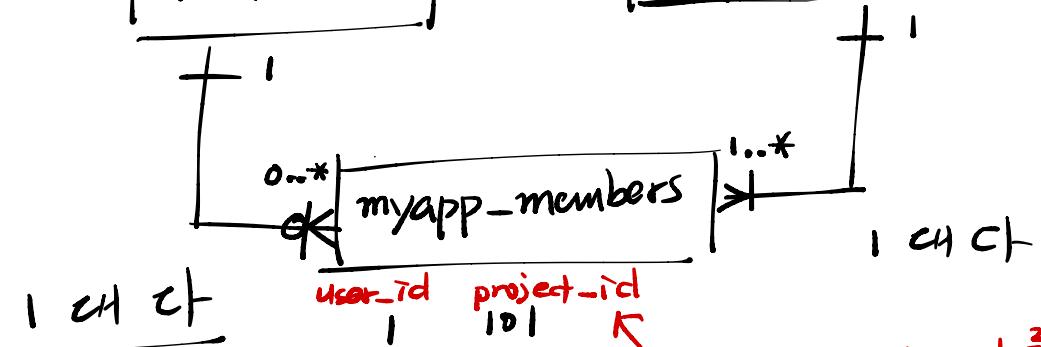
myapp-users	
1	2

- 1 흐름
- 2 일정
- 3 주제

myapp-projects	
101	p1
102	p2
103	p3

* 외부키 (foreign key)

- 다른 테이블의 pk를 가리킨다
- 같은 pk를 사용할 수 있다



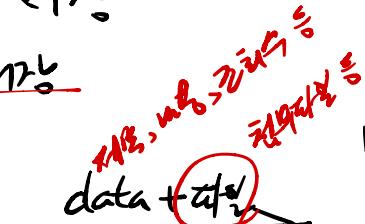
1 CH C1

	user_id	project_id
1	101	
1	103	
2		101
2		102
2		103

다른 테이블의 pk를 → 가리킨다
"Foreign key"

* DB의 특성 짚기

① 데이터가 구조화된



구조화된 형태

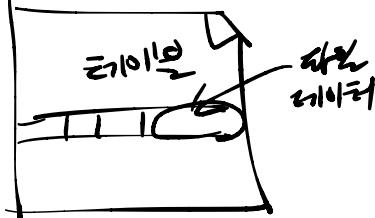
데이터의 구조가
정해져야 한다.
(구조화된 형태)



구조화된 형태
구조화된 형태

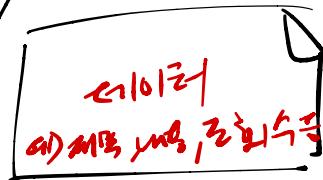


구조화된 형태
구조화된 형태



② 자료를 분리해서 저장

데이터 자체의
구조화된 형태

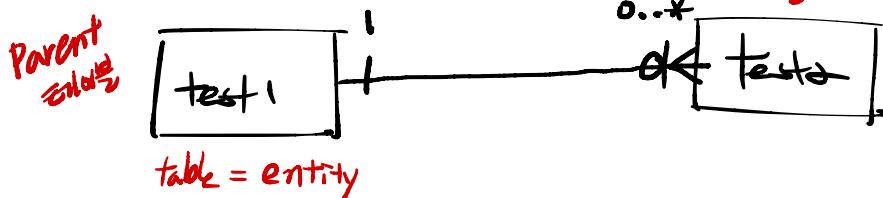


구조화된 형태
구조화된 형태

구조화된 형태



44. child table
(Foreign key)



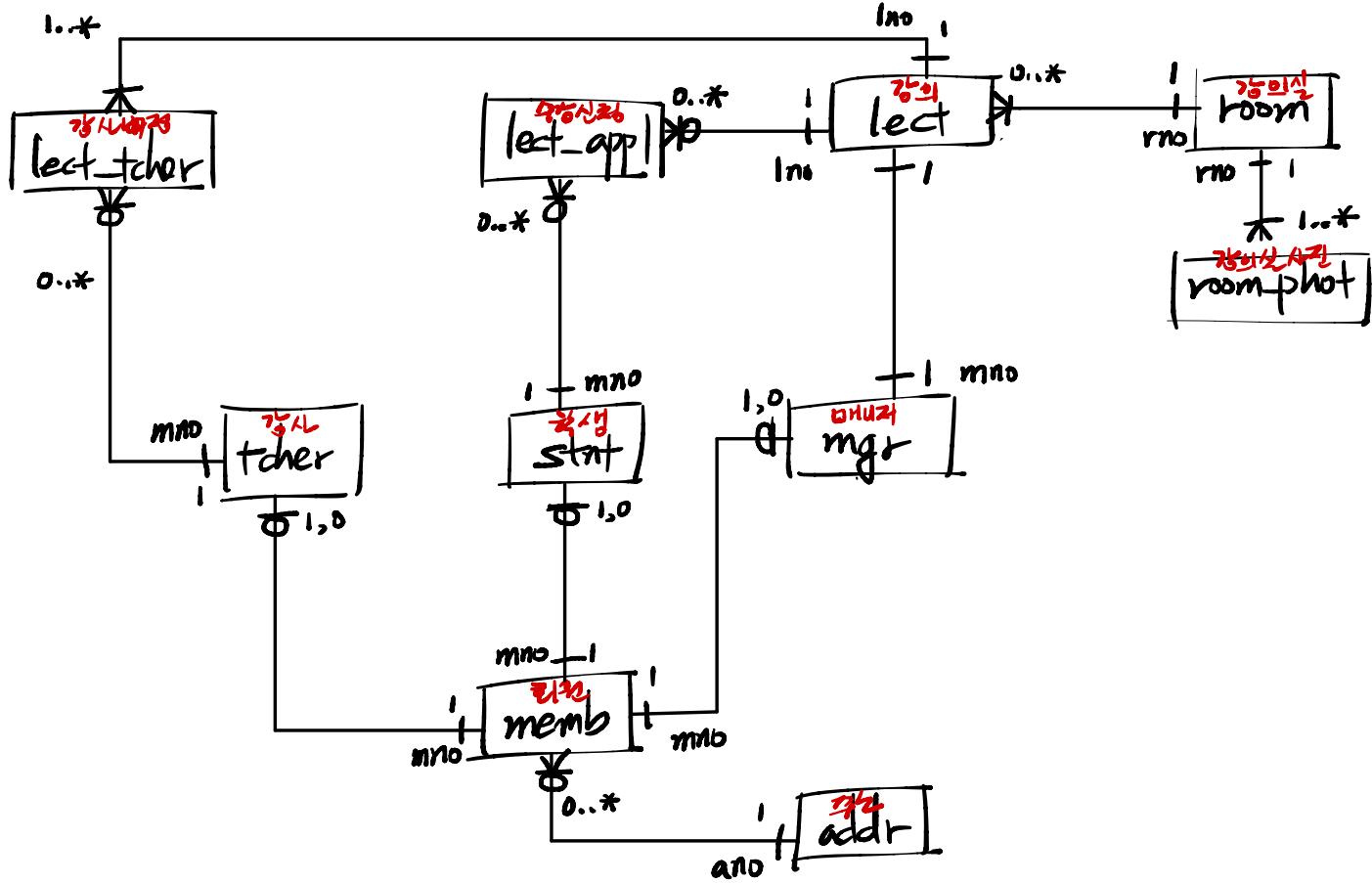
ERD
Information Engineering \rightarrow IEs
(IE \rightarrow DB)

