

Chapter 10

프로시저와 함수, 트리거



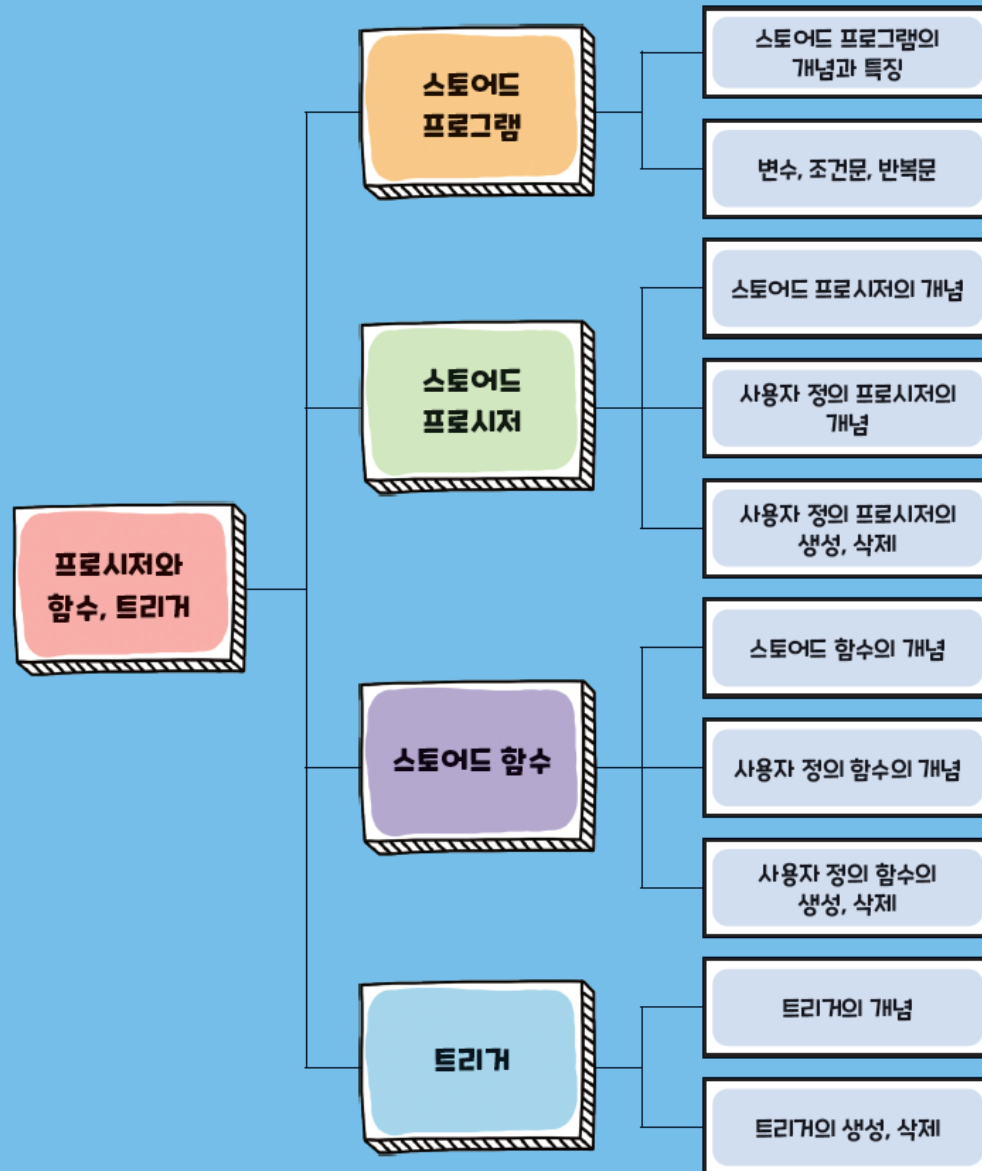
목차

1. 스토어드 프로그램
2. 스토어드 프로시저
3. 스토어드 함수
4. 트리거

학습목표

- 스토어드 프로그램의 개념과 특징을 이해할 수 있습니다.
- 스토어드 프로그램 작성을 위한 문법을 작성할 수 있습니다.
- 스토어드 프로시저와 스토어드 함수의 개념과 사용법을 이해할 수 있습니다.
- 트리거의 개념과 사용법을 이해할 수 있습니다.

