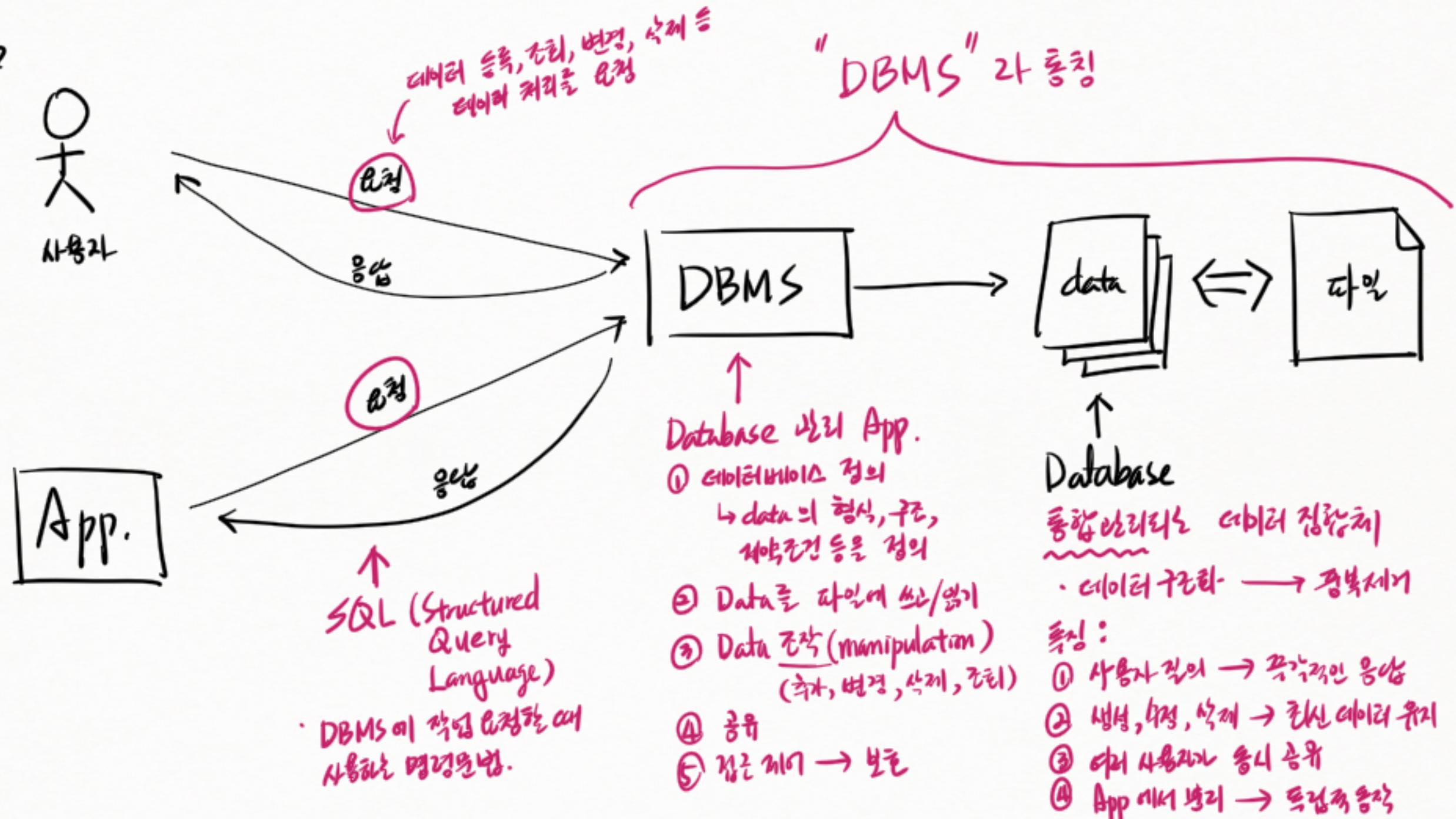


DBMS  
Database  
Management  
System

## \* Database, DBMS, SQL

★ DBMS 를 사용하는 이유?

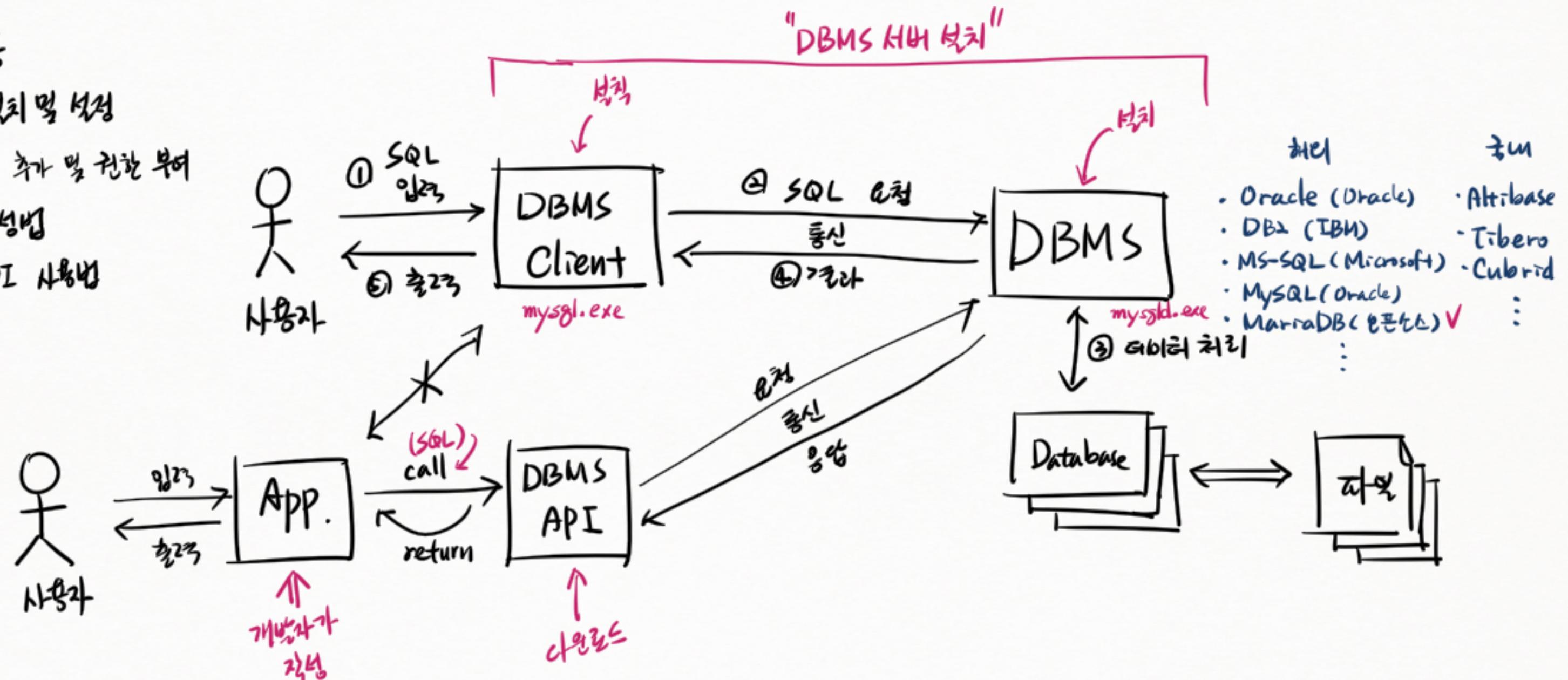
- 직접 파일 I/O 을  
프로그래밍 하기  
불편한 점 있음.
- 데이터를 차례대로  
관리하는게 번거롭다
- 프로그래밍 언어에  
생각보다  
밀접한(종속된) 명령으로  
작동할 수 있음.



## \* DBMS 핵심

### \* 핵심 내용

- ✓ ① DBMS 설치 및 설정
- ✓ ② Database 추가 및 권한 부여
- ✓ ③ SQL 작성법
- ④ DBMS API 사용법



SQL      Structured  
              Query  
              Language

## \* SQL

↳ DBMS에 데이터 처리 명령을 네트 채 사용하는  
명령을 표준 언어

## \* 흔한 SQL

= SQL 표준 언어 + DBMS 특수 언어  
⇒ DBMS 외 SQL 표준 언어 사용 가능  
⇒ App 작동할 때  
DBMS에 맞춰  
SQL은 사용할 수 있다.

SQL

### DDL (Data Definition Language)

↳ 데이터, 뷰, 프로시저, 트리거 등을  
정의하는 언어, DDL

### DML (Data Manipulation Language)

↳ 데이터를 데이터를 삽입, 수정, 삭제 등 데이터 조작

### DQL (Data Query Language)

↳ 데이터의 내용을 조회

} DML이나  
DQL

\*  $\text{entity} \approx \text{table}$   
column = attribute

name	kor	eng	math	sum	aver

row = record = tuple

## \* 진수와 decimal

(Java)

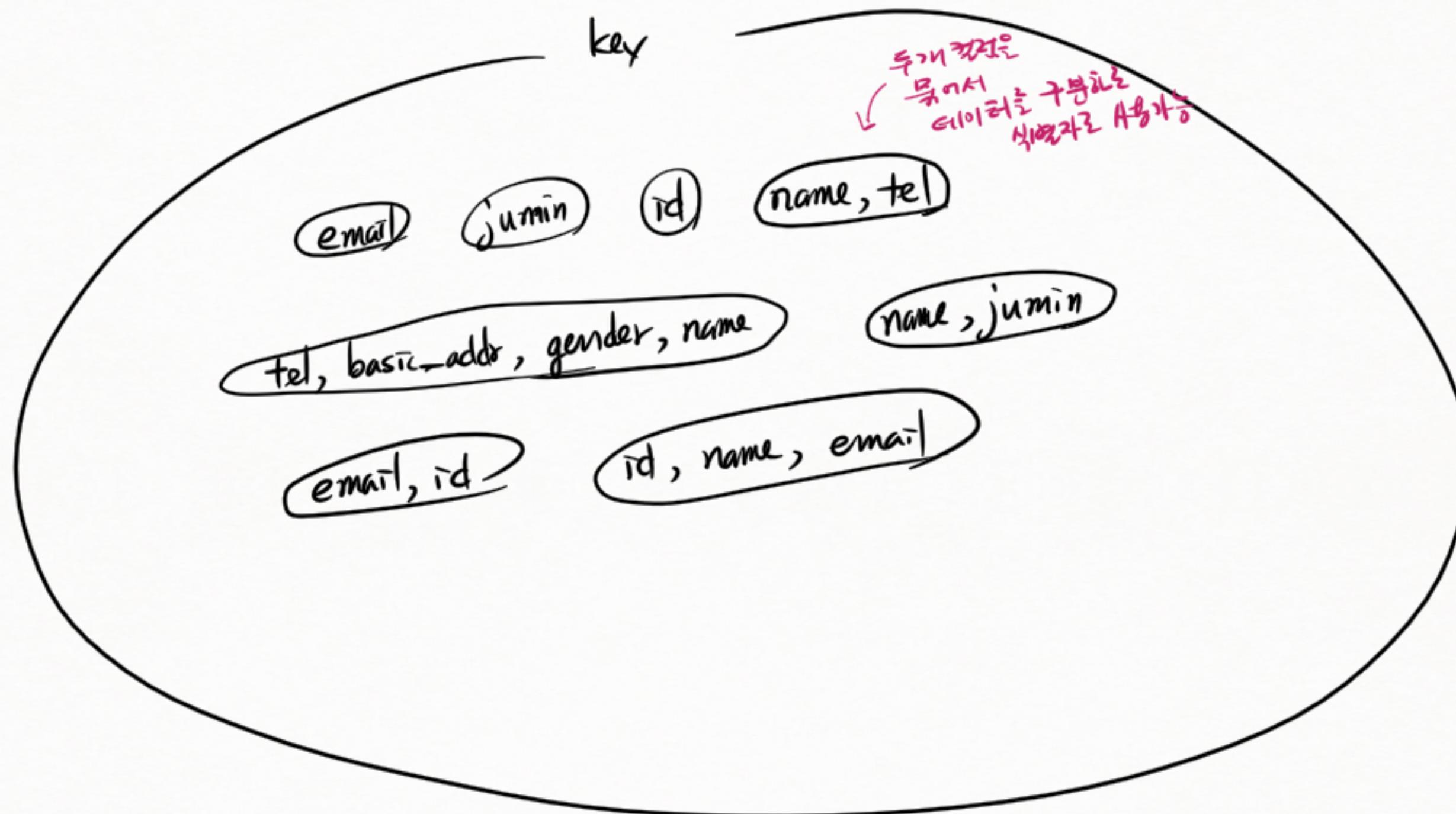
	진수	10진수
8진수	017	15
10진수	17	17
16진수	0x17	23
2진수	00010111	23

