

제 17 장 JDBC 프로그래밍

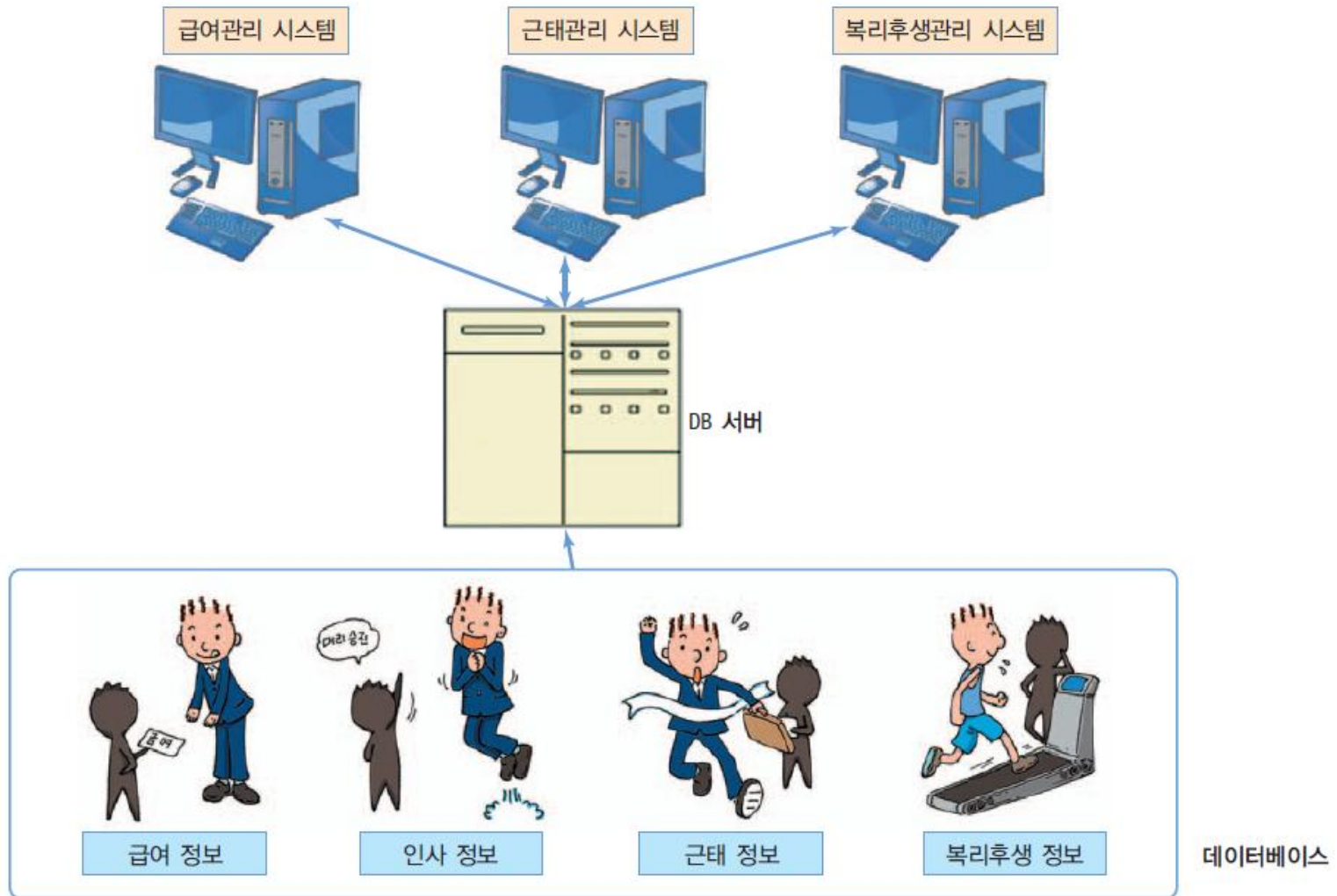
데이터베이스의 개념

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

두 테이블이 동일한
키를 가지고 있음 : 관계

키	값	
employee_id	name	테이블 A
00001	김철수	
00002	최고봉	
00003	이기자	

payroll_id	amount	employee_id	테이블 B
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

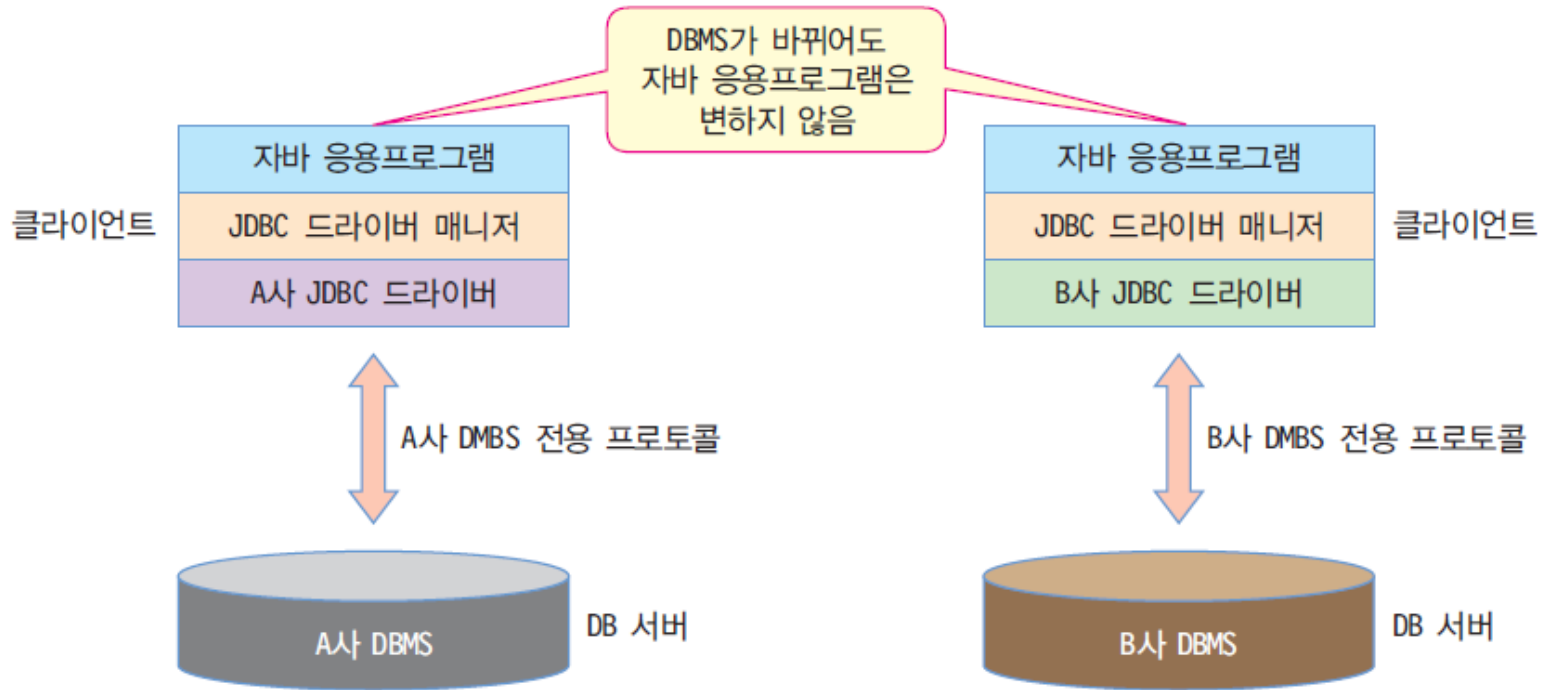
SQL과 JDBC

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



플랫폼 선택 후 다운로드

9

The screenshot shows the MySQL Community Server 5.5.29 download page. The browser address bar shows <http://www.mysql.com/downloads/mysql/>. The page title is "MySQL Community Server 5.5.29".

