



영 진 전 문 대 학 교

글로벌시스템융합과 - GSC

PHP – DB & MySQLi

정영철 교수

글로벌시스템융합과

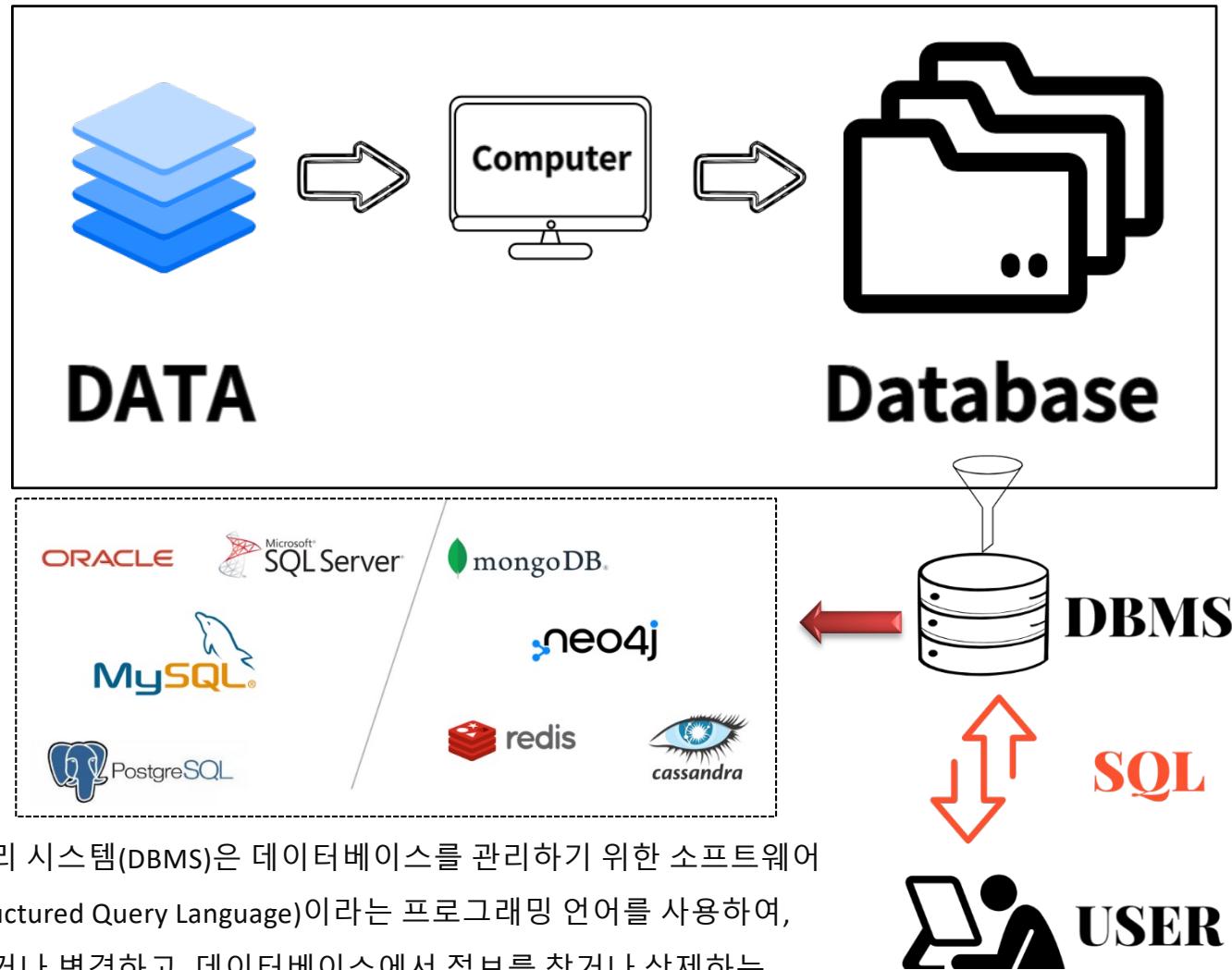


A.I WITH BETTER LIFE

데이터베이스(Database)란?

- 데이터베이스는 컴퓨터 시스템에서 데이터를 체계적으로 저장하고 관리하는 집합

DB를 사용하면 사용자는 데이터를 효율적으로 검색, 수정, 분석할 수 있다



- ✓ 데이터베이스 관리 시스템(DBMS)은 데이터베이스를 관리하기 위한 소프트웨어
- ✓ 사용자는 SQL(Structured Query Language)이라는 프로그래밍 언어를 사용하여, 데이터를 추가하거나 변경하고, 데이터베이스에서 정보를 찾거나 삭제하는 요청을 DBMS에 전달할 수 있다.

데이터베이스(Database)의 종류

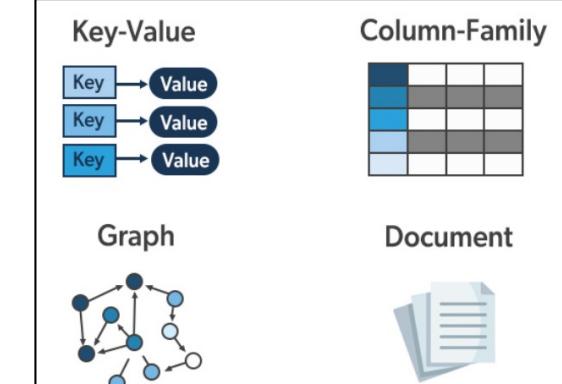
Database types (데이터베이스 종류)

Relational Database (관계형 데이터베이스)

A	B	C	D	E	F
1	스키마 (Shema)	열(Field)	열(Field) 2	열(Field) 3	열(Field) 4
2		순번	이름	학번	전화번호
4	행(Record) 1	1	홍길동	212	111-1111
5	행(Record) 2	2	김영희	333	111-1112
6	행(Record) 3	3	김철수	313	111-1113
7	행(Record) 4	4	마이클	342	111-1114
8	행(Record) 5	5	타나카	342	111-1115

- 정의: 데이터를 테이블 형태로 저장하는 데이터베이스
각 테이블은 열(COLUMNS)과 행(ROWS)을 가지며, 열은 데이터의 속성을, 행은 데이터 항목(또는 레코드)을 나타낸다. 테이블 간에는 관계(Relationships)를 통해 연결될 수 있다.
- 특징: 데이터의 구조를 사전에 정의(스키마)해야 하며, SQL(Structured Query Language)을 사용하여 데이터를 관리한다.
- 예시: MySQL, Oracle Database, Microsoft SQL Server

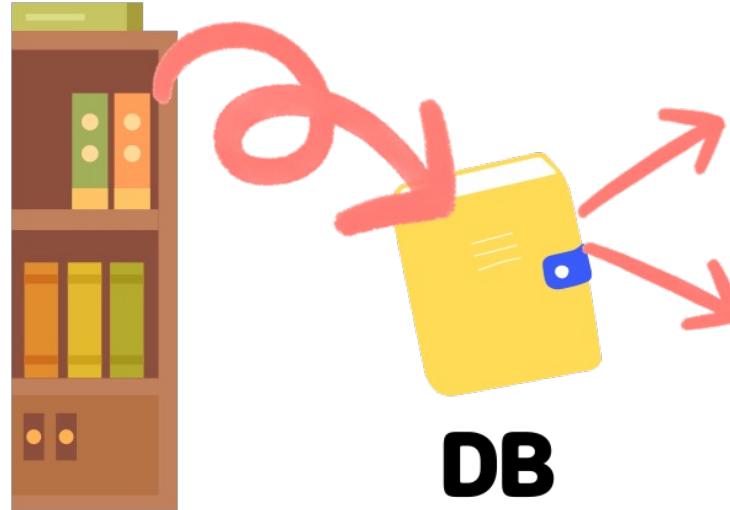
NoSQL Database (비관계형 데이터베이스)



- 정의: 관계형 데이터베이스와 달리, 비관계형 데이터베이스는 테이블, 행, 열의 구조를 따르지 않는다. 다양한 데이터 저장 모델을 사용하여, 유연하게 데이터를 저장하고 관리
- 유형:
 - 문서 지향 데이터베이스(Document-Oriented Database)
 - 키-값 저장소(Key-Value Store)
 - 와이드 컬럼 스토어(Wide-Column Store)
 - 그래프 데이터베이스(Graph Database)

관계형 데이터베이스(Relational Database)의 구성 요소

DBMS



Table

	A	B	C	D
1	순번	이름	학번	전화번호
2	1	홍길동	212	111-1111
3	2	김영희	333	111-1112
4	3	김철수	313	111-1113
5	4	마이클	342	111-1114
6	5	타나카	342	111-1115

	A	B	C	D
1	순번	이름	학번	전화번호
2	1	홍길동	212	111-1111
3	2	김영희	333	111-1112
4	3	김철수	313	111-1113
5	4	마이클	342	111-1114
6	5	타나카	342	111-1115

- **Table** : 관계형 데이터베이스의 기본 관리 단위, Matrix 구조의 종이 한 장과 같음
 - 2차원 구조: 행(Rows)과 열(Columns)
 - 행: 개별 데이터 레코드
 - 열: 데이터의 속성(필드)
- **Database(DB)** : 관련된 데이터를 저장하는 여러 테이블의 모임
 - 각 테이블은 서로 연관된 데이터를 저장하며, 복잡한 데이터 집합을 효율적으로 관리
- **Database Management System(DBMS)** : 여러 데이터베이스들의 모임 [책장]
 - 데이터베이스를 생성, 관리, 조작하고 상호작용하는 데 사용되는 소프트웨어 시스템

