

* DBMS 개념

↳ Database Management System

데이터 보관과 조회를 서비스하는 프로그램

* Database

- 실시간 접근 가능 : 데이터 처리 요청에 즉시 응답
- 동시에 풍부 : 여러 사용자가 풍부
- 데이터의 독립성 : 애플리케이션에 비중적
- 일관성 : 데이터 처리 작업이 끝난 후 데이터의 값은 유일한 상태를 유지해야 한다.
- 무결성 : 유일하지 않은 데이터의 등록과 변경, 삭제는 제한
- 보안성 : 사용자 인증과 권한 관리 ↳ 예) 기사를 작성할 수 있다. ↳ 예) 기사를 작성할 수 없다.

auth

(authentication) (authorization)

ID/PWD 일치

기능의 이용권한 검사

유일한 사용자

등록 허용?
변경 허용?

:

* DBMS

- 데이터베이스를 관리하는 S/W
- 예) Oracle, MySQL, PostgreSQL, MS-SQL, Altibase, Tibero, Cubrid 등

↳ 예) 일관한 데이터 관리

↓
실제적인 환경의 구조 관리

SQL 언어로 명령어 작성



명령어

DBMS
Client

결과 출력

↑
MariaDB
client
App.

데이터작업의 요청

DBMS 통신규칙

요청 처리 결과

MariaDB



