

16

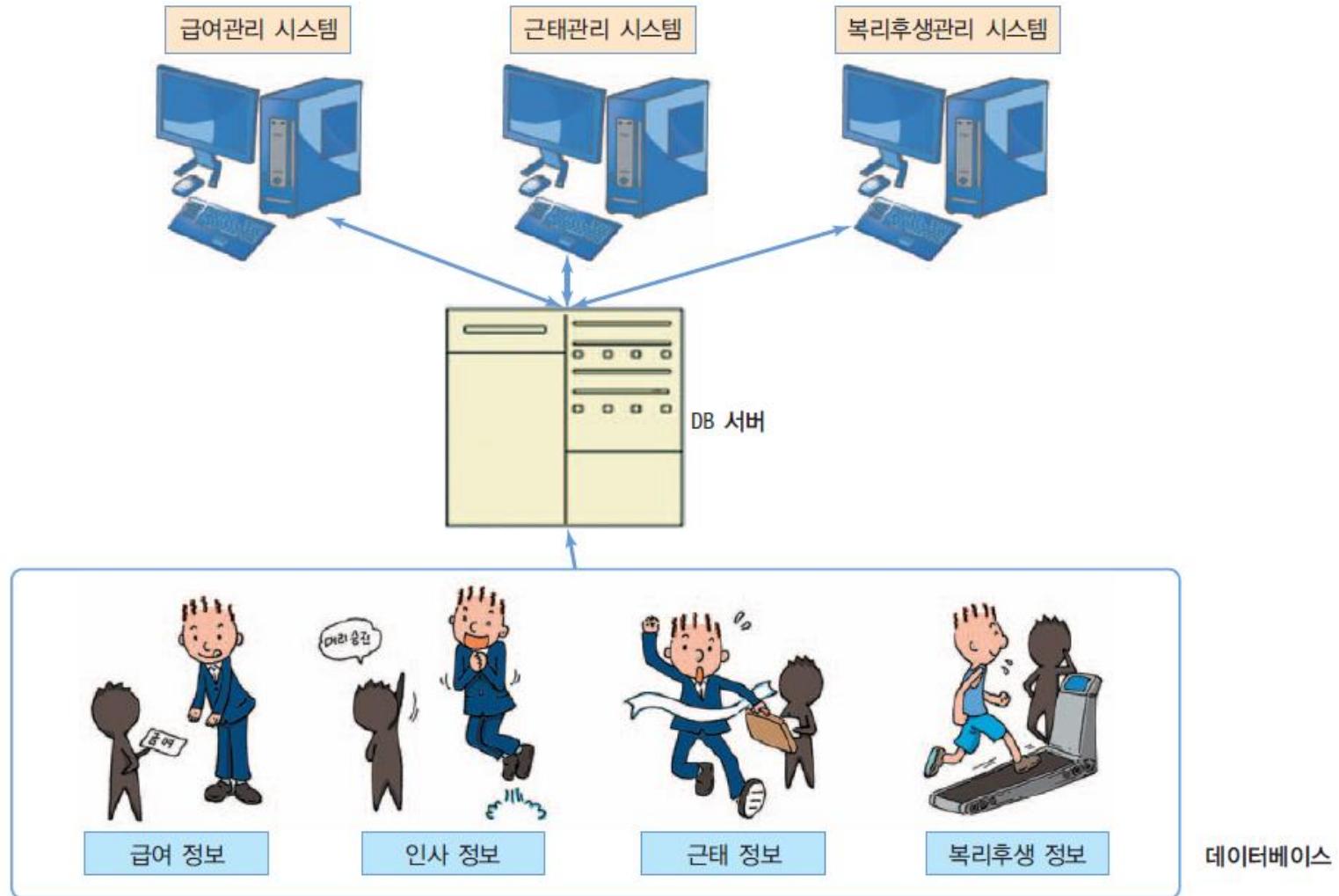
데이터베이스의 개념

2

- 데이터베이스
 - ▣ 여러 응용 시스템들의 통합된 정보들을 저장하여 운영할 수 있는 공용 데이터들의 집합
 - ▣ 데이터의 저장, 검색, 갱신을 효율적으로 수행할 수 있도록 데이터를 고도로 조직화하여 저장
- DBMS
 - ▣ 데이터베이스 관리 시스템(DataBase Management System)
 - 오라클(Oracle), 마이크로소프트의 SQL Server, MySQL, IBM의 DB2 등
- 데이터베이스 종류
 - ▣ 관계형 데이터베이스
 - 키(key)와 값(value)들의 관계를 테이블로 표현한 데이터베이스 모델
 - 키는 테이블의 열(column)이 되며 테이블의 행(row)은 하나의 레코드(record)를 표현
 - 현재 사용되는 대부분의 데이터베이스는 관계형 데이터베이스
 - ▣ 객체 지향 데이터베이스
 - 객체 지향 프로그래밍에 쓰이는 것으로, 정보를 객체의 형태로 표현하는 데이터베이스 모델
 - 오브젝트 데이터베이스(object database)라고도 부른다.

기업 내의 데이터베이스 사례

3



관계형 데이터베이스 구조

4

- 관계형 데이터 베이스
 - 데이터들이 다수의 테이블로 구성
 - 각 행은 하나의 레코드
 - 각 테이블은 키(key)와 값(value) 들의 관계로 표현
 - 여러 테이블 간에 공통된 이름의 열 포함
 - 이런 경우 서로 다른 테이블 간에 관계(relation)가 성립
 - 대부분의 데이터베이스는 관계형 데이터베이스
 - JDBC API

The diagram illustrates the concept of a relationship between two relational databases, Table A and Table B. Table A, labeled '테이블 A' (Table A), contains columns 'employee_id' and 'name'. It has three rows with data: 00001 (김철수), 00002 (최고봉), and 00003 (이기자). The 'employee_id' column is highlighted with a pink border and labeled '키' (Key). The 'name' column is highlighted with a blue border and labeled '값' (Value). A pink arrow points from the text '두 테이블이 동일한 키를 가지고 있음 : 관계' (Two tables have the same key: Relationship) to the 'employee_id' column of Table A. Table B, labeled '테이블 B' (Table B), contains columns 'payroll_id', 'amount', and 'employee_id'. It has three rows with data: 00000001 (1000000, 00002), 00000002 (2000000, 00010), and 00000003 (3000000, 00021). The 'employee_id' column is highlighted with a pink border and labeled '키' (Key). A pink arrow points from the same text '두 테이블이 동일한 키를 가지고 있음 : 관계' to the 'employee_id' column of Table B.

| employee_id | name |
|-------------|------|
| 00001 | 김철수 |
| 00002 | 최고봉 |
| 00003 | 이기자 |

| payroll_id | amount | employee_id |
|------------|---------|-------------|
| 00000001 | 1000000 | 00002 |
| 00000002 | 2000000 | 00010 |
| 00000003 | 3000000 | 00021 |

객체지향 데이터베이스

5

- 객체지향 데이터 베이스
 - ▣ 객체 지향 프로그래밍에 사용
 - ▣ 정보를 객체의 형태로 표현
 - ▣ 오브젝트 데이터베이스(object database)라고도 부름
 - ▣ 객체 모델이 그대로 데이터베이스에도 적용되므로 응용프로그램의 객체 모델과 데이터베이스의 모델이 부합됨

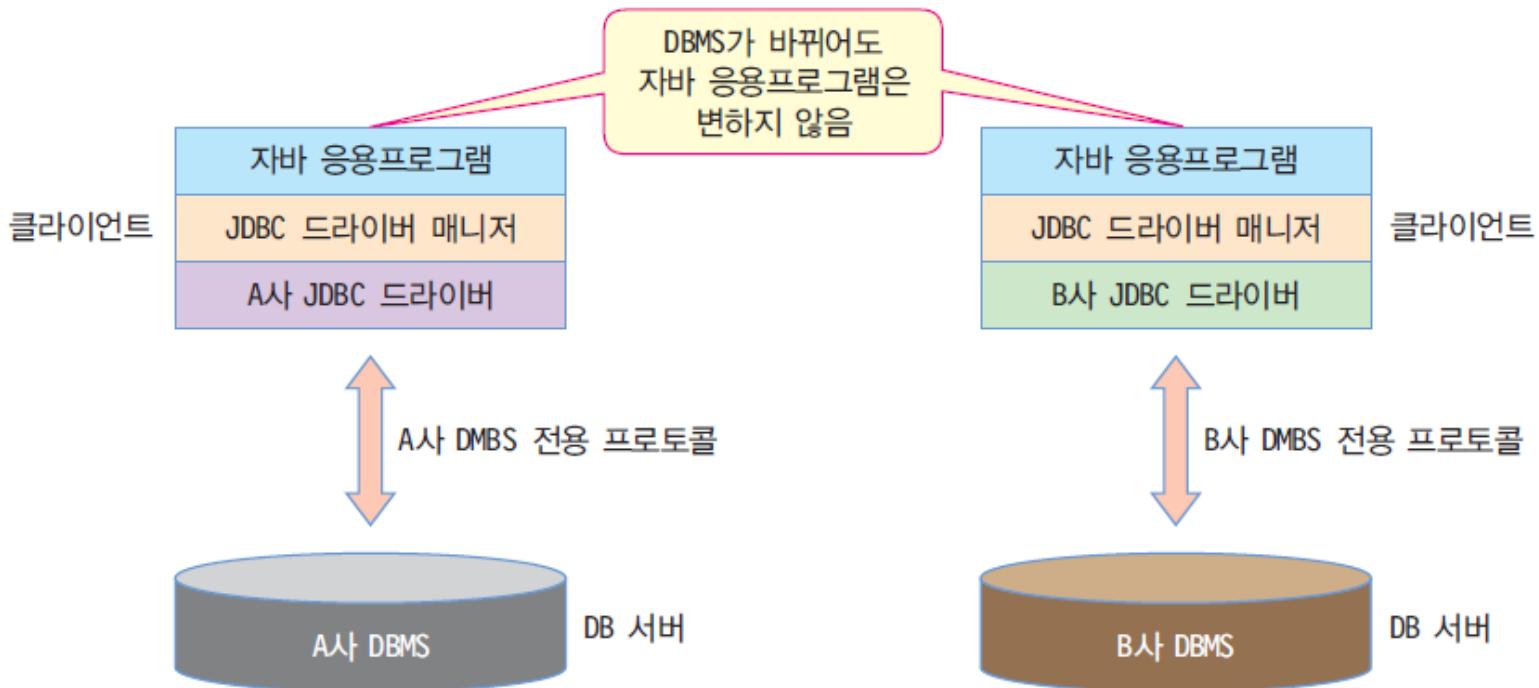
SQL과 JDBC

6

- SQL(Structured Query Language)
 - ▣ 관계형 데이터베이스 관리 시스템에서 사용
 - ▣ 데이터베이스 스키마 생성, 자료의 검색, 관리, 수정, 그리고 데이터베이스 객체 접근 관리를 위해 고안된 언어
 - ▣ 데이터베이스로부터 정보를 추출하거나 갱신하기 위한 표준 대화식 프로그래밍 언어
 - 다수의 데이터베이스 관련 프로그램들이 SQL을 표준으로 채택
- JDBC (Java DataBase Connectivity)
 - ▣ 관계형 데이터베이스에 저장된 데이터를 접근 및 조작할 수 있게 하는 API
 - ▣ 다양한 DBMS에 대해 일관된 API로 데이터베이스 연결, 검색, 수정, 관리 등을 할 수 있게 함

