



## Chapter 05

# 데이터베이스 프로그래밍

MySQL로 배우는 데이터베이스 개론과 실습

# 목차

**01**

데이터베이스 프로그래밍의 개념

**02**

저장 프로그램

**03**

데이터베이스 연동 파이썬 프로그래밍

**04**

데이터베이스 연동 웹 프로그래밍

# 학습목표

- ❖ 데이터베이스 프로그래밍의 개념을 이해한다.
- ❖ 저장 프로그램의 문법과 사용 방법을 알아본다.
- ❖ 파이썬 프로그램과 데이터베이스를 연동하는 방법을 알아본다.
- ❖ 파이썬 웹 프레임워크와 데이터베이스를 연동하는 방법을 알아본다.

## 01 데이터베이스 프로그래밍의 개념



# 데이터베이스 프로그래밍의 개념

## ❖ 프로그래밍

- 프로그램을 설계하고 소스코드를 작성하여 디버깅하는 과정

## ❖ 데이터베이스 프로그래밍

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

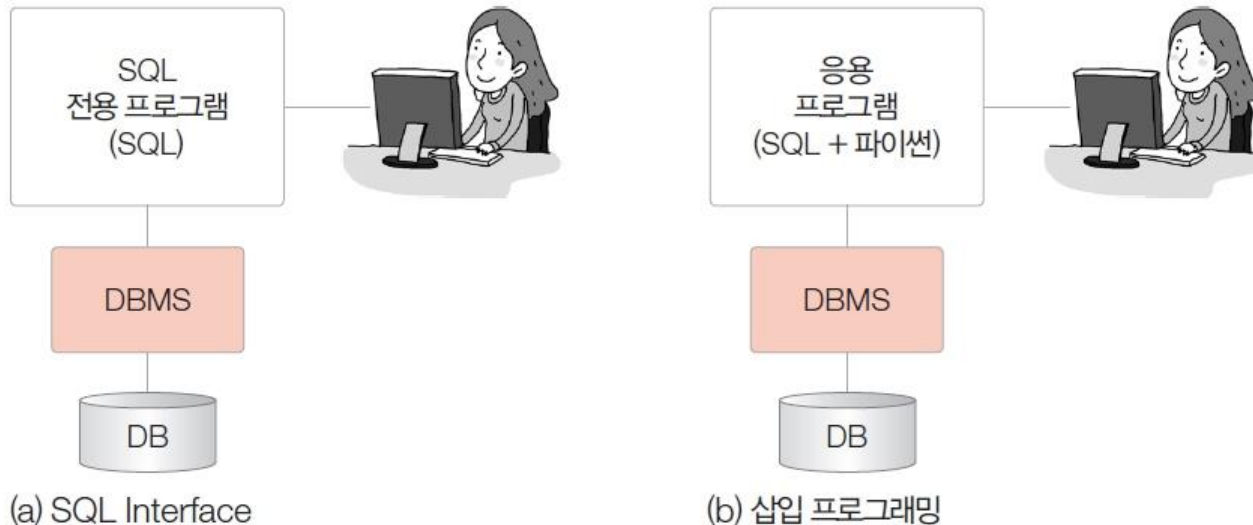


그림 5-1 데이터베이스 프로그래밍

# 데이터베이스 프로그래밍의 개념

## ❖ 데이터베이스 프로그래밍 방법

- SQL 전용 언어를 사용하는 방법
  - SQL 자체의 기능을 확장하여 변수, 제어, 입출력 등의 기능을 추가한 새로운 언어를 사용하는 방법
- 일반 프로그래밍 언어에 SQL을 삽입하여 사용하는 방법
  - 자바, C, C++, 파이썬 등 일반 프로그래밍 언어에 SQL을 삽입하여 사용하는 방법
- 웹 프로그래밍 언어에 SQL을 삽입하여 사용하는 방법
  - JSP, PHP, ASP 등의 웹 스크립트 언어나 Django, Flask 등의 웹 프레임워크에 SQL을 삽입하여 사용하는 방법
- 4GL
  - 4세대 언어의 일종으로 데이터베이스 관리 기능과 비주얼 프로그래밍 기능을 갖춘 'GUI 기반 소프트웨어 개발 도구'를 사용하여 프로그래밍하는 방법

# 데이터베이스 프로그래밍의 개념

- 데이터베이스 응용 시스템을 '하드웨어-운영체제-DBMS-프로그램 환경'으로 계층화하고 층간의 관계를 표현한 것

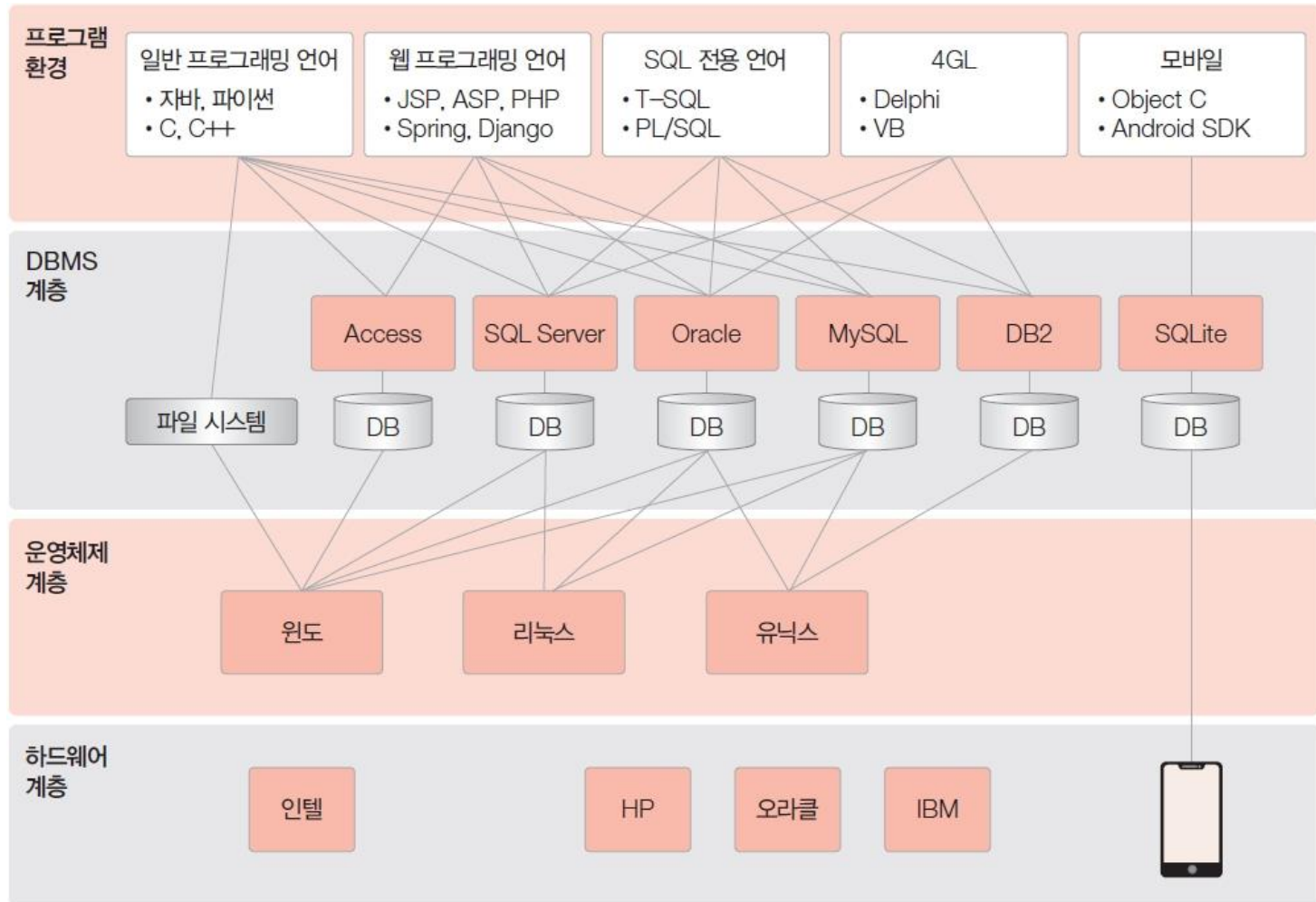


그림 5-2 DBMS 플랫폼과 데이터베이스 프로그래밍의 유형

# 데이터베이스 프로그래밍의 개념

표 5-1 DBMS의 종류와 특징

특징	Access	SQL Server	Oracle	MySQL	DB2	SQLite
제조사	마이크로소프트	마이크로소프트	오라클	오라클	IBM	리처드 힙 (오픈소스)
운영체제 기반	윈도	윈도, 리눅스	윈도, 유닉스, 리눅스	윈도, 유닉스, 리눅스	유닉스	모바일 OS (안드로이드, iOS 등)
특징	개인용 DBMS	윈도 기반 기업용 DBMS	대용량 데이터 베이스를 위한 응용	소용량 데이터 베이스를 위한 응용	대용량 데이터 베이스를 위한 응용	모바일 전용 데이터베이스

