오픈플랫폼활용

Chapter 05. 데이터베이스 프로그래밍

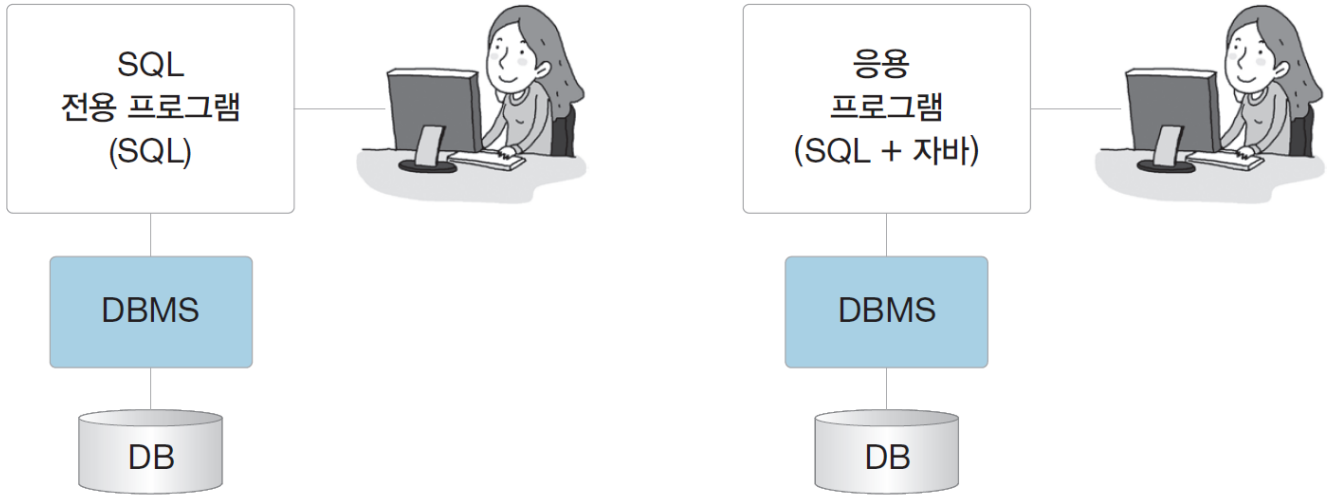
Section 01

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



(a) SQL Interface (b) 삽입 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.
  → 오라클은은 저장 프로그램 언어를 사용하며, SQL Server는 T-SQL이라는 언어를 사용함.

