16

JDBC 프로그래밍

Objectives

- 데이터베이스 개념을 이해한다.
- JDBC 구조를 이해한다.
- MySQL을 간단히 설치하고 활용할 줄 안다.
- 데이터베이스 생성 · 접속, 테이블 생성, 레코드 추가 · 삭제, 데이터 검색 · 수정 등을 위한 SQL 문을 이해한다.
- JDBC를 이용한 데이터베이스 프로그래밍을 작성해본다.

CHAPTER 16

JAVA PROGRAMMING

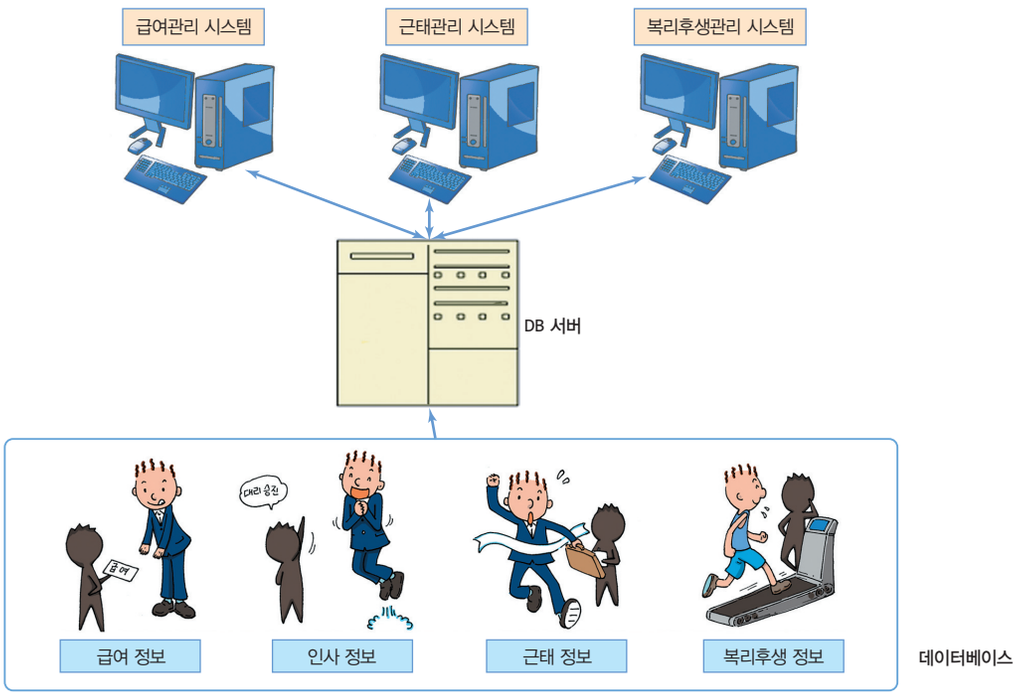
JDBC 프로그래밍

16.1 데이터베이스

데이터베이스란?

데이터베이스

데이터베이스(database)는 여러 응용 시스템들의 통합된 정보들을 저장하여 운영할 수 있는 공용 데이터들의 집합이다. 데이터베이스는 대규모의 데이터를 효율적으로 저장, 검색, 갱신할 수 있도록 데이터를 고도로 조직화하여 저장한다.



[그림 16-1] 기업 내의 여러 소프트웨어 시스템은 데이터베이스와 관련되어 있다.

예를 들어 회사의 직원을 관리하는 시스템을 생각해보자. [그림 16-1]과 같이 각 직원에 대한 인사 정보, 근태 정보, 급여 정보, 복리후생 정보들이 있을 수 있다. 이 정보들은 서로 연관되어 있어 만약 한 직원이 퇴사를 하면 모든 정보에 영향을 미친다. 또한 회사 내의 각 부서에서 사용하는 서로 다른 다수의 프로그램이 동시에 접속하여 이러한 정보들을 사용할 수 있으므로 데이터베이스는 정보들을 고도로 조직화하여 데이터의 무결성이 유지될 수 있도록 관리하여야 한다.

DBMS

DBMS

그러면 [그림 16-1]과 같이 데이터베이스에 대해 서로 다른 여러 소프트웨어 시스템에 동시 접근할 때 이를 잘 관리할 수 있는 방법은 무엇인가? 데이터베이스를 관리하는 소프트웨어 시스템을 DBMS(DataBase Management System)라고 한다. DBMS는 다수의 사용자들이 동시에 데이터베이스를 사용할 수 있도록 관리한다. 대표적인 DBMS로는 오라클(Oracle), 마이크로소프트의 SQL Server, MySQL, IBM의 DB2 등이 있다. 데이터베이스를 이용하는 모든 자바 응용프로그램 역시 반드시 DBMS에게 데이터 처리를 요청하며, 이 작업은 DBMS에 의해 일관성 있게 처리된다.

데이터베이스의 종류

데이터베이스의 종류에는 크게 관계형 데이터베이스(relational database), 객체 지향 데이터베이스(object oriented database)가 있다.

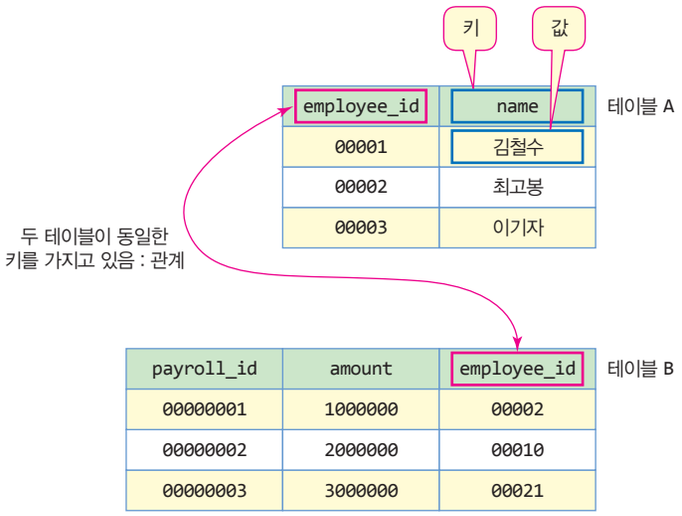
● 관계형 데이터베이스

테이블
열

관계형 데이터베이스는 [그림 16-2]와 같이 데이터들이 다수의 테이블로 구성된다. [그림 16-2]에서 첫 번째 테이블은 두 개의 열(column)로 구성되며, 두 번째 테이블은 3개의 열로 구성된다. 첫 번째 테이블은 사원 ID(employee_id)와 이름(name)만을 가지고 두 번째 테이블은 사원 ID(employee_id)와 급여대상 ID(payroll_id), 급여(amount)로 구성된다.

행
키
값

테이블의 각 행(row)은 하나의 레코드(record)이며 각 테이블은 키(key)와 값(value)들의 관계로 표현된다. 키는 테이블의 열 이름이며, 키 중에서 특정 레코드를 검색하거나 레코드들을 정렬할 때 우선적으로 참조되는 키를 일차 키(primary key)라고 한다. 여러 테이블 간에는 [그림 16-2]의 employee_id와 같이 공통된 이름의 열을 포함할 수 있으며 이런 경우 서로 다른 테이블 간에 관계(relation)가 성립된다. 현재 사용되는 대부분의 데이터베이스는 관계형 데이터베이스이며 JDBC API도 관계형 데이터베이스에 대한 API이다.



[그림 16-2] 관계형 데이터베이스 구조

● 객체 지향 데이터베이스

객체 지향 데이터베이스는 객체 지향 프로그래밍에 쓰이는 것으로, 정보를 객체의 형태로 표현하는 데이터베이스이며 오브젝트 데이터베이스(object database)라고도 부른다. 장점은 객체 모델이 그대로 데이터베이스에도 적용되므로 응용프로그램의 객체 모델과 데이터베이스의 모델이 부합하는데 있다. 그러나 현재 관계형 데이터베이스로 된 DBMS와 그에 따른 응용프로그램들이 주류를 이루고 있어 객체 지향 데이터베이스는 틈새시장을 차지하고 있다.

SQL

SQL

SQL(Structured Query Language)은 관계형 데이터베이스 관리 시스템(DBMS)에서 데이터베이스 스키마 생성, 자료의 검색, 관리, 수정, 그리고 데이터베이스 객체 접근 관리 등을 위해 고안된 언어이다. 이 언어는 다수의 데이터베이스 관련 프로그램들이 표준으로 채택하고 있다. 자바 응용프로그램에서도 SQL로 작성된 간단한 데이터베이스 처리 명령어를 DBMS에게 보내어 데이터베이스 처리를 지시한다.

JDBC

JDBC

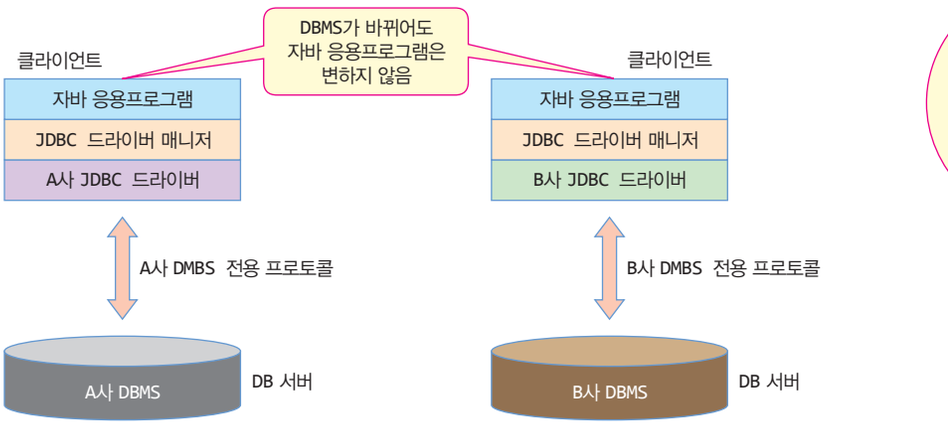
JDBC는 관계형 데이터베이스에 저장된 데이터를 접근 및 조작할 수 있게 하는 자바 API이다. JDBC는 자바 응용프로그램이 다양한 DBMS에 대해 일관된 API로 데이터베이스 연결, 검색, 수정, 관리 등을 할 수 있게 한다. 그러므로 자바 응용프로그램 개발자

는 DBMS의 종류에 관계없이 JDBC API만을 이용하면 된다.