JDBC 구조

7

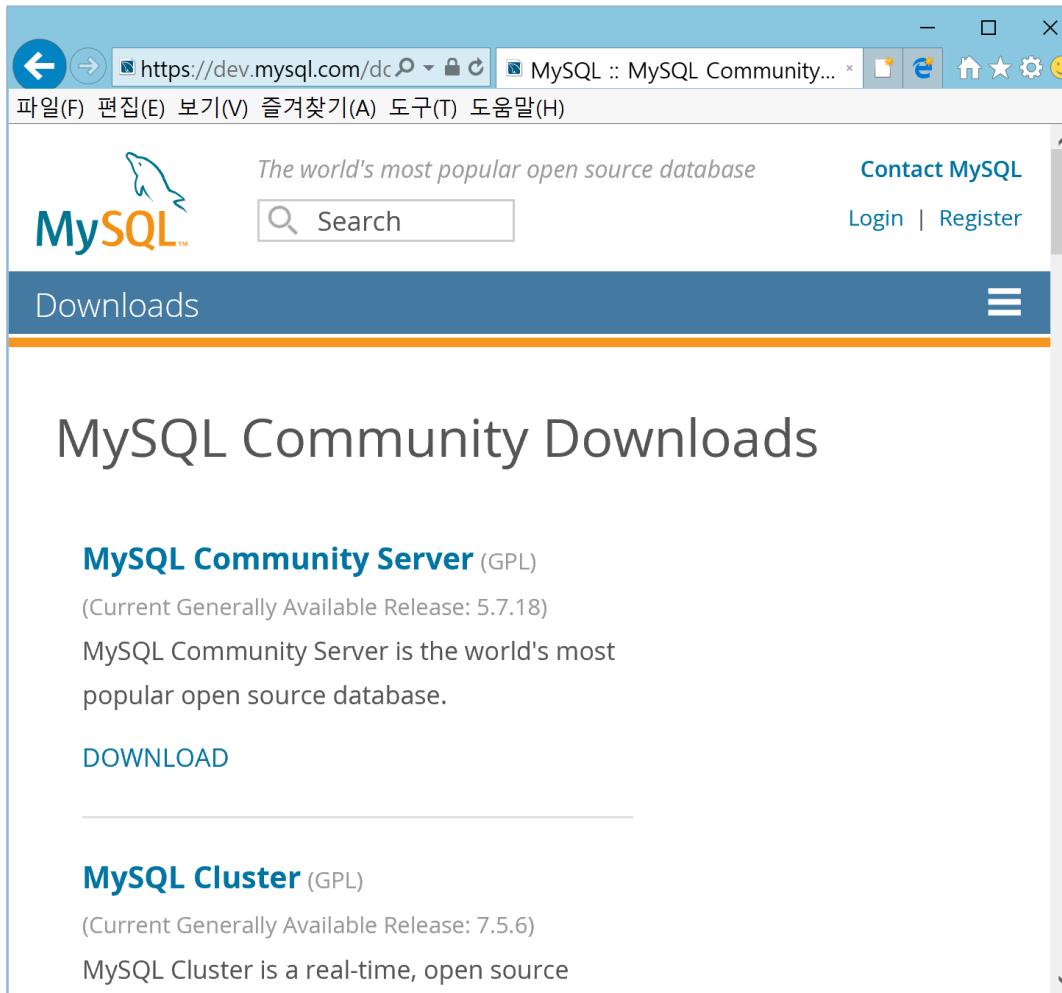


- JDBC 드라이버 매니저
 - 자바 API에서 지원하며 DBMS를 접근할 수 있는 JDBC 드라이버 로드
- JDBC 드라이버
 - DBMS마다 고유한 JDBC 드라이버 제공, JDBC 드라이버와 DBMS는 전용 프로토콜로 데이터베이스 처리
- DBMS
 - 데이터베이스 관리 시스템. 데이터베이스 생성·삭제, 데이터 생성·검색·삭제 등 전담 소프트웨어 시스템

MySQL 서버 설치 :

8

다운로드 사이트 : <http://www.mysql.com/downloads/>



The screenshot shows a web browser displaying the MySQL Community Downloads page. The URL in the address bar is <https://dev.mysql.com/downloads/>. The page header includes the MySQL logo, a search bar, and links for 'Contact MySQL' and 'Login | Register'. A blue navigation bar at the top says 'Downloads'. The main content area features a section for 'MySQL Community Server (GPL)' with a download link, and another for 'MySQL Cluster (GPL)'.

The world's most popular open source database

Contact MySQL

Login | Register

Downloads

MySQL Community Downloads

MySQL Community Server (GPL)
(Current Generally Available Release: 5.7.18)

MySQL Community Server is the world's most popular open source database.

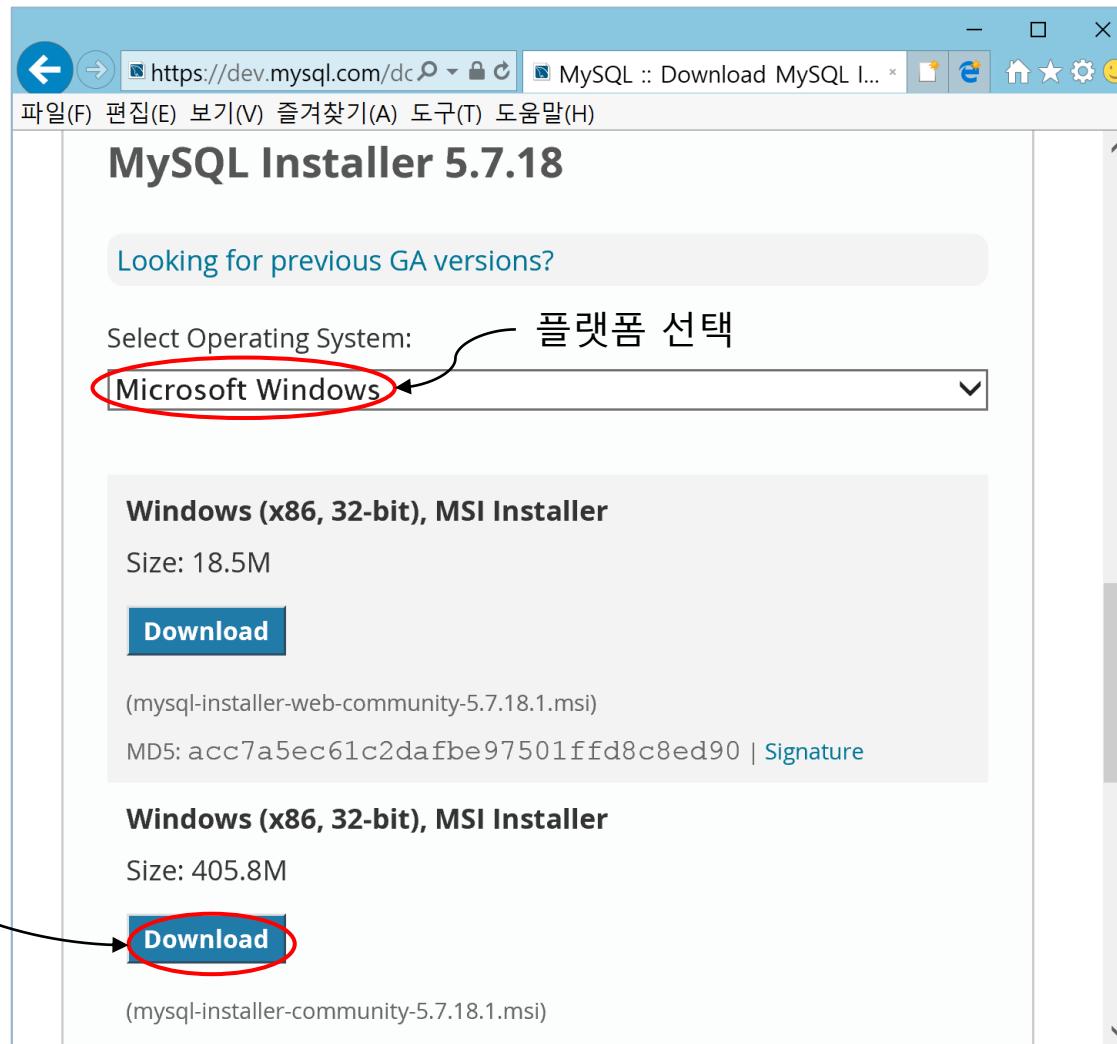
DOWNLOAD

MySQL Cluster (GPL)
(Current Generally Available Release: 7.5.6)

