

# 인덱스 실습

1. primary 키나, unique한 필드 없이 테이블 생성

```
create table usertbl  
(userid varchar(8) not null ,  
name varchar(10) not null,  
birthYear int not null,  
city varchar(5) not null);
```

2. 데이터 추가

```
insert into usertbl values ('zzz','name3',1990,'서울');  
insert into usertbl values ('aaa','name2',1990,'서울');  
insert into usertbl values ('ppp','name1',1990,'서울');
```

- 알파벳 순이 아니라 임의의 순서로 insert하고  
테이블의 index 와 데이터 확인

# 인덱스 실습

3. 테이블 삭제후 userid 를 unique 로 두고 테이블 재생성,  
userid2 보조 인덱스 추가

```
create table usertbl  
(userid varchar(8) unique not null ,  
name varchar(10) not null,  
birthYear int not null,  
city varchar(5) not null);
```

```
show index from usertbl;  
alter table usertbl add column userid2 varchar(8) not null unique;
```

4. 데이터 추가

```
insert into usertbl values ('zzz','name3',1990,'서울','aaa');  
insert into usertbl values ('aaa','name2',1990,'서울','ccc');  
insert into usertbl values ('ppp','name1',1990,'서울','bbb');
```

- 첫번째 인덱스인 userid 를 임의의 순서로 insert하고  
테이블의 index 와 데이터 확인

4. primary key 가 없는 경우에는 어떤 컬럼이 인덱스가 되는지 확인해보자.

```
drop index userid on usertbl;
```

```
-- 데이터 확인 --
```

```
create table usertbl  
(userid varchar(8) unique not null ,  
name varchar(10) unique not null,  
birthYear int not null,  
city varchar(5) not null);
```

```
alter table usertbl add column userid2 varchar(8) not null primary key;  
ALTER TABLE usertbl drop primay key;  
alter table customer add constraint pk_name primary key(name);
```

: 영어사전을 한영사전으로 변환한 결과와 같음



## Chapter 05

# 데이터베이스 프로그래밍

MySQL로 배우는 데이터베이스 개론과 실습

# 목차

**01**

데이터베이스 프로그래밍의 개념

**02**

저장 프로그램

**03**

데이터베이스 연동 자바 프로그래밍

**04**

데이터베이스 연동 웹 프로그래밍

# 학습목표

- ❖ 데이터베이스 프로그래밍의 개념을 이해한다.
- ❖ 저장 프로그램의 문법과 사용 방법을 알아본다.
- ❖ 자바 프로그램과 데이터베이스를 연동하는 방법을 알아본다.
- ❖ JSP 프로그램과 데이터베이스를 연동하는 방법을 알아본다.

## 01 데이터베이스 프로그래밍의 개념



# 데이터베이스 프로그래밍의 개념

## ❖ 프로그래밍이란?

- 프로그램을 설계하고 소스코드를 작성하여 디버깅하는 과정

## ❖ 데이터베이스 프로그래밍이란?

- DBMS에 데이터를 정의하고 저장된 데이터를 읽어와 데이터를 변경하는 프로그램을 작성하는 과정
- 일반 프로그래밍과는 데이터베이스 언어인 SQL을 포함한다는 점이 다름

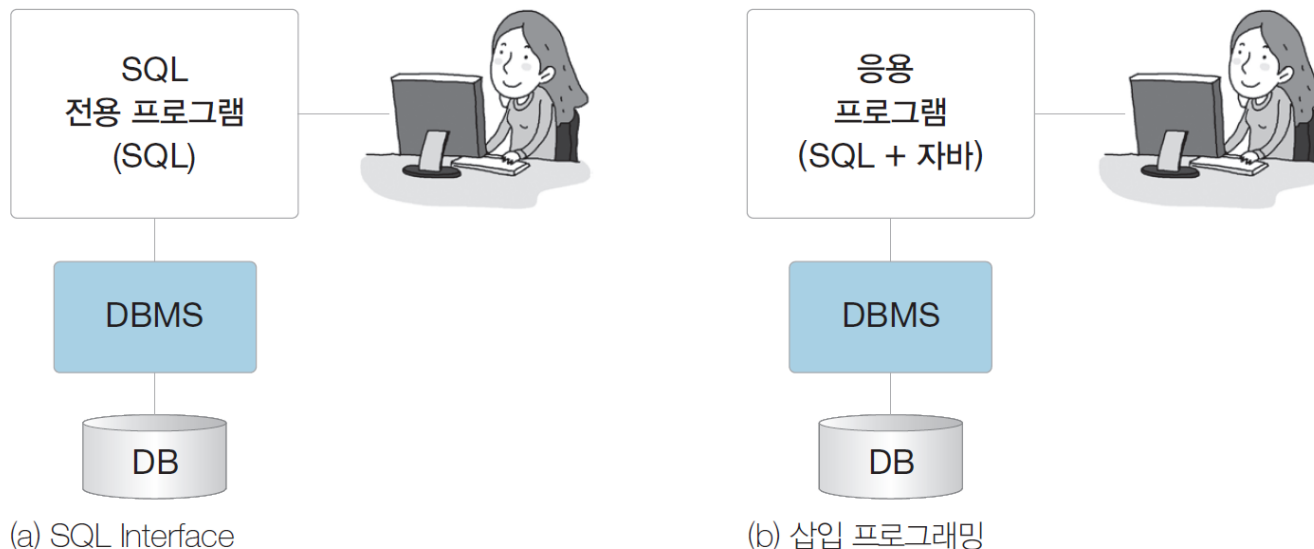


그림 5-1 데이터베이스 프로그래밍



# 데이터베이스 프로그래밍의 개념

## ❖ 데이터베이스 프로그래밍 방법

### ❶ SQL 전용 언어를 사용하는 방법

SQL 자체의 기능을 확장하여 변수, 제어, 입출력 등의 기능을 추가한 새로운 언어를 사용하는 방법.  
오라클은은 저장 프로그램 언어를 사용하며, SQL Server는 T-SQL이라는 언어를 사용함.

### ❷ 일반 프로그래밍 언어에 SQL을 삽입하여 사용하는 방법

자바, C, C++ 등 일반 프로그래밍 언어에 SQL 삽입하여 사용하는 방법.

일반 프로그래밍 언어로 작성된 응용 프로그램에서 데이터베이스에 저장된 데이터를 관리, 검색함.  
삽입된 SQL문은 DBMS의 컴파일러가 처리함.

### ❸ 웹 프로그래밍 언어에 SQL을 삽입하여 사용하는 방법

호스트 언어가 JSP, ASP, PHP 등 웹 스크립트 언어인 경우다.

### ❹ 4GL(4th Generation Language)

데이터베이스 관리 기능과 비주얼 프로그래밍 기능을 갖춘 'GUI 기반 소프트웨어 개발 도구'를 사용하여 프로그래밍하는 방법. Delphi, Power Builder, Visual Basic 등이 있음.

