



# 프로젝트로 배우는 자바 웹 프로그래밍

Servlet, JSP, JDBC

황희정 지음

## Chapter 08. 데이터베이스와 JDBC

# 목차

1. 데이터베이스 개요
2. MySQL 데이터베이스 설치와 설정
3. SQL 문 기본기 다지기
4. JDBC 기본구조와 API 이해
5. [기본실습] JDBC 프로그래밍 : MySQL 연동 JSP 프로그래밍

# 학습목표

- 데이터베이스의 기본 개념을 이해한다.
- 기본적인 SQL문을 익힌다.
- MySQL 데이터베이스를 설치하고 기본적인 사용 방법을 익힌다.
- 자바에서 데이터베이스 프로그램에 필요한 JDBC의 사용 방법을 익힌다.

## 1. 데이터베이스란?

- 데이터베이스는 데이터를 체계적으로 관리할 수 있도록 해주는 소프트웨어로 일반적으로 다음과 같이 정의 한다.

- **데이터베이스(Database)** : 여러 사람이 공유할 목적으로 방대한 데이터를 체계적으로 정리하여 저장한 것으로 이를 이용하면 데이터를 효율적으로 관리하고 검색할 수 있다.
- **데이터베이스 관리 시스템(DBMS, DataBase Management System)** : 데이터베이스를 구성하고 운영하는 소프트웨어 시스템으로, 오라클, MS SQL 서버, MySQL 등 일반적으로 알고 있는 데이터베이스 제품을 의미한다.

- 예를 들어 휴대폰에 저장되어 있는 전화번호 자체는 데이터베이스로 볼 수 있으며 메뉴에서 이름을 검색하고 저장을 가능하게 하는 내부적인 프로그램은 DBMS 로 볼 수 있다.

# 01. 데이터베이스 개요

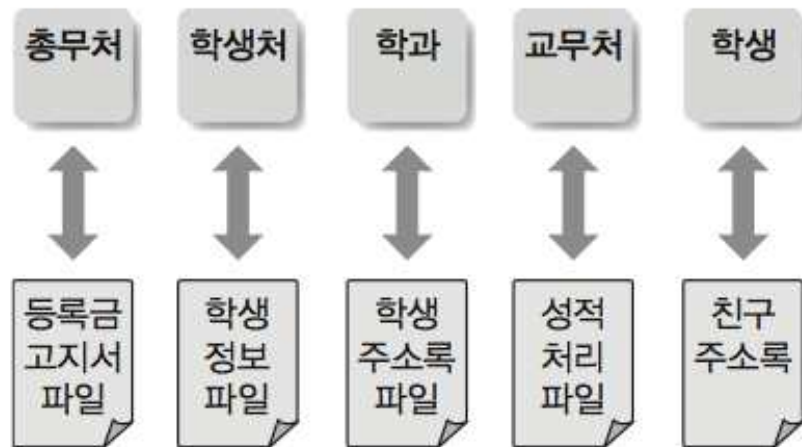
[표 8-1] 데이터베이스의 종류

종류	장점	단점
오라클	<ul style="list-style-type: none"> <li>• 사용층이 가장 넓다.</li> <li>• 제품의 우수성이 입증되었다.</li> <li>• PC는 물론 대형 서버까지 설치할 수 있다.</li> <li>• 공급 업체의 강력한 지원을 받는다.</li> <li>• 분산 처리를 지원한다.</li> <li>• 개발자를 위한 Express Edition을 제공하여, 개발자들이 더욱 손쉽게 접할 수 있다.</li> </ul>	<ul style="list-style-type: none"> <li>• DBMS를 운영하려면 많은 하드웨어 자원이 필요하다.</li> <li>• DBMS 관리가 복잡하다.</li> <li>• 동종 DBMS보다 가격이 비싸다.</li> </ul>
MySQL	<ul style="list-style-type: none"> <li>• 개발자들에게 공개된 라이선스 및 저렴한 가격으로 중소 규모 서비스에서 쉽게 도입할 수 있다.</li> <li>• 지속적인 성능 향상으로 많은 부분에서 대형 RDBMS에 사용해도 손색이 없다.</li> <li>• 오라클에 인수되었으며 상용 라이선스 및 기술 지원을 받을 수 있다.</li> </ul>	<ul style="list-style-type: none"> <li>• 기술지원 및 A/S를 받으려면 상용 라이선스가 필요하다.</li> <li>• 상용 데이터베이스보다는 성능이 떨어질 것이라는 막연한 인식이 팽배하다.</li> <li>• 대형 데이터베이스 관리 지원은 다소 부족하다.</li> <li>• 오라클에서 인수한 후 발전이 둔화되었다.</li> <li>• MariaDB로의 이전이 가속화되고 있다.</li> </ul>
MS SQL 서버	<ul style="list-style-type: none"> <li>• 사이베이스 장점을 계승했다.</li> <li>• 비교적 초기 도입 비용이 저렴하다.</li> <li>• 윈도우 서버 환경에서 최적화되었다.</li> <li>• 닷넷(.Net) 개발 플랫폼과 통합되었다.</li> </ul>	<ul style="list-style-type: none"> <li>• 윈도우 서버 운영체제에서만 동작한다.</li> <li>• 시스템을 확장할 때 라이선스 비용이 상승한다.</li> </ul>
IBM DB2	<ul style="list-style-type: none"> <li>• IBM 제품과의 호환성이 뛰어나다.</li> <li>• 비정형 데이터를 관리할 수 있다.</li> <li>• PureXML 기술을 사용한다.</li> <li>• Venom 스토리지 압축 기술을 사용한다.</li> </ul>	<ul style="list-style-type: none"> <li>• 호환성이 제한된다.</li> <li>• 공급 업체의 지원 도구가 부족하다.</li> <li>• IBM 위주로 시장이 편중되어 있다.</li> </ul>
Apache Derby	<ul style="list-style-type: none"> <li>• 100% 순수 자바 기반의 데이터베이스다.</li> <li>• 애플리케이션 임베디드 데이터베이스로, 별도의 서버 실행 과정 없이 간단하게 사용할 수 있다.</li> <li>• 애플리케이션의 테스트와 배포에 효율적이다.</li> </ul>	<ul style="list-style-type: none"> <li>• 일반적인 데이터 서비스 환경에는 부적합하다.</li> </ul>

# 01. 데이터베이스 개요

## 2. 데이터베이스의 장단점

- 데이터를 사용하고 처리하는 데 있어 가장 중요한 기술적 관점은 데이터를 저장하고 이를 이용하는 방법이다.
- 일반적으로 데이터를 저장한 파일을 데이터 파일(Data File)이라고 하는데, 데이터 파일은 서로 연관된 정보를 모아서 레코드(Record)를 만들고 레코드는 다시 연관 레코드를 모아서 파일을 만든다.
- 이렇게 만들어진 데이터 파일들은 운영체제의 파일 시스템에서 관리한다.
- 하지만 일반적인 파일 시스템은 응용 분야에 따라 동일한 정보를 여러 파일에 중복 저장하는 문제가 있다.

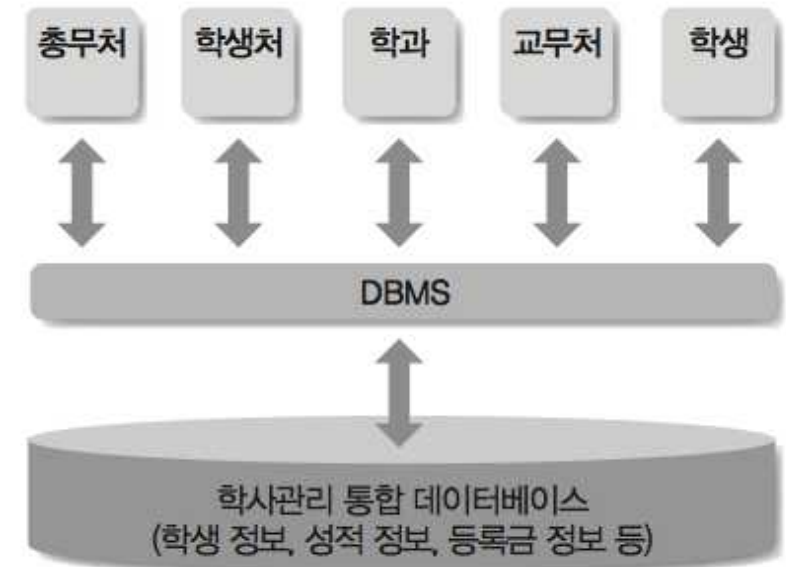


[그림 8-1] 파일 시스템

# 01. 데이터베이스 개요

## ■ 데이터베이스 이점

- 데이터 중복을 최소화할 수 있다.
- 데이터 불일치 문제를 해결할 수 있다.
- 데이터를 쉽게 공유할 수 있다.
- 정보 표준화를 이룰 수 있다.
- 데이터에 대한 보안성을 제공한다.
- 데이터의 무결성(Integrity)이 유지된다.
- 대량의 데이터를 좀더 빠르게 검색할 수 있다.
- 텍스트 이외의 다양한 데이터(이미지, 파일 등)를 관리할 수 있다.
- 애플리케이션을 개발하기가 쉽다.



[그림 8-2] 데이터베이스 시스템

# 01. 데이터베이스 개요

## ■ 데이터베이스 도입의 추가적인 요구사항

- DBMS를 위한 하드웨어(서버 장비, 하드디스크)가 추가적으로 필요하며, 이러한 하드웨어는 데이터가 증가되면 지속적으로 추가해야 하므로 비용이 많이 든다.
- 데이터베이스를 관리하는 DBA가 필요하다.
- 데이터 백업 및 복구와 관련한 전문 기술이 필요하며, 데이터 백업을 위한 비용이 든다.



## 3. 데이터베이스의 분류

- 파일형(파일 시스템)
- 세그먼트형(계층형 데이터베이스 관리 시스템 : HDBMS)
- 레코드형(네트워크형 데이터베이스 관리 시스템 : NDB)
- 테이블형(관계형 데이터베이스 관리 시스템 : RDBMS)
- 클래스형(객체지향 데이터베이스 관리 시스템 : OO/ORDB)

### ■ 관계형 데이터베이스(RDB)

- 데이터를 효율적으로 관리하려고 데이터에 관계 개념을 부여한 것으로, 대부분의 데이터 베이스가 관계형 데이터베이스에 기반을 두고 있다. 데이터의 기본 관리 단위는 테이블(Table)이고, 칼럼(Column)과 로우(Row)로 구성되어 있다. 관계라는 의미는 테이블 간 의 연관 관계를 말하는 것으로, 관계형 데이터베이스에서는 주 키와 외래 키 등을 통해 테이블 간의 관계를 기술하고 구조화한다.

### ■ 객체지향 데이터베이스(OODB)

- 객체지향 개념을 데이터베이스에 도입한 것이다. 객체지향의 일반적인 개념인 클래스, 객체, 애트리뷰트, 메서드, 인스턴스, 캡슐화, 상속 등을 기반으로 데이터를 구조화하는 시스템이다. 프로그램 언어에서와는 달리 '표준화 부재', '관리 시스템의 복잡성', '질의 최적화의 복잡성'과 같은 여러 문제로 관계형 데이터베이스를 대체하지 못했다. 순수한 객체지향 데이터베이스 개념보다는 객체 관계형 데이터베이스가 기존 관계형 데이터베이스를 확장하는 형태로 성장하게 되었다.

# 01. 데이터베이스 개요

## 4. 데이터베이스의 구성 요소

### ■ 데이터베이스 테이블(Table)

- 테이블은 관계형 데이터베이스에서 가장 기본이 되는 데이터 관리 단위로, 테이블간의 관계 표현을 통해 효과적인 데이터 관리 방법을 제공한다.

