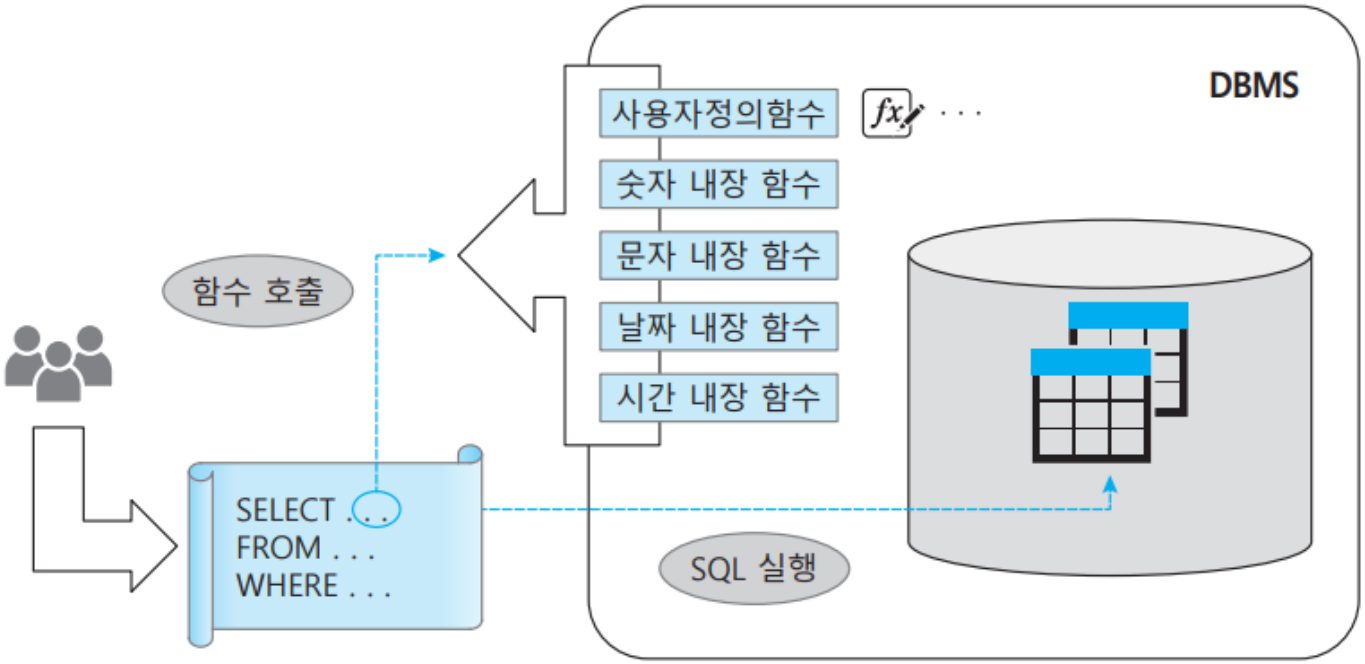
04.SQL 응용

1. 내장 함수

1.1 내장 함수의 개요

- SQL 내장 함수
  - 내장 함수(built-in function)와 사용자 정의 함수(user-defined function) 구분
  - 열 이름이나 상수 값을 입력 받아 값 하나를 결과로 반환
  - 기본 연산 함수와 시스템 정보 제공 함수, 데이터 가공 함수 등이 지원
  - SELECT절이나 WHERE절 뿐만 아니라 UPDATE SET절 등에서도 사용 가능