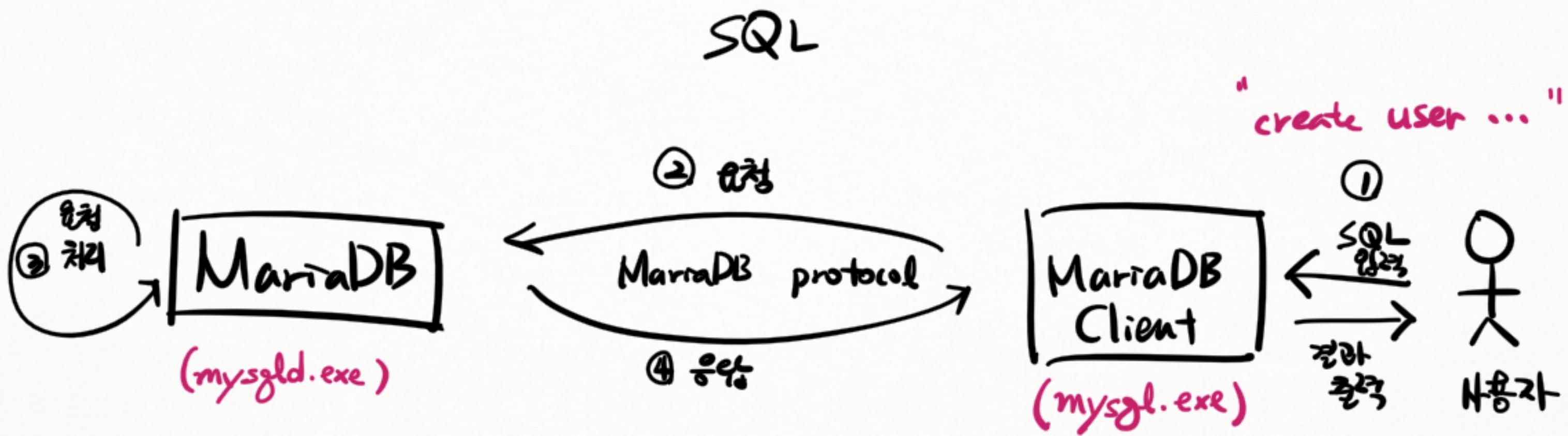
DBMS

Data
I/O

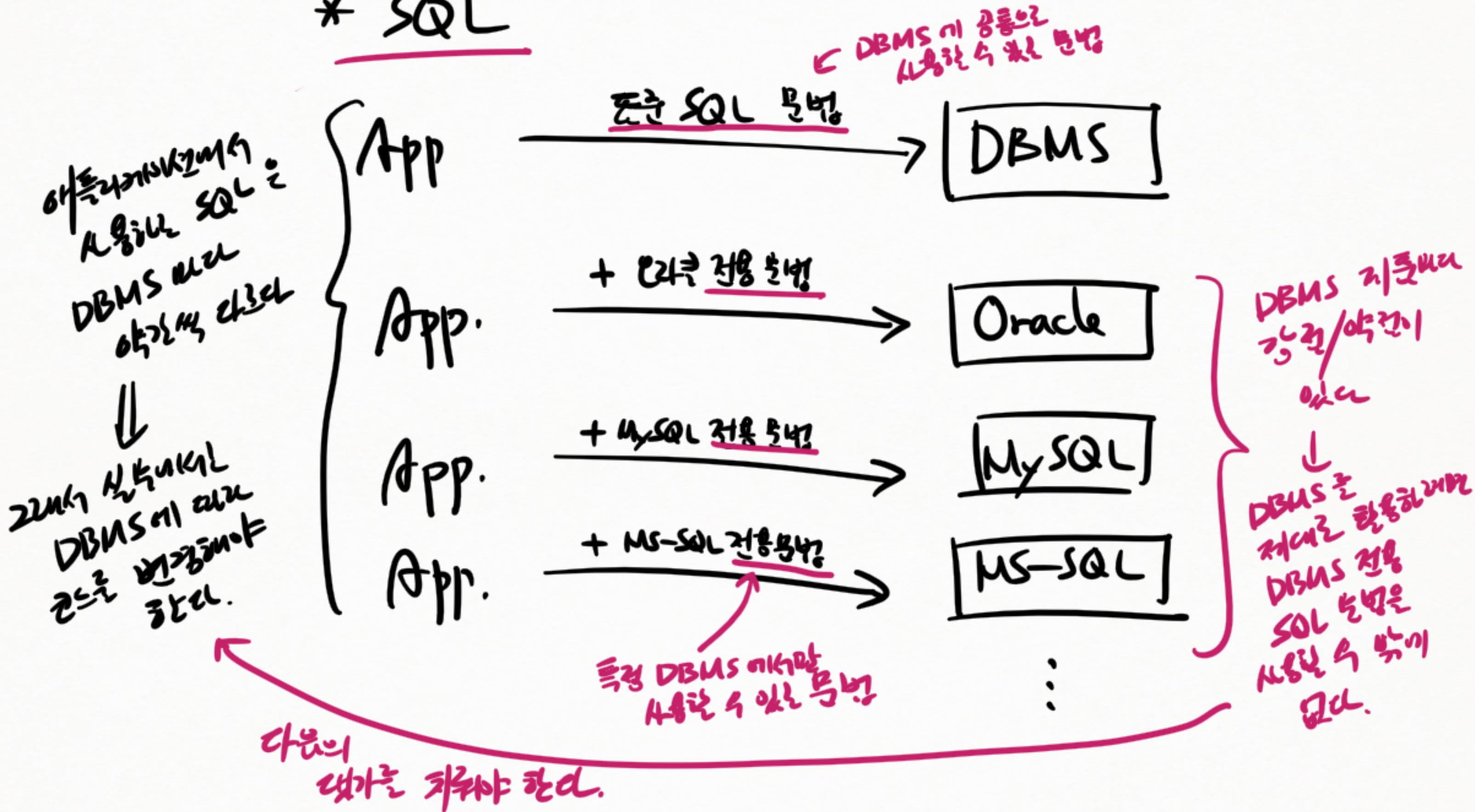
파일

- 데이터를 파일에 저장
- 파일에 저장된 데이터를 조회/변경/삭제
- 데이터베이스의 기능을 제공

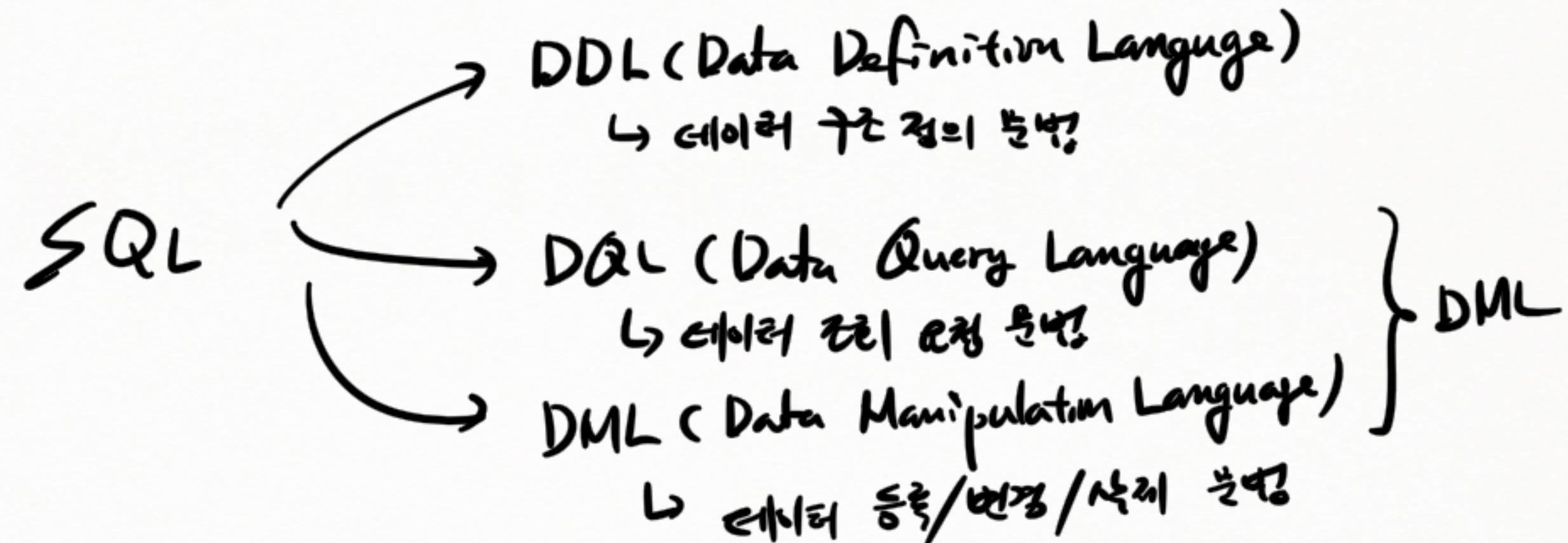
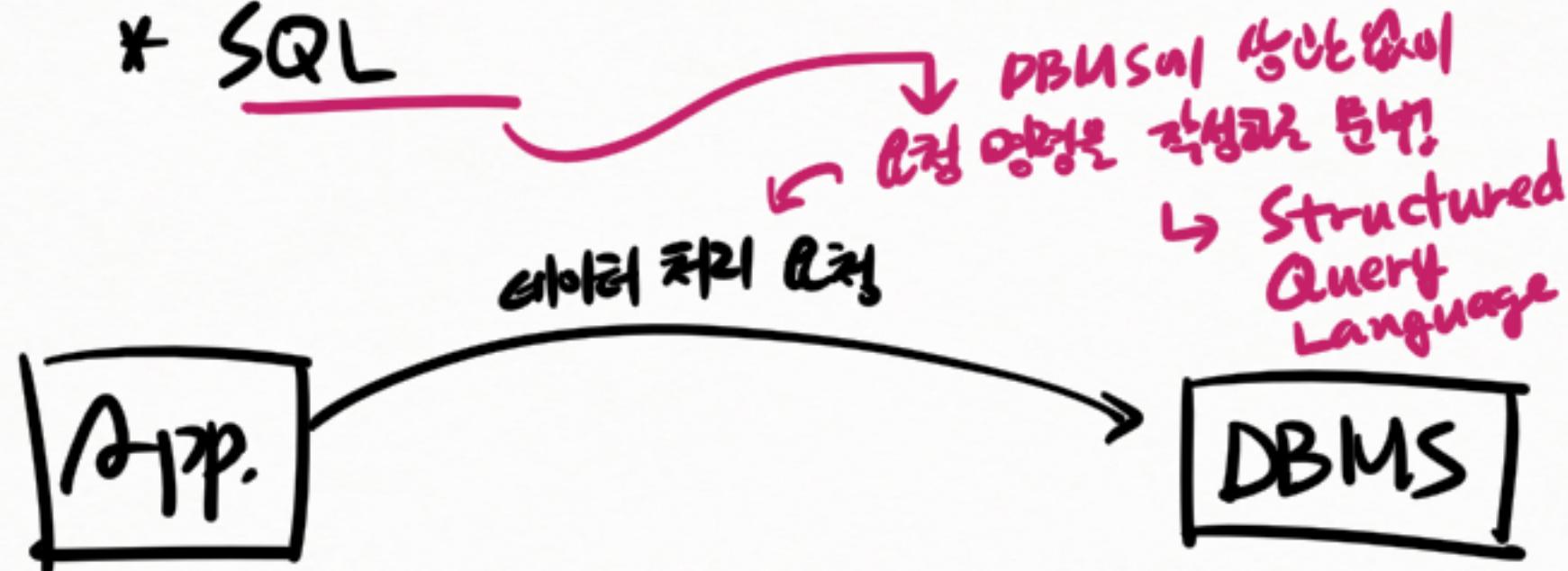
* DBMS 와 SQL



* SQL



* SQL



* SQL

① DDL → 데이터를 저장하는 다음 DB 개체를 정의한다

② $\begin{cases} DQL \\ DML \end{cases}$ → DB 개체의 데이터를 다룬다 $\Rightarrow \begin{cases} \text{Table} \\ \text{View} \\ \text{Procedure} \\ \text{Function} \\ \vdots \end{cases}$

* SQL 테스트 준비

① 사용자 추가

```
> create user 'study'@'%' identified by '1111'; <
```

② 레이터를 저장할 레이터 베이스 생성

③ 데이터베이스를 사용할 사용자의 권한을 지정

* DDL (Data Definition Language)