12 ← 정수

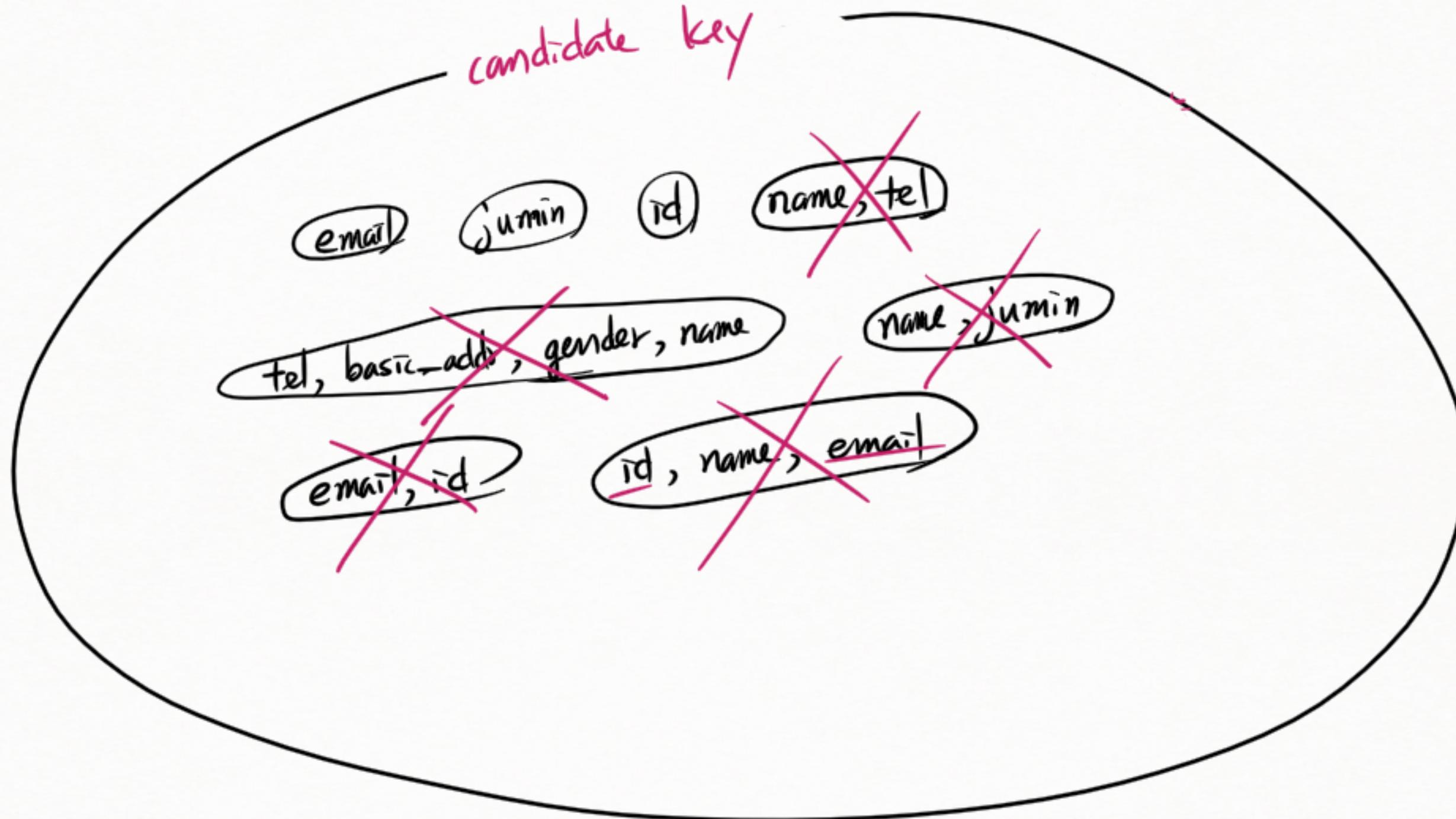
3.14 ← 부동소수점

} 10진수  
} decimal

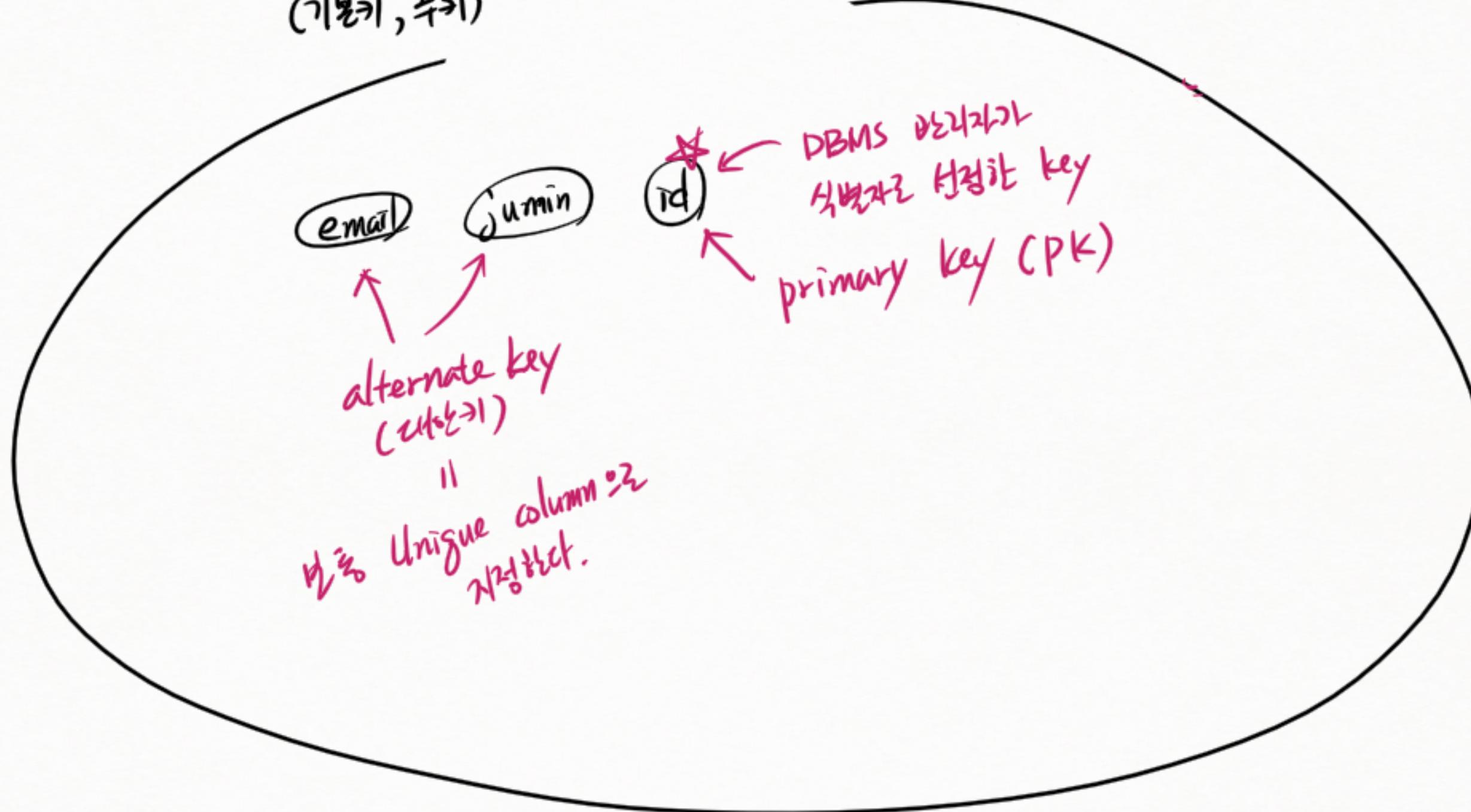
\* key : 데이터를 구분할 때 사용하는 ключ값



\* candidate key (후보키, 주어지기): 키가 여러개인 키

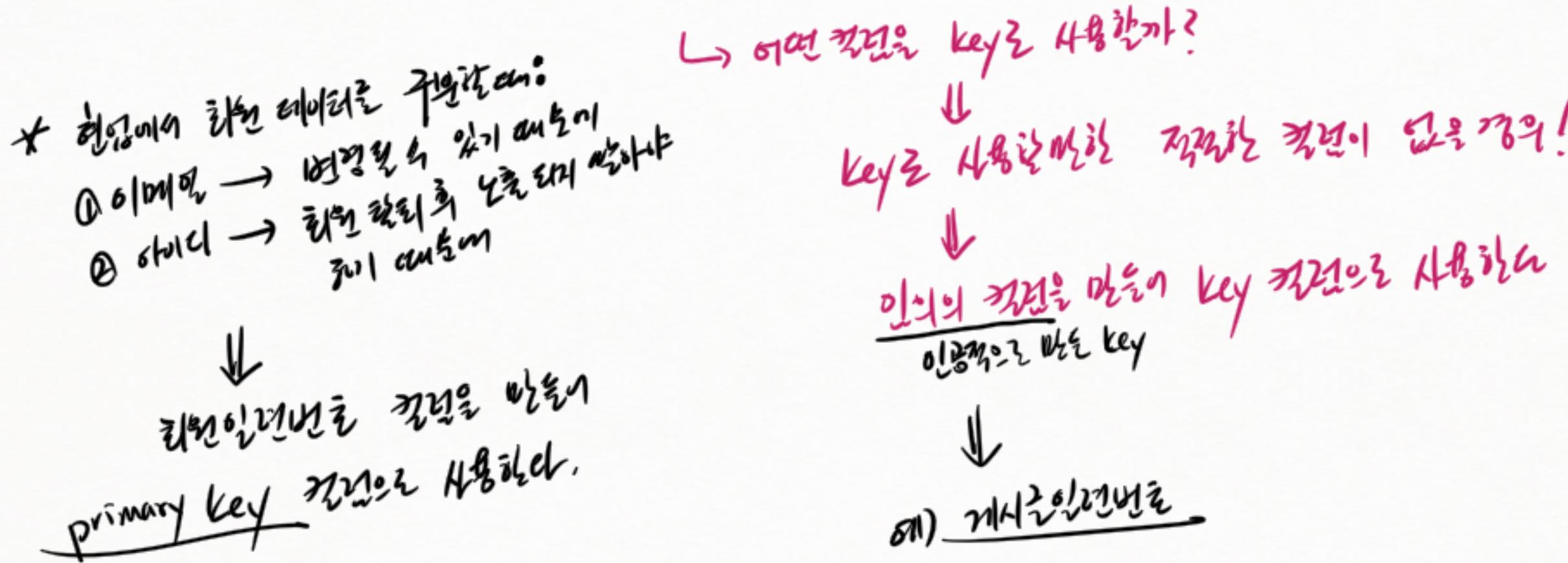


\* Primary Key: 주보기 중에서 고유값이 설정되는  
(기본키, 주기)



## \* artificial key (인공키)

Board 테이블: 세종, 세종, 애산, 조희숙, 장성호



기사

PK  
no

인공키  
번호  
PK 기본키  
설정

name, age

alternate key

PK는 아님  
PK로 충분하지  
않다  
unique 컬럼으로  
지정해야 한다.