홍길동, 서울, 1992, 02-123-1234, 남  
강동수, 남, 인천, 1993, 032-123-1111  
대구, 홍길동, 여, 1991, 010-111-2222  
이미녀, 1992, 여, 서울, 02-222-3333

(a) 정리되지 않은 형태

칼럼					칼럼 이름
이름	성별	거주지	출생년도	전화번호	
홍길동	남	서울	1992	02-123-1234	로우
강동수	남	인천	1993	032-123-1111	
홍길동	여	대구	1991	010-111-2222	
이미녀	여	서울	1992	02-222-3333	

(b) 정리된 형태

[그림 8-3] 개인 정보의 예

# 01. 데이터베이스 개요

## ■ 데이터베이스 테이블(Table)

- **테이블** : 데이터를 공통 속성으로 묶고 분류하여 기록한 형태로 데이터베이스 관리의 기본이다.
  - 예) 학생 정보 테이블(member)
  - **칼럼** : 테이블에서 이름, 성별, 거주지, 출생연도, 전화번호 등 데이터를 구별하기 위한 속성이다.  
칼럼(Column) 또는 필드(Field)라고 한다.
  - 예) 이름(name), 성별(sex), 거주지(city), 출생연도(birth), 전화번호(tel)
  - **로우** : 한 줄 단위의 데이터 집합이다. 로우(Row) 혹은 레코드(Record)라고 한다.
  - 예) [그림 8-3(b)]의 첫째 로우 : 홍길동, 남, 서울, 1992, 02-123-1234
- 
- 테이블 구성 요소의 특징은 다음과 같다.
    - 칼럼이나 로우의 위치와 순서는 아무런 의미가 없다.
    - 로우는 데이터 하나만 표시할 수 있고, 그룹이나 배열은 허용하지 않는다.
    - 각 칼럼은 특정한 형태의 값, 각 로우 데이터는 해당 칼럼에서 요구하는 형태 값만 포함 할 수 있다.

# 01. 데이터베이스 개요

## ■ 데이터베이스 키(Key)

- 데이터베이스 키는 데이터를 서로 구분하기 위한 특성을 지닌 값을 말함.
- 키(Key)
  - 데이터베이스에서는 데이터를 다른 데이터와 구분할 수 있는 고유 정보가 필요한데, 이를 키라고 한다. 키는 관계형 데이터베이스의 대표적인 특징 중 하나다. 키는 다시 주 키(Primary Key), 보조 키(Secondary Key), 후보 키(Candidate Key) 등으로 나눌 수 있다.
- 주 키(Primary Key)
  - 테이블 하나에서 키 여러 개를 가질 수 있지만, 그중 절대적으로 구분되는 키를 주 키라고 한다.
  - 즉, 테이블 하나 당 하나만 존재하며, 각각의 로우를 구분해주는 값이다.
  - 주 키는 학번, 사번, 주민등록번호 등 중복되지 않는 항목을 사용할 수 있으나 단순히 데이터 구분을 위한 주 키의 경우 특별한 의미가 없는 중복되지 않고 순차적으로 증가하는 숫자값을 키로 사용하면 된다.
  - 앞의 학생 개인정보 테이블의 예에서는 학번을 주 키로 설정 할 수 있다.

주 키

학번	이름	성별	거주지	출생년도	전화번호
201301	홍길동	남	서울	1992	02-123-1234
201302	강동수	남	인천	1993	032-123-1111
201303	홍길동	여	대구	1991	010-111-2222
201304	이미녀	여	서울	1992	02-222-3333

[그림 8-4] 주 키를 포함한 학생 정보 테이블

# 01. 데이터베이스 개요

## ■ 외래 키(Foreign Key)

- 테이블간의 관계를 나타내고 데이터의 일관성 유지를 위해 사용 한다.
- 예를 들어 거주지의 경우 특별한 제약이 없다면 "서울", "서 울", "서울특별시" 로 입력할 수 있어 만일 "서울"만 검색 한다면 원하는 결과를 얻을수 없게 된다.
- 이 때 지역코드 테이블을 두고 외래 키로 연결 한다면 이런 문제를 해결 할 수 있다.

주 키

지역 코드	지역
1	서울
2	인천
3	부산
4	대구
...	...

[그림 8-5] 지역 코드 테이블

외래 키

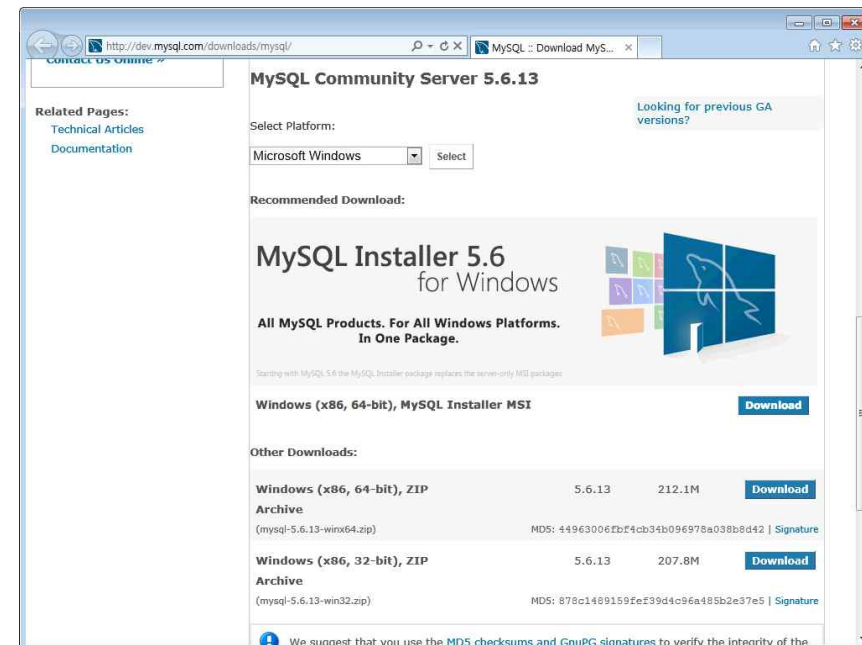
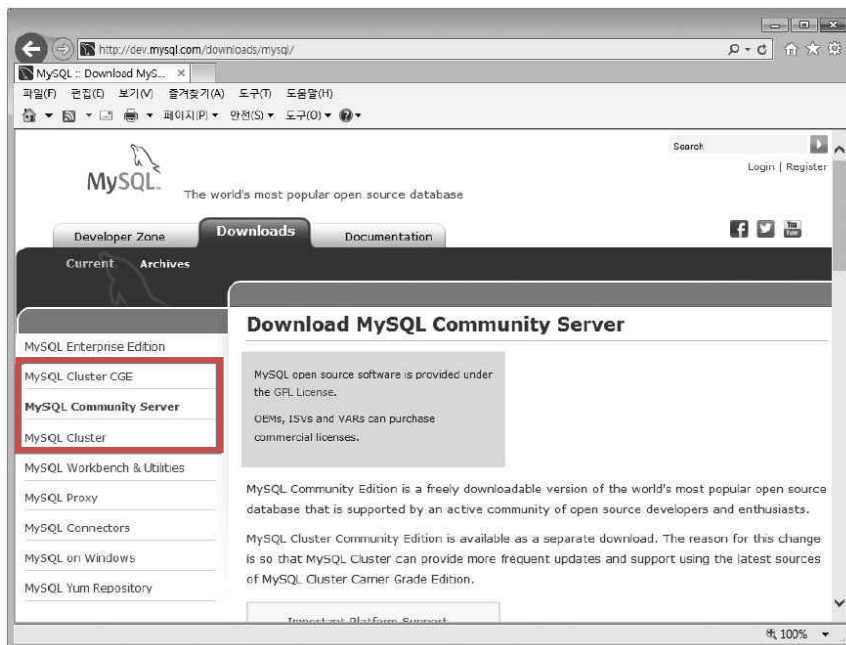
학번	이름	성별	거주지	출생년도	전화번호
201301	홍길동	남	1	1992	02-123-1234
201302	강동수	남	2	1993	032-123-1111
201303	홍길동	여	4	1991	010-111-2222
201304	이미녀	여	1	1992	02-222-3333

[그림 8-6] 수정된 학생 정보 테이블

## 02. MySQL 데이터베이스 설치와 설정

### 1. MySQL 데이터베이스 설치

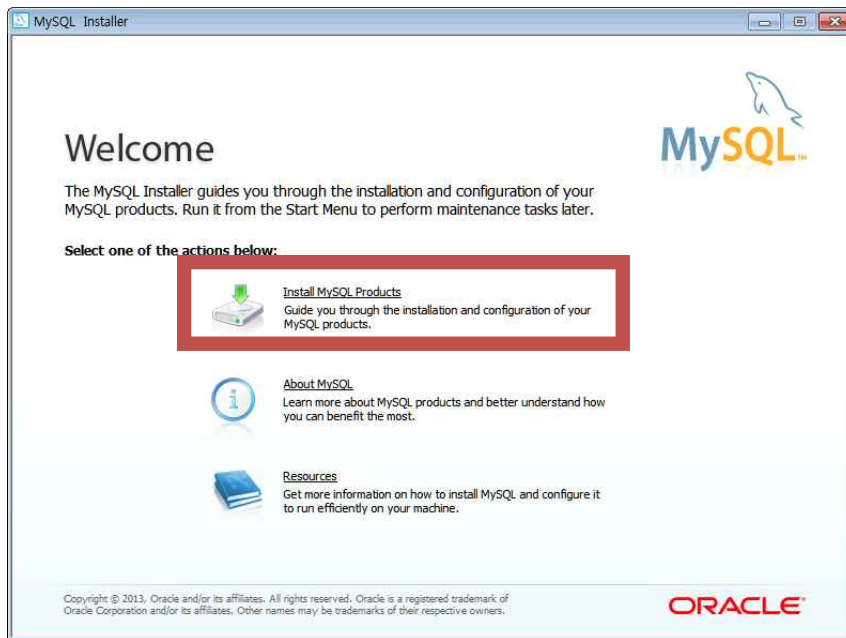
❶ <http://dev.mysql.com/downloads/>에 접속



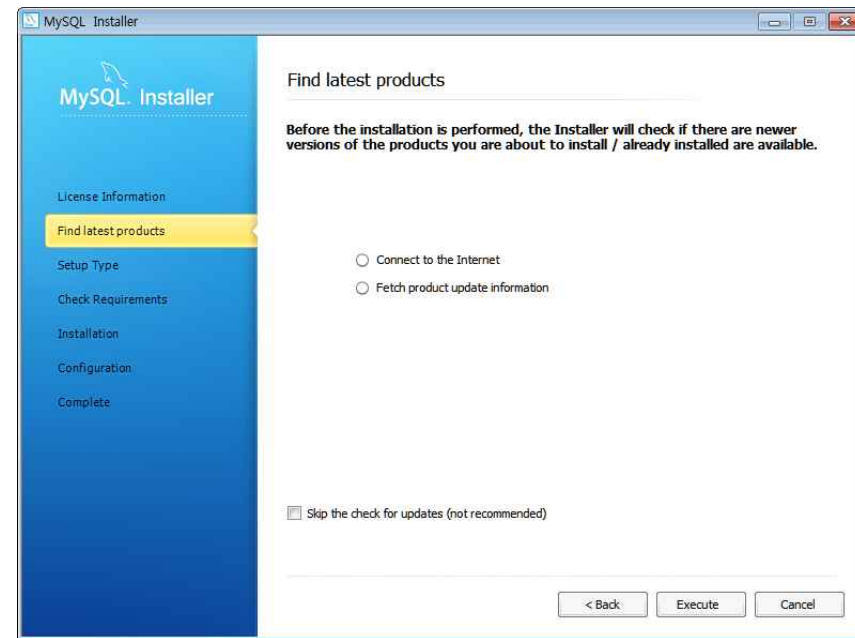
[그림 8-7] MySQL 다운로드

## 02. MySQL 데이터베이스 설치와 설정

② 다운로드한 파일을 더블클릭하여 설치를 시작.