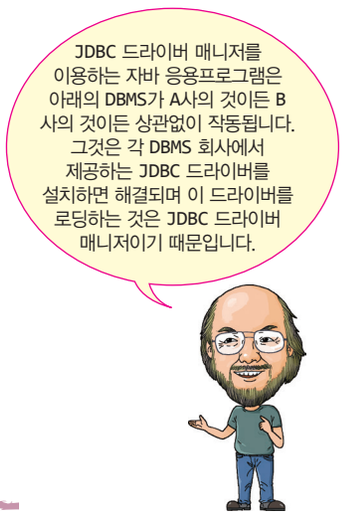
[그림 16-3]과 같이 일반적으로 DBMS를 제공하는 회사에서 JDBC 드라이버를 제공하며 자바 응용프로그램에서는 JDBC 드라이버를 JDBC 매니저를 통해 로드하여 사용한다. 따라서 사용하는 DBMS가 바뀌어도 이에 따른 JDBC 드라이버만 로드하면 되므로 자바 프로그램에는 DBMS 변경에 따른 프로그램 수정이 필요 없다. JDBC 드라이버 매니저는 자바 API에서 지원하는 클래스이다.

독자들의 이해를 위해 이들 용어를 다시 한 번 정리하였다.

- JDBC 드라이버 매니저: 자바 API에서 지원하며 DBMS를 접근할 수 있는 JDBC 드라이버 로드
- JDBC 드라이버: DBMS마다 고유한 JDBC 드라이버를 제공하며, JDBC 드라이버와 DBMS는 전용 프로토콜로 데이터베이스 처리
- DBMS: 데이터베이스 관리 시스템으로 데이터베이스 생성, 삭제, 데이터 생성, 검색, 삭제 등을 전담하는 소프트웨어 시스템



[그림 16-3] JDBC 구조



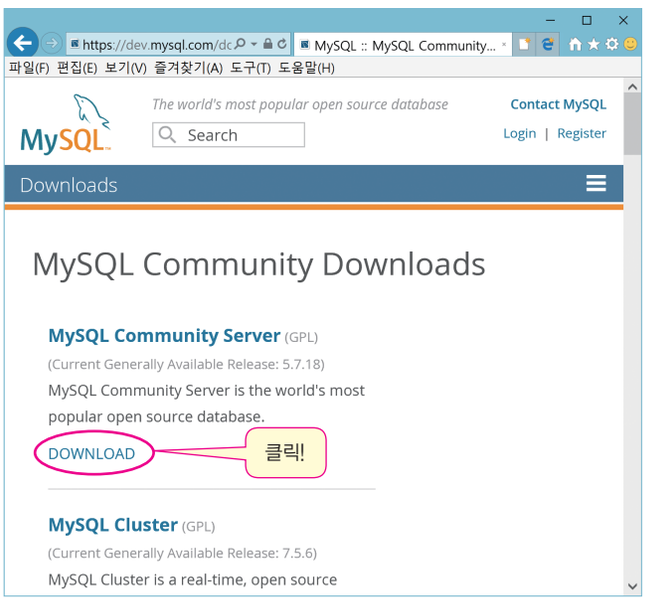
1. 오라클, SQL Server, MySQL 등을 무엇이라고 부르는가?
2. 다수의 테이블로 구성되고 테이블의 관계를 통해 데이터를 관리하는 데이터베이스 종류는 무엇인가?
3. 자바로 작성된 데이터베이스 관리 프로그램이 다양한 종류의 DBMS에 관계없이 작성될 수 있는 이유는 무엇인가?

16.2 MySQL

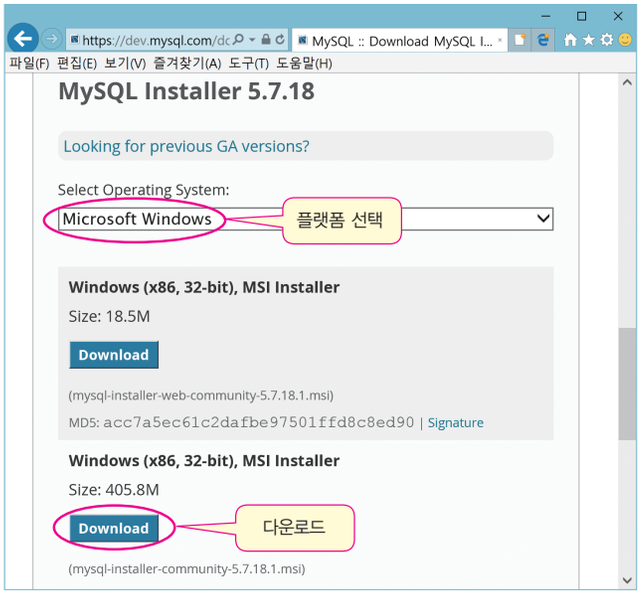
MySQL은 무료로 다운받아 설치하여 간편하게 사용할 수 있는 관계형 DBMS 중의 하나이다. 현재 MySQL은 데이터베이스 시스템으로 많이 사용되고 있다. 이 장에서는 MySQL을 이용하여 데이터베이스 응용프로그램 개발을 실습해보기로 한다.

다운로드

MySQL은 http://www.mysql.com/downloads/에서 다운로드 받을 수 있다. 다운로드 페이지에는 여러 버전이 있는데 MySQL Community Edition이 무료로 사용할 수 있는 버전이므로 이것을 클릭하면 [그림 16-4]와 같은 페이지로 이동한다. 여기서 Recommended Download인 MySQL Installer를 선택하면 [그림 16-5]와 같은 페이지로 이동하며 여기서 마이크로소프트 윈도우 플랫폼을 선택하여 다운로드를 시작한다.



[그림 16-4] MySQL 다운로드 사이트에서 다운로드 선택

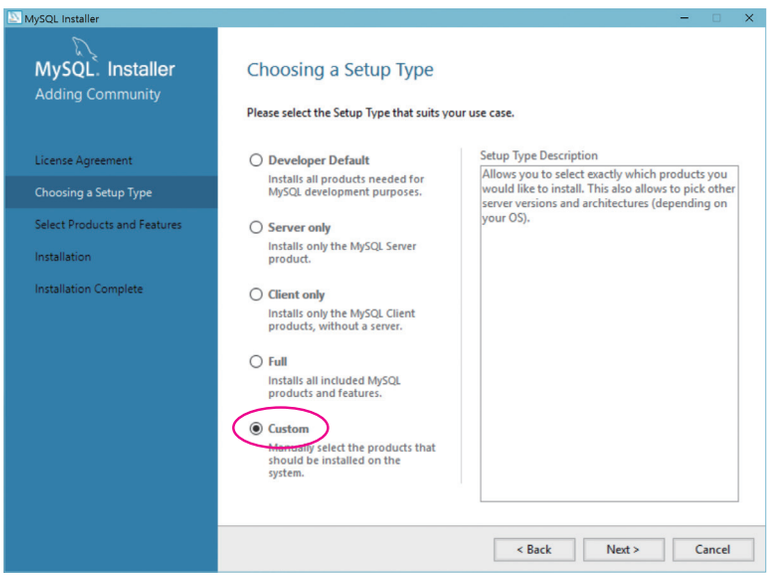


[그림 16-5] 마이크로소프트 윈도우 플랫폼 선택 후 다운로드

설치 및 설정

● 설치 타입을 Custom 타입으로 선택

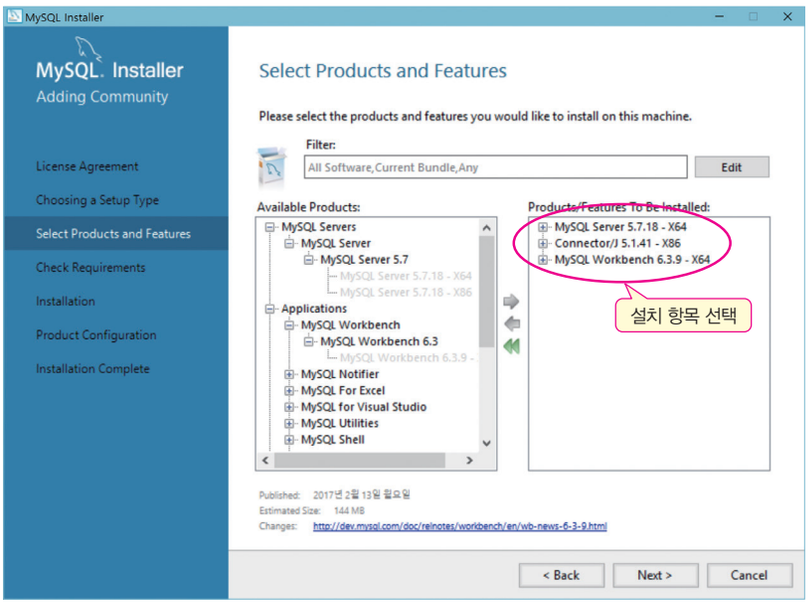
설치를 시작한 후에는 [그림 16-6]과 같이 "Custom" 타입을 선택하여 설치를 진행한다.



[그림 16-6] 설치 타입을 "Custom"으로 선택

● 설치 항목 선택

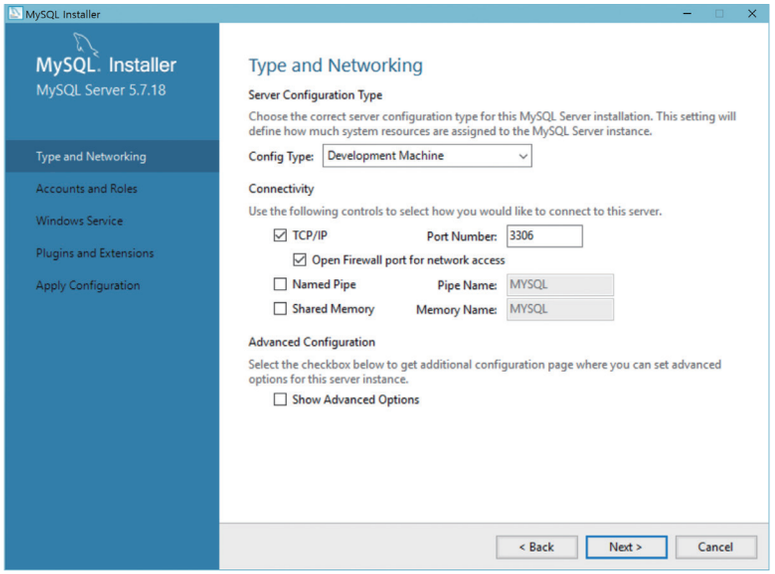
Custom 타입 설치에서는 [그림 16-7]과 같이 설치 항목을 선택할 수 있다. 여기서 실습을 위하여 필요한 항목은 MySQL Server, Connector/J, MySQL Workbench이므로 이 세 가지 항목을 플랫폼 타입에 맞게 선택한다.