기금 암호 중에는 여러 가지 작업을 같은 단계로 다해야 하는 경우가 있다  
예) [초안 → 결재]  $\Rightarrow$  주는 암호 commit

# Transaction

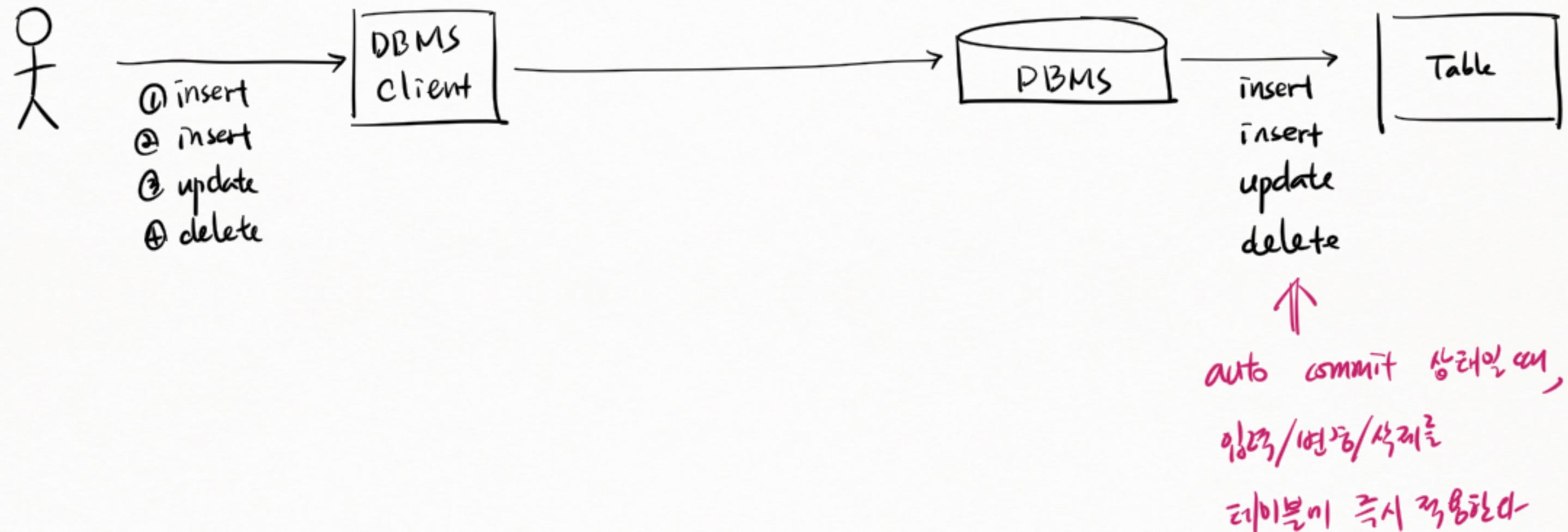
↳ 여러 모의 데이터 변경 작업을 한 번에 끝내는 것

```

graph TD
    A[작업] --> B[insert]
    A --> C[update]
    A --> D[delete]
    A --> E[데이터변경작업]
    E --> F[작업처리]
    E --> G[결과]
    F --> H[작업]
    F --> I[결과]
    G --> J[작업처리]
    G --> K[결과]
    J --> L[결과]
    K --> M[작업처리]
  
```

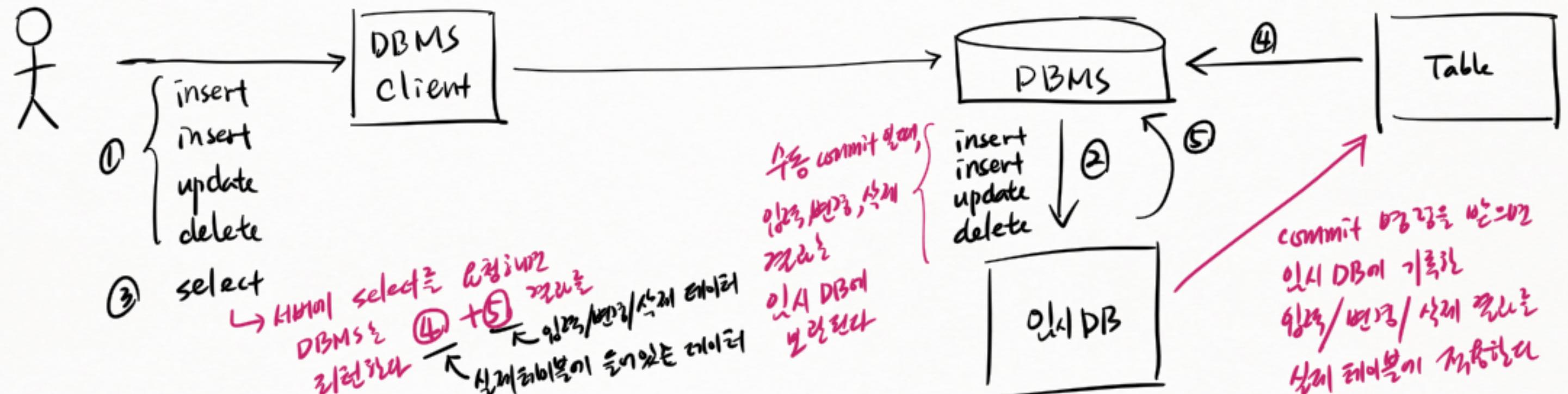
\* commit / rollback

① auto commit 상태



## \* commit / rollback

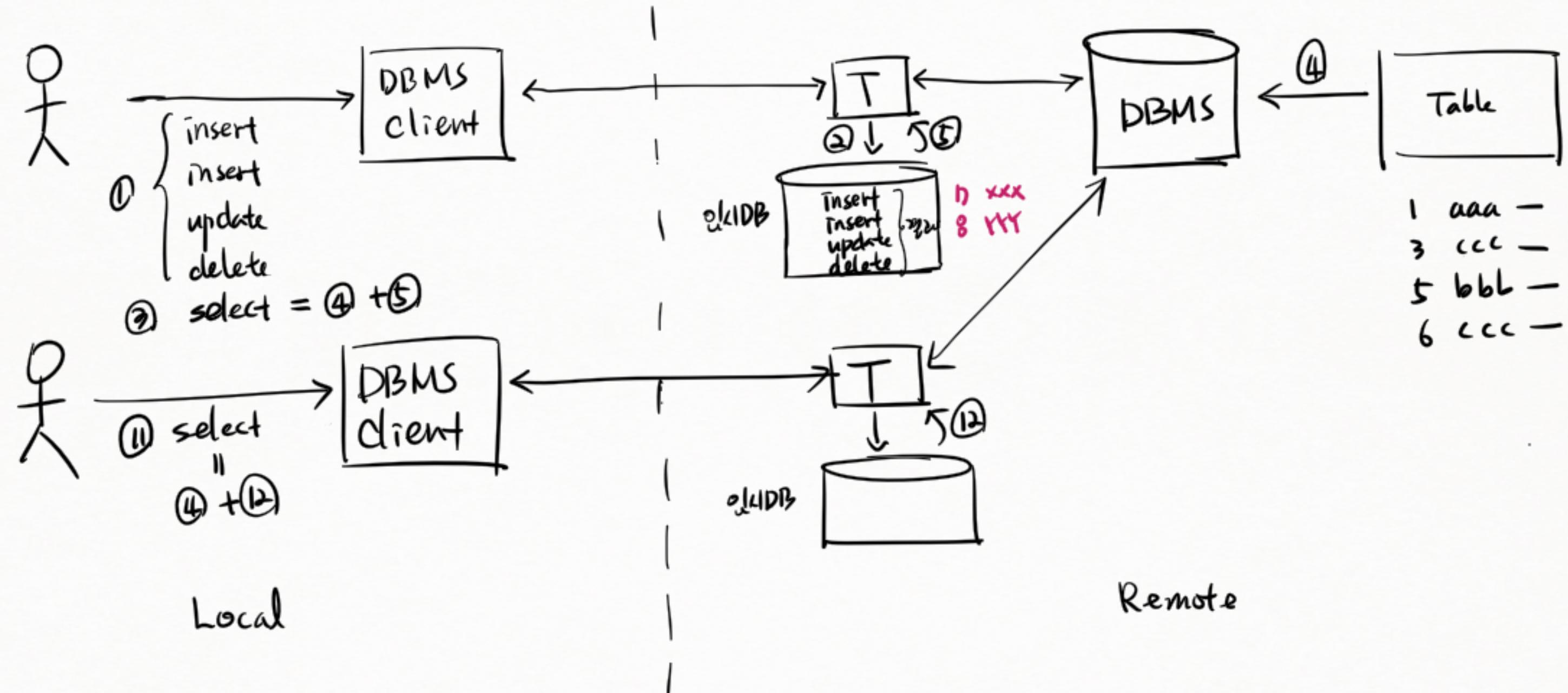
② 提交 commit  $\Rightarrow$  set autocommit = false;