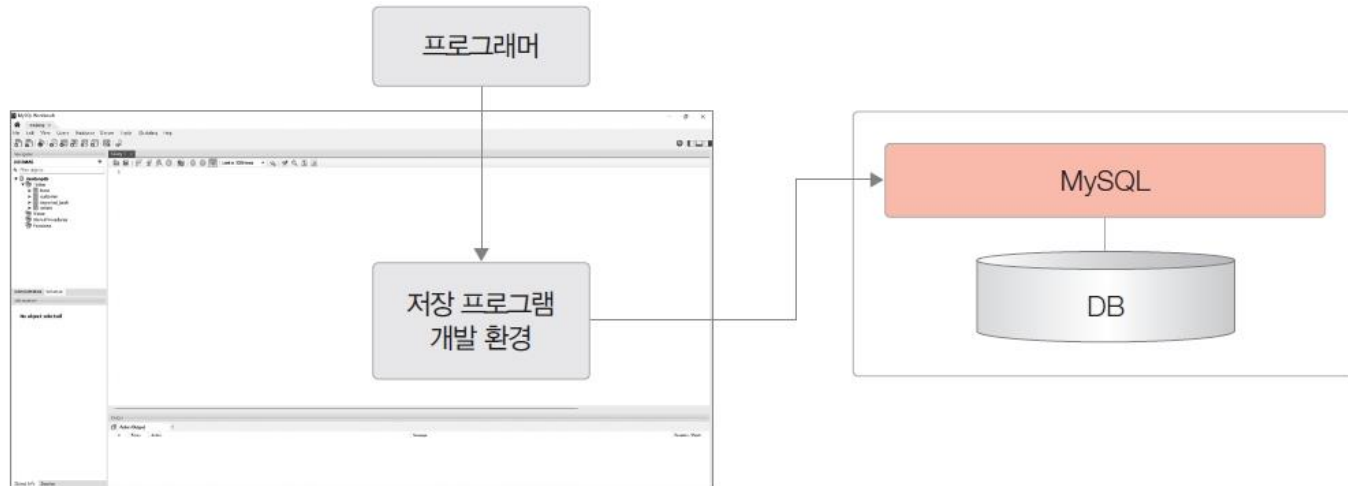


그림 5-3 저장 프로그램(Stored Program) 개발 개념



## 02 저장 프로그램

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



# 1. 저장 프로그램

- 저장 프로그램은 프로그램 로직을 프로시저로 구현하여 객체 형태로 사용
- 프로시저가 정의된 다음 MySQL(DBMS)에 저장되므로 저장 프로그램이라고 함
- MySQL에서 저장 프로그램을 정의하는 과정

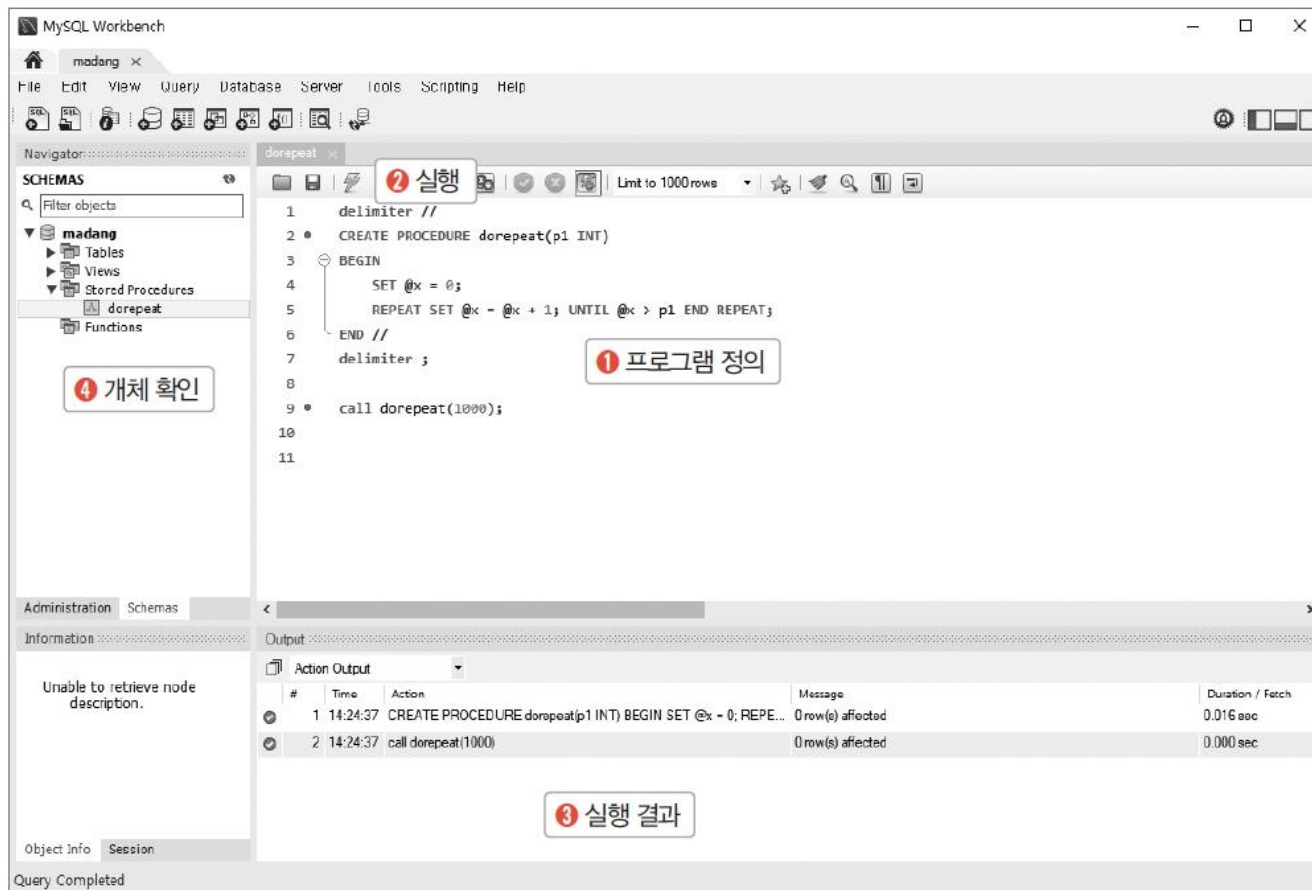


그림 5-4 프로시저를 정의하는 과정

# 1. 저장 프로그램

## ❖ 프로시저를 정의하려면 CREATE PROCEDURE 문을 사용

- 프로시저는 선언부와 실행부(BEGIN-END)로 구성됨
- 선언부에서는 변수와 매개변수를 선언하고 실행부에서는 프로그램 로직을 구현
- 매개변수는 저장 프로시저가 호출될 때 그 프로시저에 전달되는 값
- 변수는 저장 프로시저나 트리거 내에서 사용되는 값
- 소스코드에 대한 설명문은 /\*와 \*/ 사이에 기술한다. 만약 설명문이 한 줄이면 이중 대시(-- ) 기호 다음에 기술해도 됨