MySQL Cluster is a real-time, open source

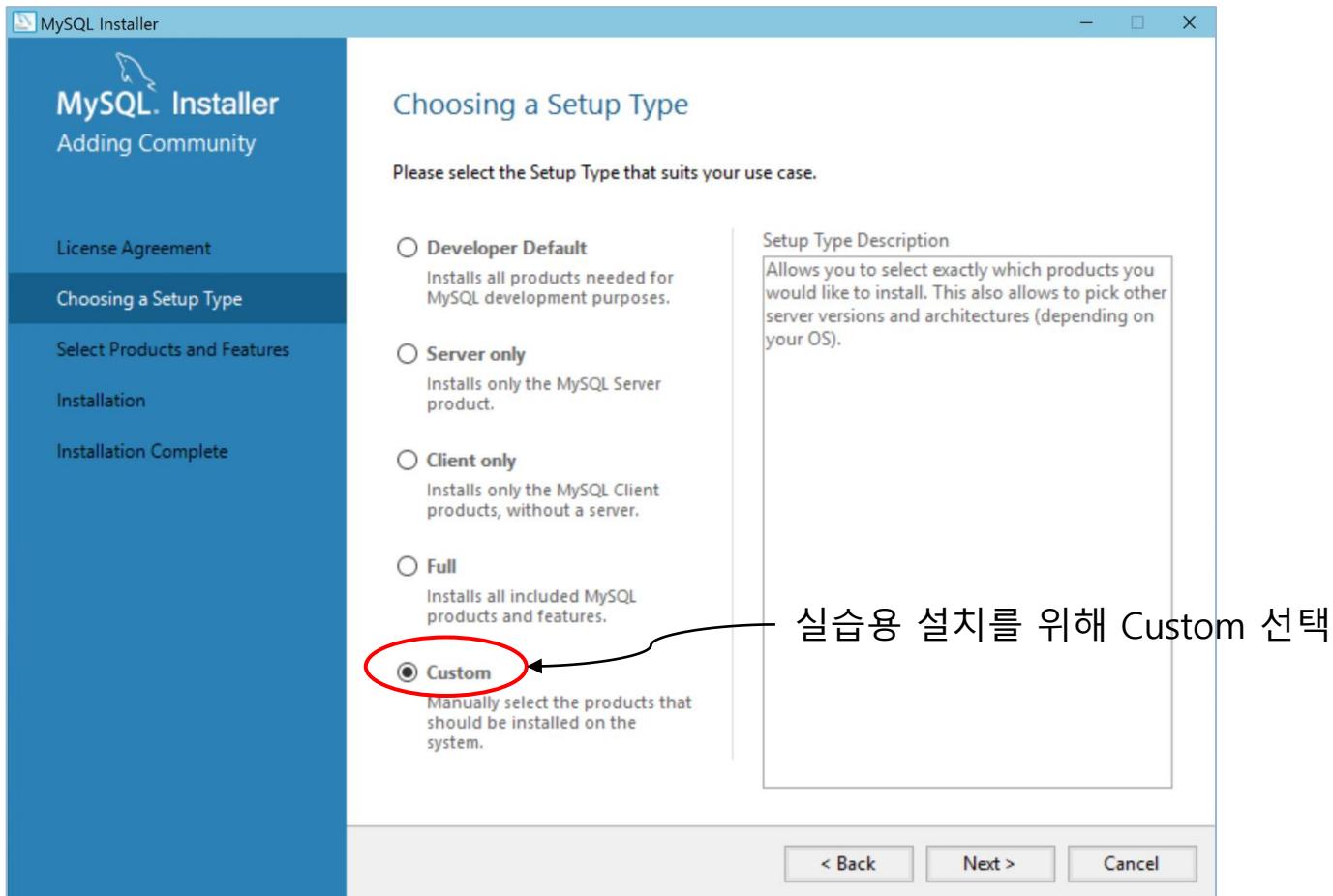
플랫폼 선택 후 다운로드

9



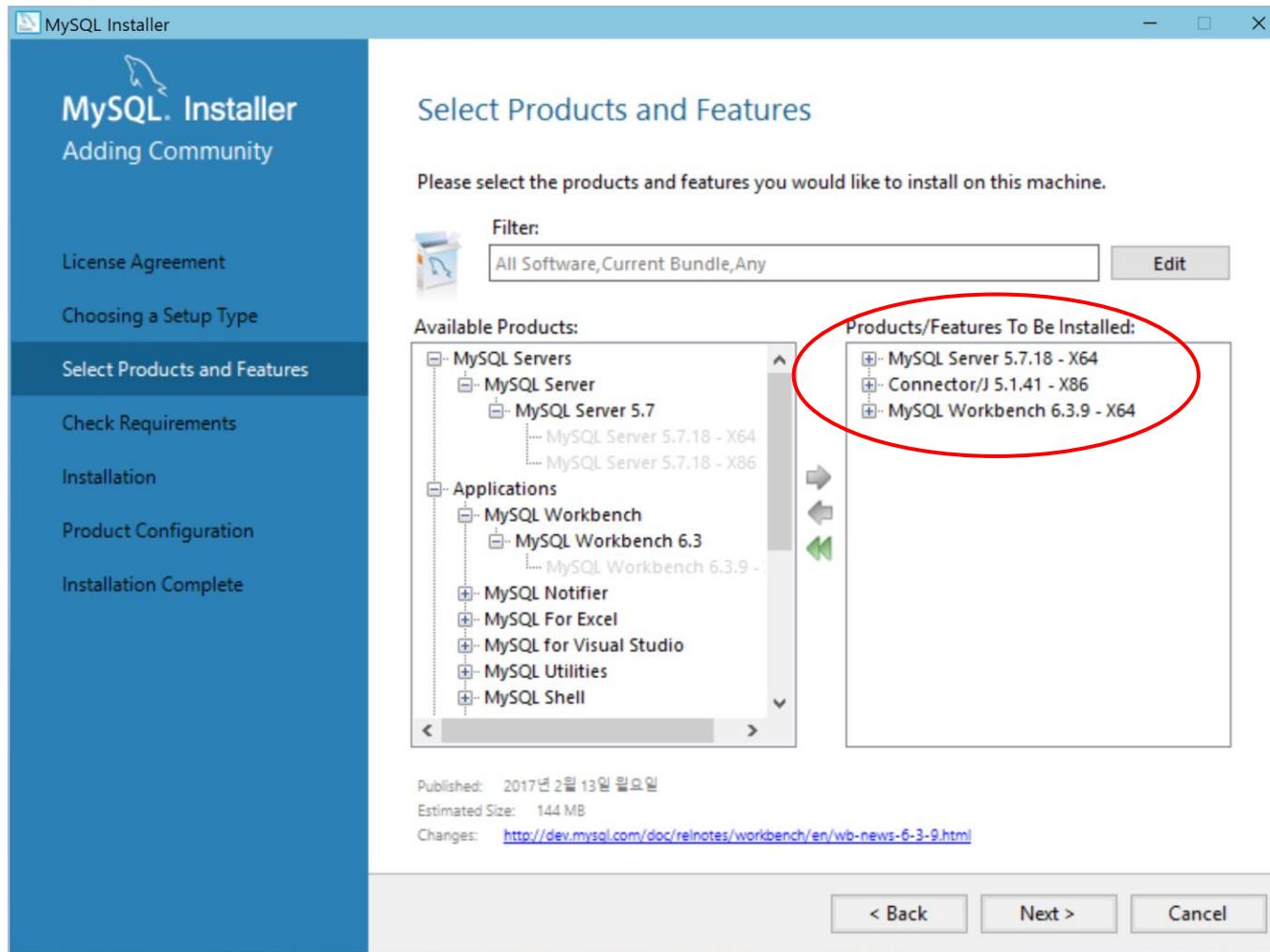
Custom 탑입으로 설치

10



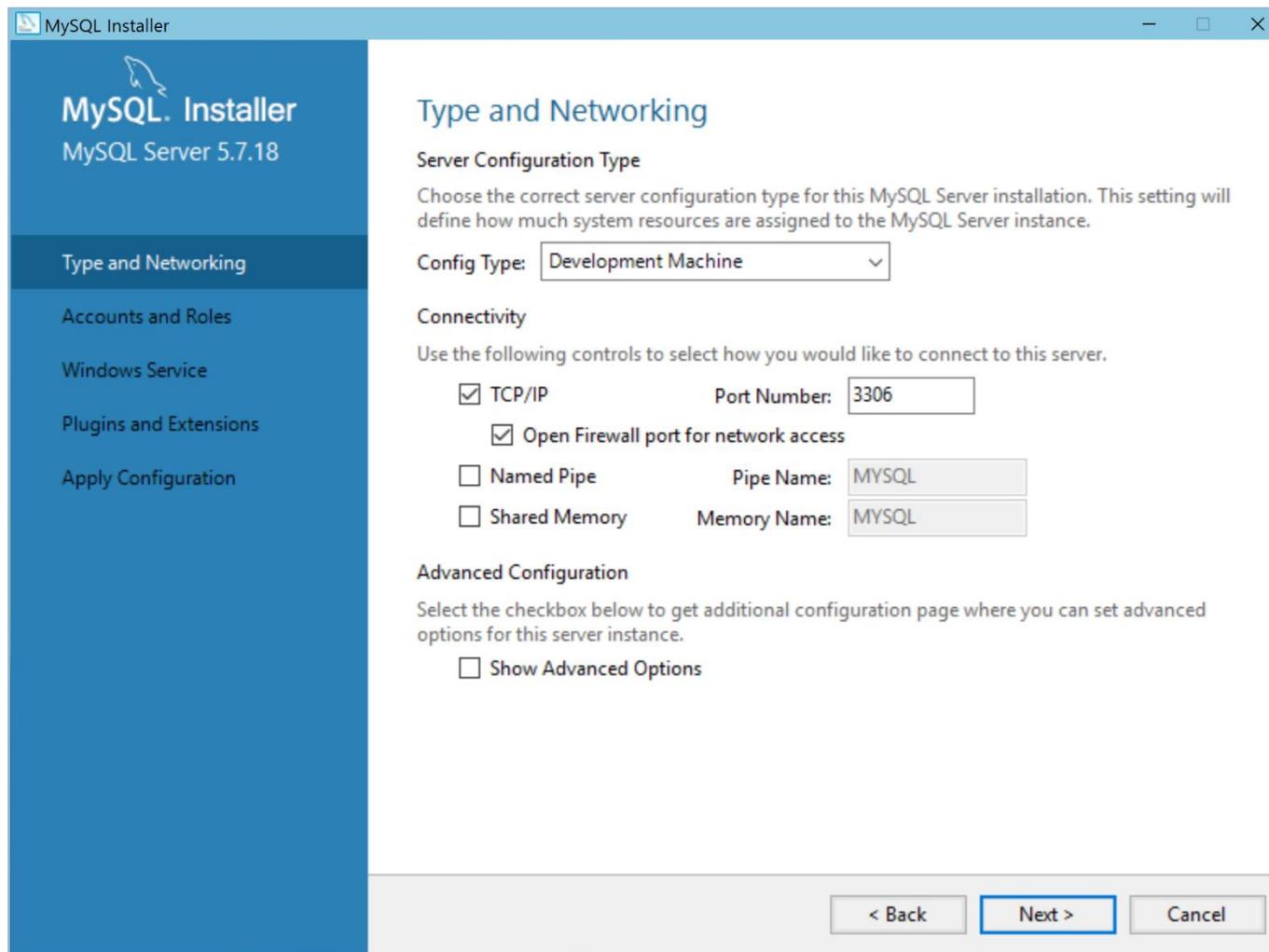
설치 항목 선택

11



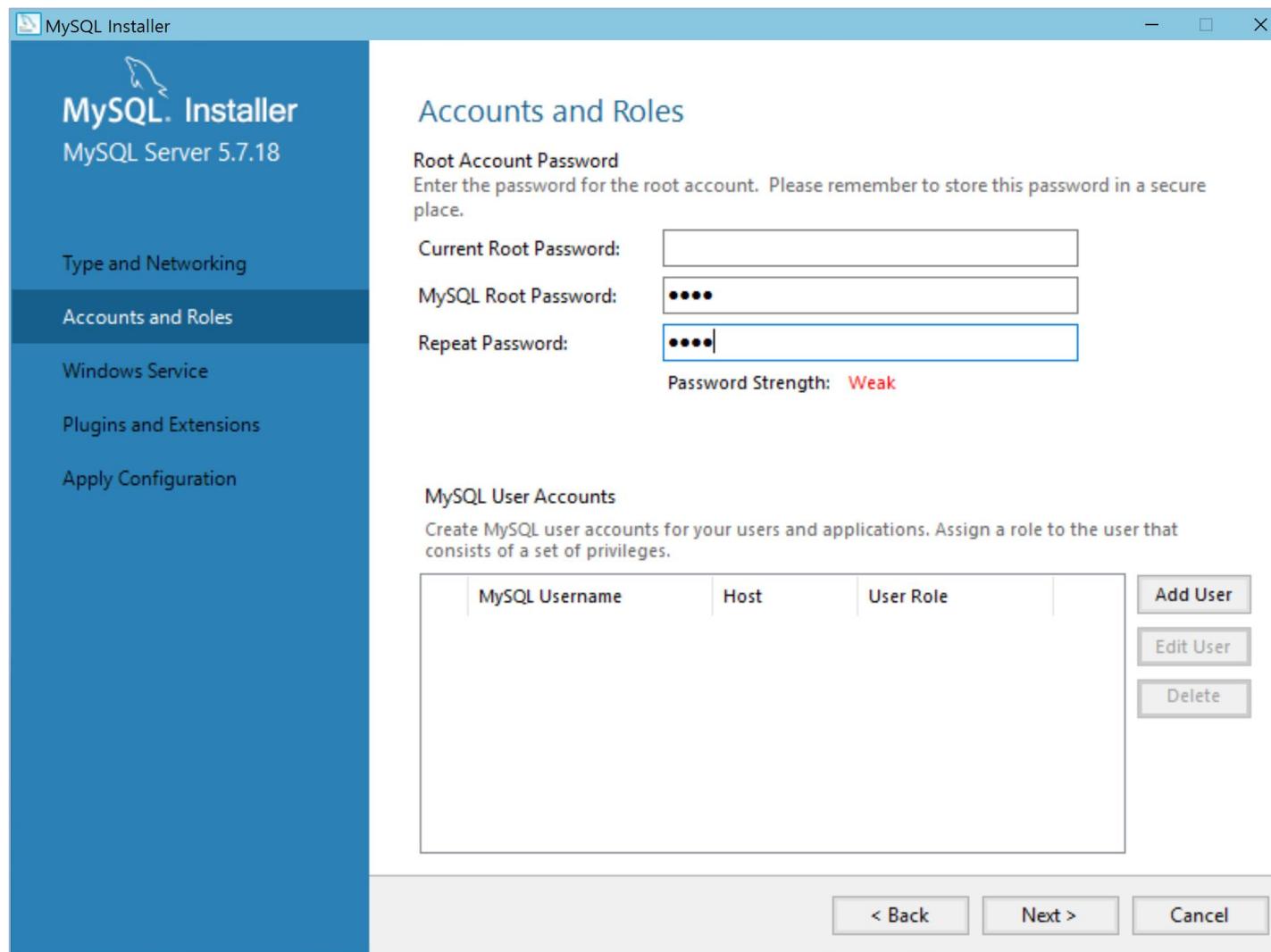
MySQL 서버 설정

12



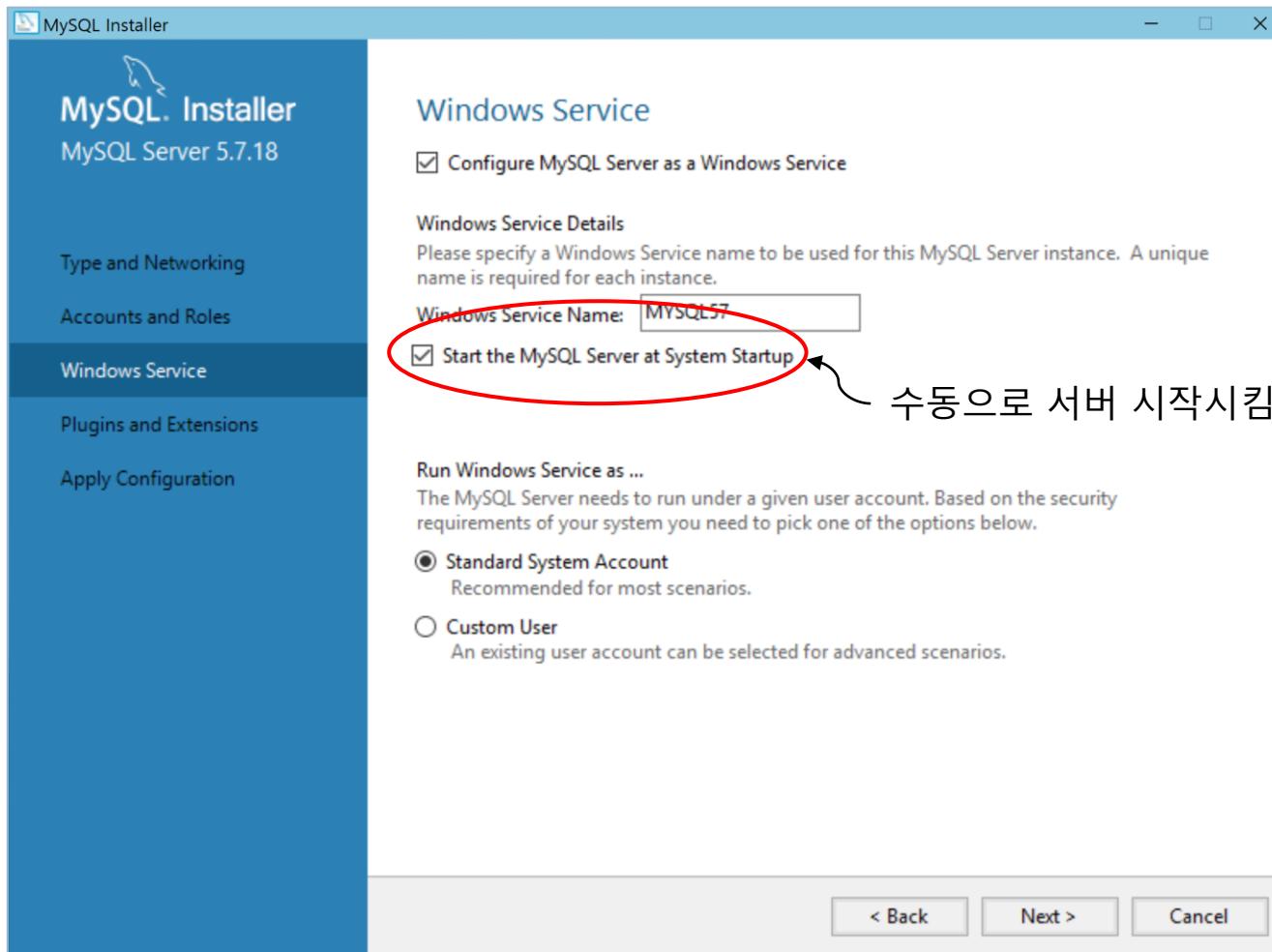