예) 1 aaa —  
3 ccc —  
5 bbb —  
6 ccc —

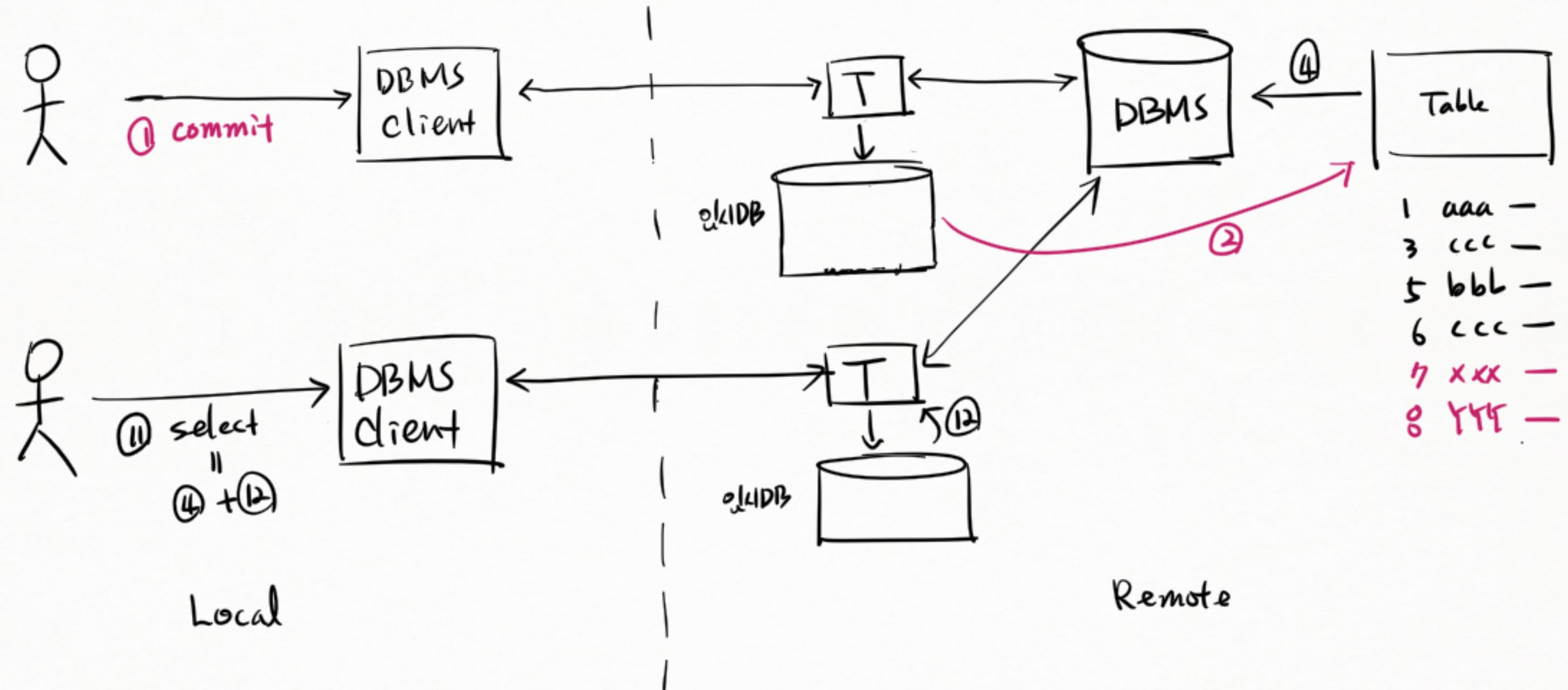
\* commit / rollback  $\Rightarrow$  더 자세히!

② 사용 commit  $\Rightarrow$  set autocommit = false;



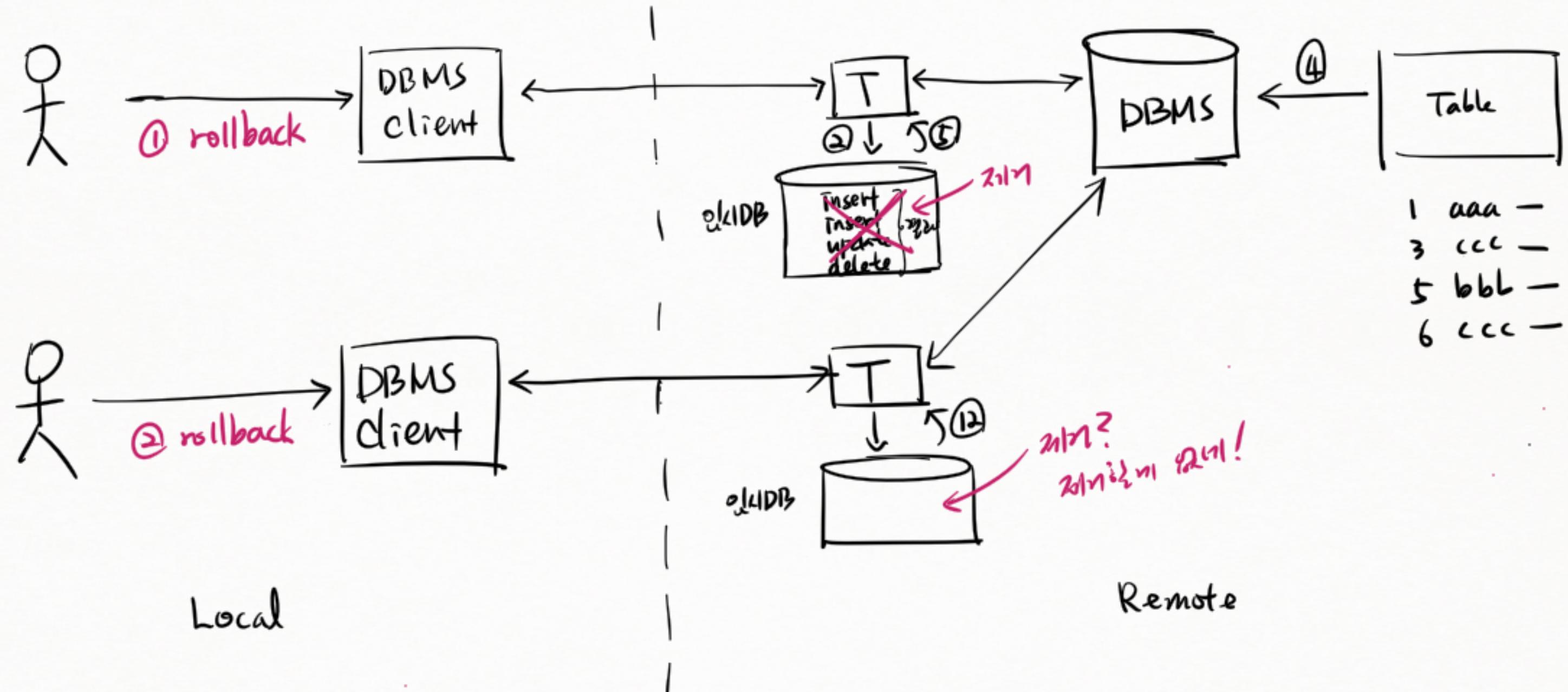
\* commit / rollback => 더 자세히!

③ commit → 임시DB에 기록된 결과를 실제 데이터베이스에 적용해줌.



\* commit / rollback

④ rollback → olcIDB ni yuxi hig tzu'i xilika.



\* select : projection / selection  
projection : 추출할 컬럼 선택

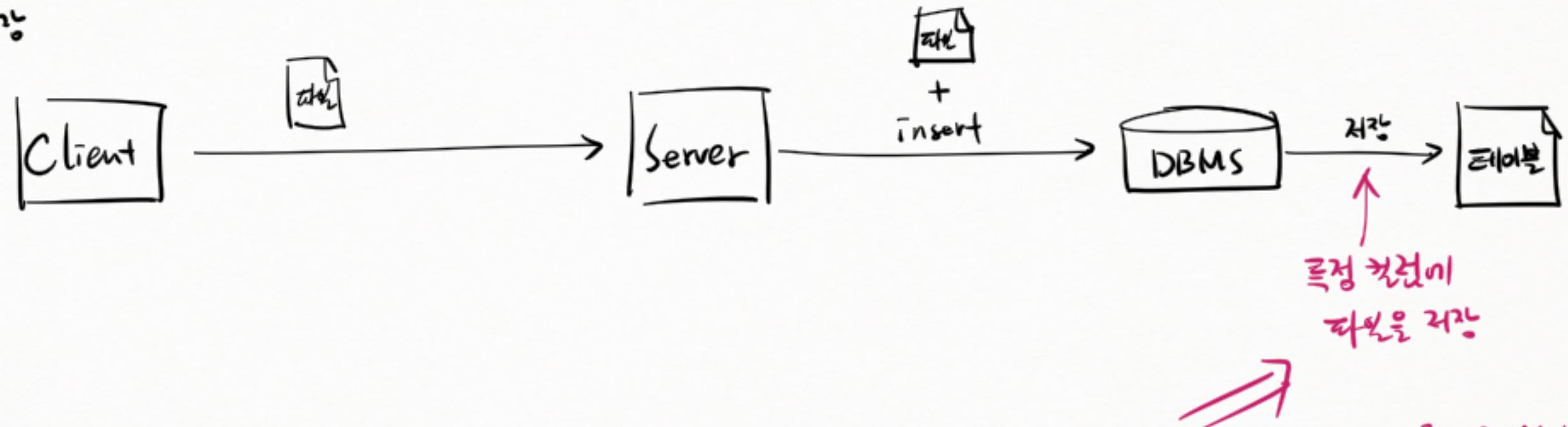
select no, name, tel  
from test1  
where working = 'Y'  
↑  
selection

no	name	class	tel	working
				Y
				Y
				N
				Y
				N
				N

selection  
↓  
추출할 데이터 선택

## \* 첫부파일 저장

### ① DBMS에 파일저장

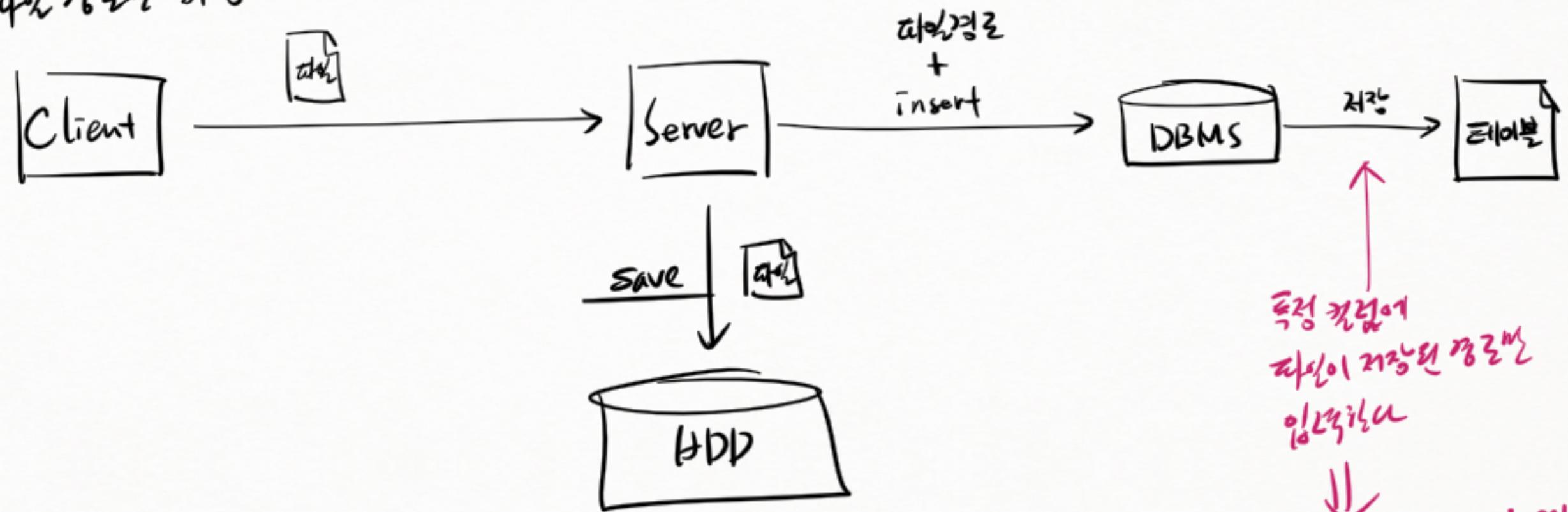


- ① 파일을 만들 때 DBMS를 경유하기 때문에 속도가 느리진다.
- ② 데이터베이스의 크기를 급격하게 늘린다.

↓  
비효율적!

## \* 첫부파일 저장

② DBMS에 파일 경로는 없음



특정 컴퓨터에  
파일이 저장된 경로는  
없도록 한다

- ↓
- ① 파일리버레이스가 접근할 때  
커뮤니케이션을 한다.
  - ② 파일은 OS가 읽기 대로 쓰기  
된다면 되는데.