① 테이블 생성

레이블 쟁쟁

create table test01 (

name varchar(50) not null,
← 커럼 타입 ← 옵션

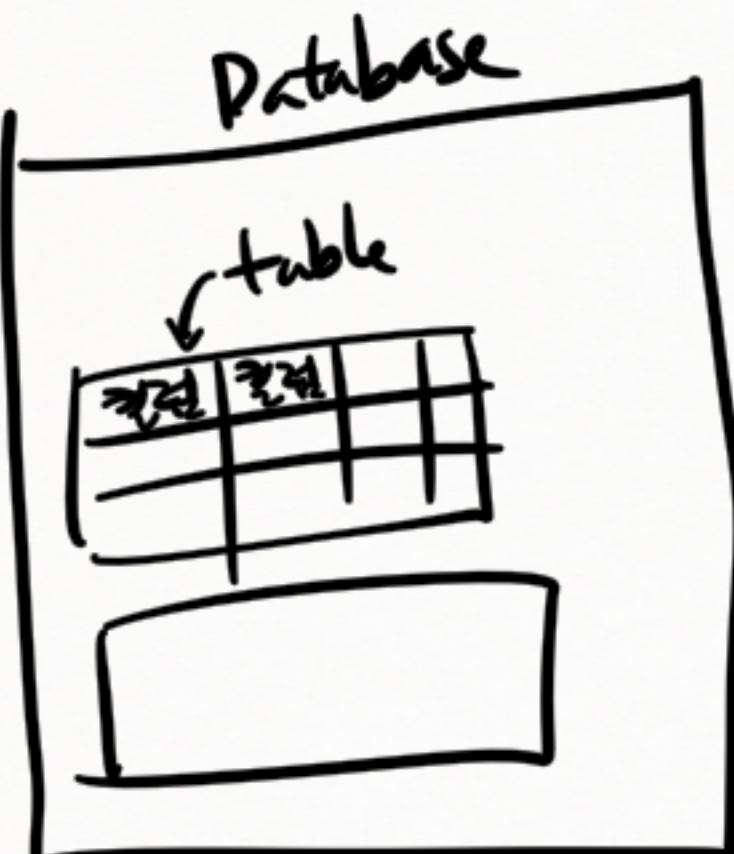
컬럼명 kor int not null,

eng int not null,

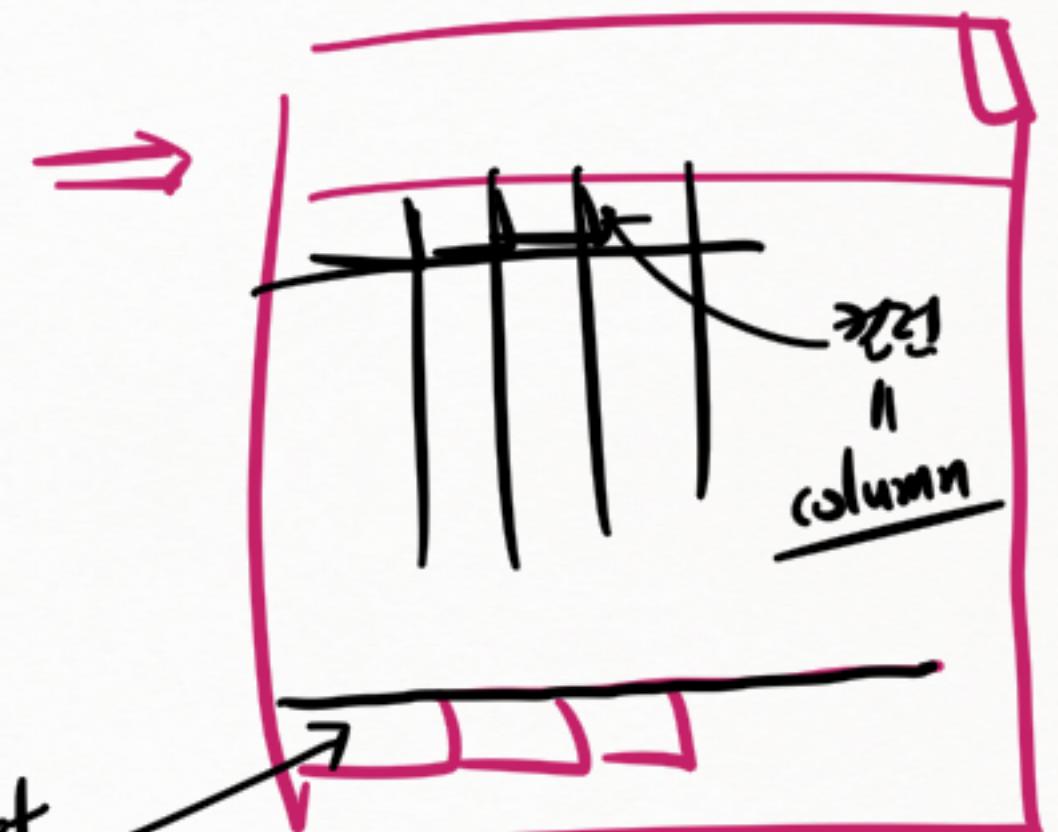
math int not null,

sum int not null,

aver float not null);