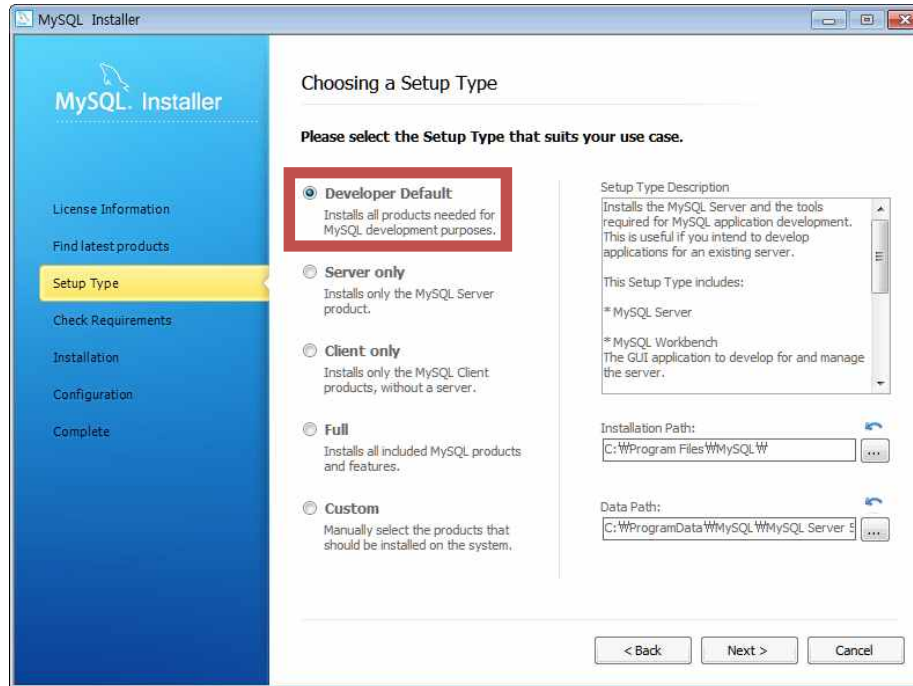
[그림 8-9] 설치 시작 화면



[그림 8-10] 최신 버전 확인

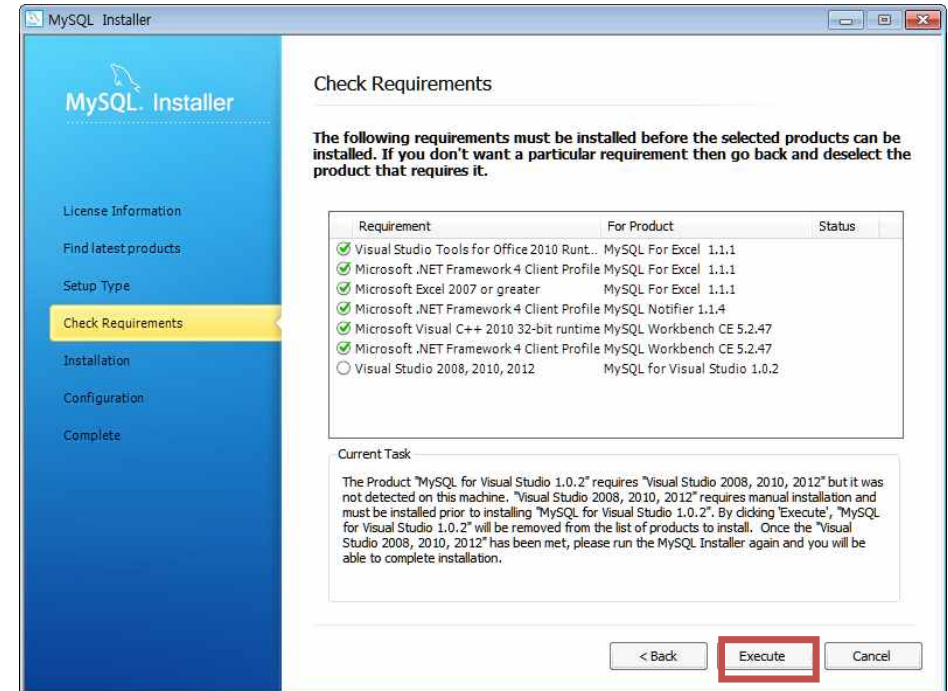
## 02. MySQL 데이터베이스 설치와 설정

3



[그림 8-11] 설치 유형 선택

4

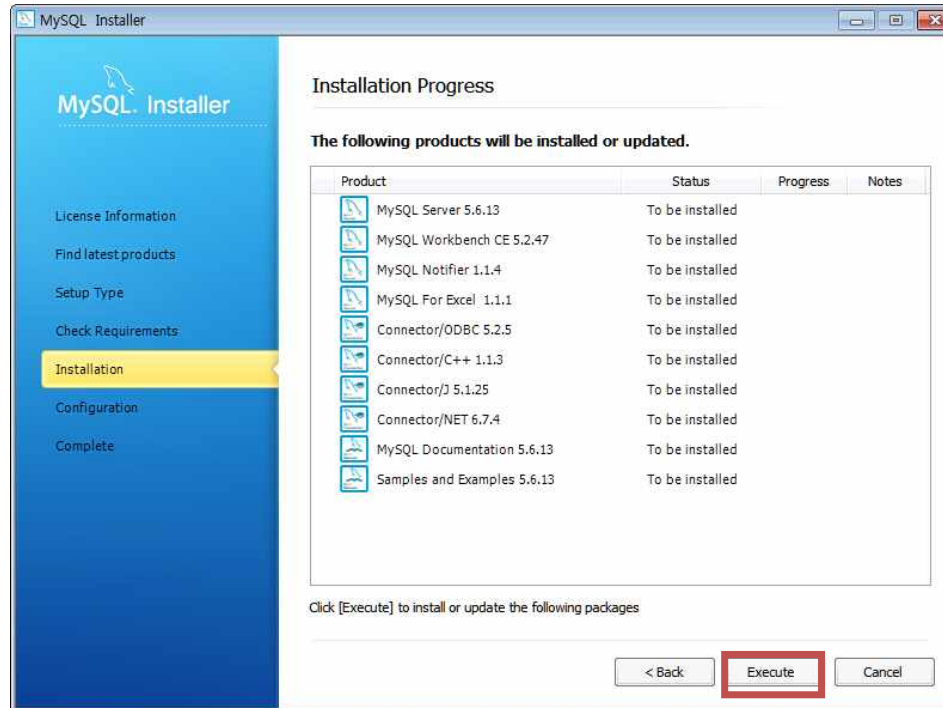


[그림 8-12] 각종 프로그램 및 라이브러리 설치



## 02. MySQL 데이터베이스 설치와 설정

5

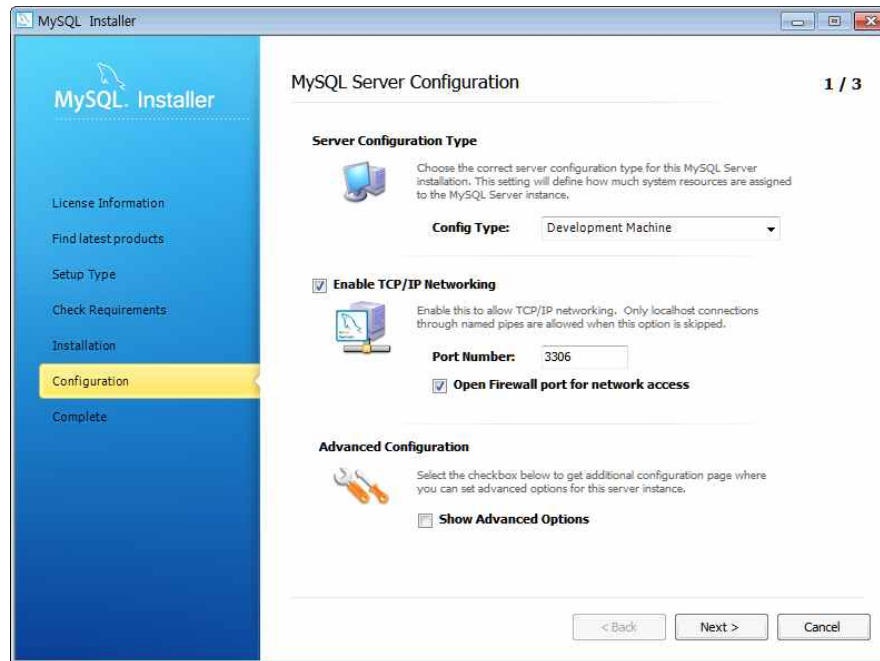


[그림 8-13] 세부 설치 항목

## 02. MySQL 데이터베이스 설치와 설정

### 2. MySQL 데이터베이스 초기 설정

#### ■ MySQL 서버 관련 설정

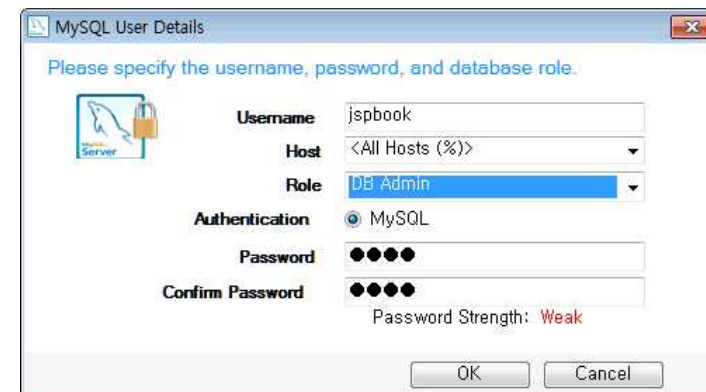


[그림 8-14] MySQL 서버 설정 옵션

#### ■ 사용자 계정 관련 설정



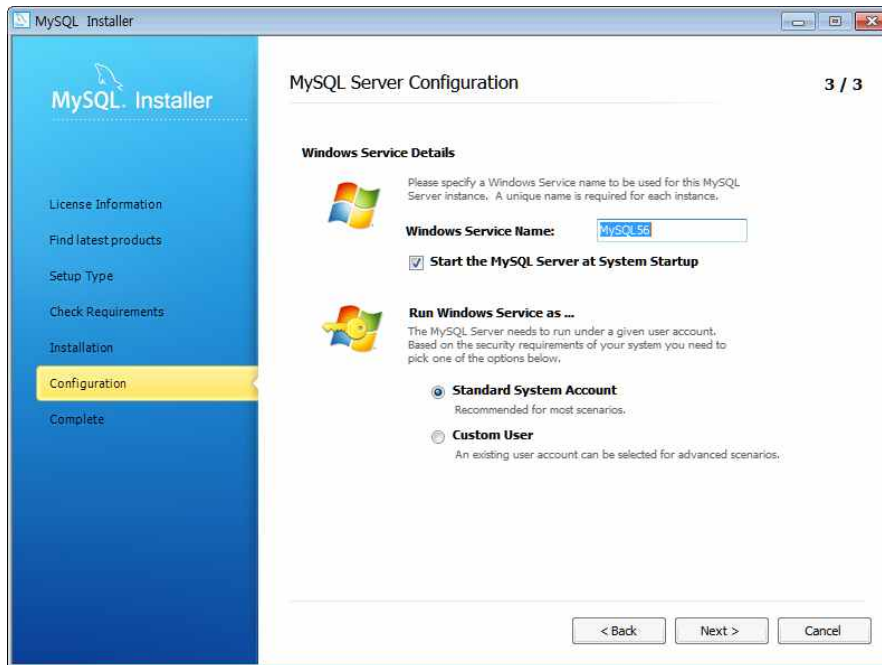
[그림 8-15] MySQL 사용자 계정 설정 옵션



[그림 8-16] MySQL 사용자 추가 화면

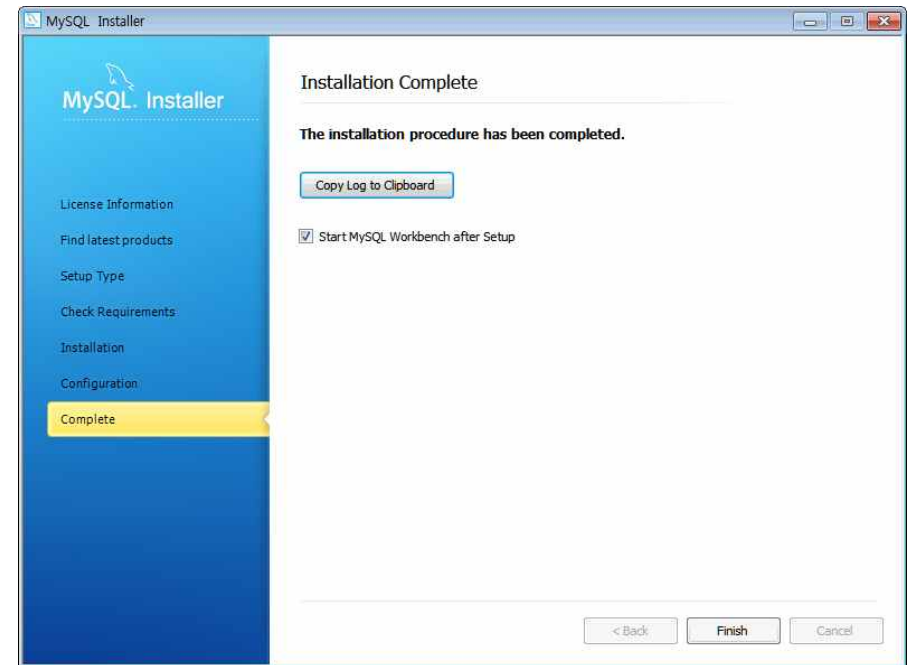
## 02. MySQL 데이터베이스 설치와 설정

### ■ 윈도우 서비스 설정



[그림 8-17] MySQL 윈도우 서비스 설정

### ■ MySQL 설치 및 설정 마무리

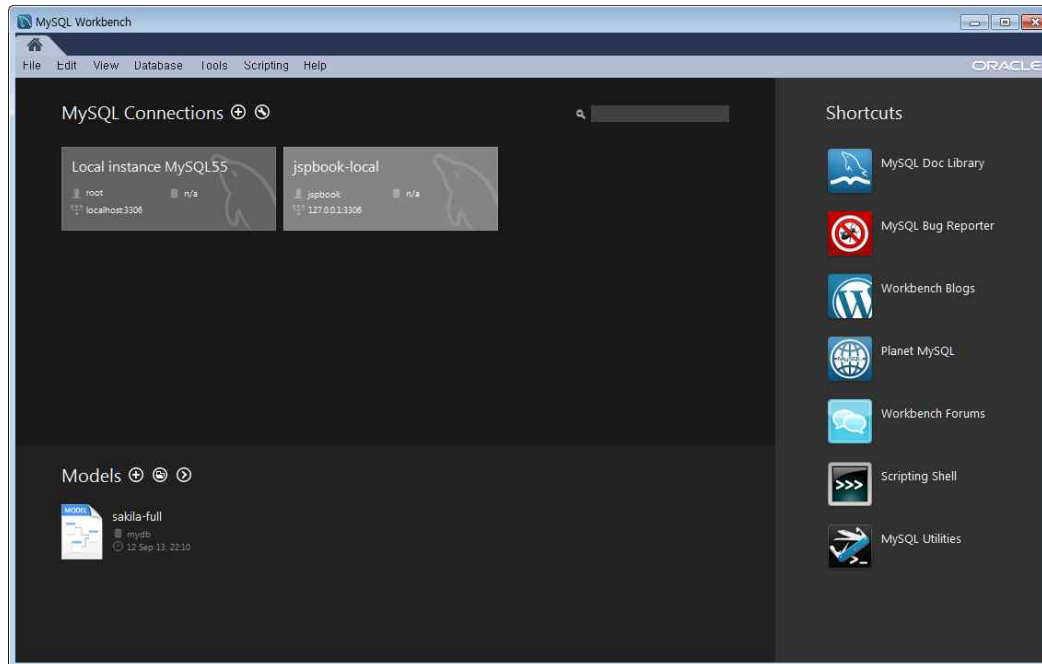


[그림 8-18] MySQL 설치 완료 화면

## 02. MySQL 데이터베이스 설치와 설정

### 3. MySQL Workbench 실행 및 사용자 설정

#### ■ MySQL Workbench 관리자 모드

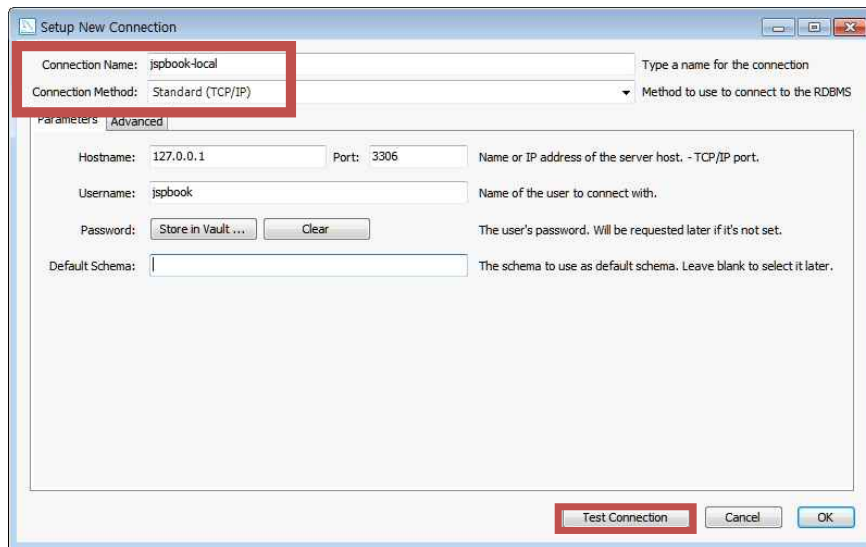


[그림 8-19] Workbench 초기 화면

## 02. MySQL 데이터베이스 설치와 설정

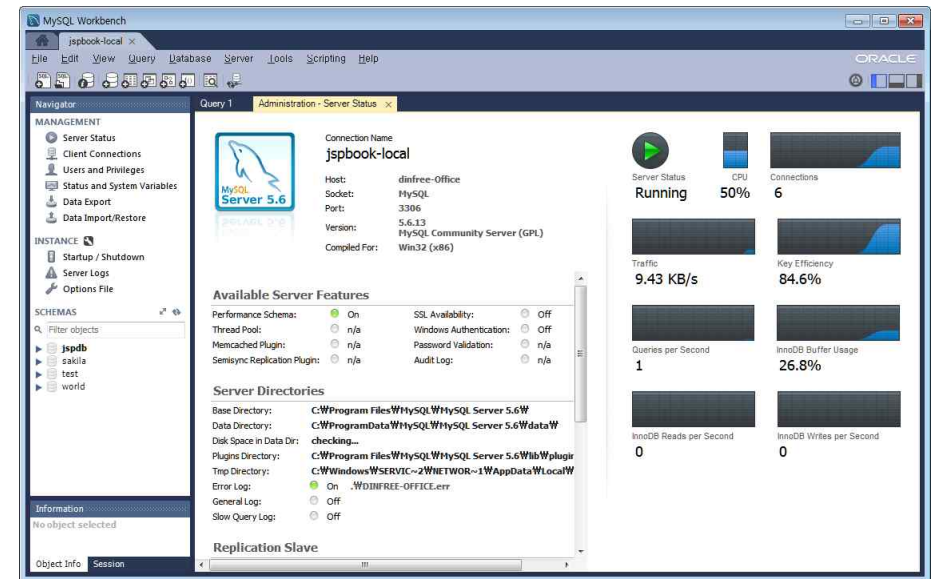
### MySQL Workbench 연결 설정

1



[그림 8-20] 연결 설정 화면

2

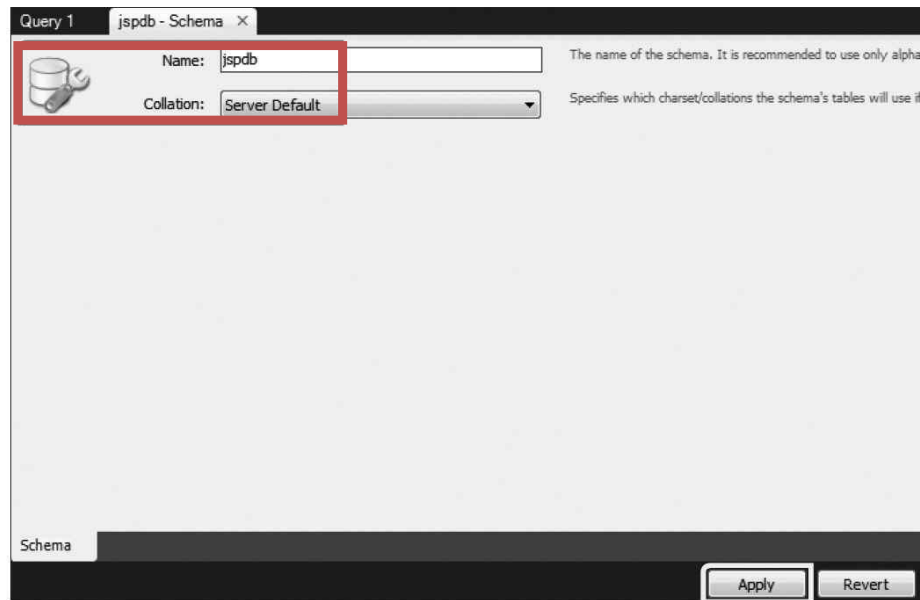


[그림 8-21] 관리자 메인 화면

## 02. MySQL 데이터베이스 설치와 설정

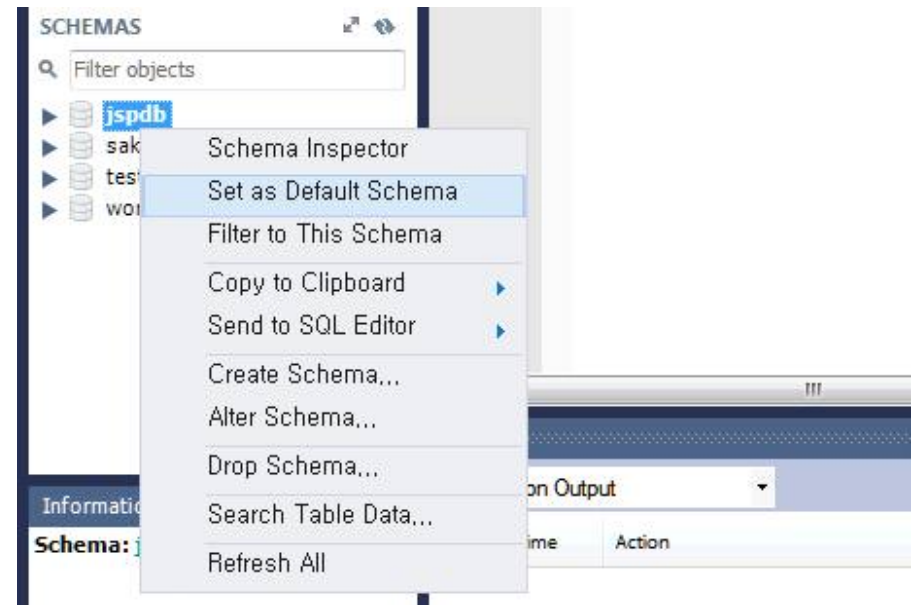
### 스키마 생성

1



[그림 8-22] 스키마 생성 화면

2



[그림 8-23] 기본 스키마 지정

# 03. SQL문 기본기 다지기

## 1. 데이터베이스 자료형

- 프로그램 언어에서와 유사하게 데이터베이스에도 데이터베이스 관리를 위한 별도의 자료형을 가지고 있음.
- 오라클, MySQL, DB2 등 모든 데이터베이스는 유형은 비슷하지만 서로 다른 자료형을 가지고 있어 특정 DB를 사용하려면 해당 DB에 맞는 자료형을 알아야 함.

