

---

---

---

---

---

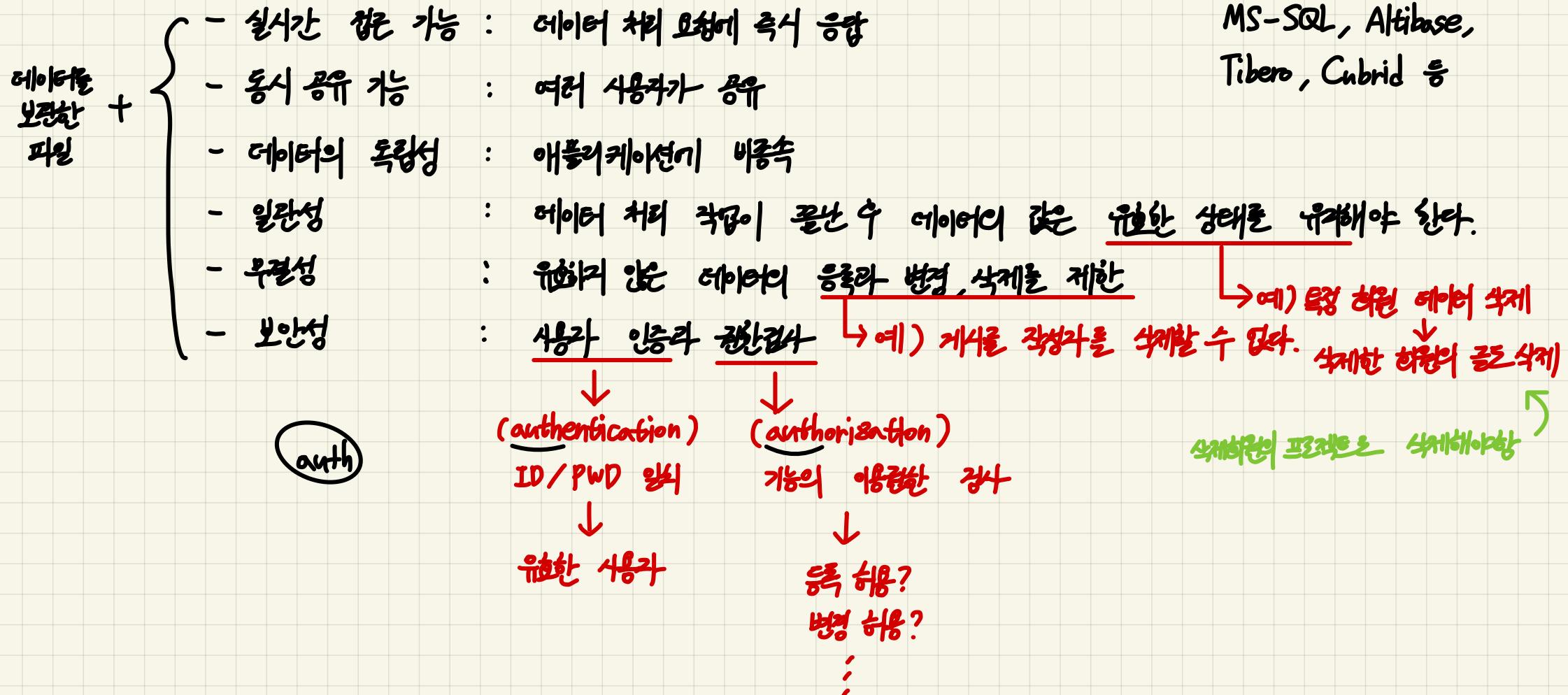


2021. 10. 05 (67주4) 11:00

## \* DBMS

Database Management System  
데이터 보관과 조작을 서비스하는 프로그램

### \* Database



2021.10.12 (

) 9:52

\* SQL 탄생

문법 A → 명령어 작성

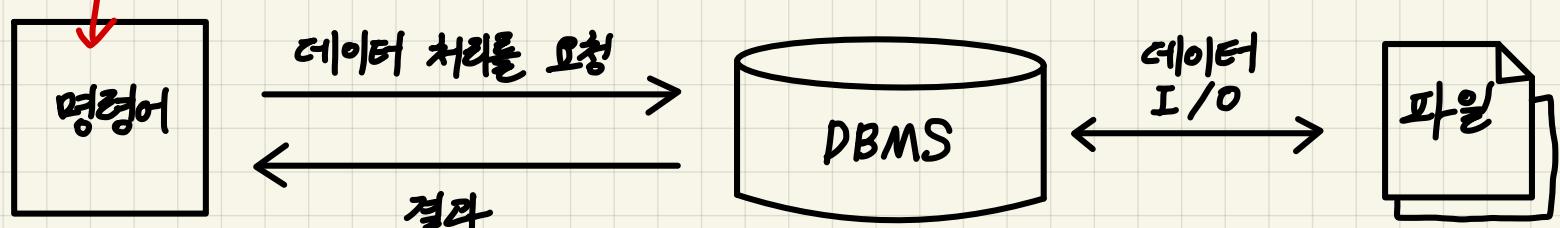


2021. 10 12 ( 일 ) 9 : 30

## \* DBMS와 SQL

명령어 작성 방법 = SQL

↳ 모든 DBMS에서 인식할 수 있는 표준문법



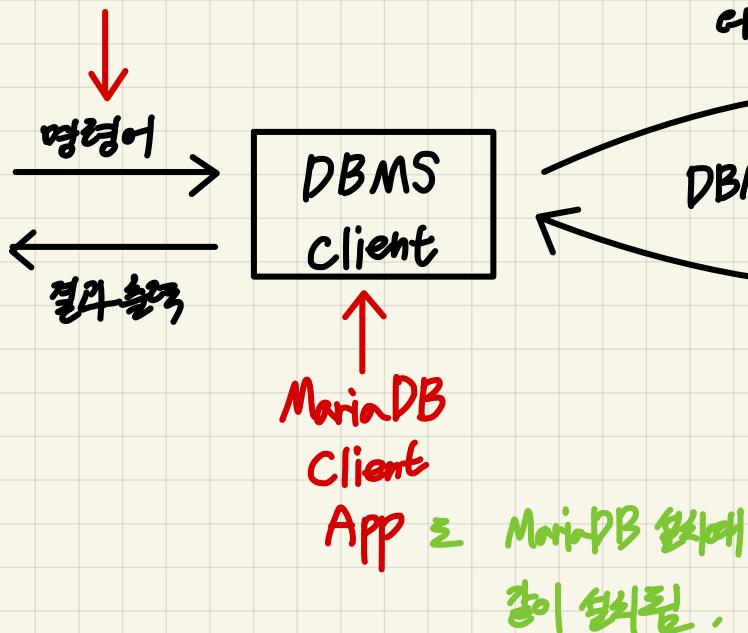
SQL 명령을 보내고 결과를 받으려면  
DBMS에서 정한 규칙에 따라 통신을 해야 한다!

↳ 문제는 DBMS 제조사 측에서 통신 규칙을 공개하지 않는다.

2021. 10. 06 (68주차) 9:30

## DBMS가 어떤 일방적인 것

SQL 언어로 명령어 작성



DBMS

MariaDB 설치 (10/5)



Data  
I/O



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

2021. 10. 05 (67주4) 13:30

\* DBMS 와 SQL  
→ MySQL

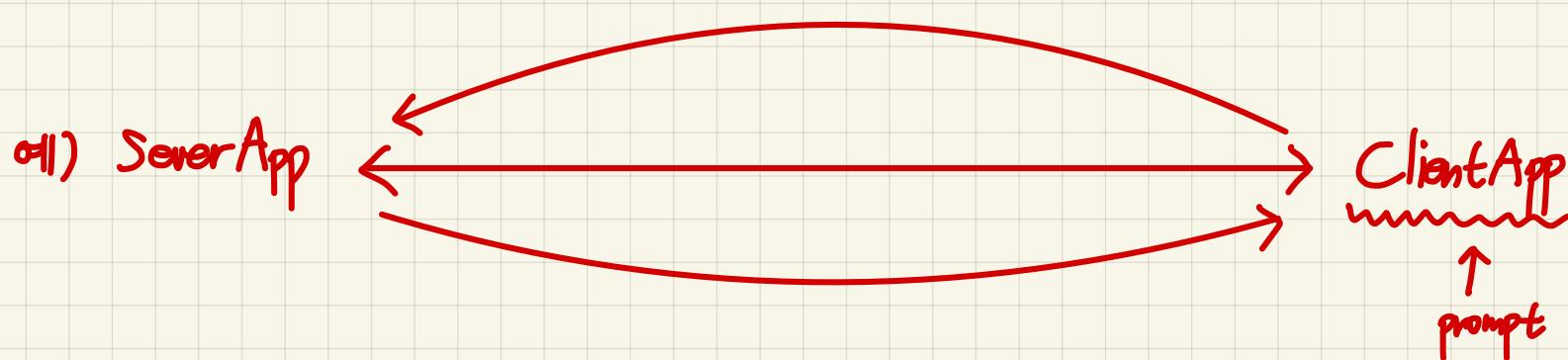
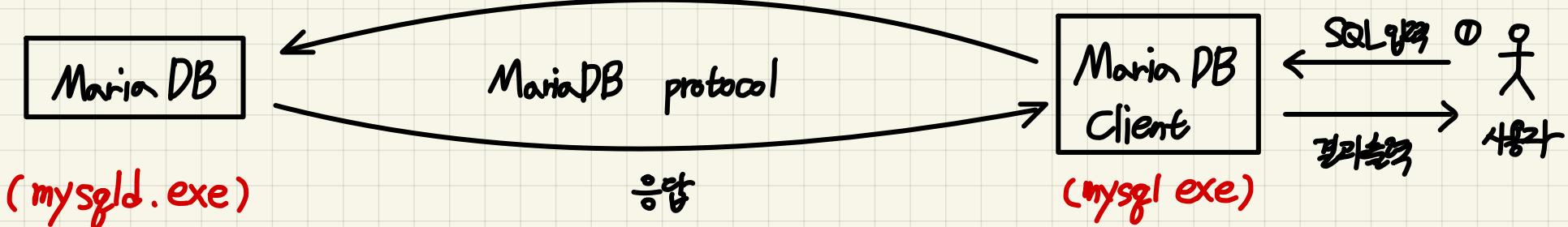
"create user ..."

"(서버에 연결)"

client에서 인트로가 아님

client가 서버에게 받은 응답에만 응답

② 모양



LTS

8 → upgrade X

11 → 0

X

17 → 0 granular 21.3

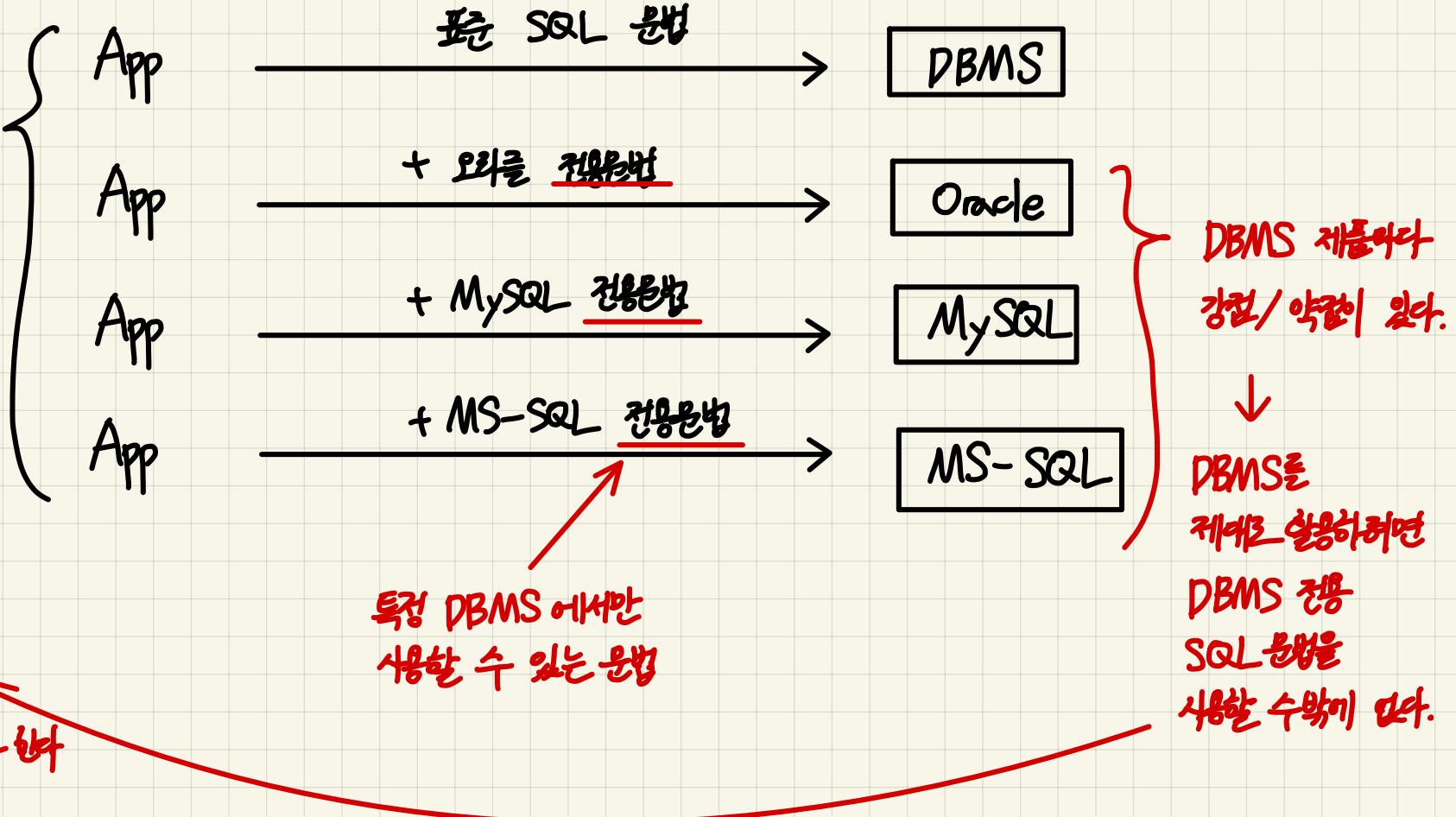
\* SQL

애플리케이션에서  
사용하는 SQL은  
DBMS 따라  
약간씩 다르다



그래서 실무에서  
DBMS에 따라  
코드를 변경해야 한다.

다음의 대기를 위하여



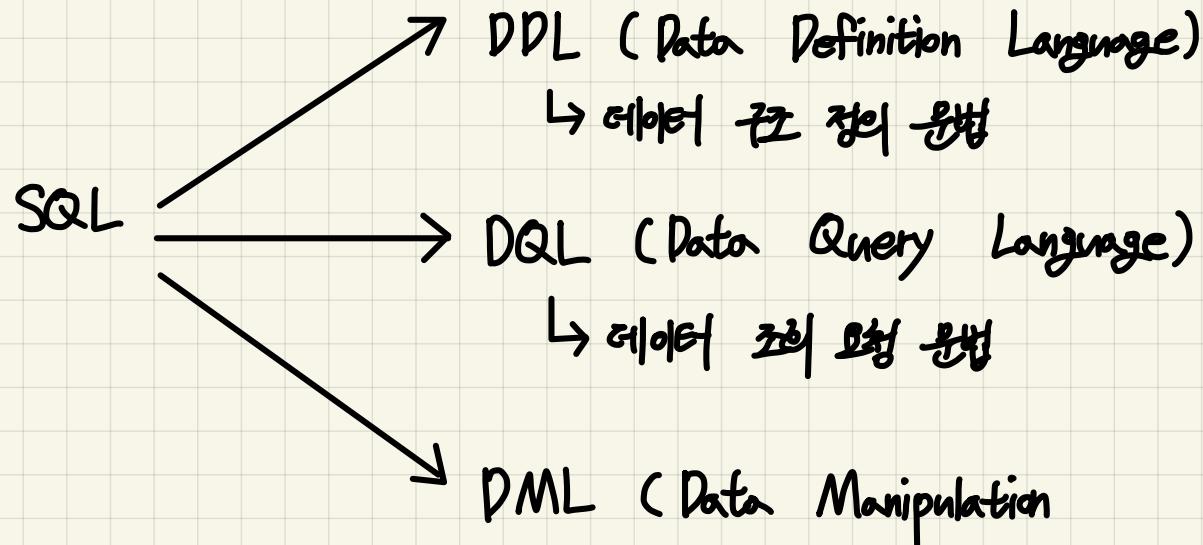
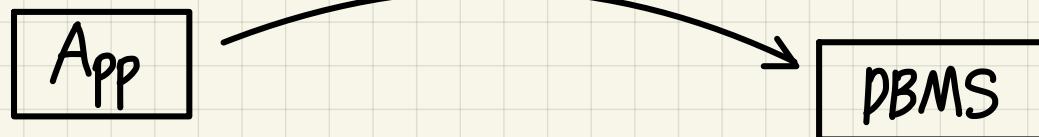
2021. 10. 05 (67주4)

11:30

## \* SQL (Structured Query Language)

DBMS에 상관없이  
요청 명령을 작성하는 언어 = SQL

데이터 처리 요청



MariaDB는 SQL의 오픈 소스임.

2021. 10. 05 (67주4) 13:30

## \* DBMS 와 SQL

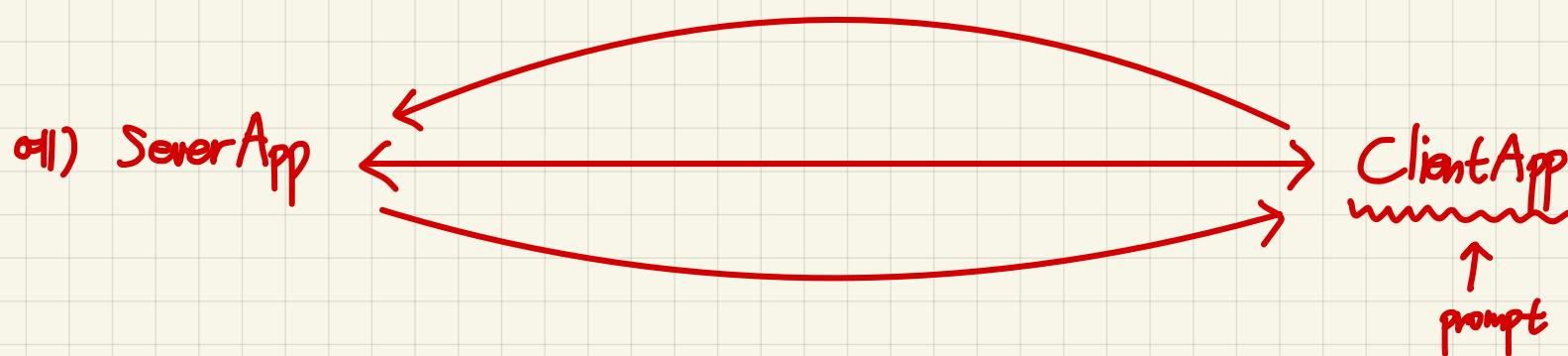
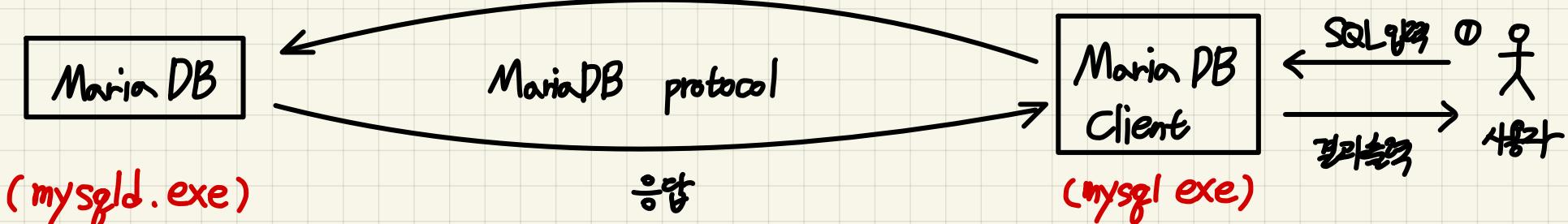
"create user ..."