Preview



Section 01

스토어드 프로그램

1. 스토어드 프로그램의 개념과 특징

■ SQL 스토어드 프로그램(Stored Program)

- 데이터베이스에서 실행되는 프로그램으로 일련의 SQL문을 포함하고 있는 데이터베이스 객체
- 주로 프로시저나 함수, 트리거 형태로 사용됨

■ 스토어드 프로그램의 특징

- 프로그래밍 언어적 특성
- 코드 캡슐화와 유지 보수
- 재사용성
- 성능 향상
- 보안
- 트랜잭션 관리
- 비즈니스 로직 분리

2. 변수

■ 변수

- 값을 저장하고 참조하는 데 사용되는 식별자

표 10-1 MySQL의 변수 유형

변수 유형	설명	예
사용자 정의 변수 (User-defined Variables)	<ul style="list-style-type: none">세션 내에서 사용자가 정의한 변수를 의미하며, 세션이 종료될 때까지 값을 유지한다.@를 접두사로 사용하며, 변수에 값을 할당하기 위해서는 SET 또는 SELECT를 사용한다.	<ul style="list-style-type: none">값 할당 SET @from = 100, @to = 10000; 또는 SELECT @from = 100, @to = 10000;사용 SELECT * FROM 고객 WHERE 마일리지 BETWEEN @from AND @to
로컬 변수 (Local Variables)	<ul style="list-style-type: none">스토어드 프로시저나 함수와 같은 스토어드 프로그램 내에서 사용되는 변수로, 해당 프로그램 실행이 끝나면 값이 소멸된다.DECLARE문을 사용하여 변수를 선언하고, SELECT나 SET을 사용하여 값을 할당한다.	<ul style="list-style-type: none">DECLARE city VARCHAR DEFAULT '과천시';
시스템 변수 (System Variables)	<ul style="list-style-type: none">MySQL 서버의 동작을 제어하고 구성하는 데 사용되는 변수로 @@를 접두사로 가진다.실행 중인 서버에서 사용되는 현재 값을 보려면 SHOW 또는 SELECT를 사용하며, 값을 변경하려면 SET을 사용한다.	<ul style="list-style-type: none">시스템 변수 확인 SHOW VARIABLES LIKE '%sort_buffer_size%'; 또는 SELECT @@sort_buffer_size;값 할당 SET @@sort_buffer_size = 262144;

3. 조건문

■ IF문

- 조건에 따라 다른 명령을 처리하고자 할 때에는 IF문을 사용함
- 조건이 여러 개일 때에는 ELSE IF로 조건을 추가할 수 있음
- ELSEIF절은 여러 개 작성이 가능하지만, ELSE는 문장 내에서 한 번만 작성 가능함
- 형식

```
IF 조건1 THEN
    조건1이 참일 때 실행할 코드
[ELSEIF 조건2 THEN
    조건2가 참일 때 실행할 코드]
[ELSE
    모든 조건이 참이 아닐 때 실행할 코드]
END IF;
```


3. 조건문

- [예제 10-1] IF문을 사용하여 두 개의 숫자 10과 5의 크기를 비교하는 프로시저를 작성하시오

```
DELIMITER $$
CREATE PROCEDURE proc_if()
BEGIN
    DECLARE x INT;
    DECLARE y INT DEFAULT 5;
    SET x = 10;

    IF x > y THEN
        SELECT 'x는 y보다 큼니다.' AS 결과;
    ELSE
        SELECT 'x는 y보다 작거나 같습니다.' AS 결과;
    END IF;
END $$
DELIMITER ;
```

1

2

3

4

스토어드 프로시저에 대한 자세한 내용은 다음 절에서 살펴봅니다.

- 스토어드 프로시저를 실행할 때는 CALL을 사용함

```
CALL proc_if();
```

▶ 실행결과

결과
x는 y보다 큼니다.

3. 조건문

■ CASE문

- 주로 복잡한 다중 조건을 처리하기 위해 사용됨
- 형식

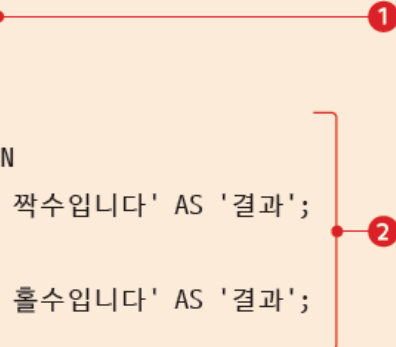
```
CASE
  WHEN 조건1 THEN
    조건1이 참일 때 실행할 코드
  [WHEN 조건2 THEN
    조건2가 참일 때 실행할 코드]
  [ELSE
    모든 조건이 참이 아닐 때 실행할 코드]
END CASE;
```

3. 조건문

- [예제 10-2] CASE문을 사용하여 숫자 10이 짝수인지 홀수인지를 판별하는 프로시저를 작성하시오.

```
DELIMITER $$
CREATE PROCEDURE proc_case()
BEGIN
    DECLARE x INT DEFAULT 10;
    DECLARE y INT;
    SET y = 10 mod 2; ①

    CASE
        WHEN y = 0 THEN
            SELECT 'x는 짝수입니다' AS '결과';
        ELSE
            SELECT 'x는 홀수입니다' AS '결과';
    END CASE;
END $$
DELIMITER ;
```



```
CALL proc_case();
```

▶ 실행결과

결과
x는 짝수입니다

4. 반복문

■ WHILE

- 주어진 조건이 참인 동안 반복적으로 코드를 실행하다가, 조건이 거짓이 되면 반복문을 종료함
- WHILE문은 반복 시작하기 전에 조건을 평가하기 때문에 조건이 처음부터 거짓이면 코드가 실행되지 않음
- 형식

```
WHILE 조건 DO  
    반복적으로 실행할 코드  
END WHILE;
```