[그림 16-7] 설치 항목 선택

● MySQL 서버 설정

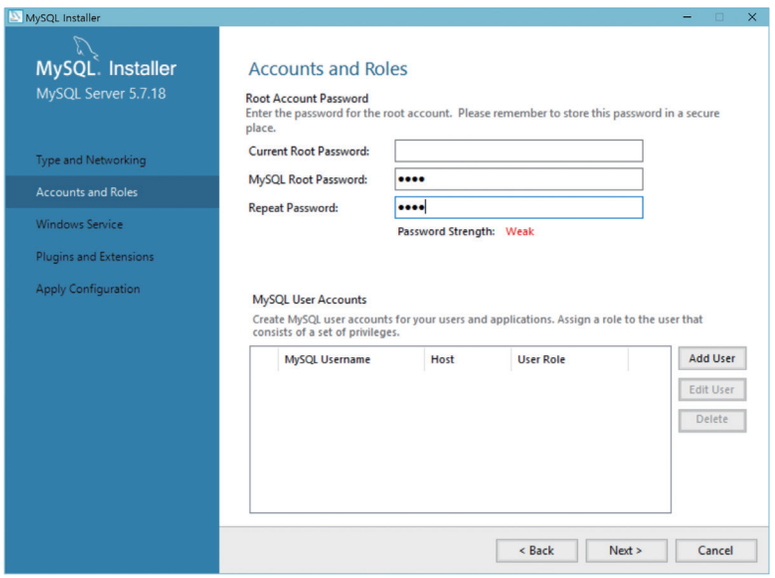
각 항목의 설치가 완료된 후에는 [그림 16-8]과 같이 MySQL Server 설정 화면이 나타난다. 화면에서 MySQL 서버는 TCP/IP 프로토콜에 3306번 포트를 사용하여 클라이언트의 접속을 받는 것이 기본 설정으로 되어 있다. Next 버튼을 눌러 기본 설정을 선택한다.



[그림 16-8] 서버 설정

● MySQL 서버의 루트 계정 암호 설정

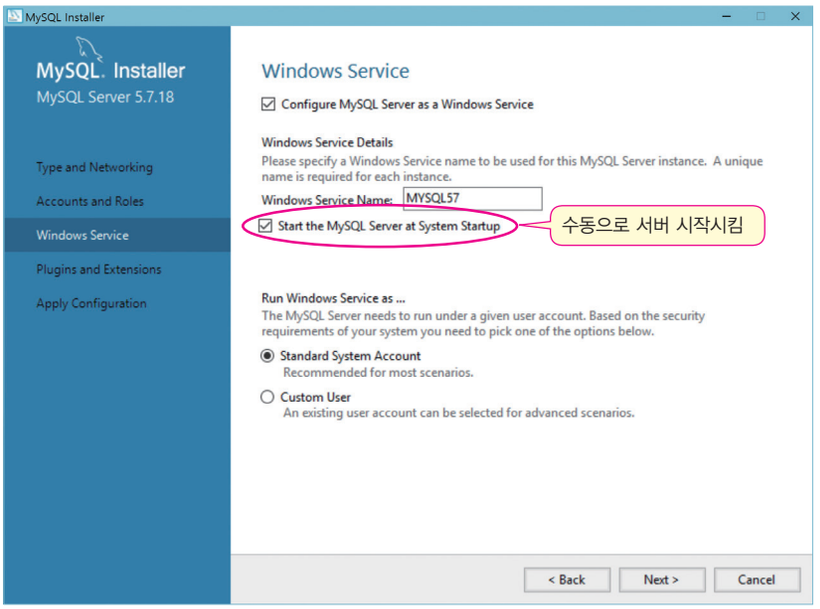
서버 설정을 하고 나면 [그림 16-9]와 같이 서버의 루트 계정 암호를 설정하는 화면이 나타난다. 실습용으로 서버를 설치 및 설정하므로 기억하기 쉬운 간단한 암호를 입력하도록 하라.



[그림 16-9] 서버 루트 계정 암호 설정

● MySQL 서버를 윈도우 서비스로 등록

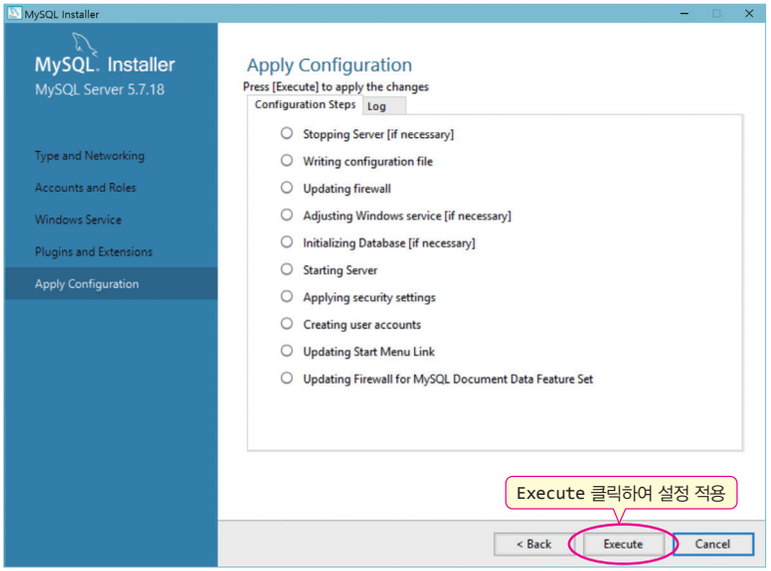
암호 설정이 끝나면 [그림 16-10]과 같이 MySQL 서버를 윈도우 서비스에 등록하는 화면이 나타난다. MySQL 서버를 윈도우 서비스로 등록하면, 윈도우가 부팅할 때 MySQL 서버를 자동으로 실행시켜 현재 컴퓨터가 DB 서버로 작동하게 하고, 관리자가 서버의 작동을 잠깐 중지시키거나 재시작 시키는 등 제어할 수 있다. 하지만, MySQL 서버는 기본적으로 시스템 자원을 많이 소모하므로, [그림 16-10]과 같이 필요할 때만 수동으로 시작하도록 설정한다. 그리고 MySQL 서비스의 윈도우 서비스 이름은 MySQL57임을 기억해두기 바란다. 이것은 추후 [그림 16-13]에서와 같이 윈도우 서비스 리스트에서 찾을 때 사용된다.



[그림 16-10] 수동으로 MySQL 서버를 시작시키도록 윈도우 서비스로 설정

● 설정 적용

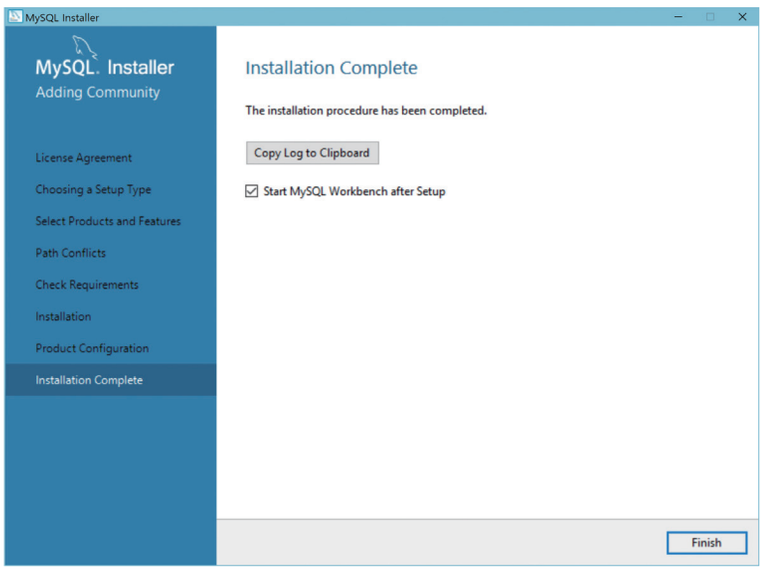
윈도우 서비스 설정이 끝나면 플러그인과 익스텐션을 설정하는 화면이 나타나는데 이 화면에서는 그냥 Next 버튼을 눌러 다음으로 넘어가면 [그림 16-11]과 같이 설정 적용 화면이 나타난다. 여기서 Execute 버튼을 눌러 설정을 적용하도록 한다.



[그림 16-11] 설정 적용

● 설치 완료

설정 적용이 완료된 후 Product Configuration 화면이 나타나는데 그냥 Next 버튼을 눌러 다음으로 넘어가면 최종적으로 [그림 16-12] 화면이 나타난다. Finish 버튼을 누르면 앞서 설치했던 MySQL Workbench가 실행된다.



[그림 16-12] 설치 완료

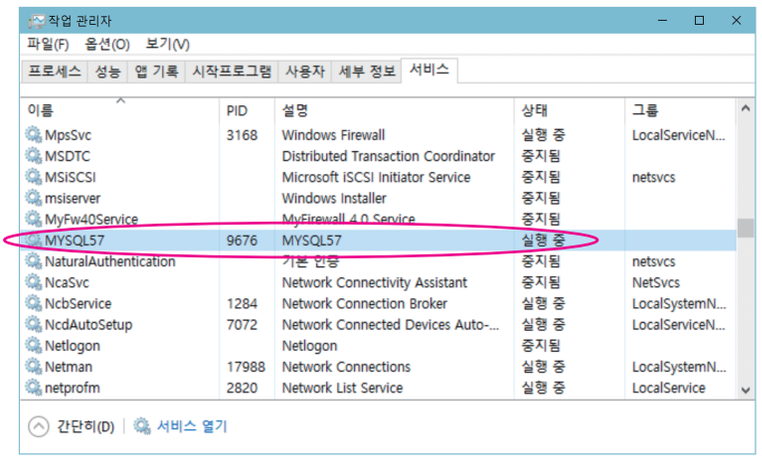
16.3 MySQL Workbench를 이용한 데이터베이스 활용

자바로 MySQL을 이용하는 데이터베이스 프로그래밍을 해보기 전에, MySQL Workbench를 이용하여 데이터베이스와 테이블을 만들고, 레코드를 쓰고, 검색하는 등의 MySQL 활용을 직접해보자.

MySQL 서버 실행

MySQL 서버는 설치 과정 중 [그림 16-10]의 과정에서 MySQL 서버를 윈도우 서비스로 설정하고, '수동으로 시작'시키도록 지정하였기 때문에, MySQL 서버가 실행되게 하려면 작업 관리자에서 서비스를 시작시켜야 한다.