[표 8-2] MySQL 데이터베이스의 자료형

자료형	유형	지원 범위 / 비고
TINYINT	• 정수형 • 크기에 따른 구분	• -128~127 • 0~255 UNSIGNED
SMALLINT		• -32768 ~ 32767 • 0 ~ 65535 UNSIGNED
MEDIUMINT		• -8388608 ~ 8388607 • 0 ~ 16777215 UNSIGNED
INT		• -2147483648 ~ 2147483647 • 0 ~ 4294967295 UNSIGNED

## 03. SQL문 기본기 다지기

자료형	유형	지원 범위 / 비고
BIGINT	• 정수형 • 크기에 따른 구분	• -9223372036854775808 ~ 9223372036854775807 • 0~18446744073709551615 UNSIGNED
FLOAT	• 실수형 • 크기에 따른 구분	-3.402823466E+38 ~ 1.175494351E-38
DOUBLE		-1.7976931348623157E+308 ~ -2.22507385850572014E-308
REAL		DOUBLE과 동의어, FLOAT과 동의어로 설정을 변경할 수 있다.
DATE	날짜	YYYY-MM-DD
DATETIME	시간을 포함한 날짜	YYYY-MM-DD HH:MM:SS
TIMESTAMP	1970-01-01 00:00:00에서 2037까지 의 최근 연산 시간 정보	YYYY-MM-DD HH:MM:SS
TIME	시간	HH:MM:SS
YEAR	연도	YY 혹은 YYYY



## 03. SQL문 기본기 다지기

자료형	유형	지원 범위 / 비고
CHAR	<ul style="list-style-type: none"><li>고정길이 문자열</li><li>왼쪽부터 채우고 남은 공간 공백</li></ul>	0 ~ 255문자
VARCHAR	<ul style="list-style-type: none"><li>가변길이 문자열</li><li>전체적으로 남은 공간 활용은 유리하지만 CHAR보다 1byte 더 소비됨</li></ul>	0 ~ 255문자
TINYTEXT	TEXT형	255문자
TEXT		65535문자
MEDIUMTEXT		16777215문자
LONGTEXT		4294967295 문자
MEDIUMBLOB	대용량 바이너리형	16777215문자
LOBLOB		4294967295문자
BLOB		4294967295문자
BINARY	CHAR 바이너리형	255바이트
VARBINARY	VARCHAR 바이너리형	65535바이트
ENUM	<ul style="list-style-type: none"><li>열거형</li><li>해당 칼럼에 올 수 있는 값이 계산된다.</li></ul>	
SET	<ul style="list-style-type: none"><li>ENUM과 유사하며 지정된 값들을 쌍으로 가질 수 있다.</li><li>비트 연산을 할 수 있다.</li></ul>	
BOOL	Boolean	true, false