"(서버에 연결)"

client에서 인트로가 아님

client가 서버에게 받은 응답에만 응답

② 모양



LTS

8 → upgrade X

11 → 0

X

17 → 0 granular 21.3

2021. 10. 05 (67주4) 13:56

## \* SQL 테스트 준비

root만 사용과 create 가능

mysql -u root -p ↵  
① 사용자 추가  
아이디 비번

↑ 차이 인식  
` backtic " singlequotation " double  
사용자 ID 접속을 허락할 PC 주소  
사용자 암호

> create user 'study'@'localhost' identified by '||||'; ↵  
↑ ↑  
사용자 ID 접속을 허락할 PC 주소  
사용자 암호

예) localhost에서 study 아이디로 접속하는 경우에 허락하겠다는 의미

> create user 'study'@'%' identified by '||||'; ↵

② 데이터를 저장할 데이터 베이스 생성

> create database studydb CHARACTER SET utf8 collate utf8\_general\_ci; ↵  
↑ ↑  
데이터베이스 이름  
: 'Id'tdb  
문자집합

③ 데이터 베이스를 사용할 사용자의 권한을 지정

> grant all on studydb.\* to 'study'@'localhost'; ↵  
부여하다  
앞의 모든 항목

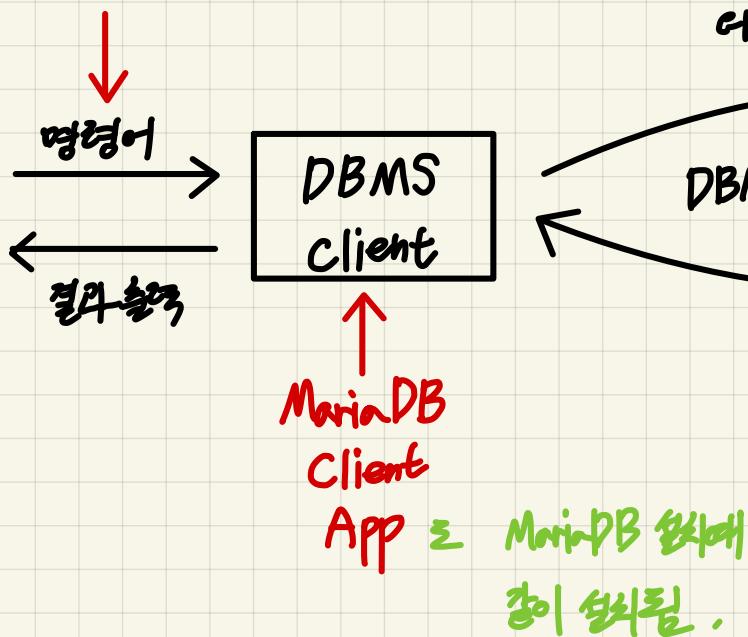
④ > use studydb ....> MariaDB [studydb]

2021.10.05 (6784) 15:13

\* DDL (Data Definition Language)

2021. 10. 06 (68일차) 9:30

SQL 언어로 명령어 작성



데이터 작업의 흐름

DBMS

MariaDB 설치 (10/5)

Data I/O



DBMS

요청처리 결과

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

① DDL → 데이터를 확장하고 이를 DB 구조를 정의한다.

② DQL { → DB 관리의  
데이터를 다룬다 } ⇒ { Table  
view  
Procedure  
Function  
:

2021. 10. 06 (68주차) 10:05

## \* 테이블

create table 테이블명 ( 칼럼, ... ) ← 테이블 정의

drop table 테이블명 ← 테이블 삭제

alter table 테이블명 변경사항 ← 테이블 변경

describle 테이블명  
||  
desc ← 테이블 정보조회

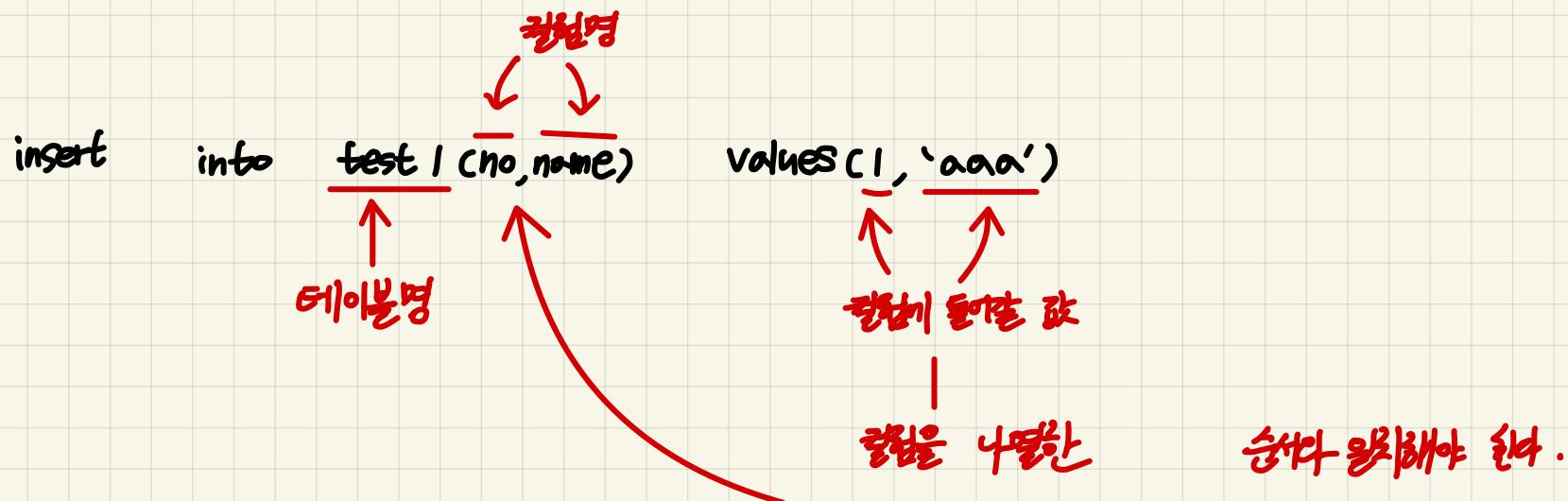
테이블을 ~~만들고~~ 만드려면 데이터 저장 가능

▷ 데이터를 저장할 틀

어떤 데이터 저장? 데이터 이름? 데이터가 들어갈 방법? 정의해야 → Data 저장하는  
방법

2021. 10. 06 (68일<sup>4</sup>) 10:30

\* insert



2021. 10. 06 (68%4) 10:38

\* select

select no, name from test1

\*

모든 행 선택

2021. 10. 06 (68일차) 11:40

\* key, candidate key, primary key / alternate key, artificial key

"

최소키

key

데이터를 구분할 때 사용할 컬럼들 \* (중복X 값들만 가능)

△ 1개 이상의 컬럼으로  
데이터 구분 가능한 것도 key이  
ex) [이름, 전화번호]

[이메일]

[아이디]

[주민번호]

[이름, 전화번호]

[아이디, 전화번호]

[이메일, 주민번호]

[이메일, 이름]

[주민번호, 이름]

[이름, 우편번호]

[이름, 아이디, 전화번호]

2021.10.06 (68일차) 11:48 \* key, candidate key, primary key / alternate key, artificial key

!!

최소키

데이터를 구분할 때 사용할 컬럼들 \* (중복X 값들만 가능)

↑ 1개 이상의 컬럼으로 데이터 구분 가능한 것도 key이  
ex) [이름, 전화번호]

key  
\* candidate key  
↳ 최소키

[이메일]

[이름, 전화번호]

[이메일, 이름]

[이름, 아이디, 전화번호]

[아이디]

[아이디, 전화번호]

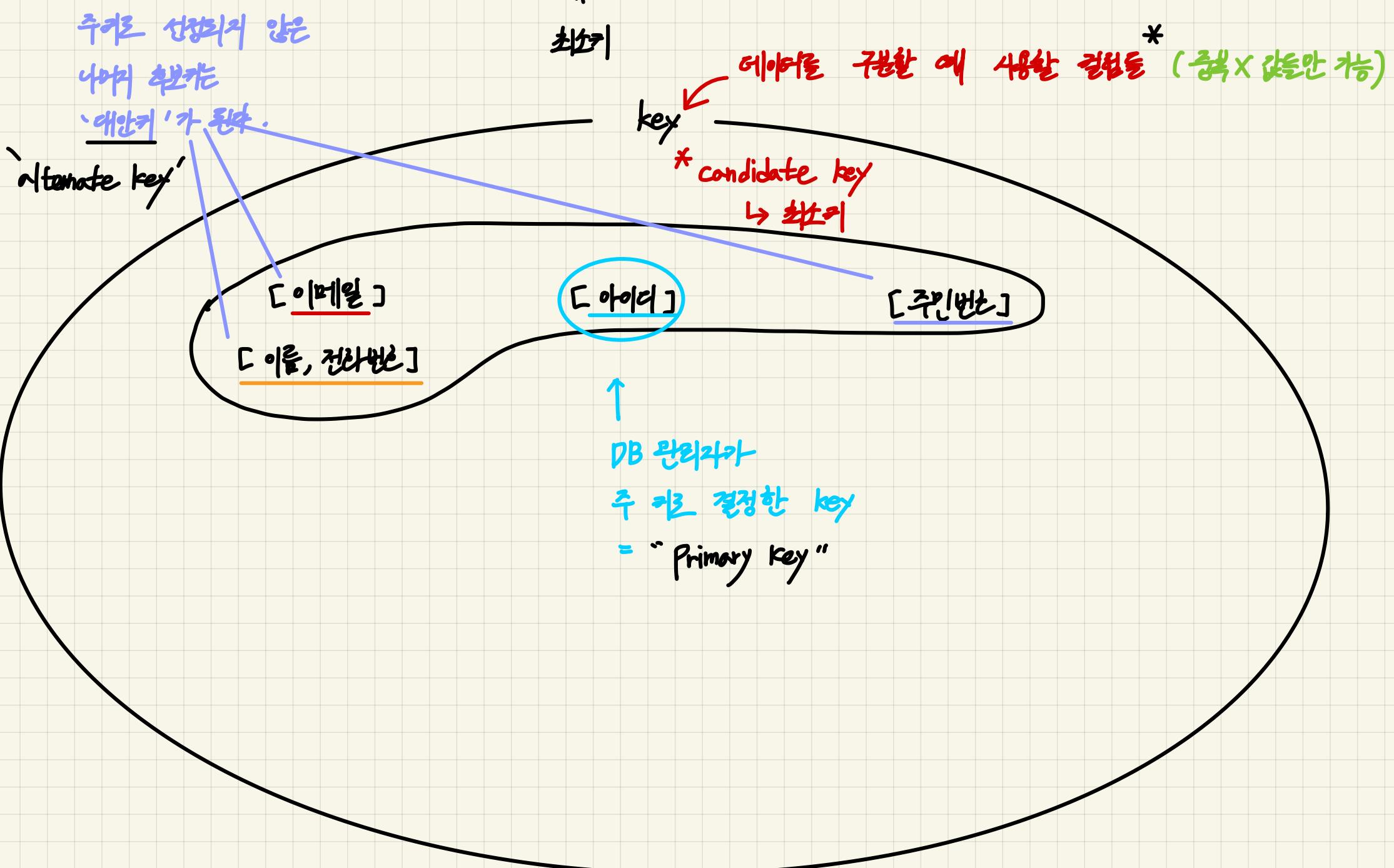
[주민번호, 이름]

[주민번호]

[이메일, 주민번호]

[주민번호, 이름]

2021. 10. 06 (68일차) 11:51 \* key, candidate key, primary key / alternate key, artificial key



2021. 10. 06 (68일차) 11:58

\*

제시글 : 제목, 내용, 작성일, 작성자, 조회수, 좋아요수

제시글 번호

Primary key?      굉장히 사용할 key가 없을 경우



Artificial key (인공키) =      입력의 결합을 만들어 PK를 결정한다.

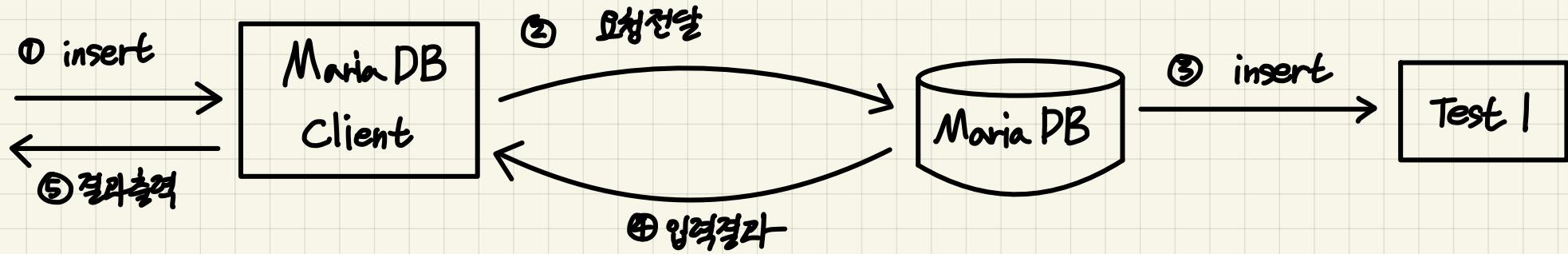
예) 일련번호

2021. 10. 06 (68주차)

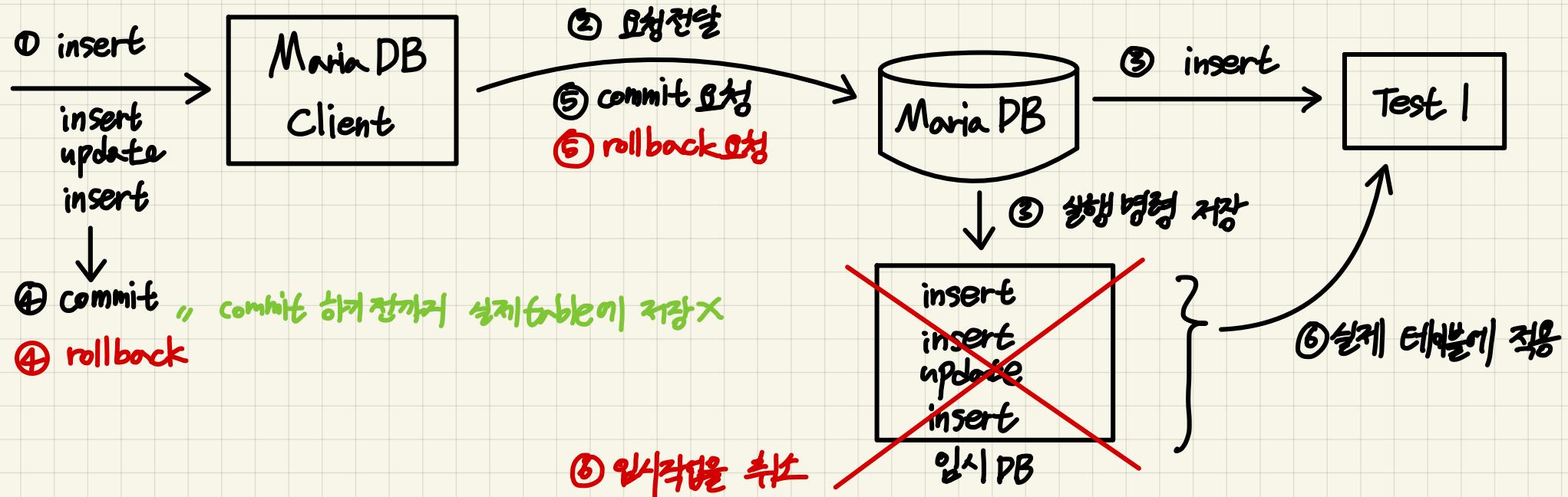
15:36

\* commit

## ① autocommit = true

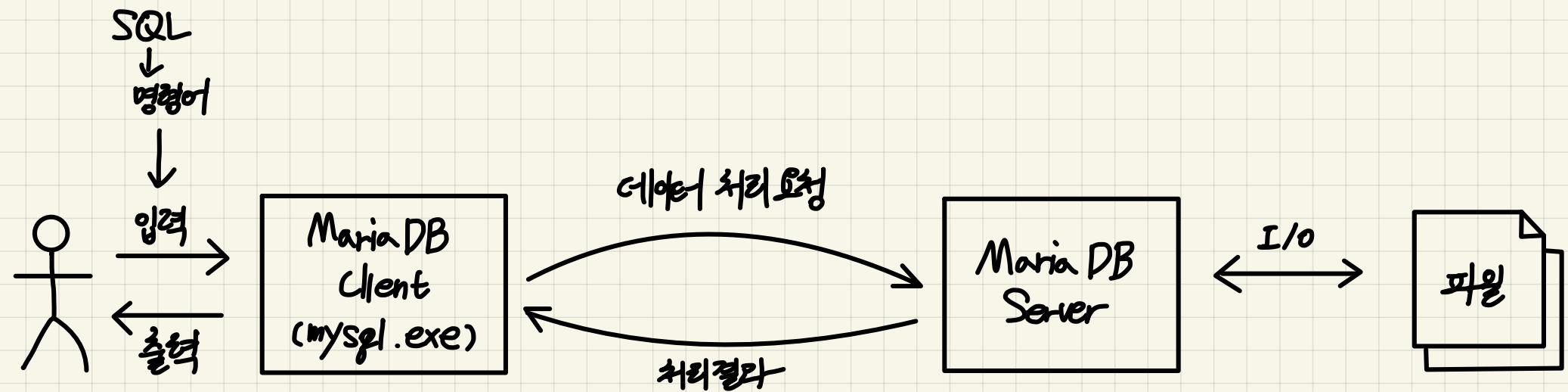


## ② autocommit = false



2021. 10. 07 (69일차) 9:30

\* select



DDL ( ① 테이블 정의 / 변경 / 삭제 )

DML ( ② 입력 / 변경 / 삭제 )  
DQL ( ③ 조회 / 검색 )

2021.10.07 (69일째) 10:30

\* selection / projection

select no, name from test1 where working='Y'