# 1. 저장 프로그램

- MySQL 명령라인에서 데이터베이스를 이용하지 않는 프로시저 dorepeat를 정의하고 실행해 본 것

```
mysql> delimiter //
mysql> CREATE PROCEDURE dorepeat(p1 INT)
-> BEGIN
-> SET @x = 0;
-> REPEAT SET @x = @x + 1; UNTIL @x > p1 END REPEAT;
-> END
-> //
Query OK, 0 rows affected (0.00 sec)
mysql> delimiter;
mysql> CALL dorepeat(1000);
Query OK, 0 rows affected (0.00 sec)
mysql> SELECT @x;
+-----+
| @x |
+-----+
| 1001 |
+-----+
1 row in set (0.00 sec)
```

# 1. 저장 프로그램

## ❖ 삽입 작업을 하는 프로시저

예제 5-1 Book 테이블에 한 개의 튜플을 삽입하는 프로시저

파일명: insertbook.sql

```
1 USE madangdb;
2 delimiter //
3 CREATE PROCEDURE InsertBook(
4   IN mybookid      INTEGER,
5   IN mybookname    VARCHAR(40),
6   IN mypublisher   VARCHAR(40),
7   IN myprice       INTEGER)
8 BEGIN
9   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

- C행의 실행 결과 : [그림 5-5]

bookid	bookname	publisher	price
1	축구의 역사	굿스포츠	7000
2	축구 아는 여자	나무수	13000
3	축구의 이해	대한미디어	22000
4	골프 바이블	대한미디어	35000
5	피겨 교본	굿스포츠	8000
6	배구 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	마당과학서적	25000

그림 5-5 InsertBook 프로시저를 실행한 후의 Book 테이블

# 1. 저장 프로그램

## ❖ 제어문을 사용하는 프로시저

표 5-2 저장 프로그램의 제어문

구문	의미	문법
DELIMITER	<ul style="list-style-type: none"><li>구문 종료 기호를 설정함</li></ul>	DELIMITER {기호}
BEGIN-END	<ul style="list-style-type: none"><li>프로그램 문을 블록화함</li><li>중첩 가능</li></ul>	BEGIN {SQL 문} END
IF-ELSE	<ul style="list-style-type: none"><li>조건의 검사 결과에 따라 문장을 선택적으로 수행함</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 반복함</li></ul>	[label:] LOOP {SQL 문   LEAVE [label]} END LOOP
WHILE	<ul style="list-style-type: none"><li>조건이 참일 경우 WHILE 문의 블록을 실행함</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 실행함</li></ul>	[label:] REPEAT {SQL 문   BREAK   CONTINUE} UNTIL <조건> END REPEAT [label:]
RETURN	<ul style="list-style-type: none"><li>프로시저를 종료함</li><li>상태값 반환 가능</li></ul>	RETURN [<식>]

# 1. 저장 프로그램

예제 5-2 동일한 도서가 있는지 점검한 후 삽입하는 프로시저

파일명: bookinsertorupdate.sql

```
1 USE madangdb;
2 delimiter //
3 CREATE PROCEDURE BookInsertOrUpdate(
4     mybookid    INTEGER,
5     mybookname  VARCHAR(40),
6     mypublisher VARCHAR(40),
7     myprice     INT)
8 BEGIN
9     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 설정문 */
14        UPDATE Book SET price = myprice
15            WHERE bookname LIKE mybookname;
```



# 1. 저장 프로그램

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

# 1. 저장 프로그램

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	배구 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



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	배구 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

B행 호출 결과

E행 호출 결과

그림 5-6 BookInsertOrUpdate 프로시저를 실행한 후의 Book 테이블

# 1. 저장 프로그램

## ❖ 결과를 반환하는 프로시저

예제 5-3 Book 테이블에 저장된 도서의 평균 가격을 반환하는 프로시저

파일명: averageprice.sql

```
1 delimiter //
2 CREATE PROCEDURE Averageprice(
3   OUT AverageVal INTEGER)
4 BEGIN
5   SELECT AVG(price) INTO AverageVal
6   FROM Book WHERE price IS NOT NULL;
7 END;
8 //
9 delimiter;
```

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

@myValue
15792

그림 5-7 AveragePrice 프로시저를 실행한 결과

# 1. 저장 프로그램

## ❖ 커서를 사용하는 프로시저

표 5-3 커서와 관련된 키워드

키워드	역할
CURSOR <cursor 이름> IS <커서 정의> DECLARE <cursor 이름> CURSOR FOR	커서를 생성함
OPEN <cursor 이름>	커서 사용을 시작함
FETCH <cursor 이름> INTO <변수>	행 데이터를 가져옴
CLOSE <cursor 이름>	커서의 사용을 끝냄

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

파일명: interest.sql

```
1 delimiter //
2 CREATE PROCEDURE Interest()
3 BEGIN
4     DECLARE myInterest INTEGER DEFAULT 0.0;
5     DECLARE price INTEGER;
6     DECLARE endOfRow BOOLEAN DEFAULT FALSE;
7     DECLARE InterestCursor CURSOR FOR
8         SELECT saleprice FROM Orders;
9     DECLARE CONTINUE handler
10        FOR NOT FOUND SET endOfRow=TRUE;
```

# 1. 저장 프로그램

```
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();

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

전체 이익 금액 = 5900
-----------------

그림 5-8 Interest 프로시저를 실행한 결과

## 2. 트리거

- 데이터의 변경(INSERT, DELETE, UPDATE)문이 실행될 때 자동으로 같이 실행되는 프로시저를 말함
- 보통 트리거는 데이터의 변경문이 처리되는 세 가지 시점, 즉 실행 전(BEFORE), 대신하여(INSTEAD OF), 실행 후(AFTER)에 동작함

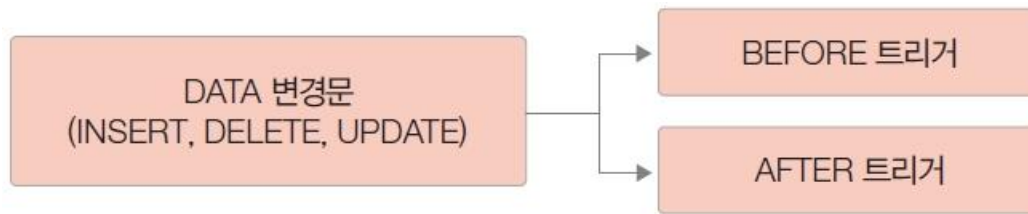


그림 5-9 데이터 변경과 트리거의 수행

- 트리거는 데이터의 변경(삽입, 삭제, 수정)이 일어날 때 부수적으로 필요한 작업인 데이터의 기본값 제공, 데이터 제약 준수, SQL 뷰의 수정, 참조무결성 작업 등을 수행
- 예) Book 테이블에 새로운 도서를 삽입할 때 Book\_log 테이블에도 삽입된 내용을 기록하여 백업한다고 가정
  - Book 테이블에 INSERT 문을 수행하면서 백업을 같이 실행할 수도 있지만 사용자 입장에서는 번거롭기도 하고 보안상 백업을 감추어야 할 경우도 있음
  - 이때 트리거를 사용하면 편함

## 2. 트리거

### ❖ AfterInsertBook 트리거를 작성해보기

- 먼저 root 계정에서 트리거 작동에 필요한 다음 문장을 실행

```
SET global log_bin_trust_function_creators=ON;
```

- 다음으로 madang 계정에서 실습을 위한 Book\_log 테이블을 생성해줌

파일명: booklog.sql

```
CREATE TABLE Book_log(  
    bookid_l INTEGER,  
    bookname_l VARCHAR(40),  
    publisher_l VARCHAR(40),  
    price_l INTEGER);
```