DBMS의 종류

RDBMS

ORACLE



Microsoft®
SQL Server®



PostgreSQL

NoSQL



mongoDB®

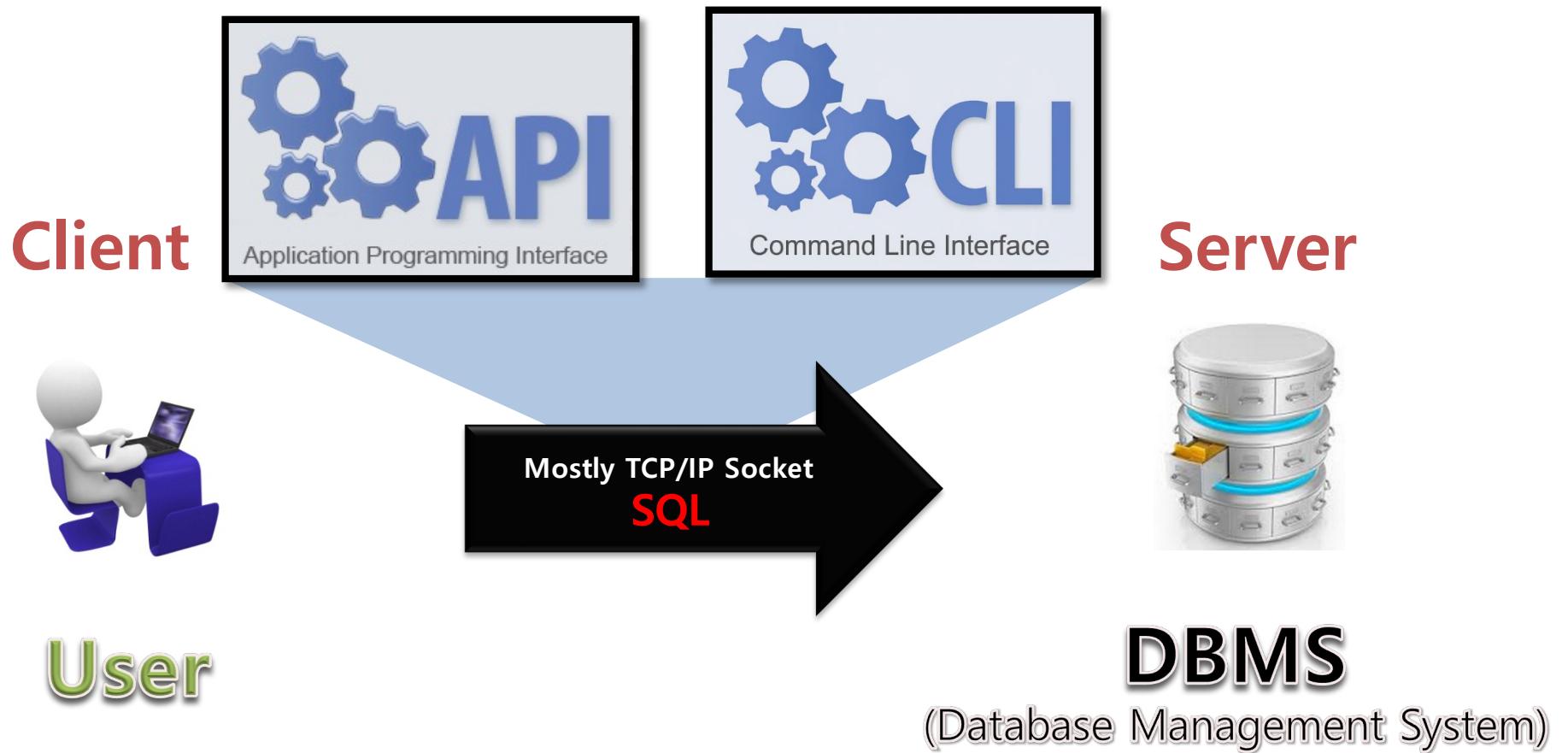


redis



cassandra

DBMS의 접근 방법?

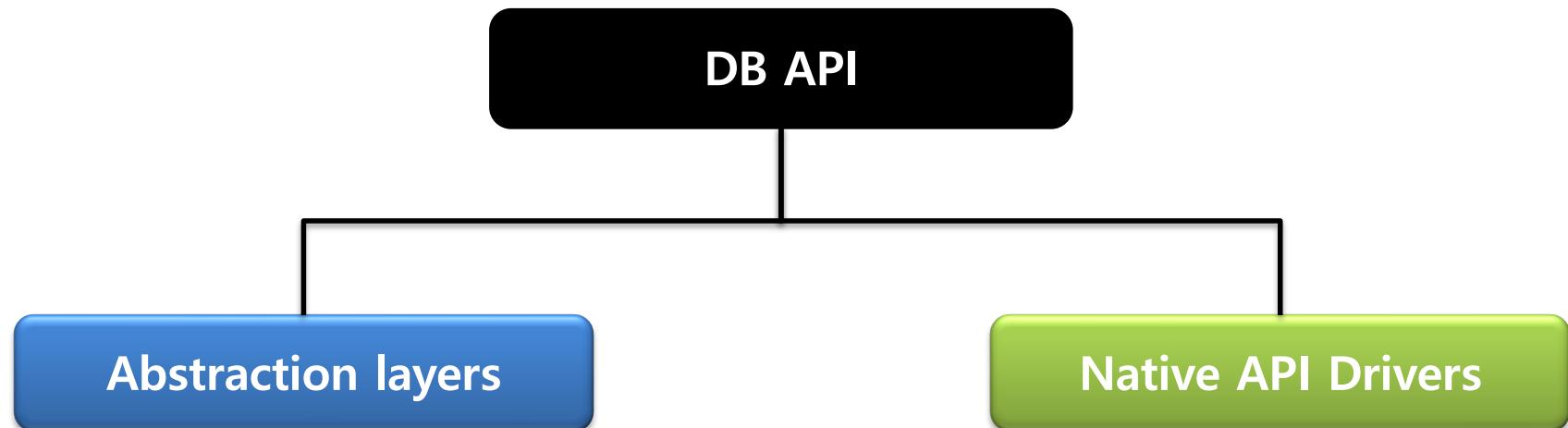


✓ DBMS는 서버로 동작한다 (일반적으로...)

- DBMS는 항상 **서버 프로그램**(예: mysqld)으로 동작
- 클라이언트(API, CLI)는 이 DB 서버에 접속하여 요청을 전송함
- DBMS 서버는 **SQL(Structured Query Language)** 요청을 처리하고 결과를 반환함

🔄 구조: 클라이언트 ↔ (소켓 통신) ↔ DB 서버(mysqld 등)

DBMS 접근을 위한 API 종류



- **Abstraction layers (Open standard API)**

- 다양한 DBMS에 대해 일관된 방식으로 접근할 수 있도록 추상화한 API

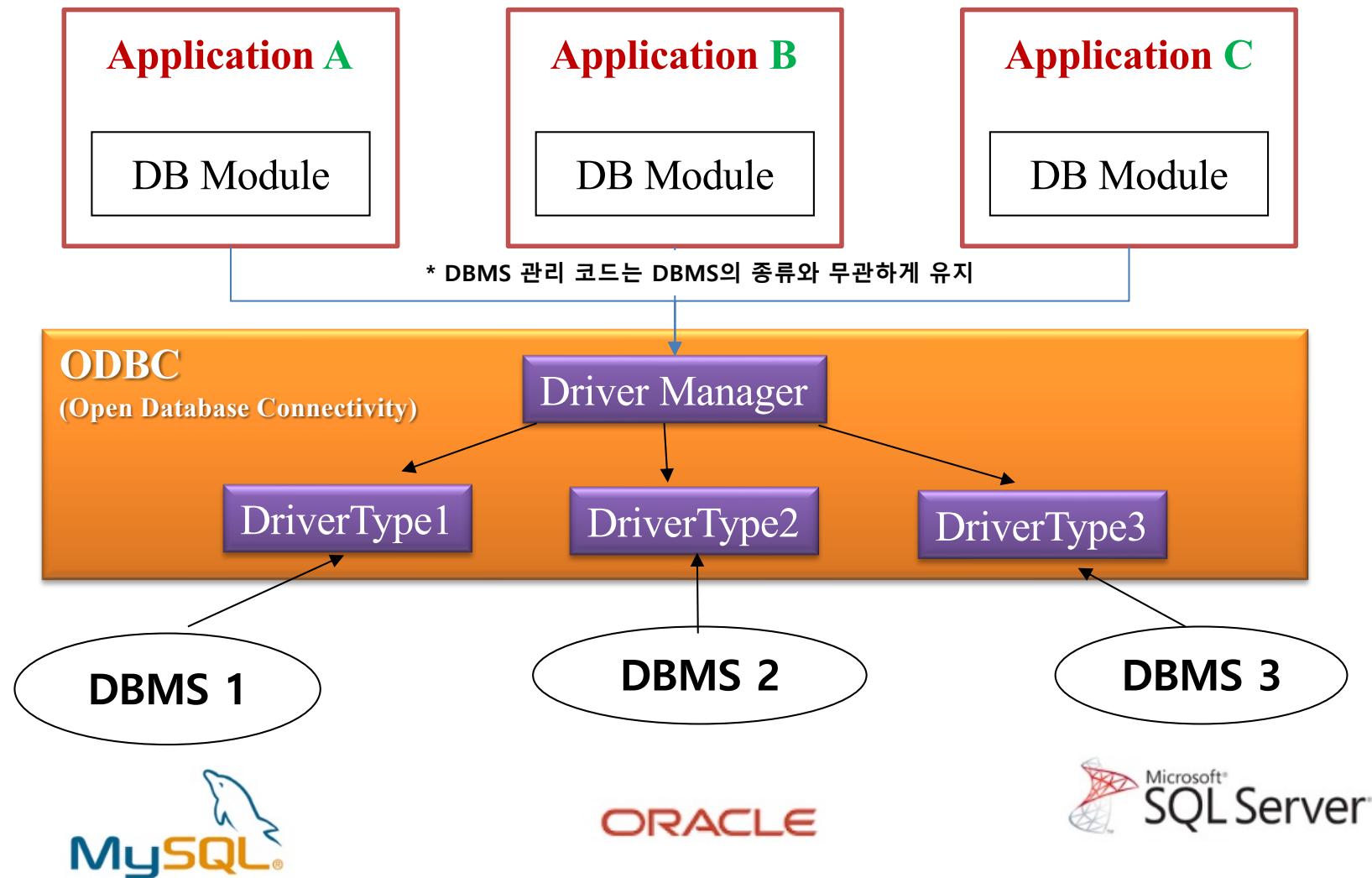
- **Native API Driver**

- DBMS 회사별로 제공하는 API

이름	주 사용 언어	설명
ODBC (Open Database Connectivity)	C/C++ 등	Microsoft가 제안한 표준 DB 인터페이스. 다양한 DB에 연결 가능
JDBC (Java Database Connectivity)	Java	Java에서 DB 접근을 위한 표준 인터페이스
PDO (PHP Data Object)	PHP	다양한 DB를 일관된 방식으로 제어하는 PHP용 인터페이스

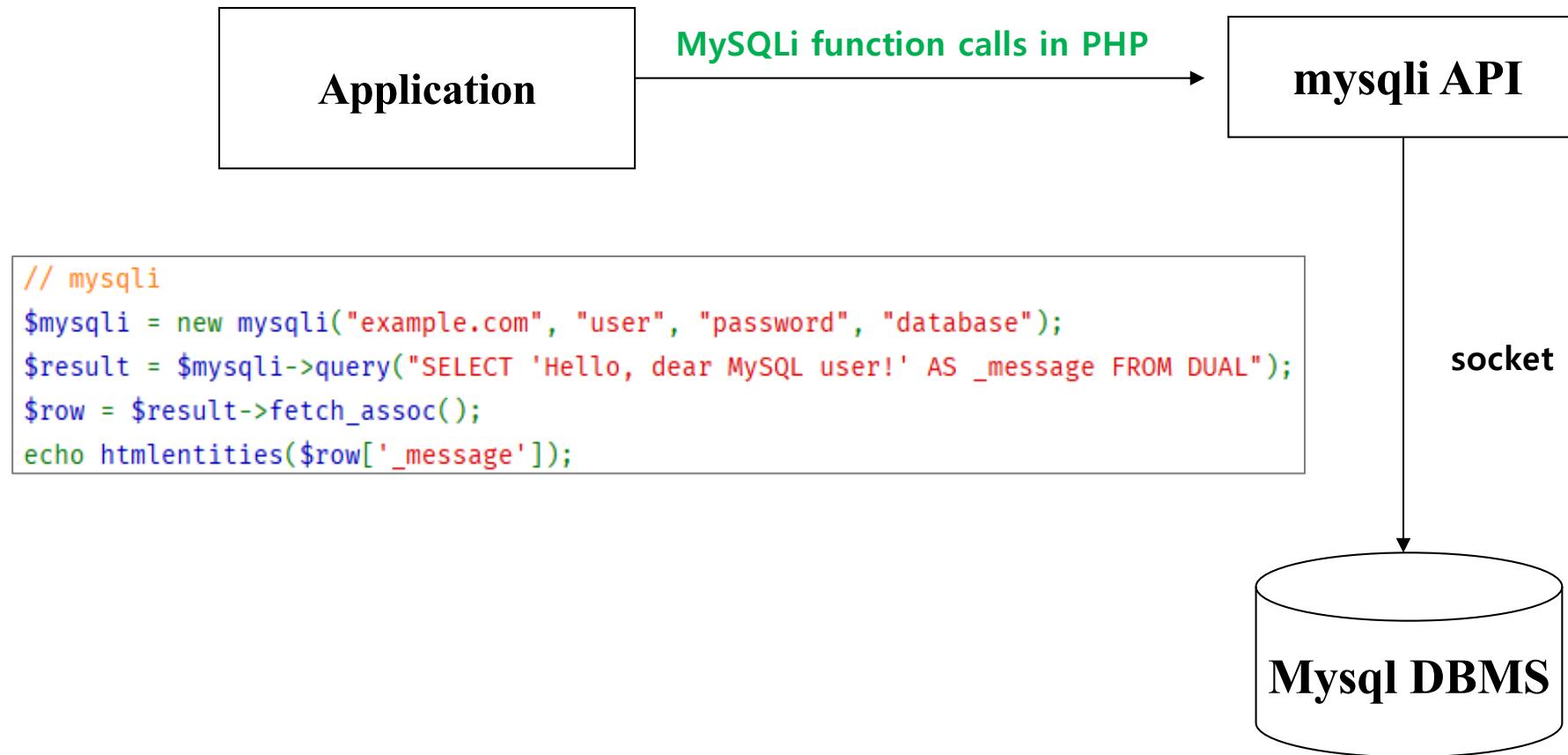
DBMS	사용 언어	Native API 예시
MySQL	PHP	mysqli_*, mysql_* 함수
PostgreSQL	C, PHP, Python 등	libpq, pg_connect()
Oracle	Java, C	OCI, Oracle JDBC Driver
SQL Server	C#, .NET	SqlClient,