MySQL 서버의 루트 계정 암호 설정

13



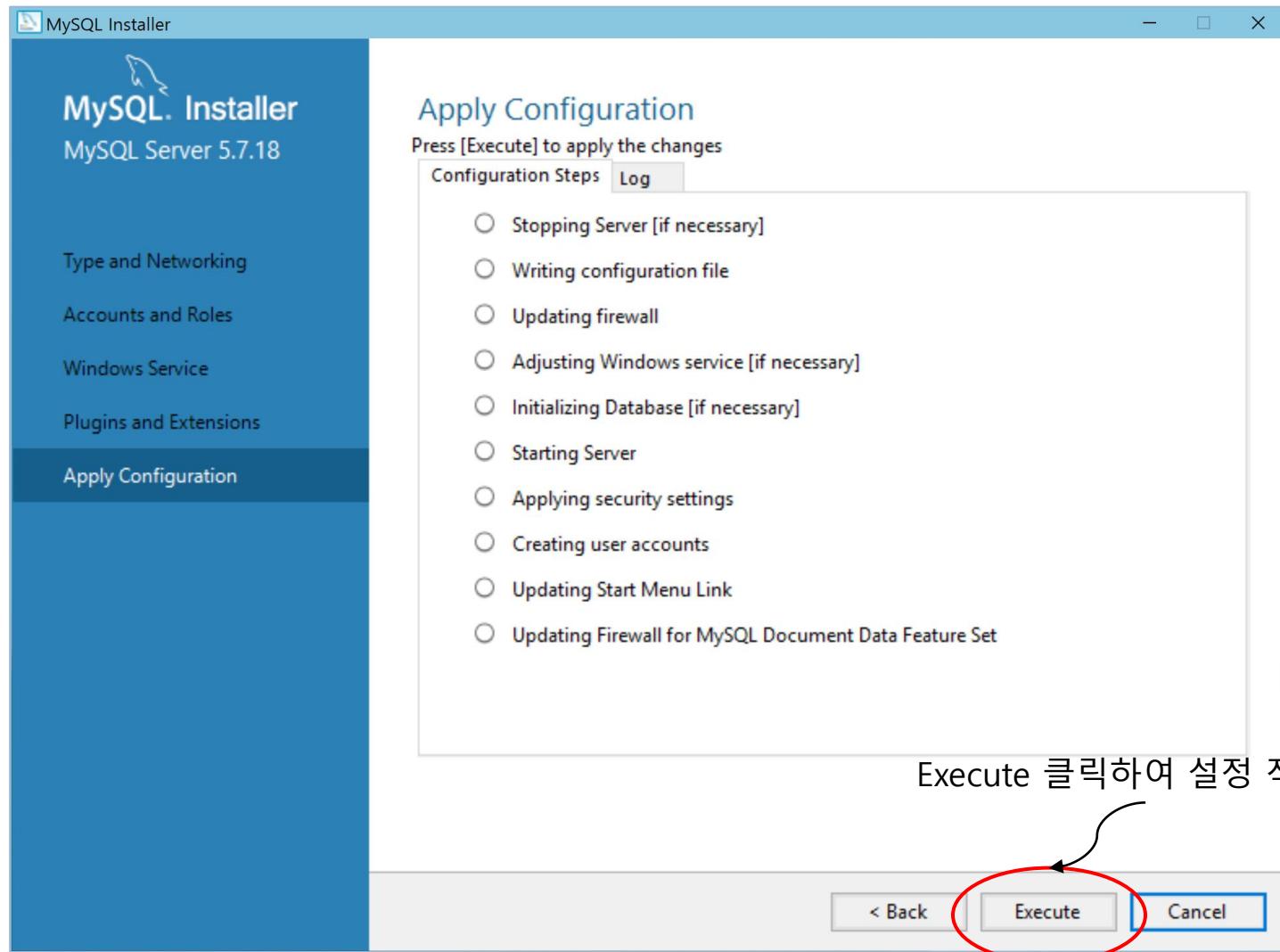
MySQL 서버를 윈도우 서비스로 등록

14



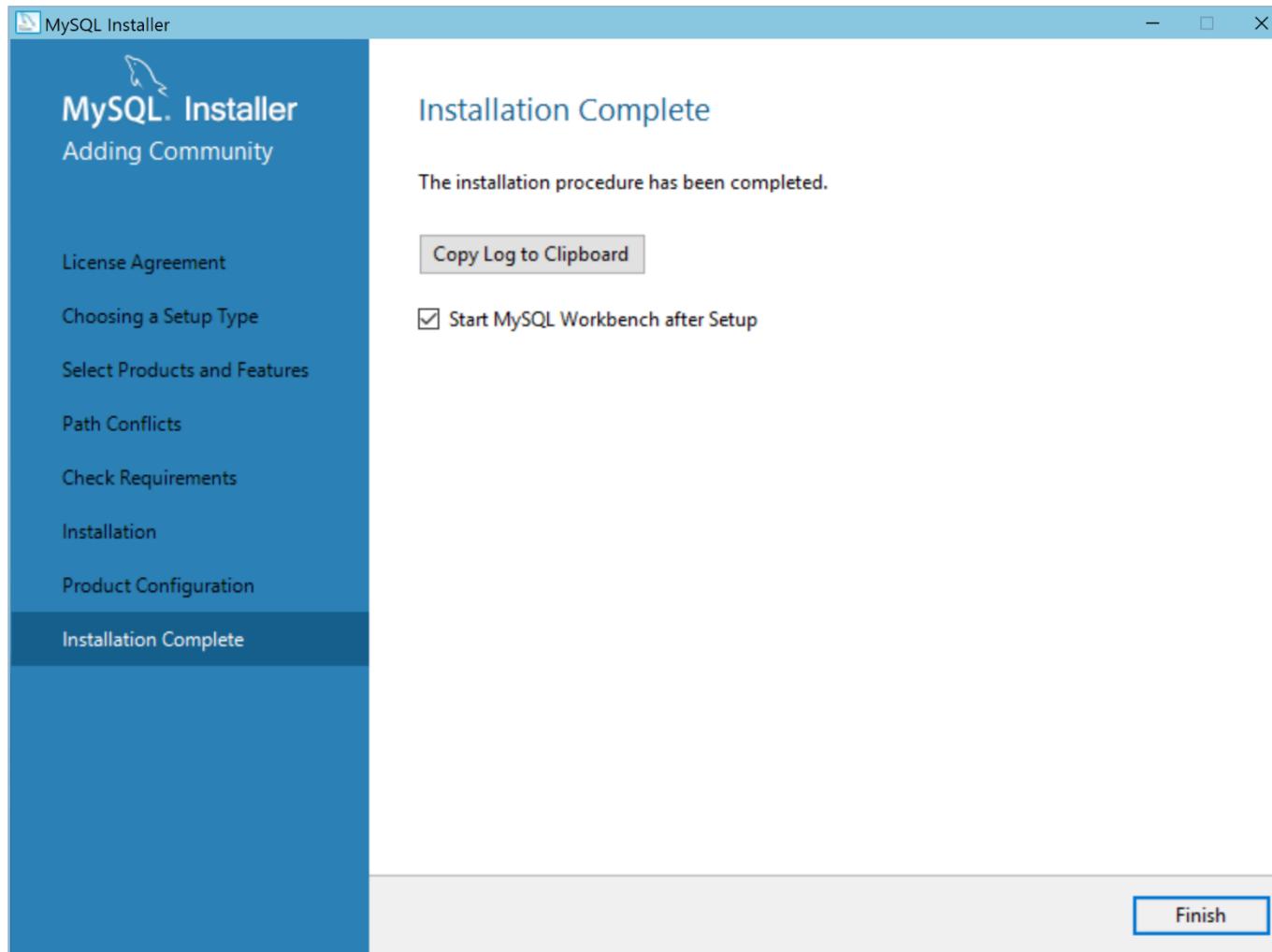
설정 적용

15



설치 완료

16



MySQL Workbench를 이용한 데이터베이스 활용

MySQL 서버 실행

18

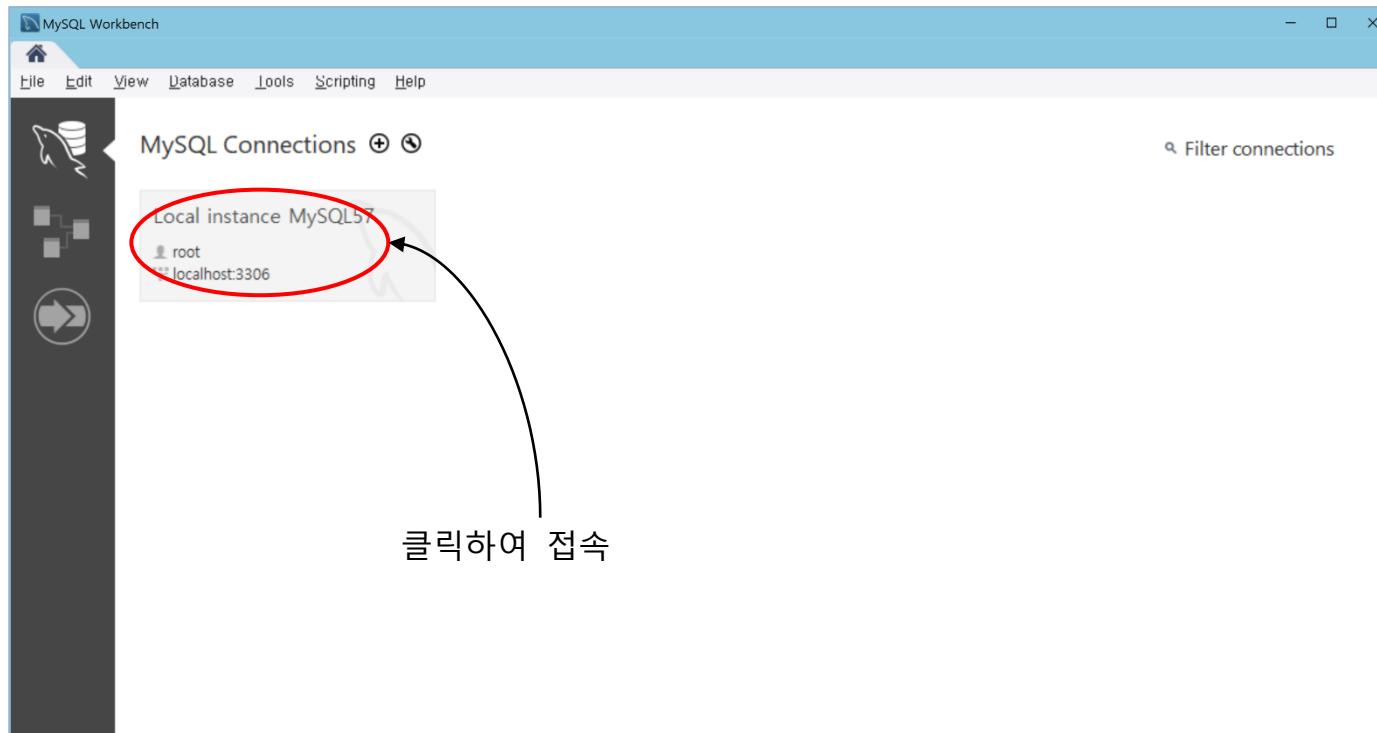
- MySQL 서버를 윈도우 서비스로 설정하고, '수동으로 시작'시키도록 지정하였기 때문에, MySQL 서버가 실행되게 하려면 작업 관리자에서 서비스를 시작시켜야 한다.



