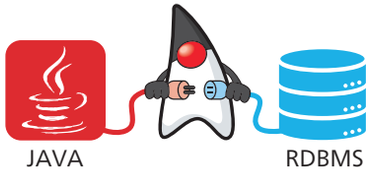
CHAPTER 18

데이터베이스 프로그래밍

▶ 다음과 같은 작업들을 수행하는 방법을 알고 있나요? 이번 장에서 함께 알아봐요.

1. JDBC를 사용하여 자바 애플리케이션을 MySQL과 같은 데이터베이스와 연결하는 방법을 알고 있나요?
2. JDBC를 사용하여 데이터베이스의 테이블을 출력하는 방법을 알고 있나요?
3. JDBC를 사용하여 데이터베이스에서 이미지를 저장하고 검색하는 방법을 알고 있나요?
4. JDBC를 사용하여 데이터베이스에서 파일을 저장하고 검색하는 방법을 알고 있나요?



학습목차

Power JAVA 3e

18.1 자바와 데이터베이스

JDBC(Java Database Connectivity)는 자바 애플리케이션과 데이터베이스를 연결하는 라이브러리이다. JDBC를 사용하면 자바 프로그램에서 데이터베이스에 접근하여서 검색이나 저장 등 여러 가지 관련된 작업을 할 수 있게 된다.

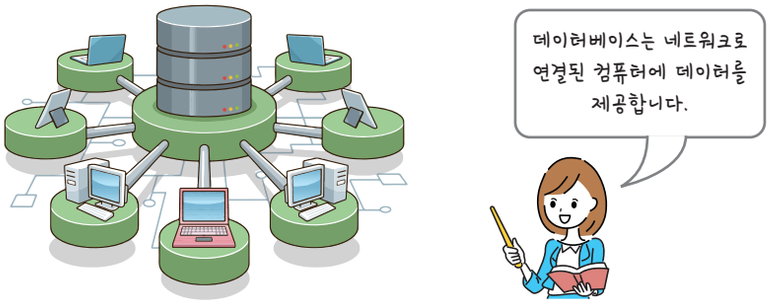


그림 18.1 데이터베이스

자바의 개발진들은 초창기부터 자바로 데이터베이스를 접근할 수 있기를 원했다. 1995년에 표준 자바 라이브러리를 확장하여 데이터베이스를 접근하려 하였다. 처음 목표는 순수하게 자바만을 사용하여서 어떤 데이터베이스라도 접근하게 하는 것이었으나, 시장에는 많은 종류의 데이터베이스가 있어서 이것은 처음부터 불가능한 일이었다. 따라서 자바는 애플리케이션 프로그래머를 위해서는 JDBC API를 제공하고 데이터베이스 업체들을 위해서는 JDBC 드라이버 API를 제공하여서, 많은 데이터베이스 업체들이 그들의 데이터베이스를 위한 드라이버를 개발할 수 있도록 하였다. 업체들은 자신들의 드라이버를 쉽게 드라이버 관리자에 등록할 수 있었다. 애플리케이션은 드라이버 관리자에게 요청을 하고 드라이버 관리자는 드라이버를 통하여 데이터베이스에게 요청을 하게 된다.

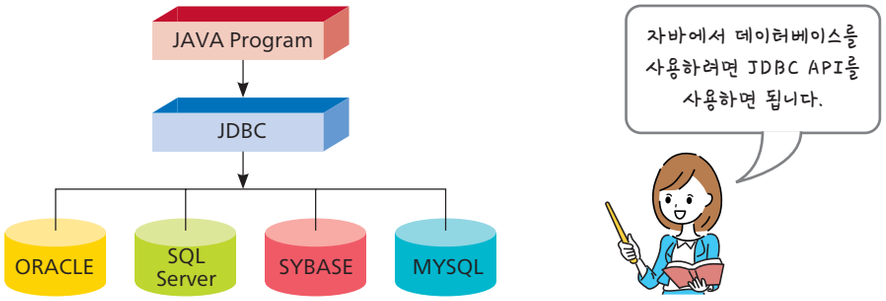


그림 18.2 자바와 데이터베이스

2계층 처리 모델과 3계층 처리 모델

JDBC API는 2계층(Two-tier)와 3계층(Three-tier) 처리 모델을 모두 지원한다.

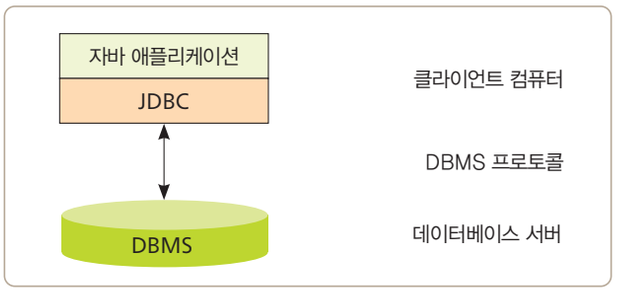


그림 18.3 2계층 모델

2계층(Two-tier) **처리 모델**에서는 자바 애플리케이션이 직접 데이터베이스 서버와 연결된다. 이 방법에서는 JDBC 드라이버가 데이터 소스와 직접 통신할 수 있어야 한다. 사용자의 SQL 명령어는 데이터베이스 서버로 전달되며, SQL 명령어의 결과가 사용자한테 보내진다. 이러한 모델은 클라이언트/서버 구성으로 언급되는데, 사용자의 컴퓨터가 클라이언트이고 데이터베이스가 있는 컴퓨터가 서버가 된다. 사용자 시스템 인터페이스는 일반적으로 사용자의 데스크탑 환경에 위치하며, 데이터베이스는 일반적으로 많은 클라이언트에 서비스를 제공하는 보다 강력한 시스템인 서버에 있다. 네트워크는 인트라넷이나 아니면 인터넷이 될 수 있다. 데이터베이스 서버는 네트워크를 통하여 연결되는 다른 컴퓨터에 있을 수 있다. **2계층 처리 모델**에서 애플리케이션 로직(코드)은 클라이언트의 사용자 인터페이스 내부 또는 서버의 데이터베이스에 묻혀 있다.

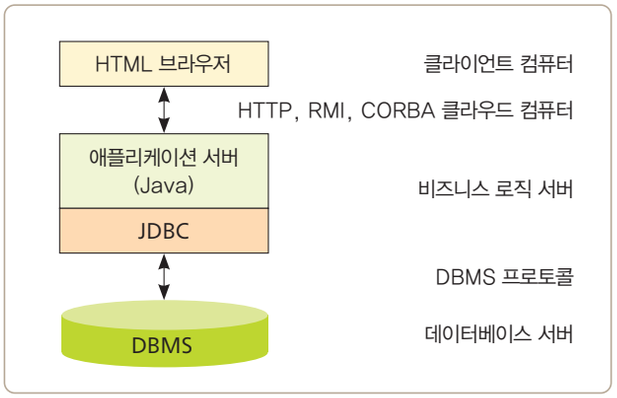


그림 18.4 3계층 모델

3계층(Three-tier) 처리 모델에서 애플리케이션 로직은 중간 계층에 있으며 데이터 및 사용자 인터페이스와 분리된다. **3계층 처리 모델**에서는 사용자 인터페이스와 데이터베이스 서버 사이에 중간 계층이 추가되었다. 트랜잭션 처리 모니터, 메시지 서버, 응용 프로그램 서버와 같이 이 중간

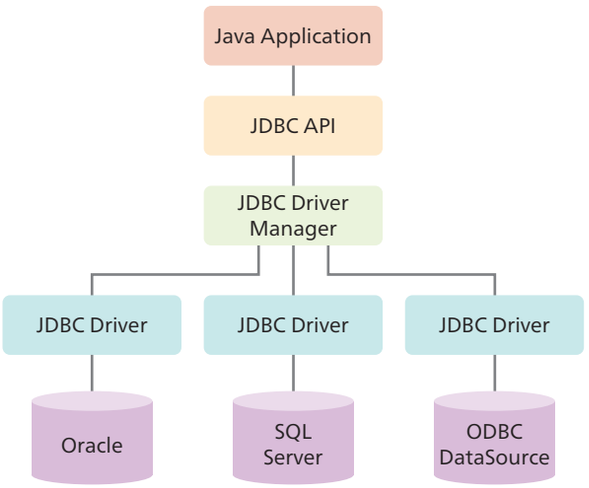
계층을 구현하는 다양한 방법이 있다. **3계층 처리 모델**은 확장 가능하고 강력하며 유연하다. 또한 여러 데이터 소스를 통합할 수 있다. 회사에서는 이러한 **3계층** 처리 모델을 좋아하는데, 주된 이유는 중간 계층을 이용하여 사용자의 접근과 비즈니스 데이터 업데이트를 제어할 수 있기 때문이다. **3계층 처리 모델은 주로** 웹 기반 시스템이 된다.

최근까지 중간 계층이 속도를 위하여 주로 C나 C++ 언어를 사용하여서 작성되었다. 하지만 자바 바이트 코드를 최적화하는 기술이 발전하였고 엔터프라이즈 자바 빈즈와 같은 기술이 개발되면서 자바 플랫폼은 중간 계층을 위한 표준 플랫폼이 되고 있다. 자바의 견고성, 멀티스레딩, 보안 기능 등이 인정받고 있는 것이다. 회사들이 애플리케이션 로직을 작성하는데 자바 언어를 점점 많이 사용함에 따라 JDBC API도 중간 계층에서 많이 사용되고 있다. JDBC의 장점이라면 연결 풀 기능(connection pooling), 분산 트랜잭션 기능(distributed transactions)을 들 수 있다.

데이터베이스 프로그램 개발 절차

자바를 이용하여서 데이터베이스 응용 프로그램을 개발하려면 몇 가지의 절차를 거쳐야 한다.

1. DBMS(DataBase Management System)를 설치하여야 한다. 데이터베이스는 간단하게는 마이크로소프트 액세스를 통하여 작성할 수 있고, MySQL이나 오라클과 같은 전문적인 DBMS를 설치할 수도 있다. 우리는 MySQL을 설치하여 사용하도록 하자.



2. 자신이 설치한 DBMS에 필요한 JDBC 드라이버를 설치한다. 이 드라이버는 DBMS 회사에서 제공한다. MySQL을 사용한다면, "Connector/J"가 바로 우리에게 필요한 JDBC 드라이버이다. 최근에는 MySQL을 설치하면 "Connector/J"도 함께 설치된다.
3. JDBC가 제공하는 기능을 이용하여 데이터베이스 응용 프로그램을 개발한다. 자바의 JDBC API가 하는 일은 자바 프로그램에서 SQL 명령어들을 데이터베이스 관리 시스템으로 보낼 수 있도록 자바와 데이터베이스를 연결하는 것이다.

중간점검

1. 자바 프로그램에서 데이터베이스를 사용하려면 어떤 API가 필요한가?
2. 2계층(Two-tier)와 3계층(Three-tier) 처리 모델을 설명해보자.

데이터베이스의 기초 18.2

이번 절에서는 데이터베이스의 기초 지식에 대하여 살펴본다. 만약 데이터베이스를 잘 알고 있는 독자라면 이 부분의 학습은 생략할 수 있다.

데이터베이스란?

데이터베이스는 데이터가 빠르게 추출될 수 있도록 데이터를 조직화하여서 저장하는 방법이다. 일반적으로 열과 행으로 이루어진 테이블에 데이터를 저장한다. **데이터베이스 관리 시스템(DBMS)**은 다수의 사용자를 위하여 데이터가 저장, 접근, 변경되는 기능을 제공한다.