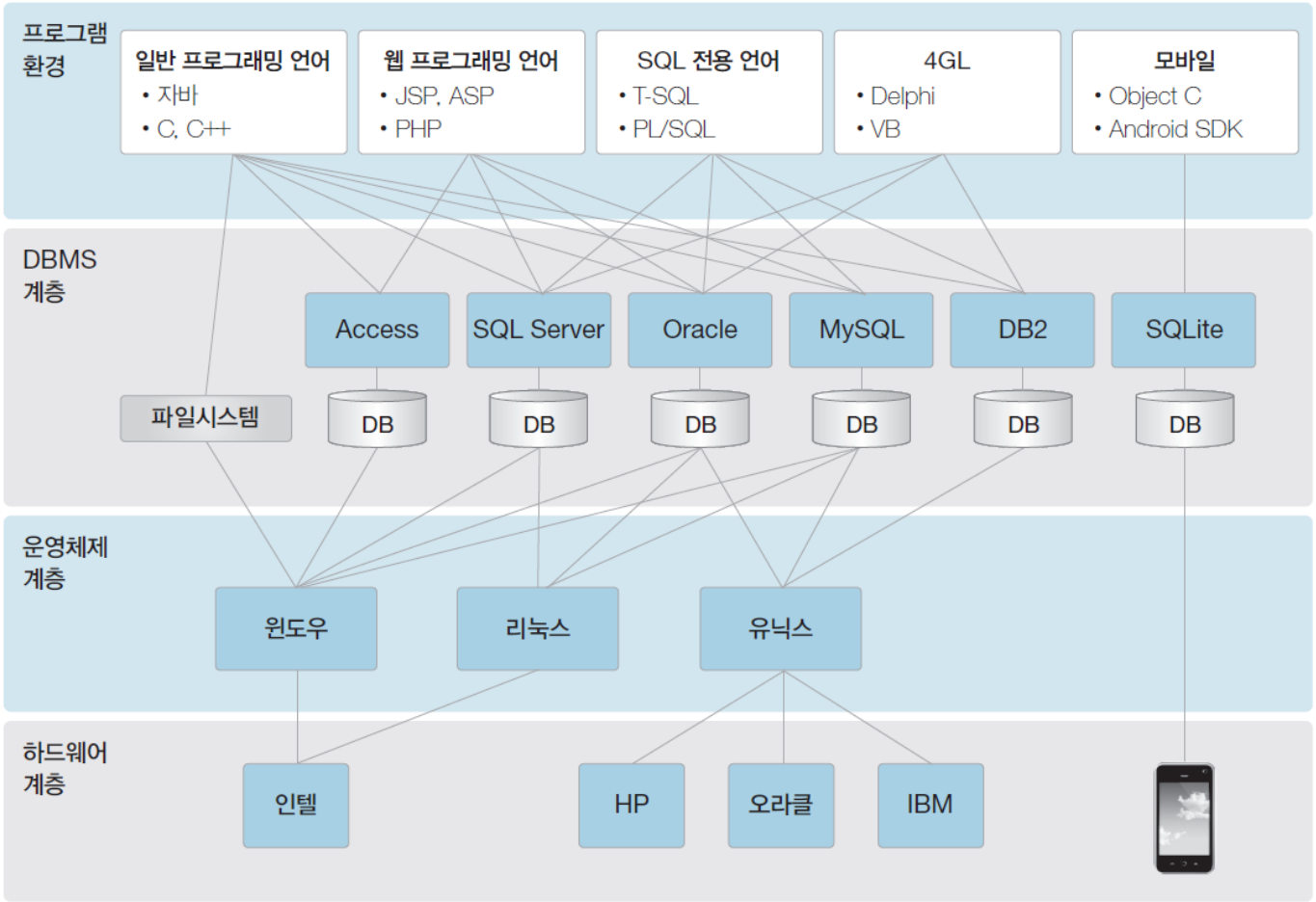
- 일반 프로그래밍 언어에 SQL을 삽입하여 사용하는 방법
  → 자바, C, C++ 등 일반 프로그래밍 언어에 SQL 삽입하여 사용하는 방법.
  → 일반 프로그래밍 언어로 작성된 응용 프로그램에서 데이터베이스에 저장된 데이터를 관리, 검색함.
  → 삽입된 SQL문은 DBMS의 컴파일러가 처리함.

- 웹 프로그래밍 언어에 SQL을 삽입하여 사용하는 방법
  → 호스트 언어가 JSP, ASP, PHP 등 웹 스크립트 언어인 경우다.

- 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

■ DBMS 플랫폼과 데이터베이스 프로그래밍의 유형



데이터베이스 프로그래밍의 개념

■ DBMS 종류와 특징

특징	Access	SQL Server	Oracle	MySQL	DB2	SQLite
제조사	마이크로 소프트사	마이크로 소프트사	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 데이터베이스를 위한 응용	소용량 데이터베이스를 위한 응용	대용량 데이터베이스를 위한 응용	모바일 전용 데이터베이스