## 2. 트리거

예제 5-5 신규 도서 삽입 후 자동으로 Book\_log 테이블에 삽입 내용을 기록하는 트리거 파일명: afterinsertbook.sql

```
1 delimiter //
2 CREATE TRIGGER AfterInsertBook
3   AFTER INSERT ON Book FOR EACH ROW
4 BEGIN
5   DECLARE average INTEGER;
6   INSERT INTO Book_log
7     VALUES(new.bookid, new.bookname, new.publisher, new.price);
8 END;
9 //
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

- A~D행에서는 AfterInsertBook 트리거를 테스트함
- B행은 새로운 튜플을 Book에 삽입하는 문장
- C행은 Book 테이블에 삽입된 내용을 확인하고, D행은 Book\_log 테이블에 삽입된 내용을 확인함

Action	Message
INSERT INTO Book VALUES(14, '스포츠 과학 1', '이상미디어', 25000	1 row(s) affected

Book 테이블 Insert (B행)



bookid	bookname	publisher	price
14	스포츠 과학 1	이상미디어	25000

Book 테이블 (C행)



bookid	bookname	publisher	price
14	스포츠 과학 1	이상미디어	25000

Book\_log 테이블 (D행)

그림 5-10 Book 테이블에 튜플을 삽입하여 트리거가 실행된 결과

### 3. 사용자 정의 함수

- MySQL에서 작성할 수 있는 사용자 정의 함수는 단일 값을 돌려주는 스칼라 함수가 일반적임
  - 예) 판매된 도서의 이익을 계산하기 위해, 각 주문 건별로 실제 판매가격인 salesprice를 입력받아 가격에 맞는 이익(30,000원 이상이면 10%, 30,000원 미만이면 5%)을 계산하여 반환하는 함수를 작성해보기

예제 5-6 판매된 도서에 대한 이익을 계산하는 함수

파일명: fnc\_interest.sql

```
1 delimiter //
2 CREATE FUNCTION fnc_Interest(
3   price INTEGER) RETURNS INT
4 BEGIN
5   DECLARE myInterest INTEGER;
6   -- 가격이 30,000원 이상이면 10%, 30,000원 미만이면 5%
7   IF price >= 30000 THEN SET myInterest = price * 0.1;
8   ELSE SET myInterest = price * 0.05;
9   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

- B-C행에서는 fnc\_Interest 함수를 테스트함
- 스칼라 함수는 단일 값을 되돌려주므로 SELECT 문에 속성 이름과 같은 위치에 작성
- SELECT 문은 Orders 테이블의 saleprice 속성을 함수의 입력값으로 결과를 출력

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

이때 다음과 같이 ERROR 1418이 발생할 수 있음

Error Code: 1418. This function has none of DETERMINISTIC, NO SQL, or READS SQL DATA in its declaration and binary logging is enabled (you \*might\* want to use the less safe log\_bin\_trust\_function\_creators variable)

아래 문장이 필요함(root 계정)

```
SET global  
log_bin_trust_function_creators=ON;
```

### 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 호출함	INSERT, DELETE, UPDATE 문이 실행될 때 자동으로 실행됨	SELECT 문에 포함됨
기능 차이	SQL 문으로 할 수 없는 복잡한 로직을 수행함	기본값 제공, 데이터 제약 준수, SQL 뷰의 수정, 참조무결성 작업 등을 수행함	속성값을 가공하여 반환, SQL 문에서 직접 사용됨

## 4. 저장 프로그램의 문법 요약

표 5-5 저장 프로그램의 기본 문법

구분	명령어		
데이터 정의어	CREATE TABLE CREATE PROCEDURE CREATE FUNCTION CREATE TRIGGER DROP		
데이터 조작어	SELECT INSERT DELETE UPDATE	연산자	산술 연산자(+, -, *, /) 비교 연산자(=, <, >, >=, <=, <>) 문자열 연산자(  ) 논리 연산자(NOT, AND, OR)
데이터 타입	INTEGER, VARCHAR(n), DATE	주석	--, /* */
변수	DECLARE 문으로 선언 치환(SET, = 사용)	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

## 연습문제 (Q5.1)

**06** 다음 프로그램을 프로시저로 작성하고 실행하시오. 데이터베이스는 마당서점 데이터베이스를 이용한다.

- (1) InsertBook( ) 프로시저를 수정하여 고객을 새로 등록하는 InsertCustomer( ) 프로시저를 작성하시오.
- (2) BookInsertOrUpdate( ) 프로시저를 수정하여 삽입 작업을 수행하는 프로시저를 작성하시오. 삽입하려는 도서와 동일한 도서가 있으면 삽입하려는 도서의 가격이 높을 때만 새로운 값으로 변경한다.

### 03 데이터베이스 연동 파이썬 프로그래밍

1. 소스코드 설명
2. 프로그램 실습



# 데이터베이스 연동 파이썬 프로그래밍 실습 환경

표 5-6 데이터베이스 연동 파이썬 프로그래밍 실습 환경

항목	프로그램
데이터베이스 프로그램	MySQL DBMS
파이썬	Python 3.x
데이터베이스와 파이썬을 연결하는 라이브러리	PyMySQL(pip install pymysql)



# 1. 소스코드 설명

- 다음의 booklist.py는 Book 테이블에 저장된 도서를 읽어와 출력하는 프로그램  
파일명: booklist.py

```
1 import pymysql
2
3 #데이터베이스 접속 관련 변수 초기화
4 host = "localhost"
5 port = 3306
6 database = "madanadb"
7 username = "madang"
8 password = "madang"
9
10 #접속 상태 확인
11 conflag=True
12
13 try:
14     print ('데이터베이스 연결 준비..')
15     conn = pymysql.connect(host=host, user=username, passwd=password, db=database,
16                             port=port, charset='utf8')
17     print ('데이터베이스 연결 성공')
18 except Exception as e:
19     print ('데이터베이스 연결 실패')
20     conflag=False
```

# 1. 소스코드 설명

```
21 #접속이 성공하면 실행
22 if conflag== True :
23     cursor = conn.cursor()          #커서 생성
24
25     sqlstring = 'select * from book;'  #SQL 문장 준비
26
27     res = cursor.execute(sqlstring)    #SQL 문장 실행
28
29     data = cursor.fetchall()          #쿼리 데이터를 가져옴
30
31     #화면에 출력
32     print ('{0}\t{1:<} \t{2:<} \t{3:>}'.format('BOOK NO','BOOK NAME',
        'PUBLISHER','PRICE'))
33     for rowdata in data:
34         print ('{0}\t{1:<} \t{2:<} \t{3:>}'.format(rowdata[0],rowdata[1],
        rowdata[2],rowdata[3]))
35
36     cursor.close()                   #커서를 닫음
37
38     conn.close()                     #데이터베이스 연결을 닫음
```