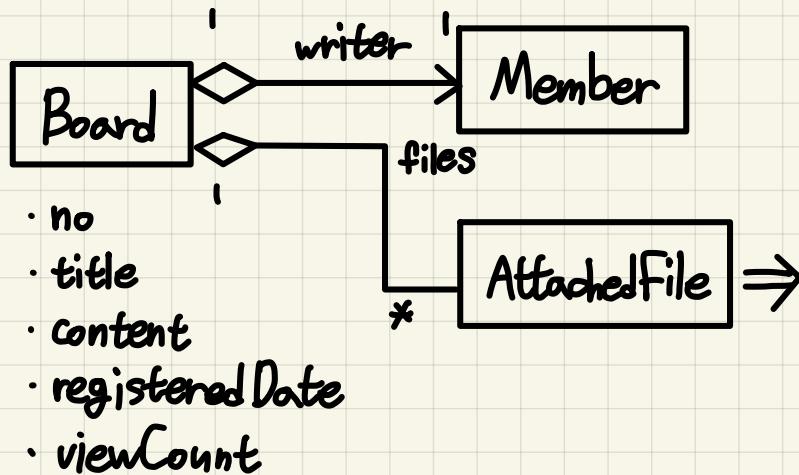
selection

no	name	class	working	tel
1	aaa			
2	bbb			
3	ccc			
4	ddd			
5	eee			
6	fff			
7	ggg			