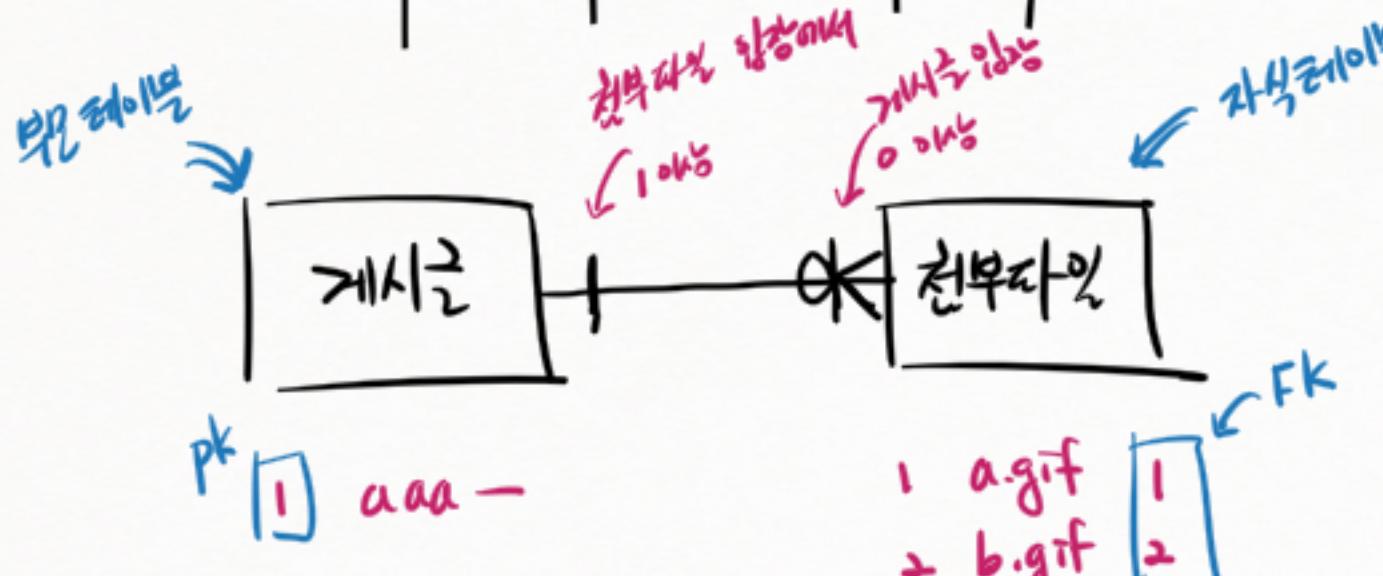
## \* 첨부파일 3. 테이블

\* 메시지를 데이터화한 첨부파일 데이터를 갖는다

메시지 테이블

<u>PK</u> no	title	content	rdt
1	aaa	-	-
2	bbb	-	-
3	ccc	-	-
4	ddd	-	-

= table = entity



Entity Relationship Diagram (ERD)  
(ER Diagram) - IE 학습

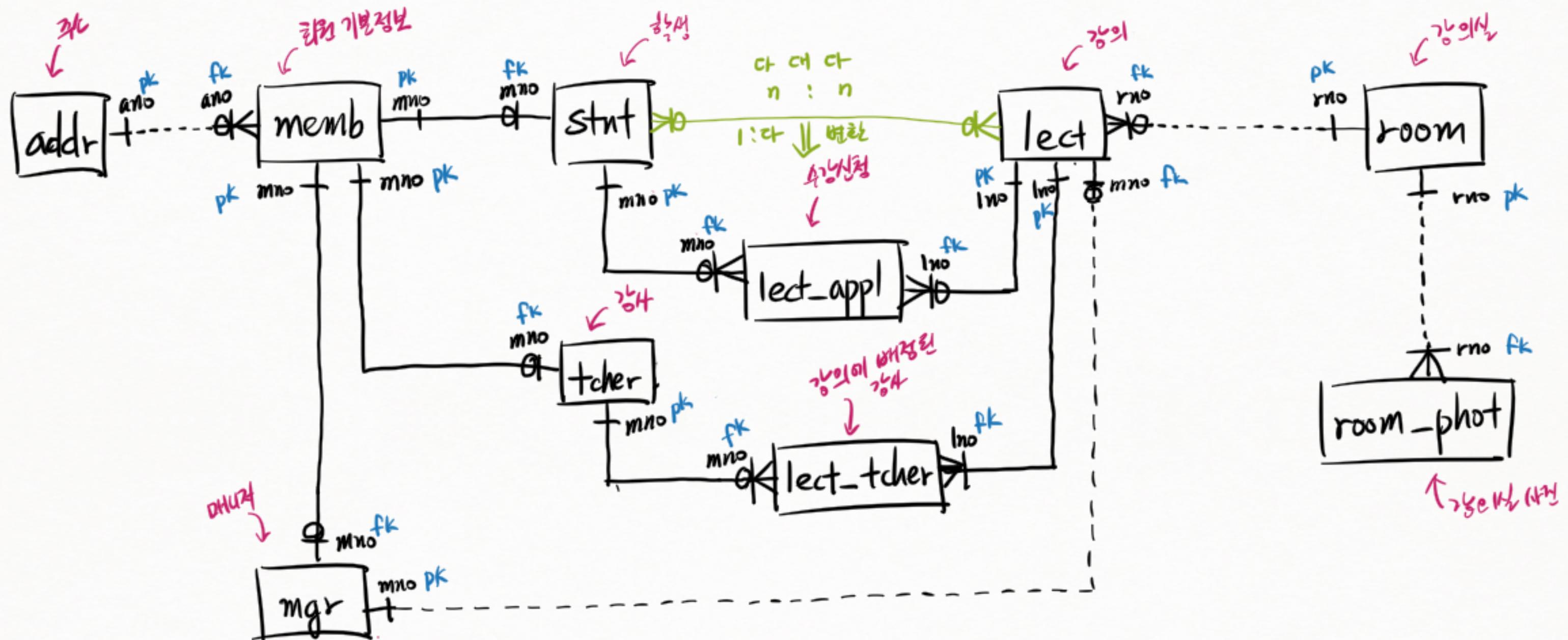
첨부파일 테이블

<u>PK</u> fno	filepath	bno
1	a.gif	1
2	b1.gif	2
3	b2.gif	2
4	b3.gif	2
5	d1.gif	4
6	d2.gif	4
7	d3.gif	4
8	d4.gif	4
9	d5.gif	4
10	d6.gif	4
11	d7.gif	4

Foreign key

다른 테이블 키값을  
갖는다는 특징.

\* 데이터베이스 주인 : 데이터베이스 ERD



\* 테이블 관계 : addr +--- memb

{ auto-increment  
artificial key(인공키)

addr  
pk  
ano

psn-no bas-addr

1 11111 강남구 -

2 11112 서초구 -

3 11113 광진구 -

FK = PK

(identifying)

식별되는

인식되는

(non-identifying)

FK ≠ PK

memb

PK  
ano

name tel ...

1 허경동 111

2 이재경 222

3 유승우 3333

4 양승근 444

fk  
ano

det-addr ...

1 613-1102

2 " 103-703

3 123-2207

4 " -

5 " -

6 " -

7 " -

8 " -

9 " -

10 " -

11 " -

12 " -

13 " -

14 " -

15 " -

16 " -

17 " -

18 " -

19 " -

20 " -

21 " -

22 " -

23 " -

24 " -

25 " -

26 " -

27 " -

28 " -

29 " -

30 " -

31 " -

32 " -

33 " -

34 " -

35 " -

36 " -

37 " -

38 " -

39 " -

40 " -

41 " -

42 " -

43 " -

44 " -

45 " -

46 " -

47 " -

48 " -

49 " -

50 " -

51 " -

52 " -

53 " -

54 " -

55 " -

56 " -

57 " -

58 " -

59 " -

60 " -

61 " -

62 " -

63 " -

64 " -

65 " -

66 " -

67 " -

68 " -

69 " -

70 " -

71 " -

72 " -

73 " -

74 " -

75 " -

76 " -

77 " -

78 " -

79 " -

80 " -

81 " -

82 " -

83 " -

84 " -

85 " -

86 " -

87 " -

88 " -

89 " -

90 " -

91 " -

92 " -

93 " -

94 " -

95 " -

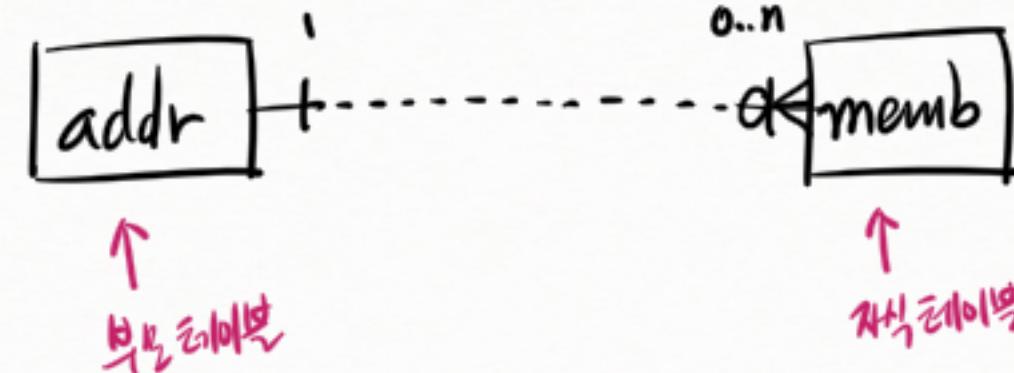
96 " -

97 " -

98 " -

99 " -

100 " -



1 : 0 까지 1	0
1 : 1	1
1 : 0 이상	0, 1
1 : 1 이상	1, 2
단계 차수에 따른 초기값	

\* 테이블 관계 : stnt  $\rightarrow$  memb

회원 등록에서  
학생 학번의 추가 정보를  
저장하는 테이블

	stnt				memb							
PK	mno	work	acc-no	bank	PK	mno	name	tel	...	ano	det-addr	...
1	Y	-	-		1	홍길동	111	2	서울대학교	101-102		
2	N	-	-		2	이재현	222	2	"	103-903		
4	Y	-	-		3	유성우	333	1	경기도아파트	123-2207		
					4	양승근	444	1	"	-		

FK = PK

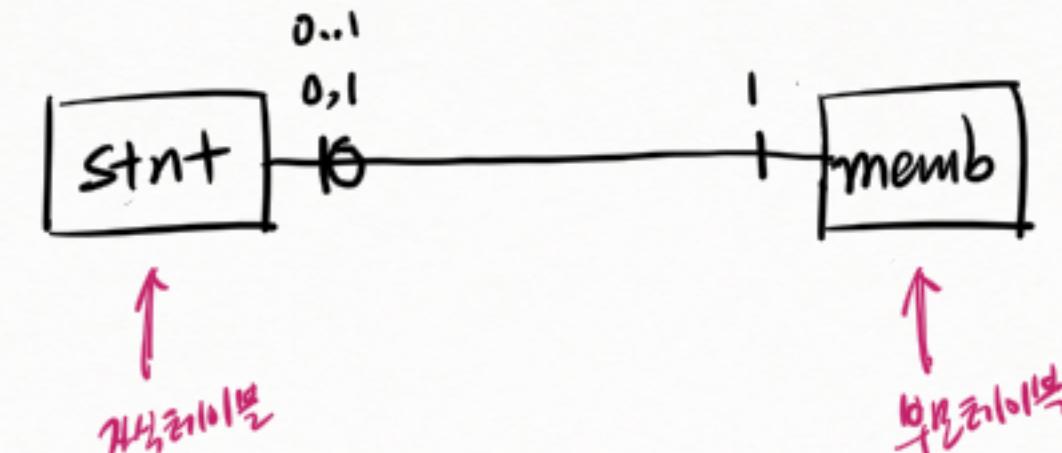
(identifying)