4. 반복문

- [예제 10-3] 1부터 10까지의 합을 출력하는 프로시저를 WHILE문으로 작성하시오

```
DELIMITER $$
CREATE PROCEDURE proc_while()
BEGIN
    DECLARE x INT DEFAULT 0;
    DECLARE y INT DEFAULT 0;

    WHILE x < 10 DO
        SET x = x + 1; ❶
        SET y = y + x; ❷
    END WHILE;
    SELECT x, y; ❸
END $$
DELIMITER ;
```

```
CALL proc_while();
```

▶ 실행결과

x	y
10	55

4. 반복문

■ LOOP은

- 탈출 조건이 없으면 무한 반복되기 때문에 반복문 내에서 LEAVE문을 사용하여 루프를 종료해야 함
- 이때 루프에 레이블명을 지정하고, 지정한 레이블명은 LEAVE문에 기술함
- 형식

```
레이블명:LOOP
    반복적으로 실행할 코드
    IF 조건 THEN
        LEAVE 레이블명;
    END IF;
END LOOP;
```

4. 반복문

- [예제 10-4] 1부터 10까지의 합을 출력하는 프로시저를 LOOP문으로 작성하시오.

```
DELIMITER $$
CREATE PROCEDURE proc_loop()
BEGIN
    DECLARE x INT DEFAULT 0;
    DECLARE y INT DEFAULT 0;

    loop_sum:LOOP ●————— ❶
        SET x = x + 1;
        SET y = y + x;
        IF x > 10 THEN ●————— ❷
            LEAVE loop_sum; ●————— ❸
        END IF;
        SELECT x, y;
    END LOOP;
END $$
DELIMITER ;
```

```
CALL proc_loop();
```

▶ 실행결과

x	y
10	55

4. 반복문

하나 더 알기 ✓

ITERATE

ITERATE도 LEAVE와 같이 반복문의 흐름을 제어하는 데 사용할 수 있습니다. LEAVE문은 현재 실행 중인 반복문을 종료하고, 반복문 바로 다음 문장으로 제어를 이동합니다. 반면 ITERATE문은 현재 반복문의 나머지 부분을 건너뛰고, 다음 반복 단계로 제어를 이동합니다.

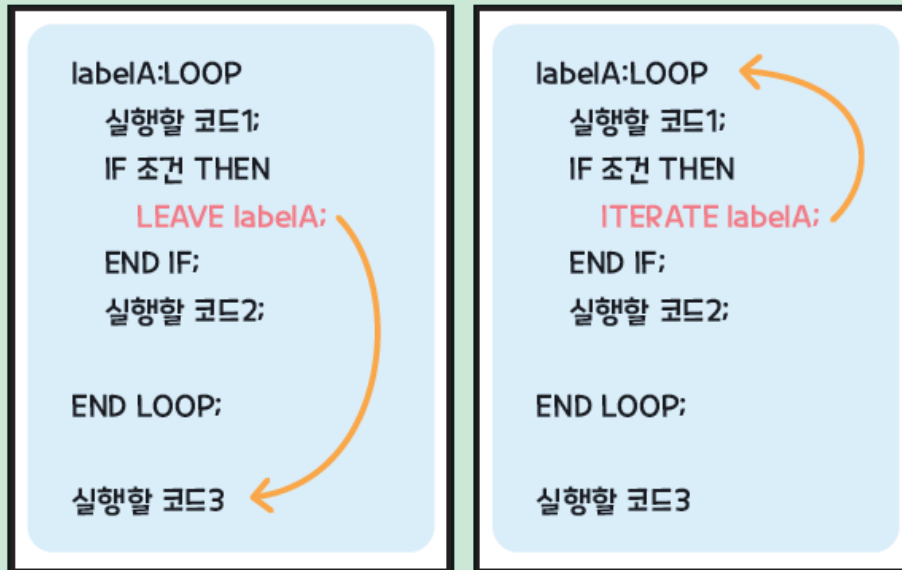


그림 10-1 LEAVE와 ITERATE의 차이점



4. 반복문

■ REPEAT

- 조건이 참이 될 때까지 코드를 실행함
- REPEAT문은 반복문 내에서 조건을 비교하기 전에 최소 한 번은 코드가 실행됨
- 형식

```
REPEAT
```

```
    반복적으로 실행할 코드
```

```
UNTIL 조건;
```

4. 반복문

- [예제 10-5] 1부터 10까지의 합을 출력하는 프로시저를 REPEAT문으로 작성하시오

```
DELIMITER $$
CREATE PROCEDURE proc_repeat()
BEGIN
    DECLARE x INT DEFAULT 0;
    DECLARE y INT DEFAULT 0;

    REPEAT
        SET x = x + 1; ①
        SET y = y + x; ②
    UNTIL x >= 10 END REPEAT; ③
    SELECT x, y;
END $$
DELIMITER ;
```

```
CALL proc_repeat();
```

▶ 실행결과

x	y
10	55

4. 반복문

확인문제

MySQL의 제어문에서 사용할 수 없는 키워드를 모두 선택하시오.