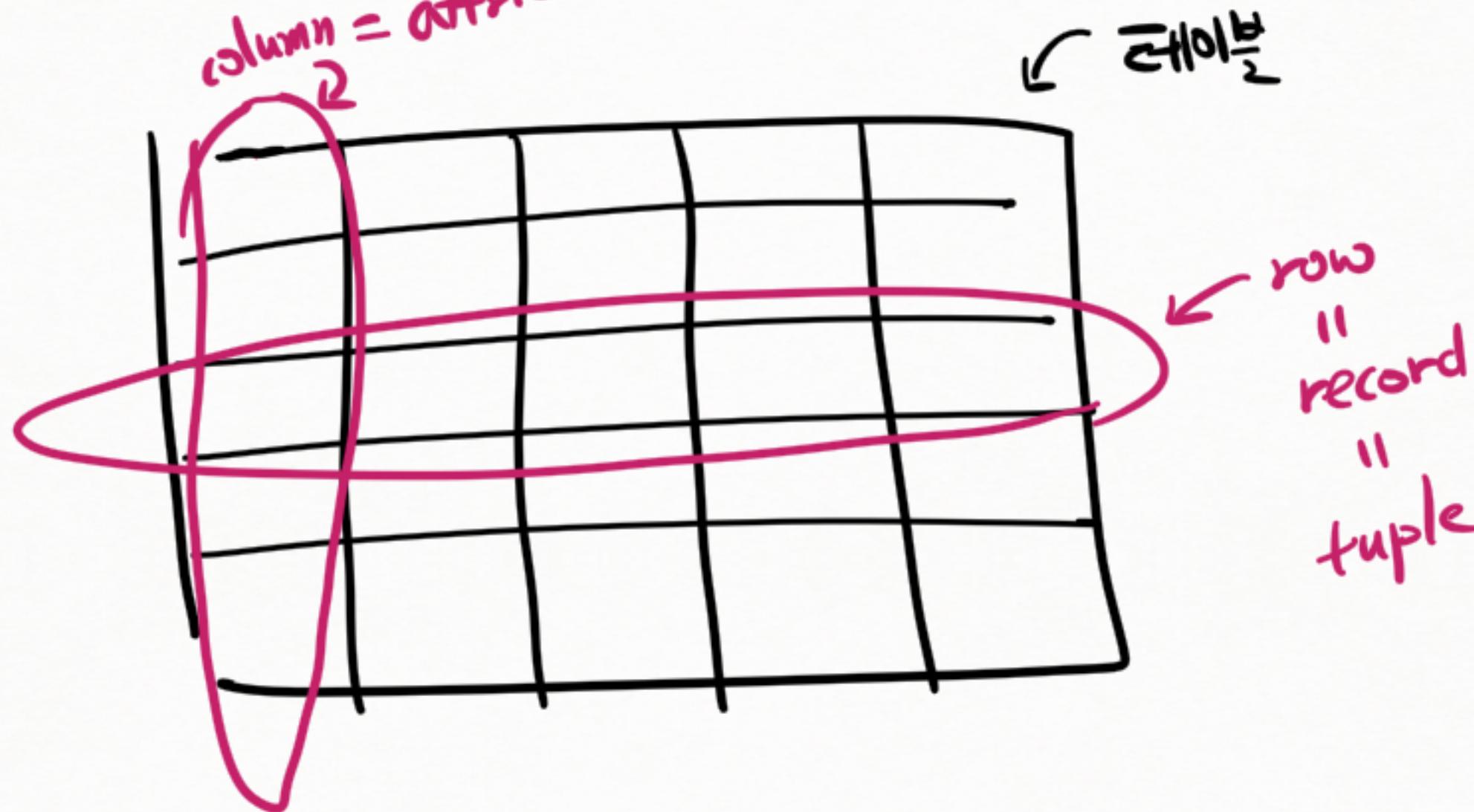


액션 나 비디



* 레이블과 row, column

column = attribute = field



* 테이블

✓ create table 테이블명 (컬럼, ...) ← 테이블 정의

✓ drop table 테이블명 ← 테이블 삭제

✓ alter table 테이블명 변경사항 ← 테이블 변경

✓ describe 테이블명 ← 테이블 정보 조회
 "desc"

* insert

insert into test1(no, name) values(1, 'aaa')

↑
레코드명

↑
컬럼명

↑
컬럼에 들어갈 값

컬럼을 나열한 순서다 일치해야 한다.

* select

select no, name from test1

 ↑↑
 컬럼명

 *
 ↑
 모든컬럼

* key, candidate key, primary key / alternate key

키/C→I

artificial key

데이터를 구현할 때 사용한 건전진*

key

[이메일]

[아이디]

[주민번호]

[이름, 전화번호]

[아이디, 전화번호]

[이메일, 주민번호]

[이메일, 이름]

[주민번호, 이름]

[이름, 아이디, 전화번호]

[이름, 주민번호]

* key, candidate key, primary key / alternate key

키/C→I

artificial key

데이터를 구현할 때 사용한 키는 *

key

candidate key

↳ 키/C

* [이디]

* [아이디]

* [주민번호]

* [이름, 전화번호]

[이디, 이름]

[아이디, 전화번호]

[주민번호, 이름]

[이름, 주민번호]

[이디, 주민번호]

[이름, 아이디, 전화번호]

* key, candidate key, primary key / alternate key

키/C

candidate key

데이터를 구분할 때 사용한 키

Alternate
key

→ 키로 설정되지 않은
내부 키로 다른 키를
'대안키'가 된다

key

candidate key
↳ 키/C

[아이디]

* [주민번호]

* [이름, 전화번호]

↑
DB 관리자가
주 키로
설정한 키

"Primary key"

* Artificial key

제시글 : 제목, 내용, 작성일, 작성자, 조회수, 좋아요수

제시글 번호

Primary key?