현재 가장 인기 있는 데이터베이스 시스템은 **관계형 데이터베이스(relational database)**이다. 관계형 데이터베이스에서는 여러 개의 테이블이 존재하고 테이블과 테이블 간에는 공통적인 데이터로 인하여 어떤 관계가 성립될 수 있다. 예를 들어서 고객들의 테이블과 주문서 테이블은 분명히 공통적으로 고객들의 정보를 포함하고 있을 것이다.

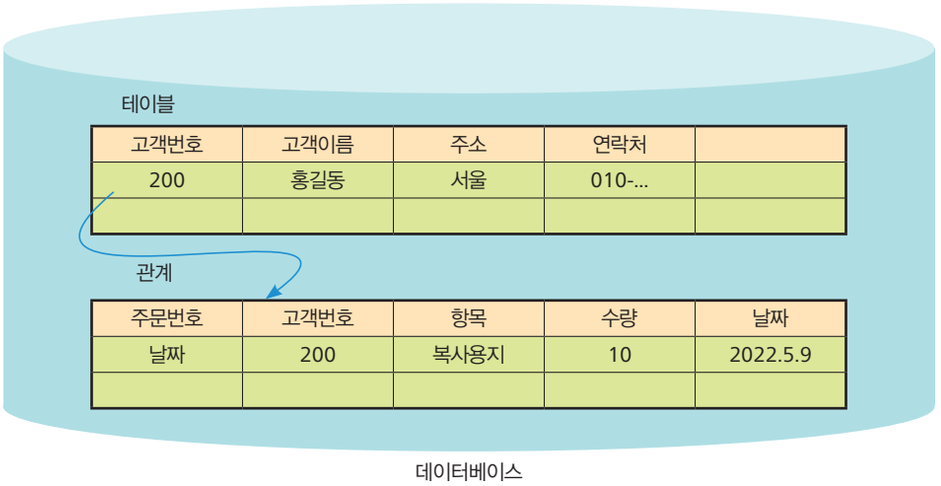
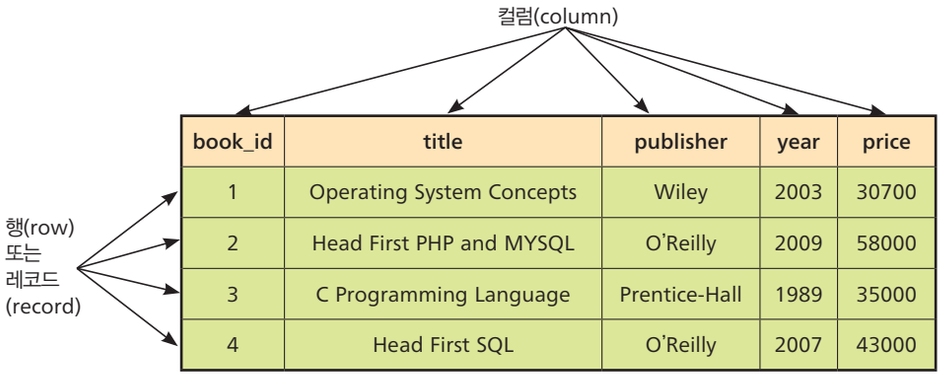


그림 18.5 데이터베이스의 개념

가장 많이 사용되는 DBMS로는 오라클, 마이크로소프트사의 SQL Server, MySQL, MongoDB 등을 들 수 있다. 이 중에서 특히 MySQL은 오픈 소스의 일종이라서 누구든지 무료로 사용할 수 있다. 관계형 데이터베이스는 SQL이라고 불리는 언어로 사용할 수 있다. SQL은 데이터베이스를 조작하는 국제적인 표준 언어이다.

테이블

테이블의 하나의 행(row)은 레코드(record)라고 불린다. 레코드는 여러 개의 컬럼(column)으로 이루어져 있다. 테이블은 무결성 법칙을 따라서 작성되어야 한다. 무결성 법칙 중의 하나가 테이블에서 각 레코드는 중복되지 않아야 한다는 것이다. 중복이 있으면 어떤 레코드가 올바른 것인지를 알기 어렵다. 대부분의 DBMS에서 사용자는 중복된 레코드가 허용되지 않도록 하나의 컬럼을 이용한다. 이러한 특정한 컬럼을 주요키(primary key)라고 한다. 예를 들어 학생 데이터에서는 학번이 주요키가 될 것이다. 주요키는 null이 되면 안 된다. 다음은 책에 대한 데이터를 테이블로 정리한 것이다.



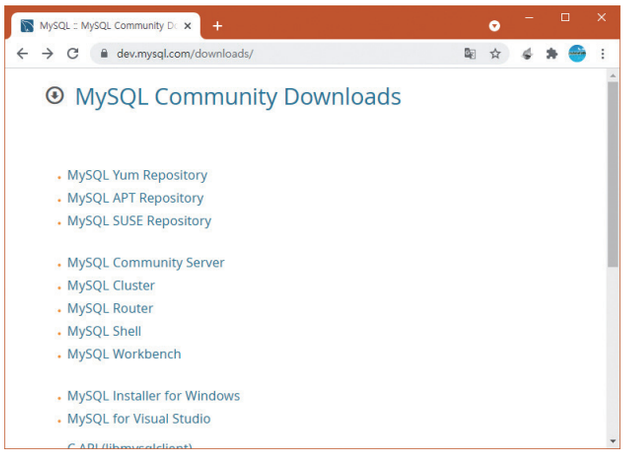
book_id	title	publisher	year	price
1	Operating System Concepts	Wiley	2003	30700
2	Head First PHP and MYSQL	O'Reilly	2009	58000
3	C Programming Language	Prentice-Hall	1989	35000
4	Head First SQL	O'Reilly	2007	43000

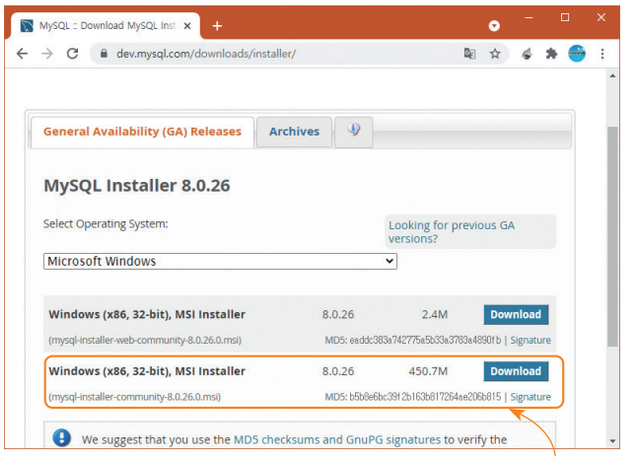
그림 18.6 데이터베이스 테이블

위의 테이블에서 주요키는 book_id이다. book_id는 레코드가 추가되면 1씩 증가되는 값이다. 따라서 동일할 수가 없다. 책 이름은 주요키로 사용할 수가 없다. 왜냐하면 책 이름은 동일할 수가 있기 때문이다. 그리고 레코드와 레코드를 비교할 때는 숫자가 문자보다 더 효율적이다.

MySQL

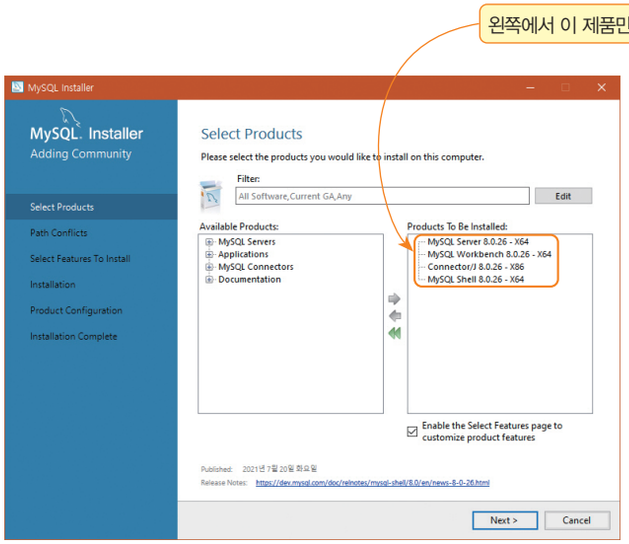
데이터베이스에 대하여 여러 가지 프로그래밍을 하려면 데이터베이스 시스템이 설치되어 있어야 한다. 무료로 설치할 수 있는 데이터베이스는 MySQL이다. MySQL은 **www.mysql.com**에서 다운로드 받을 수 있다. MySQL 웹 사이트에서 [Downloads] 탭을 선택하여 무료 버전인 MySQL Community Edition (GPL)을 다운로드한다. 상용 버전은 아주 큰 글씨로 잘 보이게 적혀있지만, 무료 버전인 Community Edition은 화면 맨 아래쪽에 아주 작은 글씨로 쓰여 있으니 잘 찾아야 한다. MySQL Community Server를 선택하고 다음 화면에서 "MySQL Installation 8.0.26 for Windows"를 선택한다. 다음 화면에서 "Windows (x86, 32-bit), MSI Installer"를 선택한다.

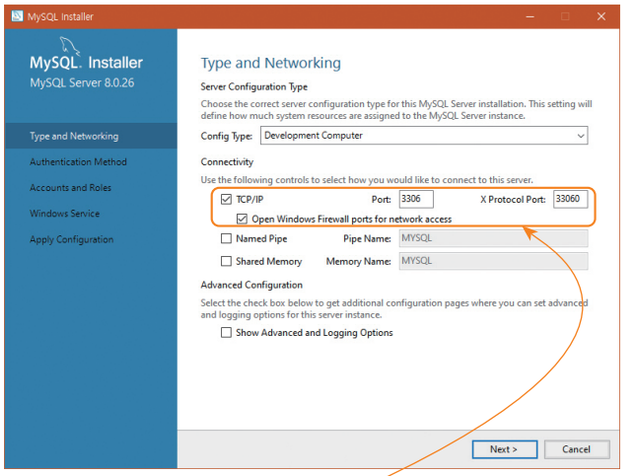




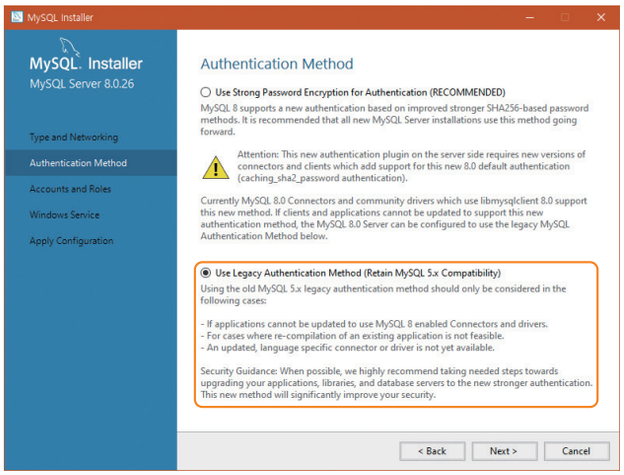
다음 화면에서 오라클 아이디로 로그인하라고 나오는데 화면의 맨 아래쪽에 보면 “No thanks, just start my download.”가 있다. 이것을 누르면 로그인을 피해갈 수 있다.