- | | | | |
|-----------|-----------|---------|------------|
| ① IF | ② CASE | ③ FOR | ④ LOOP |
| ⑤ REPEAT | ⑥ BREAK | ⑦ LEAVE | ⑧ CONTINUE |
| ⑨ ITERATE | ⑩ DECLARE | ⑪ SET | |

정답

③ FOR ⑥ BREAK ⑧ CONTINUE

Section 02

스토어드 프로시저

1. 스토어드 프로시저의 개념

■ SQL 스토어드 프로시저(Stored Procedure)

- 데이터베이스 객체 중 하나로 데이터베이스에서 수행할 수 있는 일련의 SQL문과 제어문을 저장한 SQL 스토어드 프로그램을 의미함
- SQL 스토어드 프로시저를 사용하면 하나의 요청으로 여러 개의 SQL문을 실행할 수 있음
- 데이터 검색이나 조작, 업데이트, 삭제 등과 같은 다양한 작업을 수행할 수 있음
- 스토어드 프로시저는 시스템 스토어드 프로시저와 사용자 정의 프로시저로 구분할 수 있음

1. 스토어드 프로시저의 개념

■ 시스템 스토어드 프로시저

- MySQL 데이터베이스 시스템에 내장되어 있는 특수한 유형의 저장 프로그램
- 백업, 복원, 성능 최적화, 사용자 관리 등 주로 데이터베이스 시스템의 내부 동작을 제어하고 관리하는 데 사용됨
- 데이터베이스에 내장되어 있기 때문에 별도로 설치할 필요가 없음
- 사용자가 직접 프로시저를 생성할 수도 있는데, 이러한 프로시저를 사용자 정의 프로시저라고 함

2. 사용자 정의 프로시저의 개념

■ 사용자 정의 프로시저

- 자주 사용하는 SQL문을 프로시저로 생성하여 저장한 후 필요 시에 호출해서 사용할 수 있음

■ 사용자 정의 프로시저의 구성 요소

- 프로시저 정의
- 매개변수
 - ✓ IN 매개변수
 - ✓ OUT 매개변수
 - ✓ INOUT 매개변수
- 변수
- SQL문
- 제어문

3. 사용자 정의 프로시저의 생성과 삭제

■ 사용자 정의 프로시저의 생성, 호출, 삭제

- 생성 형식

```
DELIMITER $$  
CREATE PROCEDURE 사용자 정의 프로시저명  
(  
    [IN|OUT|INOUT 매개변수]  
)  
BEGIN  
    실행코드  
END $$  
DELIMITER ;
```

- 호출 형식

```
CALL 프로시저명;
```

- 삭제 형식

```
DROP PROCEDURE 프로시저명;
```


3. 사용자 정의 프로시저의 생성과 삭제

- [예제 10-6] 고객 정보와 고객 수를 보이는 프로시저를 작성하시오.

```
DELIMITER $$
CREATE PROCEDURE proc_고객정보()
BEGIN
    SELECT *
    FROM 고객;

    SELECT COUNT(*) AS 고객수
    FROM 고객;
END $$
DELIMITER ;
```

```
CALL proc_고객정보();
```

▶ 실행결과

Result Grid		Filter Rows:	Export:	Wrap Cell Content:			
	고객번호	고객회사명	담당자명	담당자직위	주소	도시	지역
▶	ACDDR	굿모닝서울	이은광	영업 과장	송파구 잠실동 220	서울특별시	NULL
	ADHTR	엘케이 상사	김병현	영업 사원	동작구 흑석 3동 140-3	서울특별시	NULL
	AIHTR	씨엔그룹	김성민	영업 사원	가학동 301	광명시	경기도
	ANKFR	오리안무역	최지수	마케팅 과장	성북구 길음동 136-11	서울특별시	NULL
	ANRFR	남해종합식품	강준	마케팅 과장	수택동 435	구리시	경기도

Result 1 ×

Result 2

Result Grid		Filter Rows:
	고객수	
▶	89	

Result 1

Result 2 ×

3. 사용자 정의 프로시저의 생성과 삭제

- [예제 10-7] 도시를 입력하면 해당 도시의 고객 정보와 고객 수를 보이는 프로시저를 작성하시오

```
DELIMITER $$  
CREATE PROCEDURE proc_도시고객정보  
(  
    IN city VARCHAR(50) ●———— 1  
)  
BEGIN  
    SELECT *  
    FROM 고객  
    WHERE 도시 = city; ●———— 2  
  
    SELECT COUNT(*) AS 고객수  
    FROM 고객  
    WHERE 도시 = city;  
END $$  
DELIMITER ;
```

```
CALL proc_도시고객정보('부산광역시');
```