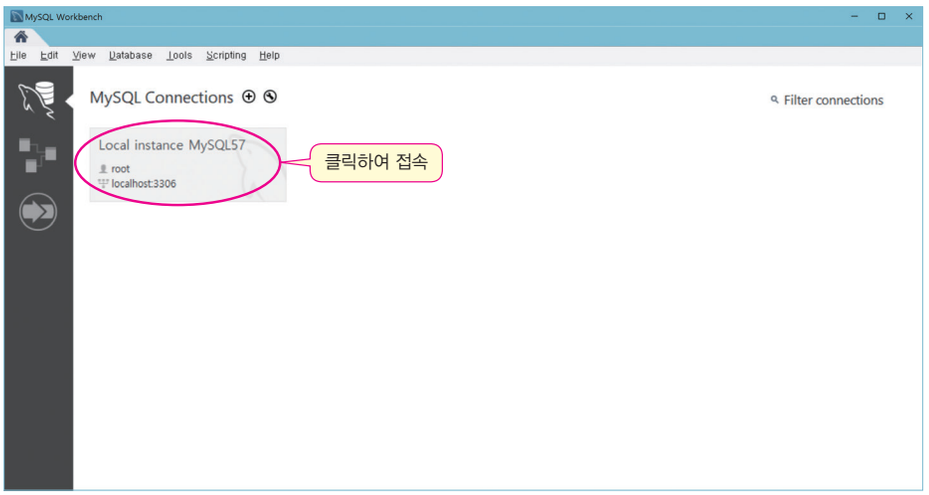
자, Ctrl-Alt-Del을 누른 후 작업 관리자를 선택하자. 작업 관리자를 실행하여 [그림 16-13]과 같이 MySQL 서버의 실행 상태를 확인해보자. MySQL 서버는 서비스 관리자에서 수동으로 실행 시작 및 중지할 수 있으며, MySQL 서버를 사용하지 않을 때는 불필요한 시스템 자원을 낭비하지 않도록 서버를 중지시킬 수 있다. [그림 16-13]은 당초 MySQL57의 사태가 '중지됨'이었지만, '실행 중'으로 바꾸어 MySQL 서버의 서비스를 실행시킨 결과이다.



[그림 16-13] MySQL 서버 실행 상태 확인, 실행 시작 및 중지

서버 접속

[그림 16-14]는 MySQL Workbench의 실행 화면이다. 첫 화면에 설치 및 설정을 종료한 서버의 인스턴스가 표시된다. 이 인스턴스를 클릭하면 서버에 접속하게 된다. Workbench 프로그램은 MySQL 서버를 이용하는 응용프로그램의 하나이다. 그러므로 Workbench도 서버에 접속할 필요가 있는 것이다.

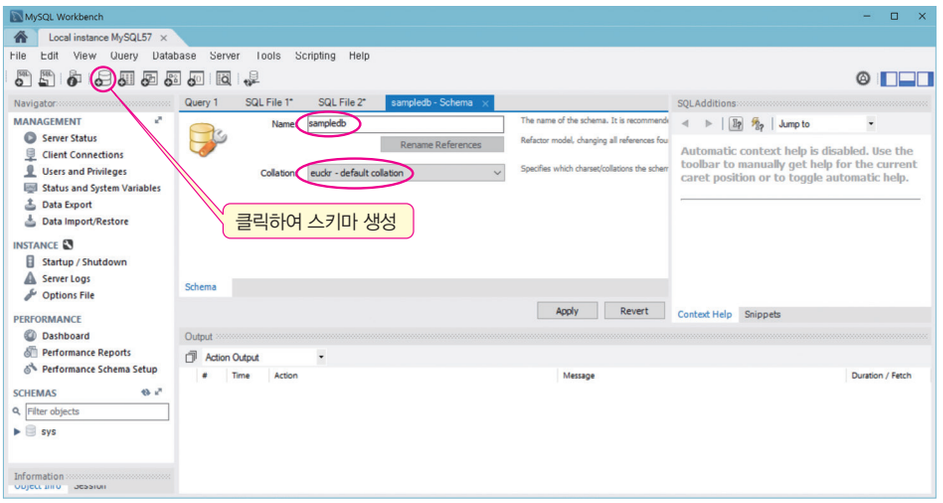


[그림 16-14] Workbench 실행 화면

자바 응용프로그램에서 추후 JDBC를 이용하여 MyDQL DBMS에 접근하여 데이터를 조작하려면 먼저 데이터베이스가 만들어져 있어야 한다. Workbench를 이용하여 데이터베이스를 생성해 보자.

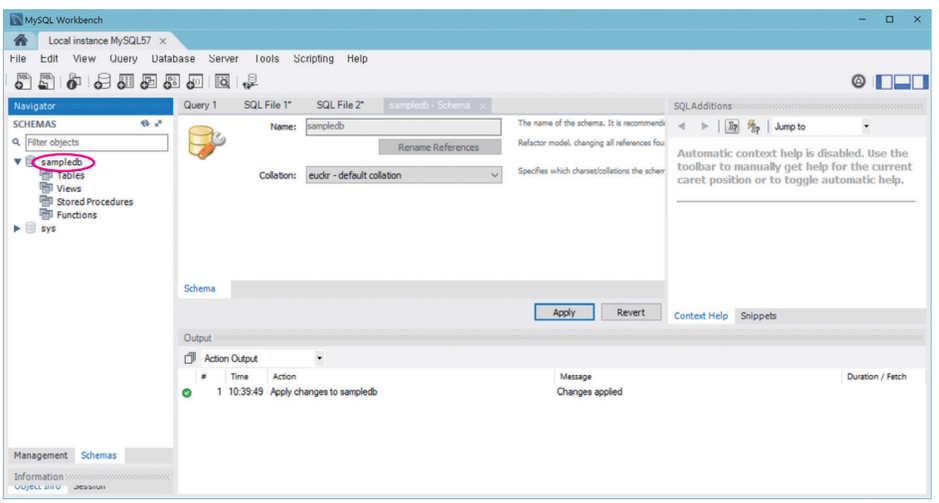
스키마 생성

메뉴에서 버튼을 클릭하면 [그림 16-15]와 같이 스키마를 생성하는 화면이 나타난다. 이름은 sampledb로 하고 Collation에는 한글을 사용할 수 있도록 euckr - default collation을 선택한 후 Apply 버튼을 클릭하여 스키마를 생성한다.



[그림 16-15] 스키마 생성

스키마 생성이 완료되면 [그림 16-16]과 같이 Navigator탭 하단에 sampledb가 표시되는 것을 볼 수 있다.



[그림 16-16] 스키마 생성 화면

테이블 생성

관계형 데이터베이스에서는 행과 열로 구성된 테이블 단위로 데이터가 저장되므로 데이터를 저장하기 위해서는 테이블을 만들어야 한다. 본 예제에서는 〈표 16-1〉과 같은 구조를 갖는 student라는 이름의 테이블을 생성해보자. 각 행은 다음과 같은 의미를 가진다.

- 첫 번째 행은 열의 이름
- 두 번째 행은 데이터 타입과 크기를 표시. char, varchar 등의 타입은 자바의 타입이 아닌 MySQL에서 사용되는 타입임에 주의

〈표 16-1〉
student 테이블 구조

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

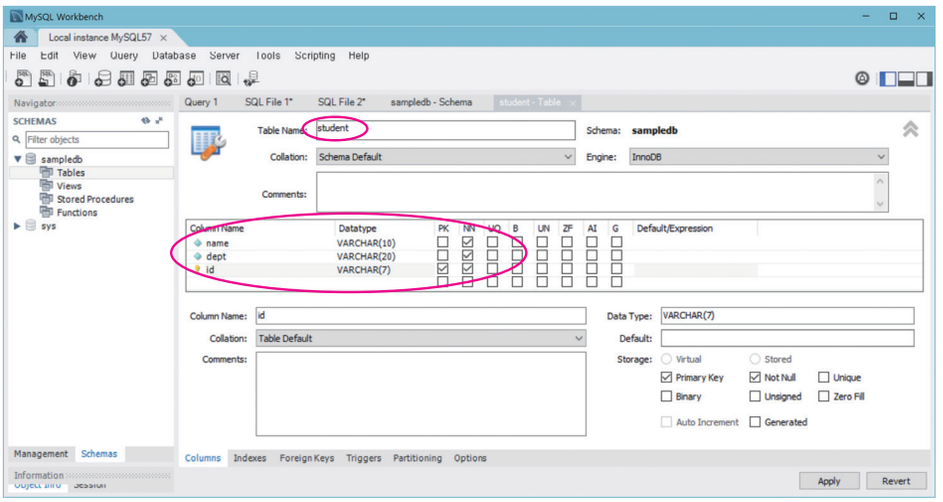
MySQL의 자세한 데이터 타입은 MySQL 사이트에서 문서를 참조하기 바란다.

이제 Workbench에서 테이블을 생성해보자. 앞서 생성한 sampledb를 더블 클릭하면 이 스키마가 선택된다. 이 상태에서 메뉴의 버튼을 누르면 [그림 16-17]과 같은 테이블 생성 화면이 나타난다. 그림에서와 같이 테이블 이름은 student를 입력하고 〈표 16-1〉의 테이블 구조에 맞도록 밑에 column도 같이 생성한다.

column을 생성하는 부분에 체크 박스가 여러 개 있으며 이 예제에서는 PK와 NN 옵션을 사용하였다. 각각이 의미하는 바는 다음과 같다.

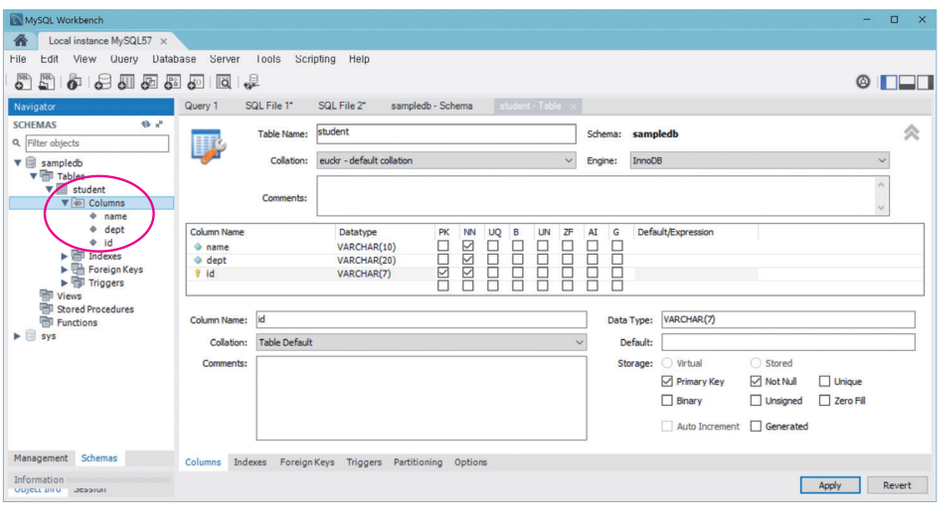
- NN: "not null"을 나타내며 해당 열의 값이 null이 될 수 없음을 의미
- PK: "primary key"를 나타내며 id를 일차 키로 지정

모두 입력을 하였으면 Apply 버튼을 클릭하며 테이블을 생성한다.