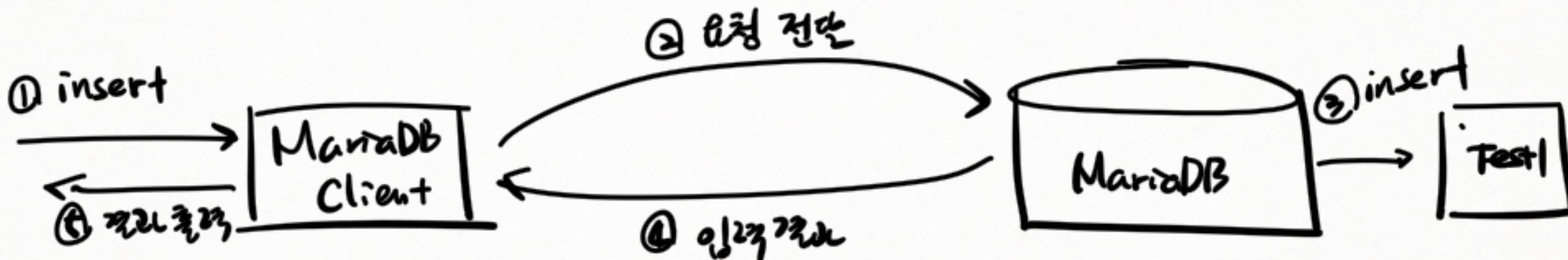
보통히 사용할 key를 없을 경우

Artificial (인공적)
key

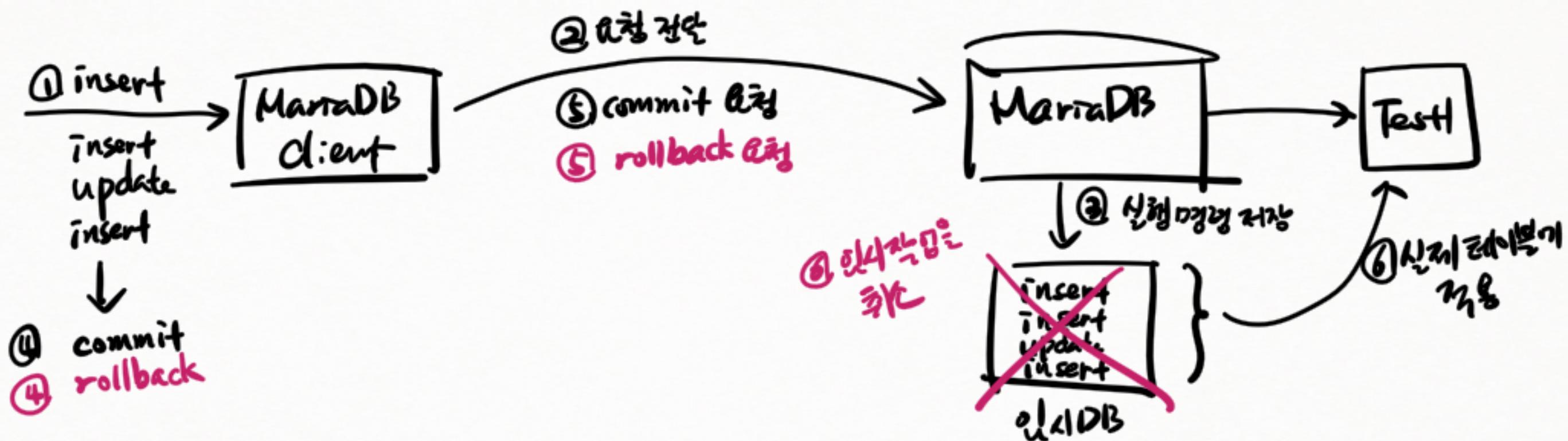
= 인공적 번호를 만들어
PK로 사용한다
↳ ex) 일련번호

* Commit

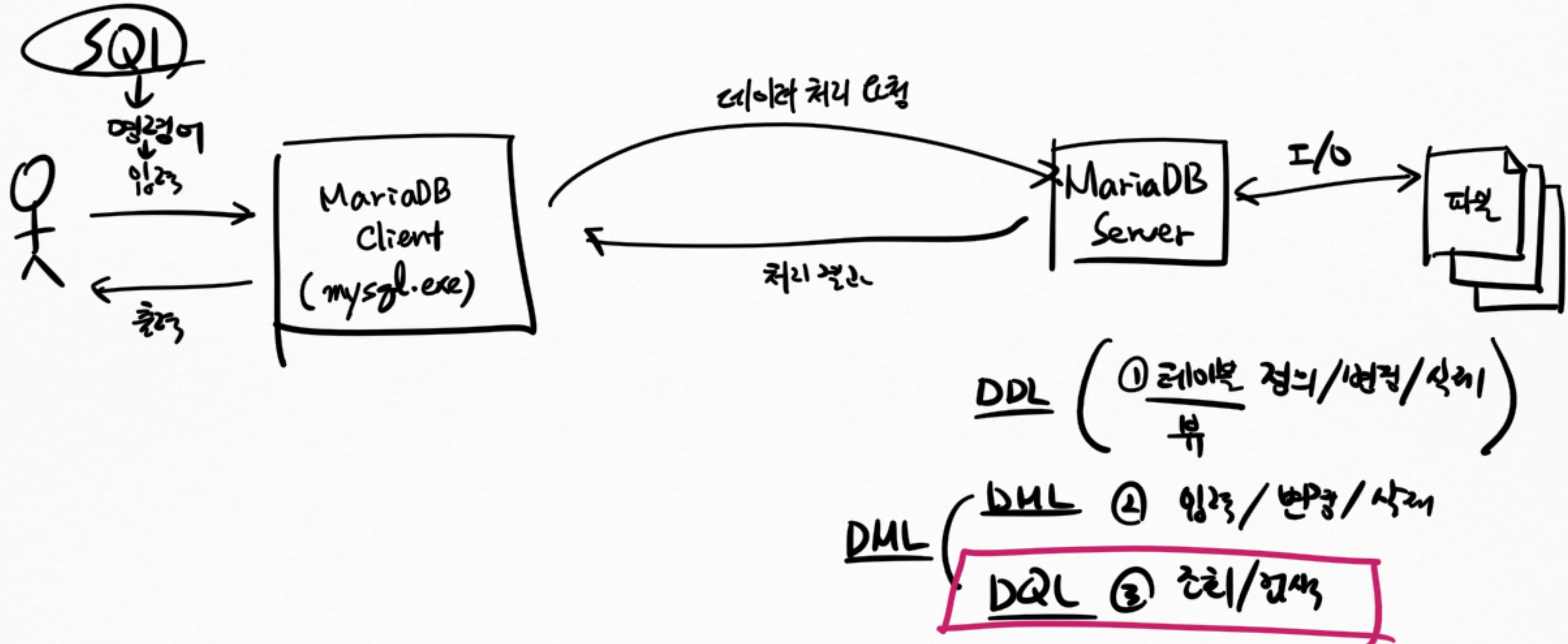
① autocommit = true



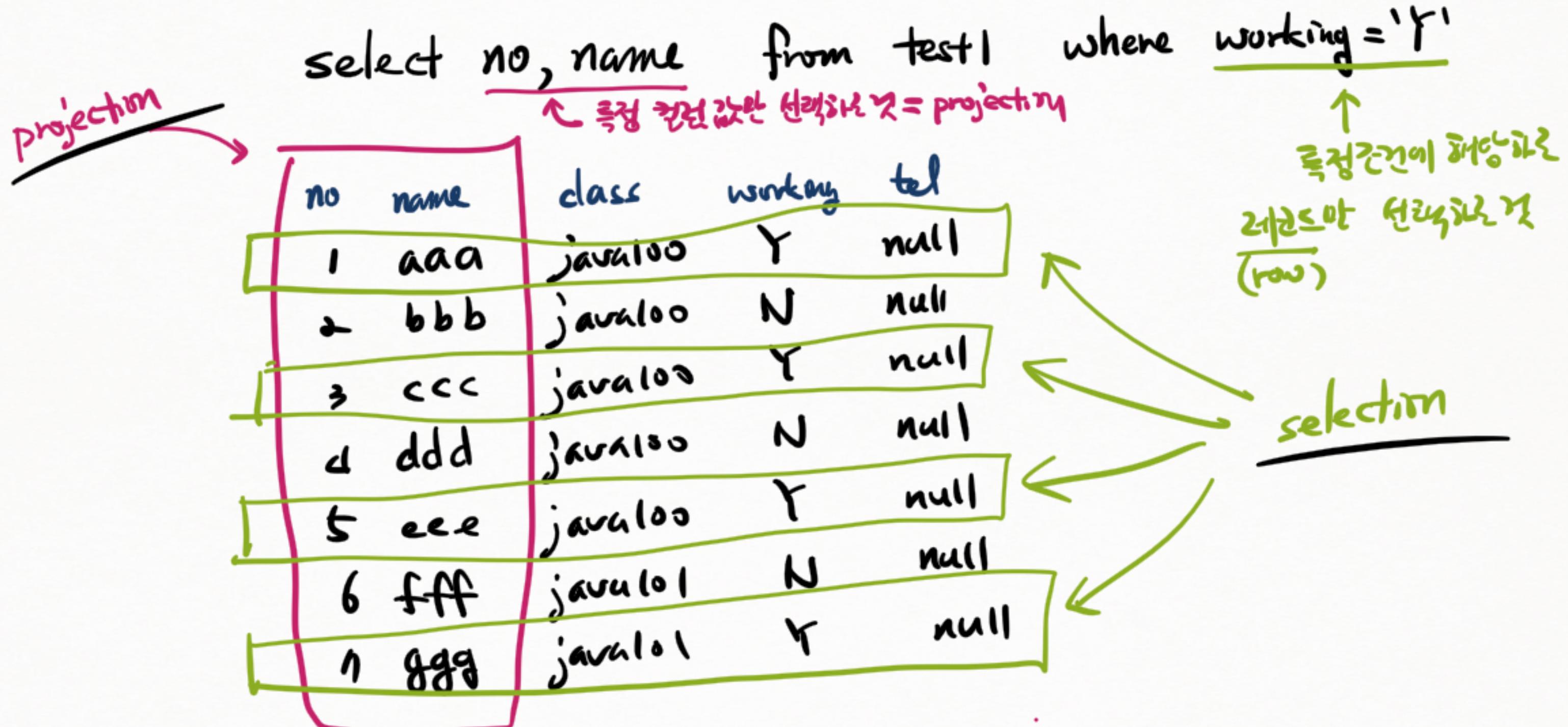
② autocommit = false



* select

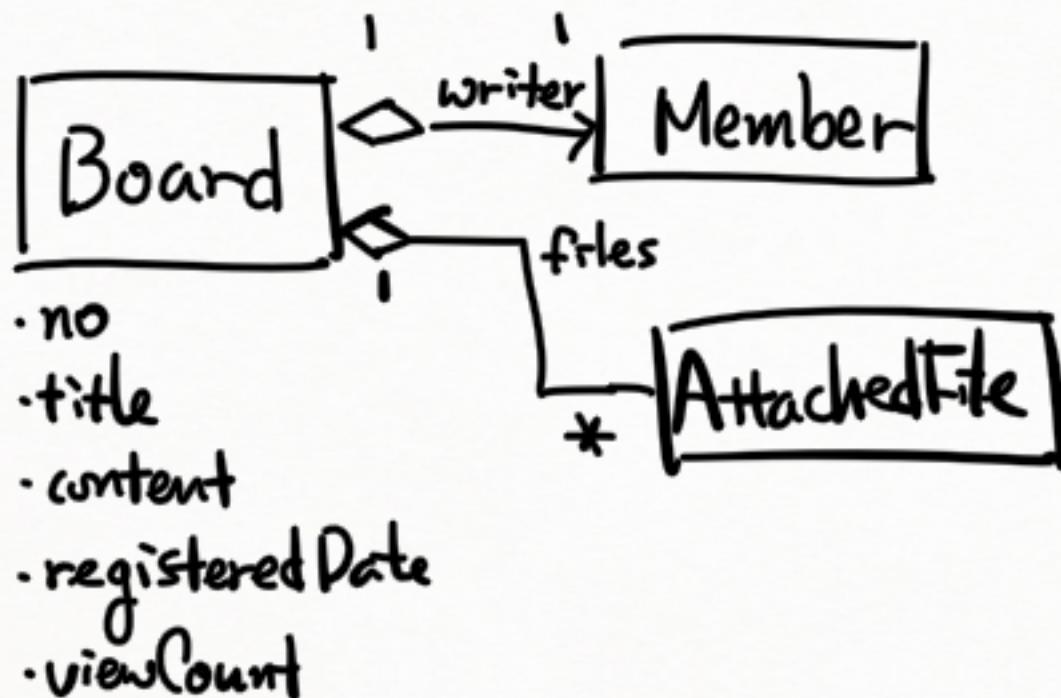


* selectin / projection



* Foreign Key

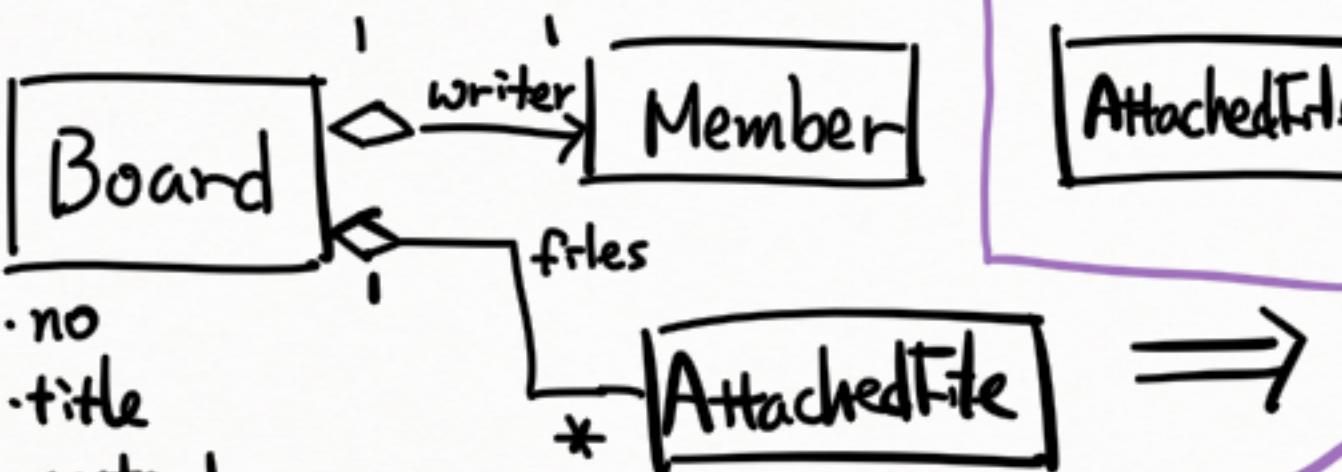
① 자식 가족



```
class Board {
    int no;
    String title;
    String Content;
    Date registeredDate;
    int viewCount;
    Member writer;
    List<AttachedFile> files;
}
```

* Foreign Key

① 자식 키체



- no
- title
- content
- registeredDate
- viewCount