On the left sidebar, there are contact numbers for various countries: Canada, Germany, France, Italy, UK, Japan, China, India, and Brazil. There are also links for "More Countries »" and "Contact Us Online »".

The main content area has a "Select Platform:" section. The "Microsoft Windows" option is selected and circled in red. An arrow points to this selection with the text "플랫폼 선택" (Platform selection).

Below the platform selection, there is a table of download links. The first row is for "Windows (x86, 32-bit), MSI Installer" (mysql-5.5.29-win32.msi), 5.5.29 version, 31.1M size. The "Download" button for this row is circled in red. An arrow points to this button with the text "다운로드!" (Download!).

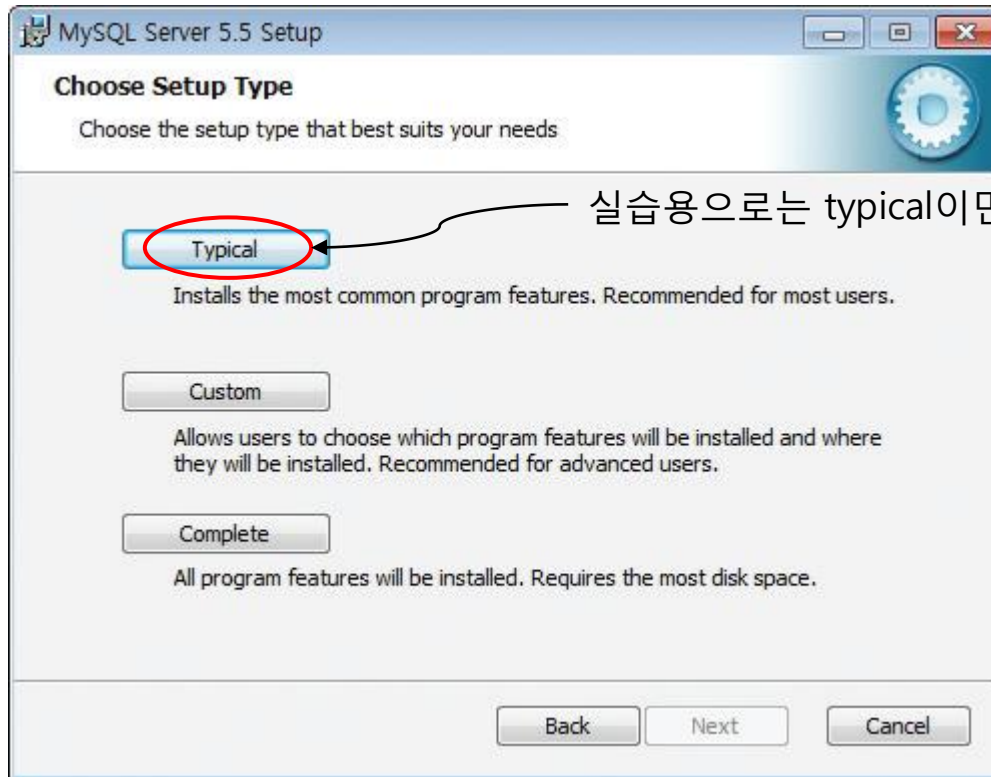
The table also lists other download options: "Windows (x86, 64-bit), MSI Installer", "Windows (x86, 32-bit), ZIP Archive", and "Windows (x86, 64-bit), ZIP Archive". Each row includes the version (5.5.29), size, and a "Download" button.

At the bottom left, there is a "Get Certified MySQL Certification" banner and a "MySQL Enterprise" logo.

Platform	Version	Size	Action
Windows (x86, 32-bit), MSI Installer (mysql-5.5.29-win32.msi)	5.5.29	31.1M	Download
Windows (x86, 64-bit), MSI Installer (mysql-5.5.29-winx64.msi)	5.5.29	32.6M	Download
Windows (x86, 32-bit), ZIP Archive (mysql-5.5.29-win32.zip)	5.5.29	143.4M	Download
Windows (x86, 64-bit), ZIP Archive (mysql-5.5.29-winx64.zip)	5.5.29	145.7M	Download