[그림 16–17] 테이블 생성

테이블 생성이 완료되면 [그림 16–18]과 같이 student 테이블 밑에 각 column들이 생성된 것을 볼 수 있다.



[그림 16–18] 생성된 테이블

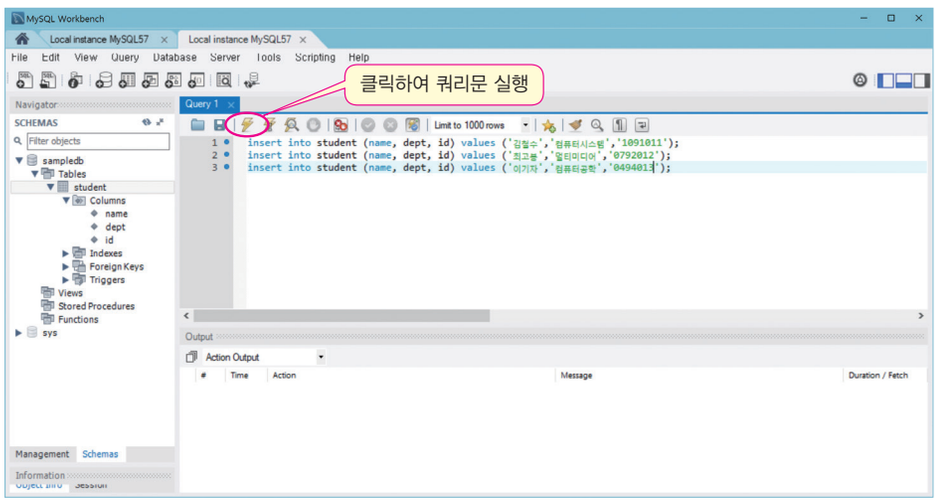
레코드 추가

테이블이 생성되었으면 데이터를 기록할 수 있다. 테이블에는 레코드 단위로 데이터를 추가하는데, 레코드 추가는 Workbench에서 Data Import Wizard를 이용할 수도 있고, SQL 쿼리문을 이용하여 추가할 수도 있다. 여기서는 쿼리를 이용하여 데이터를 추가해보자. 데이터 추가는 다음과 같이 insert 명령을 사용한다.

```
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 값들을 콤마로 구분하여 나열
- 문자 타입의 데이터는 단일 인용 부호로 묶어서 표시함에 유의

이 명령은 student 테이블에 한 레코드를 추가하는데, name, dept, id 필드의 값을 '김철수', '컴퓨터시스템', '1091011'로 각각 입력한다. Workbench에서 쿼리를 실행하려면 File 메뉴의 New Query Tab을 선택하여 쿼리탭을 열어야 한다. [그림 16-19]는 쿼리탭에 쿼리문을 입력한 후 실행시킨 화면이다.



[그림 16-19] 레코드 추가

[그림 16-19]와 같이 3개의 레코드를 추가한 후 student 테이블 구조는 〈표 16-2〉와 같이 된다.

〈표 16-2〉
3개의 레코드가 추가된 student 테이블

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

데이터 검색

데이터베이스로부터 데이터 검색은 다음과 같은 select 명령을 이용한다.

```
select name, dept, id from student where dept='컴퓨터공학';
```

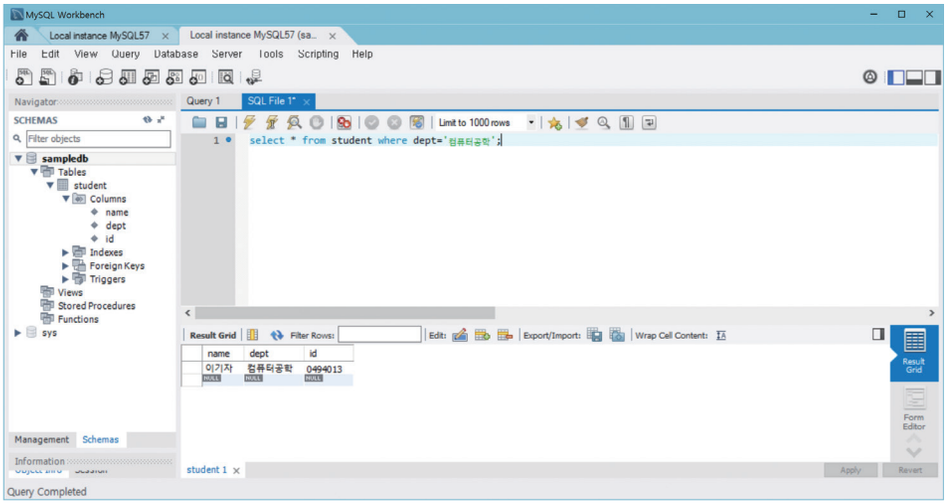
- select 다음에는 데이터를 추출할 열 이름을 콤마로 분리하여 나열
- 모든 열에 대해 데이터를 추출할 때는 *를 열 이름 대신 사용
- from 다음에 테이블 이름을 지정
- where 다음에 검색 조건 지정. 위의 예에서 dept 값이 '컴퓨터공학'인 레코드 검색
- where는 생략 가능

이 select 명령은 student 테이블에서 dept 필드의 값이 '컴퓨터공학'인 레코드를 찾아 name, dept, id 필드의 값을 출력할 것을 지시하는 명령이다.

student 테이블의 모든 레코드를 검색하여 출력하도록 지시하는 명령은 다음과 같다.

```
select * from student;
```

[그림 16-20]은 select 명령을 실제 입력하여 데이터를 검색하는 화면이다.



[그림 16-20] select 명령을 이용한 데이터 검색

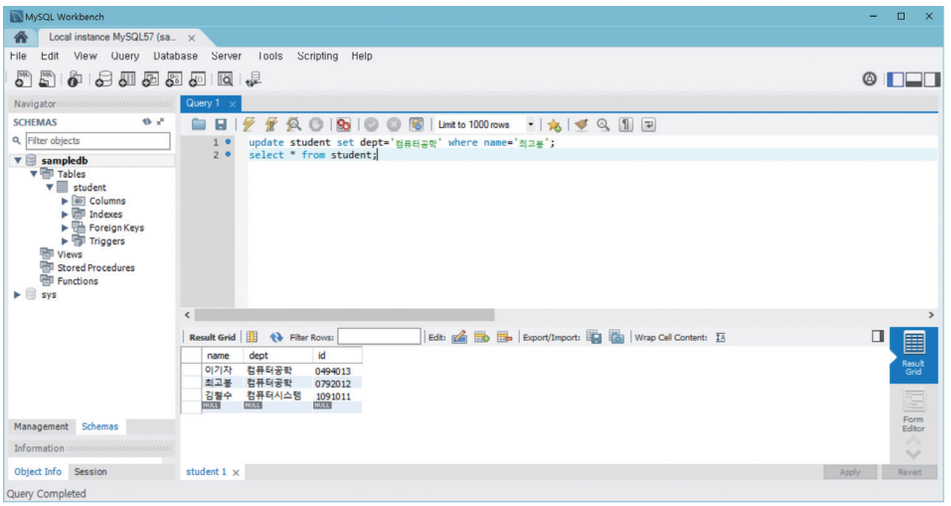
데이터 수정

데이터 수정은 다음과 같은 update 명령을 이용한다.

```
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

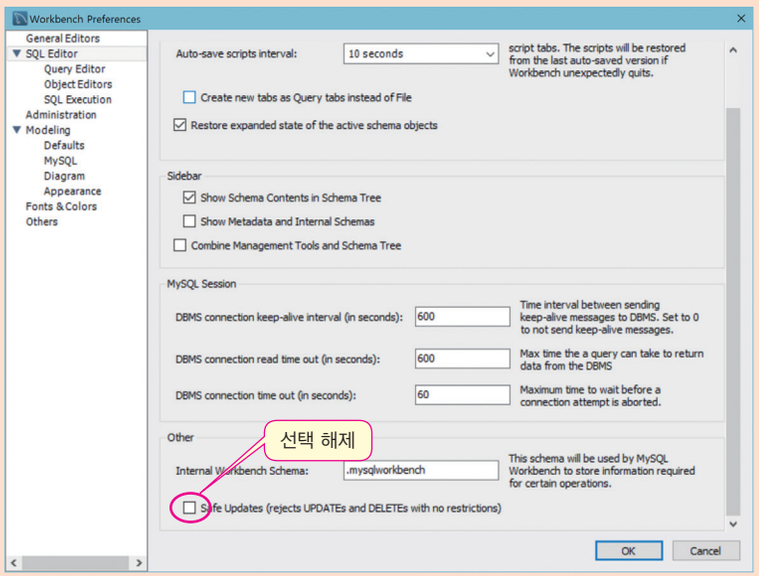
이 update 명령은 student 테이블의 name 필드의 값이 '최고봉'인 레코드를 찾아서 dept 필드의 값을 '컴퓨터공학'으로 변경하도록 지시하는 명령이다. [그림 16-21]은 update 명령을 이용하여 데이터를 수정하는 화면이다.



[그림 16-21] update 명령을 이용한 데이터 수정

Tip Workbench에서 데이터 수정 및 삭제

Workbench에서는 실수로 데이터를 수정 또는 삭제하는 것을 방지하기 위하여 기본적으로 Safe Updates 모드가 설정되어 있다. 이 모드가 설정되어 있으면 위의 예제를 실행할 때 에러가 발생하므로 Edit 메뉴의 Preferences를 선택하여 SQL Editor 항목에서 Safe Updates를 선택 해제 한 후에 데이터베이스에 다시 연결하도록 한다.