3. 사용자 정의 프로시저의 생성과 삭제

- [예제 10-7] 도시를 입력하면 해당 도시의 고객 정보와 고객 수를 보이는 프로시저를 작성하시오

▶ 실행결과

고객번호	고객회사명	담당자명	담당자직위	주소	도시	지역	전화번호	마일리지
LIDBO	진흥 상사	송아린	대표 이사	부산진구 당감 3동 611-3	부산광역시	HULL	(051)784-1945	3101
LLIWE	태산무역	박소영	영업 과장	부산진구 양정 1동 462-77	부산광역시	HULL	(051)555-5111	887
NGOHU	케이티에스씨	오유진	영업 사원	동구 범일 3동 14-1	부산광역시	HULL	(051)820-1993	8575
RICPE	뉴가든상사	김지호	영업 사원	금정구 정릉동 168-14	부산광역시	HULL	(051)12-1122	1295
TTMBO	털러헬스푸드	안수인	회계 과장	사하구 신명동 701-29	부산광역시	HULL	(051)845-9483	8062

Result 3 × Result 4

고객수
▶ 5

Result 3 Result 4 ×

Error Code 12670이 발생하면 207P의
〈하나 더 알기〉를 참고하세요.

3. 사용자 정의 프로시저의 생성과 삭제

- [예제 10-8] 주문년도와 고객의 도시를 입력하면 해당 년도에 해당 도시의 고객이 주문한 내역에 대하여 주문고객별로 주문건수를 보이는 프로시저를 작성하시오.

```
DELIMITER $$
CREATE PROCEDURE proc_주문년도시_고객정보
(
    IN order_year INT
    ,IN city VARCHAR(50)
)
BEGIN
    SELECT 고객.고객번호
        ,고객회사명
        ,도시
        ,COUNT(*) AS 주문건수
    FROM 고객 JOIN 주문
    ON 고객.고객번호 = 주문.고객번호
    WHERE YEAR(주문일) = order_year
    AND 도시 = city
    GROUP BY 고객.고객번호
        ,고객회사명;

END $$
DELIMITER ;
```

```
CALL proc_주문년도시_고객정보(2021, '공주시');
```

▶ 실행결과

고객번호	고객회사명	도시	주문건수
STCEA	오케이식품	공주시	4

3. 사용자 정의 프로시저의 생성과 삭제

- [예제 10-9] 고객회사명과 추가할 마일리지를 입력하면 해당 고객에 대하여 입력한 마일리지만큼 추가하는 프로시저를 작성하시오.

```
DELIMITER $$
CREATE PROCEDURE proc_고객회사명_마일리지추가
(
    IN company VARCHAR(50)
    ,IN add_mileage INT
)
BEGIN
    SELECT 고객번호
        ,고객회사명
        ,마일리지 AS 변경전마일리지
    FROM 고객
    WHERE 고객회사명 = company;

    UPDATE 고객
    SET 마일리지 = 마일리지 + add_mileage
    WHERE 고객회사명 = company;

    SELECT 고객번호
        ,고객회사명
        ,마일리지 AS 변경후마일리지
    FROM 고객
    WHERE 고객회사명 = company;
END $$
DELIMITER ;
```

만약 UPDATE나 DELETE문을 실행했을 때 오류가 발생한다면 2장 61p의 <하나 더 알기>를 참고하세요.

3. 사용자 정의 프로시저의 생성과 삭제

- [예제 10-9] 고객회사명과 추가할 마일리지를 입력하면 해당 고객에 대하여 입력한 마일리지만큼 추가하는 프로시저를 작성하시오.

```
CALL proc_고객회사명_마일리지추가('진영무역',1000);
```

▶ 실행결과

고객번호	고객회사명	변경전마일리지	고객번호	고객회사명	변경후마일리지
LKOFO	진영무역	9468	LKOFO	진영무역	10468

3. 사용자 정의 프로시저의 생성과 삭제

- [예제 10-10] 고객회사명을 입력하면 해당 고객의 마일리지를 변경하는 프로시저를 작성하시오. 이때 만일 고객의 마일리지가 전체 고객의 평균마일리지보다 크다면 100점을 추가하고, 그렇지 않다면 전 고객의 평균마일리지만큼으로 변경하시오.

```
DELIMITER $$  
CREATE PROCEDURE proc_고객회사명_평균마일리지로변경  
(  
    IN company VARCHAR(50)  
)  
BEGIN  
    DECLARE 평균마일리지 INT;  
    DECLARE 보유마일리지 INT;  
  
    SELECT 고객회사명  
        , 마일리지 AS 변경전마일리지  
    FROM 고객  
    WHERE 고객회사명 = company;  
  
    SET 평균마일리지 = (SELECT AVG(마일리지)  
                        FROM 고객);  
    SET 보유마일리지 = (SELECT 마일리지  
                        FROM 고객  
                        WHERE 고객회사명 = company);
```