MySQL 주요 내장 함수

구분	함수	기능
숫자 함수	ABS(값)	입력 값의 절댓값 반환 예 ABS(5) → 5, ABS(−5) → 5
	CEIL(값)	입력 값보다 큰 정수 중에서 가장 작은 수 반환 예 CEIL(1.23) → 2, CEIL(−1.23) → −1
	FLOOR(값)	입력 값보다 작은 정수 중에서 가장 큰 수 반환 예 FLOOR(1.23) → 1, FLOOR(-1.23) → −2
	ROUND(값,자리수)	입력 값을 소수점이하 자리수까지 반올림값한 값 반환 예 ROUND(34.521) → 35, ROUND(34.521,1) → 34.5
	FORMAT(값,형식)	입력 값을 형식에 맞게 변환하여 반환 예 FORMAT(1234567, '#,###,####') → 1,234,567
문자 함수	LENGTH(문자열) CHAR_LENGTH(문자열)	문자열의 길이(바이트수)를 반환 예 LENGTH('데이터베이스') → 12 예 LENGTH('database') → 8 문자열의 길이(문자수)를 반환 예 CHAR_LENGTH('데이터베이스') → 6 예 CHAR_LENGTH('database') → 8
	CONCAT(문자열_리스트)	콤마로 구분된 문자열, 컬럼 값들의 결합 반환 예 CONCAT('데이터','_','베이스') → 데이터_베이스
	LEFT/RIGHT(문자열,길이)	문자열의 왼쪽/오른쪽부터 길이만큼만 반환 예 LEFT('데이터베이스',3) → 데이터 예 RIGHT('데이터베이스',3) → 베이스
	LTRIM/RTRIM(문자열)	문자열의 왼쪽/오른쪽 공백을 제거하여 반환 예 LTRIM(' 데이터베이스 ') → 데이터베이스 예 RTRIM(' 데이터베이스 ') → 데이터베이스
	SUBSTRING(문자열,위치,길이)	문자열의 '위치'번째부터 '길이' 개수만큼의 부분 문자열을 반환 예 SUBSTRING('데이터베이스',2,3) → 이터베
	REPLACE(문자열,검색문자열,치환문자열)	문자열의 일부를 치환하여 반환 예 REPLACE('데이터베이스','데이터','지식') → 지식베이스
	REPEAT(문자열,반복횟수)	문자열을 반복 횟수만큼 반복하여 반환 예 REPEAT('*',7) → *******

MySQL 주요 내장 함수

<table>
<tr><td rowspan="11">날짜
·
시간
함수</td><td>SYSDATE( ), NOW( )</td><td>현재 날짜와 시간을 반환</td></tr>
<tr><td>CURRENT_DATE( )</td><td>현재 날짜를 반환</td></tr>
<tr><td>CURRENT_TIME( )</td><td>현재 시간을 반환</td></tr>
<tr><td>YEAR(날짜)
MONTH(날짜)
DAY(날짜)</td><td>입력 날짜의 연도/월/일 부분을 반환</td></tr>
<tr><td>HOUR(시간)
MINUTE(시간)
SECOND(시간)</td><td>입력 시간의 시/분/초 부분을 반환</td></tr>
<tr><td>LAST_DAY(날짜)</td><td>입력 날짜의 해당 월의 마지막 날짜를 반환</td></tr>
<tr><td>DATE_ADD(날짜, INTERVAL 증분값 DAY/MONTH/YEAR)</td><td>입력 날짜에서 증분값만큼 날/월/년을 더한 날짜를 반환
예 DATA_ADD('2020-12-12', INTERVAL 42 DAY) → 2021-01-23</td></tr>
<tr><td>DATE_SUB(날짜, INTERVAL 감소값 DAY/MONTH/YEAR)</td><td>입력 날짜에서 감소값만큼 날/월/년을 뺀 날짜를 반환
예 DATA_SUB('2020-12-12', INTERVAL 13 DAY) → 2019-11-29</td></tr>
<tr><td>DATE_FORMAT(날짜,'형식')</td><td>입력 날짜를 형식에 맞게 변환하여 반환
<table>
<tr><th>형식</th><th>표시 내용</th></tr>
<tr><td>%Y</td><td>4자리연도(1999, 2000)</td></tr>
<tr><td>%y</td><td>2자리연도(99, 00)</td></tr>
<tr><td>%c</td><td>월 (0 ~ 12)</td></tr>
<tr><td>%m</td><td>월 (00 ~ 12)</td></tr>
<tr><td>%M</td><td>월 (January ~ December)</td></tr>
<tr><td>%b</td><td>월 (Jan ~ Dec)</td></tr>
<tr><td>%d</td><td>일 (01 ~ 31)</td></tr>
<tr><td>%e</td><td>일 (0 ~ 31)</td></tr>
<tr><td>%a</td><td>주 (Sun ~ Sat)</td></tr>
</table>
예 DATE_FORMAT('2019-12-12','%Y %M %d %a') → 2019 December 12 Fri</td></tr>
</table>

1.2 내장 함수의 적용

1) 숫자 함수

```
SELECT ABS(+17), ABS(-17), CEIL(3.28), FLOOR(4.259) ;
```

```
SELECT 학번, SUM(기말성적)/COUNT(*), ROUND(SUM(기말성적)/COUNT(*), 2)
FROM 수강
GROUP BY 학번
```