2021. 10. 07 (69일차) 11:30

## \* Foreign Key

### ① 자바 객체

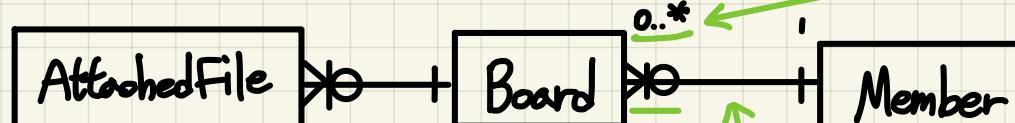


```

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

자바 객체  
기획개발 개발자만원

② 테이블 간 관계 (ER-Diagram; ERD) o 또는 그 이상



1번 개시글 → 100번 흥글동 Board table 이 Member table을 포함한다는 의미  
 2번 개시글 → 101번 일격정  
 3번 개시글 → 102번 유관순

### ERD 표기법

#### ① Information

#### Engineering Notation

용학

↓

정답으로 얻은 사실을 학문으로 체계화 시켜서 연구하는 것

↓

해당 분야에 종사하는 사람들이 시행착오를 줄이게 도와준다.

\* 개체간의 관계다  
 테이블간의 관계는 아都不是!  
 ↓  
 데이터 중복은 최소화하는 품질로 관계를 설정.

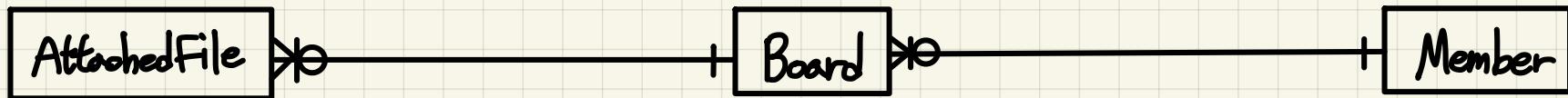
테이블

데이터 관계 (1:1 이 아님) 자바 클래스 - 테이블XXXXXX!

테이블 - 자바 클래스

2021. 10. 07 (69일차) 11:54 \* Foreign Key (외부키) - 다른 테이블의 PK 값

## ② 테이블 간 관계



AttachedFile			Board			Member		
(pk) 번호	파일명	게시글(FK)	(pk) 번호	제목	작성자(FK)	(pk) 번호	이름	
11	aaa.gif	101	101	aaaa	1	1	홍길동	
12	bbb.gif	102	102	bbbb	1	2	임꺽정	
13	ccc.gif	102	103	cccc	3	3	유관순	
14	ddd.gif	103	104	dddd	4	4	안중근	
15	eee.gif	103						

PK로 가리킴  
(참조)

PK로 가리킴  
(참조)

자식 테이블 ← → 부모 테이블  
참조

자식 테이블 ← → 부모 테이블

2021. 10. 07 (69일차) 13:25

## \* 컬럼 충복

이렇게 같은 종류의 데이터를 여러개 저장하기 위해  
컬럼이 충복으로 선언된 경우

Board

no	title	content	rdt	f1	f2	f3	f4	f5
1	aaa	_____	—	a.gif	b.gif			
2	bbb	_____	—	x.gif	y.gif	z.gif	k.gif	
3	ccc	_____	—					
4	ddd	_____	—	m1.gif	m2.gif	m3.gif	m4.gif	m5.gif

첨부파일이 없음에도

5개의 컬럼이 존재하기 때문에  
메모리 낭비

이전 컬럼 충복이나  
데이터 충복을 예외로

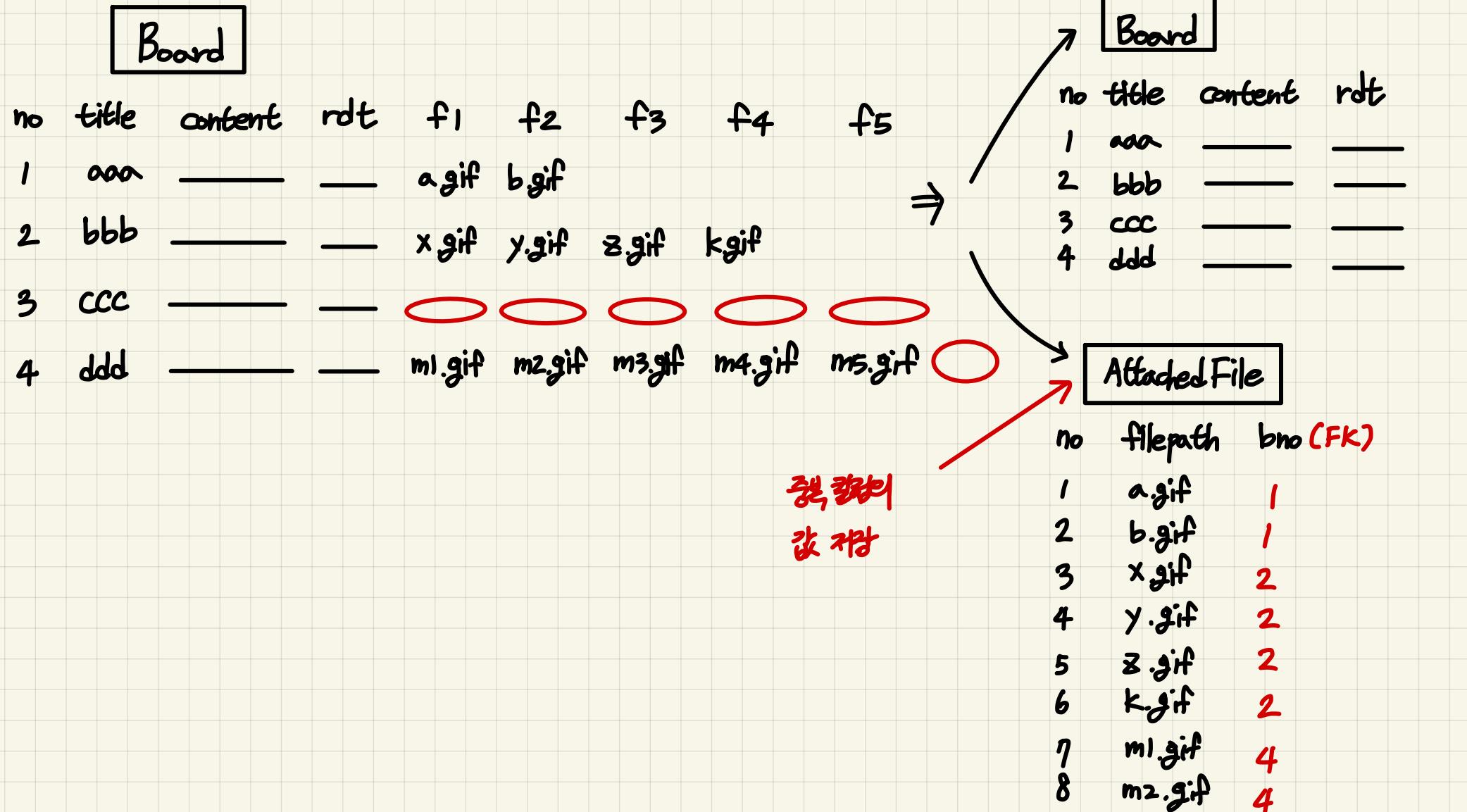
컬럼이 부족해서  
첨부파일을 더 저장할 수 없다.

충분히 충분!

다른 문제 발생

2021. 10. 07 (69일차) 13:33

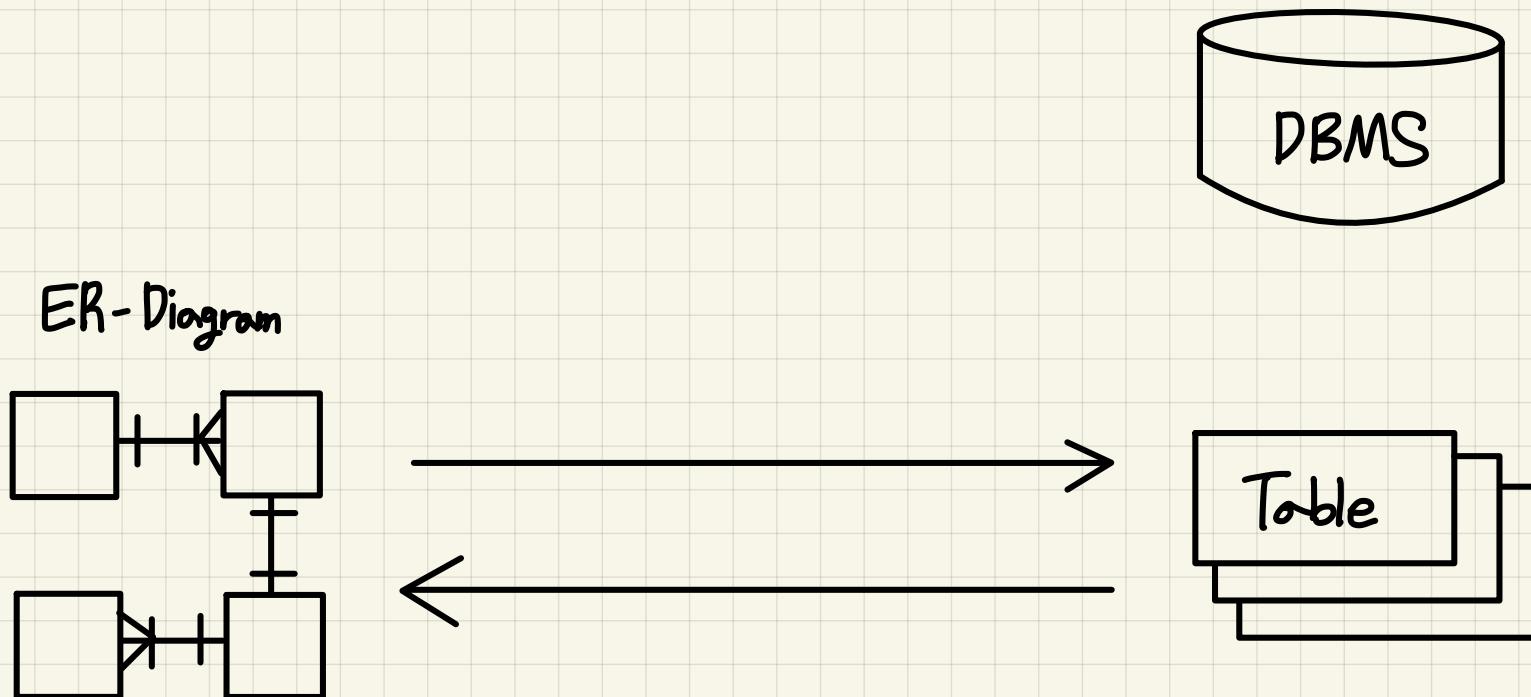
## \* 컬럼 충복



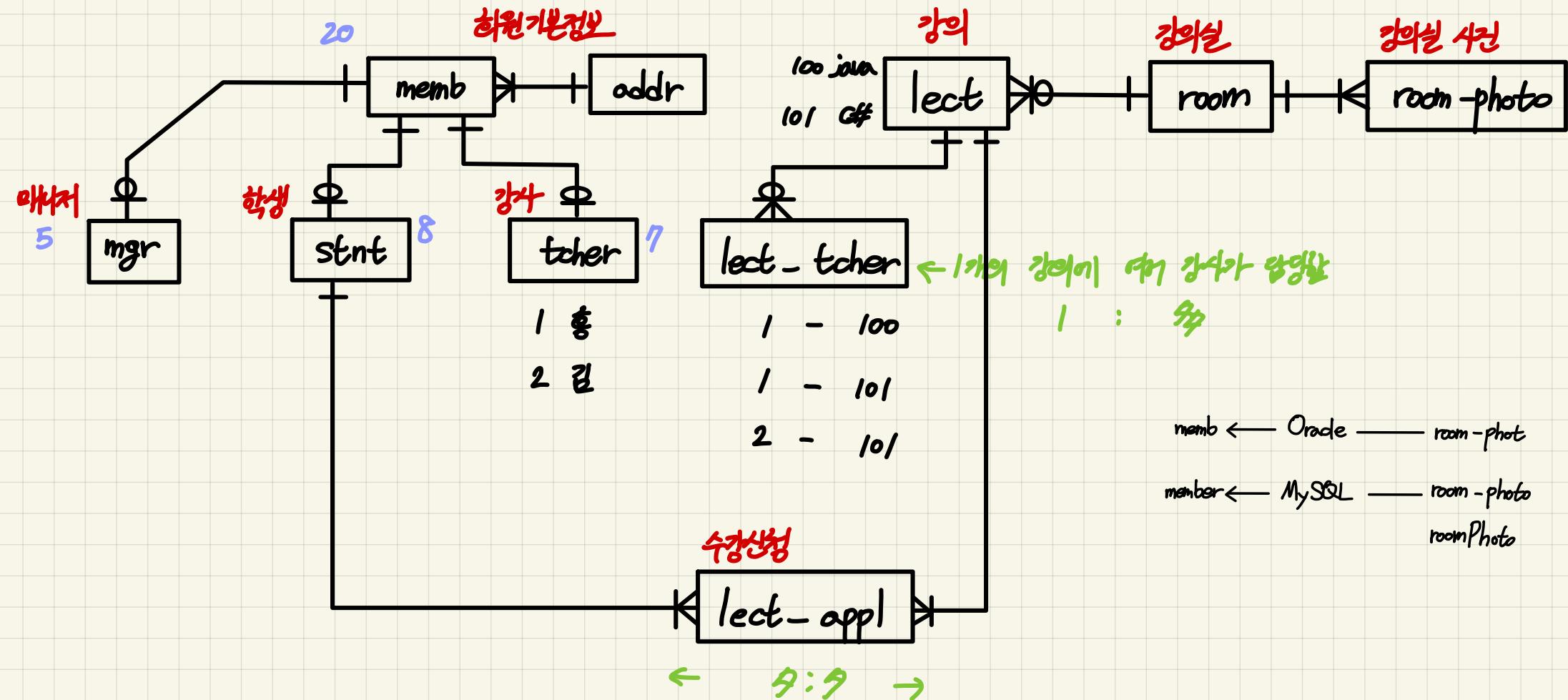
2021. 10. 07 (69주차) 14:38

\* ERD : forward / reverse engineering

제작도구 : ER-Win, exend, ...



2021. 10. 07 (69일차) 14:50 \* SQL 조인 테스트를 위한 데이터의 ERD  
예) 교육센터 관리 시스템



2021. 10. 08 (일요일) 9:40

\* 조인

← 무조건 1:1 결합

## ① cross 조인 (카티션 조인)

A	B
aaa	111
bbb	222
ccc	

↓ 두 예제의 데이터를  
하나로 합침

aaa	111
aaa	222
bbb	111
bbb	222
ccc	111
ccc	222

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

↓

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

Board		AttachedFile	
no	title	no	path
1	aaa	11	a.gif
2	bbb	12	b.gif
3	ccc	13	c.gif
4	ddd	14	d.gif

제거로  
대체는↓ 첨부파일  
제거는 → 서로 복원

{ ✓ 조인의 기준이 될 컬럼이 일치하지 않을 경우  
 ✓ 서로 상관되는 행의 이들이 같은 경우

FK 결합 컬럼을 ≠ FK가 가리키는  
 PK 컬럼이를  
 올바른 조인이 수행되지 못한다

2021. 10. 08 (일요일) 9:58 \* Natural 조인을 수행할 기준 컬럼의 이름이 일치하지 않거나 영향한 컬럼과 일치할 경우