1

3. 사용자 정의 프로시저의 생성과 삭제

- [예제 10-10] 고객회사명을 입력하면 해당 고객의 마일리지를 변경하는 프로시저를 작성하시오. 이때 만일 고객의 마일리지가 전체 고객의 평균마일리지보다 크다면 100점을 추가하고, 그렇지 않다면 전 고객의 평균마일리지만큼으로 변경하시오.

```
IF (보유마일리지 > 평균마일리지) THEN
    UPDATE 고객
    SET 마일리지 = 마일리지 + 100
    WHERE 고객회사명 = company;
ELSE
    UPDATE 고객
    SET 마일리지 = 평균마일리지
    WHERE 고객회사명 = company;
END IF;
```

```
SELECT 고객회사명
      , 마일리지 AS 변경후마일리지
FROM 고객
WHERE 고객회사명 = company;
END $$
DELIMITER ;
```


3. 사용자 정의 프로시저의 생성과 삭제

- [예제 10-10] 고객회사명을 입력하면 해당 고객의 마일리지를 변경하는 프로시저를 작성하시오. 이때 만일 고객의 마일리지가 전체 고객의 평균마일리지보다 크다면 100점을 추가하고, 그렇지 않다면 전 고객의 평균마일리지만큼으로 변경하시오.

```
CALL proc_고객회사명_평균마일리지로변경('굿모닝서울');
```

▶ 실행결과

고객회사명	변경전마일리지	고객회사명	변경후마일리지
굿모닝서울	17502	굿모닝서울	17602

3. 사용자 정의 프로시저의 생성과 삭제

- [예제 10-11] 고객회사명을 입력하면 고객의 보유 마일리지에 따라서 등급을 보이는 프로시저를 작성하시오. 이때 고객의 마일리지가 100,000점 이상이면 '최우수고객회사', 50,000점 이상이면 '우수고객회사', 그 나머지는 '관심고객회사'라고 보이시오.

```
DELIMITER $$
CREATE PROCEDURE proc_고객등급
(
  IN company VARCHAR(50),
  OUT grade VARCHAR(20)
)
BEGIN
  DECLARE 보유마일리지 INT;

  SELECT 마일리지
  INTO 보유마일리지
  FROM 고객
  WHERE 고객회사명 = company;

  IF 보유마일리지 > 100000 THEN
    SET grade = '최우수고객회사';
  ELSEIF 보유마일리지 >= 50000 THEN
    SET grade = '우수고객회사';
  ELSE
    SET grade = '관심고객회사';
  END IF;
END $$
DELIMITER ;
```

1

2

3. 사용자 정의 프로시저의 생성과 삭제

- [예제 10-11] 고객회사명을 입력하면 고객의 보유 마일리지에 따라서 등급을 보이는 프로시저를 작성하시오. 이때 고객의 마일리지가 100,000점 이상이면 '최우수고객회사', 50,000점 이상이면 '우수고객회사', 그 나머지는 '관심고객회사'라고 보이시오.

```
CALL proc_고객등급('그린로더스', @그린로더스등급); ● 3
```

```
CALL proc_고객등급('오투락', @오투락등급);
```

```
SELECT @그린로더스등급, @오투락등급; ● 4
```

▶ 실행결과

@그린로더스등급	@오투락등급
우수고객회사	관심고객회사

3. 사용자 정의 프로시저의 생성과 삭제

- [예제 10-12] 인상율과 금액을 입력하면 인상금액을 계산하고, 그 결과를 확인할 수 있는 프로시저를 작성하시오.

```
DELIMITER $$
CREATE PROCEDURE proc_인상금액
(
    IN increase_rate INT
    ,INOUT price INT ①
)
BEGIN
    SET price = price * (1 + increase_rate / 100); ④
END $$
DELIMITER ;
```

```
SET @금액 = 10000; ②
CALL proc_인상금액(10, @금액); ③
SELECT @금액; ⑤
```

▶ 실행결과

@금액
11000

```
CALL proc_인상금액(10, @금액);
SELECT @금액;
```

▶ 실행결과

@금액
12100

3. 사용자 정의 프로시저의 생성과 삭제

확인문제

MySQL의 스토어드 프로시저의 매개변수 유형이 아닌 것을 고르시오.

① IN

② OUT

③ OUTIN

④ INOUT

정답

③ OUTIN

Section 03

스토어드 함수

1. 스토어드 함수의 개념

- SQL 스토어드 함수(Stored Function)
 - 데이터베이스에서 사용할 수 있는 함수
 - SQL 스토어드 함수는 SQL문의 일부로 호출되며, 함수 내에서 로직을 실행하고 값을 반환할 수 있음
- MySQL의 주요 스토어드 함수의 유형
 - 내장 함수
 - ✓ MySQL을 설치하면 자동으로 제공되는 함수
 - 사용자 정의 함수
 - ✓ 사용자가 필요에 따라 직접 정의하는 함수