2) 문자 함수

```
SELECT LENGTH(소속학과), RIGHT(학번,2), REPEAT('*',나이), CONCAT(소속학과,'학과')
FROM 학생 ;
```

```
SELECT SUBSTRING(주소,1,2), REPLACE(SUBSTRING(휴대폰번호,5,9),'-',',')
FROM 학생 ;
```

내장 함수의 적용

3) 날짜·시간 함수

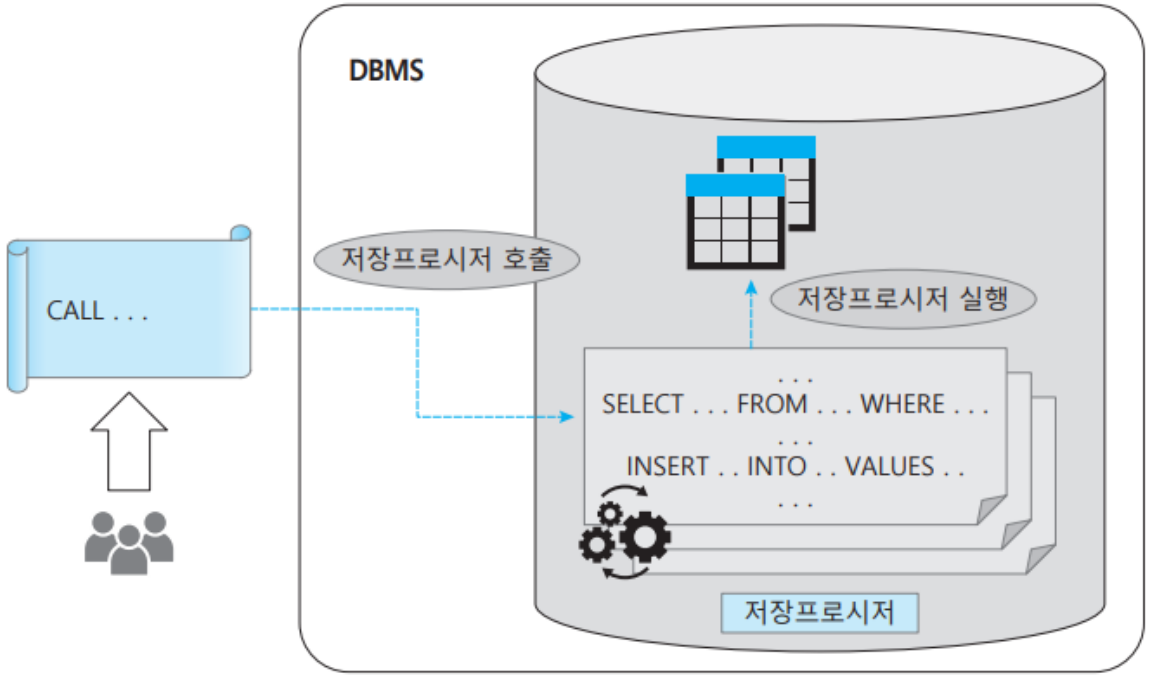
```
SELECT 신청날짜, LAST_DAY(신청날짜)
FROM 수강
WHERE YEAR(신청날짜) = '2019' ;
```

```
SELECT SYSDATE( ), DATEDIFF(신청날짜,'2019-01-01')
FROM 수강 ;
```

```
SELECT 신청날짜, DATE_FORMAT(신청날짜,'%b/%d/%y'), DATE_FORMAT(신청날짜,'%Y년%c월%e일')
FROM 수강 ;
```

2. 저장 프로시저

- 저장 프로시저(stored procedure)
  - 미리 작성하여 데이터베이스 안에 저장한 SQL 문장들의 묶음
  - 독립된 프로그램으로 데이터베이스 안에 하나의 객체로 저장
  - 장점: 최적화된 SQL문을 미리 데이터베이스에 작성해둘 수 있고 복잡한 SQL문을 전달할 필요가 없어 네트워크 부하가 줄어들며 여러 응용 프로그램간의 공유가 가능함