③ join on 컬럼값 비교로

Natural 조인의 문제점 해결

테이블 테이블

Board join AttachedFile

on Board.no = AttachedFile.bno

조인조건

2021. 10. 08 (일) 10:02

\* Natural 조인을 수행할 때,  
기준 테이블에 이름이 같은 헤더가 있는 경우

Board

bno	title	name
1	aaa	kim
2	bbb	park

AttachedFile

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

테이블

Board join AttachedFile

using (bno)

테이블

조인할 때 사용할

기준 헤더를

명시적으로 지정한다.



2021. 10. 08 (70 일차) 10:10

\* 대 데이터를 가져올 때 여러 테이블의 데이터를  
결합해서 처리하는가?



데이터 중복을 피하기 위해 데이터가 여러 테이블에 분산되어 있기 때문이다. ⇒ join을 사용  
↳ (join을 사용하는 이유)

## Project

번호	제목	담당자를	팀장전화
1	aaa	kim	010-1111-1112
2	bbb	kim	010-1111-1112
3	ccc	kim	010-1111-1111

↑  
데이터 중복

||

- 범경이 번거롭다.
- 범경 항목을 누락할 수 있다.

||

결합방식 = 데이터 } 같은 팀장인데  
일원성이 } 전화번호가  
깨진다 } 다르다?

⇒ 데이터 중복 문제  
해결

↓  
데이터를 분산

## Project

번호	제목	팀장번호
1	aaa	1
2	bbb	1

## User

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

2021. 10. 08 (일요일) 10:49

\* 조인을 수행할 때 테이터가 누락된 상황

**Lecture**

<b>No</b>	<b>name</b>	<b>tno (fk)</b>	<b>(pk) tno</b>	<b>name</b>
11	aaa	1	1	kim
12	bbb	2	2	park
13	ccc	1	3	lee
14	ddd		4	eom
15	eee			

교수 테이블은 학생으로 부터

**Teacher****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 테이터의 어려움이 있으면 결과에서 누락되는 문제가 발생

↓ outer join이 해제

**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

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
null	null	null	4	eom

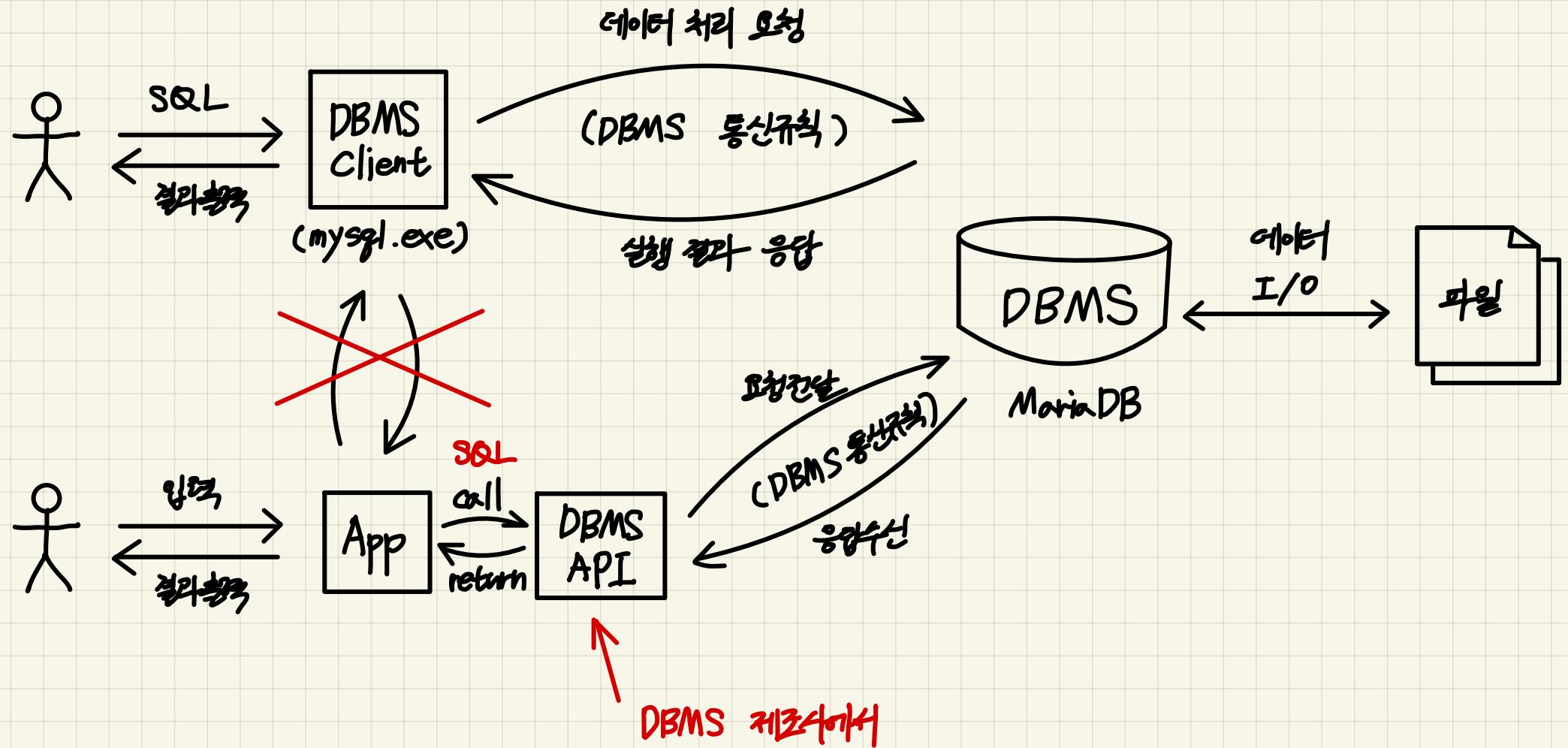
2021. 10. 08 (일) 11:47

\* 조인 테스트

Lect

Room

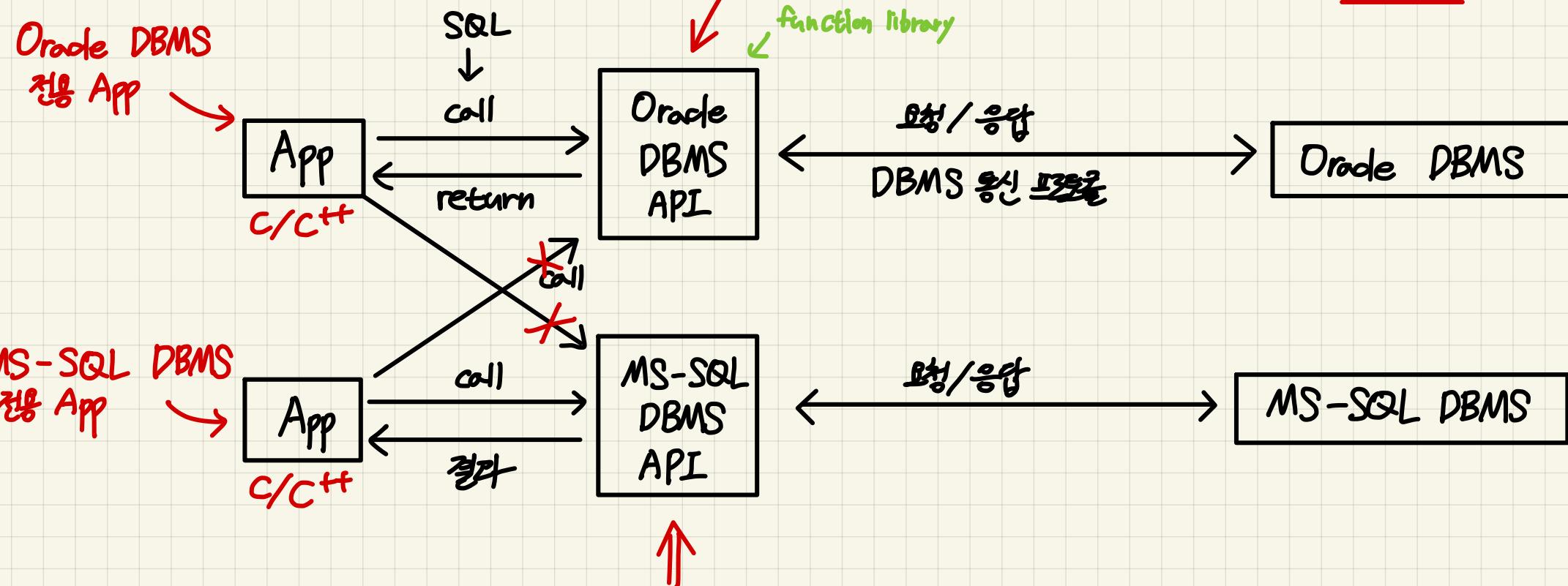
lno titl rno rname



2021. 10. 12 ( 월 ) 10:14 \* DBMS API - Native API

## \* DBMS API - Native API (= Vendor API)

클래스



문제점 : DBMS API 따라 할수 / 클래스 정의가 다르다.

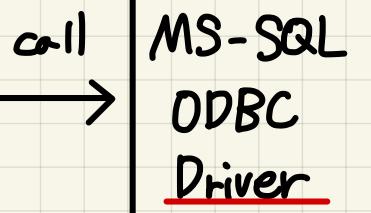
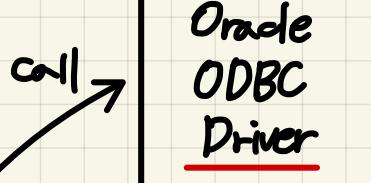
메서드 사용방법이 다르다.

2021. 10. 12 (일) 10:40 \* DBMS API - ODBC API 등장

DBMS 아파  
현대화의 갈기 때문에

ODBC API 규격  
규격에 따라 만든 API

통일된 DBMS API 규격 = 구현체가 아닌 규칙을 정한 그것  
(Open Database Connectivity)



C/C++ (.dll/.so  
.lib)

call

call

call

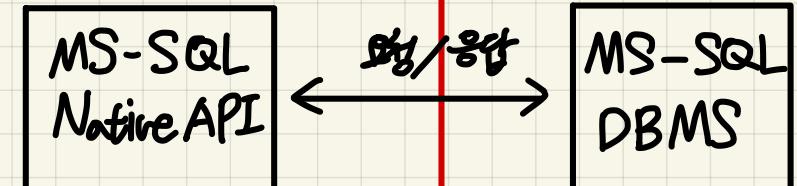
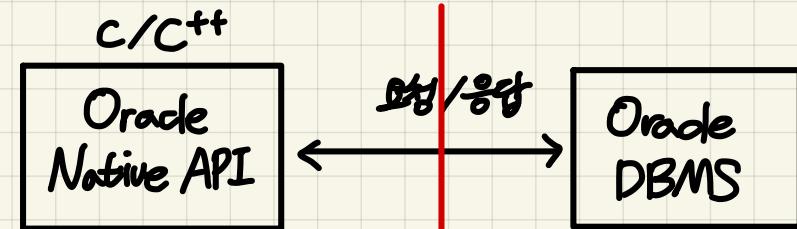
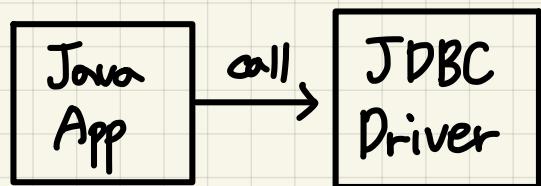
DBMS  
종종  
용량 APP

2021. 10. 12 ( 일자 ) 11:08

\* DBMS API - JDBC API 규격

↳ Java Database Connectivity API 규격

JDBC API 규격에 따라 구현한  
클래스 라이브러리



\* Type I 드라이버

- ODBC - JDBC Bridge
- JRE에 기본포함