서버 접속

19

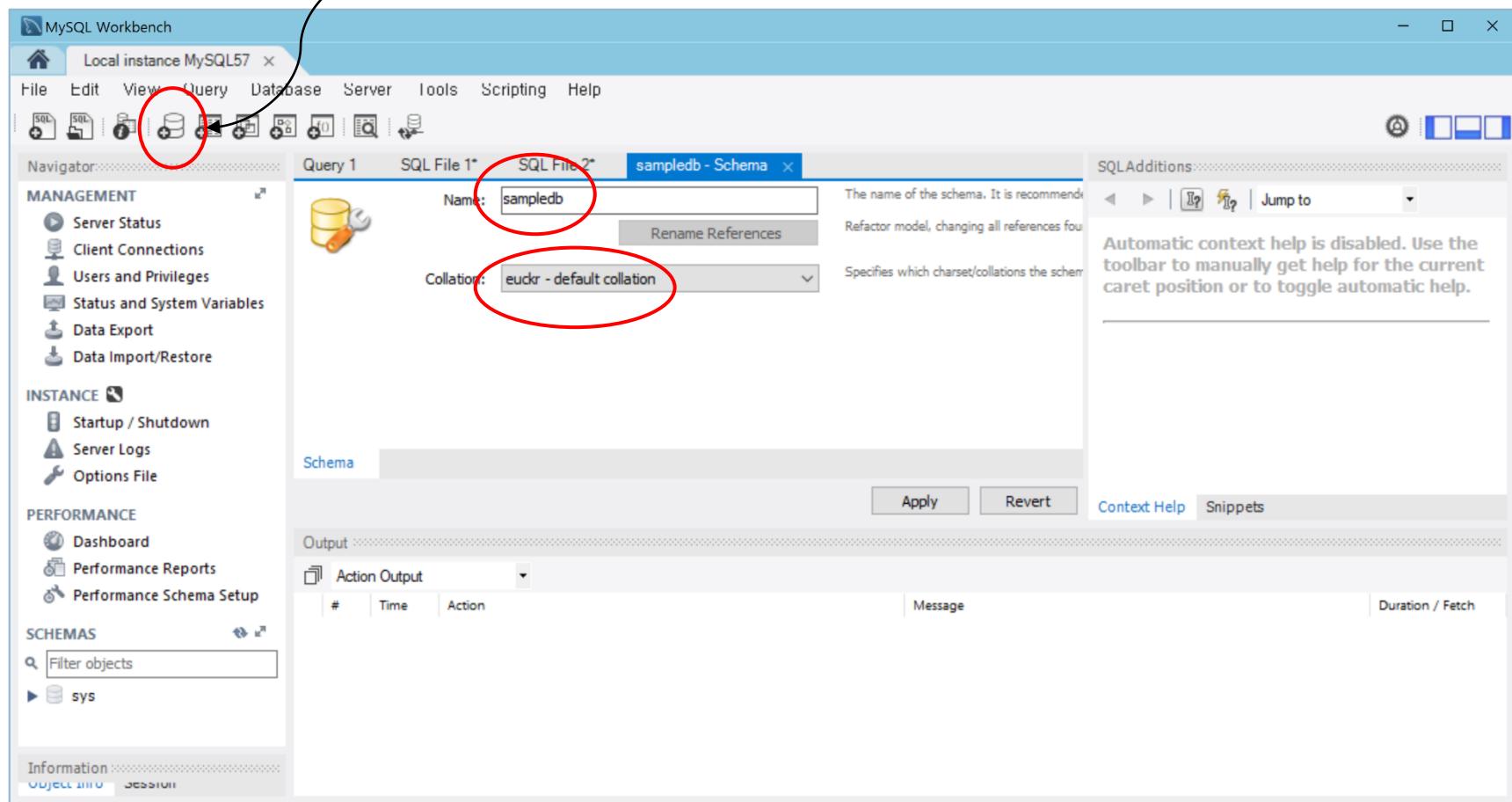
MySQL Workbench의 실행



스키마 생성

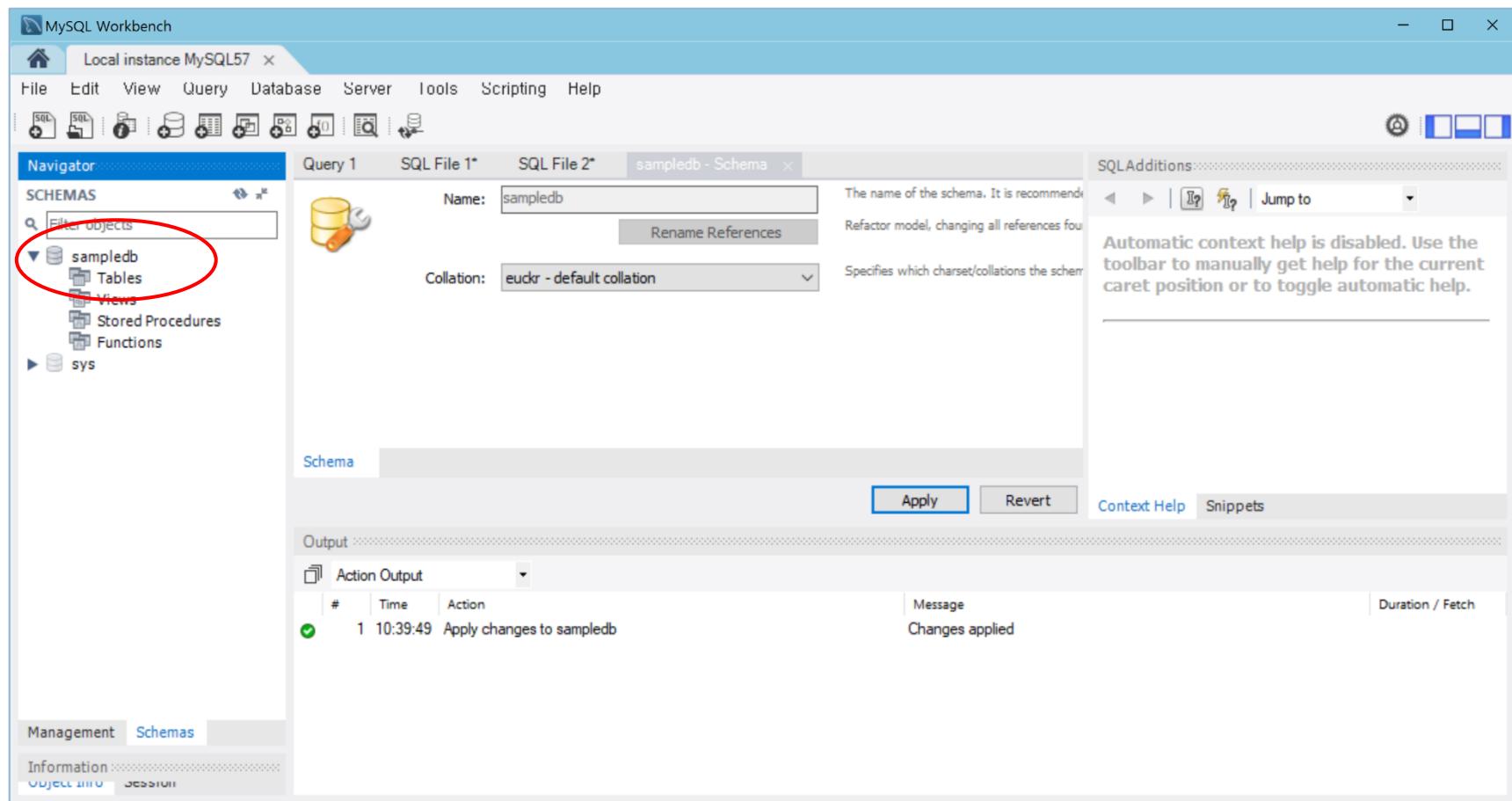
20

클릭하여 스키마 생성



스키마 생성 화면

21



테이블 생성

22

- 데이터 저장을 위해 테이블을 생성
- 다음과 같은 구조의 student 테이블 생성

| id | name | dept |
|---------|-------------|-------------|
| char(7) | varchar(10) | varchar(20) |