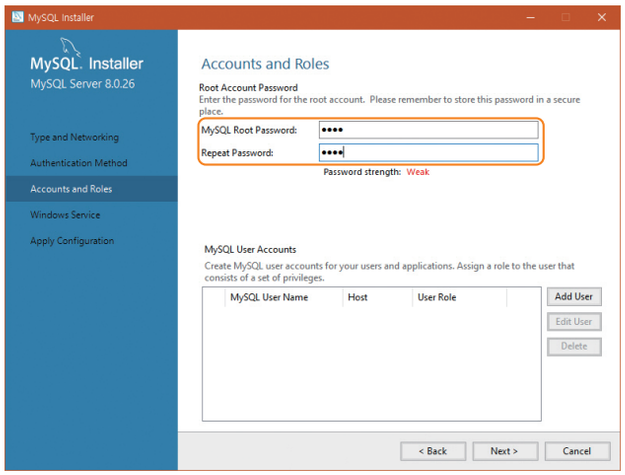
다운로드 받은 파일을 실행하면 설치가 시작된다. 설치 시에 물어보는 것은 대부분 기본 옵션을 선택하면 된다. 설치되면서 다음 화면이 등장하는데 이 중 하나가 실패로 나오면 다시 설치하여야 한다. 만약 개인 방화벽을 사용하고 있는 경우에 다음과 같이 TCP 포트 3306에 대한 접근을 허용하여야 한다.



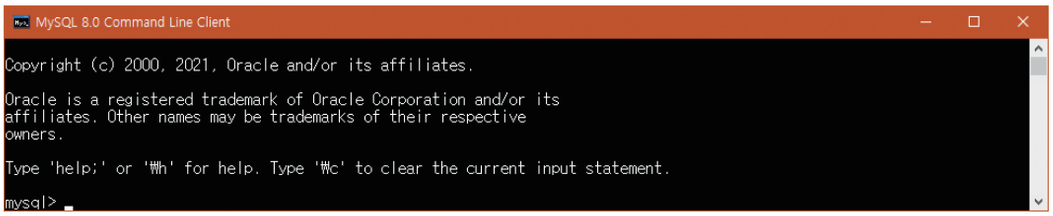


다음 화면에서 MySQL 관리자 패스워드를 설정한다. 패스워드는 반드시 기억하고 있어야 한다.





마지막 대화상자에서 [Finish]를 누르면 MySQL 서버가 시작된다. 이제부터는 MySQL을 사용할 수 있다. MySQL은 다음과 같은 명령어 행 클라이언트를 가지고 있다. 앞으로 우리는 이것을 이용하여서 여러 가지 작업을 할 것이다. 윈도우의 [시작] 버튼을 누르고 [MySQL 8.0 Command Line Client]를 찾아서 실행하여 보자.



이 상태에서 여러 가지 SQL 명령어를 넣어서 실습할 수 있다.

중간점검

1. 데이터베이스 테이블의 하나의 행은 무엇이라고 불리는가?
2. 무결성이란 무엇인가?
3. 주요키란 왜 필요한가?

SQL 18.3

데이터베이스 프로그래밍을 작성하기 전에 먼저 데이터베이스를 생성해야 한다. SQL을 사용하여 "book_db"라고 불리는 데이터베이스를 생성하여 보자.

```
+--------------------------+---------------+-------+
| title                    | publisher     | price |
+--------------------------+---------------+-------+
| Operating System Concepts | Wiley         | 30700 |
| Head First PHP and MYSQL  | OReilly       | 58000 |
| C Programming Language    | Prentice-Hall | 35000 |
| Head First SQL            | OReilly       | 43000 |
+--------------------------+---------------+-------+
```

SQL이란?

SQL 관계형 데이터베이스에서 사용하기 위하여 설계된 언어이다. 표준적인 SQL 명령어들이 있으며 이것은 모든 관계형 데이터베이스에 의하여 지원된다. SQL 명령어들은 두 가지의 카테고리로 나누어진다. **데이터 정의 명령어(Data Definition Language)**들은 테이블을 생성하거나 변경한다. **데이터 조작 명령어(Data Manipulation Language)**는 데이터를 추출, 추가, 삭제, 수정한다. 많이 사용되는 SQL 명령어를 요약하면 다음과 같다.

구분	명령어	설명
데이터 정의 명령어 (Data Definition Language)	CREATE	사용자가 제공하는 컬럼 이름을 가지고 테이블을 생성한다. 사용자는 컬럼의 데이터 타입도 지정하여야 한다. 데이터 타입은 데이터베이스에 따라 달라진다. CREATE TABLE은 보통 DML보다 적게 사용된다. 왜냐하면 이미 테이블이 만들어져 있는 경우가 많기 때문이다.
	ALTER	테이블에서 컬럼을 추가하거나 삭제한다.
	DROP	테이블의 모든 레코드를 제거하고 테이블의 정의 자체를 데이터베이스로부터 삭제하는 명령어이다.
	USE	어떤 데이터베이스를 사용하는지를 지정한다.
데이터 조작 명령어 (Data Manipulation Language)	SELECT	데이터베이스로부터 데이터를 쿼리하고 출력한다. SELECT 명령어들은 결과 집합에 포함시킬 컬럼을 지정한다. SQL 명령어 중에서 가장 자주 사용된다.
	INSERT	새로운 레코드를 테이블에 추가한다. INSERT는 새롭게 생성된 테이블을 채우거나 새로운 레코들를 이미 존재하는 테이블에 추가할 때 사용된다.
	DELETE	지정된 레코드를 테이블로부터 삭제한다.
	UPDATE	테이블에서 레코드에 존재하는 값을 변경한다.

데이터베이스 생성하기

데이터베이스에 데이터를 저장하기 전에 당연히 데이터베이스부터 생성하여야 한다. 여기서는 MySQL의 명령어인 CREATE를 사용하여서 데이터베이스를 생성해보기로 하자. CRAETE는 다음과 같은 구문을 가진다.

```
CREATE TABLE 테이블이름  (컬럼이름1  자료형1,  컬럼이름2  자료형2, ...);
```

책에 대한 데이터베이스와 테이블을 생성하기 위하여 [MySQL 8.0 Command Line Client]에서 다음과 같은 명령어를 입력하여 실행해보자.

```
DROP DATABASE book_db;
```

먼저 같은 이름의 데이터베이스가 이미 있을 수도 있으므로 book_db 데이터베이스를 삭제한다.

```
CREATE DATABASE book_db;
```

book_db라고 하는 데이터베이스를 생성한다.

```
USE book_db;
```

지금부터 book_db 데이터베이스를 사용한다는 의미이다.

```
CREATE TABLE books (
   book_id INT NOT NULL auto_increment,
   title VARCHAR(50),
   publisher VARCHAR(30),
   year VARCHAR(10),
   price INT,
   PRIMARY KEY(book_id)
);
```

book_db 안에 books라고 하는 테이블을 생성한다. book_id와 title, publisher, year, price 등이 모두 컬럼이 된다. 컬럼 이름 다음에는 컬럼의 타입을 적어준다. 많이 사용되는 컬럼의 타입은 INT, DECIMAL(n, d), CHAR(n), VARCHAR(n), DATE 등이 있다. CHAR(n)은 글자의 개수가 일정한 필드를 나타내고 VARCHAR(n)은 글자의 개수가 가변적일 때 사용한다.

book_id 컬럼의 데이터 타입은 INT이다. 이 컬럼은 NULL이 되면 안 된다. 또한 이 컬럼 값은 레코드가 추가될 때마다 자동으로 증가한다. book_id 컬럼이 주요키(primary key)가 된다. 주요키는 테이블에서 각각의 레코드를 구별할 수 있는 유일한 값이다. 모든 테이블은 주요키를 가져야 한다.

레코드 추가하기

데이터베이스를 생성하였으면 다음 작업은 레코드를 추가하는 것이다. 레코드 추가는 INSERT 명령어를 사용한다. INSERT 문장의 형식은 먼저 데이터를 삽입하기를 원하는 컬럼들을 나열하고 실제의 데이터를 뒤이어 나열하면 된다.

```
INSERT INTO 테이블이름  [(컬럼이름1, 컬럼이름2, ...)] VALUES (값1, 값2, ...);
```

다음과 같은 문장들을 입력하여 테이블에 몇 개의 레코드를 추가해본다.

```
mysql> USE book_db;

INSERT INTO books (title, publisher, year, price)
   VALUES('Operating System Concepts', 'Wiley', '2003', 30700);

INSERT INTO books (title, publisher, year, price)
   VALUES('Head First PHP and MYSQL', 'OReilly', '2009', 58000);

INSERT INTO books (title, publisher, year, price)
   VALUES('C Programming Language ', 'Prentice-Hall', '1989', 35000);

INSERT INTO books (title, publisher, year, price)
   VALUES('Head First SQL', 'OReilly', '2007', 43000);
```

레코드 검색하기

SELECT 문장은 **쿼리(query)**라고도 불리는데 테이블에서 정보를 얻어내기 위하여 사용된다. 이 명령어는 출력하고 싶은 컬럼과 테이블을 지정한다.

```
SELECT 컬럼이름 FROM 테이블이름 [ WHERE 조건 ] [ORDER BY 정렬방식]
```

다음 SELECT 명령어는 books라는 테이블에서 title, publisher, price 컬럼을 출력한다. FROM 절은 테이블을 지정한다.

```
mysql> USE book_db;
mysql> SELECT title, publisher, price FROM books;
+--------------------------+---------------+-------+
| title                    | publisher     | price |
+--------------------------+---------------+-------+
| Operating System Concepts | Wiley         | 30700 |
| Head First PHP and MYSQL  | OReilly       | 58000 |
| C Programming Language    | Prentice-Hall | 35000 |
| Head First SQL            | OReilly       | 43000 |
+--------------------------+---------------+-------+
```

SELECT 명령어의 결과로 나오는 레코드들의 집합을 결과 집합(result set)이라고 한다. 다음의 코드는 테이블 안의 모든 레코드를 포함하는 결과 집합을 생성한다. 왜냐하면 모든 컬럼을 요청하고 있고 조건이 없기 때문이다. 여기서 SELECT * 는 모든 컬럼을 선택함을 의미한다.

```
mysql> SELECT * from books;
+---------+--------------------------+---------------+------+-------+
| book_id | title                    | publisher     | year | price |
+---------+--------------------------+---------------+------+-------+
|       1 | Operating System Concepts | Wiley         | 2003 | 30700 |
|       2 | Head First PHP and MYSQL  | OReilly       | 2009 | 58000 |
|       3 | C Programming Language    | Prentice-Hall | 1989 | 35000 |
|       4 | Head First SQL            | OReilly       | 2007 | 43000 |
+---------+--------------------------+---------------+------+-------+
```

SELECT를 사용할 때 선택을 위한 조건을 명시할 수 있다. 이런 경우에는 결과 집합이 제시된 조건을 만족하는 레코드들이 된다. WHERE 절은 레코드를 선택하는 기준을 제공한다. 예를 들어서 title이 'Head First'로 시작되는 레코드만 선택하려면 다음과 같이 입력하면 된다.

```
mysql> SELECT * FROM books WHERE title LIKE 'Head First%';
+---------+--------------------------+-----------+------+-------+
| book_id | title                    | publisher | year | price |
+---------+--------------------------+-----------+------+-------+
|       2 | Head First PHP and MYSQL | OReilly   | 2009 | 58000 |
|       4 | Head First SQL           | OReilly   | 2007 | 43000 |
+---------+--------------------------+-----------+------+-------+
```

키워드 LIKE는 문자열을 비교할 때 사용된다. 이것은 와일드 카드(wildcard)를 가지고 있는 패턴을 사용할 수 있다. 'Head First%'에서 퍼센트(%) 기호는 문자열 'Head First'에 0개 이상의 문자를 더하는 것을 의미한다. 따라서 'Head First PHP' 또는 'Head First SQL'이 매칭이 된다. 그러나 'Head'는 안 된다. 다른 와일드 카드로는 언더라인 (_)이 있다. 이것은 하나의 글자대신 사용할 수 있다.