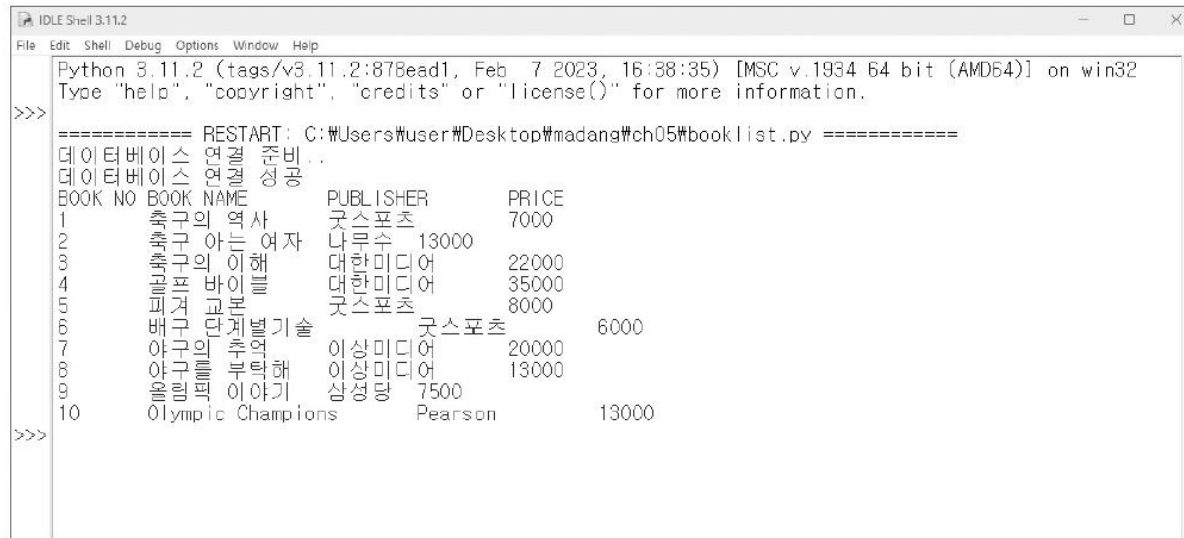
# 1. 소스코드 설명

- 앞에 제시한 소스코드에 대한 설명으로, 파이썬이 PyMySQL을 사용하여 데이터베이스에 접속한 후 데이터를 조회해서 화면에 출력하는 과정

표 5-7 PyMySQL을 사용하여 데이터 가져오기

라인	주요 명령	설명
01	import pymysql	MySQL 데이터베이스에 접속할 수 있게 준비함
03~08	접속 정보 설정	서버의 IP, 데이터베이스 이름, 사용자, 비밀번호, 포트를 설정함
15	pymysql.connect()	DBMS의 접속 정보를 이용해 MySQL에 접속함
23	conn.cursor()	데이터베이스 질의 결과를 순회하고 조작하기 위한 커서를 생성함
25	SQL 문장 준비	Book 테이블을 SELECT하는 SQL 문장을 준비함
27	cursor.execute(sqlstring)	커서를 이용해 SQL 문장을 실행함
29	cursor.fetchall()	커서를 이용해 모든 데이터를 튜플 형태로 가져옴 • fetchone(): 데이터를 한 행씩 튜플로 가져옴 • fetchall(): 결과 집합의 모든 행을 가져옴
32~35	print	MySQL에서 가져온 데이터(튜플)를 화면에 출력함
36	cursor.close()	커서 개체를 닫음
38	conn.close()	MySQL 데이터베이스와의 연결을 닫음

# 1. 소스코드 설명



```
Python 3.11.2 (tags/v3.11.2:878ead1, Feb 7 2023, 16:38:35) [MSC v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\user\Desktop\madang\ch05\booklist.py =====
데이터베이스 연결 준비...
데이터베이스 연결 성공
BOOK NO BOOK NAME PUBLISHER PRICE
1 축구 역사 굿스포츠 7000
2 축구하는 여자 나루수 13000
3 축구의 이해 대한미디어 22000
4 골프 바이블 대한미디어 35000
5 피겨 교본 굿스포츠 8000
6 배구 단계별 기술 굿스포츠 6000
7 야구의 추억 이상미디어 20000
8 야구를 부탁해 이상미디어 13000
9 올림픽 이야기 삼성당 7500
10 Olympic Champions Pearson 13000
>>>
```

그림 5-12 booklist.py의 실행 결과

- 각 단계가 실행되는 개략적인 흐름도는 [그림 5-13]과 같음

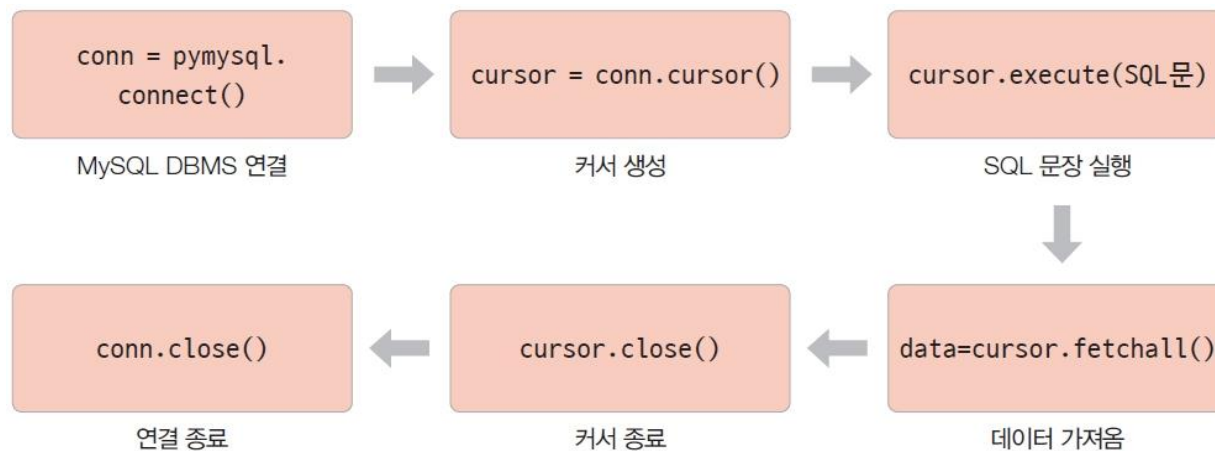


그림 5-13 파이썬 프로그램의 데이터베이스 연동 흐름

# 1. 소스코드 설명

- 파이썬은 인터프리터 언어로 다양한 라이브러리를 지원함
- Python DB 모듈에는 여러 종류가 있는데 여기서는 PyMySQL이라는 모듈을 사용

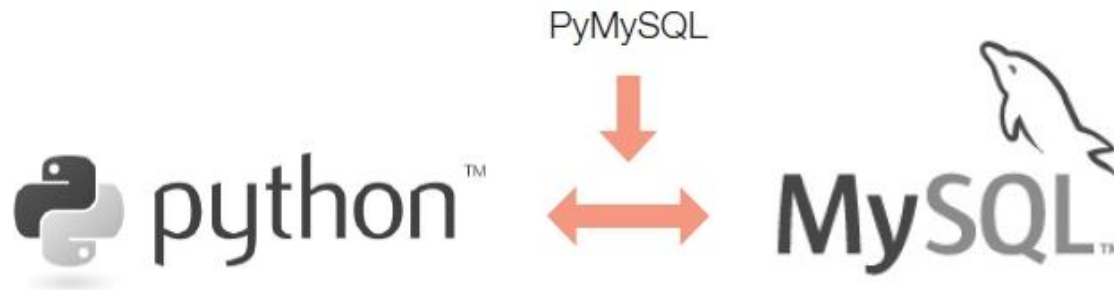


그림 5-14 파이썬의 MySQL 접속

표 5-8 파이썬 MySQL DB api 접속 라이브러리 종류

PyMySQL	순수한 파이썬으로 구현된 MySQL 접속 라이브러리
mysql-connector-python	오라클의 공식 지원 모듈로 MySQL 8.0 이상에서 사용 가능함
Mysqclient	PyMySQL 개발사에서 개발한 C로 구현한 라이브러리