# 데이터베이스 프로그래밍의 개념

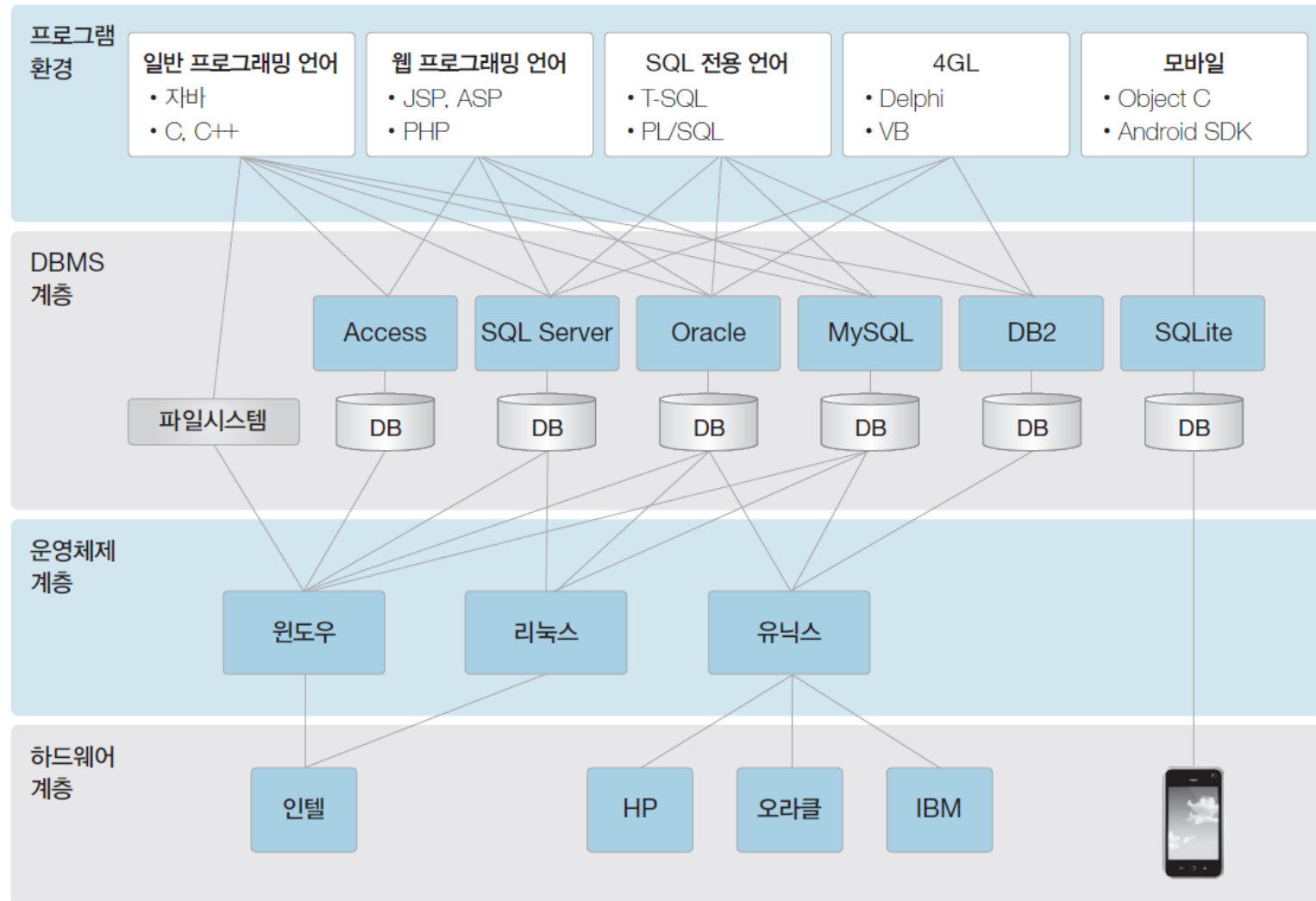


그림 5-2 DBMS 플랫폼과 데이터베이스 프로그래밍의 유형

# 데이터베이스 프로그래밍의 개념

표 5-1 DBMS의 종류와 특징

특징	Access	SQL Server	Oracle	MySQL	DB2	SQLite
제조사	마이크로소프트사	마이크로소프트사	오라클사	오라클사	IBM사	리처드 힙 (오픈소스)
운영체제 기반	윈도우	윈도우 리눅스	윈도우, 유닉스, 리눅스	윈도우, 유닉스, 리눅스	유닉스	모바일 OS (안드로이드, iOS 등)
용도	개인용 DBMS	윈도우 기반 기업용 DBMS	대용량 데이터 베이스를 위한 응용	소용량 데이터 베이스를 위한 응용	대용량 데이터 베이스를 위한 응용	모바일 전용 데이터베이스

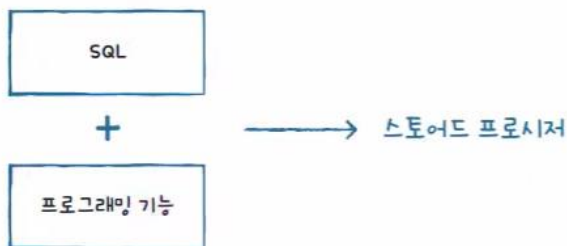
## 02. 저장 프로그램

1. 저장 프로그램
2. 트리거
3. 사용자 정의 함수
4. 저장 프로그램 문법 요약



# 1. 저장 프로그램

- 저장 프로그램(Stored Program) : 데이터베이스 응용 프로그램을 작성하는 데 사용하는 MySQL의 SQL 전용 언어(저장루틴(프로시저, 함수), 트리거, 이벤트로 구성)
- SQL 문에 변수, 제어, 입출력 등의 프로그래밍 기능을 추가하여 SQL 만으로 처리하기 어려운 문제를 해결함



- 저장 프로그램은 Workbench에서 바로 작성하고 컴파일한 후 결과를 실행함

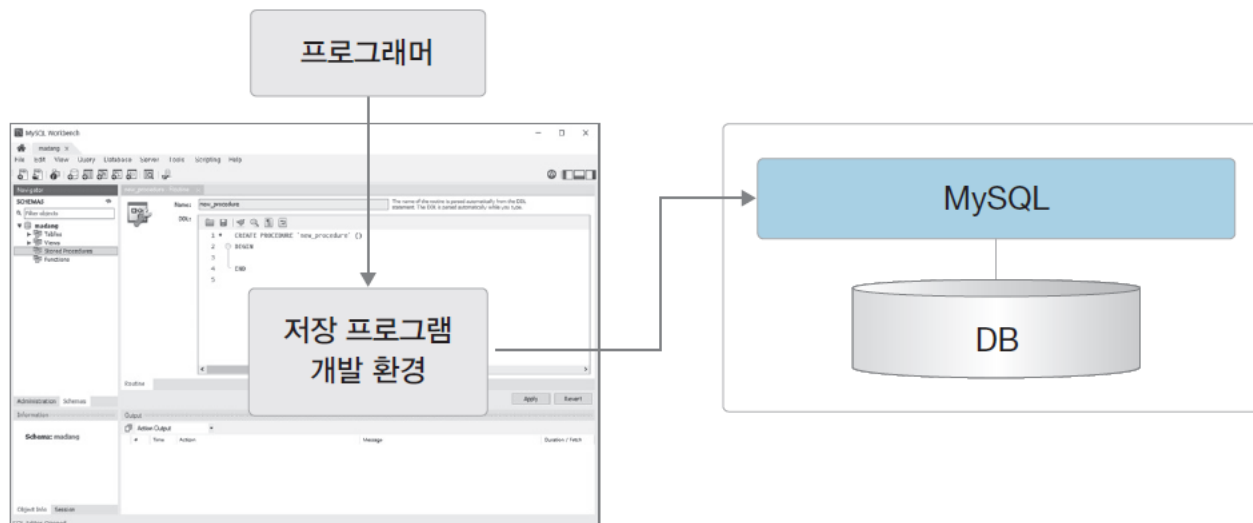


그림 5-3 저장 프로그램 개발 환경

# 1. 저장 프로그램

## ❖ 프로시저

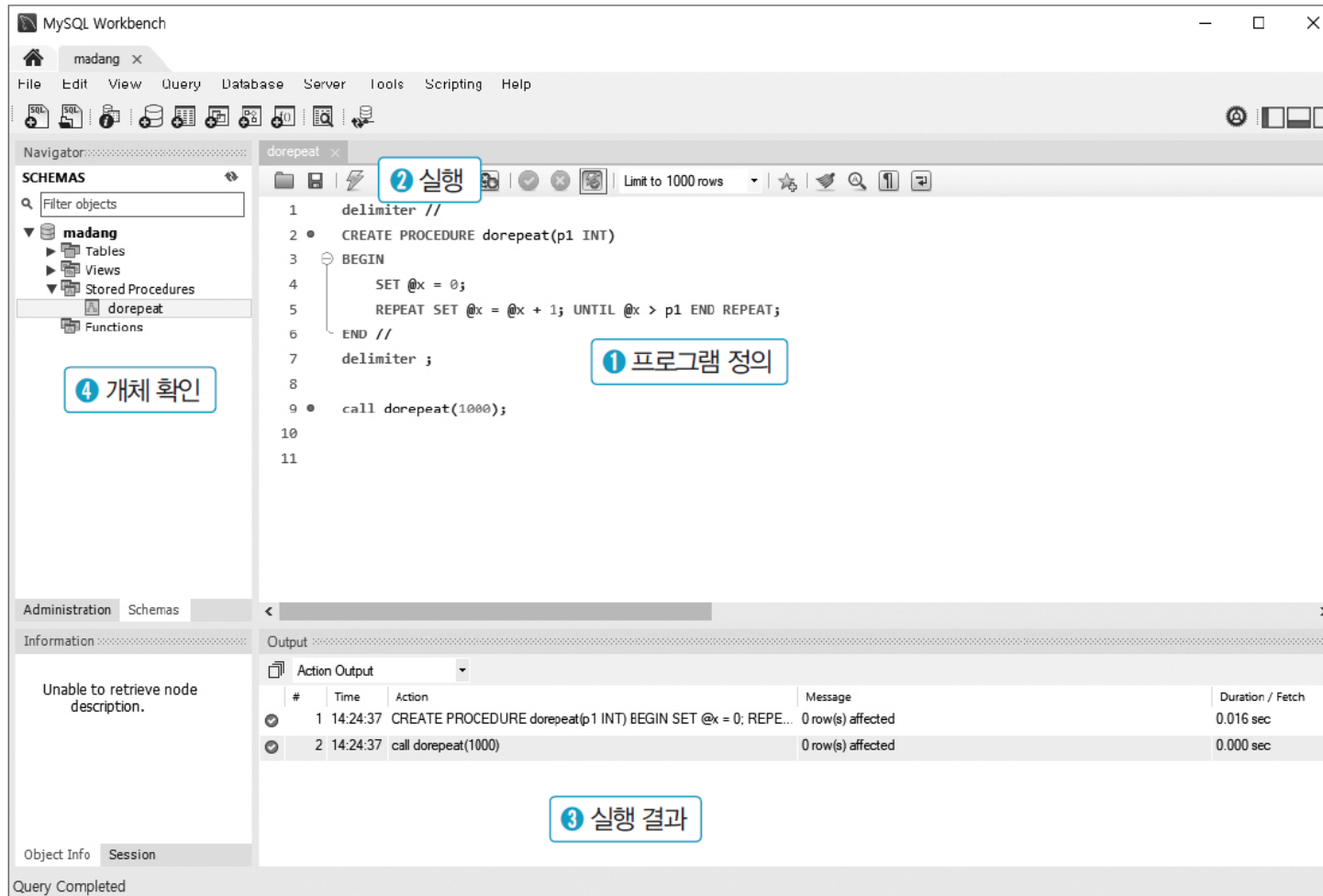
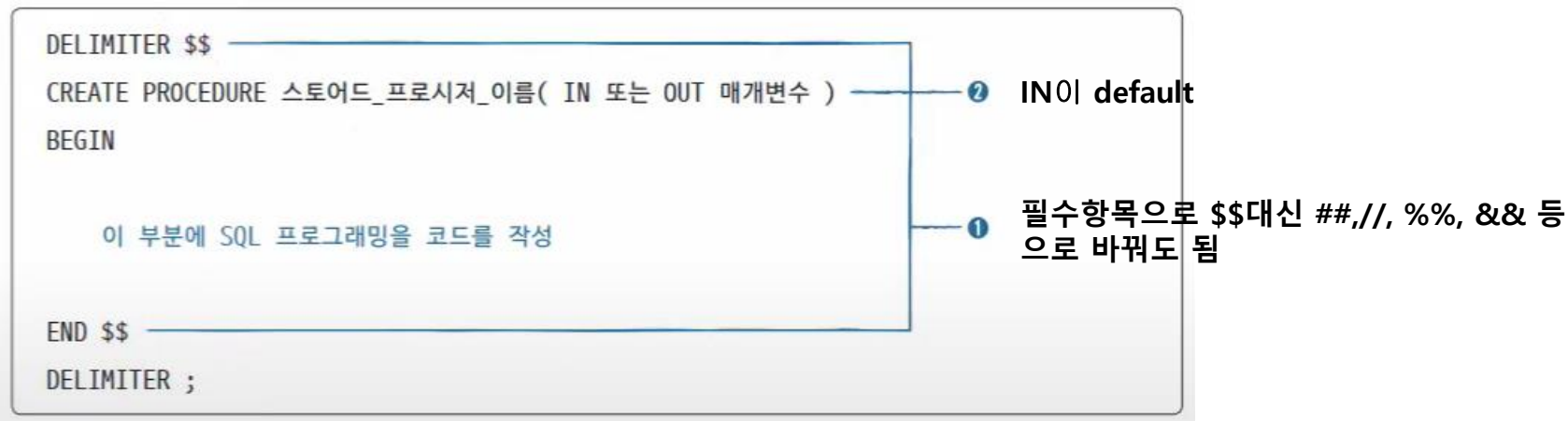