WHERE 절에서 등호와 부등호를 사용하면 숫자를 비교할 수 있다. 아래의 문장은 가격이 30000원보다는 높고 50000원보다는 낮은 책을 선택한다.

```
mysql> SELECT * FROM books WHERE price > 30000 and price < 50000;
+---------+--------------------------+---------------+------+-------+
| book_id | title                    | publisher     | year | price |
+---------+--------------------------+---------------+------+-------+
|       1 | Operating System Concepts | Wiley         | 2003 | 30700 |
|       3 | C Programming Language    | Prentice-Hall | 1989 | 35000 |
|       4 | Head First SQL            | OReilly       | 2007 | 43000 |
+---------+--------------------------+---------------+------+-------+
```

레코드들을 정렬하여 출력하려면 ORDER BY를 사용한다. 다음의 문장은 모든 책을 발행연도로 정렬하여 출력한다.

```
mysql> SELECT * FROM books ORDER BY year;
+---------+--------------------------+---------------+------+-------+
| book_id | title                    | publisher     | year | price |
+---------+--------------------------+---------------+------+-------+
|       3 | C Programming Language   | Prentice-Hall | 1989 | 35000 |
|       1 | Operating System Concepts | Wiley         | 2003 | 30700 |
|       4 | Head First SQL           | OReilly       | 2007 | 43000 |
|       2 | Head First PHP and MYSQL | OReilly       | 2009 | 58000 |
+---------+--------------------------+---------------+------+-------+
4 rows in set (0.03 sec)
```

레코드 수정하기

SQL의 UPDATE 명령어를 사용하면 레코드를 변경할 수 있다. UPDATE 명령어는 다음과 같은 구문을 가진다. 만약 조건을 주지 않으면 모든 레코드가 변경된다.

```
UPDATE 테이블이름 SET 컬럼명 = 새로운값, ... [WHERE 조건];
```

아래의 SQL 문장은 만약 발행연도가 19XX이면 가격을 30000원으로 변경한다.

```
mysql> USE book_db;

mysql> UPDATE books SET price = 30000 WHERE year LIKE '19%';
Query OK, 1 row affected (0.02 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> SELECT * FROM books;
+---------+--------------------------+---------------+------+-------+
| book_id | title                    | publisher     | year | price |
+---------+--------------------------+---------------+------+-------+
|       1 | Operating System Concepts | Wiley         | 2003 | 30700 |
|       2 | Head First PHP and MYSQL | OReilly       | 2009 | 58000 |
|       3 | C Programming Language   | Prentice-Hall | 1989 | 30000 |
|       4 | Head First SQL           | OReilly       | 2007 | 43000 |
+---------+--------------------------+---------------+------+-------+
```

레코드 삭제하기

SQL의 DELETE 명령어를 사용하면 현재의 레코드를 삭제할 수 있다. DELETE 명령어는 다음과 같은 구문을 가진다.

```
DELETE FROM 테이블이름 [WHERE 조건];
```

아래의 SQL 문장은 만약 발행연도가 19XX인 레코드가 있다면 삭제한다.

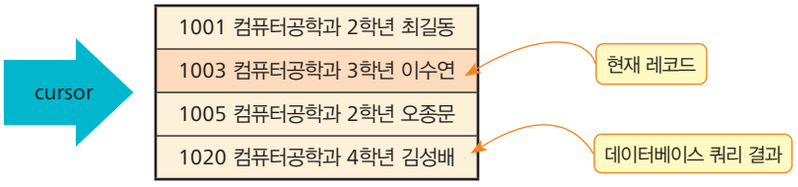
```
mysql> USE book_db;

mysql> DELETE FROM books WHERE year LIKE '19%';
Query OK, 1 row affected (0.03 sec)

mysql> select * from books;
+---------+--------------------------+----------------+------+-------+
| book_id | title                    | publisher      | year | price |
+---------+--------------------------+----------------+------+-------+
|       1 | Operating System Concepts | Wiley          | 2003 | 30700 |
|       2 | Head First PHP and MYSQL  | OReilly        | 2009 | 58000 |
|       4 | Head First SQL            | OReilly        | 2007 | 43000 |
+---------+--------------------------+----------------+------+-------+
```

결과 집합(Result Sets)과 커서(Cursors)

쿼리의 조건을 만족하는 레코드들의 집합이 **결과 집합(result set)**이다. 결과 집합에서 사용자는 커서를 사용하여 한 번에 한 레코드씩 데이터에 접근할 수 있다. **커서(cursor)**는 결과 집합의 레코드들을 포함하고 있는 파일에 대한 포인터라고 간주할 수 있다. 이 포인터는 현재 접근되고 있는 레코드들을 가리킨다. 커서는 사용자로 하여금 결과 집합에서 각각의 레코드들을 처리할 수 있도록 도와준다. 커서는 레코드들에 대하여 반복 처리를 할 때에 이용된다. 대부분의 DBMS은 결과 집합이 생성될 때 커서가 자동적으로 만들어 진다. 커서는 정방향이나 역방향으로 움직일 수 있다. 따라서 특정한 레코드로 이동할 수 있다.

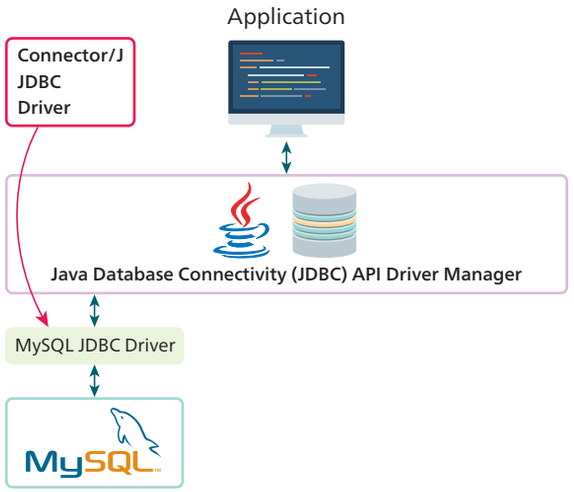


중간점검

1. SQL 명령어들은 어떻게 분류할 수 있는가?
2. SQL 명령어 중에서 테이블을 생성하는 명령어는 무엇인가?
3. SQL 명령어 중에서 특정 조건을 주어서 쿼리하는 명령어는 무엇인가?
4. 쿼리의 결과로 나온 레코드들의 집합을 무엇이라고 하는가?

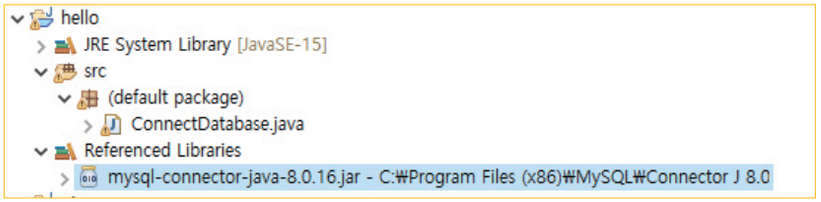
JDBC를 이용한 프로그래밍 18.4

MySQL JDBC 드라이버 Connector/J



자바와 데이터베이스를 연결하는 형태는 여러 가지가 있지만 가장 간명한 것은 해당 데이터베이스 회사에서 제공하는 JDBC 드라이버를 설치하는 것이다. JDBC 드라이버란 자바가 특정한 회사 제품의 데이터베이스에 접근할 수 있도록 해주는 드라이버의 일종이다. 우리가 비디오 카드를 구매하였어도, 비디오 카드를 구동하는 드라이버가 없으면 사용할 수 없는 것처럼 데이터베이스도 해당 데이터베이스 회사가 제공하는 JDBC 드라이버가 있어야 자바가 그 데이터베이스를 사용할 수 있다.

MySQL에서의 JDBC는 Connector/J라고 불린다. 여러분들이 앞 절에서처럼 "MySQL Installer"를 통하여 MySQL을 설치하였다면 이미 Connector/J는 설치되었다. 따라서 추가로 설치할 필요는 없다. 다만 Connector/J를 자바 가상 기계가 이 드라이버 파일을 찾을 수 있도록 하여야 한다. 클래스 경로를 나타내는 환경 변수인 CLASSPATH를 설정하거나 이클립스에서 실행한다면 외부 라이브러리를 추가하여야 한다. 이클립스에서 외부 라이브러리를 추가하려면 프로젝트에서 마우스 오른쪽 버튼을 눌러서 [Build Path] → [Add External Archives...]를 선택하고 C:\Program Files (x86)\MySQL\Connector J 8.0 폴더에 있는 mysql-connector-java-8.0.16.jar을 선택한다. 버전 번호는 다를 수 있다.



JDBC를 이용한 데이터베이스 사용 절차

JDBC를 이용하여서 데이터베이스를 사용하는 전형적인 절차는 다음과 같다.

1. URL로 지정된 JDBC 드라이버를 적재(load)한다.
2. 사용자 이름과 패스워드를 가지고 데이터베이스에 연결한다.
3. SQL 문장을 작성하여 전송하고 실행한다. SQL 명령어의 결과로 생성되는 결과 집합을 얻는다.
4. 결과 집합을 화면에 표시하거나 결과 집합을 처리한다. 사용이 끝나면 연결을 해제한다.