Abstraction layer: ODBC 구성 예제



- 표준 인터페이스는 다양한 DBMS를 동일한 방식으로 접근할 수 있도록 해주는 추상화 계층을 제공
- 개발자는 ODBC, JDBC, PDO와 같은 표준화된 메서드와 SQL 문법을 통해 DBMS와 상호작용 실행
- 이를 통해 어떤 종류의 DBMS를 사용하더라도, 애플리케이션 코드의 변경 없이도 동일한 방식으로 DB 작업이 가능

Native API : mysqli 사용 예



- Native API는 각 DBMS 제조사 또는 커뮤니티에서 제공하는 고유한 API
- 특정 DBMS에 최적화되어 있으며, 해당 DBMS의 기능을 가장 직접적이고 고성능으로 제어할 수 있다
- 다만, DBMS 간 호환성이 떨어지며, 특정 DBMS에 종속(Vendor Lock-in)되는 단점

PHP에서 MySQL DBMS 사용 방법 : mysqli & PDO

	ext/mysqli	PDO_MySQL
PHP version introduced	5.0	5.1
Included with PHP 7.x and 8.x	Yes	Yes
Development status	Active	Active
Lifecycle	Active	Active
Recommended for new projects	Yes	Yes
OOP Interface	Yes	Yes
Procedural Interface	Yes	No
API supports non-blocking, asynchronous queries with mysqlnd	Yes	No
Persistent Connections	Yes	Yes
API supports Charsets	Yes	Yes
API supports server-side Prepared Statements	Yes	Yes
API supports client-side Prepared Statements	No	Yes
API supports Stored Procedures	Yes	Yes
API supports Multiple Statements	Yes	Most
API supports Transactions	Yes	Yes
Transactions can be controlled with SQL	Yes	Yes
Supports all MySQL 5.1+ functionality	Yes	Most

mysqli 특징

- MySQLi (MySQL Improved)는 기존 mysql_* 함수의 향상된 확장(extension)
- PHP 5.0 이상부터 지원, MySQL 4.1 이상부터 사용 가능
- 기존 mysql_* 함수는 PHP 7.0부터 완전히 제거됨.

항목	설명
지원 방식	- 객체지향(OOP) 및 절차지향(Procedural) 인터페이스 모두 지원
보안	- Prepared Statement 제공 → SQL Injection 방지에 유리
성능	- Persistent Connection 지원 (옵션 사용)
기능	- Multiple Query Execution 가능 (multi_query()) - Stored Procedure 및 Transaction 지원
확장성	- mysqli_driver, mysqli_result, mysqli_stmt, mysqli_warning 등의 객체 클래스 지원
버전 정보	- PHP 8.x에서도 계속 유지되며, PDO와 함께 대표적인 DB 연동 확장 으로 사용됨

CLI를 이용한 SQL 기본 명령어 실습

DBMS 접속, DB&Table 생성, 삭제, 수정, 보기

CLI를 이용하여 MySQL 서버 접속

```
.env  
1 DB_PASSWORD=root
```

```
sh-5.1# mysql -u root -p
```

```
Enter password:
```

```
Welcome to the MySQL monitor. Commands end with ; or \g.
```

```
Your MySQL connection id is 35
```

```
Server version: 8.0.42 MySQL Community Server - GPL
```

```
Copyright (c) 2000, 2025, Oracle and/or its affiliates.
```

```
Oracle is a registered trademark of Oracle Corporation and/or its  
affiliates. Other names may be trademarks of their respective  
owners.
```

```
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
```

```
mysql> ■
```

DB 관련 SQL : DBMS 내 생성되어 있는 DB 목록 확인

```
mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| myapp           |
| mysql            |
| performance_schema |
| sys              |
+-----+
5 rows in set (0.00 sec)

mysql>
```

✓ SHOW DATABASES;

- 현재 MySQL 서버에 존재하는 데이터베이스 목록을 확인

✓ 주요 포인트

- mysql, information_schema 등은 **시스템 DB**
- 직접 생성한 DB(testdb 등)도 함께 표시됨
- 사용 중인 유저 권한에 따라 일부 DB는 보이지 않을 수 있음

■ SQL 명령어 구분자 ;

- SQL 문장의 종결(끝) 을 명시
- 여러 SQL 명령어를 순차적으로 실행할 때 명확한 구분 역할

DB 관련 SQL : DB 생성

```
mysql> create database gsc;  
Query OK, 1 row affected (0.02 sec)
```

```
mysql> show databases;  
+-----+  
| Database |  
+-----+  
| gsc      |  
| information_schema |  
| myapp    |  
| mysql    |  
| performance_schema |  
| sys      |  
+-----+  
6 rows in set (0.01 sec)
```

- CREATE DATABASE 생성 DB 이름
 - ✓ 새로운 데이터베이스를 생성
 - ✓ 이미 존재하는 경우 오류 발생

```
mysql> create database if not exists gsc;  
Query OK, 1 row affected, 1 warning (0.00 sec)
```

```
mysql> show warnings;
```

Level	Code	Message
Note	1007	Can't create database 'gsc'; database exists

```
+-----+  
| Level | Code | Message |  
+-----+  
| Note  | 1007 | Can't create database 'gsc'; database exists |  
+-----+  
1 row in set (0.00 sec)
```

- DB가 존재하지 않을 경우만 생성
 - ✓ 이미 존재하는 경우 경고 발생

DB 관련 SQL : DB 삭제

```
mysql> drop database gsc;
Query OK, 0 rows affected (0.04 sec)

mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| myapp |
| mysql |
| performance_schema |
| sys |
+-----+
5 rows in set (0.00 sec)
```

- 기존 데이터베이스를 삭제
- 주의: DB 내의 모든 테이블 및 데이터가 삭제됨

```
mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| myapp |
| mysql |
| performance_schema |
| sys |
+-----+
5 rows in set (0.00 sec)

mysql> drop database if exists gsc;
Query OK, 0 rows affected, 1 warning (0.00 sec)
```

- DB가 존재할 경우 삭제
- ✓ 삭제 DB가 존재하지 않을 경우 경고 발생

DB 관련 SQL : 작업 DB 선택

```
mysql> create database gsc;  
Query OK, 1 row affected (0.01 sec)
```

```
mysql> show databases;
```

Database
gsc
information_schema
myapp
mysql
performance_schema
sys

```
+-----+  
| Database |  
+-----+  
| gsc |  
| information_schema |  
| myapp |  
| mysql |  
| performance_schema |  
| sys |  
+-----+  
6 rows in set (0.00 sec)
```

```
mysql> use gsc;
```

```
Database changed
```

```
mysql> select database();
```

database()
gsc

```
+-----+  
| database() |  
+-----+  
| gsc |  
+-----+  
1 row in set (0.00 sec)
```

```
mysql> █
```

- 작업할 데이터베이스(DB)를 선택하는 명령어
- 이후의 SQL 문은 해당 데이터베이스에서 실행됨

Table 관련 SQL : 특정 DB 내 새로운 Table 생성 (1)

- 테이블 생성

```
mysql> CREATE TABLE student (
    ->     no INT AUTO_INCREMENT PRIMARY KEY,
    ->     std_id CHAR(10) UNIQUE NOT NULL,
    ->     id VARCHAR(30) UNIQUE NOT NULL,
    ->     password VARCHAR(100) NOT NULL,
    ->     name VARCHAR(50) NOT NULL,
    ->     age TINYINT,
    ->     birth DATE
    -> );
```

Query OK, 0 rows affected (0.05 sec)

```
mysql> show tables;
```

```
+-----+
| Tables_in_gsc |
+-----+
| student      |
+-----+
1 row in set (0.00 sec)
```

```
mysql> desc student;
```

```
+-----+
| Field   | Type      | Null | Key | Default | Extra          |
+-----+
| no      | int       | NO   | PRI  | NULL    | auto_increment |
| std_id  | char(10)  | NO   | UNI  | NULL    |                |
| id      | varchar(30)| NO   | UNI  | NULL    |                |
| password| varchar(100)| NO   |       | NULL    |                |
| name    | varchar(50) | NO   |       | NULL    |                |
| age     | tinyint   | YES  |       | NULL    |                |
| birth   | date      | YES  |       | NULL    |                |
+-----+
7 rows in set (0.01 sec)
```

```
mysql> 
```

- student table schema

```
CREATE TABLE student (
    no INT AUTO_INCREMENT PRIMARY KEY,          -- 순번 (자동 증가, 기본키)
    std_id CHAR(10) UNIQUE NOT NULL,            -- 학번 (고유값)
    id VARCHAR(30) UNIQUE NOT NULL,             -- 로그인 ID (고유값)
    password VARCHAR(100) NOT NULL,              -- 비밀번호
    name VARCHAR(50) NOT NULL,                  -- 이름
    age TINYINT,                                -- 나이
    birth DATE                                    -- 생년월일
);
```

필드명	설명	속성
no	레코드 순번	AUTO_INCREMENT, PRIMARY KEY
std_id	학번	UNIQUE, NOT NULL
id	사용자 ID	UNIQUE, NOT NULL
password	비밀번호	해시 저장 고려, NOT NULL
name	이름	NOT NULL
age	나이	TINYINT, 선택사항
birth	생년월일	DATE, 선택사항

<https://dev.mysql.com/doc/refman/8.4/en/data-types.html>

Table 관련 SQL : 특정 DB 내 새로운 Table 생성 (2)

- 자주 사용하는 필드 데이터 타입 <https://dev.mysql.com/doc/refman/8.4/en/data-types.html>

자료형	설명
INT	<ul style="list-style-type: none">• 정수형
VARCHAR(n)	<ul style="list-style-type: none">• 최대 n자까지의 문자열
TEXT	<ul style="list-style-type: none">• 긴 문자열 (게시글 본문 등)
DATE	<ul style="list-style-type: none">• 날짜 (YYYY-MM-DD)
DATETIME	<ul style="list-style-type: none">• 날짜+시간 (YYYY-MM-DD HH:MM:SS)
TIMESTAMP	<ul style="list-style-type: none">• 날짜+시간 + 자동 시간기록 지원
BOOLEAN	<ul style="list-style-type: none">• TINYINT(1)으로 처리됨 (0/1)