[그림 16-22] Safe Updates 모드 해제

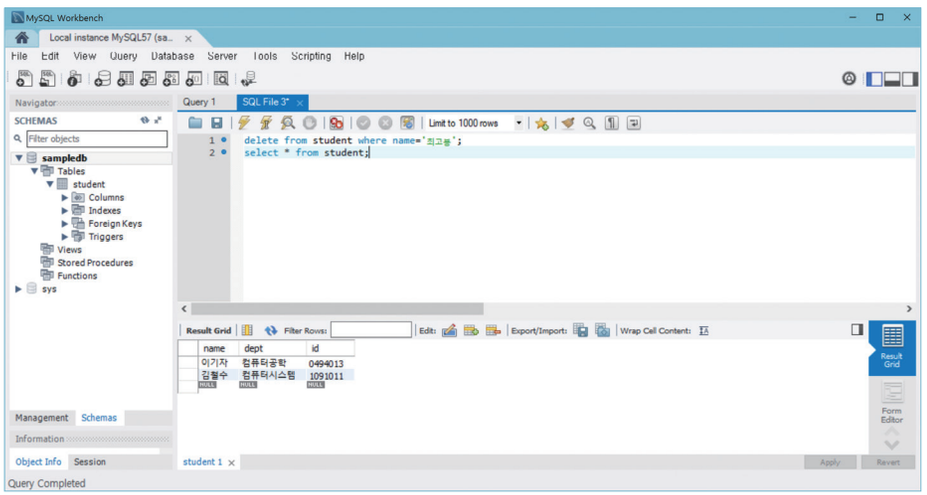
레코드 삭제

테이블에 들어 있는 레코드의 삭제는 다음과 같은 delete 명령을 이용한다.

```
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

이 delete 명령은 student 테이블에서 name 필드의 값이 '최고봉'인 레코드를 찾아 삭제하도록 지시하는 명령이다. [그림 16-23]은 delete 명령을 이용하여 레코드를 삭제하는 화면이다.



[그림 16-23] delete 명령을 이용한 레코드 삭제

CHECK TIME

1 본문에서 만든 student 테이블에서 dept가 '컴퓨터시스템'인 레코드를 찾아 레코드를 모두 삭제하라.

2 다음 표와 같은 테이블 University를 만들고 5개의 레코드를 추가하라.

id	name	location
char(7)	varchar(20)	varchar(100)

16.4 자바의 JDBC 프로그래밍

앞 절에서 MySQL의 명령행 도구를 이용하여 콘솔에서 데이터베이스를 생성하고, 데이터의 추가, 검색, 수정, 삭제 등을 실행하였다. 이제 자바로 데이터베이스를 조작하는 응용프로그램을 작성하는 방법을 설명해보자. JDBC 프로그래밍이란 JDBC API를 이용하여 데이터의 추가, 삭제, 수정, 검색 등을 할 수 있는 자바 응용프로그램을 작성하는 것이다. 이 절의 설명은 MySQL이 설치되어 있고 sampledb가 만들어진 상황에서 계속된다.

데이터베이스 연결 설정

JDBC 프로그래밍의 가장 첫 번째 단계는 데이터베이스와의 연결이다. 데이터베이스와의 연결을 위해서는 우선 데이터베이스의 JDBC 드라이버의 로드가 이루어져야 한다.

● MySQL 서버의 JDBC 드라이버 로드

JDBC 드라이버

JDBC 드라이버를 로드하기 위해 드라이버 클래스 파일을 로드한다. 다음과 같이 자바의 Class 클래스의 forName() 메소드를 이용하면 특정 클래스 파일을 읽어 들일 수 있다.

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

위의 코드는 MySQL의 JDBC 드라이버인 com.mysql.jdbc.Driver 클래스를 로드하여 드라이버 인스턴스를 생성하고 DriverManager에 등록한다. DriverManager 클래스는 [그림 16-3]에서의 JDBC 드라이버 매니저 역할을 하는 클래스이다. JDBC 드라이버의 클래스 이름은 사용하는 DBMS에 따라 다를 수 있으므로 해당 DBMS의 JDBC 드라이버 문서를 참조해야 한다. 만일 로드 중에 JDBC 드라이버가 없으면 ClassNotFoundException 발생하므로 반드시 try/catch 문을 사용한다.

● 자바 응용프로그램과 JDBC의 연결

Connection 객체

DriverManager는 자바 응용프로그램을 JDBC 드라이버에 연결시켜주는 클래스이다. 아래 코드와 같이 DriverManager.getConnection() 메소드를 호출하여 데이터베이스에 연결하고 Connection 객체를 반환한다.

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

- getConnection()에서 jdbc: 이후에 지정되는 URL 형식은 DBMS에 따라 다르므로 JDBC 문서 참조
- MySQL 서버가 현재 동일한 컴퓨터에서 동작하므로 서버 주소를 localhost로 지정
- MySQL의 경우 디폴트로 3306 포트를 사용
- sampledb는 앞서 생성한 DB의 이름
- "root"는 DB에 로그인할 계정 이름이며, ""는 root의 패스워드

예제 16-1 samplesdb 데이터베이스에 연결하는 JDBC 프로그램 작성

JDBC를 이용하여 samplesdb 데이터베이스에 연결하는 자바 응용프로그램을 작성하라. 만약 MySQL 서버가 실행 중이지 않으면 다음과 같은 에러가 발생한다.

```
DB 연결 에러
```

이 경우는 윈도우의 서비스 관리자에서 MySQL 서버를 시작시키면 된다.

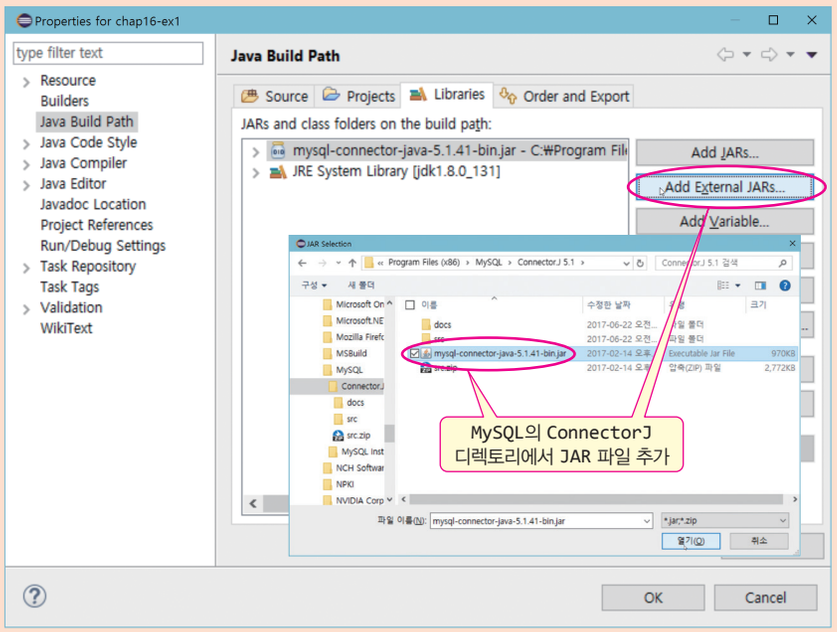
```
1  import java.sql.*;
2
3  public class JDBC_Ex1 {
4     public static void main (String[] args) {
5        try {
6           Class.forName("com.mysql.jdbc.Driver"); // MySQL 드라이버 로드
7           Connection  conn  =  DriverManager.getConnection("jdbc:mysql:
                              // localhost:3306/sampledb", "root","password");
                                   // JDBC 연결, password는 root 계정 패스워드 입력
8           System.out.println("DB 연결 완료");
9        } catch (ClassNotFoundException e) {
10          System.out.println("JDBC 드라이버 로드 에러");
11       } catch (SQLException e) {
12          System.out.println("DB 연결 에러");
13       }
14    }
15 }
```

실행 결과

```
DB 연결 완료
```

Tip JDBC 드라이버 경로 지정

예제 16-1을 실행하면 대부분은 "JDBC 드라이버 로드 에러"가 발생할 것이다. 이는 실행할 때 JDBC 드라이버를 찾지 못해서 발생하는 것이므로 JDBC 드라이버의 경로를 지정해주어야 한다. 명령행에서 실행할 때는 -classpath 옵션 다음에 JDBC 드라이버의 경로를 지정해주어야 하며 이클립스에서는 [그림 16-24]와 같이 패키지 Properties의 Java Build Path의 Libraries 탭에서 Add External JARs 버튼을 눌러 드라이버를 추가해준다.



[그림 16-24] 이클립스에서 JDBC 드라이버 라이브러리 추가

데이터베이스 사용

Statement 클래스
ResultSet 클래스

자바에서 데이터베이스에 연결 후에는 16.3절에서 설명한 MySQL 명령행 도구에서 사용한 SQL 문을 똑같이 사용하여 데이터베이스에 접근한다. 자바에서 SQL문을 실행하기 위해서는 Statement 클래스를 이용하고, SQL문 실행 결과를 얻어오기 위해서는 ResultSet 클래스를 이용한다. Statement 클래스에서 자주 사용되는 메소드는 〈표 16-3〉과 같다.

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

〈표 16-3〉
Statement 클래스 메소드

데이터를 검색하기 위해서는 executeQuery() 메소드를 사용하고, 추가, 수정, 삭제와 같이 데이터 변경은 executeUpdate() 메소드를 이용한다.

executeQuery()
executeUpdate()

ResultSet 객체는 현재 데이터의 행(레코드 위치)을 가리키는 커서(cursor)를 관리한다. 초기 값은 첫 번째 행 이전을 가리키도록 되어있다. 따라서 ResultSet 클래스는 주로 커서의 위치와 관련된 메소드와 레코드를 가져오는 메소드를 제공한다. ResultSet 클래스에서 자주 사용되는 메소드는 〈표 16-4〉와 같다.

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 columnLable)	Xxx는 해당 데이터 타입을 나타내며 현재 행에서 지정된 열 이름(columnLable)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.

〈표 16-4〉
ResultSet 클래스 메소드

데이터 검색

● 테이블의 모든 데이터 검색

student 테이블의 모든 데이터를 검색하는 코드는 다음과 같다.

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

검색된 결과는 rs에 들어 있다.

● 특정 열만 검색

테이블에서 특정 열만 검색하고 싶은 경우는 '*' 대신에 검색할 열의 이름을 콤마로 분리하여 나열한다. 다음은 student 테이블에서 모든 레코드에 대해 name과 id 필드만을 검색한다.

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

검색된 결과는 rs에 들어 있다.

● 조건 검색

select 문
where 절

특정 조건에 부합하는 데이터를 검색하고 싶은 경우는 다음 코드와 같이 select 문의 where 절을 이용하여 조건에 맞는 데이터 검색한다. 다음은 student 테이블에서 id 필드의 값이 0494013인 레코드를 검색한다.

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

검색된 결과는 rs에 들어 있다.

● 검색된 데이터의 사용

검색의 결과는 ResultSet 객체에 저장된다. ResultSet 클래스의 메소드를 이용하여 커서가 가리키는 현재 행에 대해 열의 값을 읽어 온다. 예를 들어 ResultSet 클래스의 객체 rs에 저장된 name 필드의 값과 id 필드의 값을 얻어내기 위해서는 각각 다음과 같이 한다.