식별되는

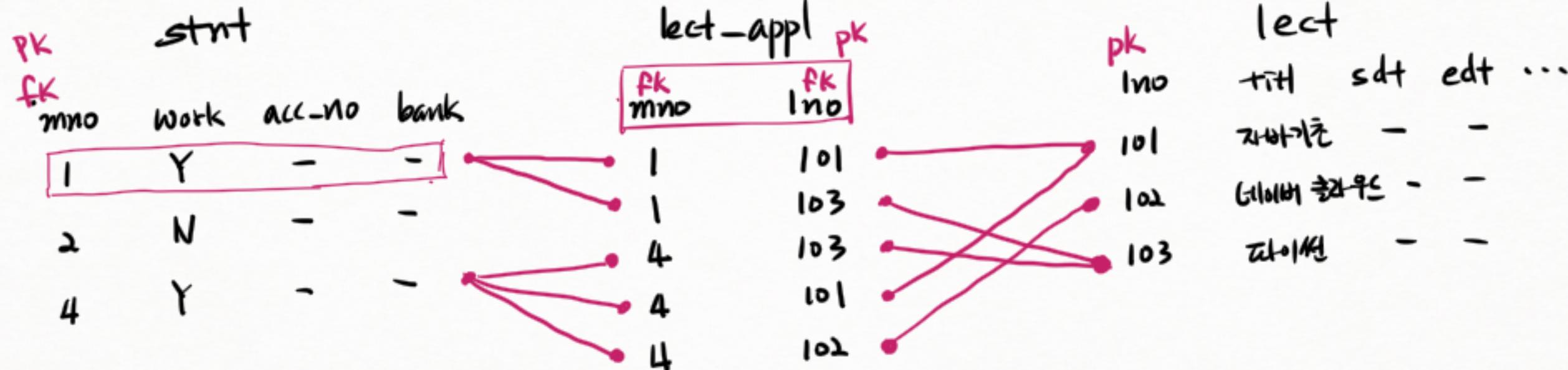
인식되는

(non-identifying)

FK  $\neq$  PK



\* 테이블 관계 : stnt  $\rightarrow$  lect  
다 대 다 관계



$FK = PK$   
(identifying)

식별관련

비식별관련

(non-identifying)

$FK \neq PK$

관계 테이블 : 두 테이블의 관계를 저장한다. 보통 다대다 관계를  
개선하기 위해 만든다.



## \* DB 모델링

DB 모델링



(데이터를 잘 관리)  
(데이터 간의 관계를 정의한다)



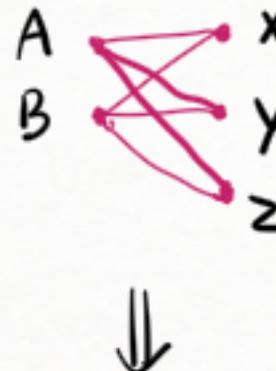
데이터 중복 제거.



- ① 데이터 관리가 힘들다
  - ↳ 여러 곳에 같은 데이터가 저장
  - ↳ 변경하는데 여러 곳을 변경해야 한다
- ② 변경누락 발생 → 데이터 결함 발생

## \* JOIN

① cross join (= cartesian join) ② natural join



A X

A Y

A Z

B X

B Y

B Z

.

	board	attach-file	
bno	title	fno path	bno
1	aaa	100 a.gif 1	
2	bbb	101 b.gif 1	
		102 c.gif 2	

같은 이름을 가진 컬럼의 값을 기준으로  
두 테이블의 데이터를 합친다.

1	aaa	100	a.gif	1
1	aaa	101	b.gif	1
2	bbb	102	c.gif	2

③ join ~ on

	board	attach-file	
no	title	no path	bno
1	aaa	100	a.gif 1
2	bbb	101	b.gif 1
		102	c.gif 2

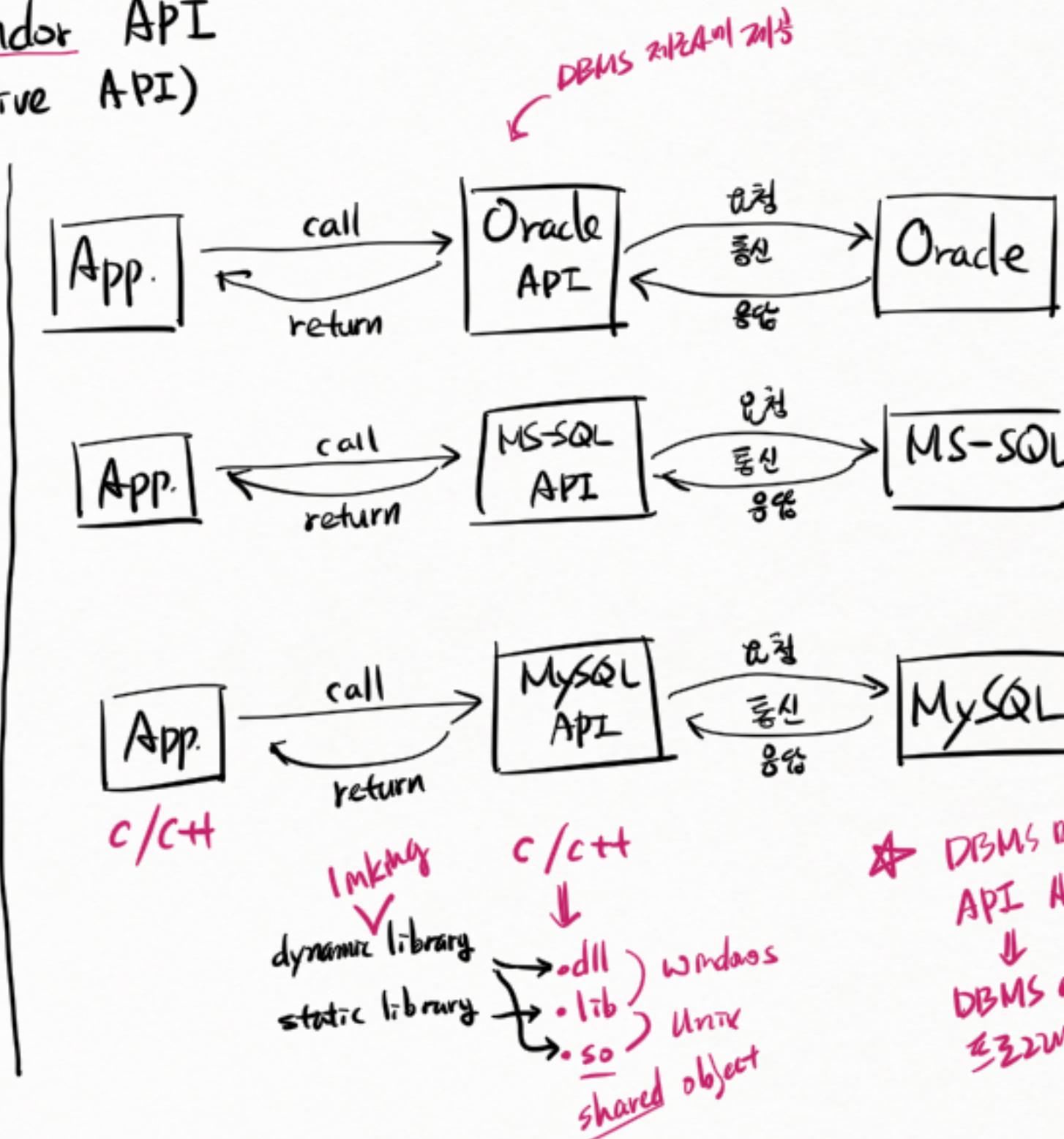
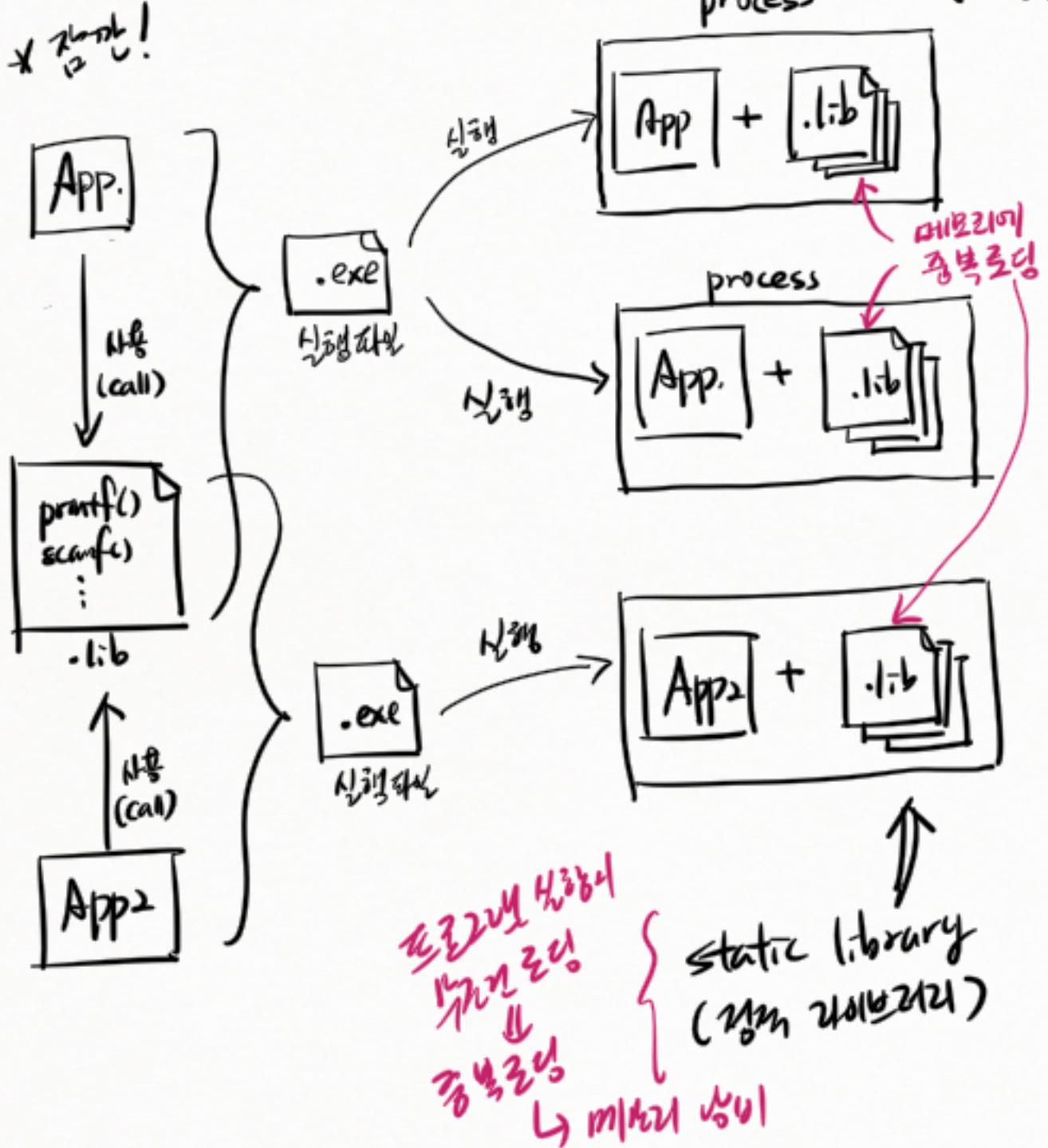
두 테이블의 데이터를 조인하는 기본 컬럼의  
이름이나 다른 경우 사용하고 유용  
(pk 커런티영 ≠ fk 커런티영)