Section 02

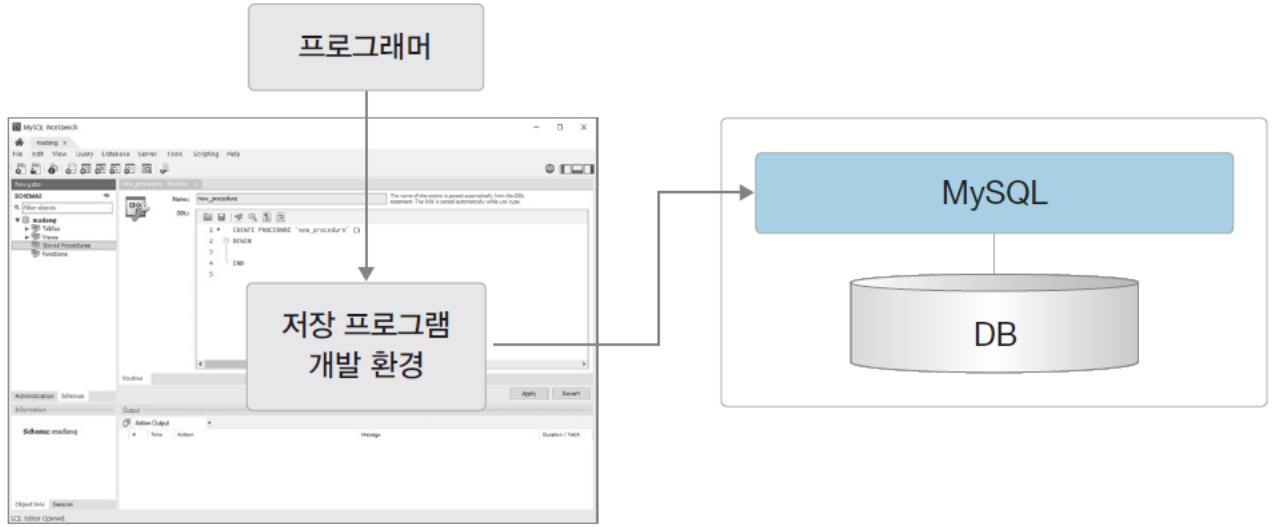
저장 프로그램

저장 프로그램

■ 저장 프로그램(Stored Program) : 데이터베이스 응용 프로그램을 작성하는 데 사용하는 MySQL의 SQL 전용 언어

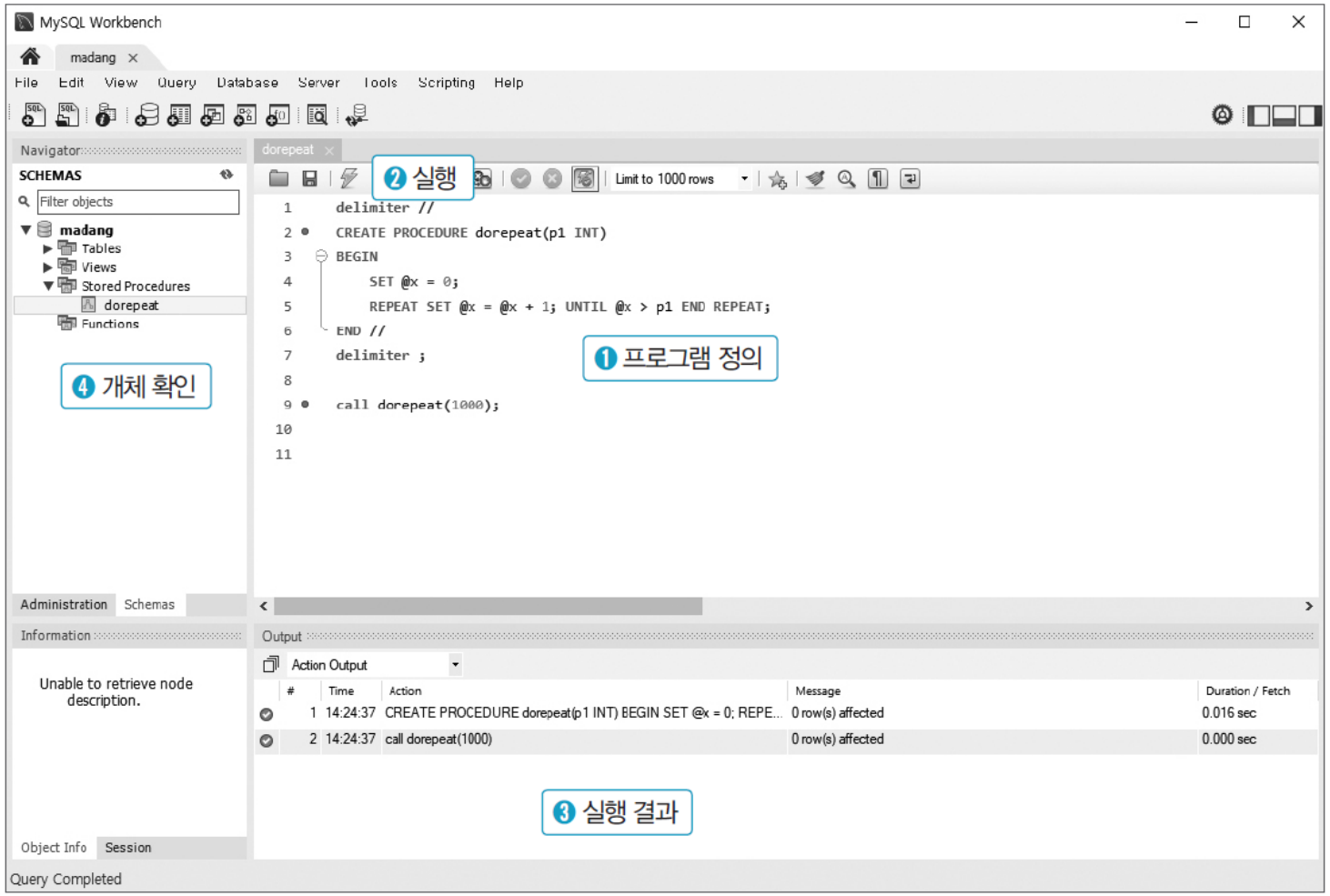
■ SQL 문에 변수, 제어, 입출력 등의 프로그래밍 기능을 추가하여 SQL 만으로 처리하기 어려운 문제를 해결함

■ 저장 프로그램은 Workbench에서 바로 작성하고 컴파일한 후 결과를 실행함



저장 프로그램

■ 프로시저


MySQL Workbench
madang
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
Filter objects
madang
Tables
Views
Stored Procedures
dorepeat
Functions
4 개체 확인
Administration Schemas
Information
Unable to retrieve node description.
Object Info Session
dorepeat
2 실행
Limit to 1000 rows
1 delimiter //
2 CREATE PROCEDURE dorepeat(p1 INT)
3 BEGIN
4 SET @x = 0;
5 REPEAT SET @x = @x + 1; UNTIL @x > p1 END REPEAT;
6 END //
7 delimiter ;
8
9 call dorepeat(1000);
10
11
1 프로그램 정의
Output
Action Output
Time Action Message Duration / Fetch
1 14:24:37 CREATE PROCEDURE dorepeat(p1 INT) BEGIN SET @x = 0; REPE... 0 row(s) affected 0.016 sec
2 14:24:37 call dorepeat(1000) 0 row(s) affected 0.000 sec
3 실행 결과
Query Completed

저장 프로그램

■ 프로시저를 정의하려면 CREATE PROCEDURE 문을 사용함

■ 정의 방법

- 프로시저는 선언부와 실행부(BEGIN-END)로 구성됨
- 선언부에서는 변수와 매개변수를 선언하고 실행부에서는 프로그램 로직을 구현함
- 매개변수(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