Typical 타입으로 설치

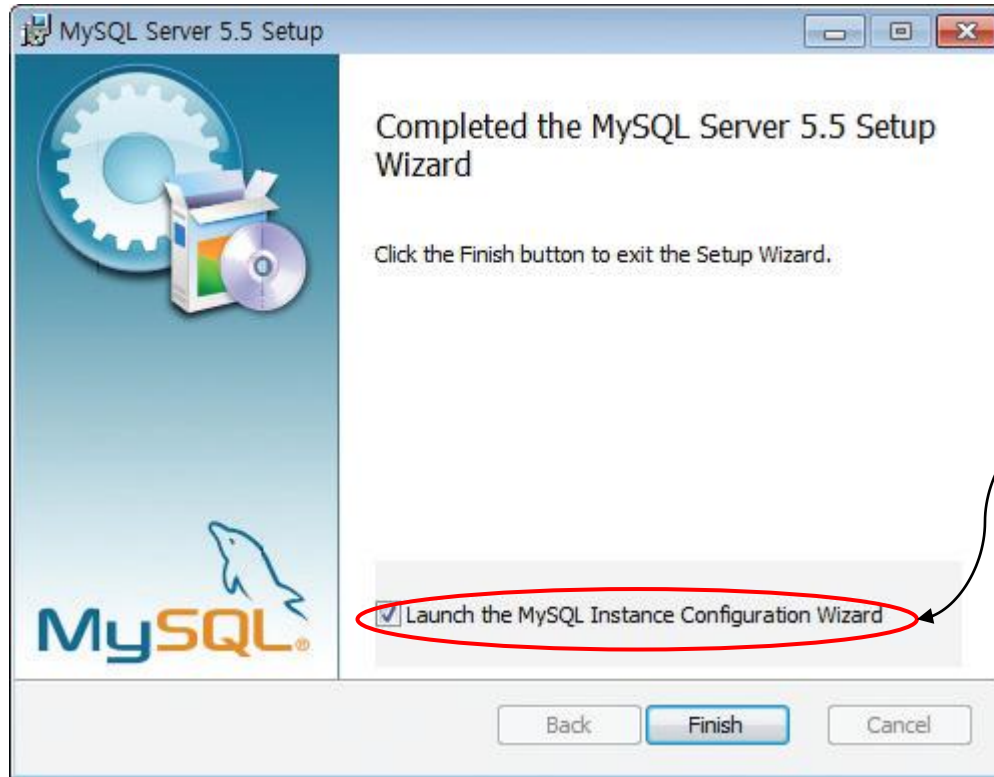
10



실습용으로는 typical이면 충분

설치 완료 및 서버 설정

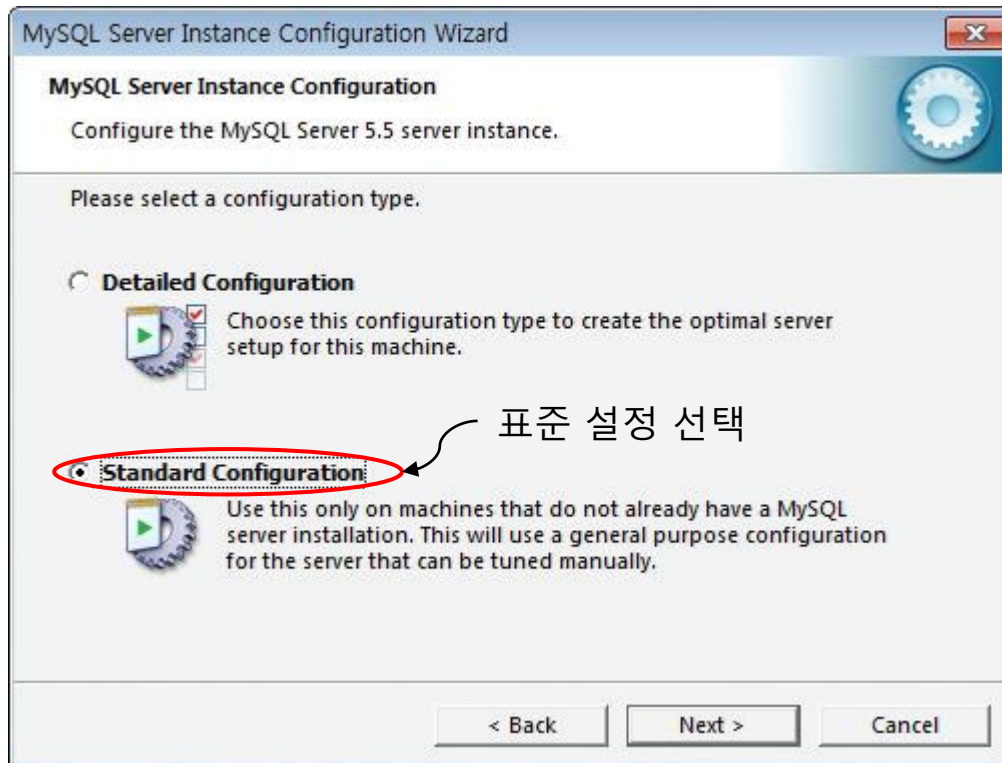
11



서버 설정 선택

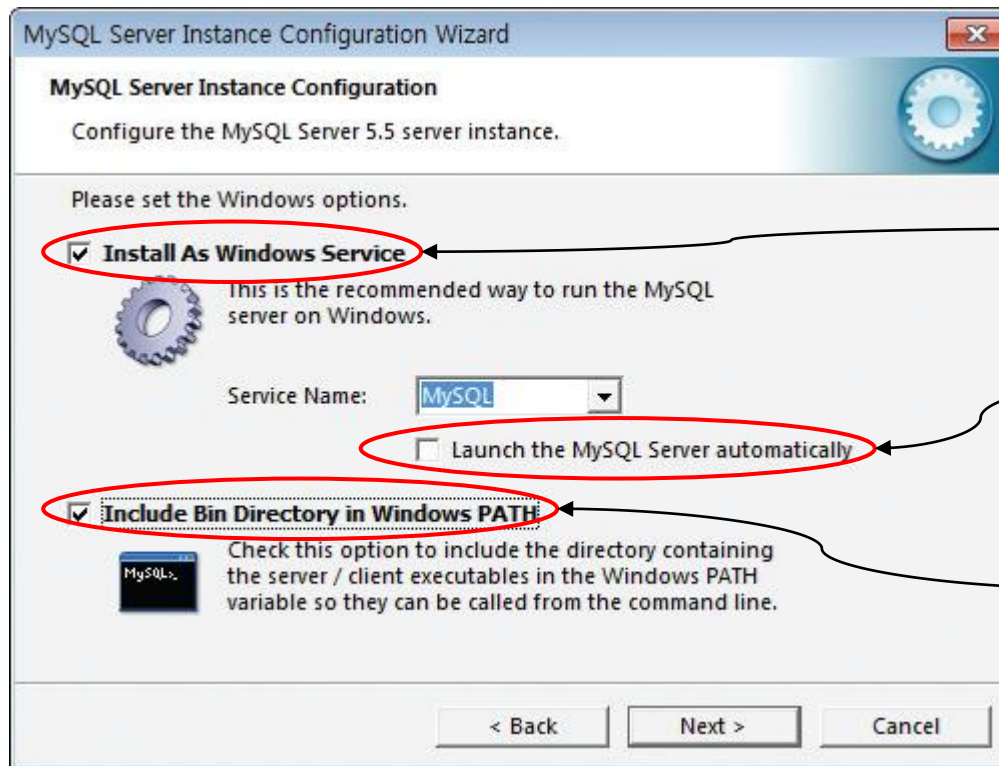
표준 서버 설정

12



서버 인스턴스 설정

13



MySQL서버가 윈도우 서비스로 동작

수동으로 서버 시작 시킴

환경변수에 경로 포함

익명 계정 생성


14

MySQL Server Instance Configuration Wizard

MySQL Server Instance Configuration
Configure the MySQL Server 5.5 server instance.


Please set the security options.

☐ **Modify Security Settings:**

 New root password: Enter the root password.
Confirm: Retype the password.

☐ Enable root access from remote machines

☒ **Create An Anonymous Account** 실습용으로 익명 계정 생성

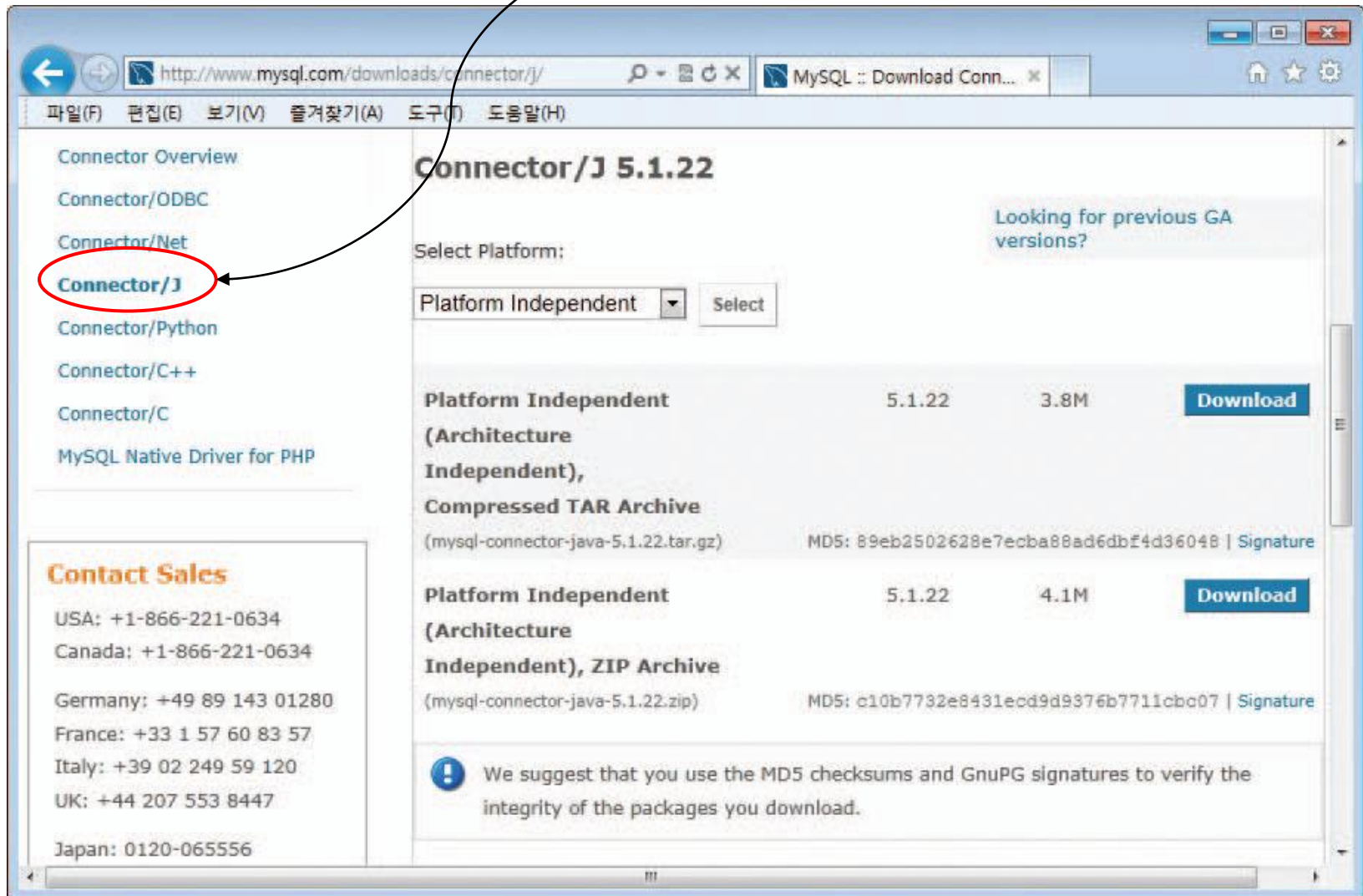
 This option will create an anonymous account on this server. Please note that this can lead to an insecure system.