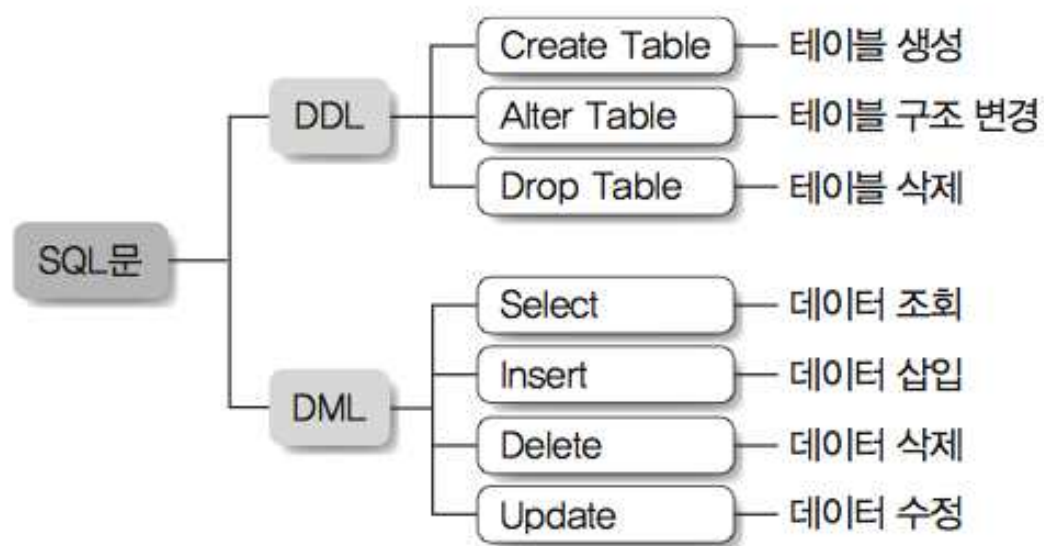
## 03. SQL문 기본기 다지기

- 많은 자료형이 있는 이유는 데이터를 효과적으로 관리하고 연산하기 위함.
- 이름만 다를뿐 프로그램 언어에서의 자료형과 유사하다고 생각 하면 됨.
- 다만 BLOB, TEXT 등은 대용량 비정형 데이터 처리를 위한 특수한 형태로 다음과 같은 특징을 가진다.
  - 인덱스를 생성할 수 없다.
  - 지정된 최대 크기보다 작은 문자열을 저장해도 공백이 제거되지 않는다.
  - 기본 값을 지정할 수 없으므로 NOT NULL은 무효다.
  - 테이블이 아닌 다른 영역에 저장된다.

# 03. SQL문 기본기 다지기

## 2. 기본 SQL 문법 및 실습

- SQL(Structured Query Language) 는 데이터베이스의 데이터를 관리하기 위한 쿼리 언어임.
- 데이터베이스 종류와 상관 없이 모든 데이터베이스는 SQL 을 통해서만 데이터 관리가 가능함.
- 기본적으로는 ANSI 표준이며 데이터베이스 회사별로 조금씩 차이가 있음.



[그림 8-24] 기본 SQL문

## 03. SQL문 기본기 다지기

### ■ 테이블 생성 명령 : CREATE TABLE

- 테이블을 생성하기 위한 명령문.
- 자료가 들어 있는 테이블의 구조 변경은 어렵기 때문에 테이블 구조 생성은 신중하게 결정.
- 기본 문법

```
CREATE TABLE 테이블_이름 (  
    컬럼_이름 데이터형(크기) 옵션,  
    컬럼_이름 데이터형(크기),  
    .....  
)
```

- 기본적으로는 컬럼 이름과 데이터형(크기) 형식으로 이루어짐.
- 마지막 컬럼 에는 "," 를 넣지 않는다.

# 03. SQL문 기본기 다지기

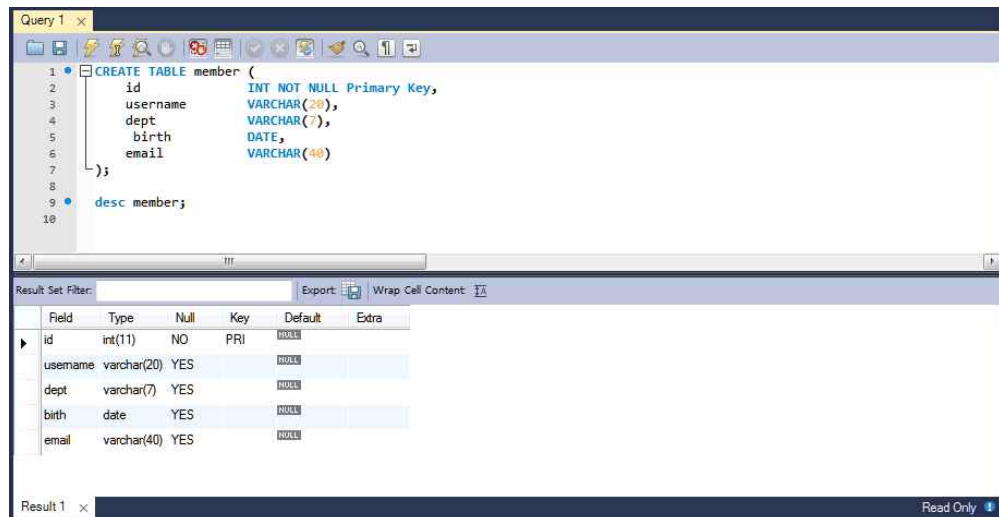
- [실습] 교재 p.304 참고

id	username	dept	birth	email

[그림 8-25] CREATE TABLE 사용 예

```
CREATE TABLE member (  
    id          INT NOT NULL Primary Key,  
    username    VARCHAR(20),  
    dept        VARCHAR(7),  
    birth       DATE,  
    email       VARCHAR(40)  
);
```

- 결과물



Query 1

```
1 CREATE TABLE member (  
2     id          INT NOT NULL Primary Key,  
3     username    VARCHAR(20),  
4     dept        VARCHAR(7),  
5     birth       DATE,  
6     email       VARCHAR(40)  
7 );  
8  
9 desc member;  
10
```

Result Set Filter: Export: Wrap Cell Content

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	
username	varchar(20)	YES		NULL	
dept	varchar(7)	YES		NULL	
birth	date	YES		NULL	
email	varchar(40)	YES		NULL	

Result 1 Read Only

[그림 8-26] CREATE TABLE 실행 결과

## 03. SQL문 기본기 다지기

### ■ 테이블 변경(수정) 명령 : ALTER TABLE

- 테이블을 수정하기 위한 명령문.
- 모든 테이블의 구조를 변경할수는 없으며 다음과 같은 제약을 가짐

[표 8-3] ALTER TABLE 사용의 제약 조건

데이터 유무	가능한 작업
자료가 있는 경우	<ul style="list-style-type: none"><li>• 칼럼의 데이터 자료형을 변경할 수 있다.</li><li>• 칼럼의 폭을 줄이거나 늘릴 수 있다.</li><li>• NULL이나 NOT NULL 속성을 갖는 칼럼을 추가시킬 수 있다.</li></ul>
자료가 없는 경우	<ul style="list-style-type: none"><li>• 칼럼의 데이터 자료형을 변경할 수 없다.</li><li>• 폭을 줄일 수는 없지만, 늘릴 수는 있다.</li><li>• NOT NULL 속성을 갖는 필드는 추가시킬 수 있으나 NULL 속성이 있는 필드는 추가시킬 수 없다.</li></ul>

- 기본 문법

ALTER TABLE 테이블\_이름 [ADD/MODIFY/DROP/ENABLE/DISABLE] (칼럼\_이름 데이터형)

## 03. SQL문 기본기 다지기

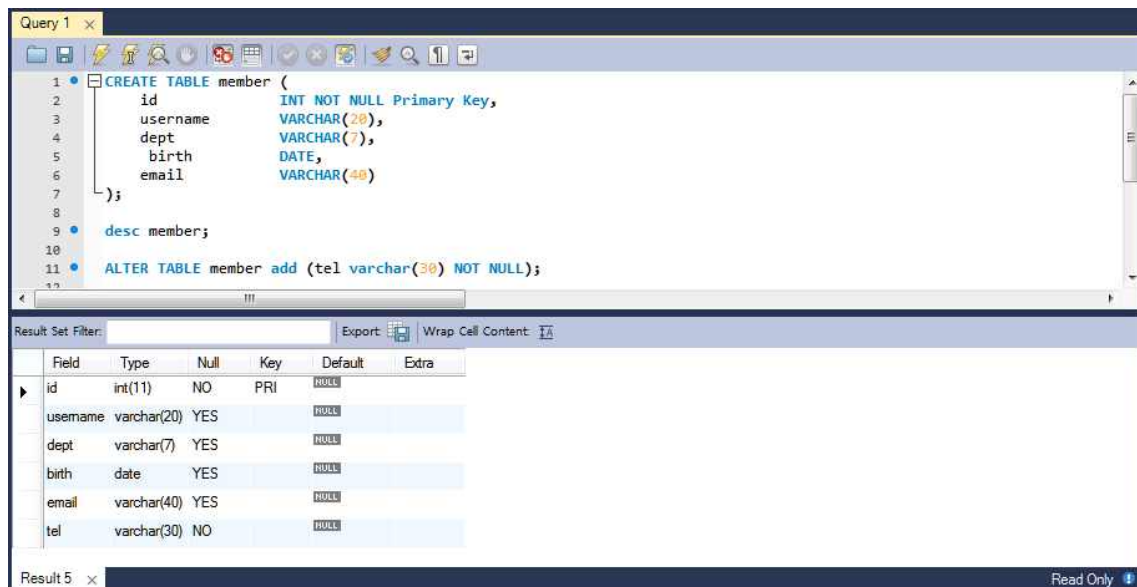
- [실습] 교재 p.305 ~ 306 참고

id	username	dept	birth	email	tel

[그림 8-27] ALERT TABLE 사용 예

```
ALTER TABLE member add (tel varchar(30) NOT NULL);  
ALTER TABLE member modify username varchar(10);  
ALTER TABLE member DROP PRIMARY KEY;
```

- 결과물



[그림 8-28] ALERT TABLE 실행

## 03. SQL문 기본기 다지기

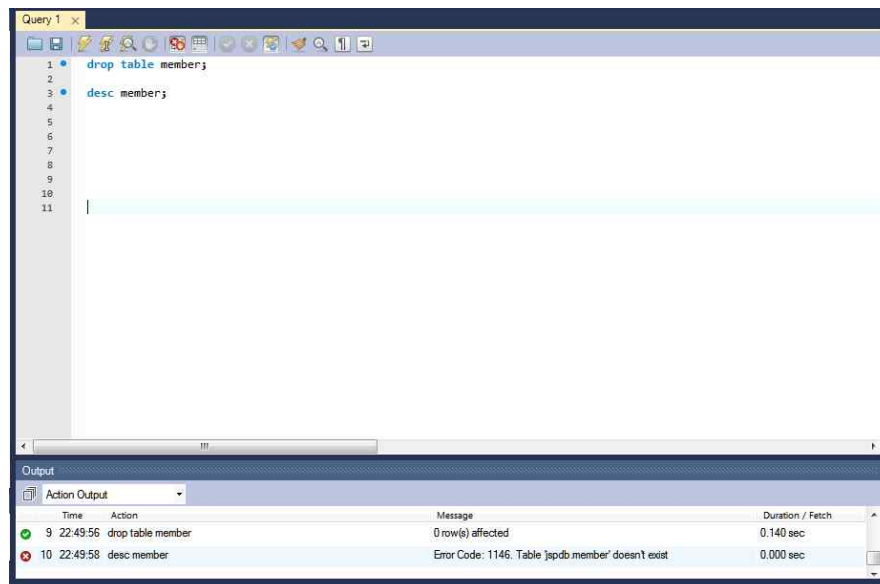
- 테이블을 삭제하기 위한 명령문.
- 테이블 drop 시 모든 데이터도 함께 삭제됨. 외래키로 연결된 경우 옵션을 통해 연결된 데이터를 모두 삭제할수도 있음.
- 기본 문법

DROP TABLE 테이블\_이름

- [실습] [교재 p.306 참고](#)

DROP TABLE member ;

- 결과물



[그림 8-29] DROP TABLE 실행 결과



# 03. SQL문 기본기 다지기

## ■ 테이블 데이터 추가 명령 : INSERT

- 테이블에 데이터를 추가하기 위한 명령문.
- 기본 문법

INSERT INTO 테이블\_이름(삽입할 칼럼\_이름...) VALUES(칼럼에 넣을 값...)