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 호출하여 사용할 수도 있음

예제 5-1 Book 테이블에 한 개의 투플을 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;
```

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

저장 프로그램

■ 제어문을 사용하는 프로시저

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

구문	의미	문법
DELIMITER	• 구문 종료 기호 설정	DELIMITER {기호}
BEGIN-END	• 프로그램 문을 블록화시킴 • 중첩 가능	BEGIN {SQL 문} END
IF-ELSE	• 조건의 검사 결과에 따라 문장을 선택적으로 수행	IF <조건> THEN {SQL 문} [ELSE {SQL 문}] END IF;
LOOP	• LEAVE 문을 만나기 전까지 LOOP을 반복	[label:] LOOP {SQL 문 | LEAVE [label] } END LOOP

WHILE	• 조건이 참일 경우 WHILE 문의 블록을 실행	WHILE <조건> DO
		{SQL 문 | BREAK | CONTINUE}
		END WHILE

REPEAT	• 조건이 참일 경우 REPEAT 문의 블록을 실행	[label:] REPEAT {SQL 문 | BREAK | CONTINUE} UNTIL <조건> END REPEAT [label:]
RETURN	• 프로시저를 종료 • 상태값을 반환 가능	RETURN [<식>]

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
```

저장 프로그램

■ 제어문을 사용하는 프로시저

- BookInsertOrUpdate 프로시저를 실행한 후 Book 테이블

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
15	스포츠 즐거움	마당과학...	25000

B행 호출 결과

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
15	스포츠 즐거움	마당과학...	20000

E행 호출 결과

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;
```

@myValue
15792

저장 프로그램

■ 커서를 사용하는 프로시저

- 커서(cursor)는 실행 결과 테이블을 한 번에 한 행씩 처리하기 위하여 테이블의 행을 순서대로 가리키는 데 사용함

키워드	역할
CURSOR 〈cursor 이름〉 IS 〈커서 정의〉	커서를 생성
OPEN 〈cursor 이름〉	커서의 사용을 시작
FETCH 〈cursor 이름〉 INTO 〈변수〉	행 데이터를 가져옴
CLOSE 〈cursor 이름〉	커서의 사용을 끝냄

저장 프로그램

예제 5-4 Orders 테이블의 판매 도서에 대한 이익을 계산하는 프로시저 **Interest.sql**

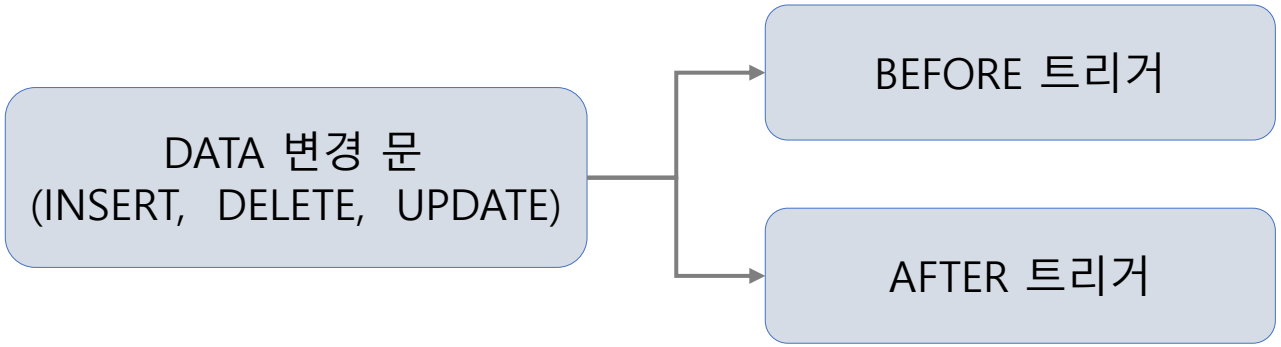
```
01  delimiter //
02  CREATE PROCEDURE Interest()
03  BEGIN
04    DECLARE myInterest INTEGER DEFAULT 0.0;
05    DECLARE Price INTEGER;
06    DECLARE endOfRow BOOLEAN DEFAULT FALSE;
07    DECLARE InterestCursor CURSOR FOR
08                  SELECT saleprice FROM Orders;
09    DECLARE CONTINUE handler
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();
```

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

트리거

■ 트리거(trigger) : 데이터의 변경(INSERT, DELETE, UPDATE) 문이 실행될 때 자동으로 따라서 실행되는 프로시저



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_l INTEGER,
  bookname_l VARCHAR(40),
  publisher_l VARCHAR(40),
  price_l 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
```