자료형	최대 길이
TINYTEXT	255 bytes
TEXT	65,535 bytes
MEDIUMTEXT	16MB
LONGTEXT	4GB

- 주요 필드 제약 조건

제약조건	설명
PRIMARY KEY	<ul style="list-style-type: none">• 테이블의 고유 ID (중복 불가 + NULL 불가)
AUTO_INCREMENT	<ul style="list-style-type: none">• 자동 번호 증가 (보통 PK에 사용)
UNIQUE	<ul style="list-style-type: none">• 중복 값 입력 금지
NOT NULL	<ul style="list-style-type: none">• NULL 입력 불가
DEFAULT	<ul style="list-style-type: none">• 기본값 지정
ON UPDATE	<ul style="list-style-type: none">• 수정 시 값 자동 변경

Table 관련 SQL : 특정 DB-Table 내 Column 수정

- Column 추가

```
mysql> alter table student add email varchar(100);
Query OK, 0 rows affected (0.08 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

```
mysql> desc student;
```

Field	Type	Null	Key	Default	Extra
no	int	NO	PRI	NULL	auto_increment
std_id	char(10)	NO	UNI	NULL	
id	varchar(30)	NO	UNI	NULL	
password	varchar(100)	NO		NULL	
name	varchar(50)	NO		NULL	
age	tinyint	YES		NULL	
birth	date	YES		NULL	
email	varchar(100)	YES		NULL	

- Column 삭제

```
mysql> alter table student drop column email;
Query OK, 0 rows affected (0.05 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

```
mysql> desc student;
```

Field	Type	Null	Key	Default	Extra
no	int	NO	PRI	NULL	auto_increment
std_id	char(10)	NO	UNI	NULL	
id	varchar(30)	NO	UNI	NULL	
password	varchar(100)	NO		NULL	
name	varchar(50)	NO		NULL	
age	tinyint	YES		NULL	
birth	date	YES		NULL	

- Column 이름 + 자료형 변경

```
mysql> alter table student change name full_name varchar(100);
Query OK, 0 rows affected (0.07 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

```
mysql> desc student;
```

Field	Type	Null	Key	Default	Extra
no	int	NO	PRI	NULL	auto_increment
std_id	char(10)	NO	UNI	NULL	
id	varchar(30)	NO	UNI	NULL	
password	varchar(100)	NO		NULL	
full_name	varchar(100)	YES		NULL	
age	tinyint	YES		NULL	
birth	date	YES		NULL	

- Column 자료형 변경

```
mysql> alter table student modify age int;
Query OK, 0 rows affected (0.00 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

```
mysql> desc student;
```

Field	Type	Null	Key	Default	Extra
no	int	NO	PRI	NULL	auto_increment
std_id	char(10)	NO	UNI	NULL	
id	varchar(30)	NO	UNI	NULL	
password	varchar(100)	NO		NULL	
full_name	varchar(100)	YES		NULL	
age	int	YES		NULL	
birth	date	YES		NULL	

Table 관련 SQL : 특정 DB-Table 삭제

- 해당 Table 삭제

```
mysql> show tables;
+-----+
| Tables_in_gsc |
+-----+
| student      |
+-----+
1 row in set (0.01 sec)

mysql> drop table student;
Query OK, 0 rows affected (0.02 sec)

mysql> show tables;
Empty set (0.00 sec)
```

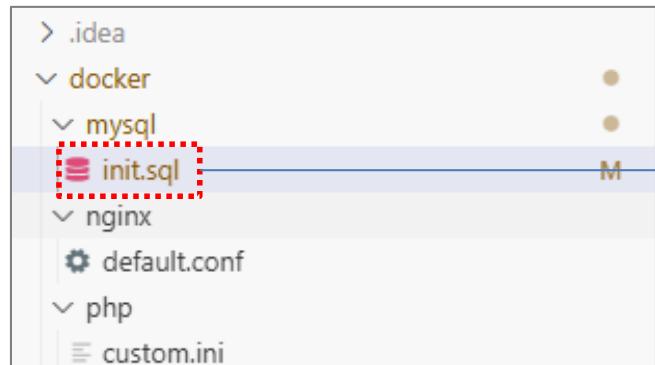
예제 DB & Table 생성 : 스크립트 활용(1)

```
mysql> show databases;  
+-----+  
| Database |  
+-----+  
| gsc |  
| information_schema |  
| mysql |  
| performance_schema |  
| sys |  
+-----+  
5 rows in set (0.00 sec)
```

```
mysql> show tables;  
+-----+  
| Tables_in_gsc |  
+-----+  
| student |  
+-----+  
1 row in set (0.00 sec)
```

```
mysql> desc student;  
+-----+-----+-----+-----+-----+  
| Field | Type | Null | Key | Default | Extra |  
+-----+-----+-----+-----+-----+  
| no | int | NO | PRI | NULL | auto_increment |  
| std_id | varchar(20) | NO | UNI | NULL |  
| id | varchar(20) | NO | UNI | NULL |  
| password | varchar(100) | NO | | NULL |  
| name | varchar(50) | NO | | NULL |  
| age | int | YES | | NULL |  
| birth | date | YES | | NULL |  
+-----+-----+-----+-----+-----+  
7 rows in set (0.00 sec)
```

예제 DB & Table 생성 : 스크립트 활용(2)



```
docker > mysql > init.sql
1  -- 데이터베이스 생성 (존재하지 않으면)
2  CREATE DATABASE IF NOT EXISTS gsc CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci;
3
4  -- gsc 데이터베이스 사용
5  USE gsc;
6
7  -- student 테이블 생성
8  CREATE TABLE IF NOT EXISTS student (
9      no INT AUTO_INCREMENT PRIMARY KEY,
10     std_id VARCHAR(20) NOT NULL UNIQUE,
11     id VARCHAR(20) NOT NULL UNIQUE,
12     password VARCHAR(100) NOT NULL,
13     name VARCHAR(50) NOT NULL,
14     age INT,
15     birth DATE
16  );
-- 순번 (자동 증가)
-- 학번 (유일)
-- 로그인 ID (유일)
-- 비밀번호 (해싱 필요)
-- 이름
-- 나이
-- 생년월일
```

CLI를 이용한 SQL 기본 명령어 실습

Record 삽입, 삭제, 수정, 검색

SQL : 레코드 삽입 – insert into... values

```
INSERT INTO 테이블명 (열1, 열2, ...) VALUES (값1, 값2, ...);
```

- ✓ **테이블명**: 데이터를 삽입할 대상
- ✓ **테이블열(필드)**: 어떤 열에 데이터를 넣을지 지정
- ✓ **값(데이터)**: 각 열에 대응하는 입력 데이터

```
mysql> insert into student (std_id, id, password, name) values('25', 'king', '321', 'pos');  
Query OK, 1 row affected (0.01 sec)
```

```
mysql> select * from student;  
+----+-----+-----+-----+-----+  
| no | std_id | id   | password | name  | age   | birth    |  
+----+-----+-----+-----+-----+  
| 3  | 24     | test | 123    | hong  | 20    | 2002-02-03 |  
| 4  | 25     | king | 321    | pos   | NULL  | NULL     |  
+----+-----+-----+-----+-----+  
2 rows in set (0.00 sec)
```

```
mysql> select * from student;  
Empty set (0.00 sec)
```

- 컬럼명을 생략할 수도 있지만, 명시적으로 쓰는 것을 권장
- 모든 컬럼에 값을 넣는다면 입력 값에 대한 컬럼명 생략 가능 (단, 순서 일치 필수)

```
mysql> insert into student values(null, '24', 'test', '123', 'hong', 20, '2002-02-03');  
Query OK, 1 row affected (0.01 sec)
```

```
mysql> select * from student;  
+----+-----+-----+-----+-----+  
| no | std_id | id   | password | name  | age   | birth    |  
+----+-----+-----+-----+-----+  
| 3  | 24     | test | 123    | hong  | 20    | 2002-02-03 |  
+----+-----+-----+-----+-----+  
1 row in set (0.00 sec)
```

SQL : 레코드 수정 – update ... set ... where ...

UPDATE 테이블명

SET 열1 = 값1, 열2 = 값2, ...

WHERE 조건;

- SET 뒤에 수정할 필드와 값을 지정
- WHERE 조건이 없으면 모든 행이 수정되므로 주의!

```
mysql> select * from student;
```

no	std_id	id	password	name	age	birth
3	24	test	123	hong	20	2002-02-03
4	25	king	321	pos	NULL	NULL

2 rows in set (0.00 sec)

```
mysql> update student set id='queen', password='100' where id='test';
```

Query OK, 1 row affected (0.01 sec)

Rows matched: 1 Changed: 1 Warnings: 0

```
mysql> select * from student;
```

no	std_id	id	password	name	age	birth
3	24	queen	100	hong	20	2002-02-03
4	25	king	321	pos	NULL	NULL

2 rows in set (0.00 sec)

SQL : 레코드 검색 – select ... from ... where ...

```
SELECT 열1, 열2, ... FROM 테이블명 WHERE 조건;
```

- *를 사용하면 모든 컬럼 선택: SELECT * FROM 테이블명;

```
mysql> select * from student;
```

```
+-----+-----+-----+-----+-----+-----+
| no   | std_id | id    | password | name   | age    | birth
+-----+-----+-----+-----+-----+-----+
| 3    | 24     | queen | 100     | hong   | 20     | 2002-02-03 |
| 4    | 25     | king  | 321     | pos    | NULL   | NULL
+-----+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)
```

```
mysql> select id, password from student where age > 10;
```

```
+-----+
| id   | password |
+-----+
| queen | 100    |
+-----+
1 row in set (0.00 sec)
```