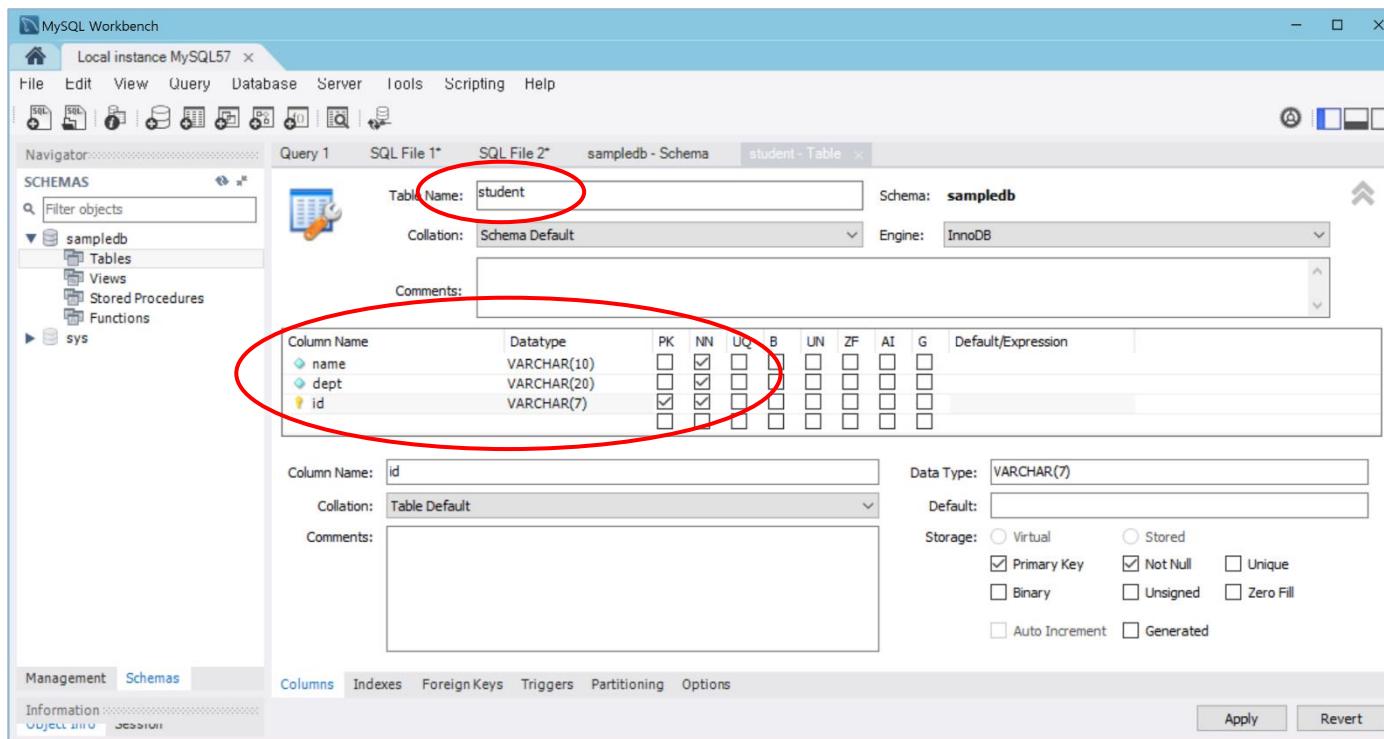
- name은 varchar 타입으로 10자
- dept는 varchar 타입으로 20자
- id은 char 타입으로 7자
- 저장할 데이터

| id | name | dept |
|---------|------|--------|
| 1091011 | 김철수 | 컴퓨터시스템 |
| 0792012 | 최고봉 | 멀티미디어 |
| 0494013 | 이기자 | 컴퓨터공학 |

테이블 생성

23

□ 테이블 생성



테이블 생성 결과

24

MySQL Workbench

Local instance MySQL57

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

sampledb

Tables

student

Columns (highlighted with a red circle)

Indexes

Foreign Keys

Triggers

Views

Stored Procedures

Functions

sys

Query 1 SQL File 1* SQL File 2* sampledb - Schema student - Table

Table Name: student Schema: sampledb

Collation: euckr - default collation Engine: InnoDB

Comments:

| Column Name | Datatype | PK | NN | UQ | B | UN | ZF | AI | G | Default/Expression |
|-------------|-------------|-------------------------------------|-------------------------------------|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|--------------------|
| name | VARCHAR(10) | <input type="checkbox"/> | <input checked="" type="checkbox"/> | <input type="checkbox"/> | |
| dept | VARCHAR(20) | <input type="checkbox"/> | <input checked="" type="checkbox"/> | <input type="checkbox"/> | |
| id | VARCHAR(7) | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input type="checkbox"/> | |

Column Name: id Data Type: VARCHAR(7)

Collation: Table Default

Comments:

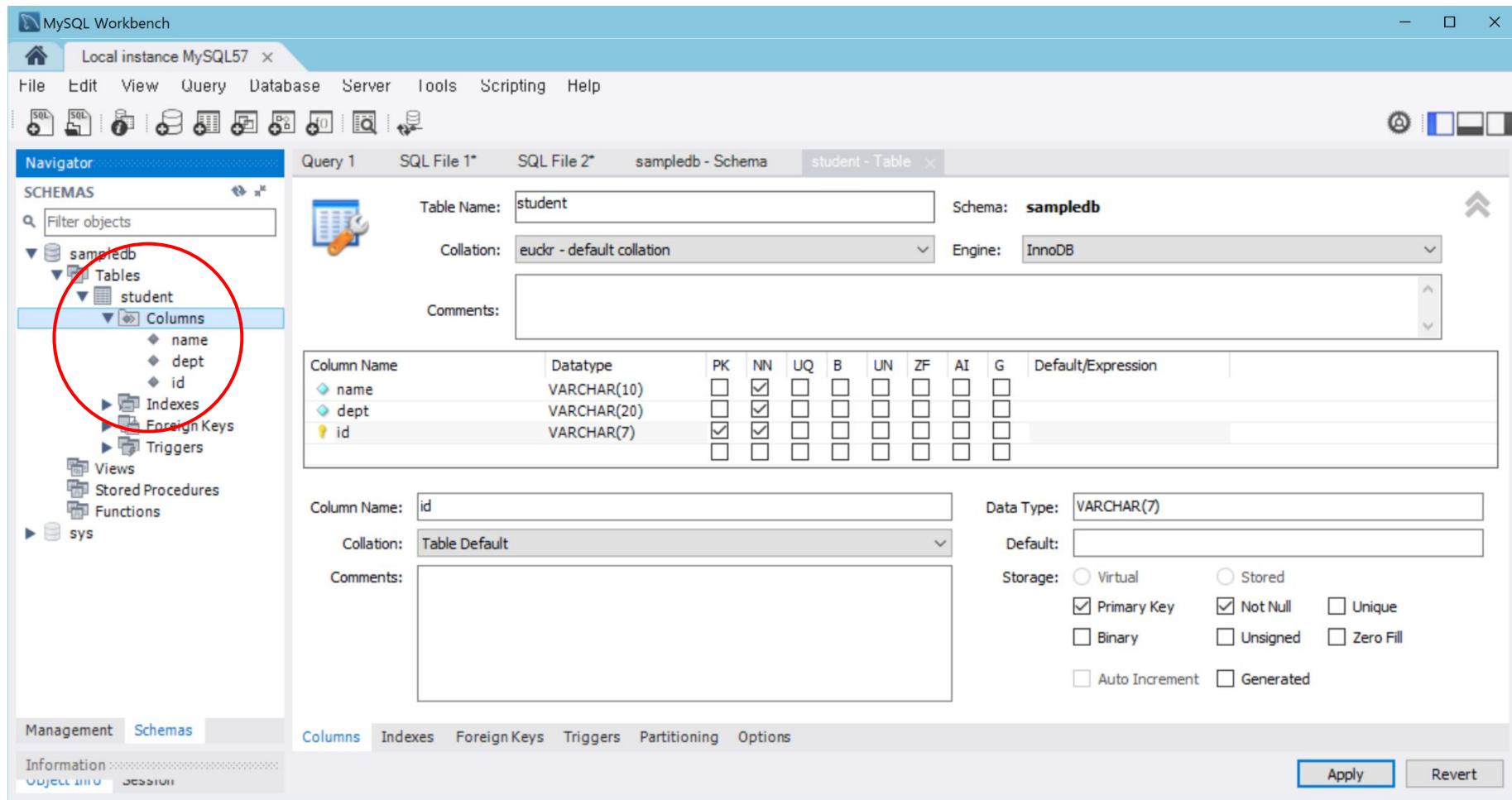
Storage: Virtual Stored
 Primary Key Not Null Unique
 Binary Unsigned Zero Fill
 Auto Increment Generated