Local

Remote

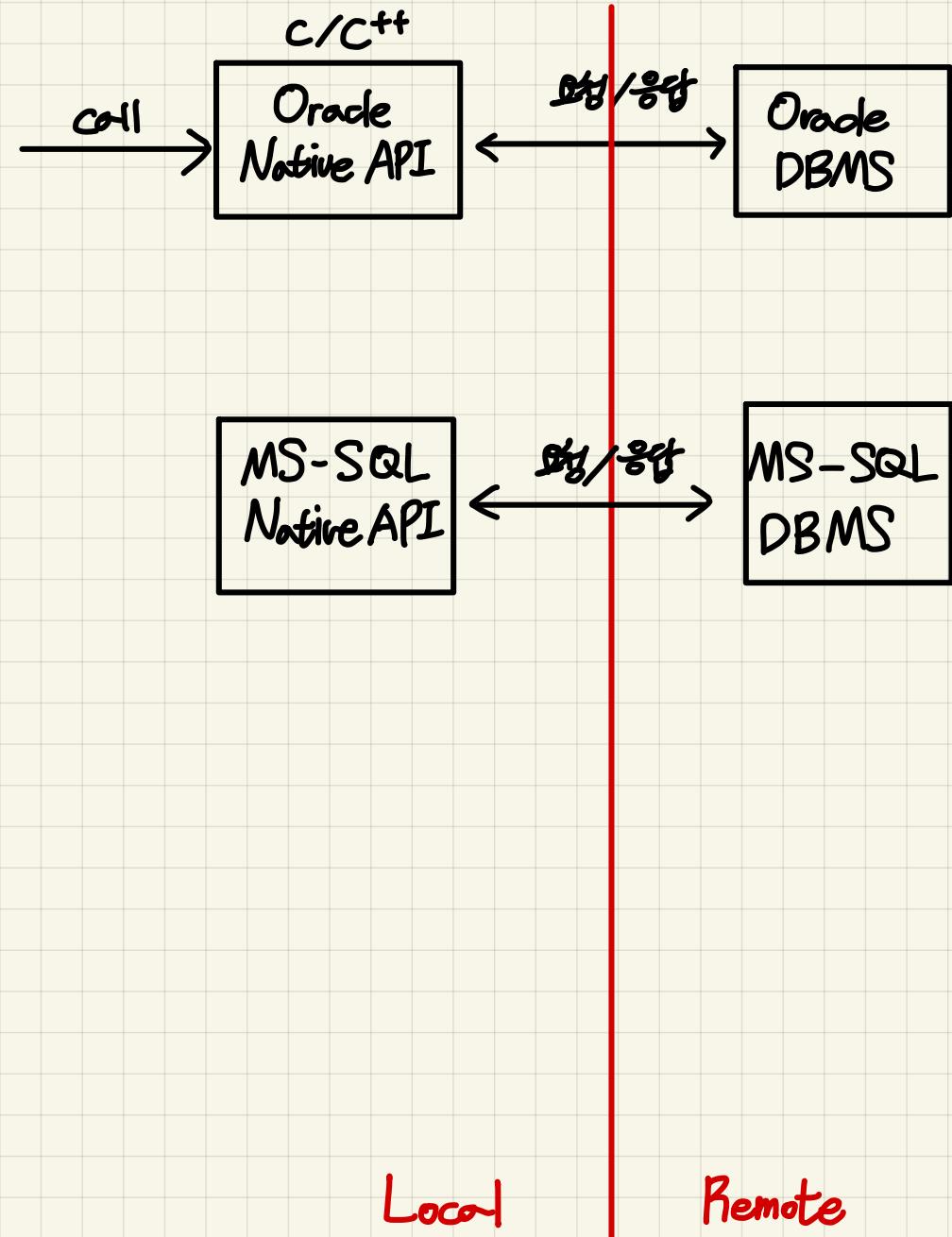
2021. 10. 12 ( 일자 ) 11:20

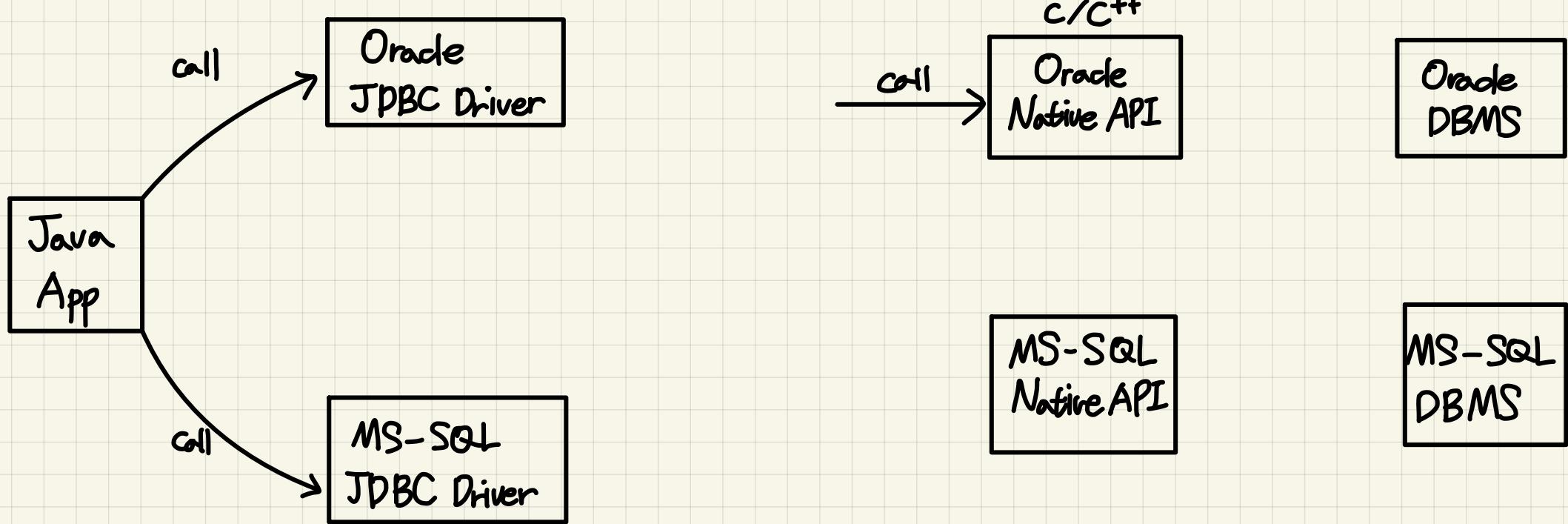
## \* DBMS API - JDBC API 규격

### \* Type 2 드라이버

- Native API 드라이버
- JRE에 포함 안됨  
↳ DBMS Vendor에서  
별도로 다운로드 해야 한다.
- 프로그램에 Native API  
설계해야 한다.

구현





#### \* Type 4 드라이버

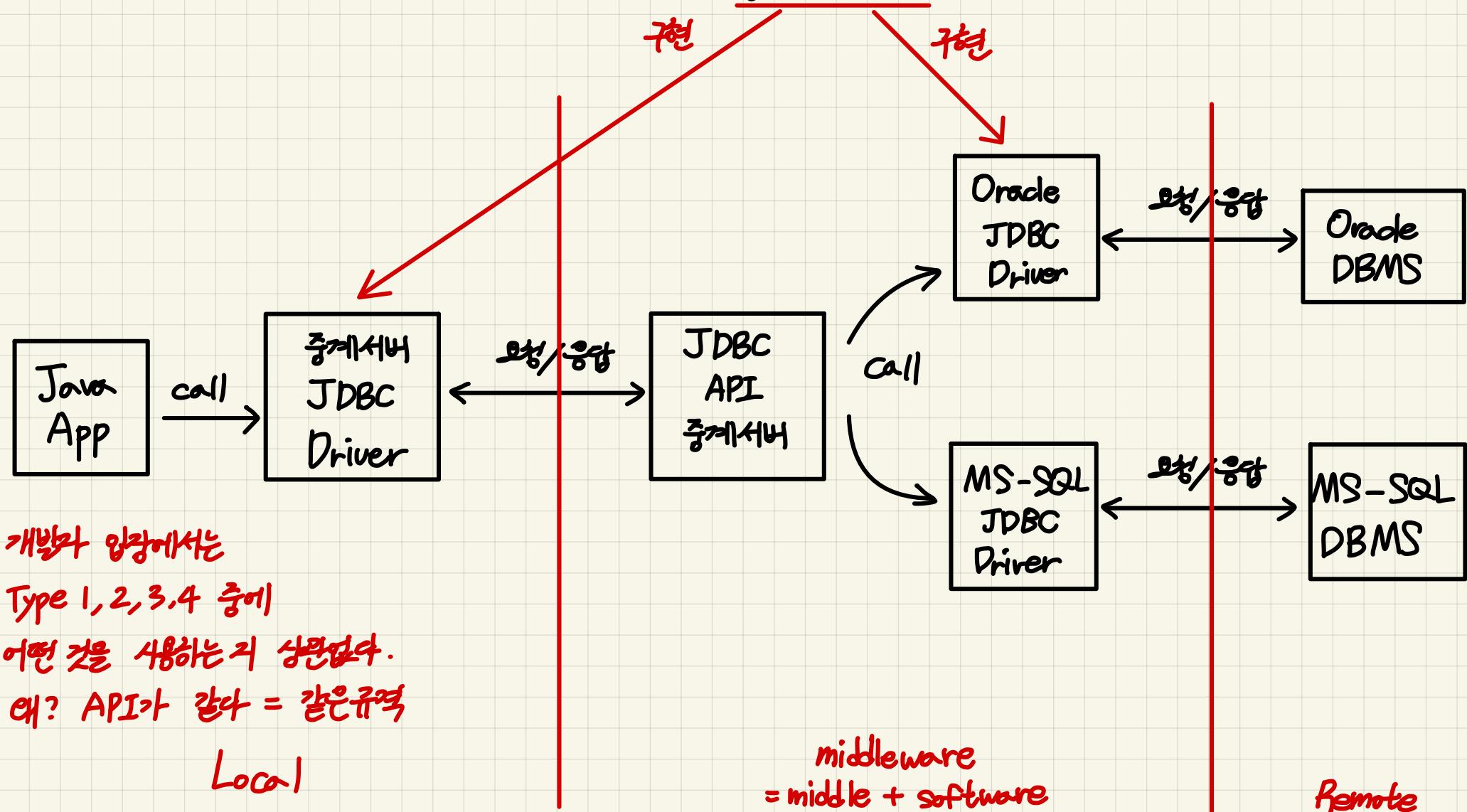
- Network protocol 드라이버
- DBMS Vendor에서 별도 배포로드 한다.
- C/C++ 헤더 → pure Java 드라이버
- 로컬에 C/C++ 라이브러리

Local

Remote

2021. 10. 12 ( 일자 ) 11:43

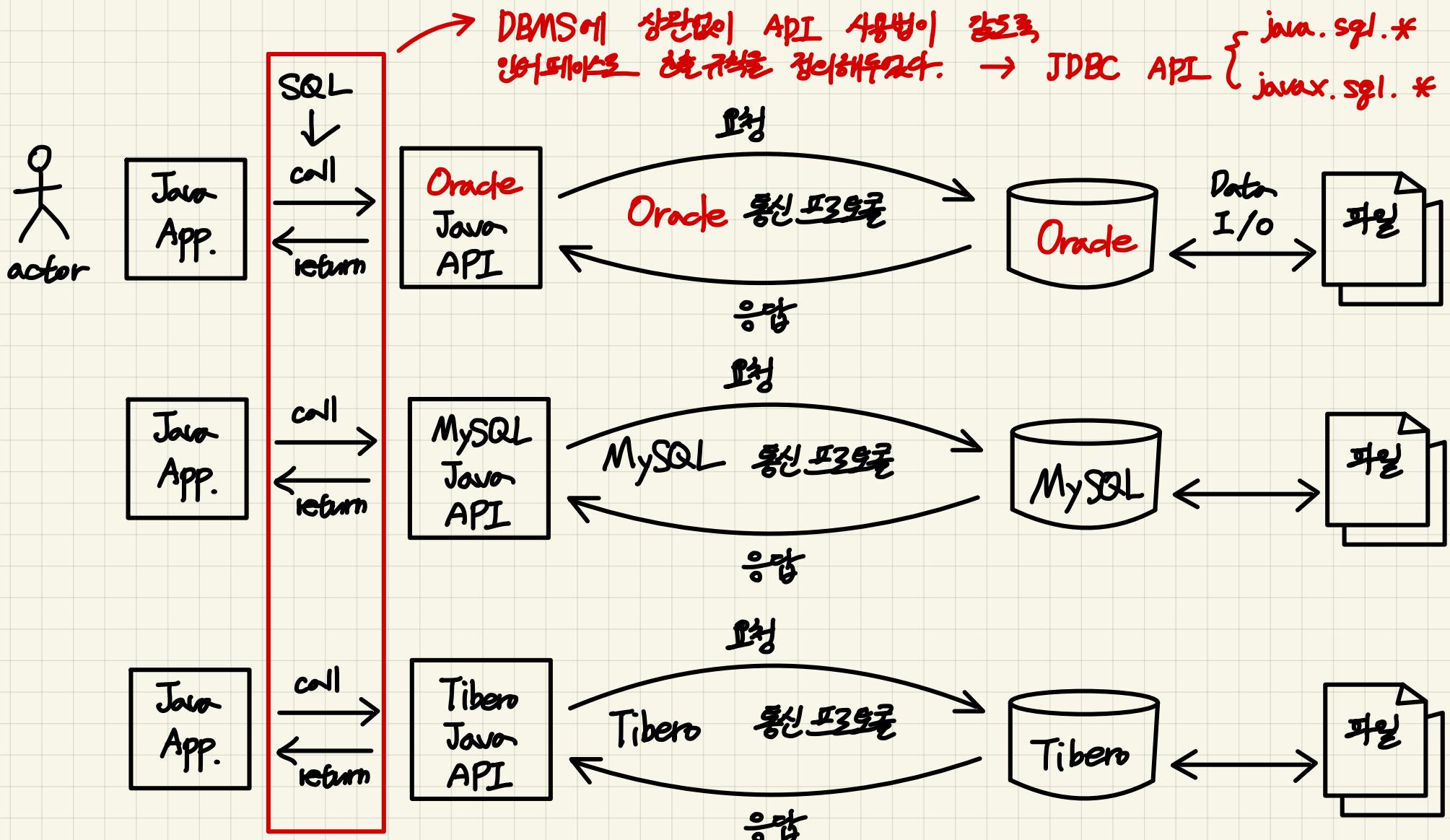
## \* DBMS API - JDBC API 규격



2021. 10. 12 ( 일 ) 11:00

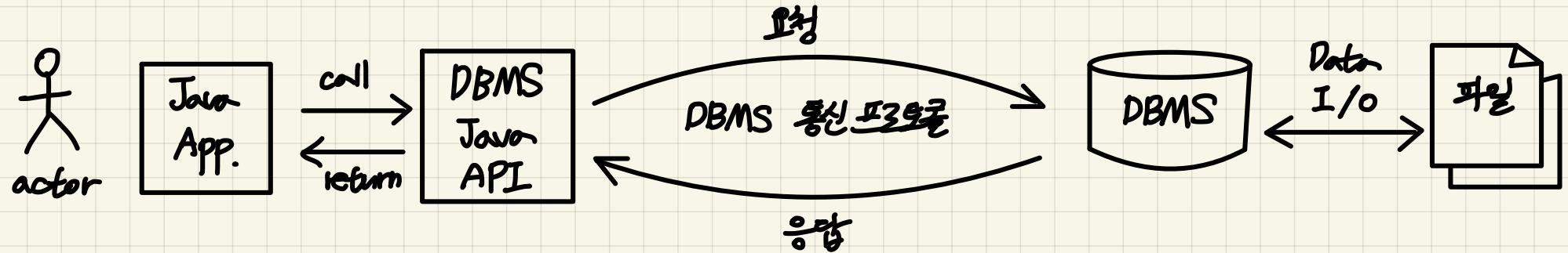
2021. 10. 08 (일) 14:25

## \* DBMS API 와 JDBC API

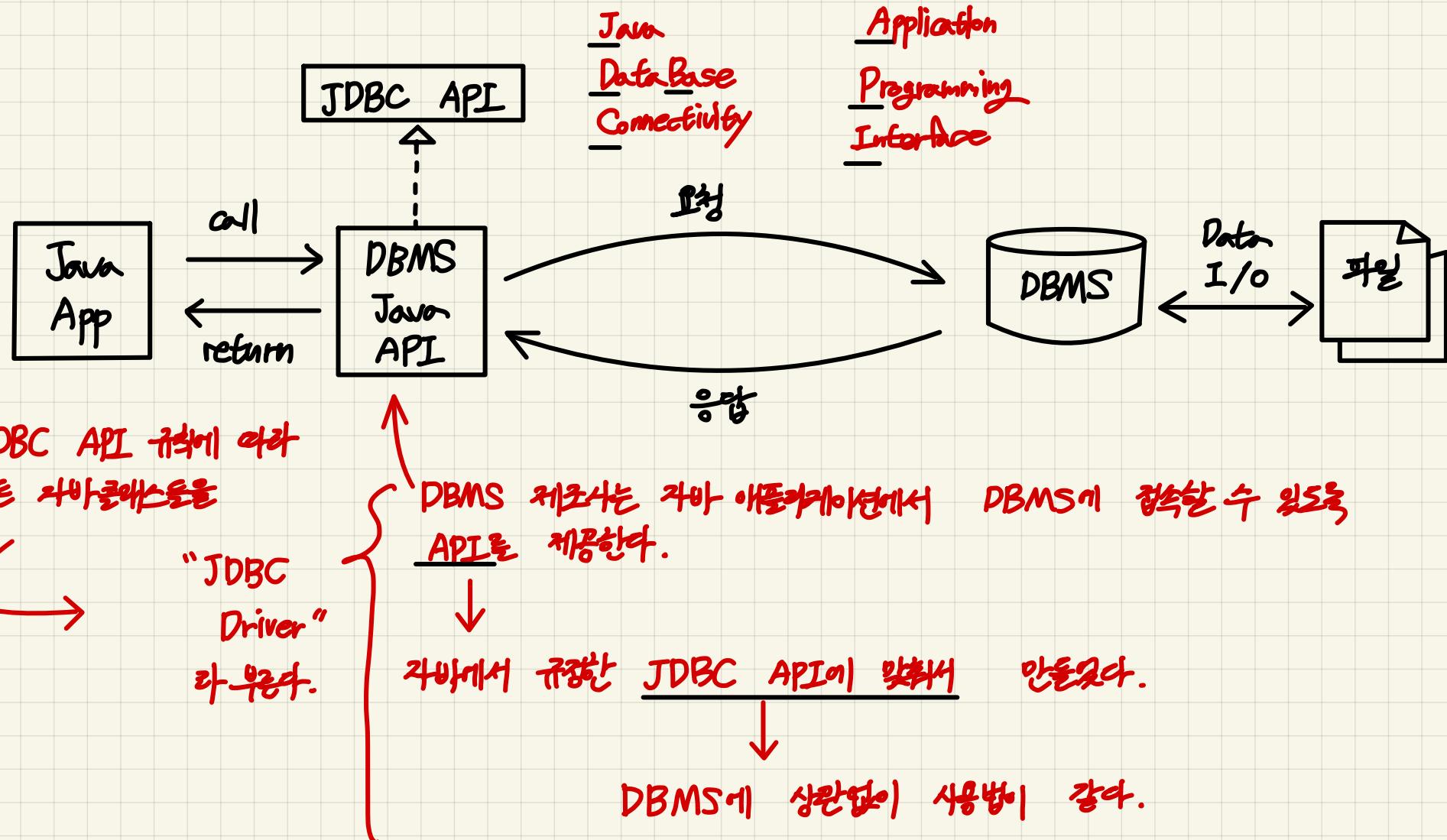


2021. 10. 08 (일) 14:30

## \* Java 와 DBMS API



• DBMS API



2021. 10. 08 (일) 14: 57

\* JDBC 프로그래밍

- ✓ API 구조에 따라 클래스를 정의
- ✓ 그 클래스를 이용하여 DBMS에 접속
- ✓ SQL을 전달하여 데이터 처리 요구

JDBC API의 사용법에 따라

JDBC Driver이 들어있는 클래스를 이용하여

DBMS 서버로 접속하여

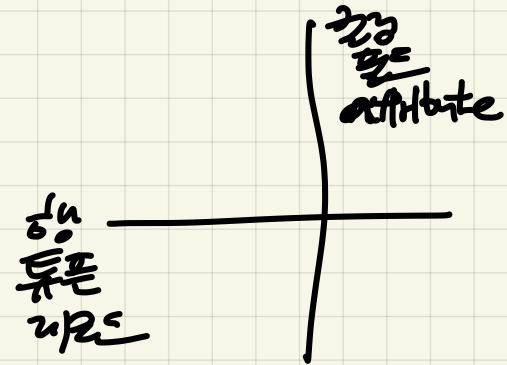
데이터를 처리하도록 코드를 작성하는 것!

2021. 10. 08 (일) 15:05

\* JDBC 프로그래밍 준비

① Maria DB JDBC 드라이버 준비  
(Type 4)

② JDBC API

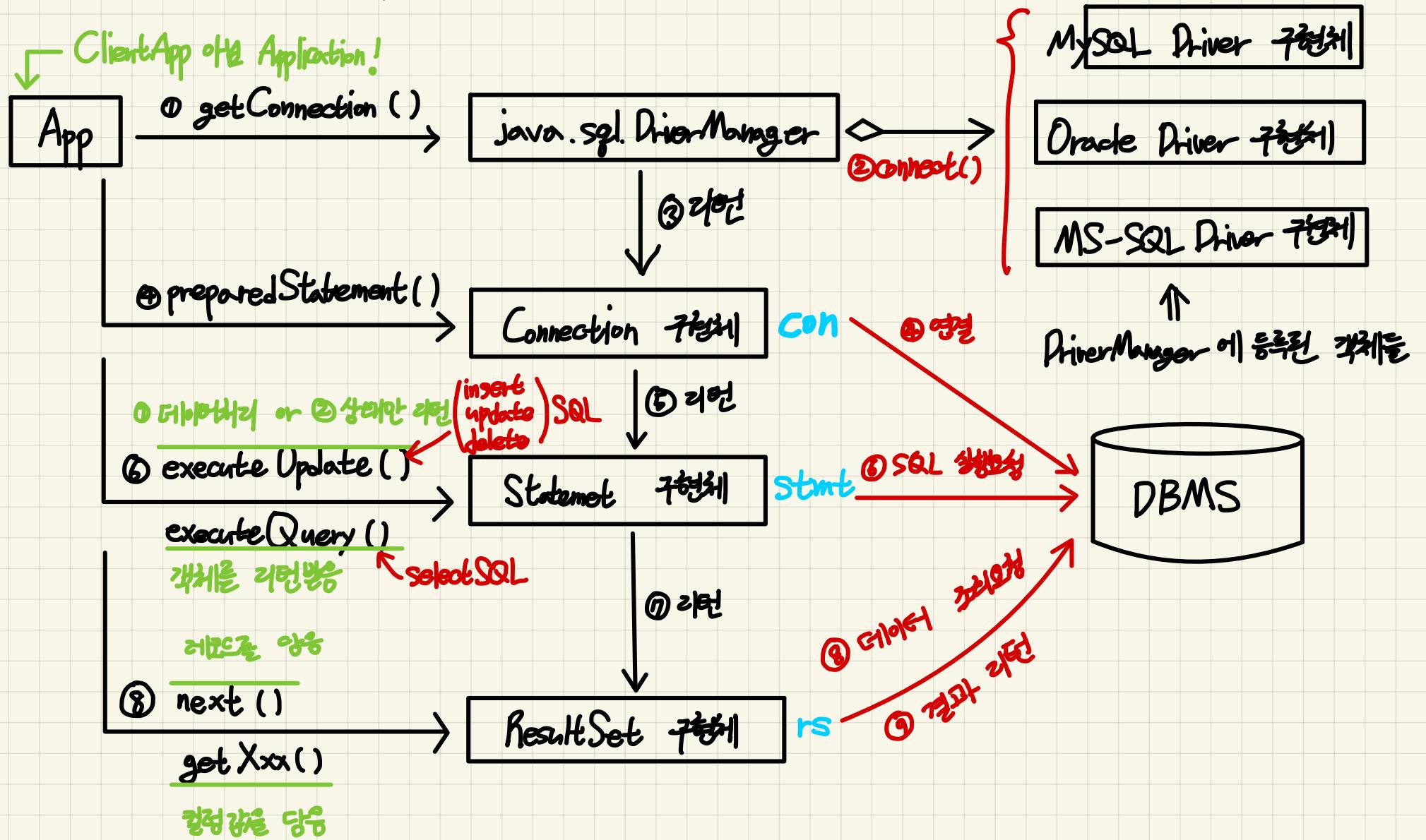


execute Update →   
→ 실행할 명령문.  
executeQuery →  
→ 결과를 반환함.

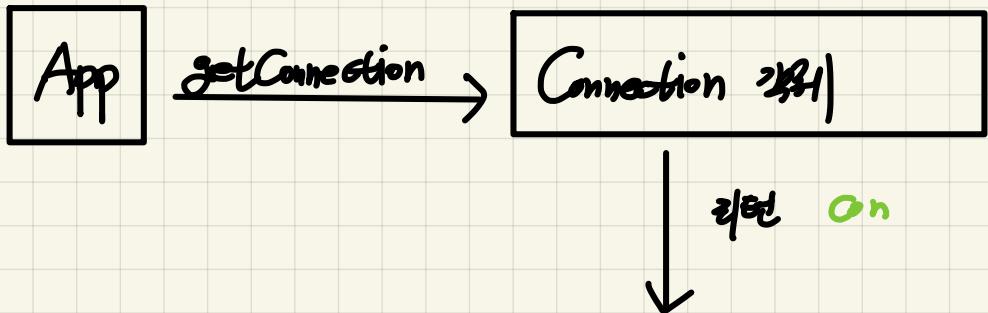
Update  
처리하는 명령어입니다.  
Result Set

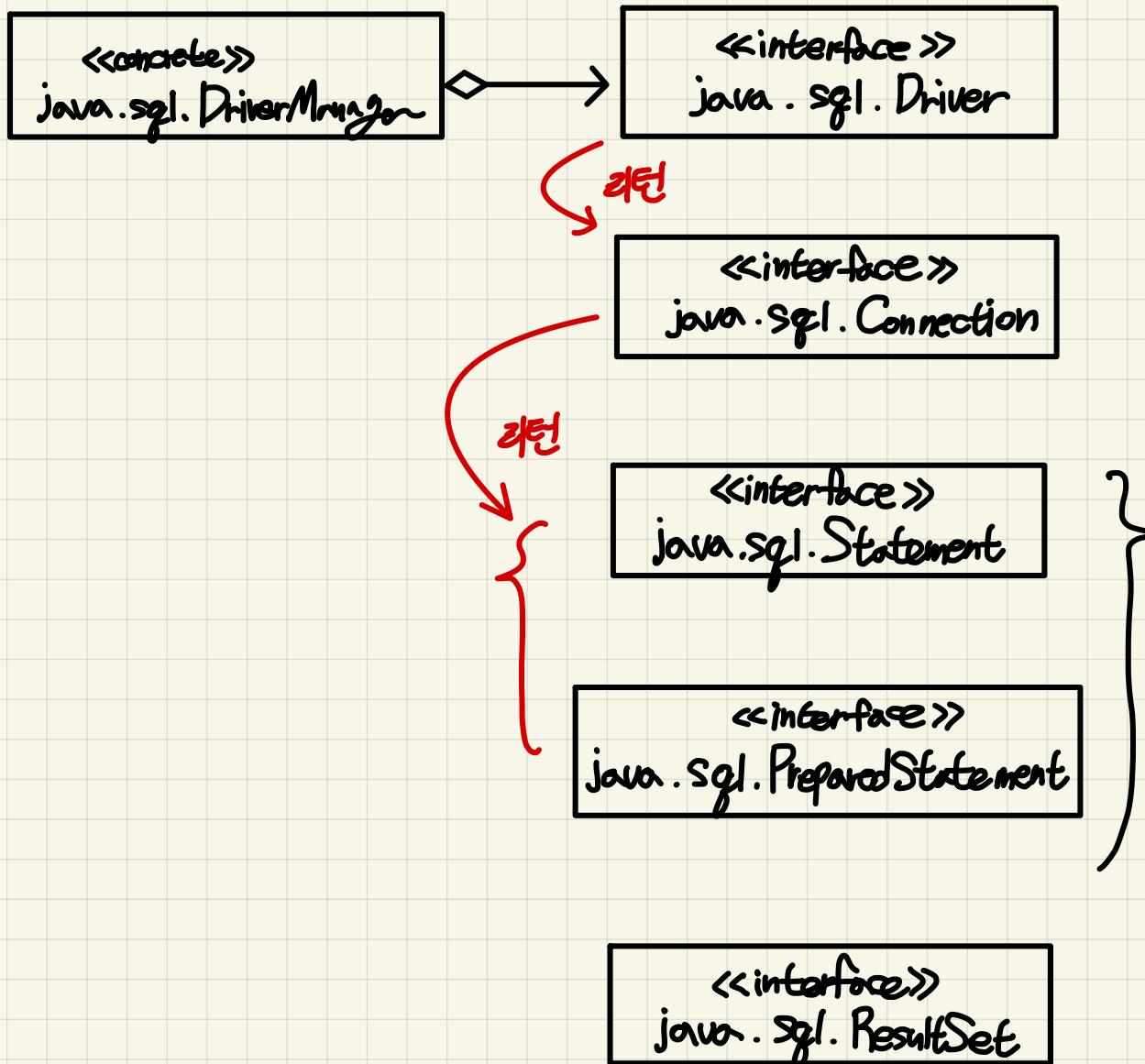
execute

## \* JDBC 프로그래밍 기본 울적



~~requestAgent = null; ~~삭제~~~~

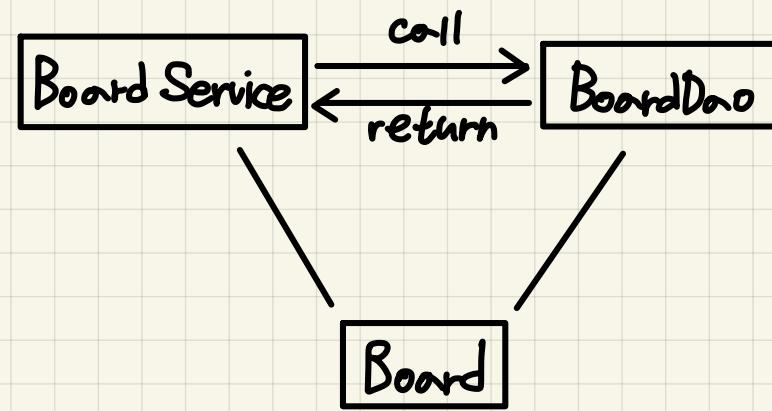




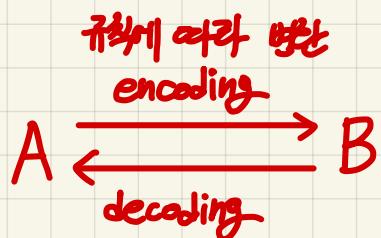
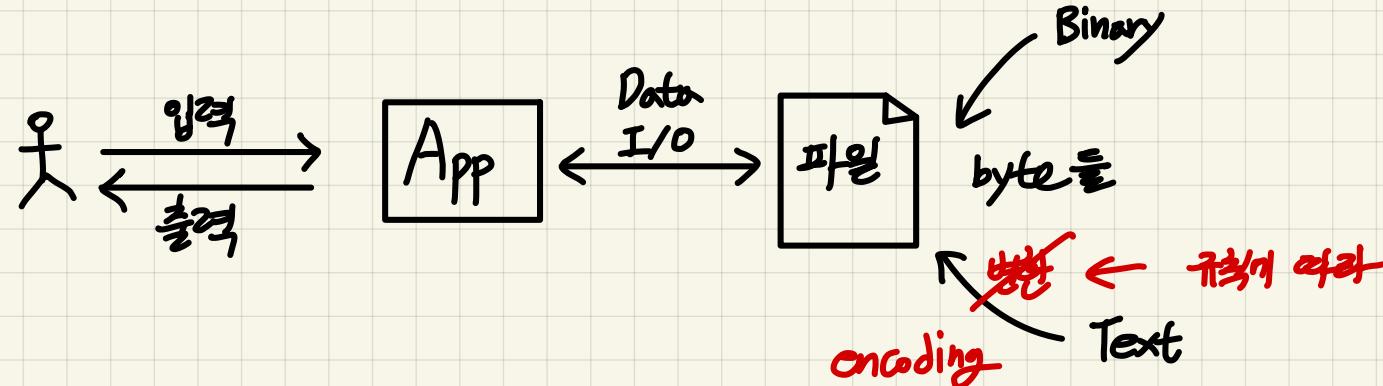
- JDBC 드라이버 정보 제공
- DBMS 연결 관리
- DBMS에 연결
- SQL 질의 수행 객체 생성

2021. 10. 12 (일) 15:50

\* DAO 와 VO



2021.10.13 (13주차) 9:30



Software as a Service = SaaS

Platform as a Service = PaaS

Auth : 인증과 권한을 아조함.

- 사용자 인증 (Authentication)
- 권한에 따라 처리 (Authorization)
- 데이터 권리 일관성
- 데이터 무결성 보장
- 동시 사용

---

Database

2021. 10. 13 (월) 6:00 DBMS 관계를

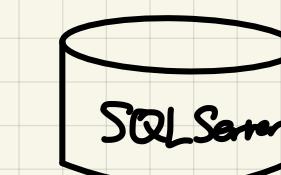
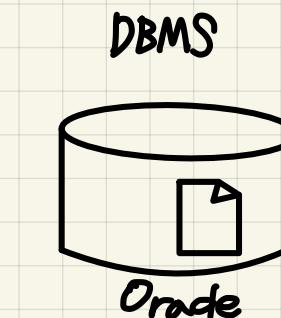
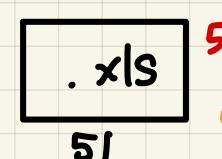
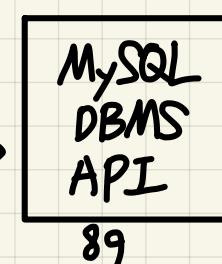
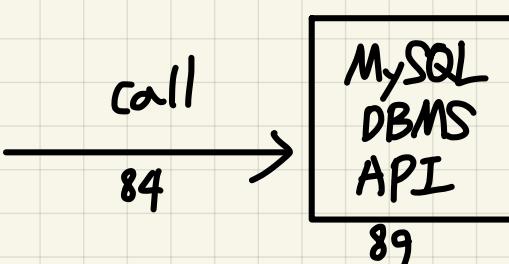
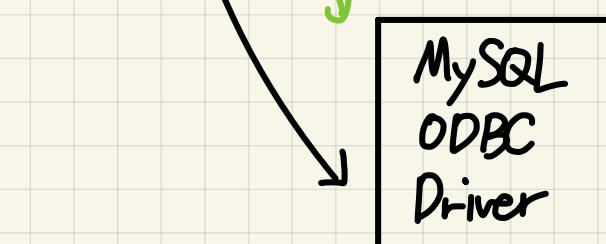
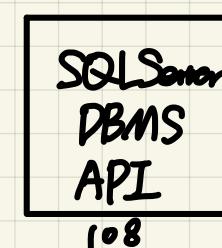
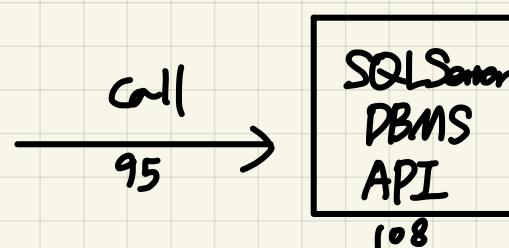
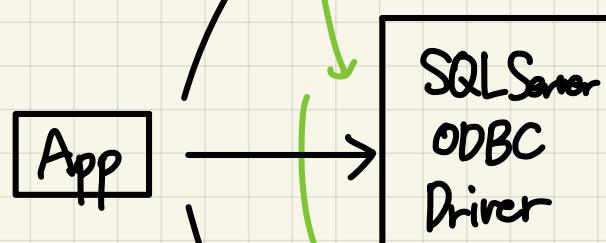
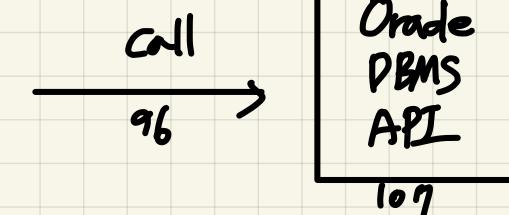