- 검색어 포함

```
SELECT * FROM student  
WHERE username LIKE '%영%';
```

- 범위 조건

```
SELECT * FROM student  
WHERE age BETWEEN 20 AND 25;
```

- 결과 개수 제한

```
SELECT * FROM student  
LIMIT 5;
```

- 정렬

```
SELECT * FROM student  
ORDER BY age DESC;
```

➤ 오름차순: ASC

SQL : 레코드 삭제 – **delete from... where ...**

```
DELETE FROM 테이블명 WHERE 조건;
```

- WHERE 절을 꼭 사용. 안 쓰면 모든 레코드가 삭제!

```
mysql> select * from student;
+----+-----+-----+-----+-----+-----+
| no | std_id | id    | password | name   | age   | birth  |
+----+-----+-----+-----+-----+-----+
| 3  | 24    | queen | 100     | hong   | 20    | 2002-02-03 |
| 4  | 25    | king  | 321     | pos    | NULL  | NULL   |
+----+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)
```

```
mysql> delete from student where id='queen';
Query OK, 1 row affected (0.02 sec)
```

```
mysql> select * from student;
+----+-----+-----+-----+-----+-----+
| no | std_id | id    | password | name   | age   | birth  |
+----+-----+-----+-----+-----+-----+
| 4  | 25    | king  | 321     | pos    | NULL  | NULL   |
+----+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

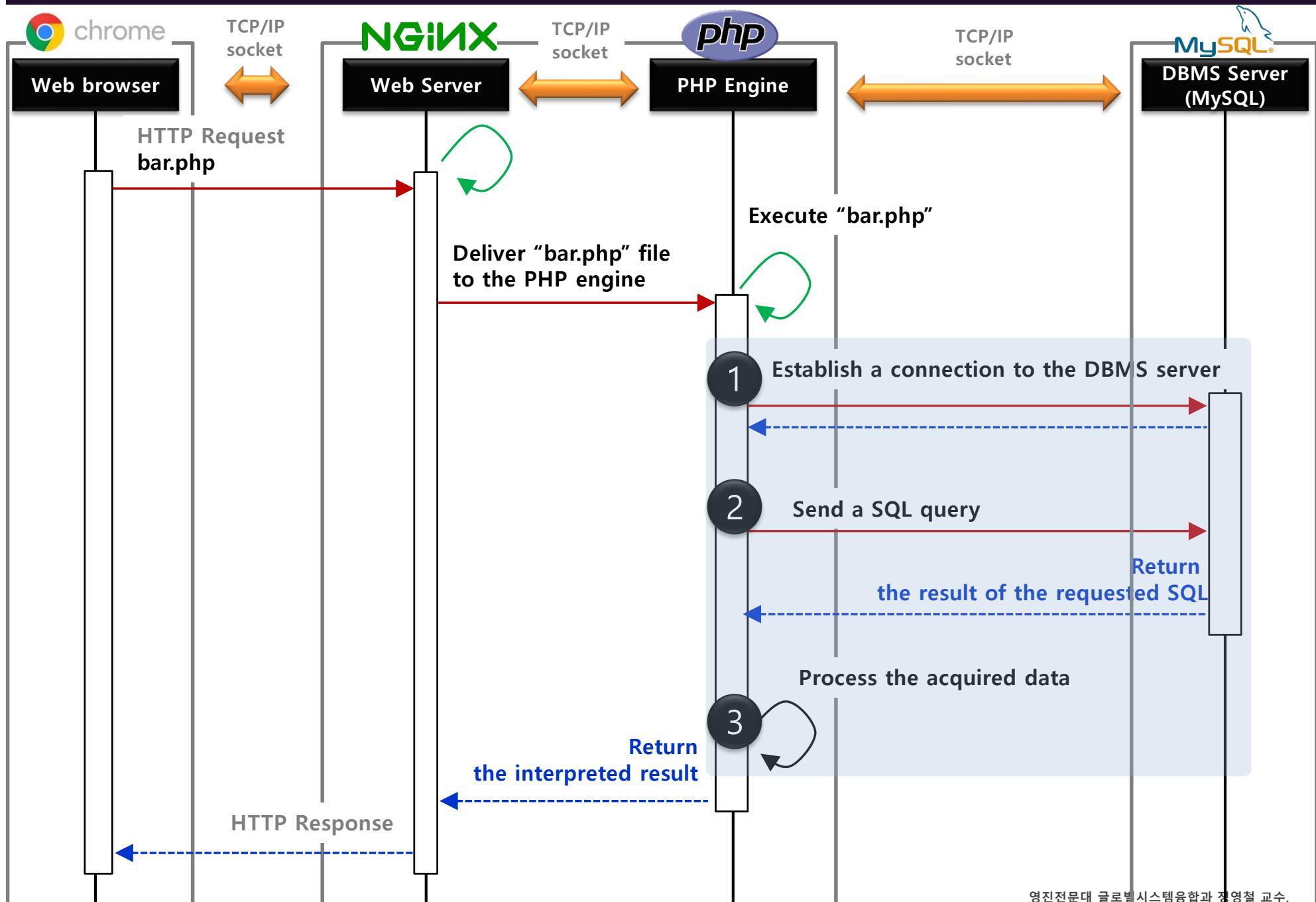
```
mysql> delete from student;
Query OK, 1 row affected (0.01 sec)
```

```
mysql> select * from student;
Empty set (0.00 sec)
```

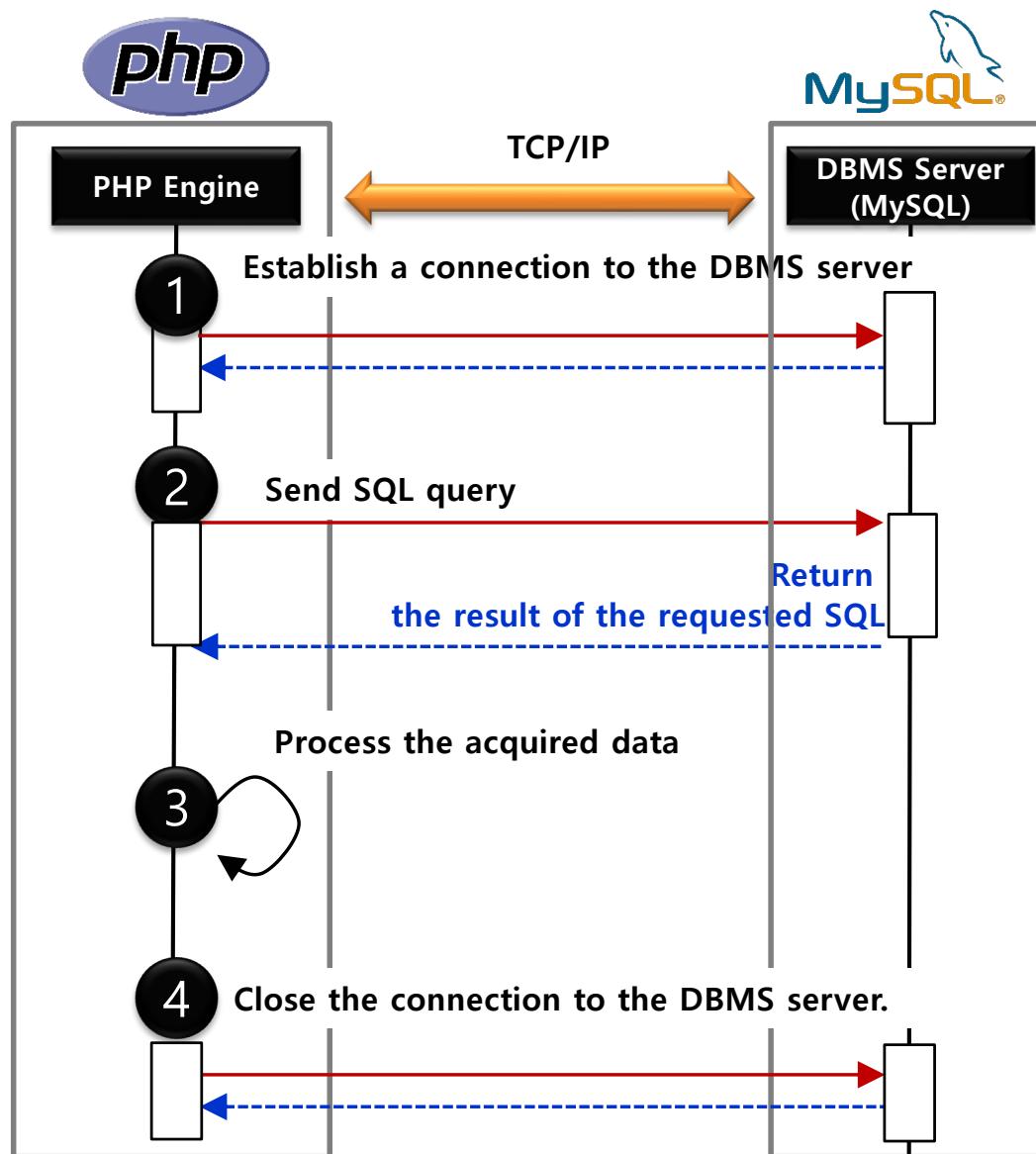
```
mysql> █
```

Web Application에서 DBMS와의 작업 절차

Processing Flow Between PHP Engine and DBMS (1)



Processing Flow Between PHP Engine and DBMS (2)

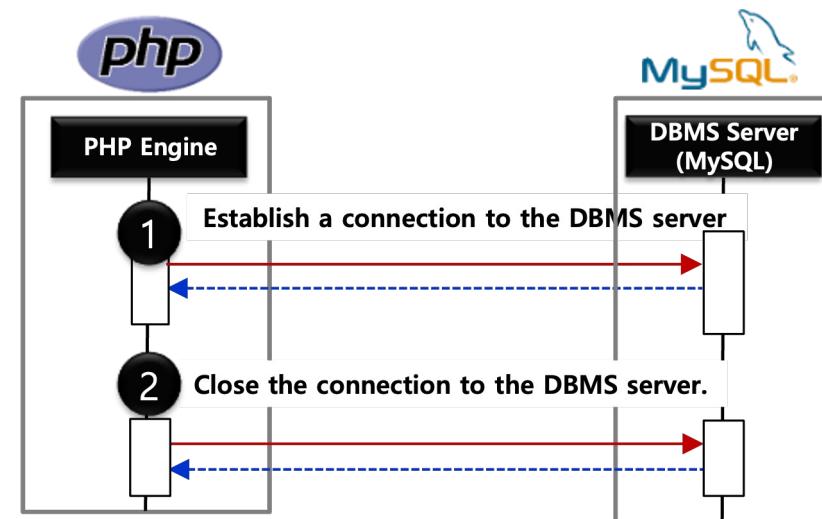


- 1 `$conn = new mysqli()`
- 2 `$conn->query()`
- 3 `$mysqli_result:`
 - `fetch_assoc()`
 - `fetch_array()`
 - `fetch_object()`
- 4 `$conn->close()`

MySQLi : DBMS 접속

✓ db_conf.php

```
<?php
class db_info {
    const DB_URL = 'db';
    const USER_ID = 'root';
    const PASSWD = 'root';
    const DB = 'gsc';
}
```



✓ db_connection_exam.php

```
<?php
require_once('./db_conf.php');

$db_conn = new mysqli(db_info::DB_URL, db_info::USER_ID, db_info::PASSWD, db_info::DB);

if($db_conn->connect_errno) {
    echo "DB 연결 실패";
    exit(-1);
}

echo "DB 연결 성공";
$db_conn->close();
```

1

2

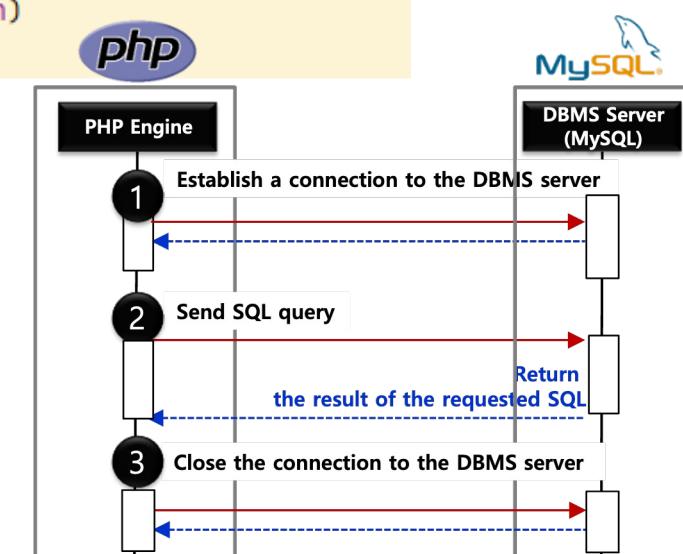
Mysql DBMS 서버 접속
- 접속 성공 시 : 객체 반환
- 접속 실패 시 : false

Error code 리턴
- Error가 없을 경우 0 리턴
- 여러 메시지 종류는 “docs/mysqld_error.txt” 참고

- DBMS connection 종료
- php 페이지 종료 시 GC에 의해 자동으로 종료
- close() 호출 시 코드 실행 중 자원 즉시 반납

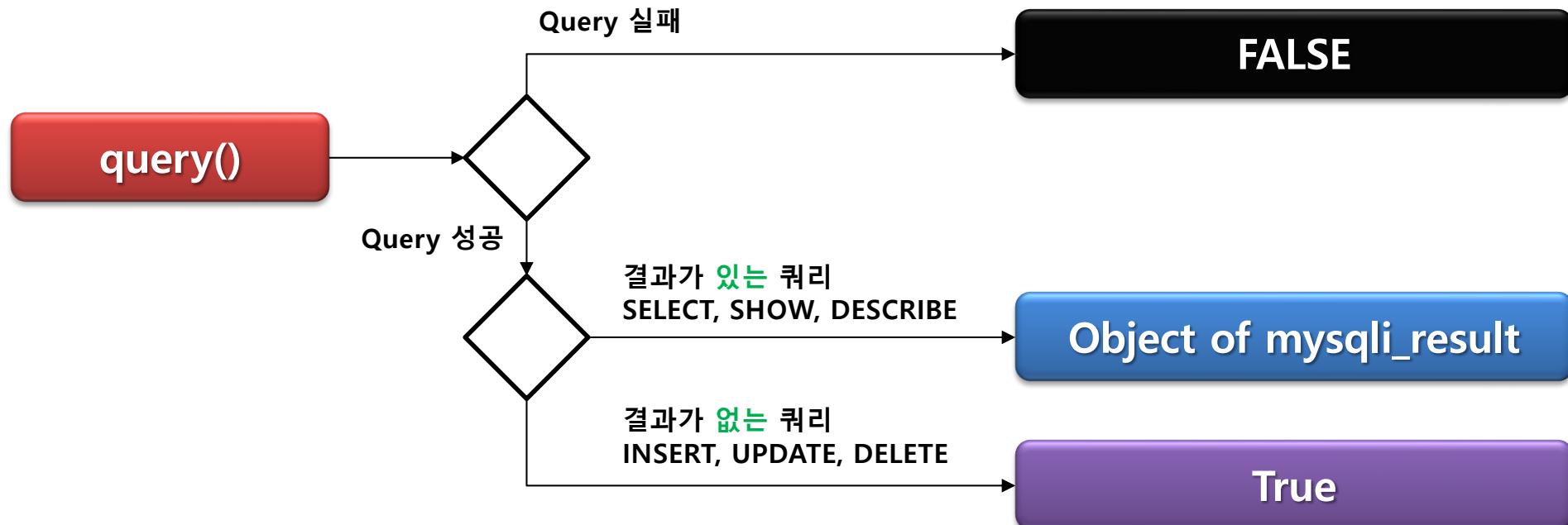
MySQLi : SQL 전송