```
String sName = rs.getString("name"); // name 필드 값 읽기
String id  = rs.getString("id"); // id 필드 값 읽기
```

다음 코드는 ResultSet 객체에 저장된 각 행의 모든 열을 순차적으로 출력하는 코드이다.

```
while (rs.next()) { // rs에 저장된 다음 행으로 커서를 옮긴다.
    System.out.println(rs.getString("name"));
    System.out.println(rs.getString("id"));
    System.out.println(rs.getString("dept"));
}
```

열의 값을 읽기 위해서는 열의 데이터 타입을 알고 있어야 하며 그에 맞는 ResultSet 클래스의 메소드를 호출해야 한다. 또는 모든 데이터 타입에 대해 getString() 메소드로 읽을 수 있는데 이 경우는 모든 값이 문자열로 반환되므로 프로그램 내에서 적절한 데이터 타입으로 변환해서 사용해야 한다.

데이터 검색과 출력 **예제 16-2**

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

```
1  import java.io.*;
2  import java.sql.*;
3
4  public class JDBC_Ex2 {
5     public static void main (String[] args) {
6        Connection conn;
7        Statement stmt = null;
8        try {
9           Class.forName("com.mysql.jdbc.Driver"); // MySQL 드라이버 로드
10          conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/
                                                sampledb", "root","password");
                                    // JDBC 연결, password는 root 계정 패스워드 입력
11          System.out.println("DB 연결 완료");
12          stmt = conn.createStatement(); // SQL문 처리용 Statement 객체 생성
13          ResultSet srs = stmt.executeQuery("select * from student");
                                                        // 테이블의 모든 데이터 검색
14          printData(srs, "name", "id", "dept");
```

```
15          srs = stmt.executeQuery("select name, id, dept from student where
                                    name='이기자'"); // name이 "이기자"인 레코드만 검색
16          printData(srs, "name", "id", "dept");
17       } catch (ClassNotFoundException e) {
18          System.out.println("JDBC 드라이버 로드 에러");
19       } catch (SQLException e) {
20          System.out.println("SQL 실행 에러");
21       }
22    }
23    // 레코드의 각 열의 값 화면에 출력
24    private static void printData(ResultSet srs, String col1, String col2,
                                    String col3) throws SQLException {
25       while (srs.next()) {
26          if (col1 != "")
27             System.out.print(new String(srs.getString("name")));
28          if (col2 != "")
29             System.out.print("\t|\t" + srs.getString("id"));
30          if (col3 != "")
31             System.out.println("\t|\t" + new String(srs.getString("dept")));
32          else
33             System.out.println();
34       }
35    }
36 }
```

실행 결과

```
DB 연결 완료
이기자 | 0494013 | 컴퓨터공학
김철수 | 1091011 | 컴퓨터시스템
이기자 | 0494013 | 컴퓨터공학
```

데이터의 변경

추가, 수정, 삭제와 같이 데이터에 변경을 가하는 조작은 executeUpdate() 메소드를 이용한다.

● 레코드 추가

insert 문

새로운 레코드를 추가하기 위해서는 SQL의 insert 문을 사용한다. 다음은 insert 문을 이용하여 데이터를 추가하는 코드이다.

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

● 데이터 수정

update 문

기존 열의 값을 수정하기 위해서는 SQL의 update 문을 사용한다. 다음은 update 문을 이용하여 조건에 맞는 테이블의 열의 값을 수정하는 코드이다.

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

● 레코드 삭제

delete 문

레코드를 삭제하기 위해서는 SQL의 delete 문을 사용한다. 다음은 delete 문을 이용하여 조건에 맞는 테이블의 행을 삭제하는 코드이다.

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

데이터의 변경 예제 16-3

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.

```
1  import java.io.*;
2  import java.sql.*;
3
4  public class JDBC_Ex3 {
5     public static void main (String[] args) {
6        Connection conn;
7        Statement stmt = null;
8
9        try {
10          Class.forName("com.mysql.jdbc.Driver"); // MySQL 드라이버 로드
11          conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/
                                                sampledb", "root","password");
                                        // JDBC 연결, password는 root 계정 패스워드 입력
12          System.out.println("DB 연결 완료");
13          stmt = conn.createStatement(); // SQL문 처리용 Statement 객체 생성
14          stmt.executeUpdate("insert into student (name, id, dept) values('아
                                    무개', '0893012', '컴퓨터공학');"); // 레코드 추가
```

```
15          printTable(stmt);
16          stmt.executeUpdate("update student set id='0189011' where name='아무
                                개'"); // 데이터 수정
17          printTable(stmt);
18          stmt.executeUpdate("delete from student where name='아무개'");
                                                                  // 레코드 삭제
19          printTable(stmt);
20       } catch (ClassNotFoundException e) {
21          System.out.println("JDBC 드라이버 로드 에러");
22       } catch (SQLException e) {
23          System.out.println("SQL 실행 에러");
24       }
25    }
26    // 레코드의 각 열의 값 화면에 출력
27    private static void printTable(Statement stmt) throws SQLException {
28       ResultSet srs = stmt.executeQuery("select * from student");
29       while (srs.next()) {
30          System.out.print(new String(srs.getString("name")));
31          System.out.print("\t|\t" + srs.getString("id"));
32          System.out.println("\t|\t" + new String(srs.getString("dept")));
33       }
34    }
35 }
```

실행 결과

```
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
```

1 다음과 같은 구조를 갖는 cdinfo라는 테이블을 생성하는 SQL 문을 작성하라. 일차 키는 cd_id로 한다.

cd_id	title	publisher	artist	price
char(5)	varchar(50)	varchar(30)	varchar(20)	int

2 위에서 생성된 cdinfo 테이블에 다음과 같은 레코드를 추가하는 자바 코드를 작성하라.

cd_id	title	publisher	artist	price
a0001	홍길동 1집	오늘 레코드	홍길동	8000

3 cdinfo 테이블에서 price가 10000원인 CD의 타이틀을 모두 출력하는 자바 코드를 작성하라.

SUMMARY

요약

- 데이터베이스는 여러 응용 시스템들의 통합된 정보들을 저장하여 운영할 수 있는 공용 데이터들의 집합이다.
- 데이터베이스를 관리하는 시스템을 DBMS라고 한다.
- 현재 사용되는 대부분의 데이터베이스는 관계형 데이터베이스이며 JDBC API도 관계형 데이터베이스에 대한 API이다.
- SQL은 관계형 데이터베이스 관리 시스템에서 데이터베이스 스키마 생성, 자료의 검색 · 관리 · 수정, 데이터베이스 객체 접근 관리 등을 위해 고안된 언어이다.
- JDBC는 자바에서 관계형 데이터베이스에 저장된 데이터를 접근 및 조작할 수 있게 하는 API로서 다양한 DBMS에 대해 일관된 API로 데이터베이스 연결, 검색, 수정, 관리 등을 할 수 있게 한다.
- JDBC 드라이버는 DBMS 회사에서 제공한다.
- JDBC 프로그래밍 순서는 JDBC 드라이버 로드, 데이터베이스 연결, SQL 문장 실행 및 실행 결과 사용, 연결 해제의 순으로 구성된다.
- JDBC 프로그래밍에서 데이터베이스 연결, SQL 실행, 결과 사용을 위해 각각 Connection 클래스, Statement 클래스, ResultSet 클래스를 이용한다.

Open Challenge 데이터베이스로 사진 저장 및 추출하기

목적

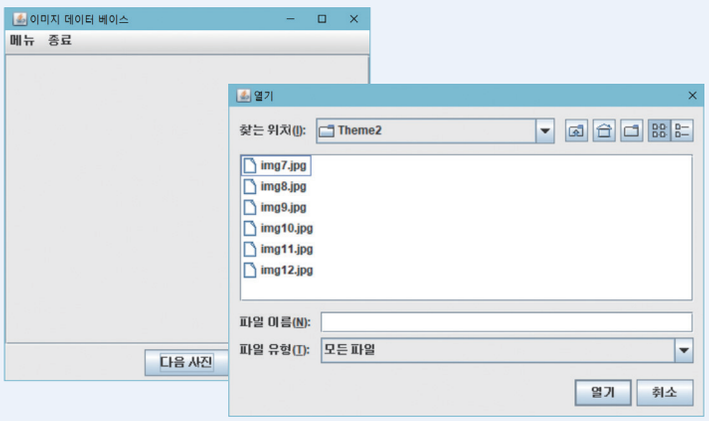
JDBC를 활용하여 데이터베이스 입출력 연습

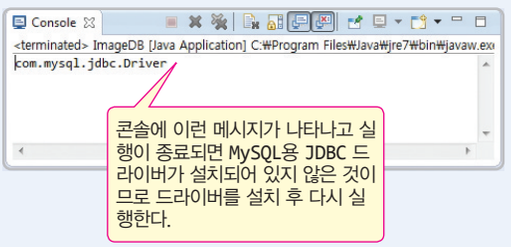
문자열이나 정수 등과 같은 단순한 데이터가 아닌 사진과 같은 바이너리 데이터를 저장해보자. 바이너리 데이터를 데이터베이스에 저장하기 위해서는 BLOB(Binary Large OBject)라는 데이터 타입을 사용한다. MySQL에서 16MB 이하의 데이터를 저장할 수 있는 데이터 타입은 mediumblob이다. 테이블 구조는 다음과 같다. 난이도 7

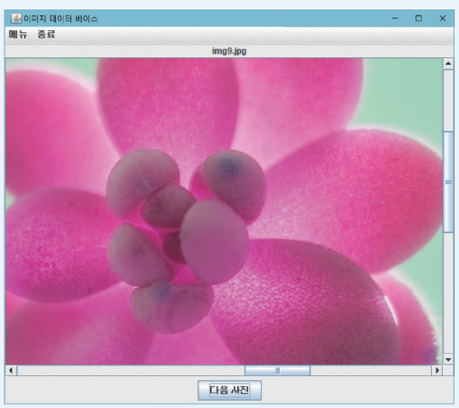
ID	FILENAME	FILE
int(11)	varchar(50)	mediumblob

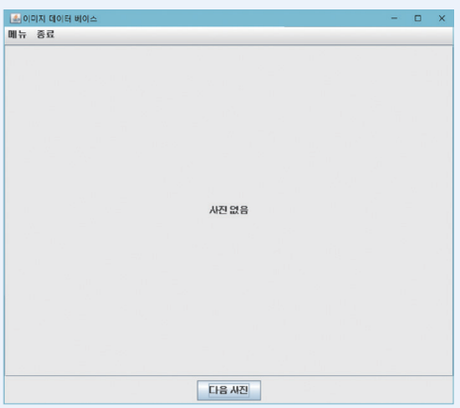
mediumblob 타입은 16MB까지 지원하나 큰 사진을 저장하려면 MySQL의 설정도 변경하여야 하므로 여기서는 편의상 1MB 이하의 사진만 저장하도록 하자.

메뉴는 사진 저장과 모든 사진 보기, 그리고 프로그램 종료가 있으며 버튼을 만들어 버튼을 누를 때마다 다음 사진이 보이도록 한다. 실행 예시 화면은 아래와 같다.









연습문제 EXERCISE

이론문제

• 홀수 문제는 정답이 공개됩니다.

1. JDBC에 대한 설명으로 잘못된 것은?
   ① 자바에서 데이터베이스를 사용할 수 있게 하는 API
   ② JDBC 드라이버는 JDBC 매니저를 통해 로드한다.
   ③ DBMS가 바뀌어도 이에 따른 JDBC 드라이버만 로드하면 되므로 프로그램 수정이 필요 없다.
   ④ 관계형 및 객체 지향 데이터베이스용 API를 각각 제공한다.

2. 데이터베이스의 테이블, 행, 열에 대해 설명하라.

3. 관계형 데이터베이스 관리 시스템에서 데이터베이스 스키마 생성, 자료의 검색 · 관리 · 수정, 데이터베이스 객체 접근 관리 등을 위해 고안된 언어를 무엇이라 하는가?

4. Statement 클래스와 ResultSet 클래스에 대해 설명하라.

5. 다음 표와 같은 구조의 테이블이 있다.

id	name
int	varchar(20)

데이터베이스에서 id 필드의 값을 읽으려고 한다. 빈칸을 채워라.