< Back Next > Cancel

JDBC 드라이버 설치

15

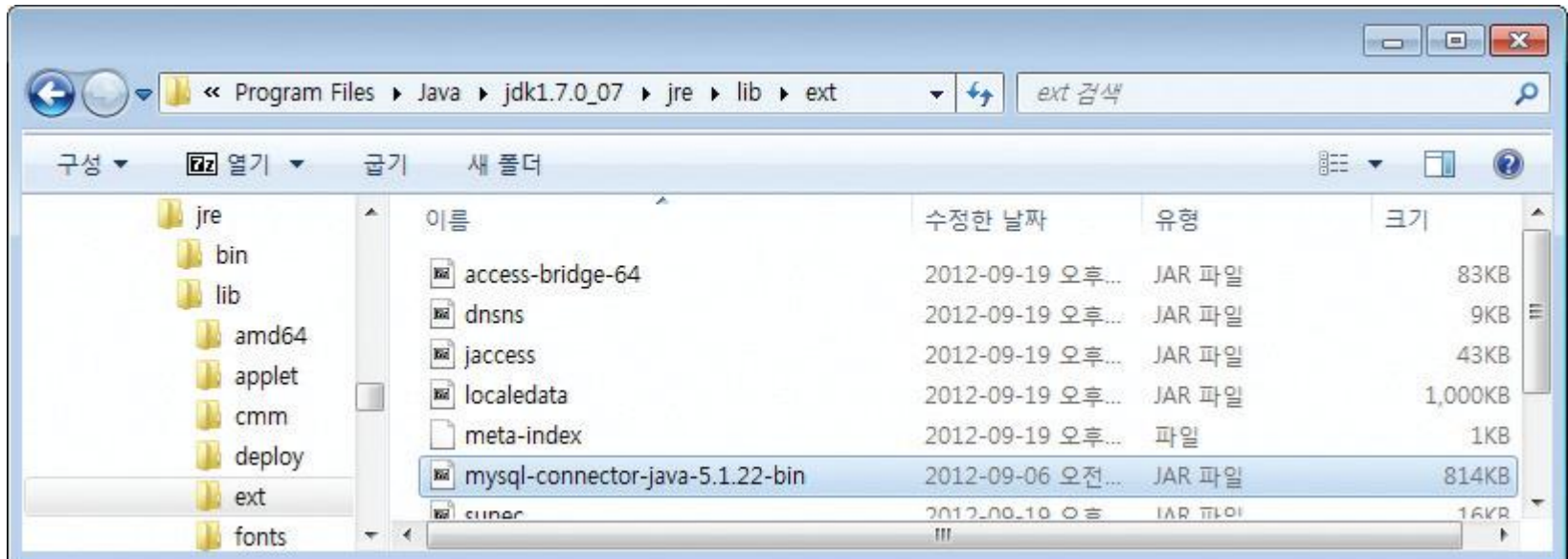
MySQL 다운로드 페이지에
서 Java용 Connector 선택



JDBC 드라이버 설치

16

- Java용 드라이버 복사
 - ▣ ZIP을 풀어 JAR 파일을 JDK 설치 디렉터리 밑의 JRE\LIB\EXT 디렉터리에 복사
 - ▣ JRE만 설치한 경우는 JRE 설치 디렉터리 밑의 LIB\EXT 디렉터리에 복사

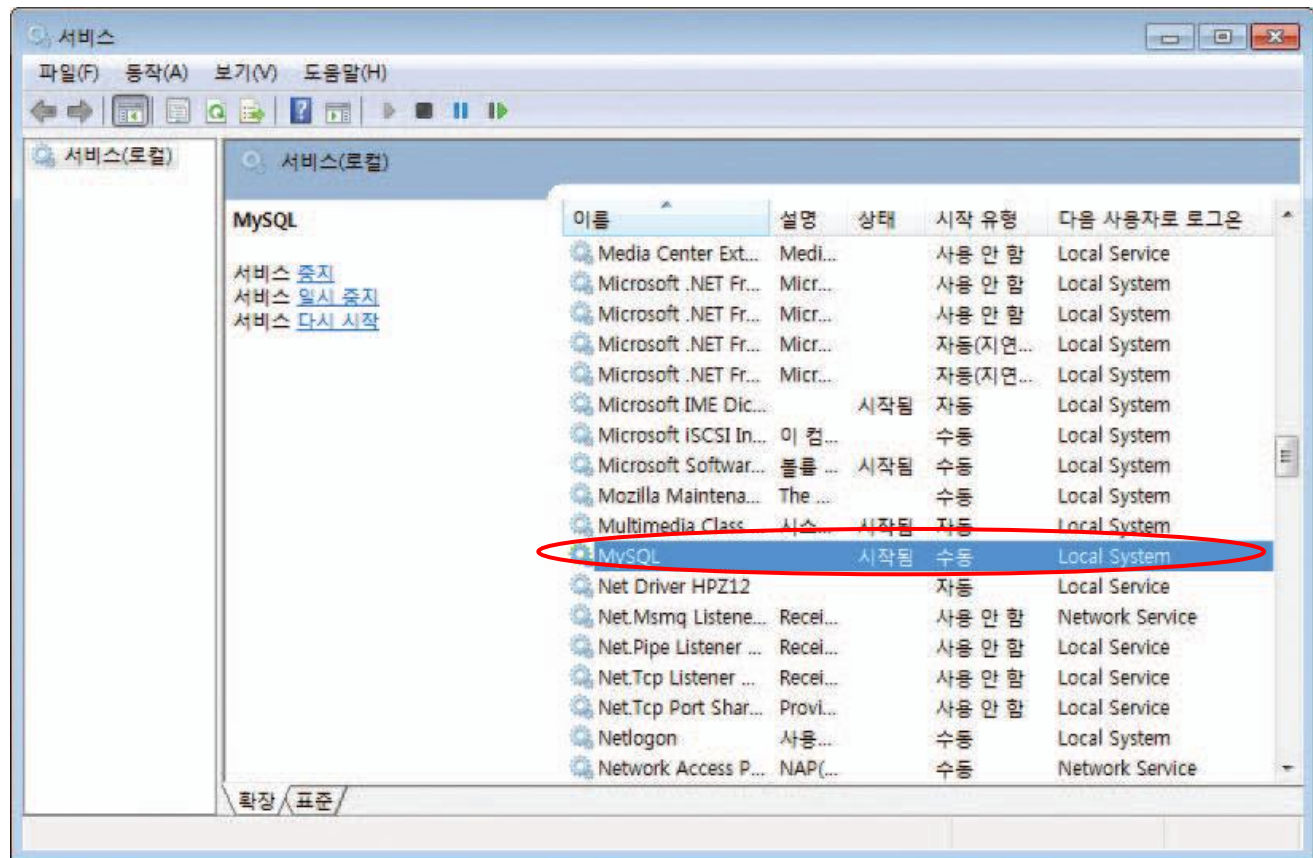


MySQL 서버 실행

17

□ 서비스 관리자 이용

- 윈도우 서비스로 동작하게 설정하였으므로 제어판 시스템 및 보안 관리 도구에서 서비스 관리자를 실행하여 실행 상태 확인
- 서비스 관리자에서 수동으로 실행 시작 및 중지