```
1  <?php
2  // DB 설정 포함
3  require_once('./db_conf.php');
4
5  // MySQL 연결 시도
6  $db_conn = new mysqli(db_info::DB_URL, db_info::USER_ID, db_info::PASSWD, db_info::DB);
7
8  // 연결 오류 처리
9  if ($db_conn->connect_errno) {
10      echo "DB 연결 실패: " . $db_conn->connect_error;
11      exit;
12 }
13
14 // SQL 직접 문자열로 작성 (실습용)
15 $query = "
16     INSERT INTO student (std_id, id, password, name, age, birth)
17     VALUES ('25001', 'test1', '1234', 'Kim', 20, '2002-02-20')
18 ";
19
20 // 쿼리 실행
21 if ($db_conn->query($query) === TRUE) {
22     echo "레코드 추가 성공";
23 } else {
24     echo "레코드 추가 실패: " . $db_conn->error;
25 }
26
27 // 연결 종료
28 $db_conn->close();
```



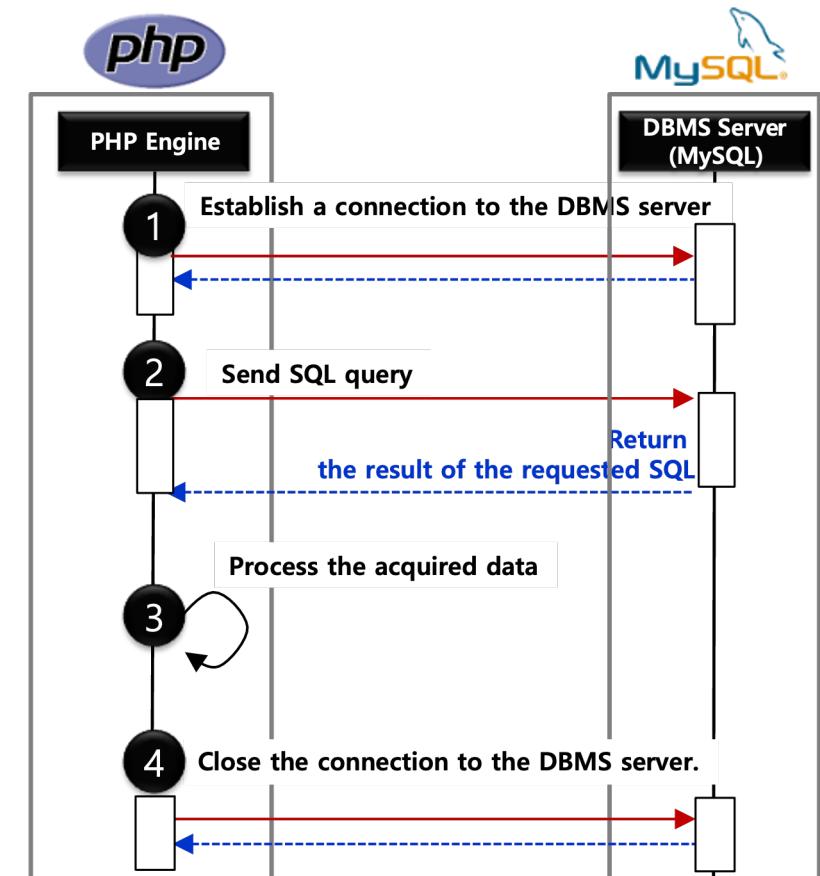
MySQLi : SQL 결과 값 처리 (1)

SQL 구문 종류	반환값	설명
SELECT, SHOW, DESCRIBE 등 <u>결과가 있는 쿼리</u>	mysql_result 객체	<ul style="list-style-type: none">• 결과 행들의 집합
INSERT, UPDATE, DELETE 등 <u>결과가 없는 쿼리</u>	true	<ul style="list-style-type: none">• 쿼리 성공 시
모든 종류	false	<ul style="list-style-type: none">• 쿼리 실패 시 (SQL 오류 등)



MySQLi : SQL 결과 값 처리 (2)

```
<?php  
// DB 설정 포함  
require_once('./db_conf.php');  
  
// MySQL 연결 시도  
$db_conn = new mysqli(db_info::DB_URL, db_info::USER_ID, db_info::PASSWD, db_info::DB); 1  
  
// 연결 오류 처리  
if ($db_conn->connect_errno) {  
    echo "X DB 연결 실패: " . $db_conn->connect_error;  
    exit;  
}  
  
// SQL 문자열 작성  
$query = "SELECT * FROM student";  
  
// 쿼리 실행  
$result = $db_conn->query($query); 2  
  
if (!$result) {  
    echo "쿼리 실행 실패: " . $db_conn->error;  
    $db_conn->close();  
    exit;  
}
```

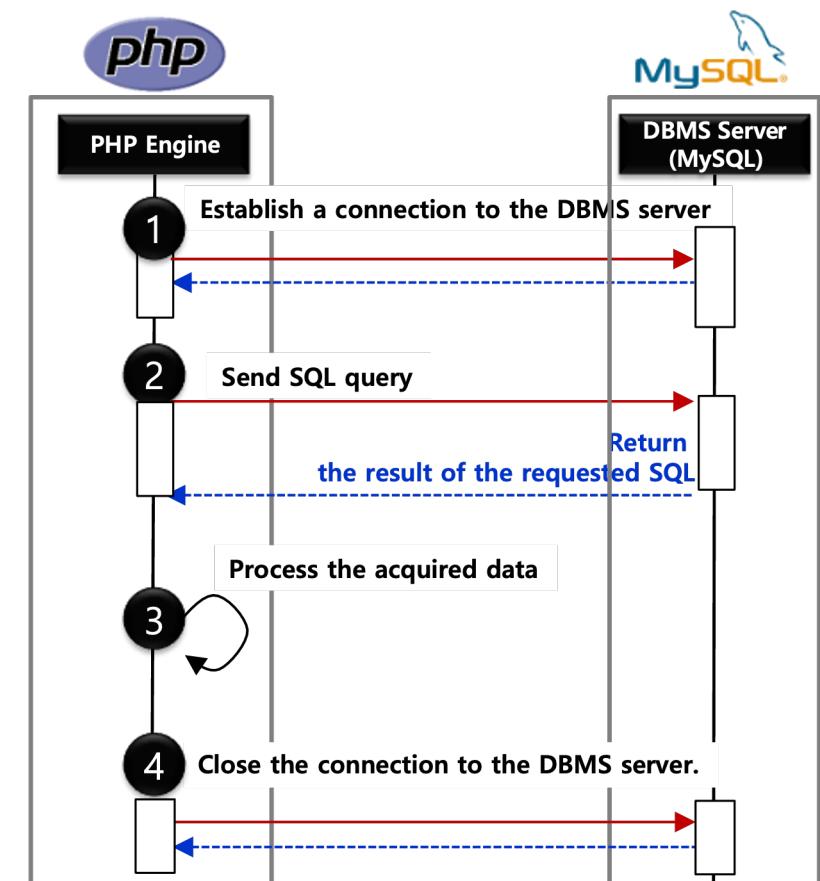


MySQLi : SQL 결과 값 처리 (3)

```
// 결과 출력  
while ($record = $result->fetch_assoc()) {  
    foreach ($record as $key => $value) {  
        echo htmlspecialchars($key) . " => " . htmlspecialchars($value) . "<br>";  
    }  
    echo "<hr>";  
}  
  
// 리소스 정리  
$result->free(); // 결과셋 메모리 해제  
$db_conn->close(); // DB 연결 종료
```

3

4



MySQLi : SQL 결과 값 처리 (4)

```
mysqli_result implements Traversable {  
  
    /* Properties */  
    int $current_field; // 현 필드의 인덱스 값 유지, mysqli_fetch_field() 호출 시 반환  
    int $field_count; // 특정 SQL Query 결과 값 레코드에 대한 필드 개수  
    array $lengths; // 특정 SQL Query 결과 값 레코드의 필드 사이즈  
    int $num_rows; // 특정 SQL Query 결과 값 레코드의 개수  
  
    /* Methods */  
    bool data_seek ( int $offset ) // 특정 SQL Query 결과 값 레코드 Random access 접근, offset 이용  
    mixed fetch_all ([ int $resulttype = MYSQLI_NUM ]) // 특정 SQL Query 결과 2차원 배열로 반환  
    mixed fetch_array ([ int $resulttype = MYSQLI_BOTH ]) // 특정 SQL Query 결과 값 레코드를 정수 또는 연관 배열 값으로 반환  
    array fetch_assoc ( void ) // 특정 SQL Query 결과 값 레코드를 연관 배열 값으로 반환  
    object fetch_field_direct ( int $fieldnr )  
    object fetch_field ( void )  
    array fetch_fields ( void )  
    object fetch_object ([ string $class_name [, array $params ]] ) // 특정 SQL Query 결과 값 레코드를 객체 값으로 반환  
    mixed fetch_row ( void ) // 특정 SQL Query 결과 값 레코드를 Numerical 배열 값으로 반환  
    bool field_seek ( int $fieldnr )  
    void free ( void )  
}
```

MySQLi: mysqli_fetch_XXX() (1)

- mysqli_fetch_XXX() 계열 함수는 SELECT 쿼리 실행 후 반환된 결과 집합(result set)에서 행(row/record)을 하나씩 가져오는 함수
- XXX 자리에 들어가는 값에 따라 결과를 어떤 형식으로 반환할지 달라진다

함수명	반환 형식	설명
mysqli_fetch_row()	숫자 인덱스 배열	<ul style="list-style-type: none">• 인덱스로 접근: \$row[0], \$row[1]
mysqli_fetch_assoc()	연관 배열	<ul style="list-style-type: none">• 컬럼명을 키로 접근: \$row['name']
mysqli_fetch_array()	숫자+연관 배열	<ul style="list-style-type: none">• 둘 다 가능 (MYSQLI_ASSOC, MYSQLI_NUM 옵션 지정 가능)
mysqli_fetch_object()	객체 형식	<ul style="list-style-type: none">• \$row->name 식으로 접근
mysqli_fetch_all()	다차원 배열	<ul style="list-style-type: none">• 모든 행을 한 번에 배열로 반환 (PHP 5.3+)

MySQLi: mysqli_fetch_XXX() (2)