그림 5-4 프로시저를 정의하는 과정

# 1. 저장 프로그램

## ❖ 프로시저

### ■ 프로시저를 정의하려면 CREATE PROCEDURE 문을 사용함



### ■ 정의 방법

- 프로시저(①)는 선언부와 실행부(BEGIN-END)로 구성됨
- 선언부(②)에서는 변수와 매개변수를 선언하고 실행부에서는 프로그램 로직을 구현함
- IN 또는 OUT매개변수(parameter)는 저장 프로시저가 호출될 때 그 프로시저에 전달되는 값
- 변수(variable)는 저장 프로시저나 트리거 내에서 사용되는 값
- 소스코드에 대한 설명문은 /\*와 \*/ 사이에 기술
- 설명문이 한 줄이면 이중 대시(-- ) 기호 다음에 기술해도 됨

# 1. 저장 프로그램

## ■ 특징

- Stored procedure를 만드는 순간에는 존재하지 않는 테이블을 이용하여 프로그램 작성 가능함. ==> 실행 즉 호출하기 전에만 존재하면 됨
- Stored procedure 도 데이터베이스의 개체 중 하나임, 즉 테이블처럼 데이터베이스 내부에 저장됨.
- 일반적으로 SQL문의 구분자는 ; 를 사용하는데 세미콜론이 SQL의 끝인지, Stored procedure의 끝인지 모호해질 수 있어서 \$\$ 가 나올 때까지는 Stored procedure가 끝난 것이 아니라는 것을 표시하는 것임=>구분자 \$\$ 는 \$ 하나로 적어도 되지만 명백히 하기 위해 2개를 사용함.
- 변수에 대한 별도 선언 없이 @변수명으로 사용할 수 있지만, Stored procedure의 내부에서만 사용하는 변수를 [DECLARE 변수명 데이터타입]으로 선언하고 사용해야 한다.



# 1. 저장 프로그램

- **command line client** 에서 **db**를 사용하지 않는 프로시저 정의하고 실행

```
mysql> delimiter //
```

```
mysql> create procedure dorepeat(p1 int)
```

```
-> begin
```

```
-> set @x = 0;
```

```
-> repeat set @x = @x + 1 ; until @x > p1 end repeat;
```

```
-> end
```

```
-> //
```

```
mysql> delimiter ;
```

```
mysql> call dorepeat(1000);
```

```
mysql> select @x;
```

```
+-----+
```

```
| @x   |
```

```
+-----+
```

```
| 1001 |
```

```
+-----+
```

# 참고

- 동적 sql문을 stored procedure로 만들수 있다.

```
mysql> delimiter $$  
mysql> create procedure dynamic_proc(  
    -> in tablename varchar(20)  
    -> )  
    -> begin  
    -> set @sqlqry = concat('select * from ',tablename);  
    -> prepare myqry from @sqlqry;  
    -> execute myqry;  
    -> deallocate prepare myqry;  
    -> end $$
```

Query OK, 0 rows affected (0.01 sec)

```
mysql> delimiter ;
```

## 참고

- 동적 sql문을 stored procedure로 만들수 있다.

```
mysql> call dynamic_proc('customer');
```

custid	name	address	phone
1	박지성	영국 맨체스타	000-5000-0001
2	김연아	대한민국 서울	000-6000-0001
3	장미란	대한민국 강원도	000-7000-0001
4	추신수	미국 클리블랜드	000-0000
5	박세리	대한민국 대전	NULL

```
5 rows in set (0.02 sec)
```

```
Query OK, 0 rows affected (0.05 sec)
```

# 1. 저장 프로그램

## ❖ 삽입 작업을 하는 프로시저

- 프로시저로 데이터 삽입 작업을 하면 좀 더 복잡한 조건의 삽입 작업을 인자 값만 바꾸어 수행할 수도 있고, 저장해두었다가 필요할 때마다 호출(call)하여 사용할 수도 있음

예제 5-1 Book 테이블에 한 개의 튜플을 삽입하는 프로시저

InsertBook.sql

```
01 use madang;
02 delimiter //
03 CREATE PROCEDURE InsertBook(
04     IN myBookID INTEGER,
05     IN myBookName VARCHAR(40),
06     IN myPublisher VARCHAR(40),
07     IN myPrice INTEGER)
08 BEGIN
09     INSERT INTO Book(bookid, bookname, publisher, price)
10         VALUES(myBookID, myBookName, myPublisher, myPrice);
11 END;
12 //
13 delimiter ;

A /* 프로시저 InsertBook을 테스트하는 부분 */
B CALL InsertBook(13, '스포츠과학', '마당과학서적', 25000);
C SELECT * FROM Book;
```

# 1. 저장 프로그램

## ❖ 삽입 작업을 하는 프로시저

bookid	bookname	publisher	price
1	축구의 역사	굿스포츠	7000
2	축구아는 여자	나무수	13000
3	축구의 이해	대한미디어	22000
4	골프 바이블	대한미디어	35000
5	피겨 교본	굿스포츠	8000
6	역도 단계별기술	굿스포츠	6000
7	야구의 추억	이상미디어	20000
8	야구를 부탁해	이상미디어	13000
9	올림픽 이야기	삼성당	7500
10	Olympic Champions	Pearson	13000
13	스포츠과학	마당과학...	25000

그림 5-5 InsertBook 프로시저를 실행한 후 Book 테이블

# 1. 저장 프로그램

## ❖ 결과를 반환하는 프로시저

예제 5-3 Book 테이블에 저장된 도서의 평균가격을 반환하는 프로시저

AveragePrice.sql

```
01 delimiter //
02 CREATE PROCEDURE AveragePrice(
03   OUT AverageVal INTEGER)
04 BEGIN
05   SELECT AVG(price) INTO AverageVal
06   FROM Book WHERE price IS NOT NULL;
07 END;
08 //
09 delimiter ;
```

```
A /* 프로시저 AveragePrice를 테스트하는 부분 */
B CALL AveragePrice(@myValue);
C SELECT @myValue;
```

출력 매개변수 - @변수명  
결과 반환후 변수에 담긴 내용 확인

@myValue

15792

그림 5-7 AveragePrice 프로시저를 실행한 결과

# 1. 저장 프로그램

## ❖ 삽입작업과 결과를 반환하는 프로시저

**예제** testbook 테이블에 자료를 추가하고 Auto increment 된 id 값 반환

```
delimiter //
01 CREATE PROCEDURE maxID(
02     IN myBookName VARCHAR(40),
03     OUT myid INTEGER
04 )
05 BEGIN
06     INSERT INTO testtbl(bookname)
07     VALUES(myBookName);
08     select max(id) into myid from testtbl;
09 END;
//
delimiter ;
```

```
A /* 프로시저 maxID를 테스트하는 부분 */
B CALL maxID('첫 번째 책', @myId);
C CALL maxID('두 번째 책', @myId);
D CALL maxID('세 번째 책', @myId);
SELECT @myId;
```