저장 프로시저

- CREATE PROCEDURE문

```
CREATE PROCEDURE 프로시저_이름
BEGIN
 ...
 SQL 명령문 ;
 ...
END
```

삽입 · 수정 저장 프로시저 생성

예제 7-1 '과목' 테이블에 데이터를 추가(입력 또는 수정)하는 저장 프로시저를 작성하시오

```
❶ DELIMITER //
❷ CREATE PROCEDURE InsertOrUpdateCourse (
❸   IN CourseNo    VARCHAR(4),
    IN CourseName  VARCHAR(20),
    IN CourseRoom  CHAR(3),
    IN CourseDept  VARCHAR(20),
    IN CourseCredit  INT)
❹ BEGIN
❺   DECLARE Count INT;
❻   SELECT COUNT(*) INTO Count FROM 과목 WHERE 과목번호 = CourseNo  ;
❼   IF (Count = 0) THEN
      INSERT INTO 과목(과목번호, 이름, 강의실, 개설학과, 시수)
      VALUES(CourseNo, CourseName, CourseRoom, CourseDept, CourseCredit) ;
    ELSE
      UPDATE 과목
      SET 이름 = CourseName,  강의실 = CourseRoom, 개설학과 = CourseDept, 시수 = CourseCredit
      WHERE 과목번호 = CourseNo ;
    END IF ;
  END ❽ //
❾ DELIMITER ;
```

저장 프로시저 호출

● 삽입 저장 프로시저의 호출

예제 7-2 'InsertOrUpdateCourse' 저장 프로시저를 호출하여 '과목' 테이블에 '연극학개론' 과목을 등록하시오.

```
-- 행 삽입 예
⑩ CALL InsertOrUpdateCourse('c006', '연극학개론', '310', '교양학부', 2) ;
⑪ SELECT * FROM 과목 ;
```

● 수정 저장 프로시저의 호출

예제 7-3 'InsertOrUpdateCourse' 저장 프로시저를 호출하여 '과목' 테이블의 '연극학개론' 과목 강의실을 '410'으로 수정하시오.

```
-- 행 수정 예
⑫ CALL InsertOrUpdateCourse('c006', '연극학개론', '410', '교양학부', 2) ;
⑬ SELECT * FROM 과목 ;
```

2.2 검색 저장 프로시저의 생성 및 활용

예제 7-4 '수강' 테이블에서 중간 성적 혹은 기말 성적으로 특정 점수 이상을 받은 학생수를 반환하는 'SelectAverageOfBestScore' 프로시저를 작성하시오.

```
DELIMITER //
CREATE PROCEDURE SelectAverageOfBestScore (
 ❶ IN Score INT,
 ❷ OUT Count INT)
BEGIN
 ❸ DECLARE NoMoreData INT DEFAULT FALSE;
    DECLARE Midterm INT ;
    DECLARE Final INT ;
    DECLARE Best INT ;
 ❹ DECLARE ScoreListCursor CURSOR FOR
       SELECT 중간성적, 기말성적 FROM 수강 ;
 ❺ DECLARE CONTINUE HANDLER FOR NOT FOUND SET NoMoreData = TRUE ;
       SET Count = 0;
 ❻ OPEN ScoreListCursor ;
 ❼ REPEAT
 ❽     FETCH ScoreListCursor INTO Midterm, Final ;
       IF Midterm > Final THEN
          SET Best = Midterm ;
       ELSE
          SET BEST = Final ;
       END IF ;
       IF (Best >= Score) THEN
          SET Count = Count + 1 ;
       END IF ;
    UNTIL NoMoreData END REPEAT ;
 ❾ CLOSE ScoreListCursor ;
END ;
//
DELIMITER ;
```

프로시저 호출 및 삭제

- 검색 저장 프로시저의 호출

예제 7-5 'SelectAverageOfBestScore' 저장 프로시저를 호출하여 중간 혹은 기말 성적 중 95점 이상 받은 학생수를 검색하시오.

```
-- 행 검색 예
CALL SelectAverageOfBestScore(95, @Count) ;
SELECT @Count ;
```