## 2. 프로그램 실습

### ❖ 프로그램 실습 단계

표 5-9 파이썬 프로그램 실습 단계

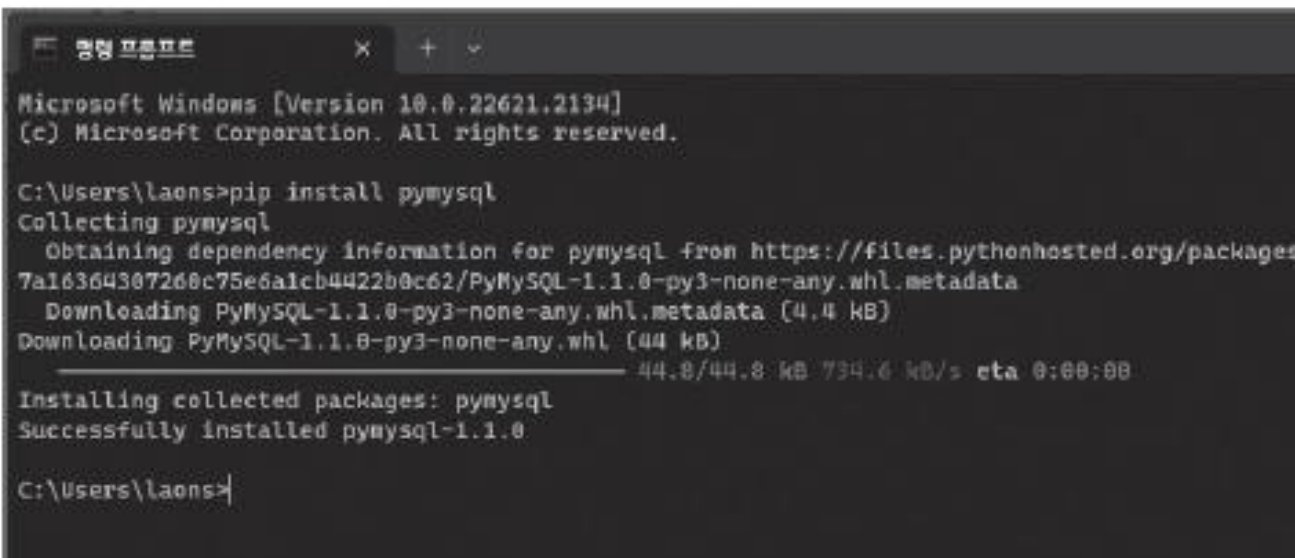
단계	세부 단계	프로그램	참조
[1단계] DBMS 설치 및 환경설정	① MySQL 8.x 설치 ② MySQL 사용자(madang)와 데이터베이스(madangdb) 생성	MySQL 8.x	부록 A.1~A.2 부록 B.3
[2단계] 데이터베이스 준비	① 마당서점 데이터베이스 준비(demo_madang.sql)		부록 B.3
[3단계] 실행	① 파이썬 설치 ② PyMySQL 라이브러리 설치 ③ 파이썬 프로그램 준비(booklist.py) ④ 실행	PyMySQL	부록 C.1~C.2

- [1단계] DBMS 설치 및 환경설정
  - ❶ MySQL 8.x 설치
  - ❷ SQL 접속을 위한 사용자 (madang) 설정

## 2. 프로그램 실습

- [2단계] 데이터베이스 준비
  - ❶ 마당서점 데이터베이스 준비(demo \_ madang.sql)
- [3단계] 실행(명령 프롬프트 이용)
  - ❶ 파이썬 설치
  - ❷ PyMySQL 라이브러리 설치

```
pip install pymysql
```



```
명령 프롬프트
Microsoft Windows [Version 10.0.22621.2134]
(c) Microsoft Corporation. All rights reserved.

C:\Users\laons>pip install pymysql
Collecting pymysql
  Obtaining dependency information for pymysql from https://files.pythonhosted.org/packages/7a/16/36/43/07268c75e6a1cb4422b0c62/PyMySQL-1.1.0-py3-none-any.whl.metadata
  Downloading PyMySQL-1.1.0-py3-none-any.whl.metadata (4.4 kB)
  Downloading PyMySQL-1.1.0-py3-none-any.whl (44 kB)
    44.8/44.8 kB 734.6 kB/s eta 0:00:00
Installing collected packages: pymysql
Successfully installed pymysql-1.1.0

C:\Users\laons>
```

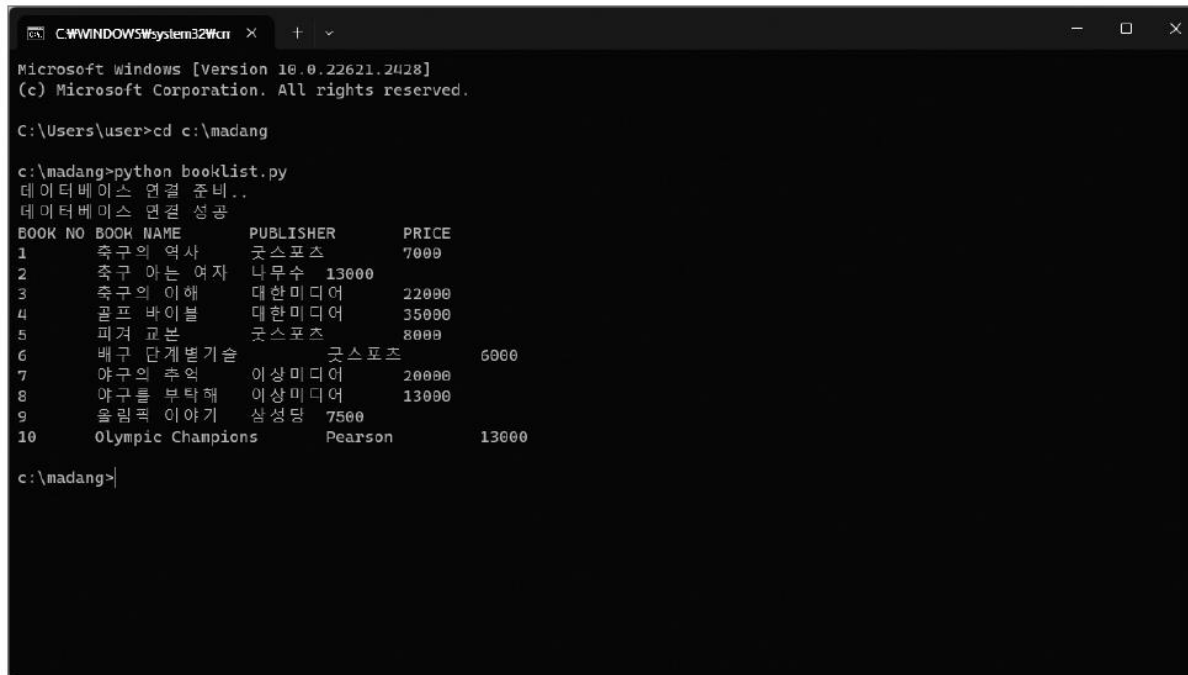
## 2. 프로그램 실습

③ 파이썬 프로그램 booklist.py 준비

④ 실행

```
cd c:\madang
```

```
python booklist.py
```



```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Version 10.0.22621.2428]
(c) Microsoft Corporation. All rights reserved.

C:\Users\user>cd c:\madang

c:\madang>python booklist.py
데이터베이스 연결 준비..
데이터베이스 연결 성공

BOOK NO  BOOK NAME  PUBLISHER  PRICE
1   축구 역사  굿스포츠  7000
2   축구 아는 여자  나무수  13000
3   축구 이해  대한미디어  22000
4   골프 바이블  대한미디어  35000
5   피겨 교본  굿스포츠  8000
6   배구 단계 별 기술  굿스포츠  6000
7   야구의 추억  이상미디어  20000
8   야구를 부탁해  이상미디어  13000
9   올림픽 이야기  삼성당  7500
10  Olympic Champions  Pearson  13000

c:\madang>
```

그림 5-16 명령 프롬프트에서 booklist.py를 실행한 결과 화면



## 04 데이터베이스 연동 웹 프로그래밍

1. 소스코드 설명
2. 프로그램 실습



# 데이터베이스 연동 웹 프로그래밍 실습 환경

표 5-10 데이터베이스 연동 웹 프로그래밍 실습 환경

항목	프로그램
데이터베이스 프로그램	MySQL 8.x
파이썬	Python 3.x
웹 프레임워크	Flask

- 본 실습을 위해 Flask 프로젝트에 필요한 폴더를 c:\madang 내에 다음과 같이 추가 생성함
- 각 파일은 지정된 위치에 메모장이나 IDLE을 통해 작성함

C:\madang\madangweb

C:\madang\madangweb\templates

# 1. 소스코드 설명

파일명: index.py(madangweb 폴더에 위치)