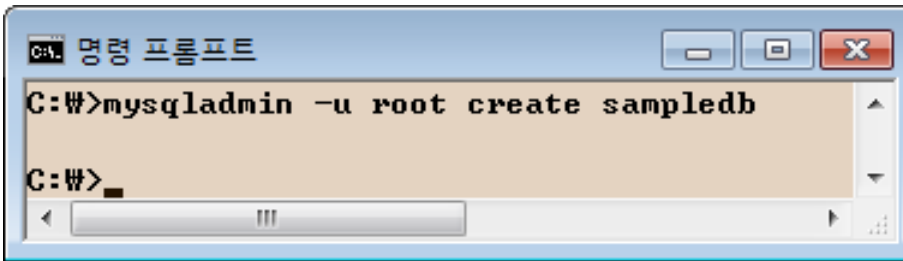


실습용 데이터베이스 생성하기

18

□ 데이터베이스 생성

- ▣ MySQL 설치 디렉터리의 bin\mysqladmin.exe 프로그램 이용



```
C:\>mysqladmin -u root create sampledb  
C:\>
```

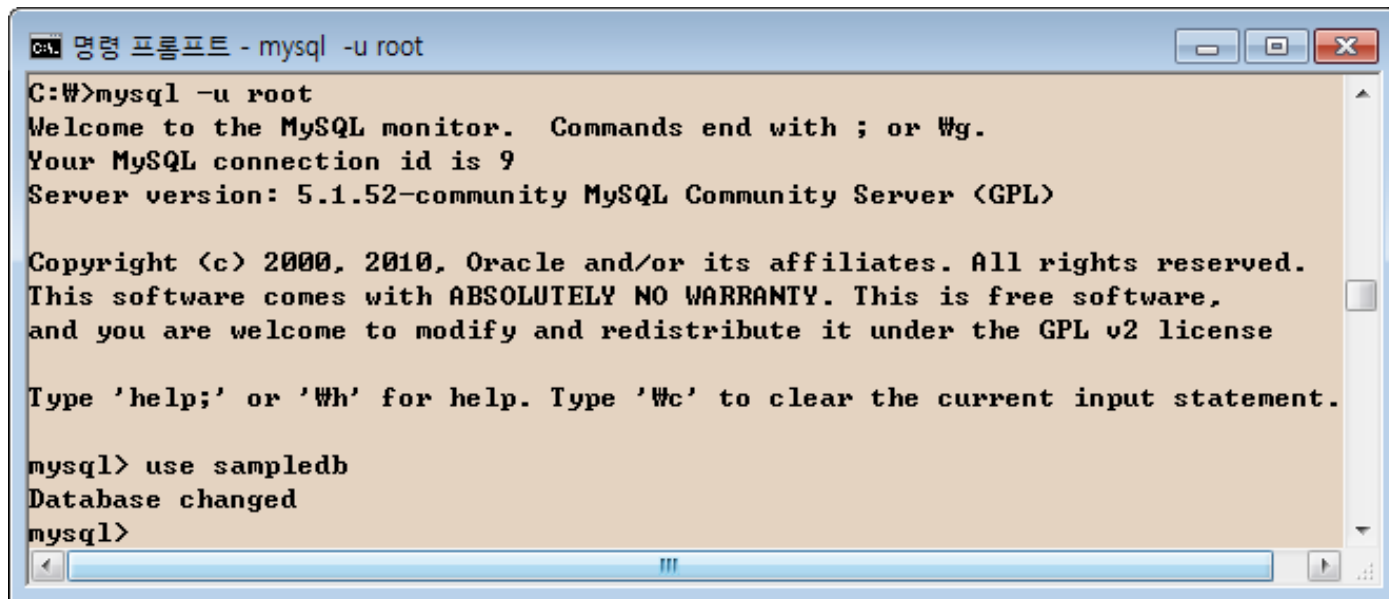
- -u 옵션은 root 계정으로 명령 수행을 의미
- create는 DB 생성 명령
- sampledb는 생성할 DB 이름

데이터 베이스 접속 및 사용

19

□ DB 사용

- ▣ MySQL 설치 디렉터리의 bin\mysql.exe로 생성된 DB에 접속



```
명령 프롬프트 - mysql -u root
C:\>mysql -u root
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 9
Server version: 5.1.52-community MySQL Community Server (GPL)

Copyright (c) 2000, 2010, Oracle and/or its affiliates. All rights reserved.
This software comes with ABSOLUTELY NO WARRANTY. This is free software,
and you are welcome to modify and redistribute it under the GPL v2 license

Type 'help;' or 'Wh' for help. Type 'Wc' to clear the current input statement.

mysql> use sampled
Database changed
mysql>
```

- -u 옵션은 root 계정으로 명령 수행을 의미
- use는 DB 사용 명령
- sampledb는 사용할 DB 이름

테이블 생성

20

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

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

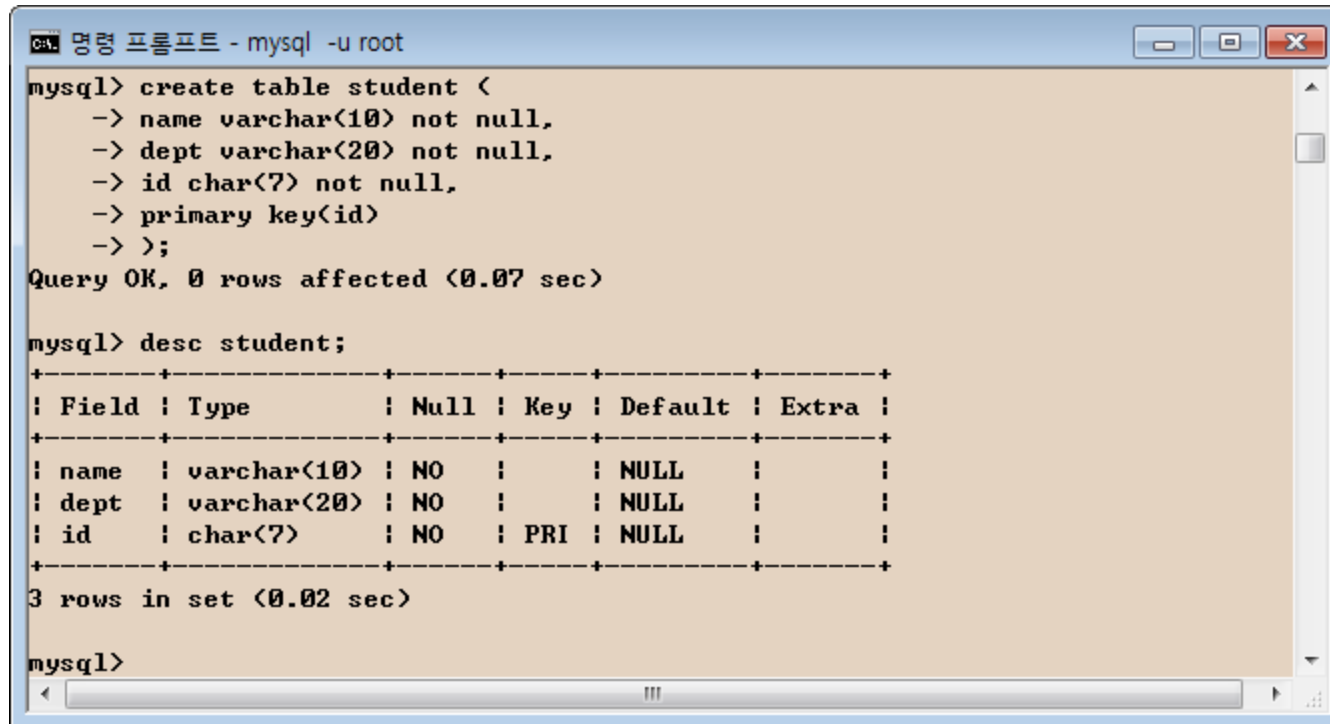
- ▣ name은 varchar 타입으로 10자
 - ▣ dept는 varchar 타입으로 20자
 - ▣ id은 char 타입으로 7자
- 저장할 데이터