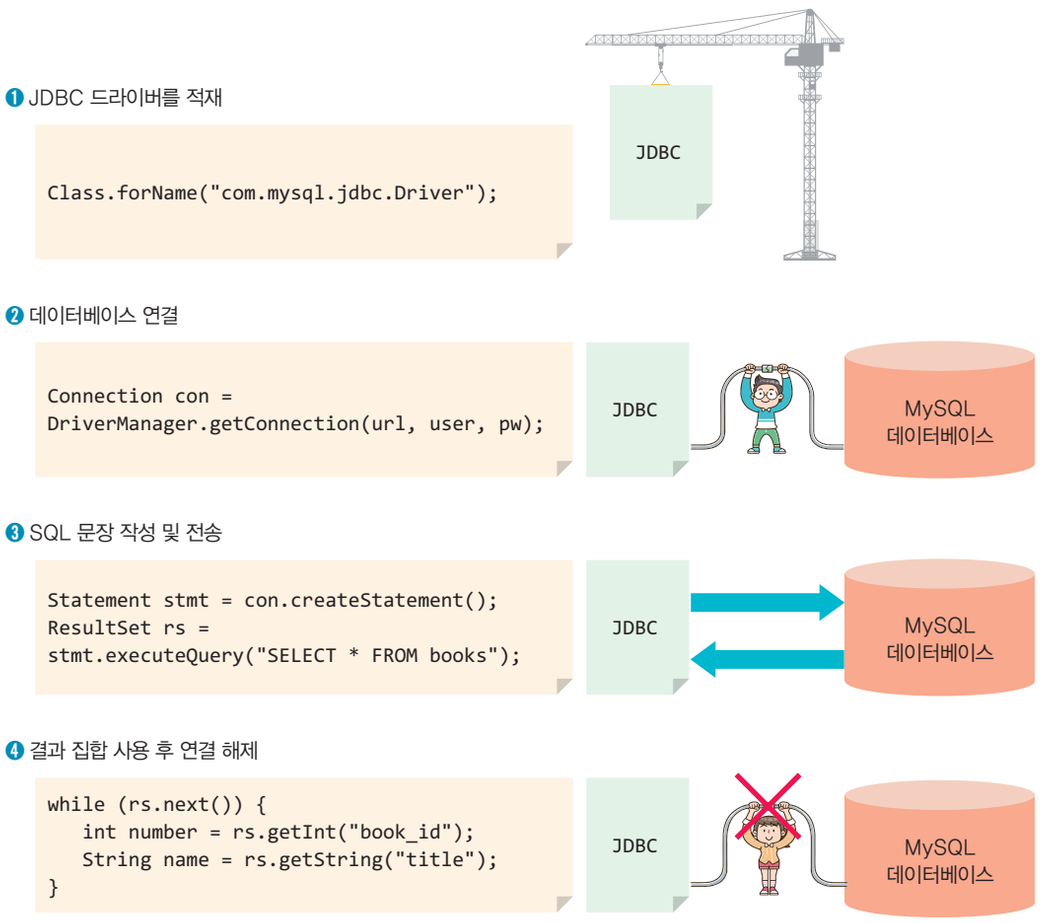


그림 18.7 JDBC를 이용한 데이터베이스 사용 절차

드라이버 클래스 적재

첫 번째 단계는 드라이버 클래스를 등록하여서 프로그램 안으로 드라이버 클래스 파일을 적재하여야 하는 것이다. 흔히 JDBC 드라이버는 동적으로 적재하고 초기화한다. 동적으로 라이브러리를 적재하려면 Class 클래스의 forName()이라는 메소드를 사용한다. Class 클래스는 JVM에서 사용하는 클래스들의 정보를 가지고 있는 메타 클래스이다. 예를 들어서 우리가 설치한 MySQL JDBC 드라이버를 등록하려면 다음과 같은 문장을 사용한다.

```
try {
   Class.forName("com.mysql.cj.jdbc.Driver");
}
catch (ClassNotFoundException e) {
   System.out.println("드라이버를 찾을 수 없습니다.");
}
```

forName() 메소드는 ClassNotFoundException을 발생할 수 있기 때문에, 반드시 try/catch 블록을 사용하여야 한다. JDBC 드라이버의 인스턴스는 우리가 생성하지 않는다. Class.forName()을 호출하면 드라이버 클래스가 정적 블록을 통해 자신의 인스턴스를 생성하고 초기화한 후에 DriverManager에 등록한다. JDBC 4.0 이후 버전부터는 forName() 메소드를 호출하지 않아도 클래스 경로에서 사용할 수 있는 드라이버가 자동으로 적재되지만 만약 이전 버전을 사용한다면 forName() 메소드를 호출하여야 한다. 여기에서는 안전하게 forName() 메소드를 사용하였다.

데이터베이스 연결

드라이버를 적재하였으면 드라이버를 통하여 데이터베이스 시스템과 연결할 수 있다. 연결하기 위해서는 DriverManager 클래스의 정적 메소드인 getConnection()을 호출한다. 이 메소드는 데이터베이스 연결을 확립하게 되고 매개 변수로 (데이터베이스 URL, 사용자 아이디, 패스워드)를 요구한다.

```
String url = "jdbc:mysql://localhost:3306/book_db?characterEncoding=UTF-8 &
               serverTimezone=UTC";
String user = "root";
String password = "password";
con = DriverManager.getConnection(url, user, password);
```

URL 매개 변수는 다음과 같은 문법을 가진다.

```
jdbc:subprotocol:subname
```

여기서 만약 MySQL을 사용하고 있다면 subprotocol은 mysql이다. subname은 데이터베이스 이름이다. 만약 네트워크에 있는 데이터베이스 파일이라면 완전한 URL이 된다. 로컬 컴퓨터에 있는 데이터베이스라면 //localhost:3306/book_db와 같이 된다. 여기서 3306은 포트 번호이다. 최근 MySQL 버전에서는 반드시 문자 엔코딩과 서버 기준 시각을 붙여서 보내주어야 한다.

사용자 아이디와 패스워드는 물론 데이터베이스 서버에 등록되어 있어야 한다. MySQL의 경우에는 설치 시에 아이디와 패스워드를 설정할 수 있다. 아이디는 항상 “root”이고 패스워드는 설치 시 지정한 패스워드이다. getConnection() 메소드는 SQLException을 발생할 수 있다.

예제 18-1

이제는 이 모든 것을 한데 묶어보자. 앞에서 설명한대로 [MySQL 8.0 Command Line Client]를 사용하여서 book_db를 미리 생성하여야 오류가 발생하지 않는다.

ConnectDatabase.java

```
01  import java.sql.*;
02
03  public class ConnectDatabase {
04     public static Connection makeConnection() {
05        String url = "jdbc:mysql://localhost:3306/book_db?characterEncoding=UTF-8
06                                                          & serverTimezone=UTC";
07        String id = "root";
08        String password = "password";
09        Connection con = null;
10        try {
11           Class.forName("com.mysql.cj.jdbc.Driver");
12           System.out.println("드라이버 적재 성공");
13           con = DriverManager.getConnection(url, id, password);
14           System.out.println("데이터베이스 연결 성공");
15        } catch (ClassNotFoundException e) {
16           System.out.println("드라이버를 찾을 수 없습니다.");
17        } catch (SQLException e) {
18           System.out.println("연결에 실패하였습니다.");
19        }
20        return con;
21     }
22
23     public static void main(String arg[]) throws SQLException {
24        Connection con = makeConnection();
25        con.close();
26     }
27  }
```

실행 결과

```
드라이버 적재 성공
데이터베이스 연결 성공
```

만약 "드라이버를 찾을 수 없습니다."라는 오류 메시지가 나오면 "Connector/J"가 제대로 설치되지 않은 것이다. 또한 "연결에 실패하였습니다." 메시지가 나오면 MySQL 안에 book_db를 올바르게 생성하였는지 체크한다.

SQL 문장 실행

데이터베이스를 연결한 후에는 SELECT와 같은 SQL 문장들을 실행할 수 있다. 이때 사용되는 것이 Connection, Statement, ResultSet 인터페이스이다.

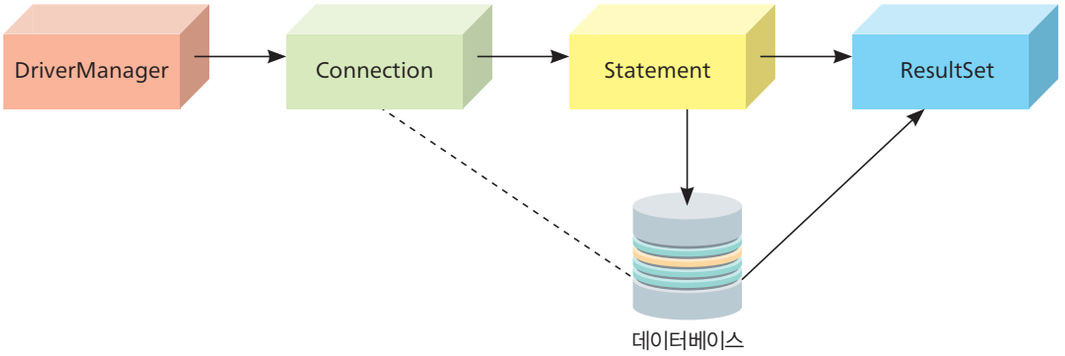


그림 18.8 Connection, Statement, ResultSet 인터페이스의 역할

예를 들어서 SELECT 문장을 실행시키려면 다음과 같은 문장을 사용한다.

```
Statement s = con.createStatement();          // 문장 객체 생성
String query = "SELECT * FROM books ORDER BY book_id";       // SQL 문장 생성
ResultSet rows = s.executeQuery(query);       // SQL 문장 실행
```

여기서 쿼리의 실행으로 생성되는 결과 집합은 rows라는 변수에 저장된다.

결과 집합에서 이동

다음 단계는 결과 집합에서 레코드를 하나씩 접근하여 작업을 해야 한다. executeQuery 메소드에 의하여 반환된 ResultSet 객체에는 SELECT 문장에 의하여 추출된 모든 레코드가 들어 있다. 하지만 우리는 한 번에 하나의 레코드만 접근할 수 있다. 이것을 위하여 커서(cursor)라는 포인터가 제공된다. 이 커서를 움직이는 다양한 메소드가 제공된다. 예를 들어 결과 집합에서 레코드를 하나씩 처리하는 예는 다음과 같다.

```
while(rows.next())
{
   // 현재 레코드를 처리한다.
   int id = rs.getInt("book_id");
   ...
}
```

커서 객체는 previous(), first(), last(), absolute(), relative(), beforeFirst(), afterLast()와 같은 다양한 이동 메소드들을 가지고 있다.

결과 집합 처리

다음 단계는 레코드에서 컬럼의 값을 추출하는 단계이다. 이것을 위하여 많은 메소드들이 준비되어 있다. 이 메소드들은 두 가지의 카테고리로 나눌 수 있다. 하나는 컬럼을 이름으로 접근하고 다른 하나는 컬럼을 번호로 접근한다. 만약 번호를 안다면 숫자로 접근하는 것이 더 효율적이다. 현재 레코드에서 학번과 이름을 추출하는 코드를 작성하여 보면 다음과 같다.

```
int id = row.getInt("id");
String name = row.getString("name");
```

다음 코드에서는 동일한 작업을 컬럼 번호를 이용하여서 수행한다.

```
int id = row.getInt(1);
String name = row.getString(2);
```

자바에서는 인덱스 번호가 0부터 시작하지만 SQL에서는 1부터 시작한다. 레코드에서 컬럼값을 추출하는 메소드는 getXXX()와 같은 형태를 가진다.

예제 18-2

앞에서 작성한 데이터베이스에서 책을 전부 검색하여 콘솔에 출력하는 프로그램을 작성해 보자.

SQLSelectTest.java

```
01 import java.sql.*;
02
03 public class ConnectDatabase {
04    public static Connection makeConnection() {
05       String url = "jdbc:mysql:
06         //localhost:3306/book_db?characterEncoding=UTF-8&serverTimezone=UTC";
07       String id = "root";
08       String password = "password";
09       Connection con = null;
10       try {
11          Class.forName("com.mysql.cj.jdbc.Driver");
12          System.out.println("드라이버 적재 성공");
13          con = DriverManager.getConnection(url, id, password);
14          System.out.println("데이터베이스 연결 성공");
```

```
15        } catch (ClassNotFoundException e) {
16           System.out.println("드라이버를 찾을 수 없습니다.");
17        } catch (SQLException e) {
18           System.out.println("연결에 실패하였습니다.");
19        }
20        return con;
21     }
22
23       public static void main(String arg[]) throws SQLException {
24             Connection con = makeConnection();
25             Statement stmt = con.createStatement();
26             ResultSet rs = stmt.executeQuery("SELECT * FROM books");
27             while (rs.next()) {
28                    int id = rs.getInt("book_id");
29                    String title = rs.getString("title");
30                    System.out.println(id + " " + title);
31             }
32             con.close();
33          }
34  }
```

실행 결과

```
드라이버 적재 성공
데이터베이스 연결 성공
1 Operating System Concepts
2 Head First PHP and MYSQL
3 C Programming Language
4 Head First SQL
```

레코드 추가, 수정, 삭제

만약 레코드를 추가하거나 수정, 삭제하려면 executeUpdate() 메소드를 사용하여야 한다. executeUpdate() 메소드는 얼마나 많은 레코드들이 변경되었는지를 반환한다. 이 반환값을 이용하여서 데이터들이 제대로 추가되었는지를 확인할 수 있다. 예를 들어 책의 이름, 출판사, 발행연도, 가격을 받아서 테이블에 추가하는 함수를 작성해보면 다음과 같다.