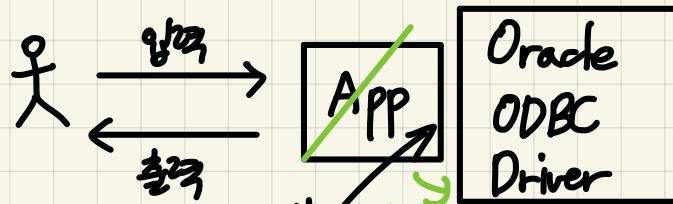
90 - DBMS 풍향기동 ↓ 향기

+  
10 - 유통기동 향기

↳ 물류와 생기는 유통기동

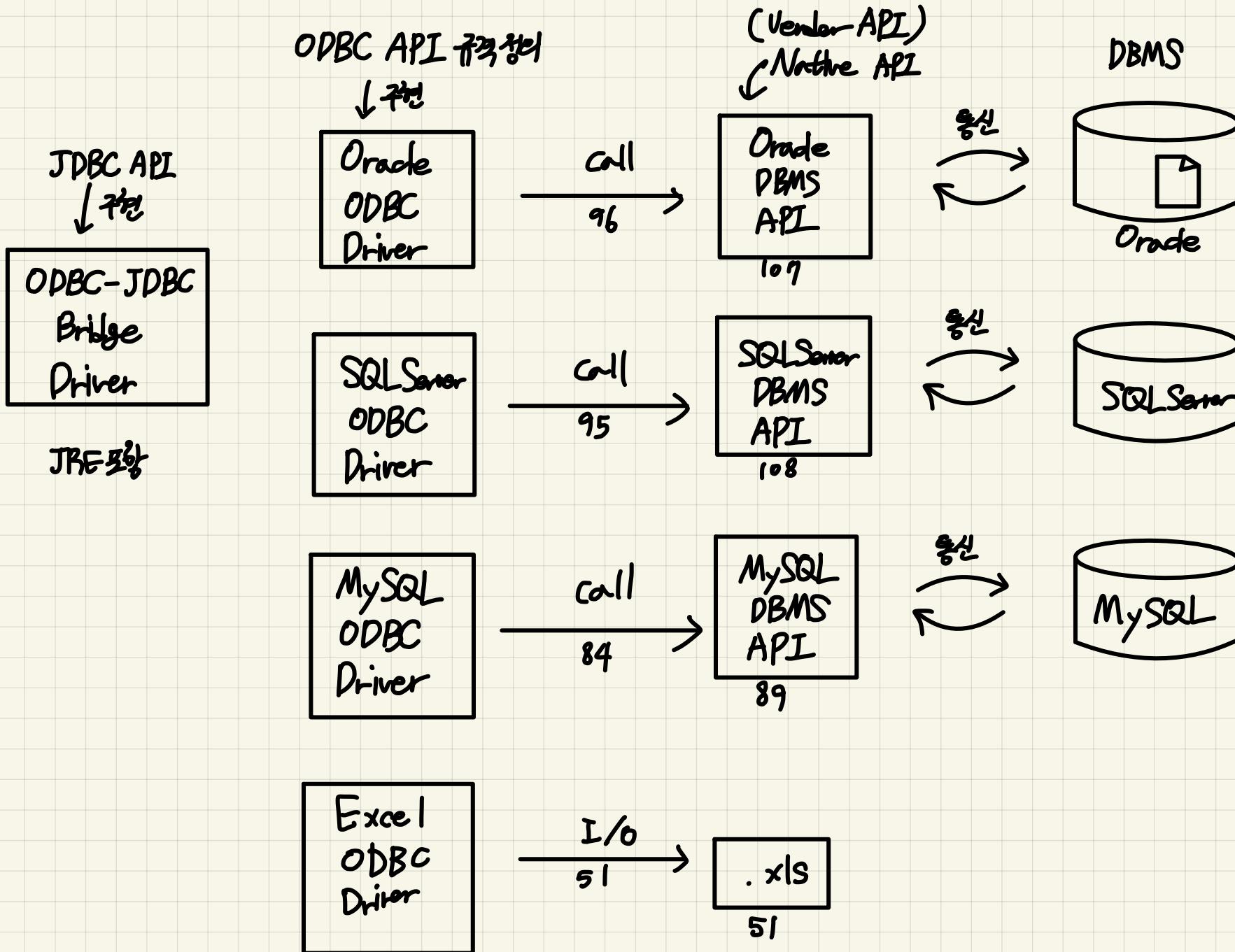
ODBC API 구조정리

↓ 주제 ↱ DBMS 능력만족만 주제! ①



DBMS마다  
네트워크통신  
다름

2021.10.13 (13주차) 6:30



\* JDBC API 주요 객체

<< concrete >>

java.sql.DriverManager

\* DAO 와 VO

\* Statement      vs      Prepared Statement

2021.10.13 (월) 14:10

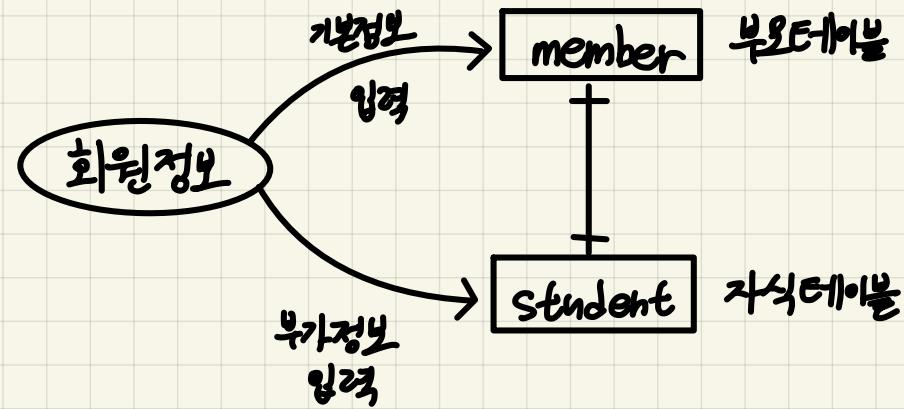
### \* Transaction

↳ 여러 개의 데이터 변경 작업을 한 단위로 묶은 것.

(insert, update, delete)

자동 커밋으로 설정

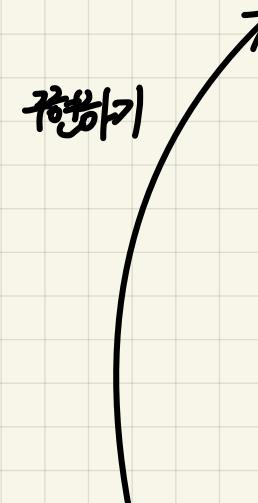
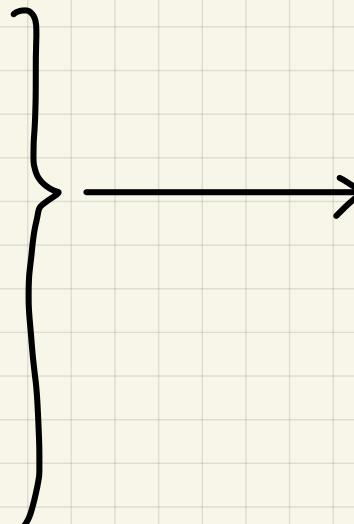
예1) 회원 정보 입력



회원정보를 2개의 테이블에  
분산해서 저장할 때,

2개 테이블에 모두  
정상적으로 데이터가 저장되어야만  
유효하다.

둘 중 한개라도 입력에 실패하면  
이전에 입력한 값은 취소해야 한다.



이렇게  
작업을

묶어서 이루어 하는  
"transaction"이라 한다.

`Set autocommit = false`

변경 작업 1

변경 작업 2

:

`Commit / rollback`

↑

모든 변경 작업이 성공했을 때  
선택 테이블이 적용이라고 명령을 내린다.

## 예2) 물품 구매하기

Set auto commit = false

한 차례로  
묶는다

{ 주문 테이블에 주문정보 입력  
결제 테이블에 결제정보 입력  
배송 요청 테이블 배송 요청 정보 입력 }

Commit

transaction

↳ 다른 말로

한 개의 업무를 가르치는  
언어다.

✓ 보험가입 ( 학원등록 → 보험등록 → 결제등록 )

✓ 수강신청 ( 수강신청 정보등록 → 결제등록 )



쉽게 기억의 규칙 예제의 경우와 같이  
여러 개의 작업을 한 번으로 다루는 경우가 많다.

2021.10.13 (13주차) 14:30

### 예3) 멤버 탈퇴

set autocommit = false ← 트랜잭션 시작

트랜잭션

{ update → 회원 정보 테이블에서 탈퇴 회원의 기본정보를 입력의 값으로 변경하고 비활성화로 설정한다.

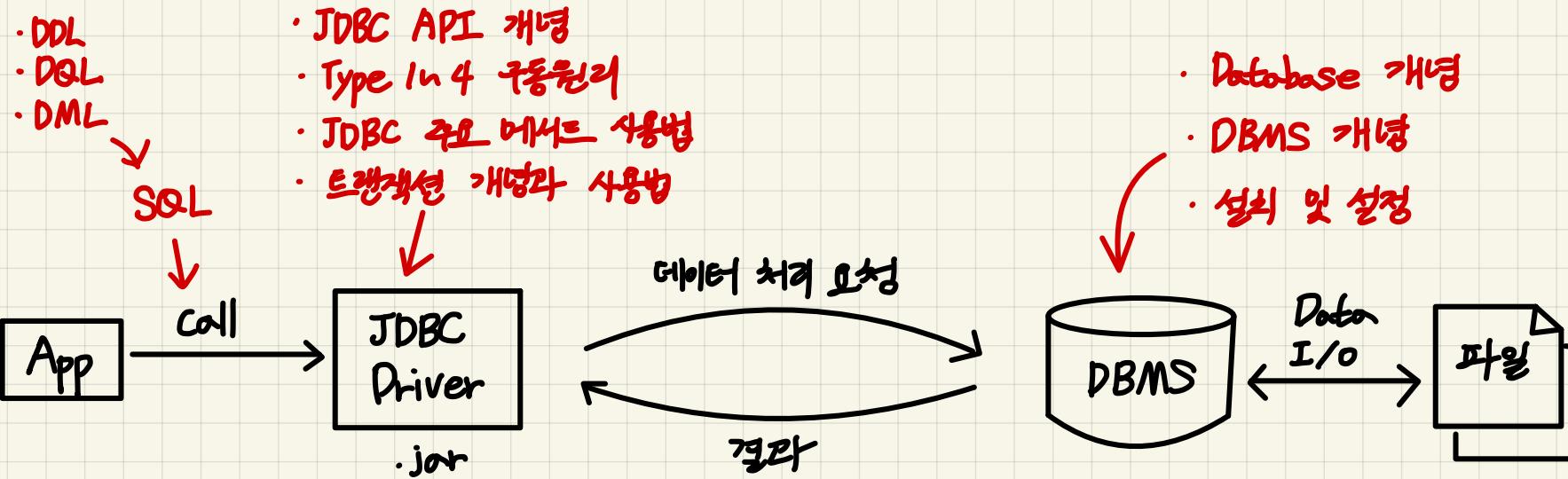
delete → 회원이 할당된 작업 정보 삭제

delete → 프로젝트 멤버 목록에서 제거

Commit ← 트랜잭션 종료

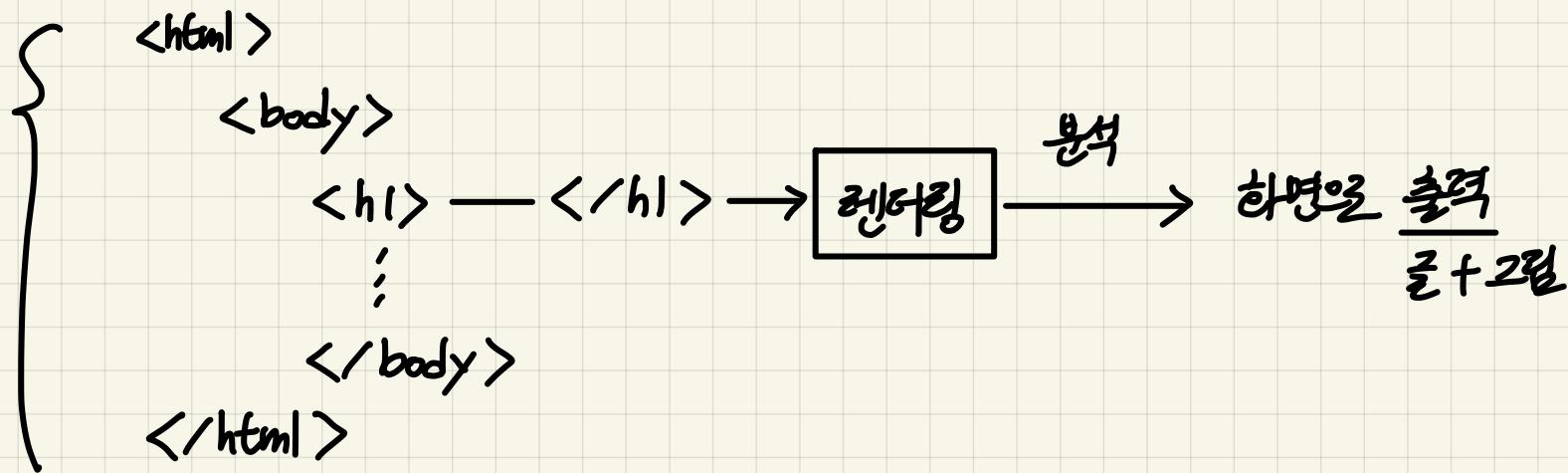
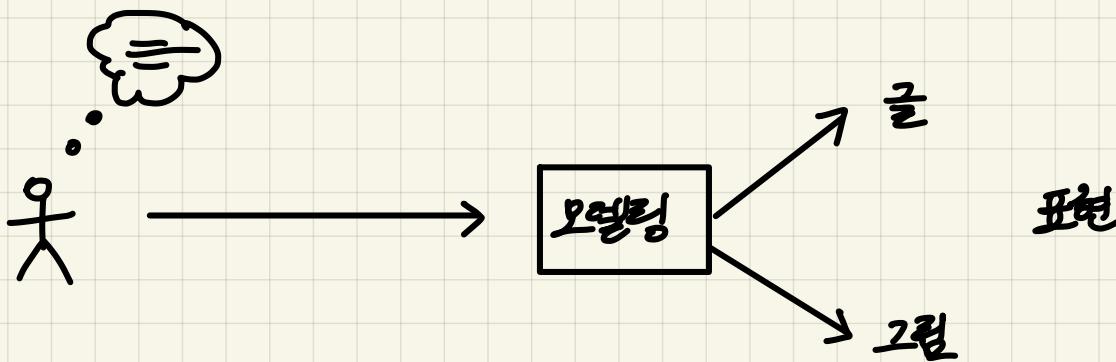
2021.10.14 (월) 9:30

\*

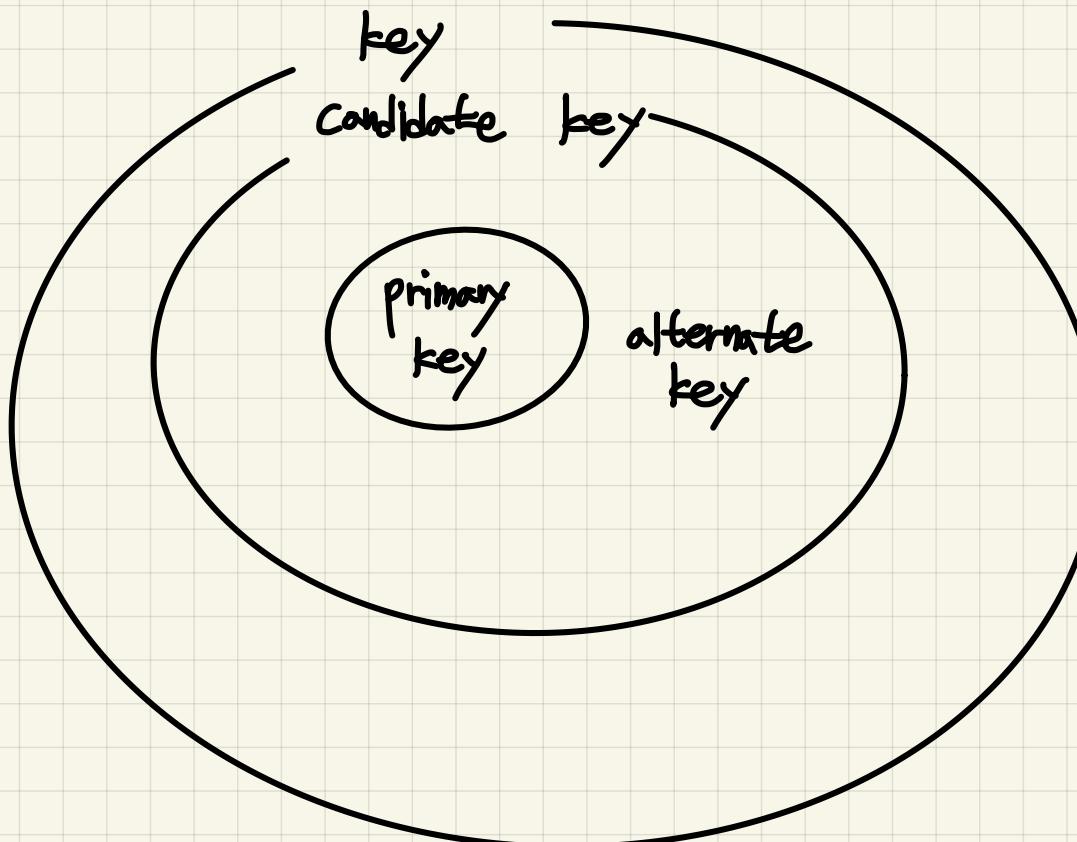


2021.10.14 (14주4) 10:00

\* DB 모델링



2021.10.14 ( 14주차 ) 10:08



+ Artificial key  
" "  
Surrogate key

① 논리 모델 정의 → ② 물리모델 정의 → ③ SQL 생성

↳ 일상적인 개념으로 표현

↳ DBMS에 맞춰서 표현

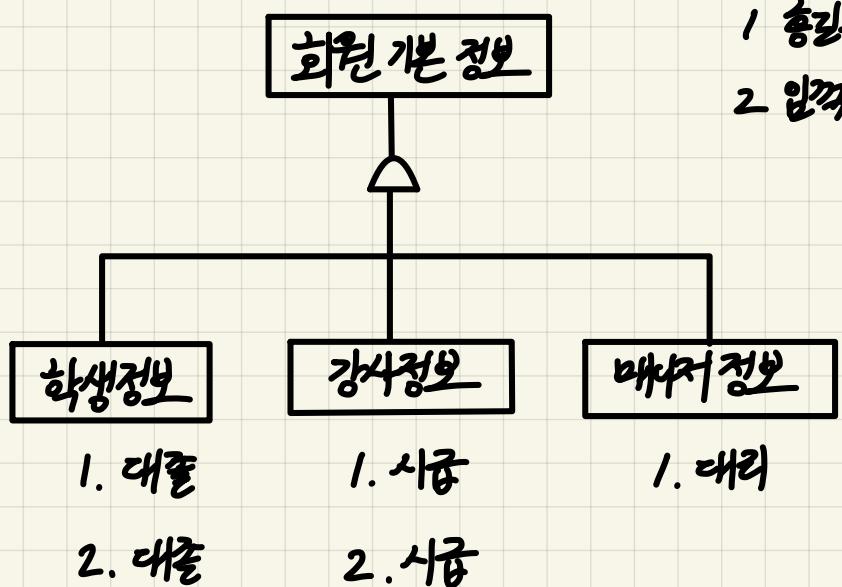
- 1) 엔티티 식별 및 컬럼 식별
  - 2) PK 설정 또는 인용키 도입
  - 3) 제 1 정규화 → 데이터 충복 및 컬럼 충복 제거
  - 4) 제 2 정규화 → 여러 PK에 종속되지 않은 데이터 식별
  - 5) 제 3 정규화 → 일반 컬럼에 종속된 데이터 삭제
  - 6) 다대다 관계 해소 / 포함관계, 배제관계 식별
  - 7) 한계치수 지정
  - 8) 대안키 → 유니크로 설정
  - 9) 커럼의 유통을 설정
- 1) DBMS 관계에 따라 테이블 및 컬럼명 설정
  - 2) 도메인 정의 및 적용
  - 3) 자동 증가 커널 지정
  - 4) 기본 값 및 제약 조건

2021.10.14 (74주차) 12:00

## \* 포함 관계와 배타적 관계

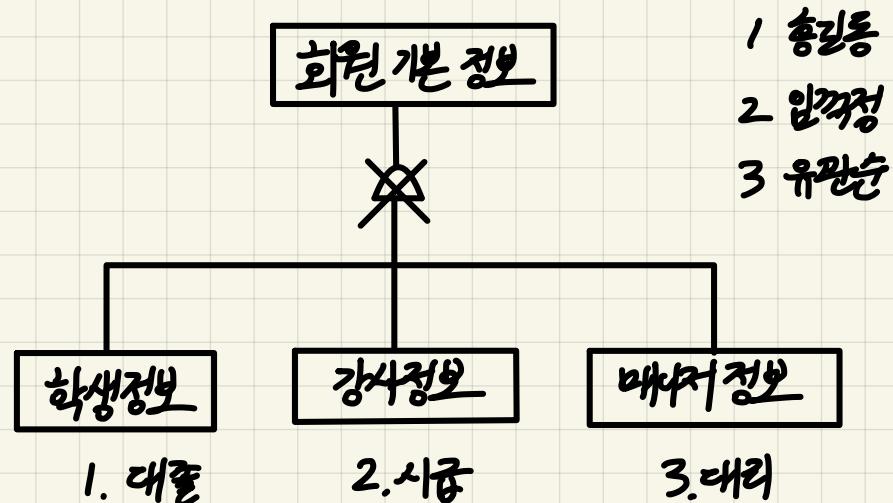
### ① 포함 관계 (include)

→ 여러개를 조합할 수 있다.



### ② 배타적 관계 (exclude)

→ 선종기 관계, 라이선스화  
콘텐츠와 안정

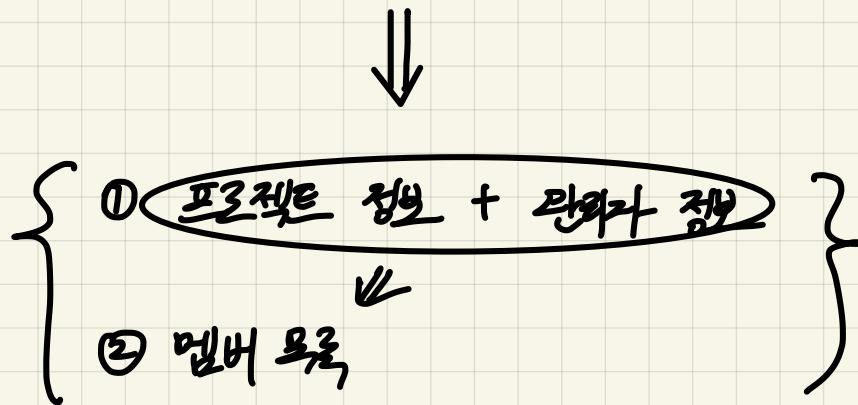