출력 매개변수 - @변수명  
결과 반환후 변수에 담긴 내용 확인

테스트 테이블은 id 값과 책이름 컬럼으로 구성

# 1. 저장 프로그램

## ❖ 제어문을 사용하는 프로시저

- 저장 프로그램의 제어문은 어떤 조건에서 어떤 코드가 실행되어야 하는지를 제어하기 위한 문법으로, 절차적 언어의 구성요소를 포함함

표 5-2 저장 프로그램의 제어문

구문	의미	문법
DELIMITER	<ul style="list-style-type: none"><li>구문 종료 기호 설정</li></ul>	DELIMITER {기호}
BEGIN-END	<ul style="list-style-type: none"><li>프로그램 문을 블록화시킴</li><li>중첩 가능</li></ul>	BEGIN {SQL 문} END
IF-ELSE	<ul style="list-style-type: none"><li>조건을 검사 결과에 따라 문장을 선택적으로 수행</li></ul>	IF <조건> THEN {SQL 문} [ELSE {SQL 문}] END IF;
LOOP	<ul style="list-style-type: none"><li>LEAVE 문을 만나기 전까지 LOOP을 반복</li></ul>	[label:] LOOP {SQL 문   LEAVE [label]} END LOOP
WHILE	<ul style="list-style-type: none"><li>조건이 참일 경우 WHILE 문의 블록을 실행</li></ul>	WHILE <조건> DO {SQL 문   BREAK   CONTINUE} END WHILE
REPEAT	<ul style="list-style-type: none"><li>조건이 참일 경우 REPEAT 문의 블록을 실행</li></ul>	[label:] REPEAT {SQL 문   BREAK   CONTINUE} UNTIL <조건> END REPEAT [label:]
RETURN	<ul style="list-style-type: none"><li>프로시저를 종료</li><li>상태값을 반환 가능</li></ul>	RETURN [<식>]



# 1. 저장 프로그램

## 예제 입력받은 회원의 주소에 따라 다른 메시지 출력하는 프로시저

```
01
02
03 use madang
04 delimiter //
05 CREATE PROCEDURE customerAddress(
06     custName VARCHAR(40)
07 )
08 BEGIN
09     DECLARE myaddr VARCHAR(10);
10     SELECT _____ INTO myaddr FROM customer
11     WHERE _____;
12     IF _____ THEN
13         SELECT '대한민국에 거주하시는군요';
14     ELSE
15         SELECT '외국에 거주하시는군요';
16     END IF;
17 END;
18 //
19 delimiter ;
20
21
22
```

- A CALL customerAddress('박지성');
- B CALL customerAddress('김연아');
- C

# 1. 저장 프로그램

예제 5-2 동일한 도서가 있는지 점검한 후 삽입하는 프로시저

**BookInsertOrUpdate.sql**

```
01 use madang
02 delimiter //
03 CREATE PROCEDURE BookInsertOrUpdate(
04     myBookID INTEGER,
05     myBookName VARCHAR(40),
06     myPublisher VARCHAR(40),
07     myPrice INT)
08 BEGIN
09     DECLARE mycount INTEGER;
10     SELECT count(*) INTO mycount FROM Book
11         WHERE bookname LIKE myBookName;
12     IF mycount!=0 THEN
13         SET SQL_SAFE_UPDATES=0; /* DELETE, UPDATE 연산에 필요한 설정 문 */
14         UPDATE Book SET price = myPrice
15             WHERE bookname LIKE myBookName;
16     ELSE
17         INSERT INTO Book(bookid, bookname, publisher, price)
18             VALUES(myBookID, myBookName, myPublisher, myPrice);
19     END IF;
20 END;
21 //
22 delimiter ;
```

```
A -- BookInsertOrUpdate 프로시저를 실행하여 테스트하는 부분
B CALL BookInsertOrUpdate(15, '스포츠 즐거움', '마당과학서적', 25000);
C SELECT * FROM Book; -- 15번 투플 삽입 결과 확인
D -- BookInsertOrUpdate 프로시저를 실행하여 테스트하는 부분
E CALL BookInsertOrUpdate(15, '스포츠 즐거움', '마당과학서적', 20000);
F SELECT * FROM Book; -- 15번 투플 가격 변경 확인
```

# 1. 저장 프로그램

## ❖ 제어문을 사용하는 프로시저

bookid	bookname	publisher	price
1	축구의 역사	굿스포츠	7000
2	축구하는 여자	나무수	13000
3	축구의 이해	대한미디어	22000
4	골프 바이블	대한미디어	35000
5	피겨 교본	굿스포츠	8000
6	역도 단계별기술	굿스포츠	6000
7	야구의 추억	이상미디어	20000
8	야구를 부탁해	이상미디어	13000
9	올림픽 이야기	삼성당	7500
10	Olympic Champions	Pearson	13000
13	스포츠과학	마당과학...	25000
15	스포츠 즐거움	마당과학...	25000

B행 호출 결과



bookid	bookname	publisher	price
1	축구의 역사	굿스포츠	7000
2	축구하는 여자	나무수	13000
3	축구의 이해	대한미디어	22000
4	골프 바이블	대한미디어	35000
5	피겨 교본	굿스포츠	8000
6	역도 단계별기술	굿스포츠	6000
7	야구의 추억	이상미디어	20000
8	야구를 부탁해	이상미디어	13000
9	올림픽 이야기	삼성당	7500
10	Olympic Champions	Pearson	13000
13	스포츠과학	마당과학...	25000
15	스포츠 즐거움	마당과학...	20000

E행 호출 결과

그림 5-6 BookInsertOrUpdate 프로시저를 실행한 후 Book 테이블

# 1. 저장 프로그램 - 커서

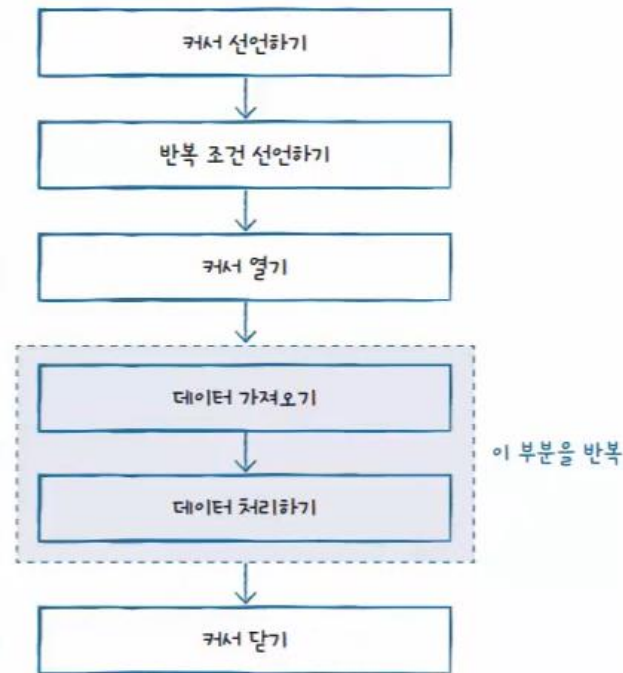
## ❖ 커서를 사용하는 프로시저

- 커서(cursor)는 실행 결과 테이블을 한 번에 한 행씩 처리하기 위하여 테이블의 행을 순서대로 가리키는 데 사용함
- 커서를 생성하여 반복 구간을 설정하여 특정 작업을 행이 끝날때까지 진행하도록 함



행의 시작			
TWC	트와이스	9	서울
BLK	블랙핑크	4	경남
WMN	여자친구	6	경기
OMY	오마이걸	7	서울
GRL	소녀시대	8	서울
ITZ	잇지	5	경남
RED	레드벨벳	4	경북
APN	에이핑크	6	경기
SPC	우주소녀	13	서울
MMU	마마무	4	전남
행의 끝			

# 1. 저장 프로그램 - 커서



키워드	역할
CURSOR <cursor 이름> IS <커서 정의>	커서를 생성 <b>DECLARE &lt;cursor이름&gt; CURSOR FOR</b>
OPEN <cursor 이름>	커서의 사용을 시작
FETCH <cursor 이름> INTO <변수>	행 데이터를 가져옴
CLOSE <cursor 이름>	커서의 사용을 끝냄