SQLInsertTest.java

```
01  import java.sql.*;
02
03  public class SQLInsertTest {
04     public static Connection makeConnection() {
05              ...// 전과 동일
06        }
07
```

```
08      public static void main(String arg[]) {
09         addBook("Artificial Intellegence", "Addison Wesley", "2002", 35000);
10      }
11
12      private static void addBook(String title, String publisher, String year,
13                                                                    int price) {
14         Connection con = makeConnection();
15         try {
16            Statement stmt = con.createStatement();
17            String s = "INSERT INTO books (title, publisher, year, price) VALUES ";
18            s += "('" + title + "','" + publisher + "','" + year + "','"
19                                                              + price + "')";
20            System.out.println(s);
21            int i = stmt.executeUpdate(s);
22            if (i == 1)
23               System.out.println("레코드 추가 성공");
24            else
25               System.out.println("레코드 추가 실패");
26         } catch (SQLException e) {
27            System.out.println(e.getMessage());
28            System.exit(0);
29         }
30            con.close();
31      }
32   }
```

실행 결과

```
드라이버 적재 성공
데이터베이스 연결 성공
INSERT INTO books (title, publisher, year, price) VALUES ('Artificial
                                    Intellegence','Addison Wesley','2002','35000')
레코드 추가 성공
```

makeConnection()은 이전에 등장하였던 함수와 동일하다. 연결이 성립되고 난 후에 Statement 객체가 생성되고 매개 변수로 전달된 값들을 이용하여 INSERT 문장이 구성된다. 이어서 executeUpdate() 메소드가 호출되어서 INSERT 문장을 실행하게 된다. 만약 반환값이 1이면 올바르게 레코드가 추가된 것이고 그렇지 않으면 추가에 실패한 것이다. UPDATE나 DELETE 문장도 같은 방법으로 실행할 수 있다. SQL 문장을 구성한 후에는 콘솔에 출력해보는 것이 디버깅에 도움이 된다.

중간점검

1. JDBC를 이용하여서 데이터베이스를 사용하는 전형적인 절차를 말해보자.
2. JDBC 드라이버를 동적으로 적재할 때 사용되는 메소드는?
3. SQL 문장이 저장되는 객체는?
4. 결과 집합이 저장되는 객체는?

Prepared Statements 사용하기 18.5

Prepared Statements는 많이 사용되는 SQL 문장이 있는 경우에 이것을 미리 SQL 문장으로 만들어두고 필요할 때마다 사용하는 기법이다. 예를 들어서 데이터베이스에 저장된 책 중에서 특정한 출판사의 책만을 찾는 쿼리를 Prepared Statements로 작성해보자.

```
SELECT books.title, books.price
   FROM books
   WHERE publisher=외부에서 제공
```

이 쿼리를 미리 만들어두고 필요할 때마다 사용해보자. Prepared Statements를 사용하면 성능이 향상된다. 외부에서 제공되는 값은 ?로 표시된다. 위와 같은 SQL 문장은 다음과 같이 Prepared Statements로 만들 수 있다.

```
String query =
   "SELECT books.title, books.price" +
   " FROM books" +
   " WHERE publisher = ?";

PreparedStatement stmt= con.prepareStatement(query);
```

Prepared Statements을 실행하기 전에 ?에 해당하는 값을 주어야 한다.

```
stmt.setString(1, "Wiley");
```

첫 번째 인수는 ? 변수의 번호이다. 두 번째 인수는 ? 변수의 값이다. 이렇게만 해놓으면 언제든지 필요할 때마다 Prepared Statements를 실행할 수 있다.

```
ResultSet rs =  stmt.executeQuery();
```

만약 INSERT, DELETE, UPDATE 문장을 사용하였다면 executeUpdate()를 호출하여야 한다.

예제 18-3

Prepared Statements 기능을 사용하여서 많이 사용되는 쿼리 문장을 미리 만들어두고 필요할 때마다 사용해 보자.

SQLPreparedTest.java

```
01  import java.sql.*;
02
03  public class SQLPreparedTest {
04     public static Connection makeConnection() {
05        ...
06     }
07
08     public static void main(String arg[]) throws SQLException {
09        Connection con = makeConnection();
10        String query = "SELECT books.title, books.price" + " FROM books"
11                                                    + " WHERE publisher = ?";
12
13        PreparedStatement stmt = con.prepareStatement(query);
14        stmt.setString(1, "Wiley");
15
16        ResultSet rs = stmt.executeQuery();
17        while (rs.next()) {
18           String title = rs.getString("title");
19           int price = rs.getInt("price");
20           System.out.println(title + " " + price);
21        }
22        con.close();
23     }
24  }
```

실행 결과

```
드라이버 적재 성공
데이터베이스 연결 성공
Operating System Concepts 30700
```

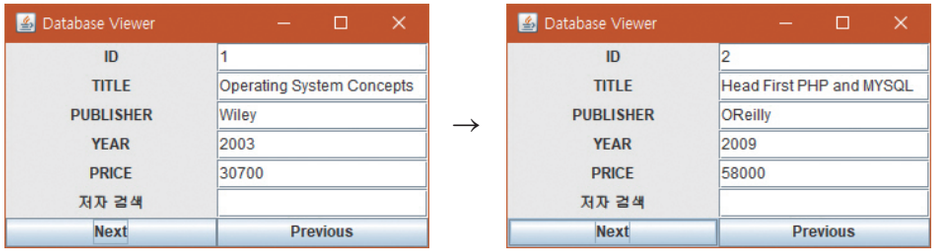
중간점검

1. Prepared Statements는 무엇인가?
2. Prepared Statements는 왜 사용하는가?

Introduction to JAVA Programming

GUI로 데이터베이스 내용 표시하기 LAB

다음과 같이 그래픽 사용자 인터페이스를 이용하여서 데이터베이스 테이블의 내용을 화면에 표시하는 프로그램을 작성하여 보자.



버튼이 눌려질 때마다 결과 집합에서 앞의 레코드로 가거나 뒤 레코드로 가면 된다. previous()와 next() 메소드를 사용한다.

Solution GUI로 데이터베이스 내용 표시하기

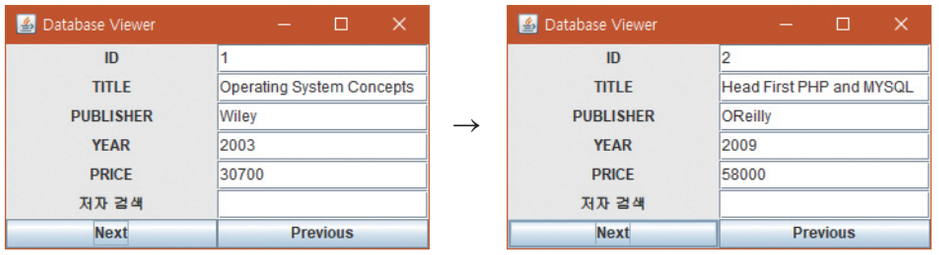
Introduction to **JAVA Programming**

SQLSelectTest.java

```
01 import java.awt.GridLayout;
02 import java.awt.event.ActionEvent;
03 import java.awt.event.ActionListener;
04 import java.sql.Connection;
05 import java.sql.DriverManager;
06 import java.sql.ResultSet;
07 import java.sql.SQLException;
08 import java.sql.Statement;
09
10 import javax.swing.JButton;
11 import javax.swing.JFrame;
12 import javax.swing.JLabel;
13 import javax.swing.JTextField;
14
15 class MyFrame extends JFrame {
16    JTextField id, title, p, year, price, author;
17    JButton previousButton, nextButton, InsertButton, deleteButton,
18          searchButton;
19    ResultSet rs;
20    Statement stmt;
21
22    public MyFrame() throws SQLException {
23       super("Database Viewer");
24       Connection con = makeConnection();
25       stmt = con.createStatement();
26       rs = stmt.executeQuery("SELECT * FROM books");
27       setLayout(new GridLayout(0, 2));
28       add(new JLabel("ID", JLabel.CENTER));
29       add(id = new JTextField());
30       add(new JLabel("TITLE", JLabel.CENTER));
31       add(title = new JTextField());
32       add(new JLabel("PUBLISHER", JLabel.CENTER));
33       add(p = new JTextField());
34       add(new JLabel("YEAR", JLabel.CENTER));
35       add(year = new JTextField());
36       add(new JLabel("PRICE", JLabel.CENTER));
```

```
37          add(price = new JTextField());
38          add(new JLabel("저자 검색", JLabel.CENTER));
39          add(author = new JTextField());
40
41          previousButton = new JButton("Previous");
42          previousButton.addActionListener(new ActionListener() {
43             public void actionPerformed(ActionEvent event) {
44                try {
45                   rs.previous();
46                   id.setText("" + rs.getInt("book_id"));
47                   title.setText("" + rs.getString("title"));
48                   p.setText("" + rs.getString("publisher"));
49                   year.setText("" + rs.getString("year"));
50                   price.setText("" + rs.getInt("price"));
51                } catch (SQLException e) {
52                   e.printStackTrace();
53                }
54             }
55          });
56
57          nextButton = new JButton("Next");
58          nextButton.addActionListener(new ActionListener() {
59             public void actionPerformed(ActionEvent event) {
60                try {
61                   rs.next();
62                   id.setText("" + rs.getInt("book_id"));
63                   title.setText("" + rs.getString("title"));
64                   p.setText("" + rs.getString("publisher"));
65                   year.setText("" + rs.getString("year"));
66                   price.setText("" + rs.getInt("price"));
67
68                } catch (SQLException e) {
69                   e.printStackTrace();
70                }
71             }
72          });
73          add(nextButton);
74          add(previousButton);
75          setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
76          setSize(350, 200);
77          // pack();
78          setVisible(true);
79       }
```

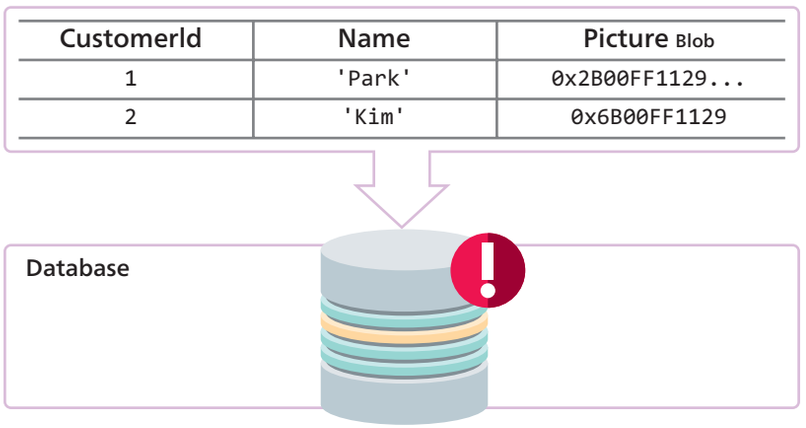
```
80
81      public static Connection makeConnection() {
82          String url = "jdbc:mysql:
83              //localhost:3306/book_db?characterEncoding=UTF-8&serverTimezone=UTC";
84          String id = "root";
85          String password = "password";
86          Connection con = null;
87          try {
88              Class.forName("com.mysql.cj.jdbc.Driver");
89              System.out.println("드라이버 적재 성공");
90              con = DriverManager.getConnection(url, id, password);
91              System.out.println("데이터베이스 연결 성공");
92          } catch (ClassNotFoundException e) {
93              System.out.println("드라이버를 찾을 수 없습니다.");
94          } catch (SQLException e) {
95              System.out.println("연결에 실패하였습니다.");
96          }
97          return con;
98      }
99  }
100
101 public class ConnectDatabase {
102     public static void main(String[] args) throws SQLException {
103         new MyFrame();
104     }
105 }
```



도전문제 새로운 레코드를 추가하는 버튼을 추가할 수 있는가? 기능도 구현되어야 한다.