② 테이블 간의 관계 (ER-Diagram; ERD)



- 1번 게시글
 - 2번 게시글
 - 3번 게시글
- 100번 조회수
 - 101번 댓글수
 - 102번 유저수

ERD 특징:

① Information Engineering Notation

* 객체 간의 관계와

테이블 간의 관계는 다르다!

아마히 일부는 관계화시키는 데로
관계를 설정.

프로그래밍 언어
위한 끝으로 객체간의
관계는 설정.

공략

↓
정권으로 일은 사실을
학문으로 체계화나마사
연구하는 것

↓
해당 분야에 종사하고 수업들이
수행되는 줄 알게 드러온다

* Foreign Key (외부키) - 다른 테이블의 pk 참조



(PK) 번호	파일명	제작자(FK)	(PK) 번호	제작자	제작자(FK)	(PK) 번호	이름
11	aaa.gif	101	101	aaaa	1	1	홍길동
12	bbb.gif	102	102	bbbb	1	2	김꺽정
13	ccc.gif	102	103	cccc	3	3	유재석
14	ddd.gif	103	104	dddd	4	4	안우진
15	eee.gif	103					

* 커먼 쿵복

Board							
no	title	content	rdt	f1	f2	f3	f4
1	aaa	—	—	a.gif	b.gif		
2	bbb	—	—	x.gif	y.gif	z.gif	t.gif
3	ccc	—	—	m1.gif	m2.gif	m3.gif	m4.gif
4	ddd	—	—	m5.gif			