```
$result = $conn->query("SELECT id, name FROM student");

$row1 = mysqli_fetch_row($result);
// $row1[0], $row1[1]

$row2 = mysqli_fetch_assoc($result);
// $row2['id'], $row2['name']

$row3 = mysqli_fetch_array($result);
// $row3[0], $row3['id']

$row4 = mysqli_fetch_object($result);
// $row4->id, $row4->name
```

함수	반환 방식	특징
✓ fetch_row()	숫자 인덱스 배열	빠르고 단순
✓ fetch_assoc()	연관 배열	컬럼명으로 접근, 실무에서 가장 많이 사용
✓ fetch_array()	둘 다 포함	혼용 가능하지만 메모리 낭비 가능
✓ fetch_object()	객체 형태	OOP 방식에 적합

Mysqli fetch_XXX : **fetch_row()**

```
1 <?php
2 require_once('./db_conf.php');
3
4 $db_conn = new mysqli(db_info::DB_URL, db_info::USER_ID, db_info::PASSWD, db_info::DB);
5
6 if ($db_conn->connect_errno) {
7     echo "DB 연결 실패: " . $db_conn->connect_error;
8     exit;
9 }
10
11 $query = "SELECT * FROM student";
12
13 if (!$result = $db_conn->query($query)) {
14     echo "쿼리 오류: " . $db_conn->error;
15     exit;
16 }
17
18 // fetch row 메서드 사용
19 while ($row = $result->fetch_assoc()) {
20     echo "학번: {$row['std_id']}<br>";
21     echo "이름: {$row['name']}<br>";
22     echo "나이: {$row['age']}<br>";
23     echo "생년월일: {$row['birth']}<br>";
24     echo "<hr>";
25 }
26
27 $db_conn->close();
```

Mysqli fetch_XXX : fetch_array()

```
1 <?php
2 // DB 설정 포함
3 require_once('./db_conf.php');
4
5 // MySQL 연결
6 $db_conn = new mysqli(db_info::DB_URL, db_info::USER_ID, db_info::PASSWD, db_info::DB);
7
8 // 연결 오류 확인
9 if ($db_conn->connect_errno) {
10     echo "DB 연결 실패: " . $db_conn->connect_error;
11     exit;
12 }
13
14 // SQL 실행
15 $query = "SELECT * FROM student";
16 $result = $db_conn->query($query);
17
18 if (!$result) {
19     echo "쿼리 오류: " . $db_conn->error;
20     exit;
21 }
22
23 // 결과 출력
24 while ($row = $result->fetch_array()) {
25     // 기본값은 MYSQLI_BOTH: 숫자 인덱스 + 연관배열 둘 다 사용 가능
26     echo "이름(인덱스 방식): " . $row[4] . "<br>"; // name 필드 (5번째 컬럼, 0부터 시작)
27     echo "이름(열렬명 방식): " . $row['name'] . "<br>"; // name 필드 (연관배열 방식)
28     echo "나이: " . $row['age'] . "<br>";
29     echo "<hr>";
30 }
31
32 // 연결 종료
33 $db_conn->close();
```

```
$row = mysqli_fetch_array($result, MYSQLI_ASSOC); // 연관배열만 사용
```

옵션 상수	설명
MYSQLI_ASSOC	연관배열만 리턴 (\$row['name'])
MYSQLI_NUM	숫자 인덱스 배열만 리턴 (\$row[4])
MYSQLI_BOTH	기본값, 둘 다 포함됨

Mysqli fetch_XXX : **fetch_assoc()**

```
1  <?php
2  // DB 설정 포함
3  require_once('./db_conf.php');
4
5  // MySQL 연결
6  $db_conn = new mysqli(db_info::DB_URL, db_info::USER_ID, db_info::PASSWD, db_info::DB);
7
8  // 연결 오류 확인
9  if ($db_conn->connect_errno) {
10    echo "DB 연결 실패: " . $db_conn->connect_error;
11    exit;
12 }
13
14 // SQL 실행
15 $query = "SELECT * FROM student";
16 $result = $db_conn->query($query);
17
18 // 쿼리 실행 오류 확인
19 if (!$result) {
20   echo "쿼리 오류: " . $db_conn->error;
21   exit;
22 }
23
24 // 결과 출력
25 while ($row = $result->fetch_assoc()) {
26   echo "학번: " . htmlspecialchars($row['std_id']) . "<br>";
27   echo "이름: " . htmlspecialchars($row['name']) . "<br>";
28   echo "나이: " . htmlspecialchars($row['age']) . "<br>";
29   echo "생년월일: " . htmlspecialchars($row['birth']) . "<br>";
30   echo "<hr>";
31 }
32
33 // 연결 종료
34 $db_conn->close();
```

Mysqli fetch_XXX : **fetch_object()**

```
1 <?php
2 // DB 설정 포함
3 require_once('../db_conf.php');
4
5 // MySQL 연결
6 $db_conn = new mysqli(db_info::DB_URL, db_info::USER_ID, db_info::PASSWD, db_info::DB);
7
8 // 연결 오류 확인
9 if ($db_conn->connect_errno) {
10     echo "DB 연결 실패: " . $db_conn->connect_error;
11     exit;
12 }
13
14 // SQL 실행
15 $query = "SELECT * FROM student";
16 $result = $db_conn->query($query);
17
18 // 쿼리 실행 오류 확인
19 if (!$result) {
20     echo "쿼리 오류: " . $db_conn->error;
21     exit;
22 }
23
24 // 결과 출력
25 while ($row = $result->fetch_object()) {
26     echo "학번: " . htmlspecialchars($row->std_id) . "<br>";
27     echo "이름: " . htmlspecialchars($row->name) . "<br>";
28     echo "나이: " . htmlspecialchars($row->age) . "<br>";
29     echo "생년월일: " . htmlspecialchars($row->birth) . "<br>";
30     echo "<hr>";
31 }
32
33 // 연결 종료
34 $db_conn->close();
```

Myqli fetch_field 시리즈 (1)

- SQL 결과셋의 각 필드(컬럼)의 구조 정보를 가져오는 함수
- mysqli_result 객체에서 컬럼 이름, 데이터형, 길이 등을 하나씩 추출함

```
1 <?php
2 require_once('./db_conf.php');
3 $db_conn = new mysqli(db_info::DB_URL, db_info::USER_ID, db_info::PASSWD, db_info::DB);
4
5 $query = "SELECT * FROM student";
6 $result = $db_conn->query($query);
7
8 echo "<h3>필드 정보 출력</h3>";
9 while ($field = $result->fetch_field()) {
10     echo "컬럼명: " . $field->name . "<br>";
11     echo "테이블명: " . $field->table . "<br>";
12     echo "자료형(type): " . $field->type . "<br>";
13     echo "길이(length): " . $field->length . "<br>";
14     echo "-----<br>";
15 }
16
17 $db_conn->close();
```

Myqli fetch_field 시리즈 (2)

Object properties

Property	Description
name	The name of the column
orgname	Original column name if an alias was specified
table	The name of the table this field belongs to (if not calculated)
orgtable	Original table name if an alias was specified
def	Reserved for default value, currently always ""
db	Database (since PHP 5.3.6)
catalog	The catalog name, always "def" (since PHP 5.3.6)
max_length	The maximum width of the field for the result set.
length	The width of the field, as specified in the table definition.
charsetnr	The character set number for the field.
flags	An integer representing the bit-flags for the field.
type	The data type used for this field
decimals	The number of decimals used (for integer fields)

DBMS Error Handling

MySQL Error Number

- 에러 번호는 DBMS(MySQL 서버)가 쿼리 실행 중 문제가 발생했을 때 반환하는 정수형 코드
- 에러 번호는 오류의 종류와 원인을 식별하기 위해 사용
 - ✓ 예) 1062는 “중복된 키 값 입력(Duplicate entry)”을 의미
- MySQL 에러는 1000번대는 SQL 계층, 2000번대는 클라이언트/연결 오류를 주로 의미

```
$conn = new mysqli(...);
$conn->query("INVALID SQL");

if ($conn->errno) {
    echo "에러 번호: " . $conn->errno;
}
```

에러 코드	의미	대표 메시지	발생 상황
1062	중복 키 오류	Duplicate entry	UNIQUE 또는 PRIMARY KEY 필드에 중복된 값 삽입 시
1049	존재하지 않는 DB	Unknown database	지정한 데이터베이스 이름이 없음
1146	테이블 없음	Table doesn't exist	존재하지 않는 테이블을 쿼리할 때
1045	로그인 인증 실패	Access denied for user	사용자명 또는 비밀번호 오류, 권한 부족
2002	서버 연결 실패	Can't connect to local MySQL server	소켓/포트 문제
1366	잘못된 문자셋	Incorrect string value	DB 컬럼의 문자셋과 입력 데이터 불일치
1054	컬럼 없음	Unknown column	잘못된 컬럼명을 SELECT, WHERE 등에 사용할 때
1064	SQL 문법 오류	You have an error in your SQL syntax	SQL 구문 오류

• 에러 번호 확인 방법

1. MySQL 공식 문서 확인: <https://dev.mysql.com/doc/mysql-errors/8.0/en/>
2. CLI에서 발생한 에러 확인 시 사용

```
mysql> INSERT ...;
ERROR 1062 (23000): Duplicate entry ...
```

DBMS 사용 시 에러/예외 처리 옵션 설정

이유	설명
보안상 이유	에러 메시지가 그대로 사용자에게 노출되면 SQL 구조, 테이블명 등 내부 정보 유출 위험이 있음. 이를 막기 위해 예외 처리로 에러 메시지를 제어해야 함
예외 상황 제어	DB 연결 실패, 쿼리 실패 등 다양한 예외 상황에 대해 정확하고 예측 가능한 흐름 제어가 필요함 (try-catch)
디버깅/로그 기록	에러 발생 시 로깅 시스템에 기록하고, 사용자에겐 명확한 알림 메시지를 보여줄 수 있음
유지보수 효율성	if (!\$result) 같은 구조보다 try-catch 기반이 예외 흐름을 명확하게 표현하고 구조화하기 좋음.

• mysqli_report() 함수

- MySQLi 함수(예: query(), prepare() 등)에서 발생하는 에러, 경고, 인덱스 문제 등을 어떻게 보고할지
를 설정하는 함수
- MySQLi 확장에서 내부 오류가 발생했을 때,
 - ① 경고만 출력할지
 - ② 예외를 던질지
 - ③ 아예 무시할지 등을 제어할 수 있다

```
mysqli_report(int $flags): bool
```

DBMS 사용 시 에러/예외 처리 옵션 설정 (1)

설정값	예외 발생 여부 (Exception)	에러/예외 출력	특징
MYSQLI_REPORT_OFF	✗ 발생하지 않음	✗ 출력하지 않음	에러는 false만 반환. 조용히 실패, 수동 체크 필요
MYSQLI_REPORT_ERROR	✗ 발생하지 않음	✓ 경고(warning) 출력	PHP 경고로 콘솔 또는 브라우저에 표시됨
MYSQLI_REPORT_STRICT	✓ 예외 발생	✗ 출력하지 않음	try-catch로 예외 처리 필요
MYSQLI_REPORT_ALL	✓ 예외 발생	✓ 경고(warning) 출력	

- mysqli_report()를 설정하더라도 일부 예외에서 무시될 수 있음 (객체 생성 중 발생 오류 등)
 - ✓ 예) new mysqli() 실행 시 객체 생성자 내부 경고는 mysqli_report() 이전 단계에서 발생
 - PHP의 객체 생성 과정 자체에서 이미 warning을 발생시켜버리기 때문에, mysqli_report(MYSQLI_REPORT_OFF)로도 완전히 제어되지 않을 수 있음

➡ 해결 방법 1 : @ 연산자 사용