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

테이블 생성

21

- create 문으로 테이블 생성



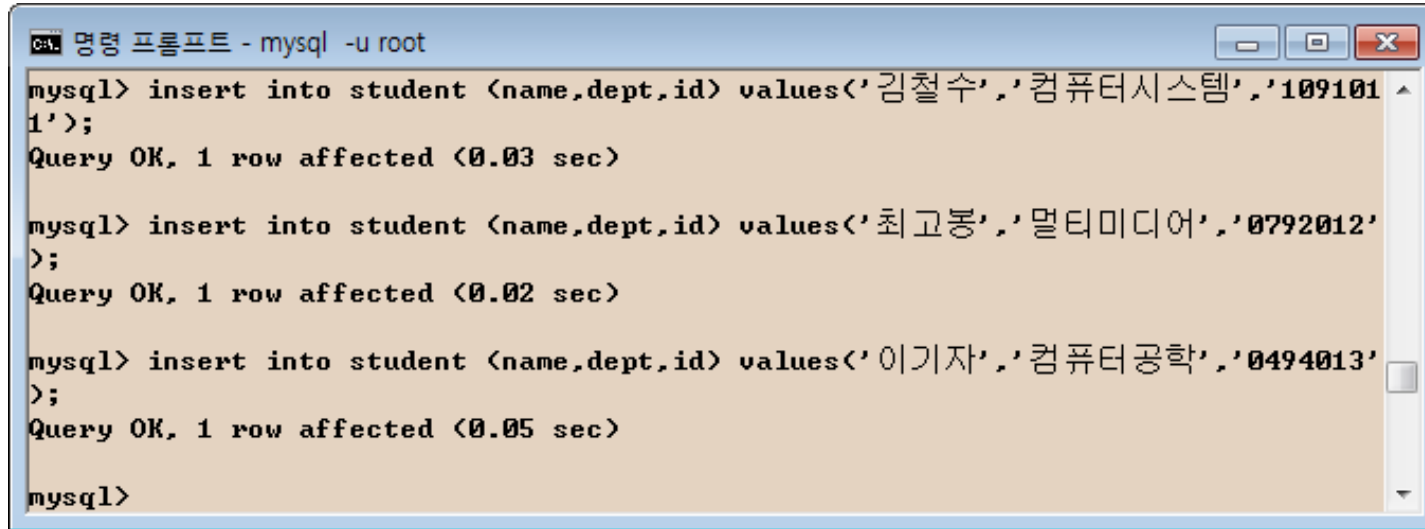
```
mysql> create table student (<br>  -> name varchar(10) not null,<br>  -> dept varchar(20) not null,<br>  -> id char(7) not null,<br>  -> primary key(id)<br>  -> );<br>Query OK, 0 rows affected (0.07 sec)<br><br>mysql> desc student;<br>+-----+<br>| Field | Type          | Null | Key | Default | Extra |<br>+-----+<br>| name  | varchar(10)   | NO   |     | NULL    |      |<br>| dept  | varchar(20)   | NO   |     | NULL    |      |<br>| id    | char(7)       | NO   | PRI | NULL    |      |<br>+-----+<br>3 rows in set (0.02 sec)<br><br>mysql>
```

- create table 다음에 테이블 이름 지정
- 열 이름 데이터 타입(데이터 크기)을 콤마로 분리하여 나열
- not null은 행의 데이터에 null값이 올 수 없음을 의미
- primary key는 키로 사용될 행 지정
- desc 명령은 생성된 테이블의 구조 표시

데이터 추가

22

□ insert 문으로 테이블의 데이터 추가



```
mysql> insert into student (name,dept,id) values('김철수','컴퓨터시스템','1091011');
Query OK, 1 row affected (0.03 sec)

mysql> insert into student (name,dept,id) values('최고봉','멀티미디어','0792012');
Query OK, 1 row affected (0.02 sec)

mysql> insert into student (name,dept,id) values('이기자','컴퓨터공학','0494013');
Query OK, 1 row affected (0.05 sec)

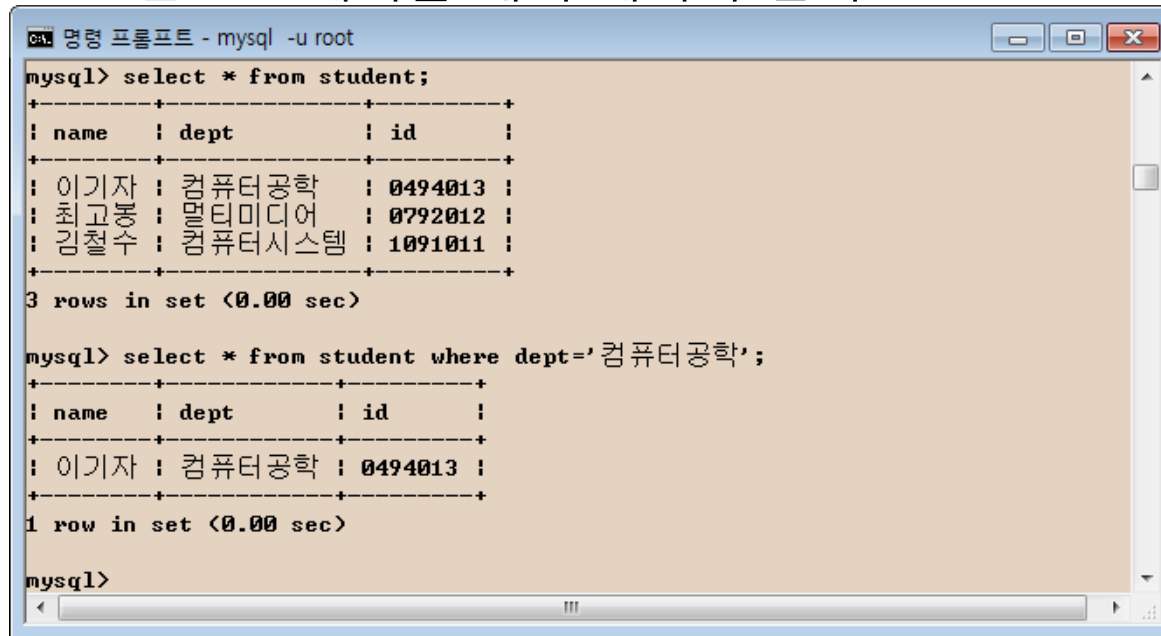
mysql>
```