- [실습] [교재 p.307 참고](#)

id	username	dept	birth	email
201301	홍길동	정보기술	1993-04-12	my@my.net
201302	아무개	정보기술	1993-08-07	tt@tt.net



201303	강기동	정보기술	1992-10-23	mm@mm.net
--------	-----	------	------------	-----------

INSERT INTO member values(201303,'강기동','정보기술','1992-10-23','mm@mm.net')  
// 테이블에 정의되어 있는 칼럼 순서대로 모든 칼럼 데이터가 들어가야 한다.  
INSERT INTO member(id,dept,username) values(201304,'정보기술','강기동2');  
// 지정된 필드의 순서에 따라 데이터가 들어간다.

[그림 8-30] INSERT 사용 예

- 결과물 : drop한 테이블을 다시 생성 후 작업

```
Query 1 x
1 CREATE TABLE member (
2   id          INT NOT NULL Primary Key,
3   username    VARCHAR(20),
4   dept        VARCHAR(7),
5   birth       DATE,
6   email       VARCHAR(40)
7 );
8
9 INSERT INTO member values(201301,'홍길동','정보기술','1993-04-12','my@my.net');
10 INSERT INTO member values(201302,'아무개','정보기술','1993-08-07','tt@tt.net');
11
12 INSERT INTO member values(201303,'강기동','정보기술','1992-10-23','mm@mm.net');
13 INSERT INTO member(id, dept, username) values(201304,'정보기술','강기동2');
14
```

[그림 8-31] INSERT 실행 결과

# 03. SQL문 기본기 다지기

## ■ 테이블 데이터 조회 명령 : SELECT

- 테이블내 데이터를 조회하기 위한 명령문.
- 기본 문법

SELECT 컬럼\_이름 FROM 테이블\_이름 WHERE 조건

- 컬럼\_이름 대신 "\*" 를 사용해 모든 컬럼을 가져올 수 있다.

- [실습] [교재 p.308 참고](#)

id	username	dept	birth	email
201301	홍길동	정보기술	1993-04-12	my@my.net
201302	아무개	정보기술	1993-08-07	tt@tt.net
...				

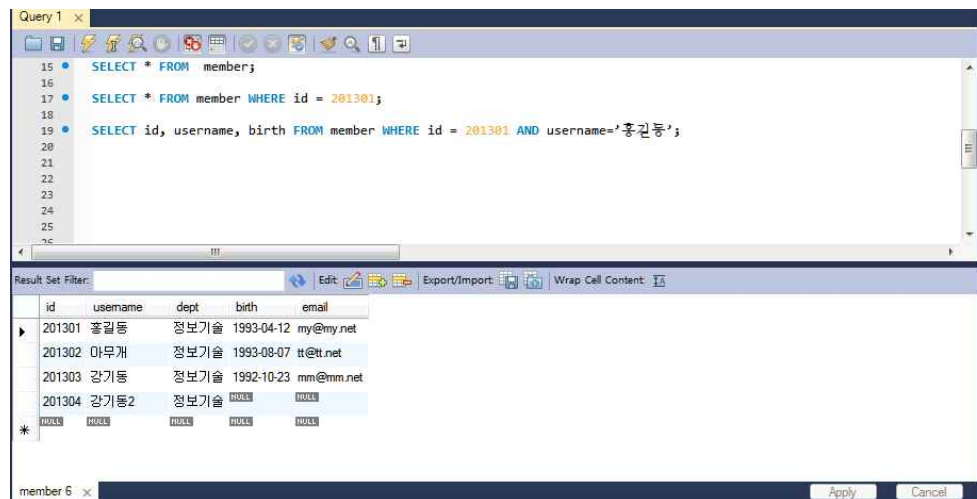
SELECT \* FROM member;

SELECT \* FROM member WHERE id = 201301;

SELECT id, username, birth FROM member WHERE id = 201301 AND username='홍길동';

[그림 8-32] SELECT 사용 예

- 결과물 : 여러 데이터 추가해 볼것.



[그림 8-33] SELECT 실행 결과

## 03. SQL문 기본기 다지기

### ■ 테이블 데이터 수정 명령 : UPDATE

- 테이블내 데이터를 수정하기 위한 명령문.
- 기본 문법

UPDATE 테이블\_이름 SET 칼럼\_이름1 = 수정할 값1, 칼럼\_이름2 = 수정할 값2  
... WHERE 조건

- WHERE 조건절을 넣지 않으면 모든 테이블 데이터가 동일하게 변경 되므로 주의해야 함.

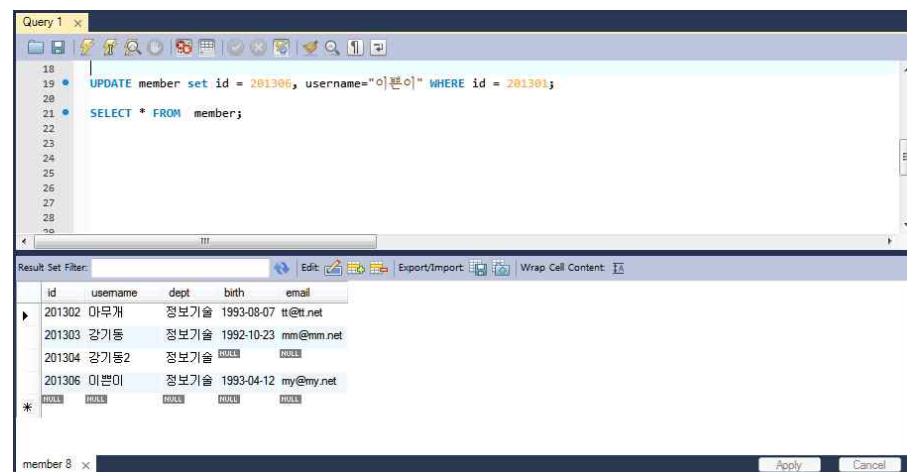
- [실습] [교재 p.309 참고](#)

id	username	dept	birth	email
201301	홍길동	정보기술	1993-04-12	my@my.net
201302	아무개	정보기술	1993-08-07	tt@tt.net
201306	이쁜이			

UPDATE member SET id = 201306, username="이쁜이" WHERE id = 201301;

[그림 8-34] UPDATE 사용 예

- 결과물 : 여러 조건으로 데이터 수정.



[그림 8-35] UPDATE 실행 결과

# 03. SQL문 기본기 다지기

## 테이블 데이터 수정 명령 : DELETE

- 테이블내 데이터를 삭제하기 위한 명령문.
- 기본 문법

DELETE FROM 테이블\_이름 WHERE 조건

- WHERE 조건절을 넣지 않으면 모든 테이블 데이터가 동일하게 삭제 되므로 주의해야 함.

- [실습] [교재 p.310 참고](#)

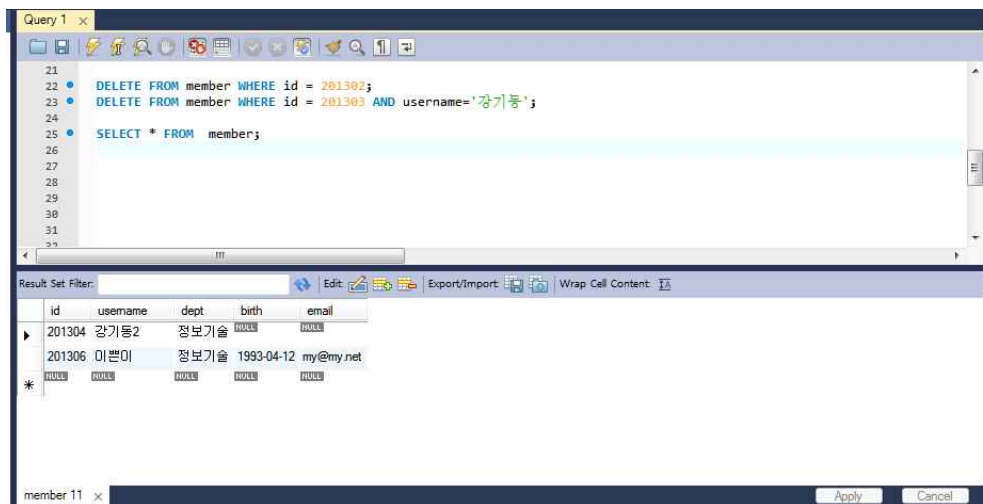
id	username	dept	birth	email
201301	홍길동	정보기술	1993-04-12	my@my.net
201302	아무개	정보기술	1993-08-07	tt@tt.net
201303	강기동	정보기술	1992-10-23	mm@mm.net
...				

삭제할 칼럼

```
DELETE FROM member WHERE id = 201302;  
DELETE FROM member WHERE id = 201303 AND name='강기동';
```

[그림 8-36] DELETE 사용 예

- 결과물 : 여러 조건으로 데이터 수정.



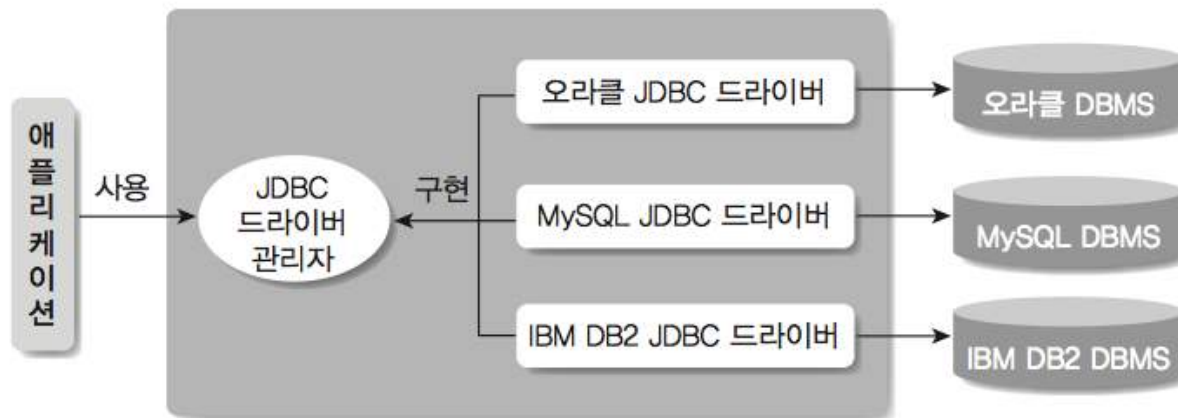
[그림 8-37] DELETE 실행 결과

# 04. JDBC 기본 구조와 API 이해

## 1. JDBC 개념과 역할

### ■ JDBC(Java Database Connectivity)란 ?

- JDBC 는 자바 프로그램에서 서로 다른 데이터베이스를 표준화된 방법으로 접속 할 수 있도록 만든 API 규격을 말한다.
- JDBC 를 사용하면 개발자는 데이터베이스 종류와 무관하게 표준화된 API를 이용해서 프로그램을 개발 할 수 있다.
- 이론적으로는 데이터베이스가 변경 되어도 프로그램은 변경할 필요가 없이 JDBC 드라이버만 해당 데이터베이스용으로 변경하면 된다.