- 다음의 index.py는 Book 테이블에 저장된 도서를 읽어와 웹 브라우저에 템플릿인 booklist.html, bookview.html을 통해 출력하는 Flask 프로그램

```
from flask import Flask, render_template, request
import pymysql

app = Flask(__name__)

db = pymysql.connect(host="localhost", user="madang", passwd="madang",
db="madangdb", charset="utf8")
cur = db.cursor()

@app.route('/')
def index():
    sqlstring = "SELECT * FROM Book"
    cur.execute(sqlstring)

    book_list = cur.fetchall()
    return render_template('booklist.html', book_list=book_list)

@app.route('/view')
def getTicket():
    id=request.args.get('id')
    sqlstring = "SELECT * FROM BOOK where bookid='"+id+"'"
    cur.execute(sqlstring)

    book = cur.fetchall()
    return render_template('bookview.html', book=book)

if __name__ == '__main__':
    app.run('0.0.0.0')
```

# 1. 소스코드 설명

파일명: booklist.html(madangweb\templates 폴더에 위치)

```
<html>
  <head>
    <title>마당서점 도서목록</title>
  </head>
  <body>
    <h2>마당서점 도서목록</h2>
```

```
    <table border=1>
      <thead>
        <tr>
          <th>책이름</th>
          <th>출판사</th>
          <th>가격</th>
        </tr>
      </thead>
      <tbody>
        {% for i in book_list %}
          <tr>
            <td><a href='/view?id={{i[0]}}'> {{ i[1] }} </a> </td>
            <td>{{ i[2] }}</td>
            <td>{{ i[3] }}</td>
          </tr>
        {% endfor %}
      </tbody>
    </table>
  </body>
</html>
```

# 1. 소스코드 설명

파일명: bookview.html(madangweb\templates 폴더에 위치)

```
<html>
  <head>
    <title>도서상세</title>
  </head>
  <body>
    <h2>마당서점 도서상세</h2>
    <table border=1>
      <tr><td>책번호</td><td>{{book[0][0]}} </td> </tr>
      <tr><td>책제목</td><td>{{book[0][1]}}</td></tr>
      <tr><td>출판사</td><td>{{book[0][2]}}</td></tr>
      <tr><td>가격</td><td>{{book[0][3]}}</td></tr>
    </table>
    <a href="/">목록보기</a>
  </body>
</html>
```

# 1. 소스코드 설명

- booklist.html과 bookview.html은 파이썬 Flask 프로그램인 index.py를 통해 호출되며, 이 프로그램들 간의 관계는 [그림 5-17]과 같음

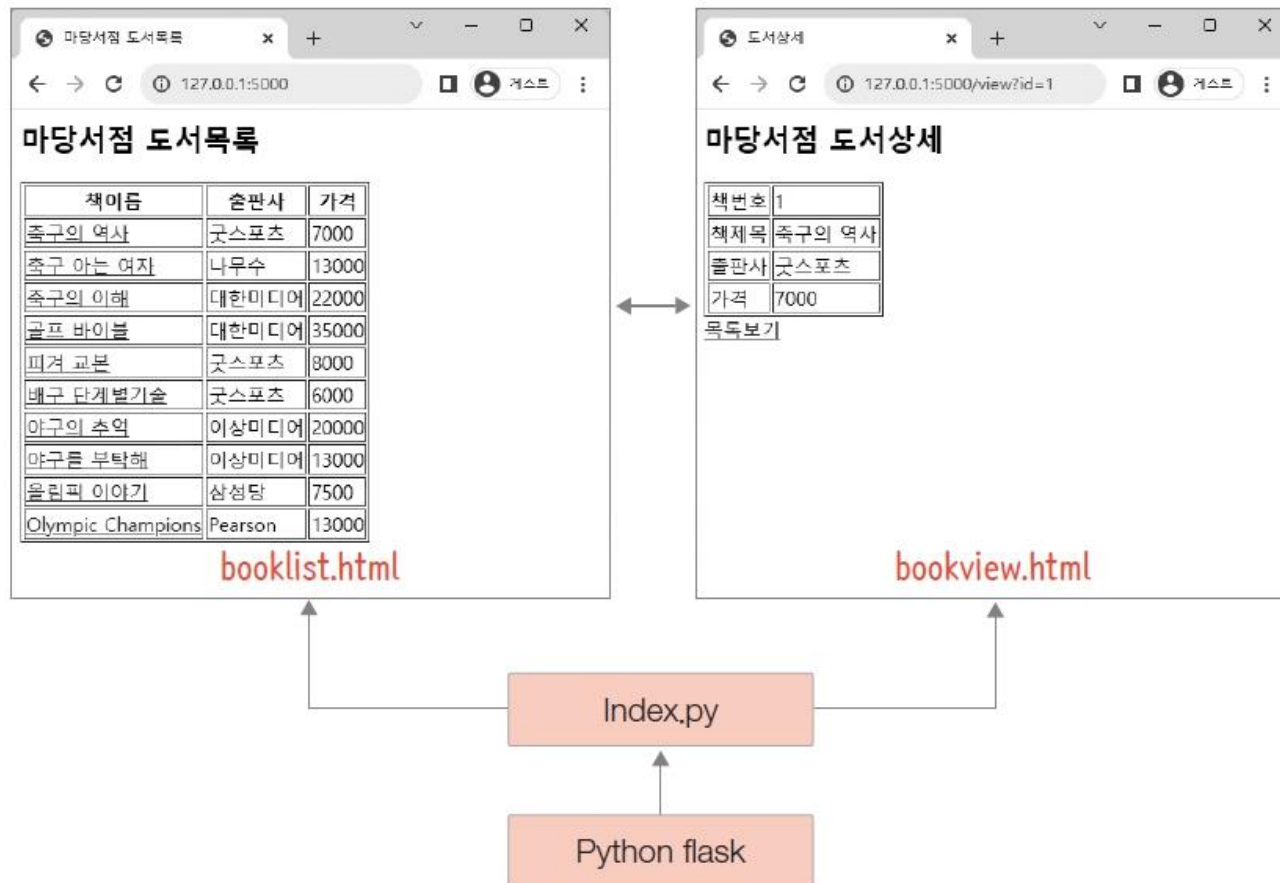


그림 5-17 각 프로그램 간 호출 관계와 브라우저에서 실행된 화면

## 2. 프로그램 실습

- 프로그램의 소스코드를 작성했다면 [표 5-11]과 같이 3단계로 실습할 수 있음

표 5-11 PHP 프로그램 실습 단계

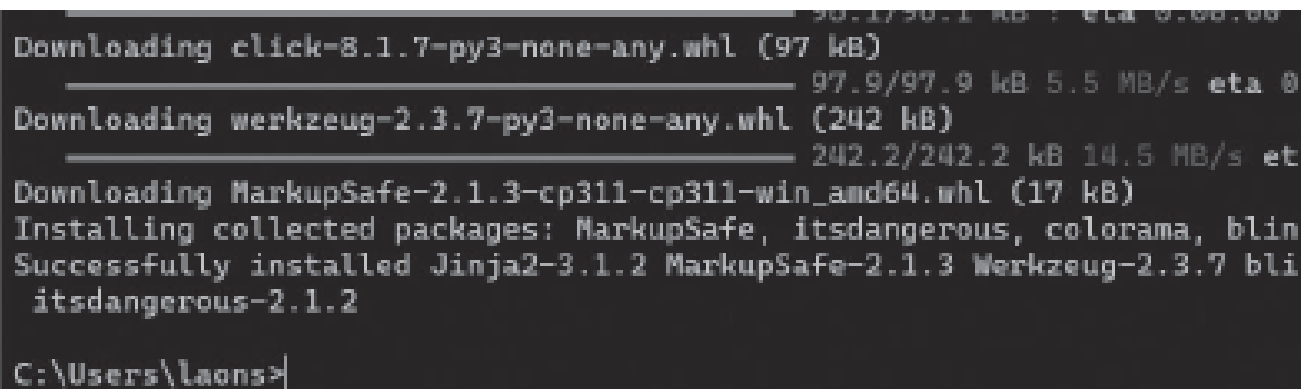
단계	세부 단계	프로그램	참조
[1단계] DBMS 설치 및 환경설정	① MySQL 8.x 설치 ② SQL 접속을 위한 사용자(madang) 및 데이터베이스(madangdb) 생성	MySQL 8.x	부록 A.1~A.2 부록 B.3
[2단계] 데이터베이스 준비	① 마당서점 데이터베이스 준비(demo_madang.sql)		부록 B.3
[3단계] 실행	① Flask 설치 ② 프로그램 준비(index.py, booklist.html, bookview.html) ③ 실행	index.py booklist.html bookview.html	부록 C.4