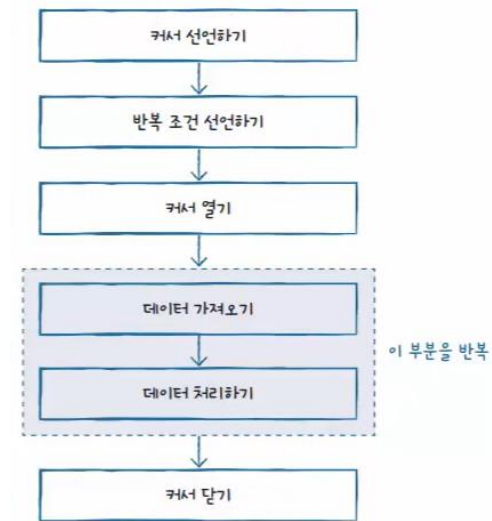
표 5-3 커서와 관련된 키워드

# 1. 저장 프로그램

예제 5-4 Orders 테이블의 판매 도서에 대한 이익을 계산하는 프로시저

Interest.sql

```
01 delimiter //
02 CREATE PROCEDURE Interest()
03 BEGIN
04     DECLARE myInterest INTEGER DEFAULT 0.0;
05     DECLARE Price INTEGER;
06     DECLARE endOfRow BOOLEAN DEFAULT FALSE;
07     DECLARE InterestCursor CURSOR FOR
08         SELECT saleprice FROM Orders;
09     DECLARE CONTINUE handler
10         FOR NOT FOUND SET endOfRow=TRUE;
11     OPEN InterestCursor;
12     cursor_loop: LOOP
13         FETCH InterestCursor INTO Price;
14         IF endOfRow THEN LEAVE cursor_loop;
15         END IF;
16         IF Price >= 30000 THEN
17             SET myInterest = myInterest + Price * 0.1;
18         ELSE
19             SET myInterest = myInterest + Price * 0.05;
20         END IF;
21     END LOOP cursor_loop;
22     CLOSE InterestCursor;
23     SELECT CONCAT(' 전체 이익 금액 = ', myInterest);
24 END;
25 //
26 delimiter ;
```



A /\* Interest 프로시저를 실행하여 판매된 도서에 대한 이익금을 계산 \*/  
B CALL Interest();

# 1. 저장 프로그램

## ❖ 커서를 사용하는 프로시저

CONCAT(' 전체 이익 금액 = ', myInterest)
전체 이익 금액 = 5900

그림 5-8 Interest 프로시저를 실행한 결과

## 2. 트리거

- 트리거(trigger) : 데이터의 변경(INSERT, DELETE, UPDATE) 문이 실행될 때 자동으로 따라서 실행되는 프로시저
- 사용자가 추가 작업을 잊어버리는 실수를 방지해줌 ==> 데이터의 무결성
- 테이블에 입력/수정/삭제되는 정보를 백업하는 용도로 활용 가능

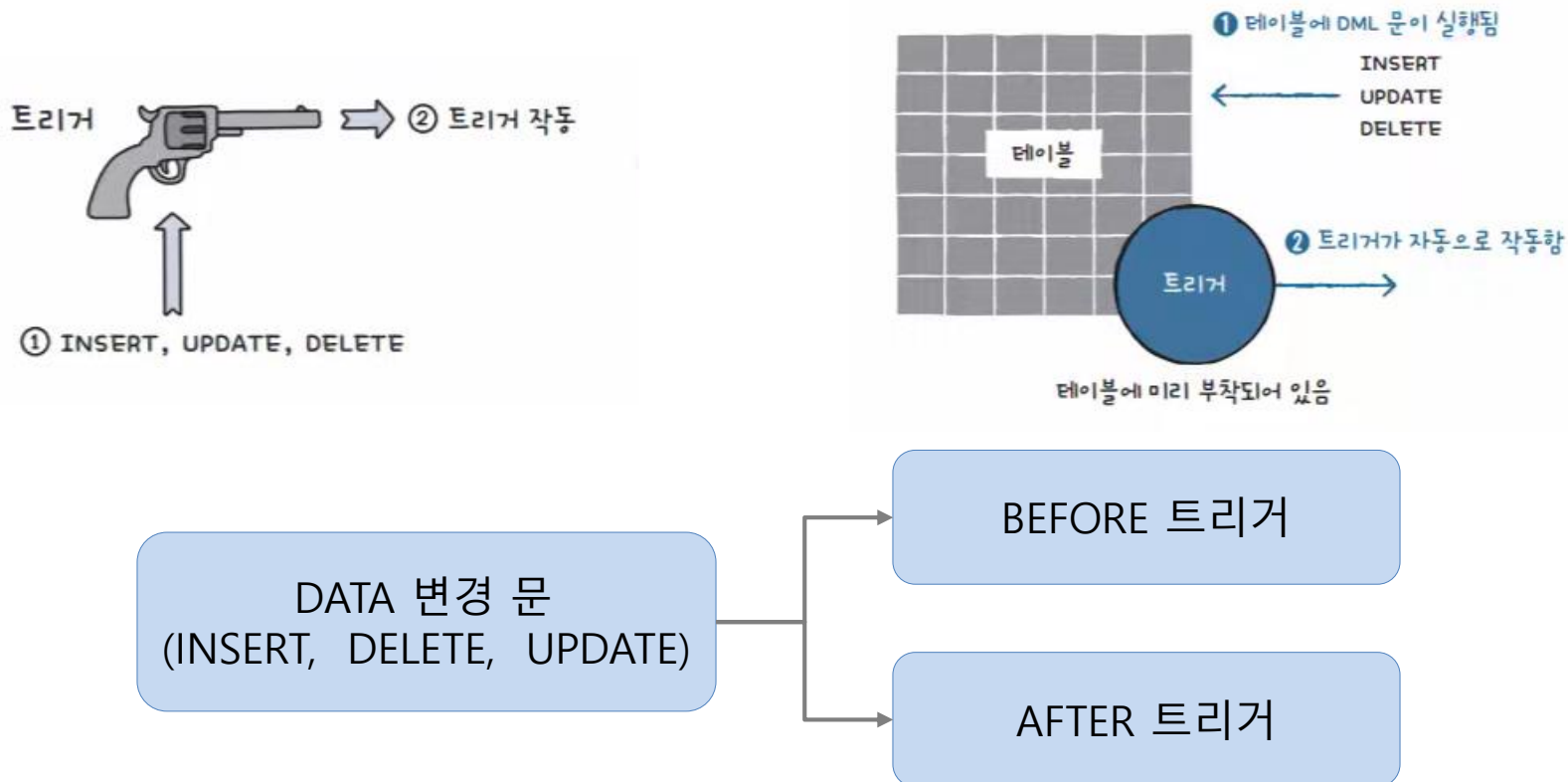


그림 5-9 데이터 변경과 트리거의 수행



## 2. 트리거

### 예제 5-5 새로운 도서를 삽입한 후 자동으로 Book\_log 테이블에 삽입한 내용을 기록하는 트리거

```
SET global log_bin_trust_function_creators=ON; /* 실습을 위해 root 계정에서 실행
```

```
/* madang 계정에서 실습을 위한 Book_log 테이블 생성해준다. */  
CREATE TABLE Book_log(  
    bookid_I INTEGER,  
    bookname_I VARCHAR(40),  
    publisher_I VARCHAR(40),  
    price_I INTEGER);
```

```
01 delimiter //  
02 CREATE TRIGGER AfterInsertBook  
03 AFTER INSERT ON Book FOR EACH ROW  
04 BEGIN  
05 DECLARE average INTEGER;  
06 INSERT INTO Book_log  
07 VALUES(new.bookid, new.bookname, new.publisher, new.price);  
08 END;  
09 //  
10 delimiter ;
```

- A /\* 삽입한 내용을 기록하는 트리거 확인 \*/
- B INSERT INTO Book VALUES(14, '스포츠 과학 1', '이상미디어', 25000);
- C SELECT \* FROM Book WHERE BOOKID=14;
- D SELECT \* FROM Book\_log WHERE BOOKID\_L='14' ; -- 결과 확인

## 2. 트리거

Action	Message
INSERT INTO Book VALUES(14, '스포츠 과학 1', '이상미디어', 25...	1 row(s) affected

Book 테이블 Insert (B 행)



bookid	bookname	publisher	price
14	스포츠 과학 1	이상미디어	25000

Book 테이블 (C 행)



bookid_1	bookname_1	publisher_1	price_1
14	스포츠 과학 1	이상미디어	25000

Book\_log 테이블 (D 행)

그림 5-10 Book 테이블에 튜플을 삽입하여 트리거가 실행된 결과

- \* Delete 에 트리거가 부착되어 있는 경우라도  
truncate table 테이블명 ; 으로 삭제하면 트리거가 발생되지 않고  
데이터가 지워지게 할 수 있다.

### 3. 사용자 정의 함수

- 스토어드 프로시저와 비슷하나 사용방법이나 용도가 조금 다름
- 사용자가 직접 만들어서 사용할 수 있는 함수

```
DELIMITER $$  
CREATE FUNCTION 스토어드_함수_이름(매개변수)  
    RETURNS 반환형식  
BEGIN  
  
    이 부분에 프로그래밍 코딩  
    RETURN 반환값;  
  
END $$  
DELIMITER ;  
SELECT 스토어드_함수_이름();
```

- RETURNS 문으로 반환할 값의 데이터 형식을 지정하고 본문 안에서는 RETURN 문으로 하나의 값을 반환해야 함
- 스토어드 함수의 매개변수는 모두 입력 매개변수이고 IN을 붙이지 않는다.
- 스토어드 함수는 다른 쿼리문의 SELECT 문 안에서 호출됨
- 스토어드 함수 안에서는 SELECT문을 사용할 수 없다.
- 스토어드프로시저는 다양한 용도로 사용가능하지만 스토어드 함수는 어떤 계산을 통해 하나의 값을 반환하는데 주로 사용한다.