2. 사용자 정의 함수의 개념

- 사용자 정의 함수의 구성 요소
 - 함수 정의
 - 입력 매개변수
 - 반환 값
 - 제어문

3. 사용자 정의 함수의 생성과 삭제

■ 사용자 정의 함수의 생성, 삭제

- 생성 형식

```
DELIMITER $$  
CREATE FUNCTION 사용자정의함수명(매개변수)  
RETURNS 반환형식  
BEGIN  
    실행코드  
    RETURN 반환값;  
END $$  
DELIMITER ;
```

- 실행 형식

```
SELECT 함수명();
```

- 삭제 형식

```
DROP FUNCTION 함수명;
```

3. 사용자 정의 함수의 생성과 삭제

- [예제 10-13] 수량과 단가를 입력하면 두 수를 곱하여 금액을 반환하는 함수를 생성하시오.

```
DELIMITER $$
CREATE FUNCTION func_금액(quantity INT, price INT)
  RETURNS INT
BEGIN
  DECLARE amount INT;
  SET amount = quantity * price;
  RETURN amount;
END $$
DELIMITER ;
```

```
SELECT func_금액(100, 4500);
```

▶ 실행결과

func_금액(100, 4500)
450000

```
SELECT *
      ,func_금액(주문수량, 단가) AS 주문금액
FROM 주문세부;
```

▶ 실행결과

주문번호	제품번호	단가	주문수량	할인율	주문금액
H0249	14	1900	9	0	17100
H0249	51	4200	40	0	168000
H0250	41	800	10	0	8000
H0250	51	4200	35	0.15	147000

3. 사용자 정의 함수의 생성과 삭제

하나 더 알기 Y

Error Code: 1418 에러 발생

log_bin_trust_function_creators 변수가 0(OFF)으로 설정된 경우에는 사용자는 함수를 생성하거나 수정할 수 없기 때문에 다음과 같은 오류가 발생합니다.

Error Code: 1418. This function has none of DETERMINISTIC, NO SQL, or READS SQL DATA in its declaration and binary logging is enabled.

이때 함수 생성에 제약을 받지 않도록 1(ON)로 설정 값을 변경하면 문제를 해결할 수 있습니다.

```
SET GLOBAL log_bin_trust_function_creators = 1;
```



3. 사용자 정의 함수의 생성과 삭제

확인문제

MySQL에서 사용자 정의 함수를 생성하는 데 사용되지 않는 키워드를 고르시오.

① FUNCTION

② RETURNS

③ RETURN

④ INOUT

정답

④

Section 04

트리거

1. 트리거의 개념

■ 트리거(Trigger)

- 데이터베이스에서 데이터 삽입, 변경 또는 삭제와 같은 특정 이벤트가 발생할 때마다 자동으로 실행되는 작업을 의미함
- INSERT, UPDATE, DELETE와 같은 이벤트가 발생할 때마다 트리거에 정의된 SQL문이 자동 실행됨
- 트리거는 복잡한 비즈니스 규칙 구현 및 데이터 검증, 보안, 로깅 등에 사용됨
- 트리거를 통해 데이터의 일관성을 유지할 수 있으며 다양한 작업을 자동화할 수 있음

■ 트리거의 구성 요소

- 트리거 정의
- 이벤트
- 테이블
- 타이밍
- 본문

2. 트리거의 생성과 삭제

■ 트리거 생성, 삭제

- 생성 형식

```
DELIMITER $$  
CREATE TRIGGER 트리거명  
BEFORE|AFTER INSERT|UPDATE|DELETE ON 테이블명  
FOR EACH ROW  
BEGIN  
    실행코드  
END $$  
DELIMITER ;
```

표 10-2 트리거 이벤트에 따른 OLD, NEW 키워드 사용 가능 여부

트리거 이벤트	OLD.컬럼명	NEW.컬럼명
INSERT	×	○
UPDATE	○	○
DELETE	○	×

- 생성 확인 형식

```
SHOW 트리거명;
```

- 삭제 형식

```
DROP TRIGGER 트리거명;
```

2. 트리거의 생성과 삭제

- [예제 10-14] 제품로그 테이블을 생성하시오. 그리고 제품을 추가할 때마다 로그 테이블에 정보를 남기는 트리거를 작성하시오.

```
CREATE TABLE 제품로그
(
  로그번호 INT AUTO_INCREMENT PRIMARY KEY,
  처리 VARCHAR(10),
  내용 VARCHAR(100),
  처리일 TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

```
DELIMITER $$
CREATE TRIGGER trigger_제품추가로그
AFTER INSERT ON 제품
FOR EACH ROW
BEGIN
  INSERT INTO 제품로그(처리, 내용)
  VALUES('INSERT', CONCAT('제품번호:', NEW.제품번호, '제품명:', NEW.제품명));
END $$
DELIMITER ;
```


2. 트리거의 생성과 삭제

- [예제 10-14] 제품로그 테이블을 생성하시오. 그리고 제품을 추가할 때마다 로그 테이블에 정보를 남기는 트리거를 작성하시오.
 - 트리거 동작 여부는 제품 테이블에 레코드를 추가하고 제품로그 테이블을 검색하여 확인함