Management Schemas

Information

Columns Indexes Foreign Keys Triggers Partitioning Options

Apply Revert



레코드 추가

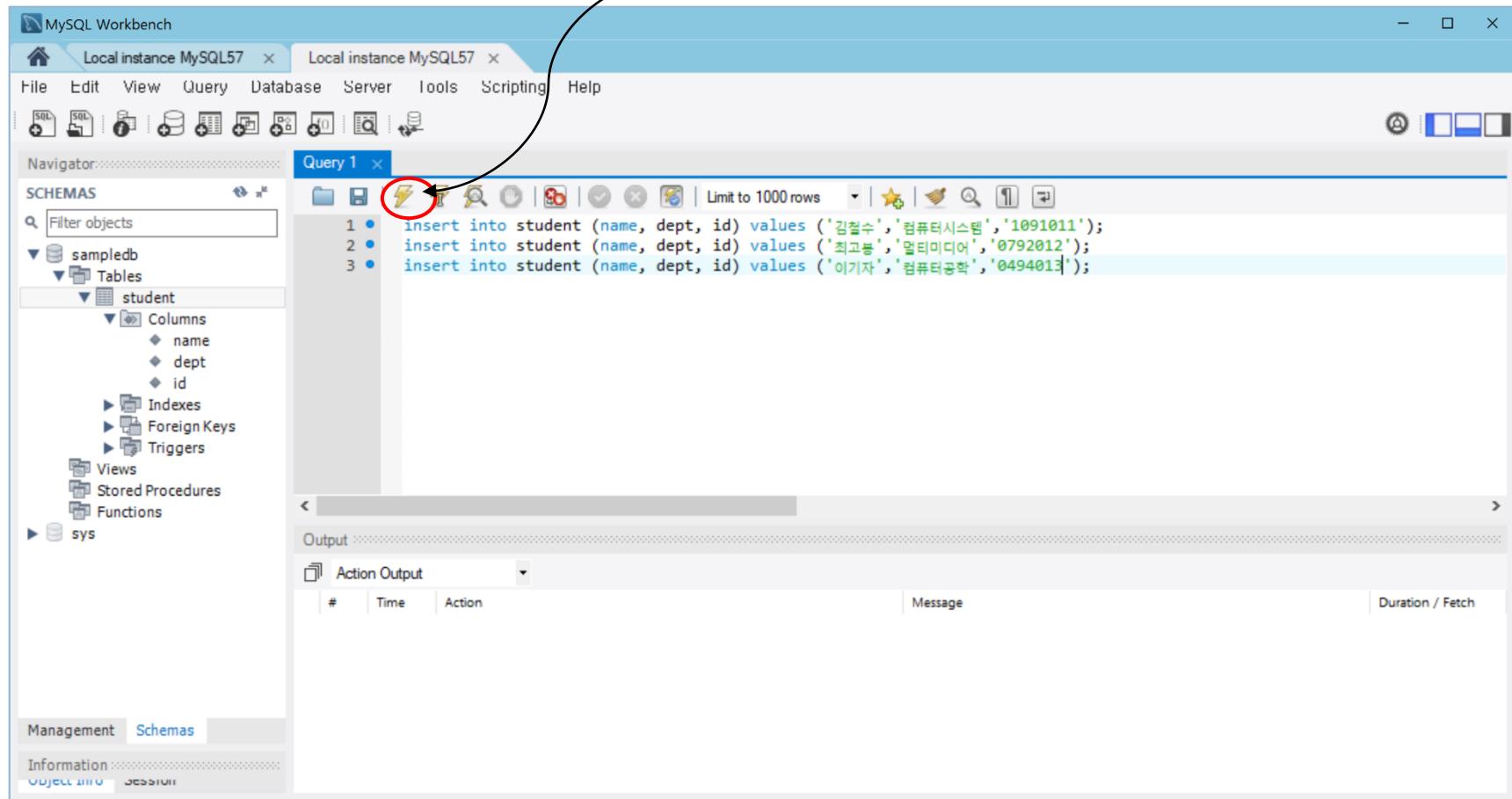
25

- 테이블에는 레코드 단위로 데이터 추가
- 명령
 - ▣ insert into student (name, dept, id) values ('김철수','컴퓨터시스템','1091011');
 - insert into 다음에 테이블 이름 지정
 - 테이블 이름 다음 괄호 안에 열 이름을 콤마로 구분하여 나열
 - values 다음 괄호 안에 열의 값을 콤마로 구분하여 나열
 - 문자 타입의 데이터는 단일 인용 부호로 묶어서 표시함에 유의

레코드 추가 사례

26

클릭하여 쿼리문 실행



데이터 검색

27

- select문으로 테이블 내의 데이터 검색
- 명령
 - ▣ select 다음에는 데이터를 추출할 열 이름을 콤마로 분리하여 나열
 - ▣ 모든 열에 대해 데이터를 추출할 때는 *를 열 이름 대신 사용
 - ▣ from 다음에 테이블 이름을 지정
 - ▣ where 다음에 검색 조건 지정. 위의 예에서 dept 값이 '컴퓨터공학'인 레코드 검색
 - ▣ where는 생략 가능

select 명령을 이용한 데이터 검색

28

The screenshot shows the MySQL Workbench interface. In the top navigation bar, there are tabs for 'Local instance MySQL57' and 'Local instance MySQL57 (sa...'. The main menu includes File, Edit, View, Query, Database, Server, Tools, Scripting, and Help. Below the menu is a toolbar with various icons for database management tasks.

In the left sidebar, under the 'Navigator' section, the 'SCHEMAS' tab is selected. It shows the 'sampledb' schema expanded, with 'Tables' selected. Inside 'Tables', the 'student' table is selected, showing its columns: name, dept, and id. Other table options like 'Indexes', 'Foreign Keys', and 'Triggers' are also visible.

The central area contains a 'Query 1' tab with the SQL query: `select * from student where dept='컴퓨터공학';`. Below the query is a 'Result Grid' showing the results of the query:

| | name | dept | id |
|---|------|-------|---------|
| 1 | 이기자 | 컴퓨터공학 | 0494013 |
| | HULL | HULL | HULL |

The bottom status bar indicates 'Query Completed'.

데이터 수정

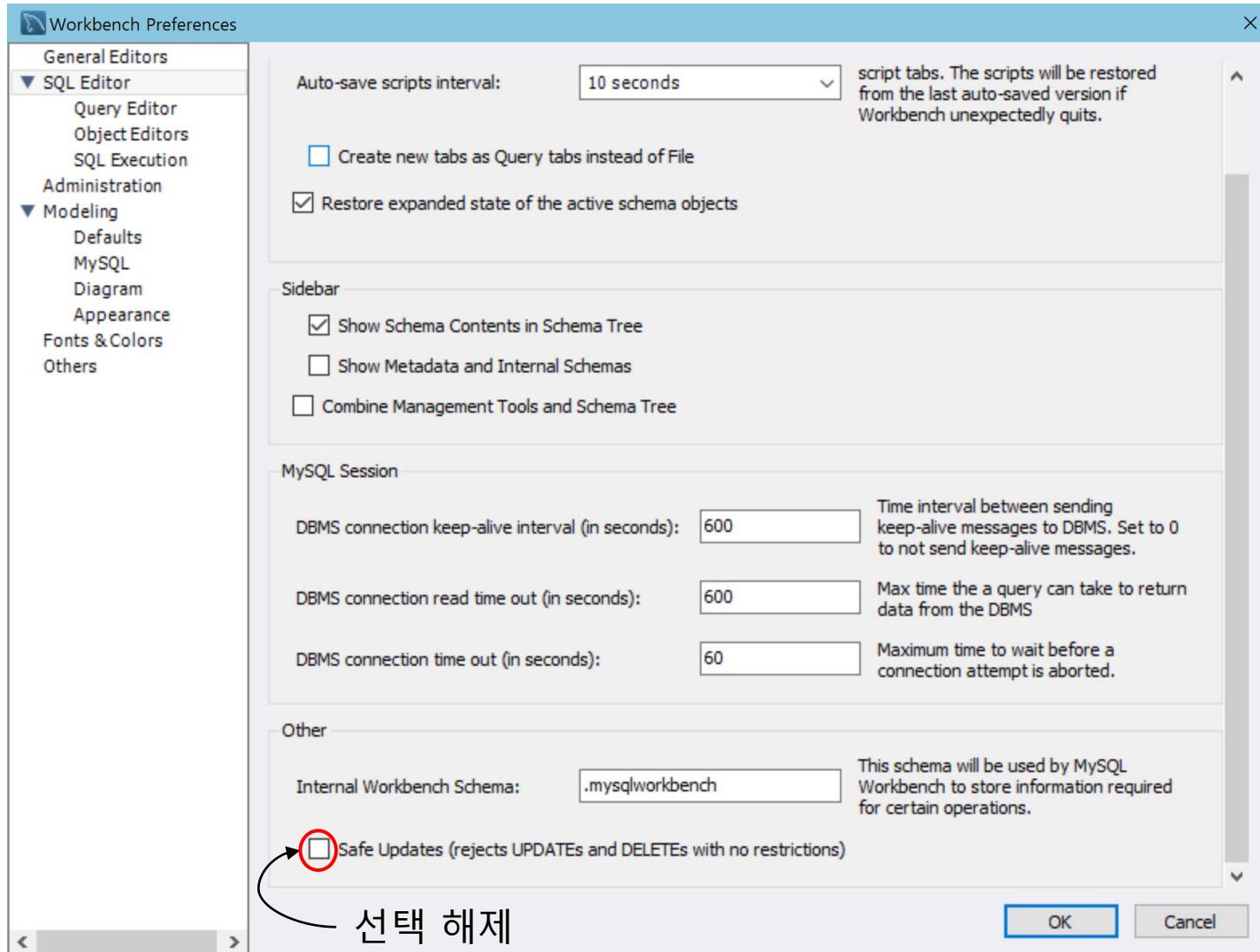
29

- update문을 이용하여 데이터 수정
- 명령
 - ▣ update student set dept='컴퓨터공학' where name='최고봉';
 - update 다음에는 테이블 이름 지정
 - set 다음에 수정할 열의 이름과 값을 콤마로 분리하여 나열
 - where 다음에는 검색 조건을 지정. 위의 예에서는 name 값이 '최고봉'인 레코드의 데이터 수정
 - where는 생략 가능

데이터 수정

30

Workbench에서 데이터 수정, 삭제 시 옵션 수정 필요



update 명령을 이용한 데이터 수정

31

The screenshot shows the MySQL Workbench interface. In the Navigator pane, the 'sampledb' schema is selected, specifically the 'student' table. The 'Query 1' editor contains the following SQL code:

```
1 • update student set dept='컴퓨터공학' where name='최고봉';
2 • select * from student;
```

The Result Grid displays the current state of the 'student' table:

| | name | dept | id |
|------|--------|---------|----|
| 이기자 | 컴퓨터공학 | 0494013 | |
| 최고봉 | 컴퓨터공학 | 0792012 | |
| 김철수 | 컴퓨터시스템 | 1091011 | |
| HULL | HULL | HULL | |

레코드 삭제

32

- delete문을 이용하여 데이터 삭제
- 명령
 - ▣ delete from student where name='최고봉';
 - delete from 다음에는 테이블 이름 지정
 - where 다음에는 검색 조건을 지정. 위의 예에서는 name 값이 '최고봉'인 레코드 삭제
 - where는 생략 가능

delete 명령을 이용한 레코드 삭제

33

The screenshot shows the MySQL Workbench interface. In the top-left corner, the title bar reads "MySQL Workbench" and "Local instance MySQL57 (sa...)".

The main window has a toolbar at the top with various icons for file operations, queries, and database management. Below the toolbar is a menu bar with File, Edit, View, Query, Database, Server, Tools, Scripting, and Help.

On the left side, there is a Navigator pane showing the database schema. It lists the "sampledb" schema, which contains a "Tables" folder with a "student" table. Other objects like Views, Stored Procedures, and Functions are also listed under sampledb. A "sys" schema is also visible.

In the center, there are two tabs: "Query 1" and "SQL File 3*". The "Query 1" tab contains the following SQL code:

```
1 delete from student where name='최고봉';
2 select * from student;
```

Below the code, the "Result Grid" shows the contents of the "student" table:

| | name | dept | id |
|---|------|--------|---------|
| 1 | 이기자 | 컴퓨터공학 | 0494013 |
| 2 | 김철수 | 컴퓨터시스템 | 1091011 |
| | HULL | HULL | HULL |

At the bottom of the interface, there are tabs for Management, Schemas, Information, Object Info, Session, and a status bar indicating "Query Completed". On the right side, there is a vertical toolbar with icons for Result Grid, Form Editor, and other tools.