```
mysqli_report(MYSQLI_REPORT_OFF);
@$db_conn = new mysqli('localhost', 'root', 'wrong_pw', 'nonexistent_db');
```

➡ 해결 방법 2 : PHP 전체 경고 레벨 억제 (error_reporting)

```
error_reporting(0);
ini_set('display_errors', '0');
mysqli_report(MYSQLI_REPORT_OFF);
$db_conn = new mysqli('localhost', 'root', 'wrong_pw', 'nonexistent_db');
```

DBMS 사용 시 예외/예외 처리 옵션 설정 (2)

- `mysqli_report()`는 MySQLi의 객체지향(OOP) 방식에서 사용할 때 **객체 메서드를 사용할 경우에만** 제대로 동작
- MySQLi는 절차적 함수 방식과 객체지향 방식 두 가지 API를 모두 제공하지만, 이 둘을 혼용해서 사용할 경우 `mysqli_report()` 설정이 의도한 대로 동작하지 않을 수 있다.
- 따라서 예외 처리를 위한 `MYSQLI_REPORT_STRICT` 설정을 사용하는 경우에는 **객체지향 방식으로 일관되게 작성하는 것이 권장**

```
// 예외 발생 시 예외 객체로 처리
mysqli_report( flags: MYSQLI_REPORT_STRICT);

try {
    // 객체 지향 방식으로 연결
    $db_conn = new mysqli( hostname: 'db' , username: 'root' , password: 'root' , database: 'gsc');

    if ($db_conn->connect_errno) {
        echo "Failed to connect to MySQL: " . $db_conn->connect_error;
        exit();
    }

    echo "Connected successfully<br>";

    // ❌ 절차적 방식 사용 (혼용)
    // 객체 지향 방식 내에서 함수형 API를 사용하면 mysqli_report 설정이 무시될 수 있음
    $result = mysqli_query($db_conn, query: "SELECT * FROM `students`"); // 존재하지 않는 테이블

    echo "이 줄은 실행되지 않아야 하지만, procedural 방식이라 예외가 발생하지 않음";

} catch (mysqli_sql_exception $e) {
    echo "예외 발생: " . $e->getMessage();
} finally {
    $db_conn->close();
}
```

DBMS 사용 시 에러/예외 처리 예제 (1)

```
1 <?php
2 // 데이터베이스 설정 정보 포함 (db_info 클래스: URL, USER_ID, PASSWD 정의)
3 require_once('./db_conf.php');
4
5 // MySQLi 에러 처리 방식 설정
6 // MYSQLI_REPORT_STRICT : 예외 발생 (mysqli_sql_exception)
7 // MYSQLI_REPORT_ERROR : 경고(warning) 출력
8 // MYSQLI_REPORT_OFF : 오류 보고 없음 (false 반환)
9 // MYSQLI_REPORT_ALL : STRICT + ERROR 조합
10 mysqli_report(MYSQLI_REPORT_STRICT);
11
12 $db_conn = null;
13
14 try {
15     // 데이터베이스 연결 시도
16     $db_conn = new mysqli(db_info::DB_URL, db_info::USER_ID, db_info::PASSWD, 'gscs');
17
18     // 문자셋 설정 (한글, 이모지 깨짐 방지)
19     $db_conn->set_charset("utf8mb4");
20
21     // SQL 쿼리 실행
22     $query = "SELECT * FROM student";
23     $result = $db_conn->query($query);
24
25     // 결과 출력 (연관 배열 방식)
26     while ($row = $result->fetch_assoc()) {
27         echo "이름: " . htmlspecialchars($row['name']) . "<br>";
28         echo "나이: " . htmlspecialchars($row['age']) . "<br>";
29         echo "<hr>";
30     }
31 }
```

DBMS 사용 시 에러/예외 처리 예제 (2)

```
32 } catch (Exception $ex) {
33     // 내부 서버 로그에 에러 기록
34     error_log("[DB ERROR] " . $ex->getMessage(), 3, "/tmp/php_custom_error.log");
35
36     // 사용자에게는 일반적인 오류 메시지만 출력 (보안 고려)
37     echo "데이터베이스 처리 중 오류가 발생했습니다. 잠시 후 다시 시도해 주세요.";
38 } finally {
39     // DB 연결 종료
40     if ($db_conn instanceof mysqli) {
41         $db_conn->close(); // 안전하게 연결 종료
42     }
43 }
```

Extra Slides

백엔드 개발에서 DBMS를 관리하는 주요 방식 (1)

✓ 텍스트 기반 SQL (Text Protocol) — 실무에서는 사용하지 않음

- 전통적인 **SQL 쿼리를 텍스트로 작성하여 DBMS에 전송**
- 가장 널리 쓰이며, 개발자가 직접 읽고 쓸 수 있음
- 예: **SELECT * FROM users WHERE id = 1;**

✓ 가장 기본적 방식

✓ 단점: 보안 취약(SQL Injection), 파싱 비용 존재

✓ 바이너리 프로토콜 (Prepared Statement 등) — 성능 & 보안 개선

- SQL을 미리 컴파일한 후 파라미터만 전달 (바이너리 형식)
- API 레벨에서 prepare → bind → execute 구조로 실행
- 예: **PDO::prepare(), mysqli_prepare()**

✓ 장점: 빠름, SQL Injection 방지

✓ 특히 반복 쿼리(루프 내 insert 등)에 유리

백엔드 개발에서 DBMS를 관리하는 주요 방식 (2)

✓ ORM (Object-Relational Mapping) 사용

- SQL을 직접 쓰지 않고, 객체지향적으로 DB를 다룬다
- 예: Laravel Eloquent (PHP), Django ORM (Python), Sequelize (Node.js)

```
// SQL 직접 작성 없이도 내부적으로 쿼리가 생성되어 전송됨  
User::where('id', 1)->first();
```

- ✓ 개발 속도 향상, 유지보수 용이
- ✓ 내부적으로는 결국 SQL(보통 Prepared Statement)을 전송

✓ 쿼리 빌더 (Query Builder) 사용

- ORM보다 조금 더 SQL에 가까운 방식으로 쿼리 구성
- SQL Injection 방지 및 가독성 확보

```
DB::table('users')->where('id', 1)->get();
```

- ✓ 코드로 SQL을 안전하게 구성
- ✓ 내부적으로 Prepared Statement로 실행되는 경우 많음

방식	설명	내부 전송 방식
• 직접 SQL (텍스트 프로토콜)	전통적인 SQL 문자열	텍스트 전송
• Prepared Statement	SQL 미리 컴파일, 파라미터 분리	바이너리 + 텍스트 혼합
• ORM	객체 기반 DB 처리	내부적으로 SQL 사용
• Query Builder	코드 기반 SQL 구성	Prepared Statement 사용

SQL 대소문자 구분 설정

항목	대소문자 구분 여부	설명
✓ SQL 키워드	✗ 구분하지 않음	<ul style="list-style-type: none">• SELECT, from, WHERE 등
✓ 테이블/컬럼 이름	✓ 또는 ✗ (OS 및 설정 따라)	<ul style="list-style-type: none">• Linux는 구분• Windows는 기본적으로 구분하지 않음
✓ 문자열 비교	✓ 또는 ✗ (collation 설정에 따라)	<ul style="list-style-type: none">• utf8mb4_general_ci는 구분 안 함• utf8mb4_bin은 구분함
➤ 문자열 비교		

```
SELECT * FROM users WHERE name = 'Jung';
```

이때 'Jung'과 'jung'이 같은 것으로 취급될지 여부는 컬럼의 collation 설정에 따라 다름

Collation 예시	대소문자 구분?	설명
utf8mb4_general_ci	✗ 구분하지 않음	ci = case-insensitive
utf8mb4_bin	✓ 구분함	bin = binary 비교

MySQL 날자&시간 관련 필드 자료형

자료형	저장 범위	타임존 변환	주 사용 목적
DATE	1000-01-01 ~ 9999-12-31	X	날짜만 저장
TIME	-838:59:59 ~ 838:59:59	X	시간만 저장
DATETIME	1000-01-01 00:00:00 ~ 9999-12-31 23:59:59	X	날짜+시간 저장
TIMESTAMP	1970-01-01 ~ 2038-01-19 (UTC 기준)	<input checked="" type="checkbox"/> (자동 변환)	날짜+시간 저장 Unix timestamp 사용

SQL Literal 예시

리터럴 종류	예시	설명
문자열	'admin', '홍길동'	사용자 이름, 코드 등
정수	100, 0, -5	가격, 수량 등
실수	3.14, -0.99	평균, 온도 등
날짜	'2025-07-16', DATE '2025-07-16'	주문일, 생일 등
시간	'15:30:00'	근무시작시간 등
논리	TRUE, FALSE, 1, 0	상태값, 사용여부 등
NULL	NULL	값 없음 표현

SQL 연산자

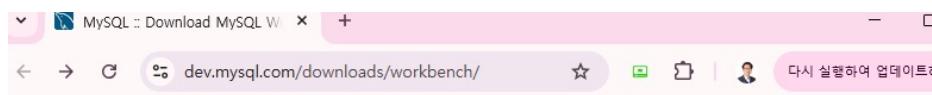
연산자	의미	예시
=	같다	age = 30
!=	같지 않다 (MySQL에서 사용)	name != '홍길동'
<>	같지 않다 (표준 SQL)	score <> 100
>	크다	age > 18
<	작다	price < 1000
>=	크거나 같다	score >= 60
<=	작거나 같다	birth_date <= '2000-01-01'
IS NULL	값이 NULL이다	deleted_at IS NULL
IS NOT NULL	값이 NULL이 아니다	email IS NOT NULL

연산자	의미	예시
AND	그리고 (모두 참)	age >= 20 AND city = 'Seoul'
OR	또는 (하나라도 참)	age < 18 OR age > 65
NOT	부정 (참 → 거짓)	NOT (status = 'active')

MySQL Workbench

- MySQL 데이터베이스를 GUI 기반으로 설계, 관리, 쿼리, 모델링할 수 있는 공식 툴

<https://dev.mysql.com/downloads/workbench/>



④ MySQL Community Downloads

↳ MySQL Workbench

The screenshot shows the MySQL Workbench 8.0.42 download page. At the top, there are tabs for "General Availability (GA) Releases" (which is selected), "Archives", and "Info". Below the tabs, it says "MySQL Workbench 8.0.42". A dropdown menu for "Select Operating System" is open, showing "Microsoft Windows" as the current selection. Under "Recommended Download:", there is a large image of the "MySQL Installer for Windows" package, which is described as containing "All MySQL Products. For All Windows Platforms. In One Package." Below the image, it says "Starting with MySQL 5.6 the MySQL Installer package replaces the standalone MSI packages." There are two download links: "Windows (x86, 32 & 64-bit), MySQL Installer MSI" and "Go to Download Page >". Under "Other Downloads:", there is a table with one row:

Windows (x86, 64-bit), MSI Installer	8.0.42	43.0M	Download
(mysql-workbench-community-8.0.42-winx64.msi)			MD5: b45ed530b2f158f2f00a6bbb646ecc1e Signature

Q/A

감사합니다



주문식교육의 산실
영진전문대학교