- insert into 다음에 테이블 이름 지정
- 테이블 이름 다음 괄호 안에 열 이름을 콤마로 구분하여 나열
- values 다음 괄호 안에 열의 값들을 콤마로 구분하여 나열
- 문자 타입의 데이터는 단일 인용 부호로 묶어서 표시함에 유의

데이터 검색

23

□ select문으로 테이블 내의 데이터 검색



```
cmd 명령 프롬프트 - mysql -u root

mysql> select * from student;
+-----+-----+-----+
| name  | dept      | id      |
+-----+-----+-----+
| 이기자 | 컴퓨터공학 | 0494013 |
| 최고봉 | 멀티미디어 | 0792012 |
| 김철수 | 컴퓨터시스템 | 1091011 |
+-----+-----+-----+
3 rows in set (0.00 sec)

mysql> select * from student where dept='컴퓨터공학';
+-----+-----+-----+
| name  | dept      | id      |
+-----+-----+-----+
| 이기자 | 컴퓨터공학 | 0494013 |
+-----+-----+-----+
1 row in set (0.00 sec)

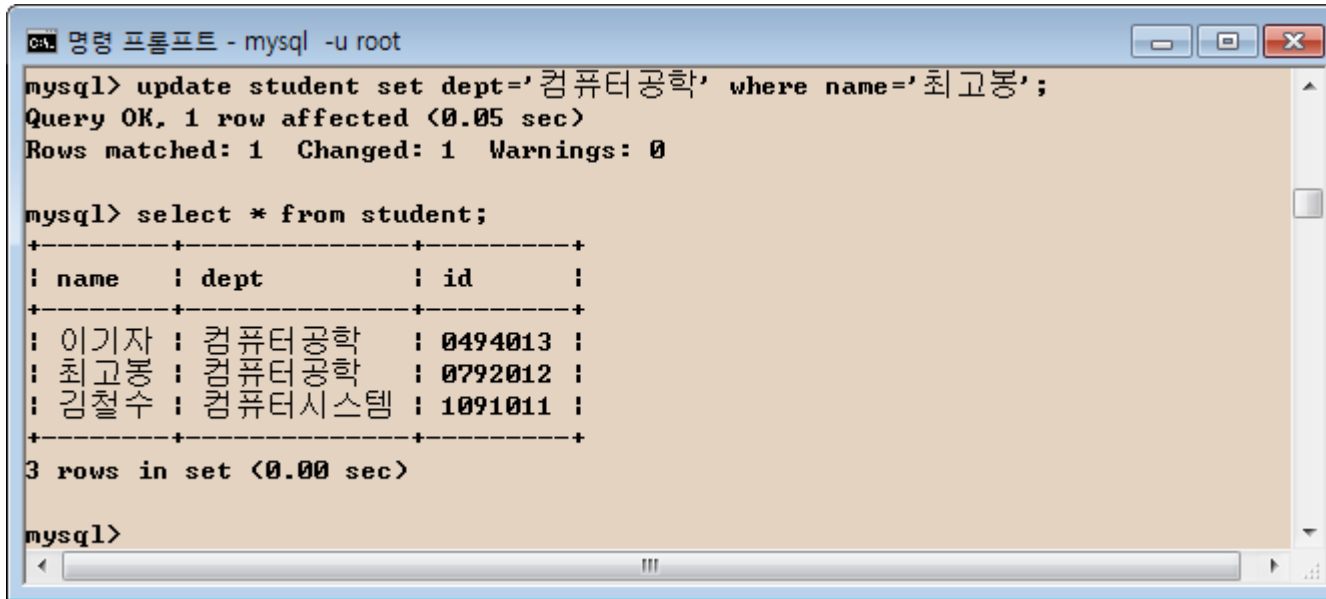
mysql>
```

- 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

데이터 수정

24

- update문을 이용하여 데이터 수정



```
mysql> update student set dept='컴퓨터공학' where name='최고봉';
Query OK, 1 row affected (0.05 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from student;
+-----+-----+-----+
| name  | dept  | id    |
+-----+-----+-----+
| 이기자 | 컴퓨터공학 | 0494013 |
| 최고봉 | 컴퓨터공학 | 0792012 |
| 김철수 | 컴퓨터시스템 | 1091011 |
+-----+-----+-----+
3 rows in set (0.00 sec)

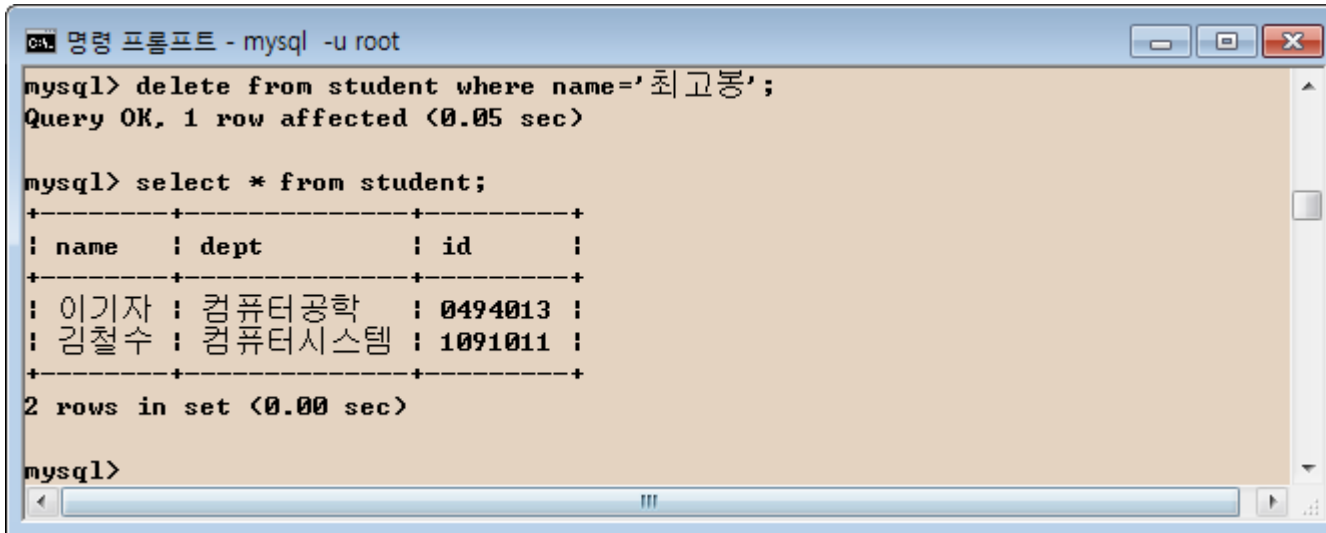
mysql>
```

- 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

레코드 삭제

25

□ delete문을 이용하여 데이터 삭제



```
mysql> delete from student where name='최고봉';
Query OK, 1 row affected (0.05 sec)

mysql> select * from student;
+-----+-----+-----+
| name  | dept  | id    |
+-----+-----+-----+
| 이기자 | 컴퓨터공학 | 0494013 |
| 김철수 | 컴퓨터시스템 | 1091011 |
+-----+-----+-----+
2 rows in set (0.00 sec)

mysql>
```

- delete from 다음에는 테이블 이름 지정
- where 다음에는 검색 조건을 지정. 위의 예에서는 name 값이 '최고봉'인 레코드 삭제
- where는 생략 가능

JDBC 프로그래밍

26

□ DB 연결 설정

▣ 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 DriverManager에 등록
- 해당 드라이버가 없으면 ClassNotFoundException 발생