```
INSERT INTO 제품(제품번호,제품명,단가,재고)
VALUES(99, '레몬캔디', 2000, 10);
```

```
SELECT *
FROM 제품
WHERE 제품번호 = 99;
```

▶ 실행결과

제품번호	제품명	포장단위	단가	재고	재고금액
99	레몬캔디	NULL	2000	10	20000

```
SELECT *
FROM 제품로그;
```

▶ 실행결과

로그번호	처리	내용	처리일
1	INSERT	제품번호:99 제품명:레몬캔디	2023-11-24 14:17:15

2. 트리거의 생성과 삭제

- [예제 10-15] 제품 테이블에서 단가나 재고가 변경되면 변경된 사항을 제품 로그 테이블에 저장하는 트리거를 생성하시오.

```
DELIMITER $$
CREATE TRIGGER trigger_제품변경로그
AFTER UPDATE ON 제품
FOR EACH ROW
BEGIN
    IF (NEW.단가 <> OLD.단가) THEN ①
        INSERT INTO 제품로그(처리, 내용)
        VALUES ('UPDATE', CONCAT('제품번호:', OLD.제품번호, ' 단가:', OLD.단가, '->', NEW.단가));
    END IF;

    IF (NEW.재고 <> OLD.재고) THEN ②
        INSERT INTO 제품로그(처리, 내용)
        VALUES ('UPDATE', CONCAT('제품번호:', OLD.제품번호, ' 재고:', OLD.재고, '->', NEW.재고));
    END IF;
END $$
DELIMITER ;
```

2. 트리거의 생성과 삭제

- [예제 10-15] 제품 테이블에서 단가나 재고가 변경되면 변경된 사항을 제품 로그 테이블에 저장하는 트리거를 생성하시오.
 - 트리거 동작 여부는 제품 테이블에서 단가나 재고 값을 변경한 후, 제품로그 테이블을 확인함

```
UPDATE 제품
SET 단가 = 2500
WHERE 제품번호 = 99;

SELECT *
FROM 제품로그;
```

▶ 실행결과

로그번호	처리	내용	처리일
1	INSERT	제품번호:99 제품명:레몬캔디	2023-11-24 14:17:15
2	UPDATE	제품번호:99 단가:2000->2500	2023-11-24 14:18:06

2. 트리거의 생성과 삭제

- [예제 10-16] 제품 테이블에서 제품 정보를 삭제하면 삭제된 레코드의 정보를 제품로그 테이블에 저장하는 트리거를 생성하시오.

```
DELIMITER $$
CREATE TRIGGER trigger_제품삭제로그
AFTER DELETE ON 제품
FOR EACH ROW
BEGIN
    INSERT INTO 제품로그(처리, 내용)
    VALUES ('DELETE', CONCAT('제품번호:', OLD.제품번호, ' 제품명:', OLD.제품명));
END $$
DELIMITER ;
```

- 트리거 동작 여부는 제품 테이블에서 레코드를 삭제한 후, 제품로그 테이블을 확인함

```
DELETE FROM 제품
WHERE 제품번호 = 99;

SELECT * FROM 제품로그;
```

▶ 실행결과

로그번호	처리	내용	처리일
1	INSERT	제품번호:99 제품명:레몬캔디	2023-11-24 14:17:15
2	UPDATE	제품번호:99 단가:2000->2500	2023-11-24 14:18:06
3	DELETE	제품번호:99 제품명:레몬캔디	2023-11-24 14:18:30

2. 트리거의 생성과 삭제

확인문제

트리거에서 이벤트가 발생한 값을 확인하기 위해 사용되는 키워드에는 OLD와 NEW가 있습니다. 이 키워드를 어느 이벤트에서 사용할 수 있는지 고르시오.

① INSERT

② UPDATE

③ DELETE

1. OLD : ()

2. NEW : ()

정답

1. ②, ③ 2. ①, ②

점검문제

문제 1

제품명의 일부를 입력하면 해당 제품들에 대하여 제품명별로 주문수량합, 주문금액합을 보이는 프로시저를 작성하시오.

▶ 실행결과

제품명	주문수량합	주문금액합
뉴그린 트로피컬 캔디	138	153900
찰스 계피 캔디	520	950000
미미 스카치 캔디	799	963700

문제 2

생일을 입력하면 연령구분을 반환하는 함수를 생성하시오.

조건

- ① 나이가 20살보다 적으면 '미성년'이다.
- ② 나이가 20살보다 많고 30살보다 적으면 '청년층'이다.
- ③ 나이가 30살보다 많고 55살보다 적으면 '중년층'이다.
- ④ 나이가 55살보다 많고 70살보다 적으면 '장년층'이다.
- ⑤ 그 이외는 '노년층'이다.

▶ 실행결과

사원번호	이름	생일	나이	연령구분
E01	이소미	1985-12-05	38	중년층
E02	배재용	1973-02-17	50	중년층
E03	유대현	1988-08-27	35	중년층
E04	최소민	1987-09-17	36	중년층