[그림 8-38] JDBC 구조

## 2. JDBC 드라이버 설치

### ■ JDBC 드라이버 설치 개요

- JDBC 드라이버는 각 DBMS 제조업체 홈페이지에서 무료로 다운로드 할 수 있다.
- MySQL 의 경우 JDBC Connector 를 별도로 다운로드 할수도 있다. MySQL 을 Developer Default 모드로 설치 했다면 관련 파일들은 이미 PC에 복사되어 있는 상태.
- MySQL JDBC 드라이버 다운로드 : <http://www.mysql.com/downloads/connector/j>

### ■ JDBC 드라이버 설치 방법

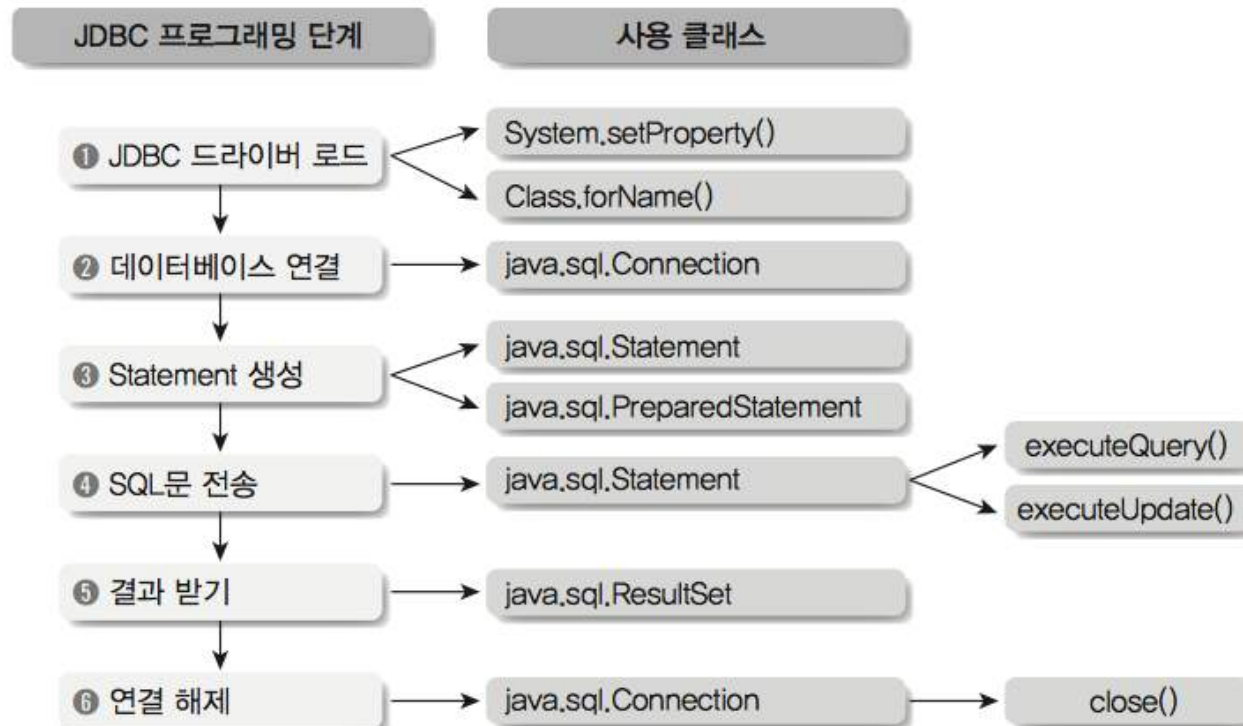
- JDBC 드라이버 설치하는 다음의 세가지 방법을 이용할 수 있으며 웹 애플리케이션의 배포를 고려할 경우 3번 방법이 적합함.
- ❶ [java 설치 디렉터리\jre7\lib\ext]에 복사하는 방법
  - ❷ [톰캣 설치 디렉터리\lib]에 복사하는 방법
  - ❸ 이클립스 프로젝트의 [WebContent\WEB-INF\lib]에 복사하는 방법

## 04. JDBC 기본 구조와 API 이해

### 3. JDBC API 이해 및 프로그래밍 단계

#### ■ JDBC 프로그래밍 개요

- JDBC 프로그래밍은 JDBC API를 이용해 정해진 절차와 규격에 따라 프로그램을 작성해야 한다.
- JDBC API는 방대한 구성이므로 상세한 기능은 API 문서를 참고하도록 한다.
- JSP에서 스크립트릿으로 작성하거나 자바 클래스로 만들어 빈즈로 연결해 사용할 수 있다. JSP에서 직접적인 JDBC 프로그래밍은 예제 실습등의 목적이 아니라면 권장되지 않는다.



[그림 8-40] JDBC 프로그래밍 단계

## 04. JDBC 기본 구조와 API 이해

### ■ 1단계: JDBC 드라이버 로딩

- JDBC API 사용을 위해서는 먼저 JDBC 규격에 따른 실제 구현된 각 JDBC 드라이버 클래스를 로딩해야 한다.
- JDBC 드라이버 로딩은 크게 두가지가 있다.

- **jdbc.drivers 환경변수 이용하기**

- 먼저 다음과 같이 jdbc.drivers 환경변수를 설정한다.

```
System.setProperty("jdbc.drivers","com.mysql.jdbc.Driver");
```

- 환경변수를 이용하는 경우에는 자동으로 드라이버를 로드하므로 추가 명시적인 작업은 필요 없으나 일반적으로는 프로그램에서 처리가 용이한 다음의 Class.forName() 을 사용한다.

- **Class.forName() 메서드 이용하기**

- 일반적으로는 이용하는 방법이다. 이 경우 원하는 JDBC 드라이버를 직접 프로그램 코드에서 로드할 수 있게 된다.

```
Class.forName("com.mysql.jdbc.Driver");
```

```
Class.forName("oracle.jdbc.driver.OracleDriver");    <- oracle
```

- com.mysql.jdbc.Driver는 MySQL 의 JDBC 드라이버 클래스 이름이다. 데이터베이스마다 클래스 이름이 다르므로 해당 DB의 이름을 정확하게 넣어줘야 한다.



## 04. JDBC 기본 구조와 API 이해

### ■ 2단계: 데이터베이스 연결

- 드라이버가 로드되면 JDBC API를 이용해 프로그램을 작성할 상태가 된 것을 의미한다.
- 실제 데이터베이스 프로그래밍을 위해서는 먼저 DB에 연결해야 하는데 DriverManager 클래스의 getConnection() 메서드를 사용한다.
- 이때 필요한 정보로 JDBC URL과 사용자 아이디, 비밀번호가 있다.
- **JDBC URL**
  - JDBC URL은 데이터베이스에 대한 다양한 정보를 포함하고 있다. JDBC URL 구조는 다음과 같다. 각 데이터베이스별로 JDBC URL이 다르므로, 사용하는 데이터베이스 매뉴얼을 참고해서 작성해야 한다.

```
jdbc:<하위 프로토콜>:<데이터 원본 식별자>
```

- MySQL은 다음과 같은 형식을 취한다.

```
jdbc:mysql:// IP주소/스키마:PORT(옵션)
```

①      ②      ③

- ① IP 주소 : MySQL 데이터베이스가 설치된 컴퓨터의 IP 주소, 또는 도메인 이름이다.
- ② 스키마 : 데이터베이스에서 생성한 스키마(데이터베이스) 이름이다.
- ③ 포트 : 기본 설정 값인 3306 포트를 사용하는 경우에는 생략할 수 있다.

## 04. JDBC 기본 구조와 API 이해

- **Connection** 클래스 인스턴스 레퍼런스 얻기

- DriverManager 의 getConnection() 메서드는 다음과 같이 구성된다.

```
Connection conn = DriverManager.getConnection(JDBC_URL,"아이디","비밀번호");
```

①            ②            ③

① JDBC\_URL : 해당 데이터베이스에 맞게 미리 정의되어 있는 문자열이다.

② 아이디와 ③ 비밀번호 : 시스템에 로그인하는 계정이 아니라 데이터베이스 자체에서 관리하는 계정이다.

## 04. JDBC 기본 구조와 API 이해

### ■ 3단계: Statement 생성

- Statement 는 데이터베이스 연결로 부터 SQL 문을 수행할 수 있도록 해주는 클래스.
- 대표적인 메서드는 다음과 같다.

- executeQuery()

SELECT문을 수행할 때 사용한다. 반환 값은 ResultSet 클래스의 인스턴스로, 해당 SELECT문의 결과에 해당하는 데이터에 접근할 수 있는 방법을 제공한다.

- executeUpdate()

UPDATE, DELETE와 같은 문을 수행할 때 사용한다. 반환 값은 int 값으로, 처리된 데이터의 수를 반환한다.

- Statement 객체

- Statement 객체는 쿼리를 문자열로 연결해야 하므로 소스가 복잡하고 오류가 발생하기 쉽다.

```
Statement stmt = conn.createStatement();

stmt.executeUpdate("insert into test values

(' "+request.getParameter("username")+"

','"+request.getParameter

("email")+" '");
```

## 04. JDBC 기본 구조와 API 이해

- PreparedStatement 는 SQL 에 필요한 변수 데이터를 "?"로 표시하고 메서드를 통해 설정하는 방식으로 Statement 보다 구조적이고 편리해 권장되는 방법이다.

```
PreparedStatement pstmt = conn.prepareStatement("insert into test  
values(?,?)");  
  
pstmt.setString(1,request.getParameter("username");  
  
pstmt.setString(2,request.getParameter("email");  
  
pstmt.executeUpdate();
```

- pstmt.setXxx() 메서드는 데이터 타입별로 제공되므로 DB 테이블 컬럼의 데이터 타입에 맞춰 사용해야 한다.
- 사용한 JDBC 리소스는 명시적으로 반납해 주는 것이 좋다.

```
stmt.close();  
  
pstmt.close();
```

## 04. JDBC 기본 구조와 API 이해

### ■ 4단계: SQL문 전송

- 데이터 조합과 함께 만들어진 SQL 문은 명시적인 처리 명령에 의해 DB로 전달되어 실행된다.
- insert, delete, update 와 같이 데이터 변경이 있는 쿼리의 경우에는 다음과 같이 executeUpdate() 문을 사용한다.

```
pstmt.executeUpdate();
```

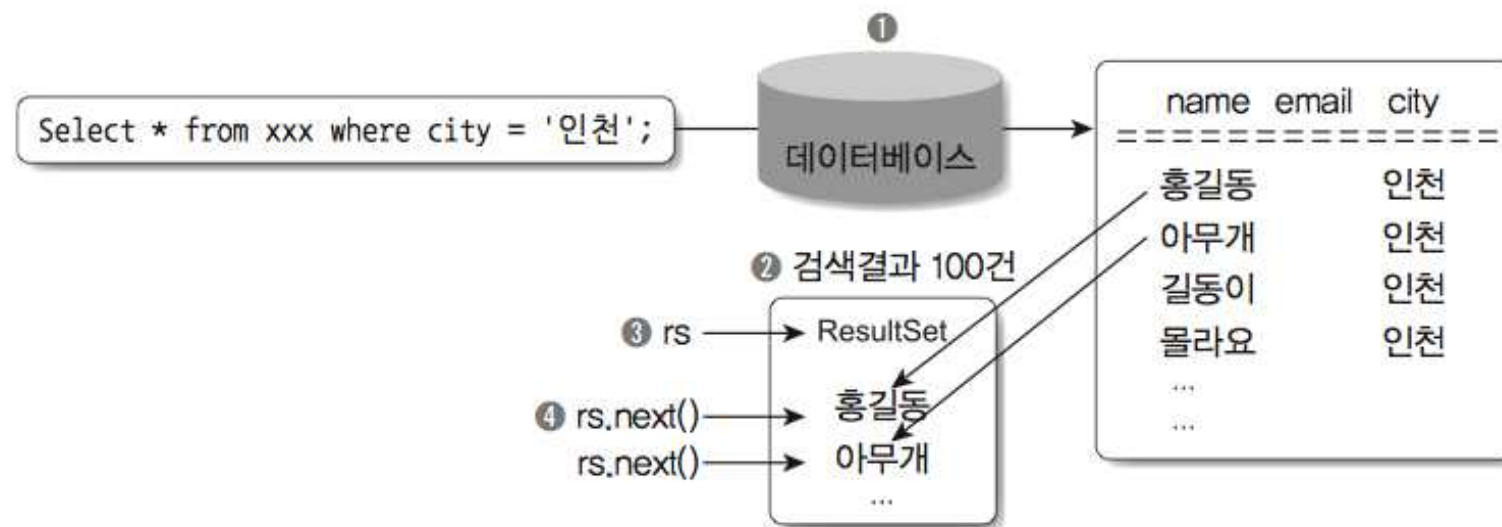
```
int count = pstmt.executeUpdate(); // 처리한 로우의 개수 반환
```

- select 문의 경우 executeQuery() 메서드를 사용하고 조회 결과를 받기 위해 ResultSet 객체를 사용한다.

## 04. JDBC 기본 구조와 API 이해

### ■ 5단계: 결과 받기

- 데이터베이스에서 조회한 결과를 받기 위해서는 Statement 나 PreparedStatement 의 executeQuery()를 사용한다.
- 조회 결과는 ResultSet 객체로 리턴되며 ResultSet 은 실제 데이터셋이 아니라 데이터에 접근할 수 있는 포인터의 집합 개념으로 이해하면 된다.