1	aaa	100	a.gif	1
1	bbb	101	b.gif	1
2	bbb	102	c.gif	2

## JDBC API

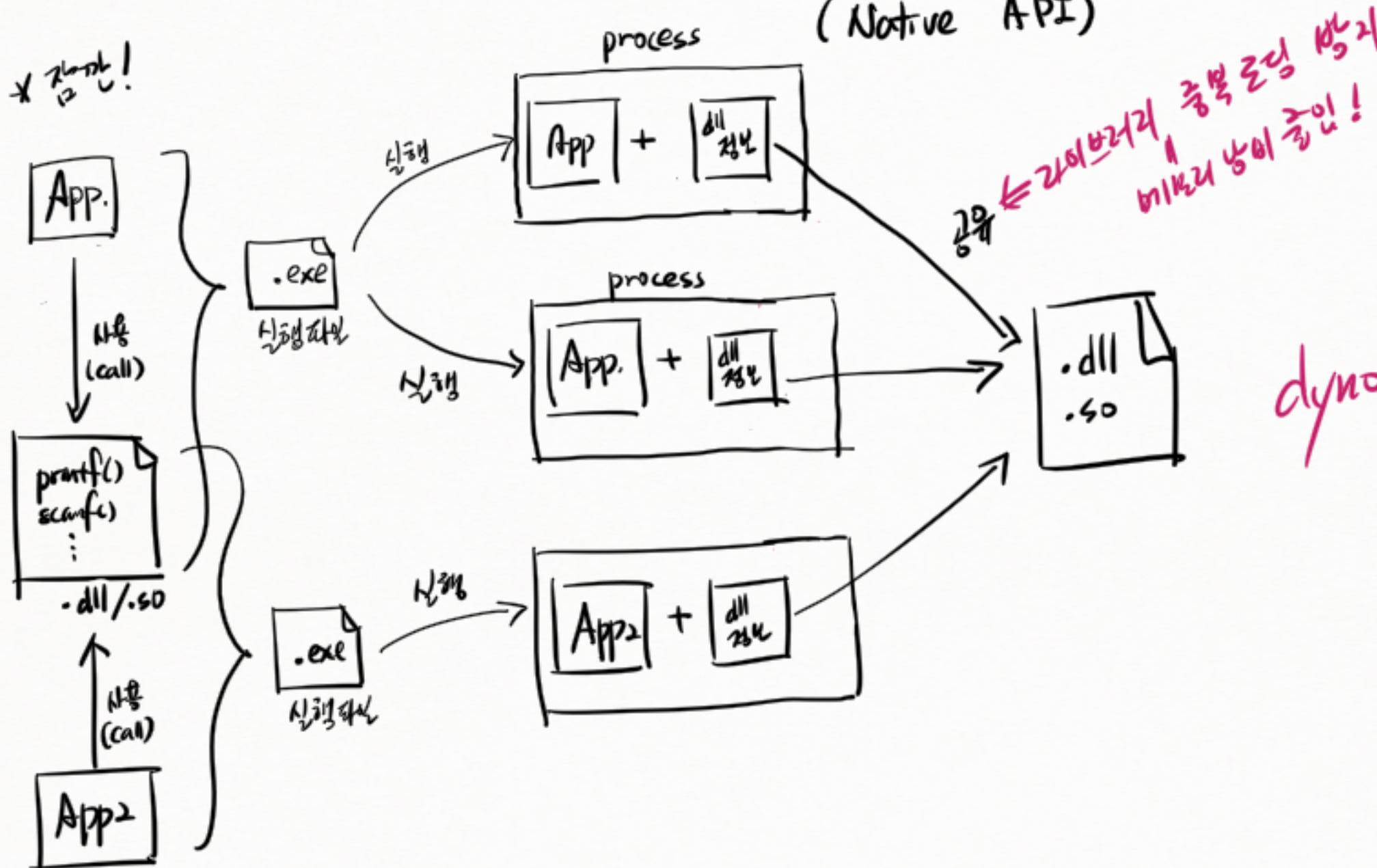
<u>Java</u>	자바
<u>Database</u>	데이터베이스
<u>Connectivity</u>	연결

## \* DBMS API : Vendor API (Native API)



\* DBMS API  
API 사용법이 다릅!  
 ↓  
 DBMS API 사용법  
 예제는 각각 다르다.

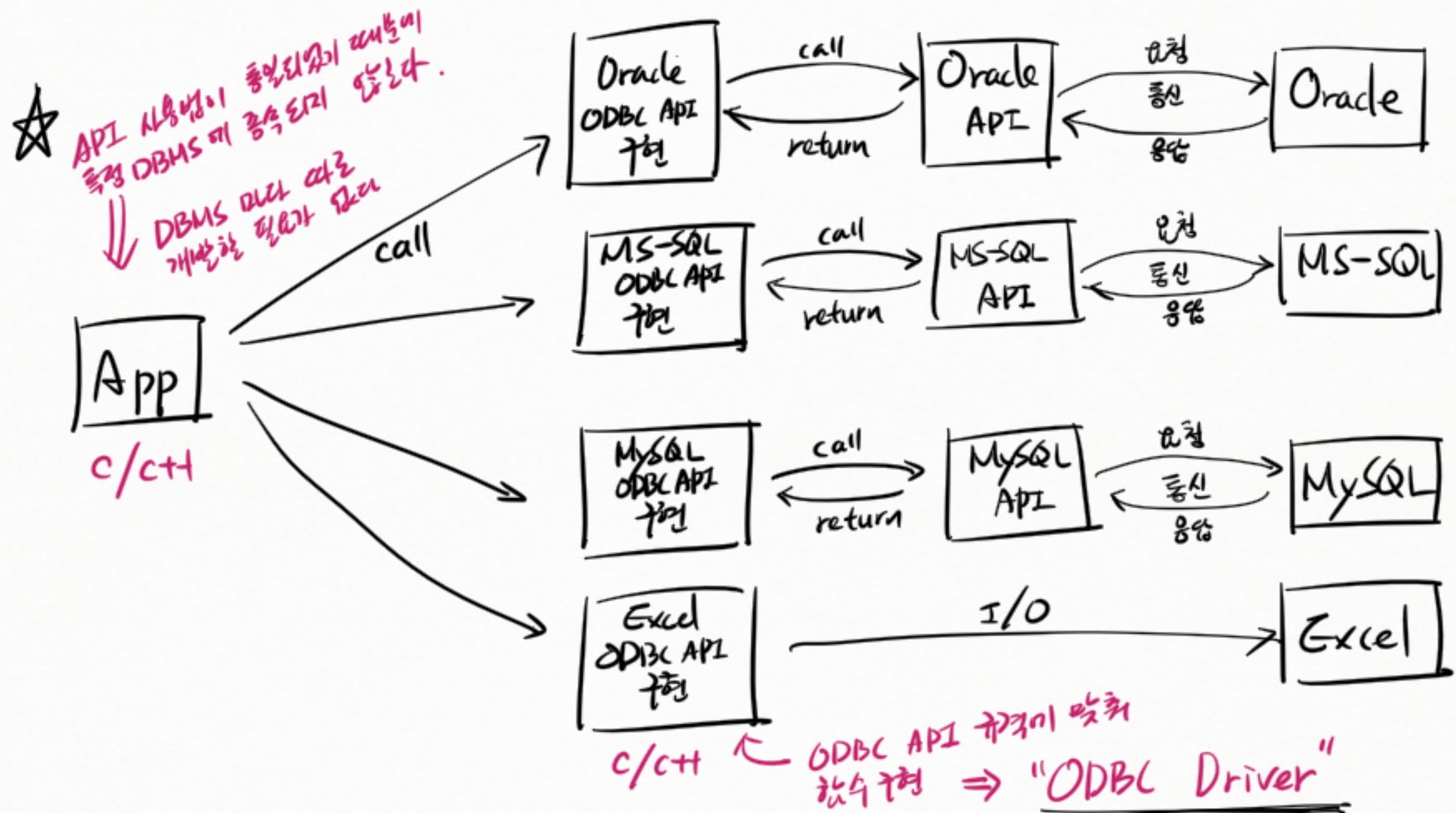
\* DBMS API : Vendor API  
process (Native API)



dynamic library  
↳ 어떤 App. 을 실행할 때  
한 번 로딩되면,  
다른 App. 은 그에 맞는  
그냥 해당한다  
↑  
중복 로딩되지 않는다!

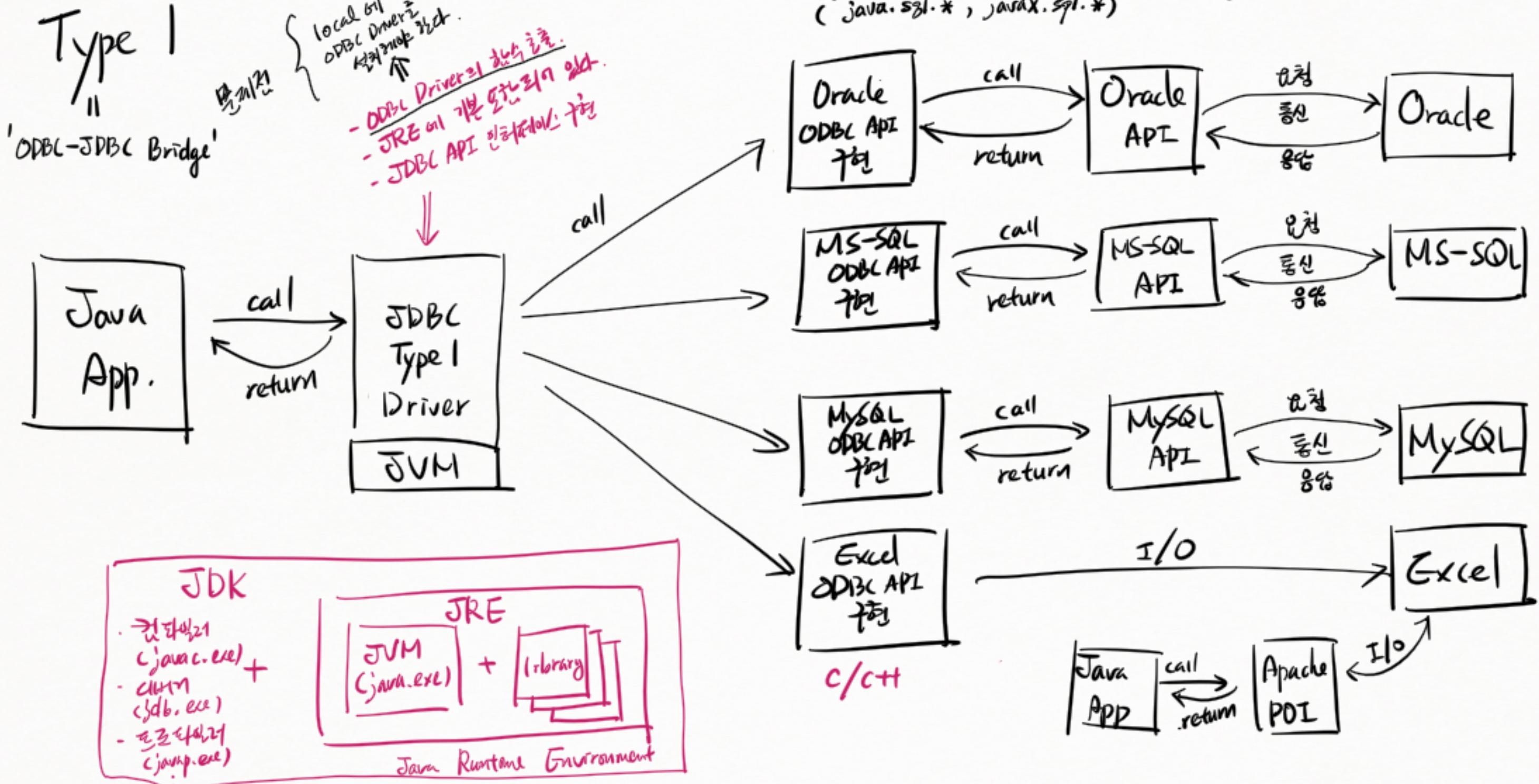
\* DBMS API : ODBC API 형태  $\xrightarrow{\text{구현}}$  ODBC Driver  
.dll/.lib/.so  
 ↳ DBMS API 구현체