이렇게 같은 종류의 파일은 여러 개 저장하기 위해 커먼 쿵복으로 선언된 경우

첨부파일이 없음에도 5개의 커먼이 존재하기 때문에 메모리 낭비

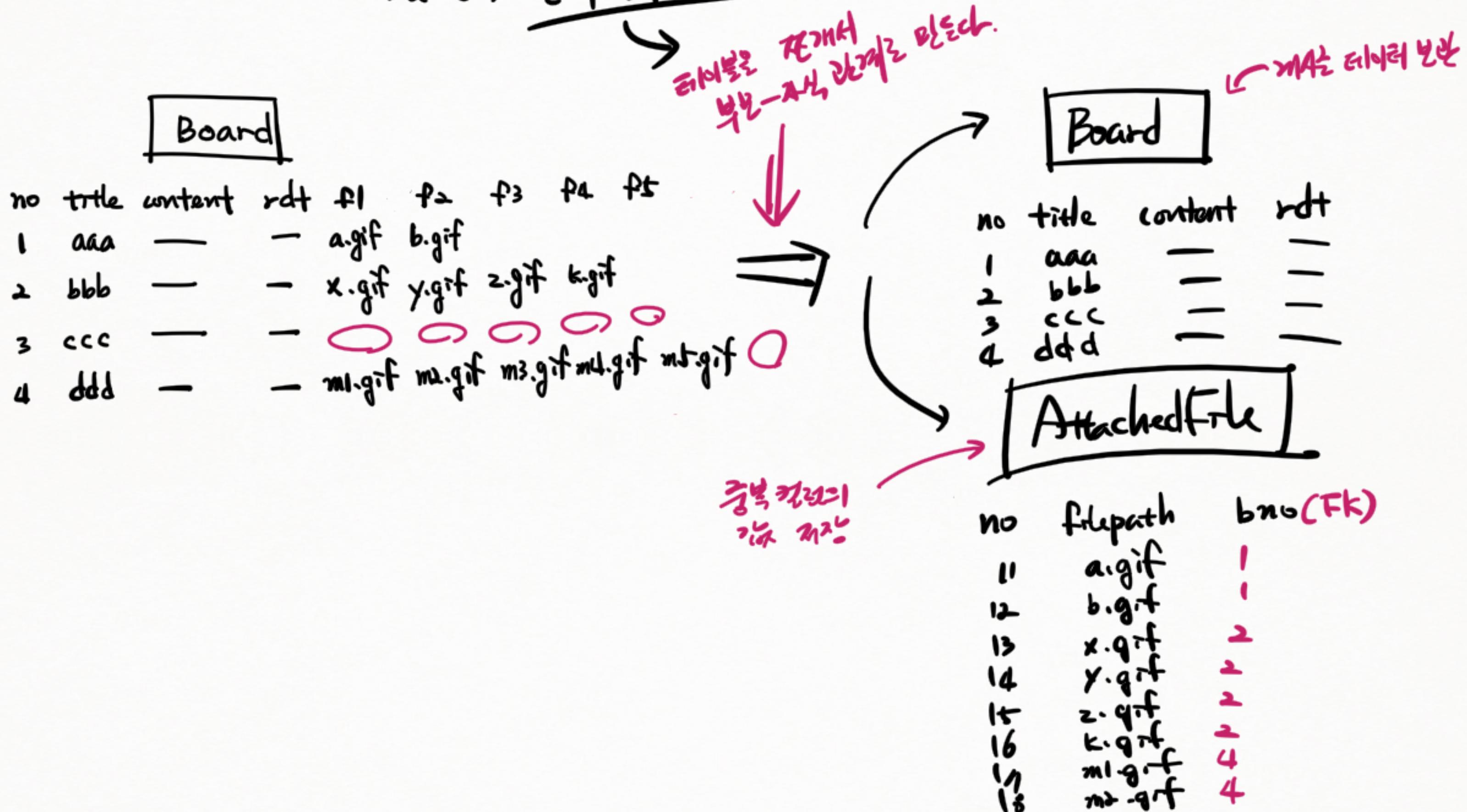
커먼이 부족하여 첨부파일을 더 저장할 수 있다

다음의 위치 발생

이전 커먼 쿵복이나 레이어 쿵복 등에 허용

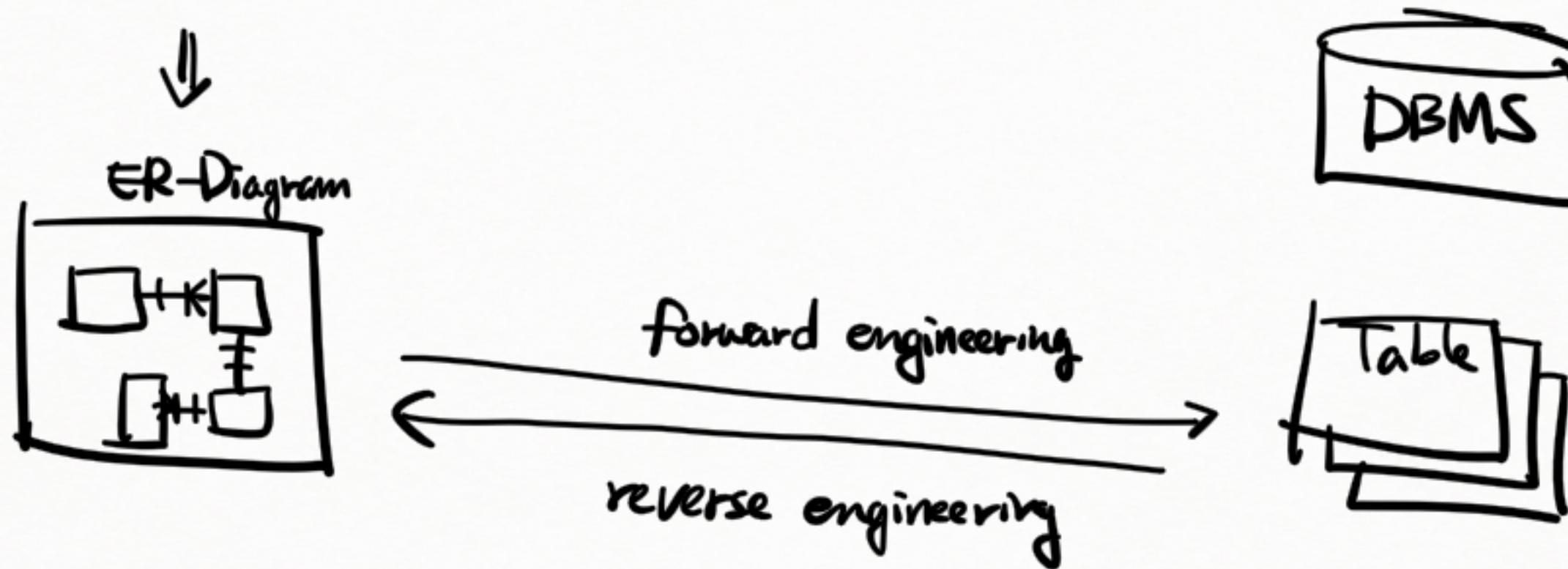
장기화 수행!