[그림 8-41] SQL 처리 결과와 ResultSet의 관계

## 04. JDBC 기본 구조와 API 이해

- ResultSet 을 이용해 데이터를 처리하는 방법은 rs.next() 메서드로 다음 데이터를 확인하고 데이터가 있을 경우 rs.getXxx() 메서드를 이용해 특정 컬럼에 해당하는 데이터를 가지고와 사용하게 된다.
- getXxx() 메서드는 데이터 타입별로 존재 하므로 컬럼 데이터 타입에 따라 적절히 사용한다.

```
ResultSet rs = pstmt.executeQuery();  
while(rs.next()) {  
    name = rs.getString(1); // or rs.getString("name");  
    age = rs.getInt(2); // or rs.getInt("email");  
}  
rs.close();
```

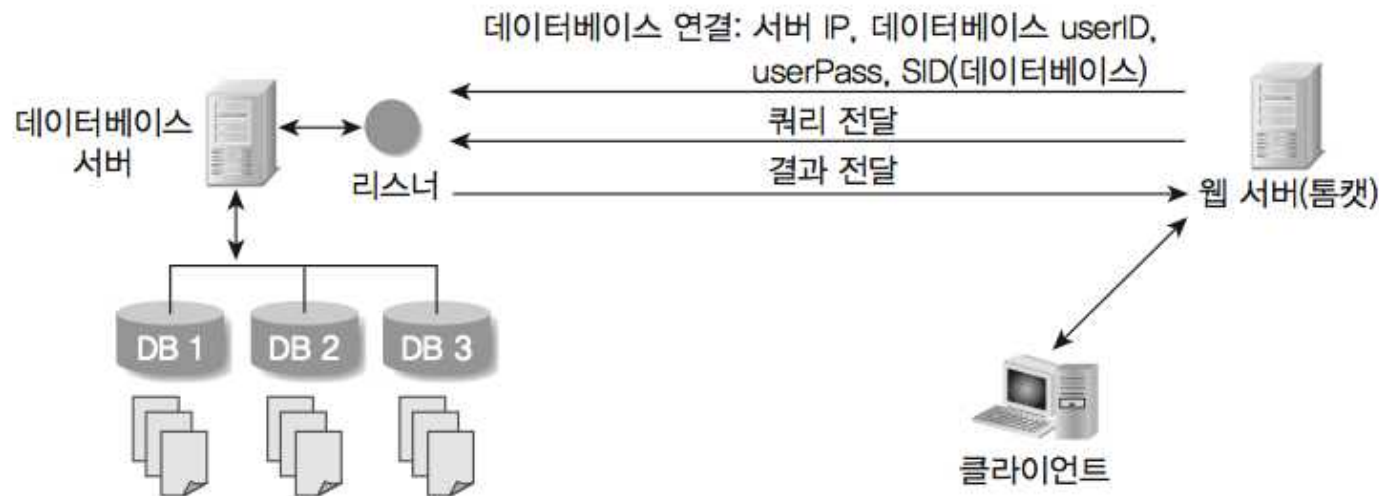
- 사용이 끝난 ResultSet 객체 역시 close() 메서드를 이용해 리소스를 반환 하도록 한다.

### ■ 6단계: 연결 해제

- 사용이 끝난 데이터베이스 연결은 conn.close() 메서드를 이용해 닫아 주도록 한다. DB 연결은 중요한 자원이고 제한적이므로 사용이 끝난 연결은 반드시 해제해 주어야 하므로 주의 하도록 한다.

## 1. 실습 개요

- 앞에서 배운 단계별 JDBC 프로그래밍 절차에 따라 간단한 입력, 조회 프로그램의 개발을 통해 JDBC 프로그램의 개발과정과 구조에 대한 이해를 높인다.
- 별도 클래스로 개발하는 것이 좋으나 여기서는 JSP와의 연계 구조를 이해하기 위해 JSP에서 스크립트릿으로 개발한다.
- 다음은 일반적인 웹 애플리케이션의 데이터베이스 연결 구조를 보여준다.



[그림 8-42] 웹 애플리케이션과 데이터베이스 연결 구조



## 05. [기본실습] JDBC프로그래밍 : MySQL 연동 JSP 프로그래밍

### ■ 화면 구성



[그림 8-43] 예제 프로그램 화면 구성



[그림 8-47] 실행된 최종 결과 화면

### ■ 데이터베이스 테이블 생성

- 예제에 사용되는 테이블은 이름과 이메일로 간단하게 구성된다.
- 별도의 키는 없으며 자료형은 모두 문자열 형인 VARCHAR 이다.
- 테이블 생성은 MySQL Workbench 를 이용한다.

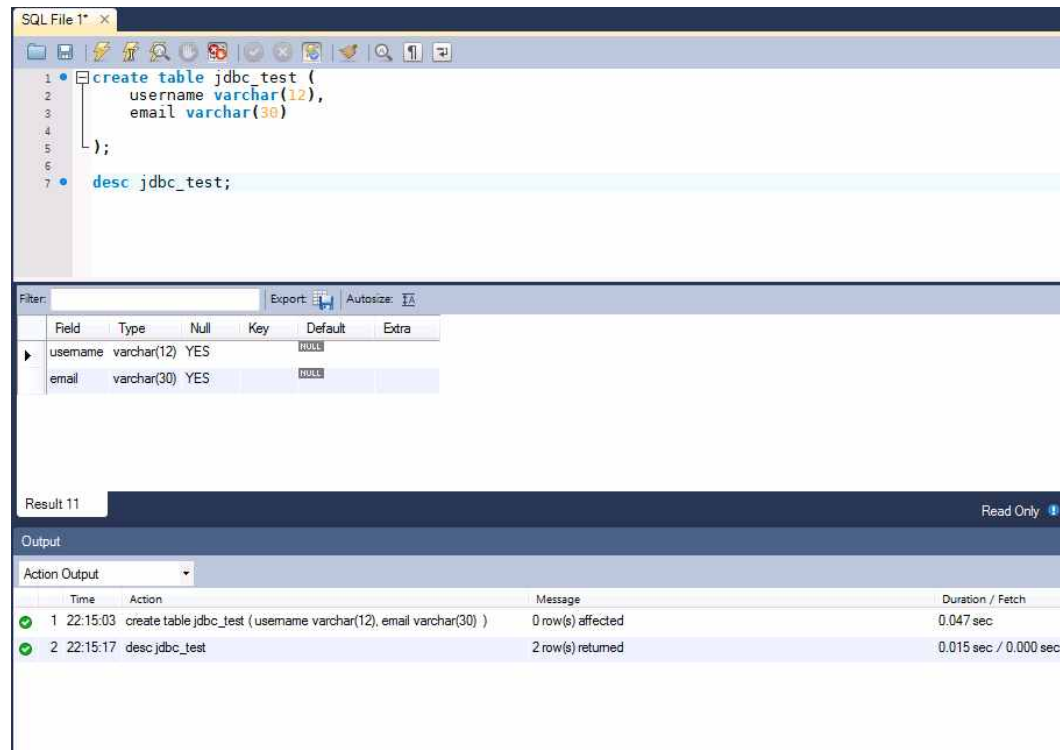
항목	칼럼 이름	자료형
이름	username	VARCHAR(12)
이메일	email	VARCHAR(30)

[그림 8-44] jdbc\_test 테이블 구조

## 05. [기본실습]JDBC프로그래밍 : MySQL 연동 JSP 프로그래밍

- [실습] 테이블 생성(jdbc\_sql.txt) – [교재 p.324 참고](#)
- 주요 소스코드 분석

```
01 CREATE TABLE jdbc_test (  
02     username VARCHAR(12),  
03     email VARCHAR(30)  
04 );
```



[그림 8-45] 테이블 생성 성공 확인

## 05. [기본실습]JDBC프로그래밍 : MySQL 연동 JSP 프로그래밍

- [실습] 기본 화면 구현(jdbctest.jsp) – [교재 p.325 참고](#)
- 주요 소스코드 분석
  - 기본 화면은 HTML form 으로 두개의 입력항목을 가진다.
  - input type 의 name 의 각각 username 과 email 로 한다.

```
08 <form name=form1 method=post>
09 등록이름 : <input type=text name=username>
10 주소 : <input type=text name=email size=20>
11 <input type=submit value="등록">
12 </form>
```

## 05. [기본실습]JDBC프로그래밍 : MySQL 연동 JSP 프로그래밍

### ■ [실습] 데이터베이스 연결 구현(jdbctest.jsp) – 교재 p.325 ~ 326 참고

### ■ 주요 소스코드 분석

- 스크립트릿을 이용해 JDBC 드라이버 클래스 이름과 접속 URL을 변수로 설정한다.
- Class.forName() 으로 드라이버를 로드하고 DriverManager.getConnection() 을 이용해 DB 연결을 만든다.

```
10 // 데이터베이스 연결 관련 정보를 문자열로 선언
11 String jdbc_driver = "com.mysql.jdbc.Driver";
12 String jdbc_url = "jdbc:mysql://localhost/jspdb";
13 // String url = "jdbc:oracle:thin:@localhost:1521:JAVA"; <-oracle
14 try{
15     // JDBC 드라이버 로드
16     Class.forName(jdbc_driver);
17
18     // 데이터베이스 연결 정보를 이용해 Connection 인스턴스 확보
19     Conn = DriverManager.getConnection(jdbc_url,"jspbook","1234");
```

## 05. [기본실습]JDBC프로그래밍 : MySQL 연동 JSP 프로그래밍

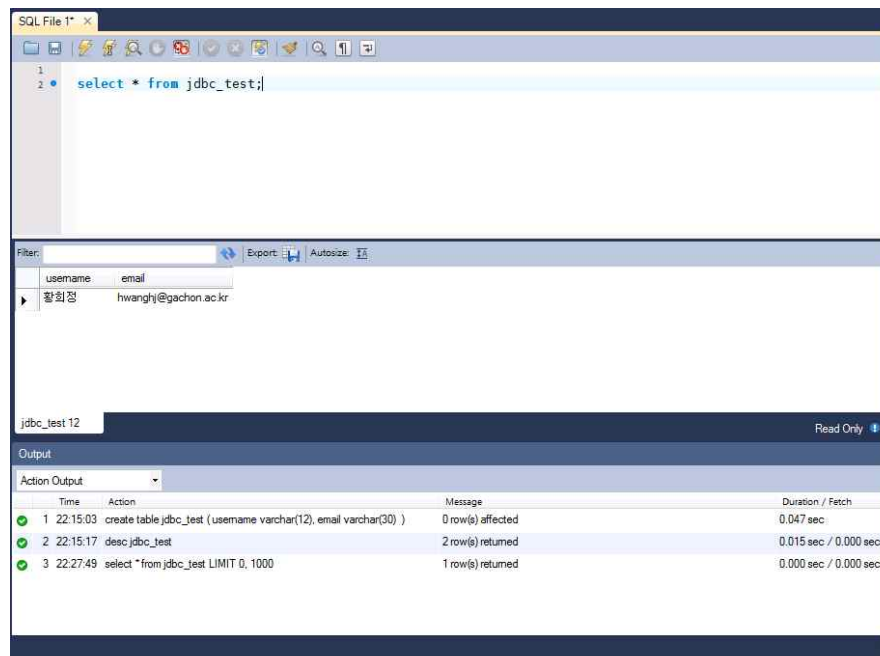
### ■ [실습] 데이터 입력 구현(jdbctest.jsp) – 교재 p.326 ~ 327 참고

### ■ 주요 소스코드 분석

- 기본 쿼리를 만들고 HTML form 에서 입력된 두개의 값을 가져와 PreparedStatement 를 구성한다.

```
21 // Connection 클래스의 인스턴스로 부터 SQL문 작성을 위한 Statement 준비
22 String sql = "insert into jdbc_test values(?,?)";
23 pstmt = conn.prepareStatement(sql);
24 pstmt.setString(1,request.getParameter("username"));
25 pstmt.setString(2,request.getParameter("email"));
```

- 쿼리 실행은 pstmt.executeUpdate() 를 이용한다.



[그림 8-46] 입력된 데이터 확인

## 05. [기본실습]JDBC프로그래밍 : MySQL 연동 JSP 프로그래밍

### ■ [실습] 데이터 출력 구현(jdbctest.jsp) – 교재 p.330 ~ 331 참고

### ■ 주요 소스코드 분석

- select 문을 이용해 DB 에 저장된 값을 가져와 출력한다.
- ResultSet 객체를 이용하고 getXxx() 메서드를 통해 데이터를 가지고 온다.

```
53 // select 문장을 문자열 형태로 구성한다.  
54 String sql = "select username, email from jdbc_test";  
55  
56 pstmt = conn.prepareStatement(sql);  
57  
58 // select를 수행하면 데이터정보가 ResultSet 클래스의 인스턴스로 반환된다.  
59 ResultSet rs = pstmt.executeQuery();
```

- for 루프를 돌며 가져온 데이터를 HTML과 조합하여 출력한다.

```
62 // 마지막 데이터까지 반복한다.  
63 while(rs.next()) {  
64     out.println(i+" : "+rs.getString(1)+" , "+rs.getString("email")+"<BR>");
```