### 3. 사용자 정의 함수

```
use madang;
```

```
/*사용하기 전에 형식적으로 실행해야 함 */
```

```
set global log_bin_trust_function_creators = 1;
```

```
drop function if exists sumFunc;
```

```
delimiter $$
```

```
create function sumFunc(number1 int, number2 int)
```

```
    returns int
```

```
begin
```

```
    return number1+number2;
```

```
end $$
```

```
delimiter ;
```

```
select sumFunc(100,200) as "합계";
```

### 3. 사용자 정의 함수

- 사용자 정의 함수는 수학의 함수와 마찬가지로 입력된 값을 가공하여 결과 값을 되돌려줌

예제 5-6 판매된 도서에 대한 이익을 계산하는 함수

**fnc\_Interest.sql**

```
01 delimiter //
02 CREATE FUNCTION fnc_Interest(
03 Price INTEGER) RETURNS INT
04 BEGIN
05 DECLARE myInterest INTEGER;
06 -- 가격이 30,000원 이상이면 10%, 30,000원 미만이면 5%
07 IF Price >= 30000 THEN SET myInterest = Price * 0.1;
08 ELSE SET myInterest := Price * 0.05;
09 END IF;
10 RETURN myInterest;
11 END; //
12 delimiter ;
```

```
A /* Orders 테이블에서 각 주문에 대한 이익을 출력 */
B SELECT custid, orderid, saleprice, fnc_Interest(saleprice) interest
C FROM Orders;
```

### 3. 사용자 정의 함수

custid	orderid	saleprice	interest
1	1	6000	300
1	2	21000	1050
2	3	8000	400
3	4	6000	300
4	5	20000	1000
1	6	12000	600
4	7	13000	650
3	8	12000	600
2	9	7000	350
3	10	13000	650

그림 5-11 Orders 테이블의 건별 이익금 계산

### 3. 사용자 정의 함수

표 5-4 프로시저, 트리거, 사용자 정의 함수의 공통점과 차이점

구분	프로시저	트리거	사용자 정의 함수
공통점	저장 프로시저		
정의 방법	CREATE PROCEDURE 문	CREATE TRIGGER 문	CREATE FUNCTION 문
호출 방법	CALL 문으로 직접 호출	INSERT, DELETE, UPDATE 문이 실행될 때 자동으로 실행됨	SELECT 문에 포함
기능의 차이	SQL 문으로 할 수 없는 복 잡한 로직을 수행	기본값 제공, 데이터 제약 준수, SQL 뷰의 수정, 참조무결성 작업 등을 수행	속성 값을 가공하여 반환, SQL 문에서 직접 사용

## 4. 저장 프로그램 문법 요약

표 5-5 저장 프로그램의 기본 문법 (계속)

구분	명령어
데이터 정의어	CREATE TABLE CREATE PROCEDURE CREATE FUNCTION CREATE TRIGGER DROP
데이터 조작어	SELECT INSERT DELETE UPDATE
데이터 타입	INTEGER, VARCHAR(n), DATE
변수	DECLARE 문으로 선언 치환(SET, = 사용)



## 4. 저장 프로그램 문법 요약

표 5-5 저장 프로그램의 기본 문법 (계속)

구분	명령어
연산자	산술연산자(+, -, *, /) 비교연산자(=, <, >, >=, <=, <>) 문자열연산자(  ) 논리연산자(NOT, AND, OR)
주석	--, /* */
내장 함수	숫자 함수(ABS, CEIL, FLOOR, POWER 등) 집계 함수(AVG, COUNT, MAX, MIN, SUM) 날짜 함수(SYSDATE, DATE, DATNAME 등) 문자 함수(CHAR, LEFT, LOWER, SUBSTR 등)
제어문	BEGIN-END IF-THEN-ELSE WHILE, LOOP
데이터 제어어	GRANT REVOKE

## 6. 다음 프로그램을 프로시저로 작성하고 실행하시오.

- 데이터베이스는 마당서점을 이용한다.

(1) InsertBook() 프로시저를 수정하여 고객을 새로 등록하는 InsertCustomer() 프로시저를 작성하시오.

### 03. 데이터베이스 연동 자바 프로그래밍

1. 소스코드 설명
2. 프로그램 실습



# 데이터베이스 연동 자바 프로그래밍

표 5-6 데이터베이스 연동 자바 프로그래밍 실습 환경

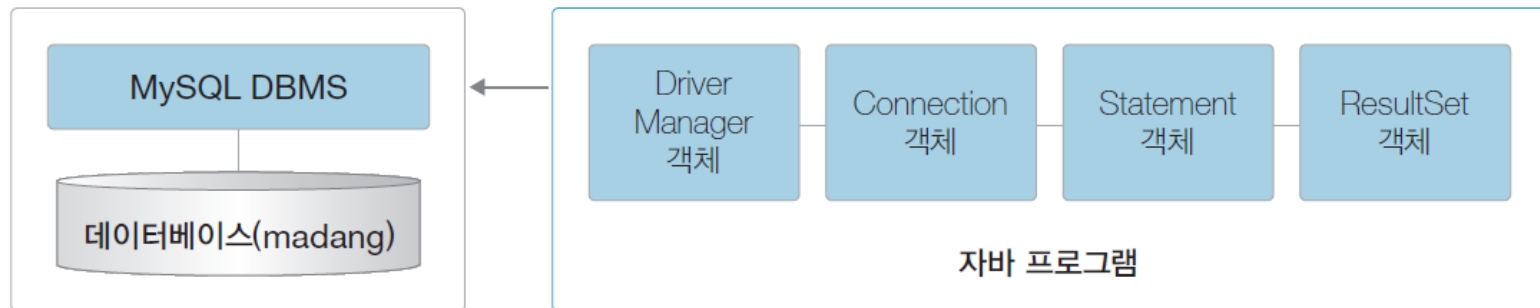
항목	프로그램
데이터베이스 프로그램	MySQL DBMS
자바 컴파일러	JDK 버전 12
데이터베이스와 자바를 연결하는 드라이버	MySQL JDBC 드라이버 Connector/J (파일명:mysql-connector-java-8.x.jar)

# 1. 소스코드 설명

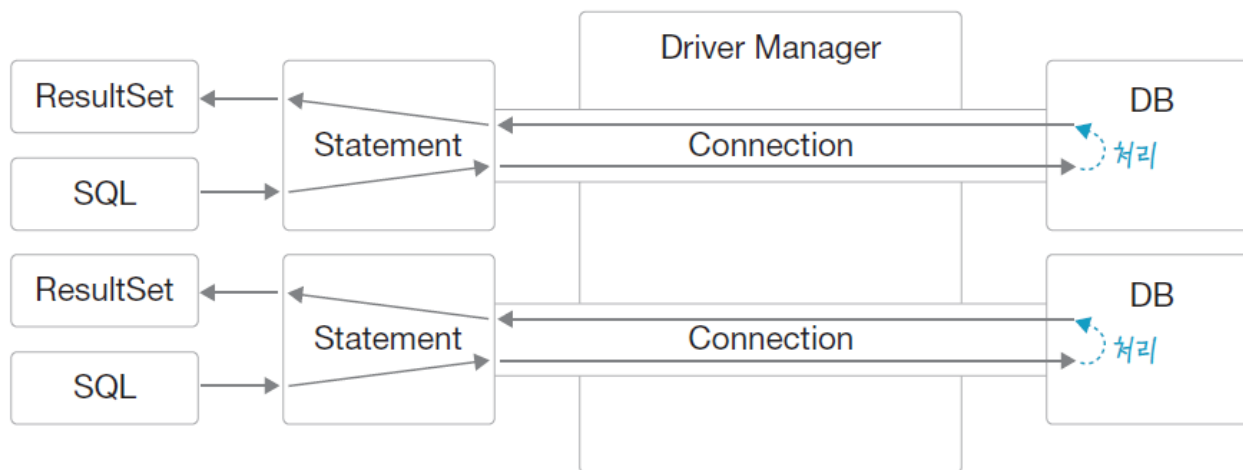
표 5-7 데이터베이스 접속 자바 클래스(java.sql)

클래스 구분	클래스 혹은 인터페이스	주요 메소드 이름	메소드 설명
java.lang	Class	Class.forName(<클래스이름>)	<클래스이름>의 JDBC 드라이버를 로딩
java.sql	DriverManager	Connection getConnection (url, user, password)	데이터베이스 Connection 객체를 생성
	Connection	Statement createStatement()	SQL 문을 실행하는 Statement 객체를 생성
		void close()	Connection 객체 연결을 종료
	Statement	ResultSet executeQuery (String sql)	SQL 문을 실행해서 ResultSet 객체를 생성
		ResultSet executeUpdate (String sql)	INSERT/DELETE/UPDATE 문을 실행 해서 ResultSet 객체를 생성
	ResultSet	boolean first()	결과 테이블에서 커서가 처음 튜플을 가리킴
		boolean next()	결과 테이블에서 커서가 다음 튜플을 가리킴
		int getInt(<int>)	<int>가 가리키는 열 값을 정수로 반환
		String getString(<int>)	<int>가 가리키는 열 값을 문자열로 반환