트리거

■ Book 테이블에 튜플을 삽입하여 트리거가 실행된 결과

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_I	bookname_I	publisher_I	price_I
14	스포츠 과학 1	이상미디어	25000

Book_log 테이블 (D 행)

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;
```

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

사용자 정의 함수

■ 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 복잡한 로직을 수행	기본값 제공, 데이터 제약 준수, SQL 뷰의 수정, 참조무결성 작업 등을 수행	속성 값을 가공하여 반환, SQL 문에서 직접 사용

저장 프로그램 문법 요약

■ 저장 프로그램의 기본 문법

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)

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

Section 03

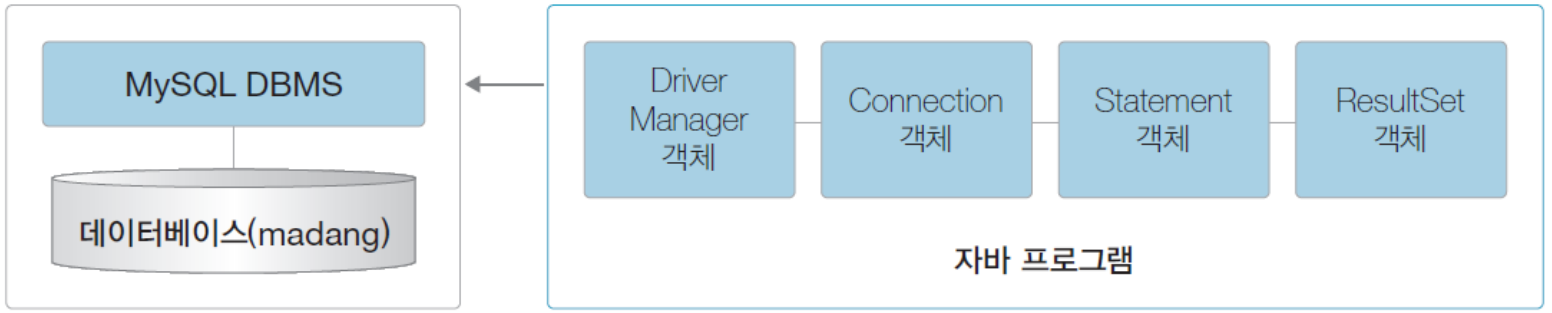