JDBC를 사용하여 이미지 저장하기 18.6

PreparedStatement 인터페이스의 setBinaryStream() 메소드는 매개 변수의 인덱스를 나타내는 정수와 InputStream 객체를 받아들이고 매개 변수를 주어진 InputStream 객체로 설정한다. 매우 큰 바이너리 값을 보내야 할 때마다 이 방법을 사용할 수 있다. SQL 데이터베이스는 Blob(Binary Large Object)이라는 데이터 유형을 제공하므로 이미지와 같은 대용량 이진 데이터를 저장할 수 있다.



ΞΔΒΨ를 사용하여 이미지 저장

JDBC 프로그램을 사용하여 데이터베이스에 이미지를 저장해야 하는 경우 아래와 같이 Blob 데이터 유형으로 테이블을 생성한다.

```
CREATE TABLE Animals(Name VARCHAR(255), Type VARCHAR(50), Image BLOB);
```

이제 JDBC를 사용하여 데이터베이스에 연결하고 위의 생성된 테이블에 값을 삽입할 Prepared Statement를 준비한다.

```
String query = aINSERT INTO Animals(Name, Type, Image) VALUES (?, ?, ?)";
PreparedStatement pstmt = con.prepareStatement(query);
```

PreparedStatement 인터페이스의 설정자 메서드를 사용하여 자리 표시자에 값을 설정하고 setBinaryStream() 메서드를 사용하여 Blob 데이터 유형 설정 값을 설정한다.

```
FileInputStream fin = new FileInputStream("dog.jpg");
pstmt.setBinaryStream(3, fin);
```

예제 18-4

다음은 JDBC 프로그램을 사용하여 MySQL 데이터베이스에 이미지를 삽입하는 방법을 보여주는 예제이다. 여기서 테이블에 Blob 데이터 유형을 만들고, 테이블의 내용을 검색한다.

```
...
public class ImageDatabase {
   public static Connection makeConnection() {
      //...
   }
   public static void main(String args[]) throws Exception {

      Connection con = makeConnection();
      Statement stmt = con.createStatement();
      //stmt.execute("DROP TABLE Animals");

      String createTable = "CREATE TABLE Animals( "
         + "Name VARCHAR(255), "
         + "Type VARCHAR(50), "
         + "Image BLOB)";
      stmt.execute(createTable);

      String query = "INSERT INTO Animals(Name, Type, Image) VALUES (?, ?, ?)";
      PreparedStatement pstmt = con.prepareStatement(query);
      pstmt.setString(1, "Dog");
      pstmt.setString(2, "WHITE DOG");
      FileInputStream fin = new FileInputStream("dog1.jpg");
      pstmt.setBinaryStream(3, fin);
      pstmt.execute();

      pstmt.setString(1, "Cat");
      pstmt.setString(2, "GRAY CAT");
      fin = new FileInputStream("cat1.jpg");
      pstmt.setBinaryStream(3, fin);
      pstmt.execute();
```

```
31        System.out.println("데이터가 추가되었습니다.");
32        ResultSet rs = stmt.executeQuery("Select *from Animals");
33        while(rs.next()) {
34           System.out.print("Name: "+rs.getString("Name")+", ");
35           System.out.print("Type: "+rs.getString("Type")+", ");
36           System.out.print("Image: "+rs.getBlob("Image"));
37           System.out.println();
38        }
39        con.close();
40     }
41  }
```

```
드라이버 적재 성공
데이터베이스 연결 성공
데이터가 추가되었습니다.
Name: Dog, Type: WHITE DOG, Image: com.mysql.cj.jdbc.Blob@327514f
Name: Cat, Type: GRAY CAT, Image: com.mysql.cj.jdbc.Blob@5b12b668
```

참고 JDBC 프로그램을 사용하여 .gif 또는 .jpeg 또는 .png 유형의 이미지만 저장하고 검색할 수 있다.

JDBC를 사용하여 이미지 추출

```
01  import java.io.FileOutputStream;
02  import java.sql.Blob;
03  import java.sql.Connection;
04  import java.sql.DriverManager;
05  import java.sql.PreparedStatement;
06  import java.sql.ResultSet;
07
08  public class ImageDatabase2 {
09     public static Connection makeConnection() {
10        ...
11     }
12     public static void main(String args[]) throws Exception {
13        Connection con = makeConnection();
14        PreparedStatement ps = con.prepareStatement("select * from Animals");
15        ResultSet rs = ps.executeQuery();
```

```
16          if (rs.next()) {
17
18              Blob b = rs.getBlob(3);
19              byte barr[] = b.getBytes(1, (int) b.length());
20
21              FileOutputStream fout = new FileOutputStream("dog11.jpg");
22              fout.write(barr);
23
24              fout.close();
25          }
26          System.out.println("이미지를 추출하였습니다.");
27          con.close();
28      }
29  }
```

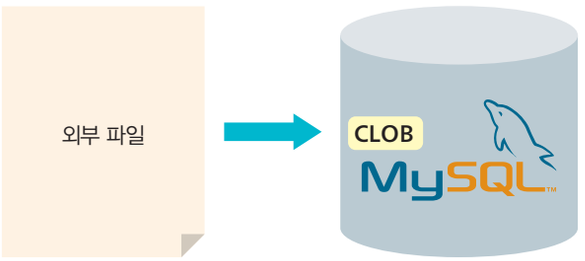
```
드라이버 적재 성공
데이터베이스 연결 성공
이미지를 추출하였습니다.
```

중간점검

1. 데이터베이스에 이미지를 저장하려면 SQL의 어떤 데이터 유형을 사용해야 하는가?
2. ResultSet에서 Blob를 추출하는 메소드 이름은 무엇인가?

JDBC를 사용하여 텍스트 파일 저장하기 18.7

데이터베이스에는 텍스트 파일도 저장할 수 있다. MySQL 데이터베이스에서 CLOB(TINYTEXT, TEXT, MEDIUMTEXT, LONGTEXT) 데이터 유형으로 저장하면 된다. JDBC는 데이터베이스의 테이블에 파일을 저장하기 위해 CLOB 데이터 유형에 대한 지원을 제공한다. PreparedStatement 인터페이스의 setCharacterStream() 메소드는 인덱스를 나타내는 정수와 Reader 객체를 매개변수로 받아서 Reader 객체(파일)의 내용을 지정된 인덱스에 저장한다.



JDBC를 사용하여 텍스트 파일 저장

JDBC 프로그램을 사용하여 데이터베이스에 파일을 저장해야 하는 경우 아래와 같이 CLOB(TINYTEXT, TEXT, MEDIUMTEXT, LONGTEXT) 데이터 유형으로 테이블을 생성한다.

```
CREATE TABLE Articles(Name VARCHAR(255), Article LONGTEXT);
```

JDBC를 사용하여 데이터베이스에 연결하고 위의 생성된 테이블에 값을 삽입할 Prepared Statement를 준비한다.

```
String query = "INSERT INTO Tutorial(Name, Article) VALUES (? , ?)";
PreparedStatement pstmt = con.prepareStatement(query);
```

PreparedStatement 인터페이스의 설정자 메소드를 사용하여 자리 표시자에 값을 설정하고 setCharacterStream() 메소드를 사용하여 CLOB 데이터 유형 값을 설정한다.

예제 18-5

CLOB를 이용하여 작업 디렉터리에 있는 test.txt 파일을 데이터베이스 테이블 안에 저장해보자. 한글이 들어간 파일의 경우에는 반드시 UTF-8로 저장된 파일이어야 한다. 코드가 다르면 데이터베이스 안에서 한글이 깨지게 된다.

```
01  import java.io.File;
02  import java.io.FileReader;
03  import java.nio.charset.StandardCharsets;
04  import java.sql.Connection;
05  import java.sql.DriverManager;
06  import java.sql.PreparedStatement;
07  import java.sql.SQLException;
08  import java.sql.Statement;
09
10  public class TextDatabase {
11     public static Connection makeConnection() {
12        String url = "jdbc:mysql:
13          //localhost:3306/book_db?characterEncoding=UTF-8&serverTimezone=UTC";
14        String id = "root";
15        String password = "1234";
16        Connection con = null;
17        try {
18           Class.forName("com.mysql.cj.jdbc.Driver");
19           System.out.println("드라이버 적재 성공");
20           con = DriverManager.getConnection(url, id, password);
21           System.out.println("데이터베이스 연결 성공");
22        } catch (ClassNotFoundException e) {
23           System.out.println("드라이버를 찾을 수 없습니다.");
24        } catch (SQLException e) {
25           System.out.println("연결에 실패하였습니다.");
26        }
27        return con;
28     }
29
30     public static void main(String args[]) throws Exception {
31        Connection con = makeConnection();
32
33        Statement stmt = con.createStatement();
34        // String dropTable = "drop table Articles;";
35        // stmt.execute(dropTable);
36        String createTable = "create table Articles(Name VARCHAR(255),
37                                                    Article LONGTEXT);";
38        stmt.execute(createTable);
39
40        String query = "insert into Articles " + "values (?,?)";
41        PreparedStatement pstmt = con.prepareStatement(query);
```

```
42
43          try {
44              File file = new File("test.txt");
45              FileReader fileReader = new FileReader(file, StandardCharsets.UTF_8);
46
47              pstmt.setInt(1, 2);
48              pstmt.setCharacterStream(2, fileReader, (int) file.length());
49
50              pstmt.executeUpdate();
51              System.out.println("파일 저장 성공");
52
53              pstmt.close();
54              con.close();
55          } catch (Exception e) {
56              e.printStackTrace();
57          }
58      }
59  }
```