pk	no	title	content	rdt
1	aaa	-	-	.
2	bbb	-	-	.
3	ccc	-	-	.
4	ddd	-	-	.
5	eee	-	-	.

row = record

pk	fno	filepath	bno	fk	test1 (no)
1	101	a1.gif	1		
2	102	a2.gif	1		
3	103	a3.gif	1		
4	104	b1.gif	5		
5	105	b2.gif	5		
6	106	x.gif	00		test1의 no 00은 오류.

* select et join - ERD (Entity Relationship Diagram)



* from → where → select → order by
①

```
mysql> select * from room;
+----+-----+-----+-----+
| rno | loc      | name   | qnty  |
+----+-----+-----+-----+
| 1  | 강남      | 501    | 30    |
| 2  | 강남      | 502    | 30    |
| 3  | 강남      | 503    | 30    |
| 4  | 종로      | 301    | 30    |
| 5  | 종로      | 302    | 30    |
| 6  | 종로      | 303    | 30    |
| 7  | 서초      | 301    | 30    |
| 8  | 서초      | 302    | 30    |
| 9  | 서초      | 501    | 30    |
| 10 | 서초     | 601    | 30    |
+----+-----+-----+-----+
```

select rno, name, concat(loc, ' ', name) as name2
① from room
order by loc asc, name2 asc;

* from → where → color by → order by
②

```
mysql> select * from room;
+----+-----+-----+-----+
| rno | loc      | name    | qnty   |
+----+-----+-----+-----+
| 1  | 강남      | 501     | 30     |
| 2  | 강남      | 502     | 30     |
| 3  | 강남      | 503     | 30     |
| 4  | 종로      | 301     | 30     |
| 5  | 종로      | 302     | 30     |
| 6  | 종로      | 303     | 30     |
| 7  | 서초      | 301     | 30     |
| 8  | 서초      | 302     | 30     |
| 9  | 서초      | 501     | 30     |
| 10 | 서초      | 601     | 30     |
+----+-----+-----+-----+
```

```
select rno, name, concat(loc, ' ', name) as name2
① from room
order by loc asc, name2 asc;
```

* from → where → select → order by

③ ↑ (가장 최종적인 결과 표시)

mysql> select * from room;

(가상 열을 표시) ③
select rno, name, concat(loc, '-', name) as name2
from room

✓rno	loc	✓name	qnty	✓ name2 (loc - name)
1	강남	501	30	강남-501
2	강남	502	30	강남-502
3	강남	503	30	강남-503
4	종로	301	30	종로-301
5	종로	302	30	종로-302
6	종로	303	30	종로-303
7	서초	301	30	서초-301
8	서초	302	30	서초-302
9	서초	501	30	서초-501
10	서초	601	30	서초-601

* from → where → select → order by

④
③ select rno, name, concat(loc, ' ', name) as name2
② from room
① order by loc asc, name2 asc;

rno	loc	name	qnty	name2
1	강남	501	30	강남-501
2	강남	502	30	강남-502
3	강남	503	30	강남-503
7	서초	301	30	서초-301
8	서초	302	30	서초-302
9	서초	501	30	서초-501
10	서초	601	30	서초-601
4	종로	301	30	종로-301
5	종로	302	30	종로-302
6	종로	303	30	종로-303

* from → where → select → order by → 결과 추출 ⑤

rno	name	name2
1	501	강남-501
2	502	강남-502
3	503	강남-503
7	301	서초-301
8	302	서초-302
9	501	서초-501
10	601	서초-601
4	301	종로-301
5	302	종로-302
6	303	종로-303

③ select rno, name, concat(loc, '-', name) as name2
① from room
④ order by loc asc, name2 asc;

select command
선택한 결과의 값을
결과 데이터로 추출한다