데이터베이스 연동 자바 프로그래밍

소스코드 설명

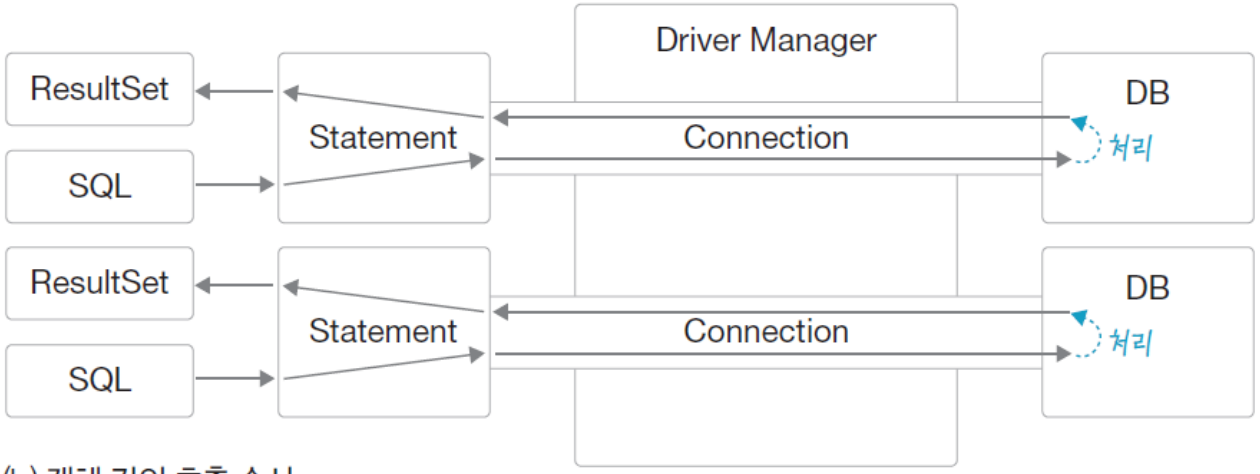
■ 데이터베이스 접속 자바 클래스(java.sql)

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 실행해서 ResultSet 객체를 생성
	ResultSet	boolean first()	결과 테이블에서 커서가 처음 투플을 가리킴
		boolean next()	결과 테이블에서 커서가 다음 투플을 가리킴
		int getInt(〈int〉)	〈int〉가 가리키는 열 값을 정수로 반환
		String getString(〈int〉)	〈int〉가 가리키는 열 값을 문자열로 반환

소스코드 설명



(a) 자바 프로그램의 데이터베이스 연동



(b) 객체 간의 호출 순서

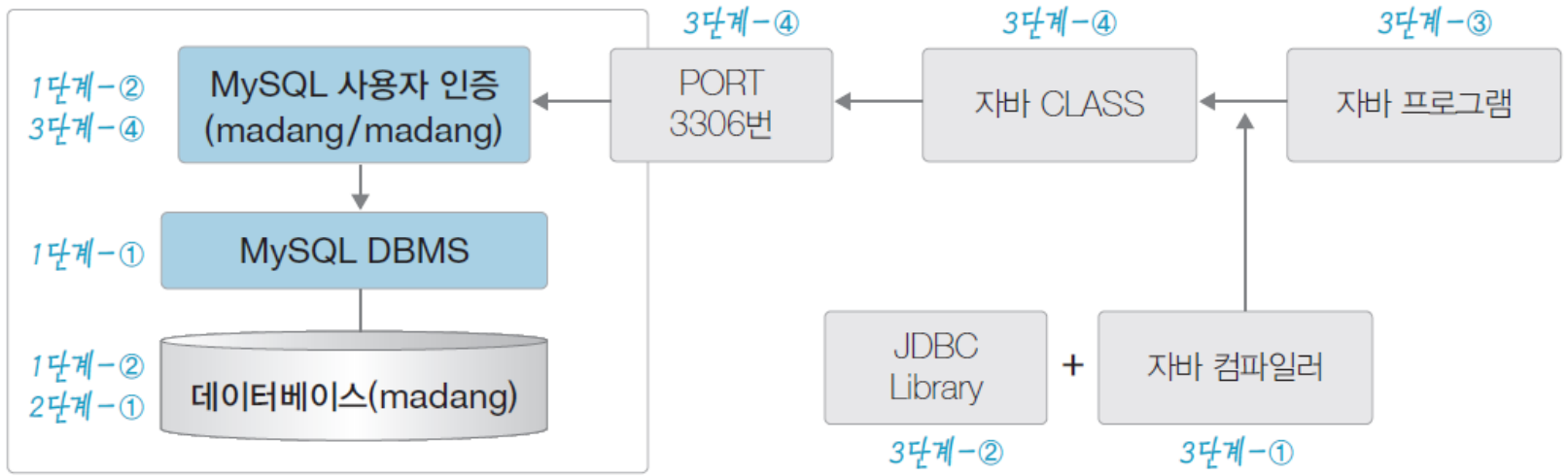
소스코드 설명

■ 자바 프로그램 실행 단계

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

소스코드 설명

■ 데이터베이스 연동 자바 프로그램의 실행 흐름도



오픈플랫폼활용

<Q&A : lsw@kopo.ac.kr>