실행 결과

```
드라이버 적재 성공
데이터베이스 연결 성공
파일 저장 성공
```

JDBC를 사용하여 텍스트 추출

앞에서 저장하였던 텍스트 파일을 꺼내서 작업 디렉터리에 다른 이름으로 저장해보자.

```
01  import java.io.FileWriter;
02  import java.io.Reader;
03  import java.sql.Clob;
04  import java.sql.Connection;
05  import java.sql.DriverManager;
06  import java.sql.PreparedStatement;
07  import java.sql.ResultSet;
08  import java.sql.SQLException;
09
10  public class TextDatabase2 {
11      public static Connection makeConnection() {
12          String url = "jdbc:mysql:
13              //localhost:3306/book_db?characterEncoding=UTF-8&serverTimezone=UTC";
14          String id = "root";
15          String password = "1234";
16          Connection con = null;
17          try {
18              Class.forName("com.mysql.cj.jdbc.Driver");
19              System.out.println("드라이버 적재 성공");
```

```
20            con = DriverManager.getConnection(url, id, password);
21            System.out.println("데이터베이스 연결 성공");
22         } catch (ClassNotFoundException e) {
23            System.out.println("드라이버를 찾을 수 없습니다.");
24         } catch (SQLException e) {
25            System.out.println("연결에 실패하였습니다.");
26         }
27         return con;
28      }
29
30      public static void main(String args[]) throws Exception {
31         Connection con = makeConnection();
32
33         PreparedStatement ps = con.prepareStatement("select * from Articles");
34         ResultSet rs = ps.executeQuery();
35         rs.next();
36
37         Clob c = rs.getClob(2);
38         Reader r = c.getCharacterStream();
39
40         FileWriter fw = new FileWriter("test11.txt");
41
42         int i;
43         while ((i = r.read()) != -1)
44            fw.write((char) i);
45
46         fw.close();
47         con.close();
48
49         System.out.println("텍스트가 추출되었습니다. ");
50            con.close();
51      }
52   }
```

실행 결과

```
드라이버 적재 성공
데이터베이스 연결 성공
텍스트가 추출되었습니다.
```

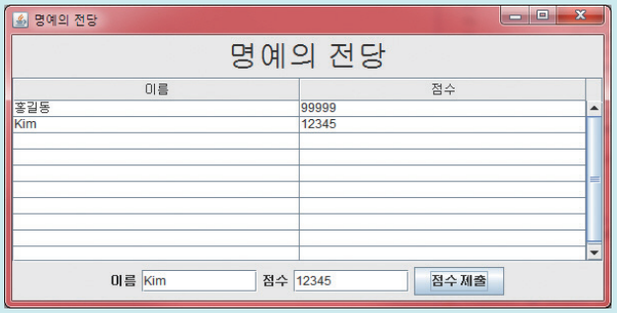
중간점검

1. 데이터베이스에 상당히 긴 텍스트를 저장하려면 SQL의 어떤 데이터 유형을 사용해야 하는가?
2. 데이터베이스의 CLOB에서 텍스트를 추출하는 절차를 설명하라.

Introduction to **JAVA Programming**

명예의 전당 Mini Project

사용자의 게임 점수를 기록하는 데이터베이스 프로그램을 생성하여 보자. 먼저 MySQL 콘솔을 실행하여서 다음과 같은 SQL 명령어를 입력한다.



JTable을 사용한다. JTabel에 대한 자세한 내용은 출판사 홈페이지에서 다운로드 받을 수 있는 부록 pdf 파일을 참조한다. 다음과 같은 SQL 명령어를 사용하여서 gamescore라고 하는 데이터베이스를 생성한다.

```
Enter password: ********
...
mysql> CREATE DATABASE gamescore;
Query OK, 1 row affected (0.00 sec)

mysql> USE gamescore;
Database changed
mysql> CREATE TABLE scores (name TEXT, score INT);
Query OK, 0 rows affected (0.00 sec)

mysql> DESCRIBE scores;
+-------+---------+------+-----+---------+-------+
| Field | Type    | Null | Key | Default | Extra |
+-------+---------+------+-----+---------+-------+
| name  | text    | YES  |     | NULL    |       |
| score | int(11) | YES  |     | NULL    |       |
+-------+---------+------+-----+---------+-------+
2 rows in set (0.02 sec)

mysql>
```

Summary

- JDBC(Java Database Connectivity)는 자바 애플리케이션과 데이터베이스를 연결하는 라이브러리이다.
- 데이터베이스 관리 시스템(DBMS)은 다수의 사용자를 위하여 데이터가 저장, 접근, 변경되는 기능을 제공한다.
- MySQL에서의 JDBC는 Connector/J라고 불린다. Connector/J를 자바 가상 기계가 이 드라이버 파일을 찾을 수 있도록 하여야 한다.
- JDBC를 이용하여서 데이터베이스를 사용하는 전형적인 절차는 다음과 같다.
- URL로 지정된 JDBC 드라이버를 적재(load)한다. → 사용자 이름과 패스워드를 가지고 데이터베이스에 연결한다. → SQL 문장을 작성하여서 전송하고 실행한다. SQL 명령어의 결과로 생성되는 결과 집합을 얻는다. → 결과 집합을 화면에 표시하거나 결과 집합을 처리한다. 사용이 끝나면 연결을 해제한다.
- Prepared Statements는 많이 사용되는 SQL 문장이 있는 경우에 이것을 미리 SQL 문장으로 만들어두고 필요할 때마다 사용하는 기법이다.
- SQL 데이터베이스는 Blob(Binary Large Object)이라는 데이터 유형을 제공하므로 이미지와 같은 대용량 이진 데이터를 저장할 수 있다.
- 데이터베이스에는 텍스트 파일도 저장할 수 있다. CLOB(TINYTEXT, TEXT, MEDIUM TEXT, LONGTEXT) 데이터 유형으로 저장하면 된다.

Exercise

1. JDBC에서 데이터베이스 드라이버를 로드하는데 사용되는 메소드는?

❶ getConnection() ❷ registerDriver()

❸ forName() ❹ setConnection()

2. 다음 중 JDBC에서 DML(Data Manipulation Language) 문을 수행하는 메소드는 무엇인가?

❶ executeResult() ❷ executeQuery()

❸ executeUpdate() ❹ execute()

3. 아래 문장의 빈칸을 채워라.

(a) 표준적인 데이터베이스 언어는 _____이다.

(b) 하나의 데이터베이스는 여러 개의 _____로 이루어진다.

(c) SQL 쿼리 문의 검색 결과를 _____ 라고 한다.

(d) SQL 쿼리 문을 실행하는 메소드는 _____ 이다.

4. 다음과 같은 질문에 답해보자. 인터넷을 통하여 조사해도 좋다.

(a) JDBC는 무엇인가?

(b) JDBC 드라이버는 무엇인가?

(c) DriverManager가 하는 일은 무엇인가?

(d) BLOB란 무엇인가?

(e) CLOB란 무엇인가?

(f) Class.forName()이 하는 일은 무엇인가?

(g) Statement와 PreparedStatement의 차이점은 무엇인가?

(h) execute(), executeQuery(), executeUpdate()의 차이점은 무엇인가?

(i) 데이터베이스에서 이미지를 어떻게 저장하고 검색할 수 있는가?

5. 다음의 설명에 부합하는 JDBC의 클래스 이름을 적어보자.

(a) _____ : 이 클래스는 JDBC API를 사용하여 데이터베이스에 연결하는 데 사용된다. 개발자는 해당 데이터베이스에 대한 JDBC 드라이버가 로드되고 JVM 메모리에서 초기화된 후에만 데이터베이스에 대한 연결을 얻을 수 있다.

(b) _____ : 이 클래스는 데이터베이스에 대한 여러 가지 작업을 실행하는 데 사용된다.

(c) _____ : 개발자가 JDBC API를 사용하여 쿼리를 실행한 후 쿼리 결과가 저장되는 클래스이다.

6. 자바에서 JDBC를 사용하는 올바른 순서를 나열하라.

❶ 연결을 종료한다.

❷ Statement 객체를 실행하고 쿼리 결과 집합을 반환한다.

❸ JDBC 드라이버를 로드하고 등록한다.

❹ 데이터베이스에 대한 연결을 연다.

❺ 쿼리를 수행하기 위해 Statement 객체를 생성한다.

❻ 결과 집합 및 Statement 객체를 닫는다.

❼ 결과 집합을 처리한다.

5. 자동차에 대한 데이터를 저장하는 데이터베이스를 설계하여 보자. 자동차는 유일한 등록 번호(license number), 제조사(manufacturer), 차종(type), 제작연도(year), 연비(efficiency)를 가지고 있다.

(a) Car 테이블을 생성하는 SQL을 작성하라.

(b) 몇 개의 데이터를 입력하는 SQL 문을 작성하라.

(c) 모든 레코드를 출력하는 SQL 문을 작성하라.

(d) 제작연도가 2021년인 차의 등록 번호를 순서대로 출력하는 SQL 문을 작성하라.

Introduction to **JAVA Programming**

Programming

1. JDBC를 통해 본문에 등장한 book_db 데이터베이스에 연결하고 다음과 같은 작업들을 수행하는 프로그램을 작성하라.

난이도: 중
주제
- JDBC 사용

(a) 데이터베이스에 있는 모든 테이블의 이름을 출력한다.

(b) 데이터베이스의 각 테이블에 대해 컬럼(열)을 출력한다.

(c) 테이블의 각 컬럼(열)에 대해 데이터 유형을 출력한다.

2. JDBC API와 MySQL을 사용하여 영화를 저장하는 자바 프로그램을 작성해보자.

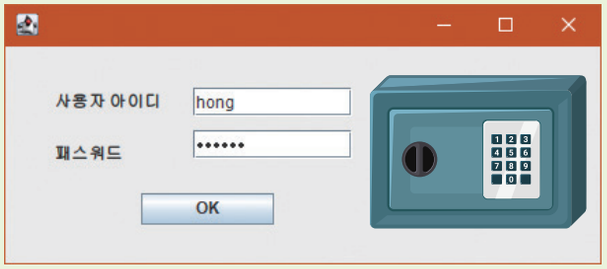
난이도: 상
주제
- JDBC 사용

- MOVIES 열이 있는 테이블을 만든다:
  id type INTEGER AUTO INCREMENT, title type VARCHAR (255), genre type VARCHAR (255), year type INTEGER.
- 세 개의 레코드를 MOVIES 테이블에 추가한다.
- 선택한 하나의 레코드를 업데이트한다.
- 지정된 ID로 선택한 레코드를 삭제한다.
- 데이터베이스의 모든 레코드를 출력한다.

3. 사용자 아이디와 패스워드를 저장하는 데이터베이스 테이블을 작성해보자. GUI 사용자 인터페이스도 함께 작성해보자. 테이블의 구조는 다음과 같다.

난이도: 상
주제
- JDBC 사용

name	password
홍길동	123456
…	…



난이도: 상

주제
- JDBC 사용

4. 스케줄을 데이터베이스에 저장하는 프로그램을 작성하여 보자. 스케줄은 제목, 날짜, 시작 시간, 종료 시간으로 구성된다. 예를 들면 다음과 같다.

- "자바 공부하기", 2022.5.27, 10:00, 12:00
- "강아지 목욕", 2022.5.28, 17:00, 18:00

스케줄을 추가하고 삭제하며, 특정한 날짜의 스케줄을 출력하는 사용자 인터페이스를 작성하라.

(a) 사용자 인터페이스를 콘솔 기반으로 작성한다.

(b) 사용자 인터페이스를 그래픽 기반으로 작성한다.

난이도: 상

주제
- JDBC 사용

5. 간단한 게시판을 만들어보자. 게시판은 메시지를 나열하며 메시지는 제목, 발송자, 발송날짜, 내용으로 구성된다. 예를 들면 다음과 같다.

- "과제 제출", 홍길동, 2022.5.27, "이번 주 과제를 금요일까지 제출,...."
- "질문", 김철수, 2022.6.10, "테이블이 두 개이고 서로 연결되어 있는 경우에는...."

메시지를 추가하고 삭제하며, 검색할 수 있는 사용자 인터페이스를 작성하라. 기타 상용 게시판에 있는 기능들을 추가하면 좋다.

(a) 사용자 인터페이스를 콘솔 기반으로 작성한다.

(b) 사용자 인터페이스를 그래픽 기반으로 작성한다.