```
ResultSet rs;
....
int id  =rs. _________________;
```

실습문제

• 홀수 문제는 정답이 공개됩니다.

1. 윈도우 PC에서 MySQL Workbench를 이용하여 bookdb라는 데이터베이스를 생성하라. 난이도 2

목적 Workbench를 이용한 데이터베이스 생성

2. MySQL Workbench를 이용하여 bookdb 데이터베이스에 다음과 같은 테이블 구조를 갖는 book 테이블을 생성하라. Primay key는 id로 하라. 난이도 2

목적 Workbench를 이용한 테이블 생성

id	title	publisher	author
int	varchar(50)	varchar(30)	varchar(30)

3. 위에서 생성한 테이블에 다음과 같은 레코드를 추가하는 SQL 문을 작성하고 MySQL Workbench를 이용하여 레코드를 추가하라. 난이도 4

목적 SQL 쿼리문으로 레코드 추가

id	title	publisher	author
0	Harry Potter	Bloomsbury	J. K. Rowling
1	The Lord of the Rings	Allen & Unwin	J. R. R. Tolkein
2	Pride and Prejudice	T. Egerton Kingdom	Jane Austen

4. MySQL Workbench를 이용하여 3번에서 추가한 레코드를 모두 삭제하라. 난이도 4

목적 SQL 쿼리문으로 레코드 삭제

5. 3번의 레코드를 자바 프로그램에서 추가하도록 프로그램을 작성하라. 난이도 5

목적 JDBC 프로그램으로 레코드 추가

6. 책의 title이 "Pride and Prejudice"인 레코드의 레코드를 찾아 title을 "Pride & Prejudice"로 author를 "제인 오스틴"으로 수정하는 자바 프로그램을 작성하라. 난이도 5

목적 JDBC 프로그램으로 레코드 수정

7. 6번에서 수정한 레코드를 찾아서 삭제하는 자바 프로그램을 작성하라. 난이도 5

목적 JDBC 프로그램으로 레코드 삭제

목적 종합적인 JDBC 프로그램 연습

8. 다음과 같은 기능을 수행하는 프로그램을 작성하라. 난이도 8

- 프로그램이 시작되면 데이터베이스의 모든 데이터를 출력하고 데이터베이스 관리 메뉴를 출력하라. 관리 메뉴는 추가, 삭제, 수정, 끝내기가 있다.
- 추가 메뉴에서는 사용자에게 id, 책 제목, 출판사, 저자 정보를 입력받아 데이터베이스에 레코드를 추가하라.
- 삭제 메뉴에서는 사용자에게 id 값을 입력받아 레코드를 삭제하라.
- 수정 메뉴에서는 사용자에게 수정할 속성의 이름을 입력받고, 속성의 현재 값과 새로운 값을 입력받아 레코드를 수정하라.

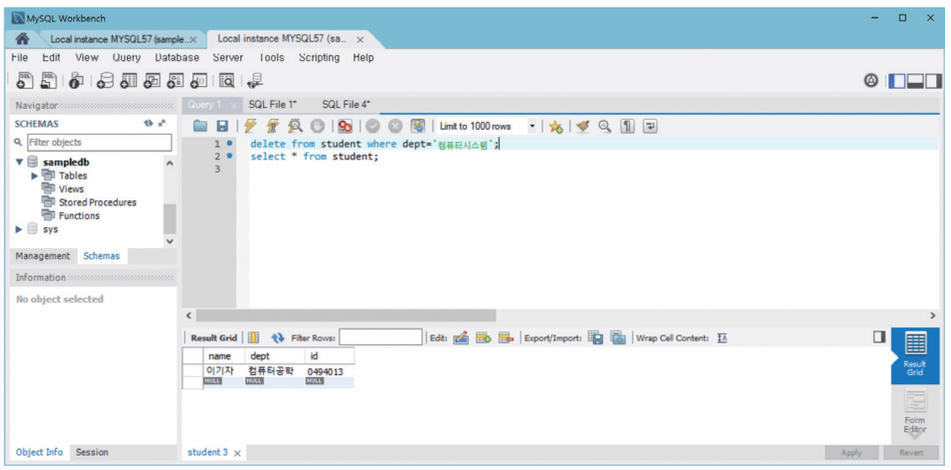
16장 Check Time 정답

Check Time(p.839)

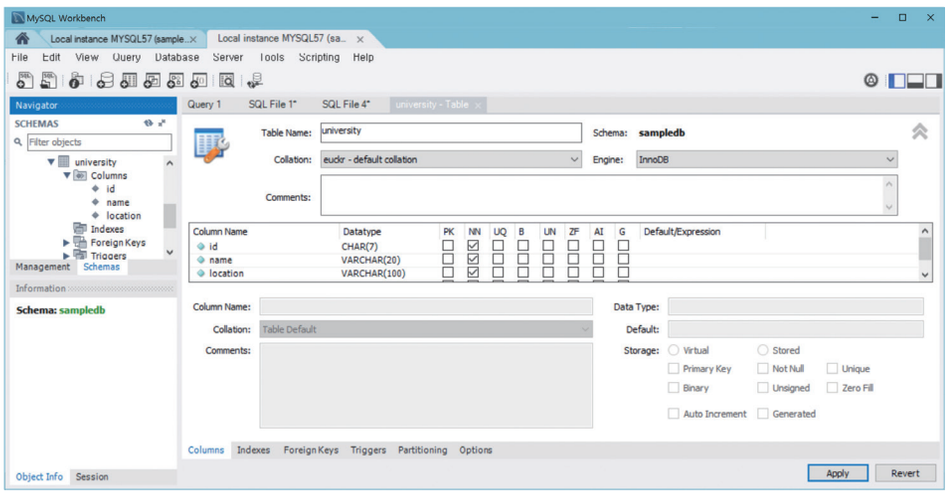
1. DBMS
2. 관계형 데이터베이스
3. DBMS가 바뀌어도 이에 따른 JDBC 드라이버만 로드하면 자바 응용프로그램은 일관된 API로 데이터베이스 연결, 검색, 수정, 관리 등을 할 수 있다.

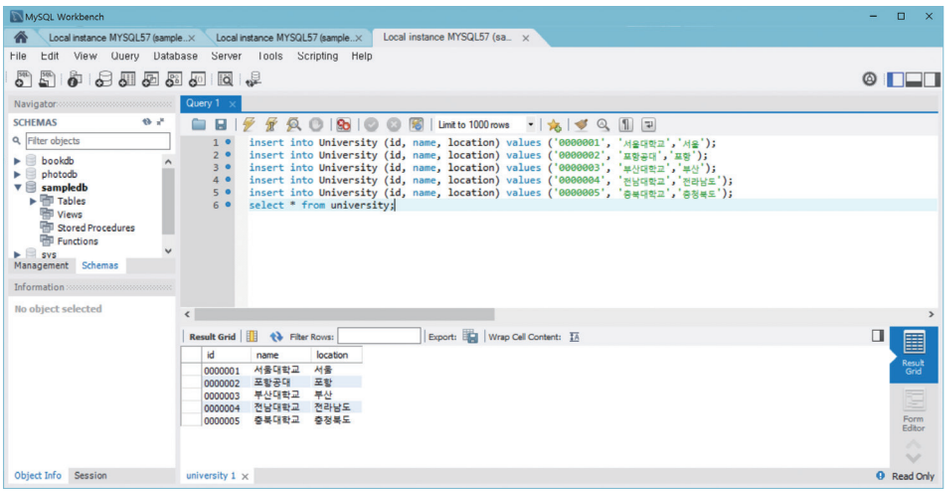
Check Time(p.855)

1.



2.





Check Time(p.865)

1.

```
Statement stmt = conn.createStatement();
stmt.execute("create table cdinfo (cd_id char(5) not null, title varchar(50) not null,
publisher varchar(30) not null, artist varchar(20), price int, primary key(cd_id));");
```

2.

```
Statement stmt = conn.createStatement();
stmt.executeUpdate("insert into cdinfo (cd_id, title, publisher, artist, price)
values('a0001','홍길동 1집','오늘 레코드','홍길동',8000);");
```

3.

```
Statement stmt = conn.createStatement();
ResultSet rs = stmt.executeQuery("select * from cdinfo where price=10000");
while (rs.next()) {
   System.out.print(rs.getString("cd_id"));
   System.out.print("\t|\t" + rs.getString("title");
   System.out.print("\t|\t" + rs.getString("publisher");
   System.out.print("\t|\t" + rs.getString("artist");
   System.out.println("\t|\t" + rs.getString("price"));
}
```