- [1단계] DBMS 설치 및 환경설정
  - ❶ MySQL 8.x 설치
  - ❷ SQL 접속을 위한 사용자 (madang) 설정

## 2. 프로그램 실습

- [2단계] 데이터베이스 준비
  - ❶ 마당서점 데이터베이스 준비 (demo \_ madang.sql)
- [3단계] 실행
  - ❶ Flask 설치

```
pip install flask
```



```
Downloading click-8.1.7-py3-none-any.whl (97 kB)
 97.9/97.9 kB 5.5 MB/s eta 0
Downloading werkzeug-2.3.7-py3-none-any.whl (242 kB)
 242.2/242.2 kB 14.5 MB/s et
Downloading MarkupSafe-2.1.3-cp311-cp311-win_amd64.whl (17 kB)
Installing collected packages: MarkupSafe, itsdangerous, colorama, blin
Successfully installed Jinja2-3.1.2 MarkupSafe-2.1.3 Werkzeug-2.3.7 bli
itsdangerous-2.1.2

C:\Users\laons>
```

그림 5-18 Flask 설치



## 2. 프로그램 실습

### ❷ 프로그램 준비(index.py, booklist.html, bookview.html)

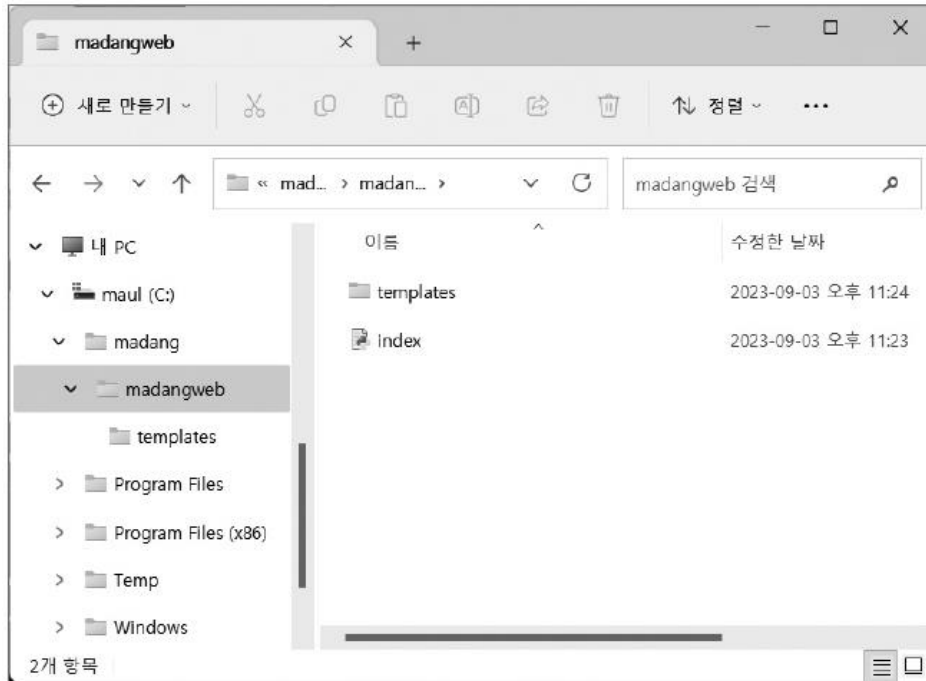


그림 5-19 index.py, booklist.html, bookview.html 파일 저장

### ❸ 실행

```
cd c:\madang\madangweb  
C:\madang\madangweb>python index.py
```

## 2. 프로그램 실습

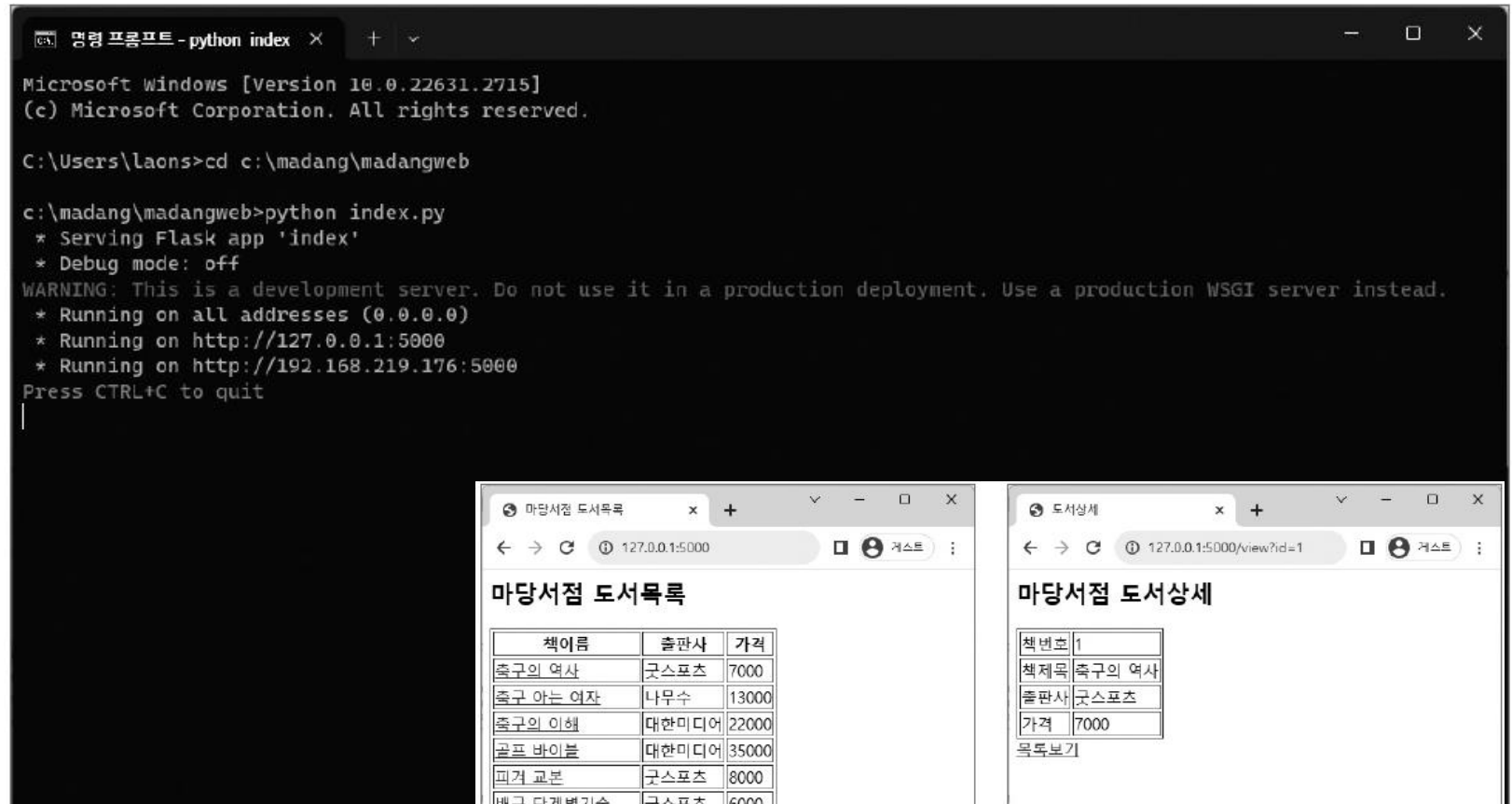


그림 5-20 Flask 웹 서비스 실행

(a) booklist 호출

(b) bookview 호출

그림 5-21 booklist와 bookview가 호출된 브라우저 화면

# 요약

1. 데이터베이스 프로그래밍
2. 삽입 프로그래밍
3. 저장 프로그램
4. 저장 프로시저
5. 커서
6. 트리거
7. 연동
8. PyMySQL