## Open Database Connectivity



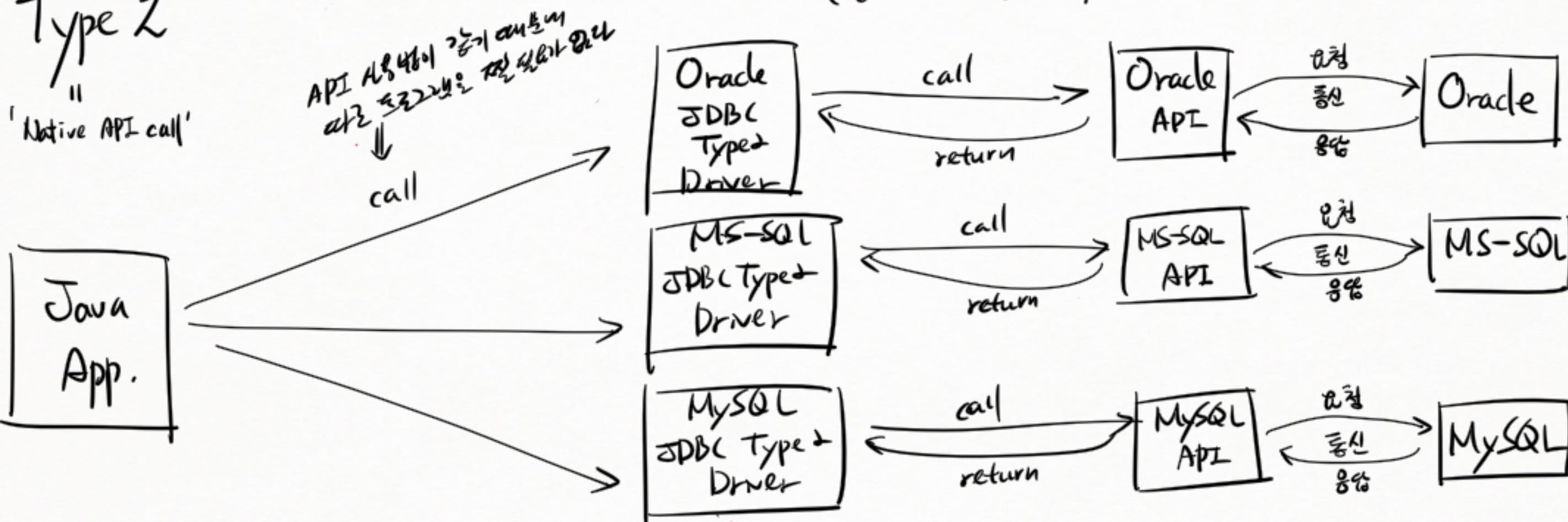
\* DBMS API : JDBC API  
 Java  
 인터페이스  
 ( java.sql.\* , javax.sql.\* )

→ JDBC Driver  
 클래스



\* DBMS API : JDBC API  
 Java 인터페이스  
 ( java.sql.\* , javax.sql.\* ) → JDBC Driver  
 클래스

Type 2

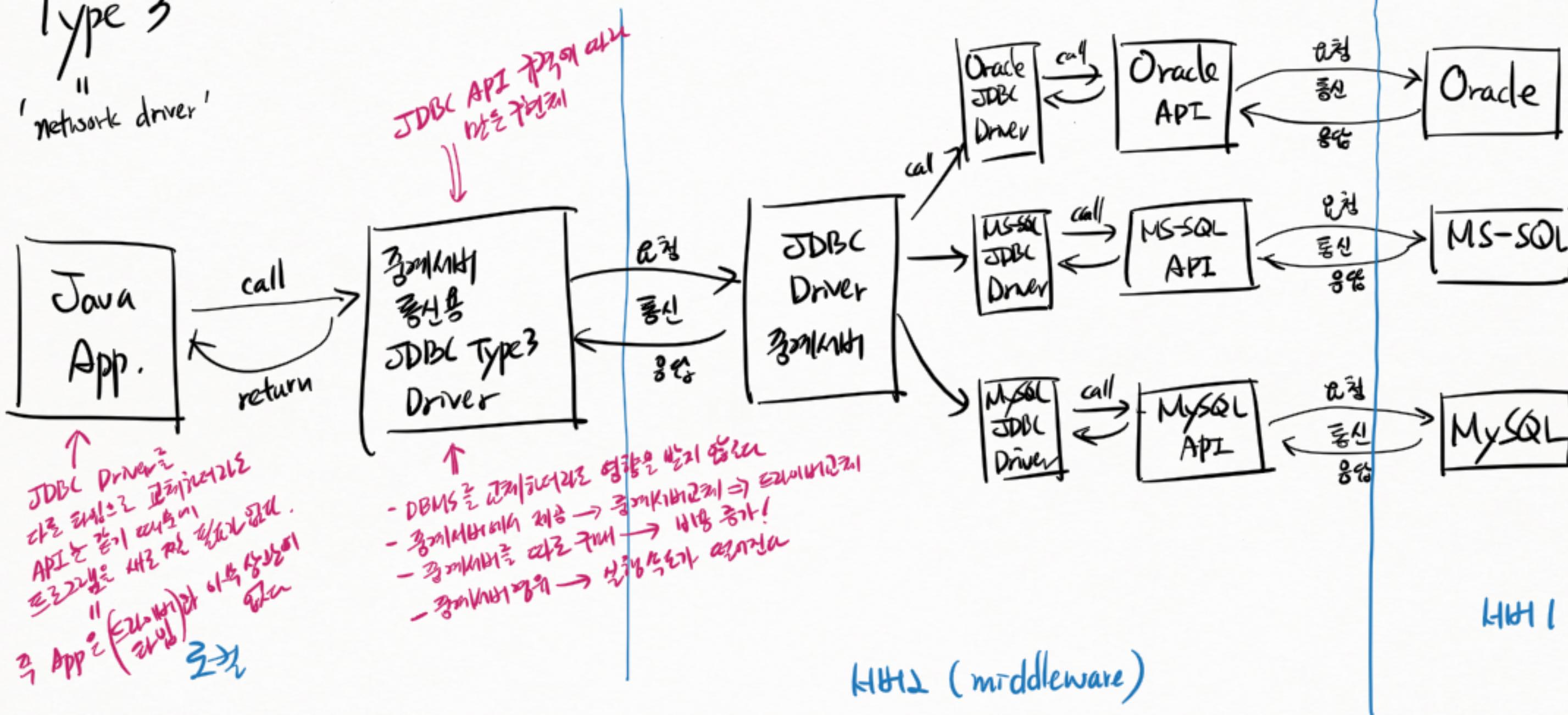


- Vendor API 사용 ⇒ 다른 드라이버 사용
- Native C/C++ API 사용 ⇒ local Vendor API 사용

DBMS를 연장하는,  
 ↳ local의 Vendor API를 통해  
 Type 2 JDBC Driver를 고친 }  
 Btw!

## Type 3

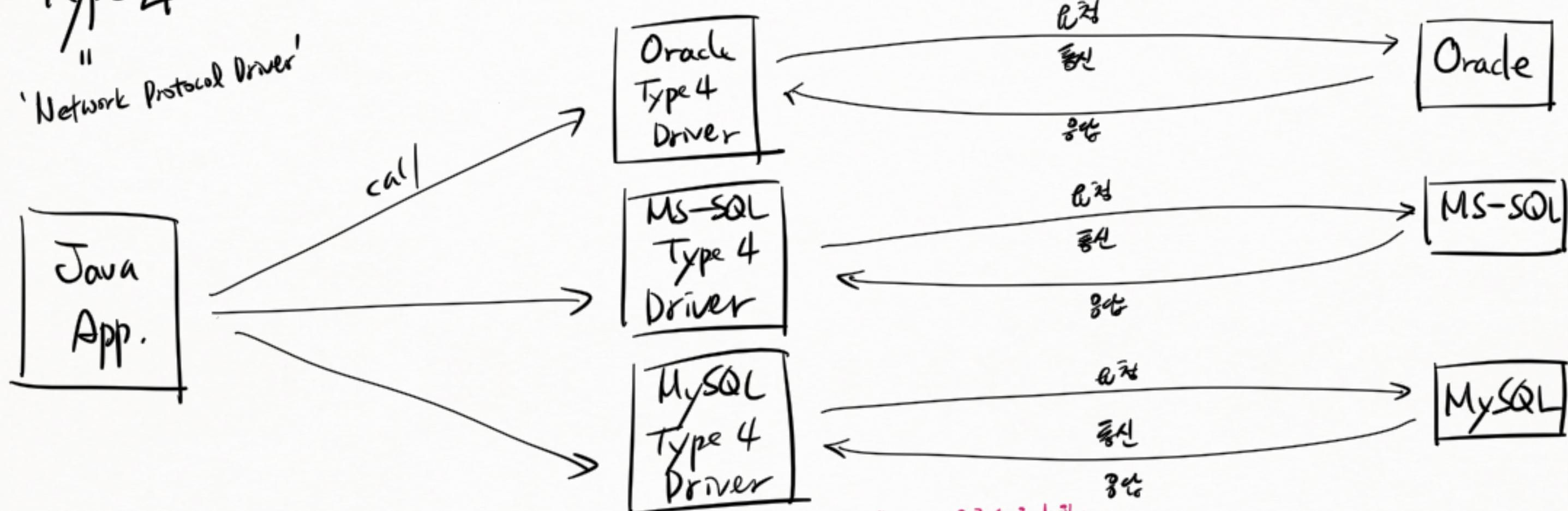
"network driver"



\* DBMS API : JDBC API  
 Java 인터페이스  
 ( java.sql.\* , javax.sql.\* ) → JDBC Driver  
 ↗ 버전  
 ↘ 클래스

Type 4

"Network Protocol Driver"



- DBMS Vendor의 API를 통해 접근하는 방식
- DBMS와 직접 통신 → local의 추가로 별도로 개발하는 방식

↓  
 C/C++과 같은 흐름 OS에 종속되는 API를 사용하는 방식

"Pure Java"

Native

## \* JDBC Driver 사용법

insert  
update  
delete  
select