2.3 저장 프로시저의 삭제

- DROP PROCEDURE 명령문의 형식

```
DROP PROCEDURE 프로시저_이름 ;
```

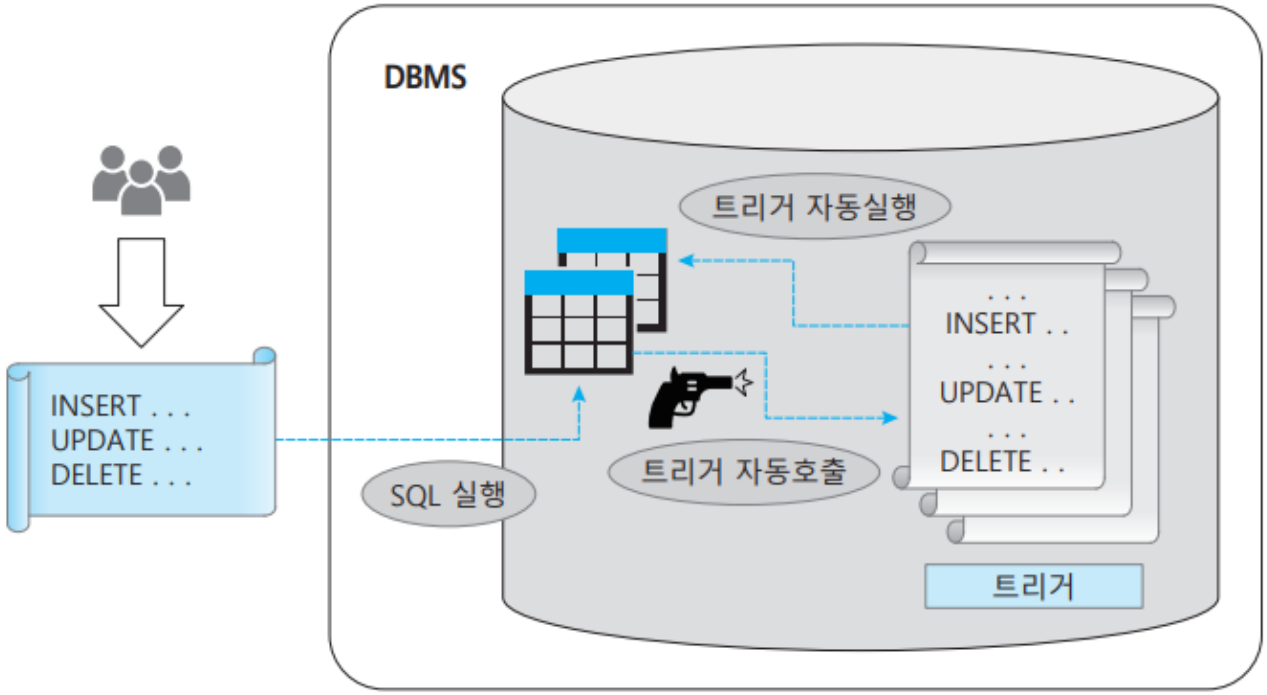
```
SHOW CREATE PROCEDURE InsertOrUpdateCourse ;
```

예제 7-6 'InsertOrUpdateCourse' 프로시저를 삭제하시오.

```
DROP PROCEDURE InsertOrUpdateCourse ;
```

3. 트리거 (trigger)

- 트리거
  - 데이터 변경 등 명세된 이벤트 발생시 감지하여 자동 실행되는 사용자 정의 프로시저
  - INSERT, UPDATE, DELETE 명령문의 실행 직전·후 자동으로 호출되어 실행
  - 보통 무결성 제약 조건을 유지하거나 업무 규칙 등을 적용하기 위해 사용



트리거

- 트리거를 생성하는 CREATE TRIGGER 명령문의 형식

```
CREATE TRIGGER 트리거_이름
[ ❶BEFORE | ❷AFTER ][ INSERT|UPDATE|DELETE ] ON 테이블이름 FOR EACH ROW
BEGIN
...
 SQL 명령문 ;
...
END
```

트리거 참조 테이블 OLD & NEW

테이블에 어떠한 처리가 이루어지기 직전의 값과 직후의 값들을 저장하는 특별한 테이블이 OLD 테이블과 NEW 테이블이다.

- OLD.열이름 : 처리 직전의 특정 열 값
- NEW.열이름 : 처리 직후의 특정 열 값

트리거의 생성 및 실행

- 트리거의 생성

```
   DELIMITER //
❻ CREATE TRIGGER AfterInsertStu
❼ AFTER INSERT ON 학생 FOR EACH ROW
   BEGIN
❽   IF ( NEW.성별 = '남' ) THEN
        UPDATE 남녀학생총수 SET 인원수 = 인원수 + 1 WHERE 성별 = '남' ;
     ELSEIF ( NEW.성별 = '여' ) THEN
        UPDATE 남녀학생총수 SET 인원수 = 인원수 + 1 WHERE 성별 = '여' ;
     END IF ;
   END ;
   //
   DELIMITER  ;
```

- 트리거의 실행

```
❾ INSERT INTO 학생
   VALUES ('s008', '최동석', '경기 수원', 2, 26, '남', '010-8888-6666', '컴퓨터') ;

   SELECT * FROM 학생 ;
❿ SELECT * FROM 남녀학생총수 ;
```