자바의 JDBC 프로그래밍

JDBC 프로그래밍

35

- 데이터베이스 연결 설정
- MySQL 서버의 JDBC 드라이버 로드

```
try {  
    Class.forName("com.mysql.jdbc.Driver");  
} catch (ClassNotFoundException e) {  
    e.printStackTrace();  
}
```

- `Class.forName()`은 동적으로 자바 클래스 로딩
- MySQL의 JDBC 드라이버 클래스인 `com.mysql.jdbc.Driver` 로드
- 드라이버의 클래스 이름은 DB의 드라이버마다 다를 수 있으므로 JDBC 드라이버 문서 참조할 것
- 자동으로 드라이버 인스턴스를 생성하여 `DriverManager`에 등록
- 해당 드라이버가 없으면 `ClassNotFoundException` 발생

자바 응용프로그램과 JDBC의 연결

36

□ 연결

```
try {  
    Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/sampledb", "root", "");  
} catch (SQLException e) {  
    e.printStackTrace();  
}
```

- DriverManager는 자바 어플리케이션을 JDBC 드라이버에 연결시켜주는 클래스로서 getConnection() 메소드로 DB에 연결하여 Connection 객체 반환
- getConnection에서 jdbc: 이후에 지정되는 URL의 형식은 DB에 따라 다르므로 JDBC 문서를 참조
 - MySQL 서버가 같은 컴퓨터에서 동작하므로 서버 주소를 *localhost*로 지정
 - MySQL의 경우 디폴트로 3306 포트를 사용
 - *sampledb*는 앞서 생성한 DB의 이름
- "root"는 DB에 로그인할 계정 이름이며, ""는 계정 패스워드

예제 16-1 : sampledb의 데이터베이스 연결하는 JDBC 프로그램 작성

37

JDBC를 이용하여 sampledb 데이터베이스에 연결하는 자바 응용프로그램을 작성하라.

```
import java.sql.*;  
  
public class JDBC_Ex1 {  
    public static void main (String[] args) {  
        try {  
            Class.forName("com.mysql.jdbc.Driver");  
            Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/sampledb", "root","");
            System.out.println("DB 연결 완료");  
        } catch (ClassNotFoundException e) {  
            System.out.println("JDBC 드라이버 로드 에러");  
        } catch (SQLException e) {  
            System.out.println("DB 연결 오류");  
        }  
    }  
}
```

DB 연결 오류

또는

DB 연결 완료

데이터베이스 사용

38

□ Statement 클래스

- SQL문을 실행하기 위해서는 Statement 클래스를 이용
- 주요 메소드

| 메소드 | 설명 |
|------------------------------------|--|
| ResultSet executeQuery(String sql) | 주어진 sql문을 실행하고 결과는 ResultSet 객체에 반환 |
| int executeUpdate(String sql) | INSERT, UPDATE, 또는 DELETE과 같은 sql문을 실행하고, sql 문 실행으로 영향을 받은 행의 개수나 0을 반환 |
| void close() | Statement 객체의 데이터베이스와 JDBC 리소스를 즉시 반환 |

- 데이터 검색을 위해 executeQuery() 메소드 사용
- 추가, 수정, 삭제와 같은 데이터 변경은 executeUpdate() 메소드 사용

데이터베이스 사용

39

ResultSet 클래스

- SQL문 실행 결과를 얻어오기 위해서는 ResultSet 클래스를 이용
- 현재 데이터의 행(레코드 위치)을 가리키는 커서(cursor)를 관리
- 커서의 초기 값은 첫 번째 행 이전을 가리킴
- 주요 메소드

| 메소드 | 설명 |
|--------------------------------|--|
| boolean first() | 커서를 첫 번째 행으로 이동 |
| boolean last() | 커서를 마지막 행으로 이동 |
| boolean next() | 커서를 다음 행으로 이동 |
| boolean previous() | 커서를 이전 행으로 이동 |
| boolean absolute(int row) | 커서를 지정된 행 row로 이동 |
| boolean isFirst() | 첫 번째 행이면 true 반환 |
| boolean isLast() | 마지막 행이면 true 반환 |
| void close() | ResultSet 객체의 데이터베이스와 JDBC 리소스를 즉시 반환 |
| Xxx getXxx(String columnLabel) | Xxx는 해당 데이터 타입을 나타내며 현재 행에서 지정된 열 이름(columnLabel)에 해당하는 데이터를 반환한다. 예를 들어, int형 데이터를 읽는 메소드는 getInt()이다. |
| Xxx getXxx(int columnIndex) | Xxx는 해당 데이터 타입을 나타내며 현재 행에서 지정된 열 인덱스(columnIndex)에 해당하는 데이터를 반환한다. 예를 들어, int형 데이터를 읽는 메소드는 getInt()이다. |

데이터베이스 사용(1)

40

□ 테이블의 모든 데이터 검색

```
Statement stmt = conn.createStatement();
ResultSet rs = stmt.executeQuery("select * from student");
```

- Statement의 executeQuery()는 SQL문의 실행하여 실행 결과를 넘겨 줌
- 위의 SQL문의 student 테이블에서 모든 행의 모든 열을 읽어 결과를 rs에 저장

□ 특정 열만 검색

```
ResultSet rs = stmt.executeQuery("select name, id from student");
```

- ▣ 특정 열만 읽을 경우는 select문을 이용하여 특정 열의 이름 지정

□ 조건 검색

```
rs = stmt.executeQuery("select name, id, dept from student where id='0494013'");
```

- ▣ select문에서 where절을 이용하여 조건에 맞는 데이터 검색

데이터베이스 사용(2)

41

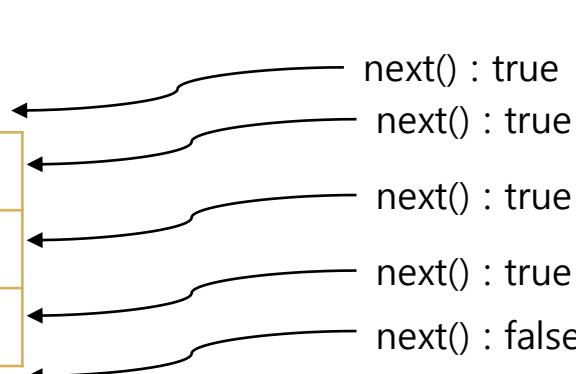
□ 검색된 데이터의 사용

```
while (rs.next()) {  
    System.out.println(rs.getString("name"));  
    System.out.println(rs.getString("id"));  
    System.out.println(rs.getString("dept"));  
}  
rs.close();
```

- Statement 객체의 executeQuery() 메소드
 - ResultSet 객체 반환
- ResultSet 인터페이스
 - DB에서 읽어온 데이터를 추출 및 조작할 수 있는 방법 제공
- next() 메소드
 - 다음 행으로 이동

| | | |
|-----|--------|---------|
| 김철수 | 컴퓨터시스템 | 1091011 |
| 최고봉 | 멀티미디어 | 0792012 |
| 이기자 | 컴퓨터공학 | 0494013 |

next() : true
next() : true
next() : true
next() : true
next() : false



예제 16-2 : 데이터 검색과 출력

42

앞서 생성한 sampledb의 student 테이블의 모든 데이터를 출력하고, 특별히 이름이 "이기자"인 학생의 데이터를 출력하는 프로그램을 작성하라.

```
import java.io.*;
import java.sql.*;

public class JDBC_Ex2 {
    public static void main (String[] args) {
        Connection conn;
        Statement stmt = null;
        try {
            Class.forName("com.mysql.jdbc.Driver");
            conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/sampledb", "root", "");
            System.out.println("DB 연결 완료");
            stmt = conn.createStatement();
            ResultSet srs = stmt.executeQuery("select * from student");
            printData(srs, "name", "id", "dept");
            srs = stmt.executeQuery("select name, id, dept from student where name='이기자'");
            printData(srs, "name", "id", "dept");
        } catch (ClassNotFoundException e) {
            System.out.println("JDBC 드라이버 로드 에러");
        } catch (SQLException e) {
            System.out.println("SQL 실행 에러");
        }
    }
}
```

예제 16-2 : 데이터 검색과 출력(소스 계속)

43

```
private static void printData(ResultSet srs, String col1, String col2, String col3)
    throws SQLException {
    while (srs.next()) {
        if (!col1.equals(""))
            System.out.print(srs.getString("name"));
        if (!col2.equals(""))
            System.out.print("₩t|₩t" + srs.getString("id"));
        if (!col3.equals(""))
            System.out.println("₩t|₩t" + srs.getString("dept"));
        else
            System.out.println();
    }
}
```

DB 연결 완료

이기자 | 0494013 | 컴퓨터공학
김철수 | 1091011 | 컴퓨터시스템
이기자 | 0494013 | 컴퓨터공학

데이터의 변경

44

□ 레코드 추가

```
stmt.executeUpdate("insert into student (name, id, dept) values('아무개',  
                  '0893012', '컴퓨터공학');")
```

- DB에 변경을 가하는 조작은 executeUpdate() 메소드 사용
- SQL문 수행으로 영향을 받은 행의 개수 반환

□ 데이터 수정

```
stmt.executeUpdate("update student set id='0189011' where name='아무개'");
```

□ 레코드 삭제

```
stmt.executeUpdate("delete from student where name='아무개'");
```

예제 16-3 : 데이터의 변경

45

앞서 생성한 sampledb의 student 테이블에 새로운 학생 정보를 추가하고, 새로 생성된 학생의 정보를 수정한 후에 다시 삭제하는 코드를 작성하라. 데이터가 변경될 때마다 모든 테이블의 내용을 출력하도록 하라.

```
import java.io.*;
import java.sql.*;

public class JDBC_Ex3 {
    public static void main (String[] args) {
        Connection conn;
        Statement stmt = null;
        try {
            Class.forName("com.mysql.jdbc.Driver");
            conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/sampledb", "root","");
            System.out.println("DB 연결 완료");
            stmt = conn.createStatement();
            stmt.executeUpdate("insert into student (name, id, dept) values('아무개', '0893012', '컴퓨터공학');");
            printTable(stmt);
            stmt.executeUpdate( "update student set id='0189011' where name='아무개'");
            printTable(stmt);
            stmt.executeUpdate("delete from student where name='아무개'");
            printTable(stmt);
        } catch (ClassNotFoundException e) {
            System.out.println("JDBC 드라이버 로드 에러");
        } catch (SQLException e) {
            System.out.println("SQL 실행 에러");
        }
    }
}
```

예제 16-3 : 데이터의 변경(소스 계속)

46

```
private static void printTable(Statement stmt) throws SQLException {
    ResultSet srs = stmt.executeQuery("select * from student");
    while (srs.next()) {
        System.out.print(srs.getString("name"));
        System.out.print("₩t|₩t" + srs.getString("id"));
        System.out.println("₩t|₩t" + srs.getString("dept"));
    }
}
```

DB 연결 완료

| | | |
|-----|---------|--------|
| 이기자 | 0494013 | 컴퓨터공학 |
| 아무개 | 0893012 | 컴퓨터공학 |
| 김철수 | 1091011 | 컴퓨터시스템 |
| 아무개 | 0189011 | 컴퓨터공학 |
| 이기자 | 0494013 | 컴퓨터공학 |
| 김철수 | 1091011 | 컴퓨터시스템 |
| 이기자 | 0494013 | 컴퓨터공학 |
| 김철수 | 1091011 | 컴퓨터시스템 |