* 첨부 중복 문제 해결



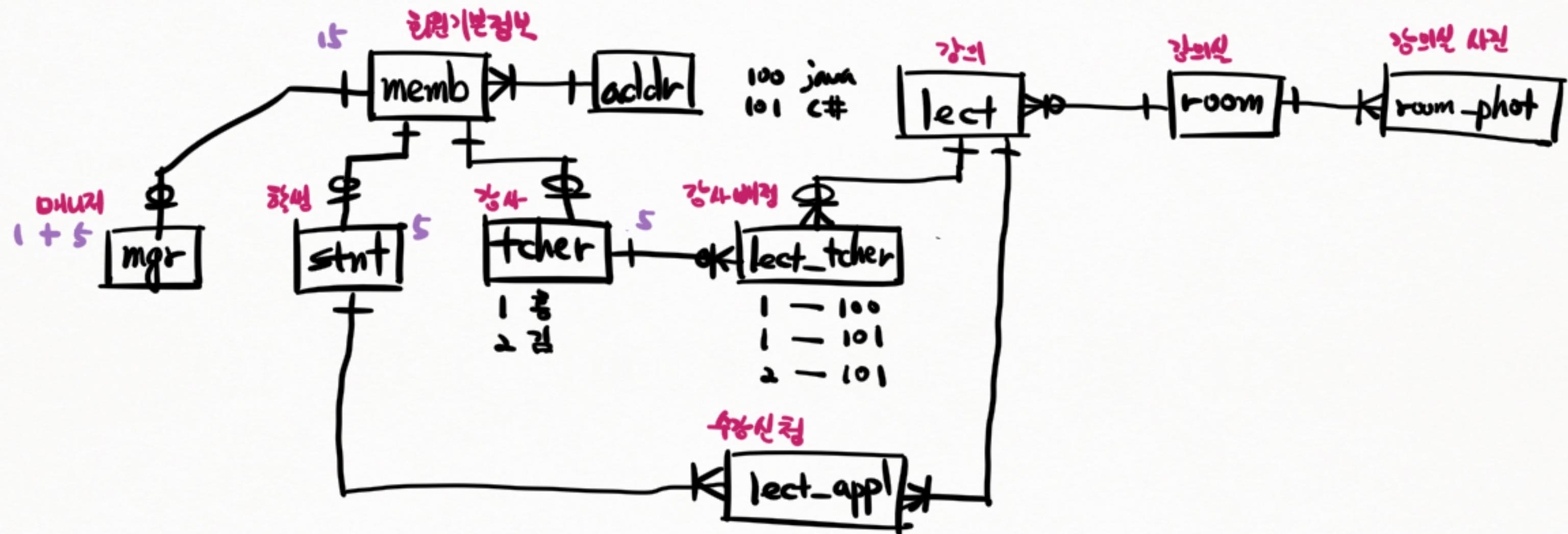
* ERD : forward / reverse engineering

제작ツ : ER-Win, eXerd, ...



* SQL 조인 테스트를 위한 데이터의 ERI

예) 교육센터 관리 시스템



* 조인

↙ 무관 1:1 조인

① cross 조인 (카리선 조인)

A	B
---	---

aaa	111
bbb	222
ccc	



aaa 111

aaa 222

bbb 111

bbb 222

ccc 111

ccc 222

② natural 조인

↑ 같은 이름의 키를 같은
기준으로 조인

Board	AttachedFile	Board	AttachedFile
-------	--------------	-------	--------------

bno	title	file	path	bno	title
1	aaa	11	a.gif	1	aaa
2	bbb	12	b.gif	1	bbb
3	ccc	13	c.gif	3	ccc
4	ddd	14	d.gif	4	ddd



1	aaa	11	a.gif	1
1	aaa	12	b.gif	1
3	ccc	13	c.gif	3
4	ddd	14	d.gif	4



{ ✓ 조인의 기준이 될 키가 암시되지 않은 경우
 { ✓ 서로 상반되는 키의 이름이 있는 경우

FK 키의 이름

PK + 가리키는
PK 키의 이름

↳ 올바른 조인이 이루어지지 못한다

* 왜 데이터를 가져올 때 여러 테이블의 데이터를
엮어서 가져온가?


데이터 중복을 피하기 위해
데이터가 여러 테이블에
포함되어 있기 때문이다.

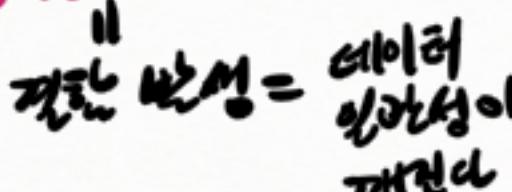
Project		
번호	제목	책장번호
1	aaa	<u>kim 010-1111-1112</u>
2	bbb	<u>kim 010-1111-1112</u>
3	ccc	<u>kim 010-1111-1111</u>

데이터 중복 문제
해결

데이터를
복제

 데이터가 중복

- 변경이 번거롭다
- 번경항목을 누락할 수 있다

 = 데이터
중복성이
생긴다

같은 책장인데
번호변경
아니다?

Project		
번호	제목	책장번호
1	aaa	aaa
2	bbb	bbb

User		
번호	이름	전화
1	kim	010-1111-1112
2	park	010-1111-2222
3	lee	010-1111-3333

- 번호 11은 전화
- 1 kim 010-1111-1112
 - 2 park 010-1111-2222
 - 3 lee 010-1111-3333

* Natural 조인을 수행할 기본 조건의 이론이
일치하지 않거나 엉뚱한 조건과 일치할 경우

③ join on ~ 문법 사용

Board join AttachedFile
on Board.no = AttachedFile.bno

조인 조건

* Natural 조인을 수행할 때

기준컬럼이 이름이 같은 것뿐 아니라 여러 개 있을 경우

Board	
bno	title
1	aaa
2	bbb

AttachedFile	
no	name
11	x.a.gif
12	x.b.gif
13	c.gif

no	name	bno
11	x.a.gif	1
12	x.b.gif	1
13	c.gif	2



Board join AttachedFile
using (bno)

↑
준인할 때 사용하는
기준 컬럼은
명시적으로 지정한다.

* 조인을 수행할 때 테이터가 누락되는 상황

Lecture				Teacher			
no	name	tno(fk)	(pk)tno		name		
11	aaa	1		1	kim		
12	bbb	2		2	park		
13	ccc	1		3	lee		
14	ddd			4	eom		
15	eee						

Lecture right outer join Teacher
on Lecture.tno = Teacher.tno

11	aaa	1	1	kim
12	bbb	2	2	park
13	ccc	1	1	kim
null	null	null	3	lee
			4	eom

① Natural Join

Lecture join Teacher using (tno)

11	aaa	1	1	kim
12	bbb	2	2	park
13	ccc	1	1	kim

→ 조인할 때 데이터의 티아리가 없으면
결과에서 누락되는 문제가 발생

② Outer Join ↴

* Lecture left outer join Teacher
on Lecture.tno = Teacher.tno

11	aaa	1	1	kim
12	bbb	2	2	park
13	ccc	1	1	kim
14	ddd		null	null
15	eee		null	null

* 주인 레스트