\* 회원 정보가 여러테이블에 포함될 수 있다.

\* 회원 정보가 3개의 테이블 중  
오직 한 개의 테이블만  
관계를 맺을 수 있다.

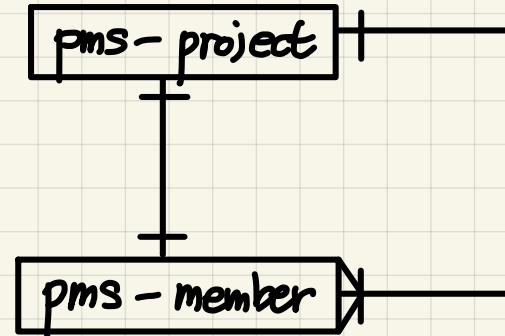
2021.10.18 (16주차) 13:14

프로젝트 정보 + 관리자 + 멤버 목록



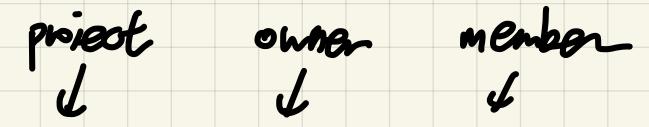
① 프로젝트 정보 + 관리자 정보 + 멤버 목록

100	
100	
100	

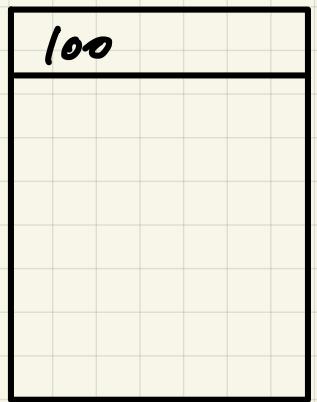


2021.10.18 (76回目)

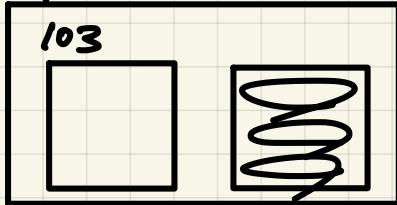
14:00



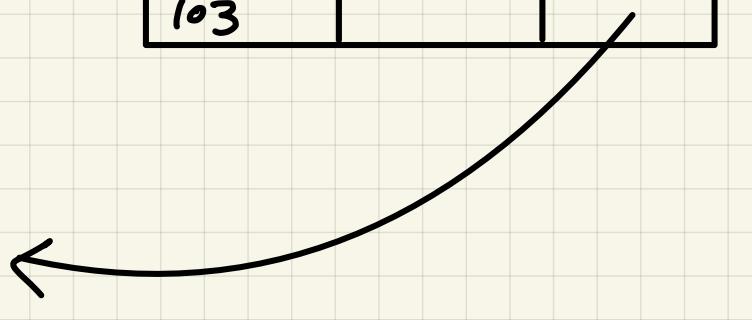
project\_no = 103



100 projects



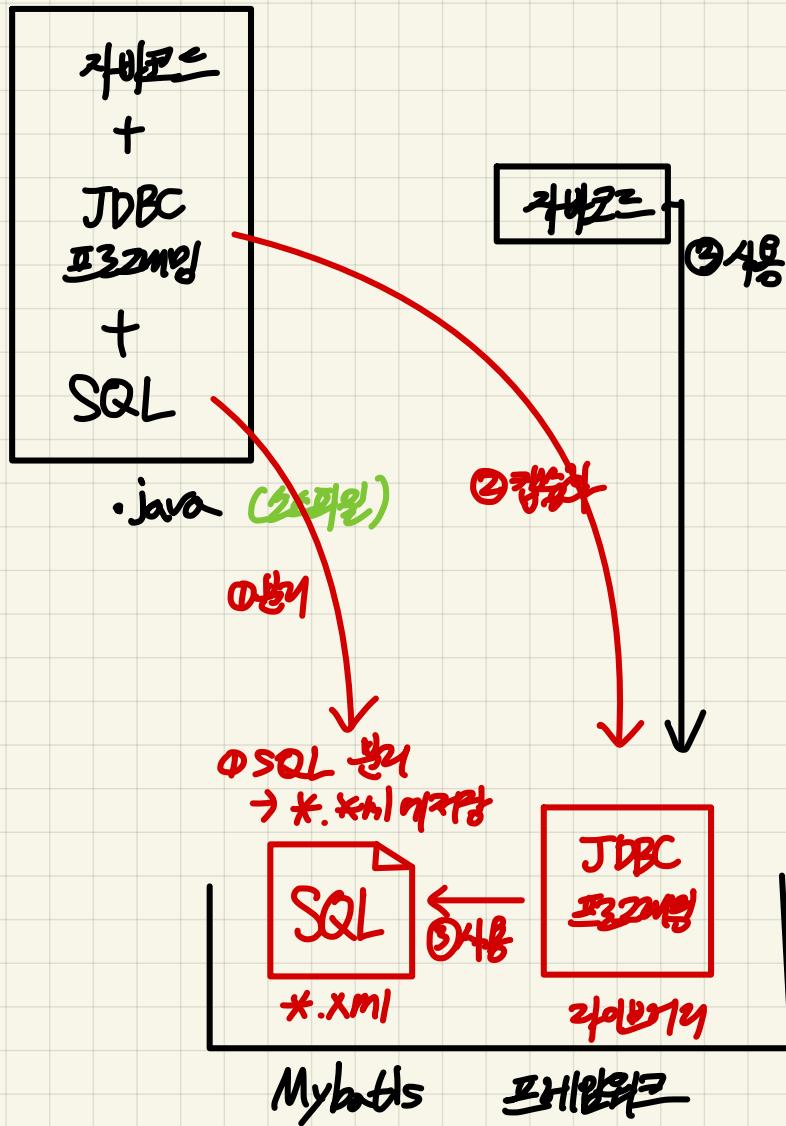
103			
-----	--	--	--



2021.10.20 (일요일) 9:33

\* Mybatis Persistence Framework ← 일종의 데이터베이스  
지속성  
어드레스 맵은 중간층.  
↳ 가능수행에 필요한 설정들을 미리 정의한 것  
= 자동으로 + 역할 수행을 위한 설정들

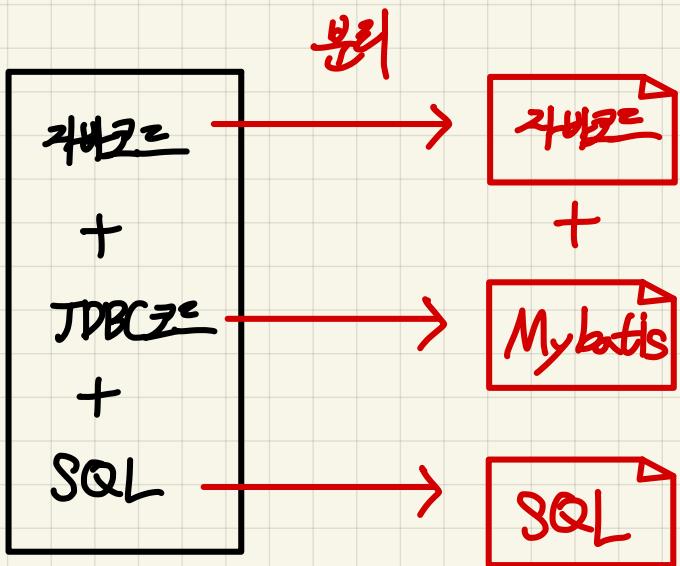
기존방식 → 솔루션(유연성이 개선)



2021.10.20 (일요일) 9:44

## \* Mybatis

Mybatis를 사용하는 이유 2가지

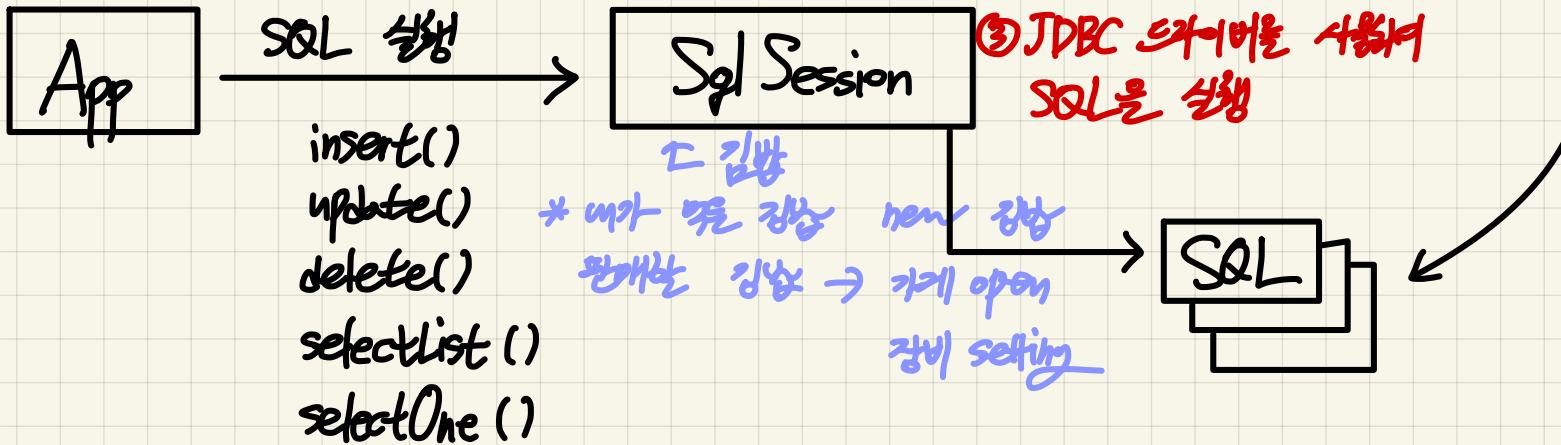
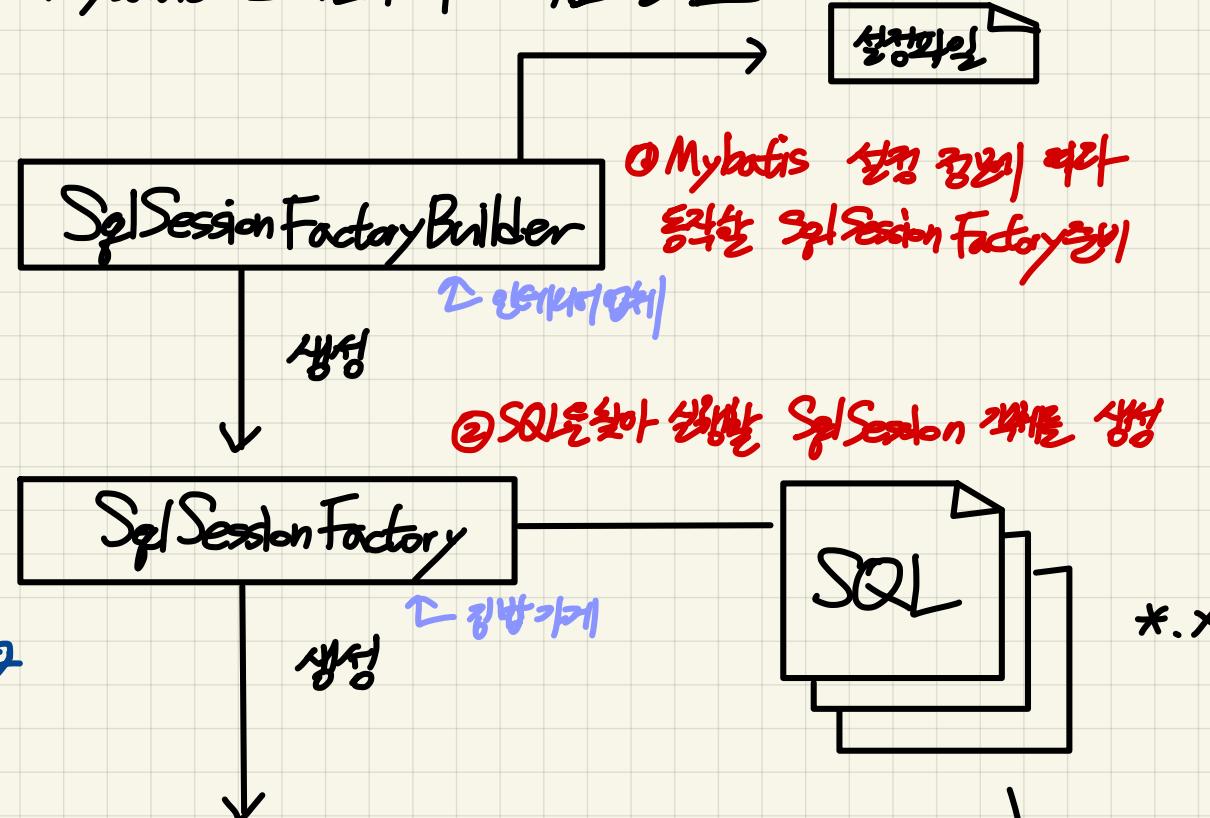


- SQL 코드를 다루기 편해
- DB 처리는

2021.10.20 (일요일) 9:48

## \* Mybatis 프레임워크의 핵심 클래스

- Builder 패턴  
↓  
여러 객체로 구성된 복잡한 주제의 객체를 생성할 때, 사용하는 설계 패턴  
예시: 조각, 차 등...

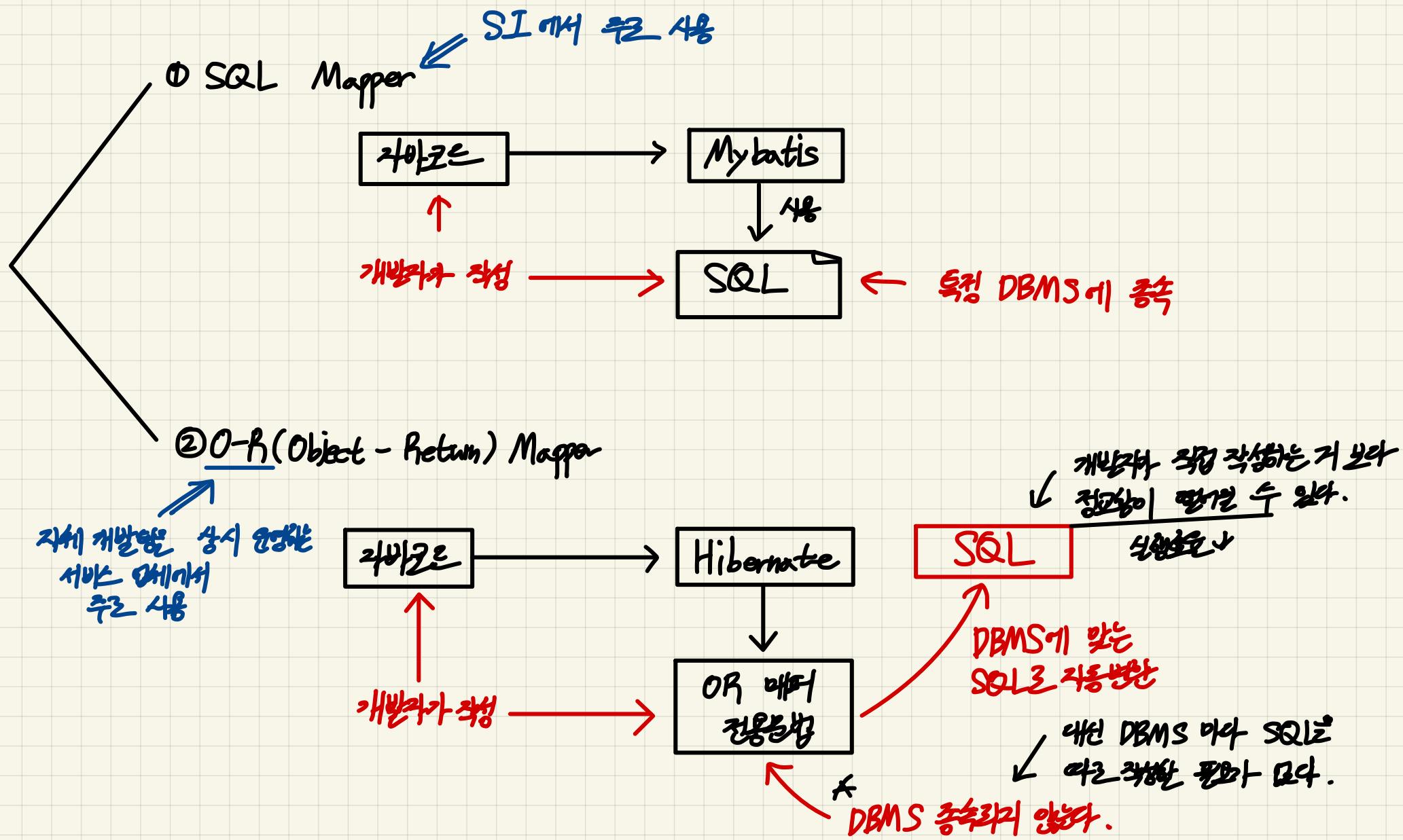


프록시  
SQL을 추출하여  
사용하기 편드록  
에코리기 조작

2021.10.20 (일요일)

10:15

## \* Data Persistence Framework



2021.10.20 (星期一) 10:44

## \* XML

## 가장 바깥쪽 애드

< ? xml version = "1.0" encoding = "UTF-8" ? > < XML 선언

XML 선언

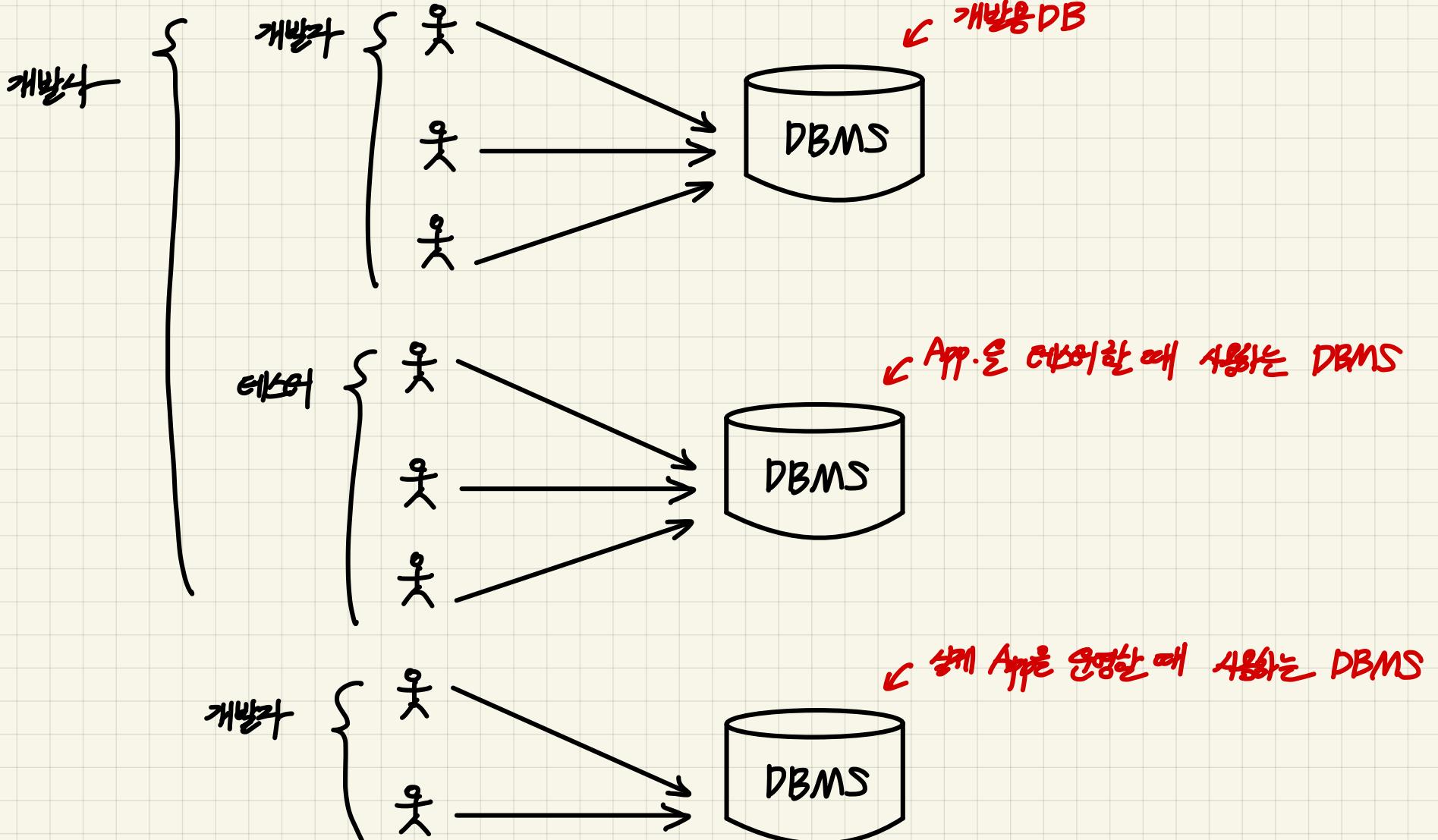
반드시 첫번째 줄에 있어야 한다.

요소 단위! (root element)  
기준!

2+ < mappers > ↘ 매핑  
  < mapper >      ↘ 속성의 값  
  </mappers>      resource = " \_\_\_\_\_ "/>  
</configuration>      ↑ 속성 (attribute)      ↑ 줄바이를 생략할 때 </>로 표기.

2021.10.20 (일) 10:55

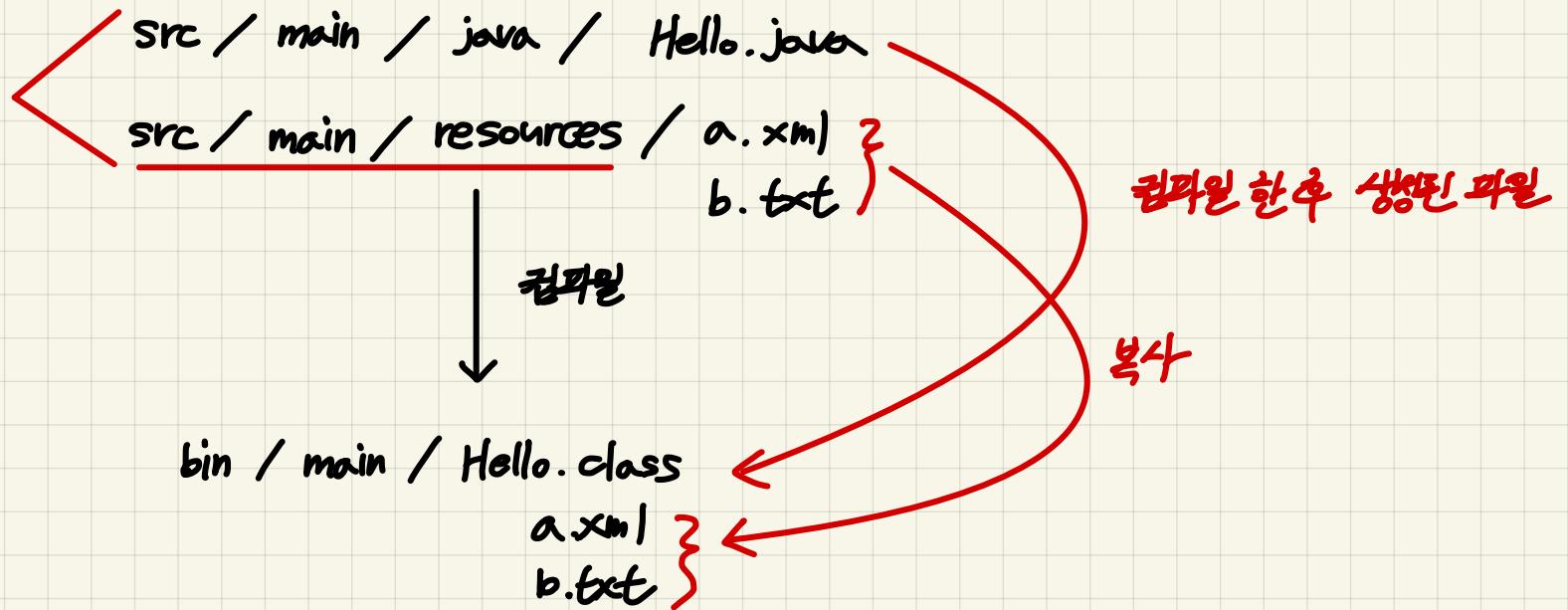
## \* 월의 DBMS 구성



2021.10.20 (일) 11:07

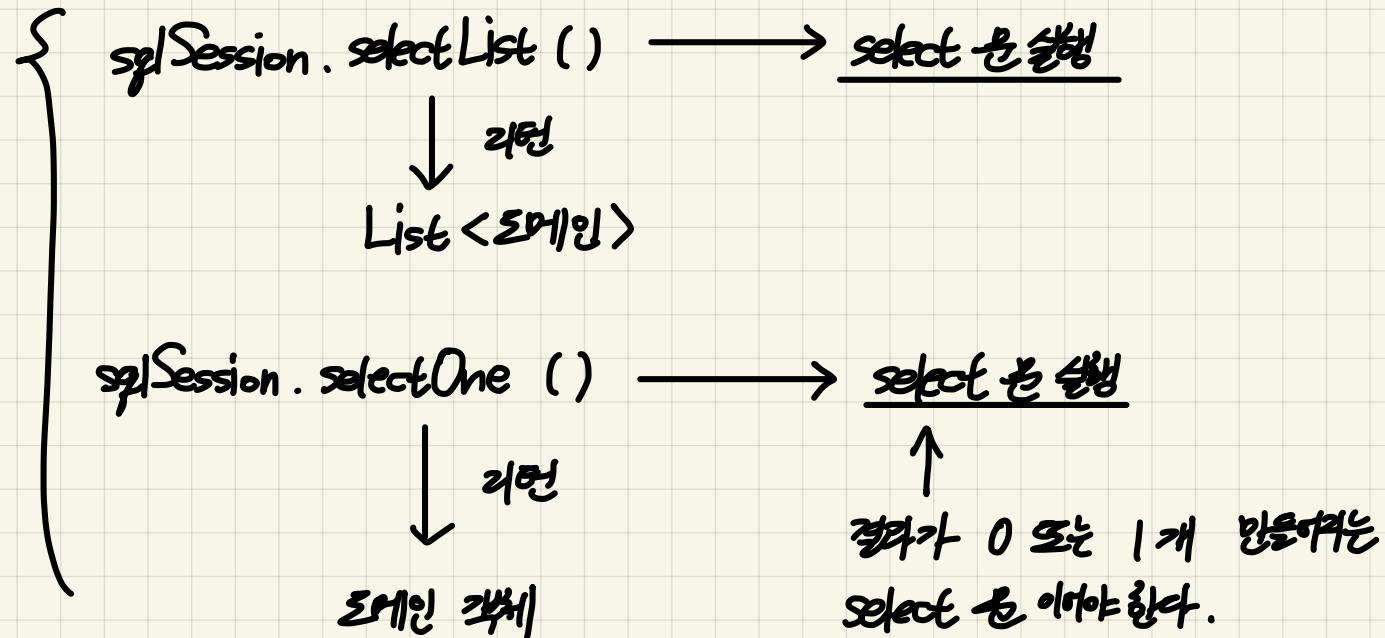
\* 소스파일 컴파일 - 자바소스와 일반파일 분리해서 관리

둘다  
자바소스  
풀어보  
간주한다.



2021.10.20 (일요일) 11:51

## \* SqlSession - selectList() 와 selectOne()



< 결과물 출력>

결과 이름 = API 인 경우 ↓  
이름.  
or

SoftBox

Contents 를 API 로 얻어온다.

ResultType = API 는 간단히

BoardMap = boardMap or 풍어한 Board 를 만들고.

BoardMap = PresenterMap의 Id 일

could not resolve type alias : (error)



ResultMap = "Board Map"

result ~~Type~~ = "Board Map"

could not find Result Map

< select id = 'selectBoard' resultType = 'Board'>

이미 Glance 라

BoardMap 이 ~~가져온다~~ <sup>보여준다</sup> "Board" 안을 갖다

| 갖다 | 쓰는거

같은 이유로 property의 경우

2021.10.20 (月) 14:55