자바 응용프로그램과 JDBC의 연결

27

▣ 연결

```
try {  
    Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/sampledб", "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 localhost로 지정
 - MySQL의 경우 디폴트로 3306 포트를 사용
 - sampledб는 앞서 생성한 DB의 이름
- "root" 는 DB에 로그인할 계정 이름이며, "" 는 계정 패스워드

예제 17-1 : sampledb의 데이터베이스 연결하는 JDBC 프로그램 작성

28

JDBC를 이용하여 sampledb 데이터베이스에 연결하는 자바 응용프로그램을 작성하라.

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

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/sampledб", "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

29

□ 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, SQL문 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

30

- 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 행으로 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 이름에 해당하는 데이터를 반환. 예를 들어, int형 데이터를 읽는 메소드는 getInt()이다.
Xxx getXxx(int columnIndex)	Xxx는 해당 데이터 타입을 나타내며 현재 행에서 지정된 열 인덱스에 해당하는 데이터를 반환. 예를 들어, int형 데이터를 읽는 메소드는 getInt()이다.

데이터베이스 사용

31

□ 테이블의 모든 데이터 검색

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

- Statement의 executeQuery()는 SQL문의 실행하여 실행 결과를 넘겨줌
- 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

32

□ 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();
```

□ Statement객체의 executeQuery() 메소드

- ResultSet 객체 반환

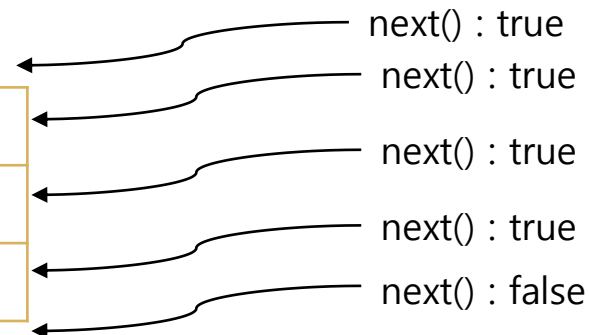
□ ResultSet 인터페이스

- DB에서 읽어온 데이터를 추출 및 조작할 수 있는 방법 제공

□ next() 메소드

- 다음 행으로 이동

김철수	컴퓨터시스템	1091011
최고봉	멀티미디어	0792012
이기자	컴퓨터공학	0494013



데이터베이스 사용

33

- ▣ ResultSet의 getXXX() 메소드
 - 해당 데이터 타입으로 열 값을 읽어옴
 - 인자로 열의 이름이나 인덱스를 줄 수 있음
 - DB의 데이터 타입에 해당하는 자바 데이터 타입으로 데이터를 읽어야 함.
 - 모든 데이터 타입에 대해 getString() 메소드로 읽을 수 있으나 사용할 때는 해당 데이터 타입으로 변환해서 사용
- ▣ ResultSet에서 모든 데이터를 다 읽어들이면 후에는 close()를 호출하여 자원 해제

한글 처리 문제

34

□ 문자열 코드 문제

▣ MySQL의 문자 집합은 ISO-8859-1

- Unicode를 사용하는 자바에서 한글이 깨져 출력
- ISO-8859-1를 Unicode로 변환하여 출력해야 함

```
String ISO_8859_1String = rs.getString("name");  
byte [] UnicodeBytes = ISO_8859_1String.getBytes("ISO-8859-1");  
String UnicodeString = new String(UnicodeBytes);  
System.out.print(UnicodeString);
```

유니코드로 변환

- 반대로 MySQL에서 사용될 한글이 포함된 문자열은 Unicode에서 ISO-8859-1로 변환해야 MySQL에서 정상적으로 처리됨

```
byte [] UnicodeBytes = "홍길동".getBytes();  
String IOS_8859_1String = new String(UnicodeBytes, "ISO-8859-1");  
stmt.executeQuery("select name, id, dept from student where name='"+ IOS_8859_1String + "'");
```

유니코드를 ISO-8859-1로 변환

예제 17-2 : 데이터 검색과 출력

35

앞서 생성한 `sampledb`의 `student` 테이블의 모든 데이터를 출력하는 프로그램과 이름이 "이기자" 인 학생의 데이터를 출력하는 프로그램을 작성하시오.

```
import java.io.UnsupportedEncodingException;
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='"+
                new String("이기자".getBytes(), "ISO-8859-1") + "'");
            printData(srs, "name", "id", "dept");
        } catch (ClassNotFoundException e) {
            System.out.println("JDBC 드라이버 로드 에러");
        } catch (SQLException e) {
            System.out.println("SQL 실행 에러");
        } catch (UnsupportedEncodingException e) {
            System.out.println("지원되지 않는 인코딩 타입");
        }
    }
}
```

예제 17-2 : 데이터 검색과 출력(소스 계속)

36

```
private static void printData(ResultSet srs, String col1, String col2, String col3)
    throws UnsupportedEncodingException, SQLException {
    while (srs.next()) {
        if (!col1.equals(""))
            System.out.print(new String(srs.getString("name").getBytes("ISO-8859-1")));
        if (!col2.equals(""))
            System.out.print("\t\t" + srs.getString("id"));
        if (!col3.equals(""))
            System.out.println("\t\t" + new String(srs.getString("dept").getBytes("ISO-8859-1")));
        else
            System.out.println();
    }
}
```

DB 연결 완료

이기자		0494013		컴퓨터공학
김철수		1091011		컴퓨터시스템
이기자		0494013		컴퓨터공학

데이터의 변경

37

□ 레코드 추가

```
stmt.executeUpdate("insert into student (name, id, dept) values('" +  
    new String("아무개".getBytes(), "ISO-8859-1") +  
    "', '0893012', '" +  
    new String("컴퓨터공학".getBytes(), "ISO-8859-1") + "');");
```

- 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='" +  
    new String("아무개".getBytes(), "ISO-8859-1") + "');");
```

- where문의 MySQL에서 처리되므로 문자열을 Unicode에서 ISO-8859-1로 변환에 주의

□ 데이터 삭제

```
stmt.executeUpdate("delete from student where name='" +  
    new String("아무개".getBytes(), "ISO-8859-1") + "');");
```

예제 17-3 : 데이터의 변경

38

앞서 생성한 `sampledb`의 `student` 테이블에 새로운 학생 정보를 추가하고, 새로 생성된 학생의 정보를 수정한 후에 다시 삭제하는 코드를 작성하시오. 데이터가 변경될 때마다 모든 테이블의 내용을 출력하도록 하시오.

```
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('" + new String("아무개".getBytes(), "ISO-8859-1")
                + "', '0893012', '" + new String("컴퓨터공학".getBytes(), "ISO-8859-1") + "');");
            printTable(stmt);
            stmt.executeUpdate("update student set id='0189011' where name='" + new String("아무개".getBytes(), "ISO-8859-1") + "'");
            printTable(stmt);
            stmt.executeUpdate("delete from student where name='" + new String("아무개".getBytes(), "ISO-8859-1") + "'");
            printTable(stmt);
        } catch (ClassNotFoundException e) {
            System.out.println("JDBC 드라이버 로드 에러");
        } catch (SQLException e) {
            System.out.println("SQL 실행 에러");
        } catch (UnsupportedEncodingException e) {
            System.out.println("지원되지 않는 인코딩 타입");
        }
    }
}
```

예제 17-3 : 데이터의 변경(소스 계속)

39

```
private static void printTable(Statement stmt) throws SQLException, UnsupportedEncodingException {
    ResultSet srs = stmt.executeQuery("select * from student");
    while (srs.next()) {
        System.out.print(new String(srs.getString("name").getBytes("ISO-8859-1")));
        System.out.print("WtWt" + srs.getString("id"));
        System.out.println("WtWt" + new String(srs.getString("dept").getBytes("ISO-8859-1")));
    }
}
```

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