# 1. 소스코드 설명



(a) 자바 프로그램의 데이터베이스 연동



(b) 객체 간의 호출 순서

그림 5-12 데이터베이스 연결 자바 객체들의 호출 관계

## 2. 프로그램 실습

표 5-8 자바 프로그램 실습 단계

단계		세부 단계	프로그램	참조
[1단계] DBMS 설치 및 환경설정		① MySQL 8.x 설치 ② MySQL 사용자(madang)와 데이터베이스(madang) 생성	MySQL 8.x	부록 A.1~A.3 부록 B.3
[2단계] 데이터베이스 준비		① 마당서점 데이터베이스 준비 (demo_madang.sql)		부록 B.3
[3단계] 자바 실행	명령 프롬프트 이용	① 자바 컴파일러 설치 ② JDBC 드라이버 설치 ③ 자바 프로그램 준비(booklist.java) ④ 컴파일 및 실행	JDK JDBC	부록 C.1~C.3
	이클립스 이용	① 자바와 이클립스 개발도구 설치 ② JDBC 드라이버 설치 ③ 자바 프로그램 준비(booklist.java) ④ 컴파일 및 실행	JDK JDBC Eclipse	부록 C.1~C.4

## 2. 프로그램 실습

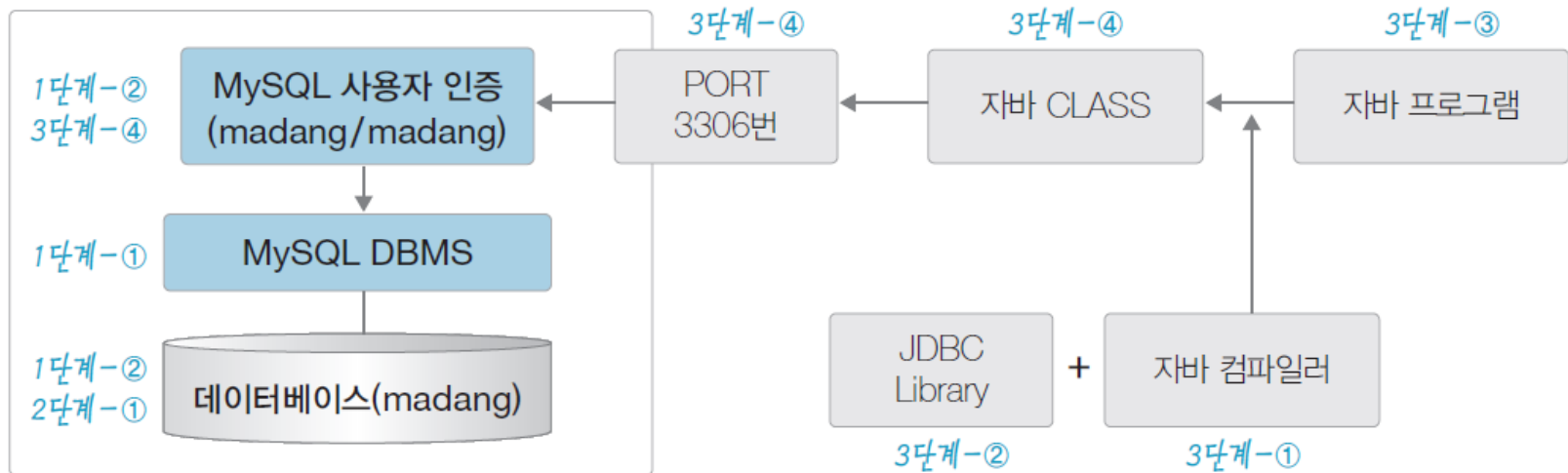


그림 5-14 데이터베이스 연동 자바 프로그램의 실행 흐름도



## 2. 프로그램 실습

### ❖ [1단계] DBMS 설치 및 환경설정

#### ① MySQL 8.x 설치

- MySQL 8.x는 이미 3장에서 설치하였다. 설치는 윈도우 10을 기준으로 한다. 아직 설치 하지 않았다면 부록 A.1~A.3을 참고하여 설치한다.

#### ② SQL 접속을 위한 사용자(madang) 설정

- MySQL에 접속하기 위한 사용자 계정(madang)과 비밀번호(madang)를 부록 B.3을 참고하여 설정한다. 정상적으로 설정되었는지 확인하기 위해 MySQL Workbench를 실행 한 후 madang/madang으로 접속해본다.

## 2. 프로그램 실습

### ❖ [2단계] 데이터베이스 준비

#### ❶ 마당서점 데이터베이스 준비(demo\_madang.sql)

- 마당서점 데이터베이스의 샘플 데이터는 이미 3장에서 설치하였다.  
이 책의 순서대로 실습을 진행하지 않았다면 부록 B.3을 참고하여 설치하면 된다.

## 2. 프로그램 실습

### ❖ [3단계(A)] 자바 실행 – 명령 프롬프트를 이용하는 방법

#### ❶ 자바 컴파일러 설치

- 부록 C.2를 참고하여 설치한다.

#### ❷ JDBC 드라이버 설치

- 부록 C.3을 참고하여 설치한다.

#### ❸ 자바 프로그램 준비(booklist.java)

- booklist.java 프로그램의 소스코드는 앞에서 설명하였다. booklist.java 파일은 메모장에서 작성하거나 예제소스 폴더의 booklist.java를 가져와 사용한다.

#### ❹ 컴파일 및 실행

```
C:\Documents and Settings\Myname>cd c:\W\madang
```

```
C:\madang>javac booklist.java
```

```
C:\madang>java booklist
```

## 2. 프로그램 실습

### ❖ [3단계(B)] 자바실행 – 이클립스를 이용하는 방법

#### ❶ 이클립스 개발도구 설치

- 부록 C.4를 참고하여 설치

#### ❷ JDBC 드라이버 설치

- 부록 C.3을 참고하여 설치

\* 이클립스에서 JDBC 드라이버는 프로젝트->Properties->Libraries->Add External Jars 로 간편설치

#### ❸ 자바 프로그램 준비(booklist.java)

#### ❹ 컴파일 및 실행

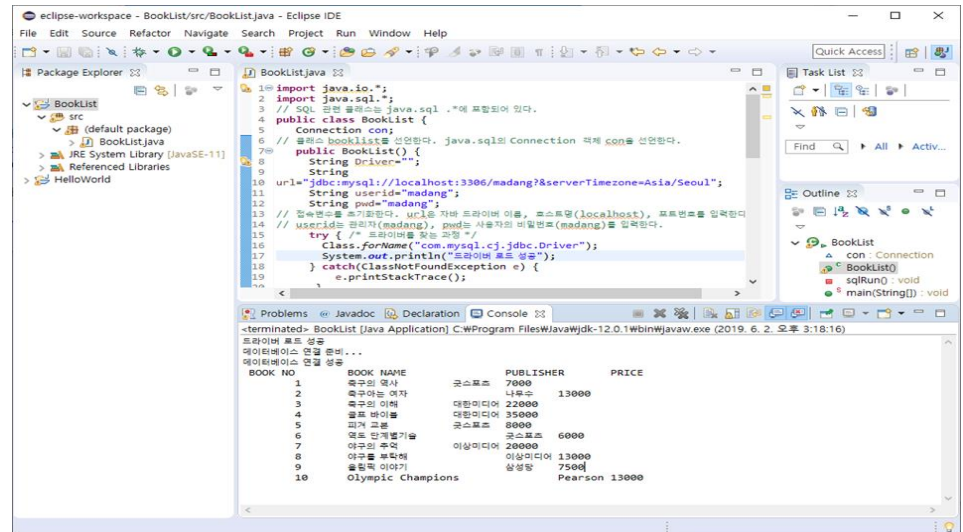


그림 5-19 이클립스에서  
booklist.java 실행 결과 화면

## 04. 데이터베이스 연동 웹 프로그래밍

1. 소스코드 설명
2. 프로그램 실습



# 데이터베이스 연동 웹 프로그래밍

표 5-9 데이터베이스 연동 웹 프로그래밍 실습 환경

항목	프로그램
데이터베이스 프로그램	MySQL 8.x
PHP	PHP 5.x
웹서버	Apache

# 1. 소스코드 설명

- PHP 프로그램은 HTML 태그에 PHP 스크립트를 끼워 넣어 작성하는데, PHP 스크립트 부분은 `<?PHP ... ?>`에 넣어서 실행시킴.