3.2 트리거의 삭제

- DROP TRIGGER 명령문의 형식

```
DROP TRIGGER 트리거_이름 ;
```

- DROP TRIGGER 적용 예

```
SHOW TRIGGERS ;
```

예제 7-8 'AfterInsertStu' 트리거를 삭제하시오.

```
DROP TRIGGER AfterInsertStu ;
```

4. 사용자 정의 함수

- 사용자 정의 함수(user defined function)
  - 사용자가 직접 정의한 함수로 DBMS 안에 독립된 데이터베이스 객체로 저장
  - SELECT문이나 프로시저 안에서 호출되어 특정 기능을 수행하고 결과 값을 반환하는 용도로 사용
    - 스칼라 함수는 하나의 값 또는 NULL을 반환
    - 테이블 함수는 각 행이 하나 이상의 열로 구성된 테이블을 반환

- 사용자 정의 함수를 생성하는 CREATE FUNCTION 명령문의 형식

```
CREATE FUNCTION 함수명(매개변수 매개변수_자료형)
RETURNS 반환값_자료형
BEGIN
  . . .
  SQL 명령문 ;
  . . .
  RETURN 반환값 ;
END
```

사용자 함수의 정의

예제 7-9 '수강' 테이블에서 학생의 학점이 A이면 '최우수', B이면 '우수', C이면 '보통' D나 F이면 '미흡'으로 변환하여 반환하는 사용자 정의 함수를 작성하시오.

```
  DELIMITER //
❶ CREATE FUNCTION Fn_Grade( grade CHAR(1) )
❷ RETURNS VARCHAR(10)
❸ BEGIN
    DECLARE ret_grade VARCHAR(10) ;
    IF ( grade = 'A' ) THEN
      SET ret_grade = '최우수' ;
    ELSEIF ( grade = 'B' ) THEN
      SET ret_grade = '우수' ;
    ELSEIF ( grade = 'C' ) THEN
      SET ret_grade = '보통' ;
    ELSEIF ( grade = 'D' OR grade = 'F' ) THEN
      SET ret_grade = '미흡' ;
    END IF ;
    RETURN ret_grade ;
  END
  //
  DELIMITER ;
```

사용자 함수의 적용

- 사용자 함수의 적용

예제 7-10 '수강' 테이블에서 전체 학생의 '학번', '과목번호', '평가학점'과 한글로 변환된 '평가등급'을 검색하시오('Fn_Grade( )' 사용자 정의 함수를 활용).

```
SELECT 학번, 과목번호, 평가학점, Fn_Grade(평가학점) AS '평가 등급' FROM 수강 ;
```

4.2 사용자 정의 함수의 삭제

- DROP FUNCTION 명령문의 형식

```
DROP FUNCTION 사용자정의함수_이름 ;
```

- DROP FUNCTION의 적용 예

```
SHOW CREATE FUNCTION Fn_Grade ;
```

예제 7-11 사용자 정의 함수 'Fn_Grade( )'를 삭제하시오.

```
DROP FUNCTION Fn_Grade ;
```

저장 프로시저, 트리거, 사용자 정의 함수의 비교

- 저장 프로시저
  - 여러 응용 프로그램 사이에 공유함으로써 처리의 일관성 향상과 보안 강화
  - 전송되는 SQL 명령문의 양을 줄이고 미리 컴파일된 코드를 호출함으로써 반복적인 처리 시성능 향상
- 트리거
  - 복잡한 데이터 무결성을 강화, 다양한 데이터 처리 업무 규칙을 구현
  - 성능이 다소 저하될 수 있음
- 사용자 정의 함수
  - 특정 값뿐만 아니라 테이블도 반환할 수 있어 제한된 SQL 명령문의 기능을 확장시키고 명령문 작성의 편의성을 향상시킴

구분	저장 프로시저	트리거	사용자 정의 함수
공통점	데이터베이스 객체 DBMS 안에 저장 절차적 언어로 작성		
차이점	CREATE PROCEDURE CALL문으로 직접 호출 실행 SQL문을 포함 반환값 제공(선택사항)	CREATE TRIGGER INSERT, UPDATE, DELETE문 실행시 자동 호출 실행 SQL문을 포함 반환값 없음	CREATE FUNCTION SELECT문 실행시 간접 호출 실행 SQL문에 포함 반환값 제공(필수사항)