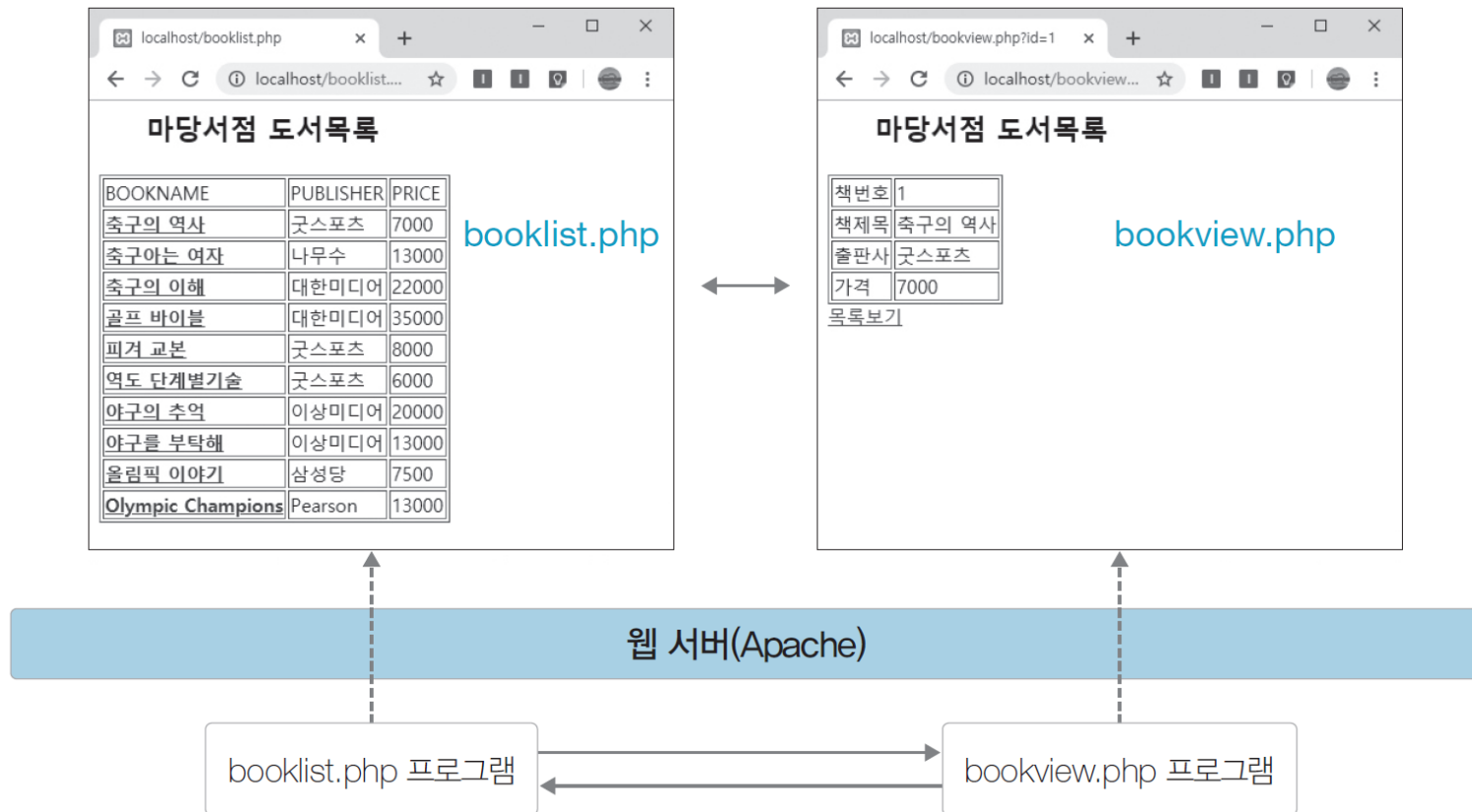


그림 5-20 booklist.php와 bookview.php의 호출 관계와 웹에서 실행된 화면

## 2. 프로그램 실습

표 5-10 JSP 프로그램 실습 단계

단계	세부 단계	프로그램	참조
[1단계] DBMS 설치 및 환경설정	① MySQL 8.x 설치 ② SQL 접속을 위한 사용자(madang) 및 데이터베이스(madang) 생성	MySQL 8.x	부록 A.1~A.3 부록 B.3
[2단계] 데이터베이스 준비	① 마당서점 데이터베이스 준비 (demo_madang.sql)		부록 B.3
[3단계] PHP 실행	① Apache, PHP 설치 -XAMPP 설치 (앞절에서 설명) ② PHP 프로그램 준비 (booklist.php, bookview.php) ③ 실행	XAMPP booklist.php bookview.php	부록 C.1~C.3 부록 C.5



## 2. 프로그램 실습

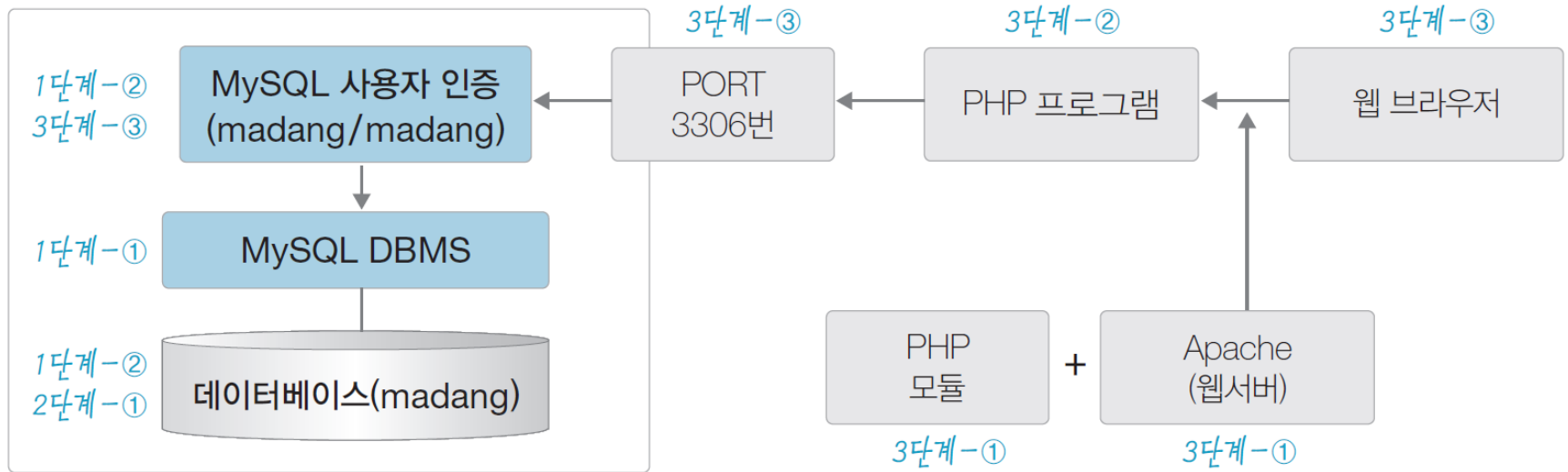


그림 5-21 데이터베이스 연동 PHP 프로그램의 실행 흐름도

## 2. 프로그램 실습

---

- [1단계] DBMS 설치 및 환경설정
- [2단계] 데이터베이스 준비
- [3단계] PHP 실행
  - ① Apache, PHP 설치 – XAMPP 설치
  - ② PHP 프로그램 준비(booklist.php, bookview.php)
  - ③ 실행

## 2. 프로그램 실습

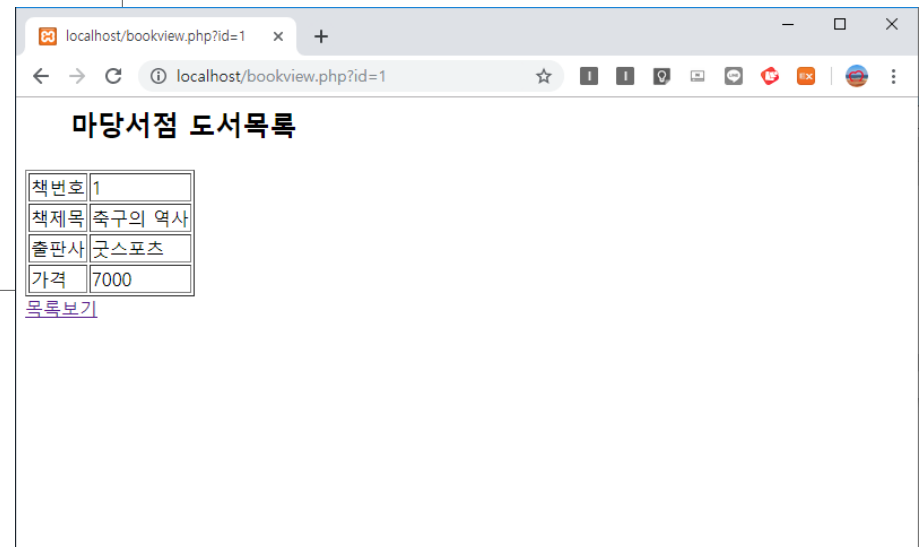
### 4 실행



마당서점 도서목록

BOOKNAME	PUBLISHER	PRICE
<a href="#">축구의 역사</a>	굿스포츠	7000
<a href="#">축구는 여자</a>	나무수	13000
<a href="#">축구의 이해</a>	대한미디어	22000
<a href="#">골프 바이블</a>	대한미디어	35000
<a href="#">피겨 교본</a>	굿스포츠	8000
<a href="#">역도 단계별기술</a>	굿스포츠	6000
<a href="#">야구의 추억</a>	이상미디어	20000
<a href="#">야구를 부탁해</a>	이상미디어	13000
<a href="#">올림픽 이야기</a>	삼성당	7500
<a href="#">Olympic Champions</a>	Pearson	13000

그림 5-23 booklist.php 실행 화면



마당서점 도서목록

책번호	1
책제목	축구의 역사
출판사	굿스포츠
가격	7000

[목록보기](#)

그림 5-24 bookview.php 실행 화면

1. 데이터베이스 프로그래밍
2. 삽입 프로그래밍
3. 저장 프로그램
4. 저장 프로시저
5. 커서
6. 트리거
7. 연동
8. JDBC(Java Database Connectivity)