Stored Program

MySQL : Procedure

대전 1반 이현정

스토어드 프로그램(Stored Program)

01. MySQL안에서 사용 가능한 **프로그래밍 언어**와 같은 기능

02. **스토어드 프로시저**, 스토어드 함수, 트리거 등이 있음

03. **쿼리를 모듈화** 시켜 필요할 때마다 호출하여 편리하게 MySQL을 운영

스토어드 프로시저(Stored Procedure)의 특징

파라미터를 받을 수 있고, 반복해서 사용할 수 있는 이름이 있는 블록

MySQL의 성능 향상

긴 코드의 쿼리를 서버에 스토어드 프로시저로 생성하여 네트워크 부하를 줄일 수 있음

간편한 유지 관리

일관된 수정 및 유지보수 등의 작업이 가능함

모듈식 프로그래밍

한번 생성 시 언제든지 실행 가능함

보안 강화 가능

스토어드 프로시저에만 접근 권한을 부여할 수 있음

스토어드 프로시저(Stored Procedure) 형식

파라미터를 받을 수 있고, 반복해서 사용할 수 있는 이름이 있는 블록

```
DELIMITER $$
CREATE  PROCEDURE  stored  procedure  name([IN|OUT|INOUT  parameter])
BEGIN

    statement list; -- SQL문, 흐름 제어문 등 포함

END $$
DELIMITER ;
```

스토어드 프로시저(Stored Procedure) 실행 형식

프로시저의 파라미터에 따라 실행 형식이 달라짐

입력

```
CALL stored procedure name (in parameter value);
```

출력

```
CALL stored procedure name (@out parameter value);
SELECT @out variable name;
```

입출력

```
SET  @inout variable name = inout parameter value;
CALL stored procedure name (@inout parameter value);
SELECT @inout variable name;
```

스토어드 프로시저(Stored Procedure) : DELIMITER

구분 문자를 다른 구분 문자로 변경하는 것

DELIMITER

구분 문자를 일시적으로 변경하여 전체 스토어드 프로그램 정의를 단일 구분으로 인식하기 위해 사용

=> 스토어드 프로그램 내부의 SQL문 실행과 스토어드 프로그램을 구분하기 위해 사용

스토어드 프로그램을 선언한 후, 구분 문자를 원래대로 변경해야 함

구분 문자는 백 슬래시(\)를 단독 사용하는 것 외에 원하는 문자로 변경 가능

일반적으로 $$을 사용

DELIMITER new delimiter

스토어드 프로시저(Stored Procedure) 내 변수 선언 및 저장

프로시저는 프로그래밍 언어처럼 변수를 선언하여 사용 가능

선언

```
DECLARE variable name data type [DEFAULT value] ;
```

저장 1

```
SET variable name = value;
```

저장 2

```
SELECT   select list
INTO     variable name
FROM     table
WHERE    condition;
```

스토어드 프로시저(Stored Procedure) 제어 구조 : IF문

프로시저는 프로그래밍 언어처럼 다양한 제어 구조를 지원

IF문 조건에 따라 statements를 실행

```
IF condition THEN
    statements;
[ELSEIF condition THEN
    statements;]
[ELSE
    statements;]
END IF;
```

스토어드 프로시저(Stored Procedure) 제어 구조 : LOOP문

프로시저는 프로그래밍 언어처럼 다양한 제어 구조를 지원

LOOP문 statements를 반복적으로 실행하고, LEAVE문을 만나면 종료

```
[begin label :] LOOP
    IF condition THEN
        LEAVE label;
    END IF;
    statements;
END LOOP [end label];
```

스토어드 프로시저(Stored Procedure) 제어 구조 : REPEAT문

프로시저는 프로그래밍 언어처럼 다양한 제어 구조를 지원

REPEAT문 UNTIL의 condition이 참이 될 때까지 statement를 반복적으로 실행

```
[begin label :] REPEAT
    statements;
UNTIL condition
END REPEAT [end label];
```

스토어드 프로시저(Stored Procedure) 제어 구조 : WHILE문

프로시저는 프로그래밍 언어처럼 다양한 제어 구조를 지원

WHILE문 condition이 참인 동안 statement를 반복적으로 실행

```
[begin label :] WHILE
condition DO
    statements;
END WHILE [end label];
```

스토어드 프로시저(Stored Procedure) 제어 구조 : ITERATE문

프로시저는 프로그래밍 언어처럼 다양한 제어 구조를 지원

ITERATE문

프로그래밍 언어의 CONTINUE와 비슷한 역할

LOOP, REPEAT, WHILE문에서 사용

지정한 label이 정의된 반복문으로 가서 계속 실행

```
[begin label :] REPEAT
    IF condition THEN
        ITERATE label;
    END IF;
    statements;
UNTIL condition
END REPEAT [end label];
```

```
[begin label :] WHILE
condition DO
 IF condition THEN
        ITERATE label;
    END IF;
    statements;
END WHILE [end label];
```

스토어드 프로시저(Stored Procedure) 제어 구조 : LEAVE문

프로시저는 프로그래밍 언어처럼 다양한 제어 구조를 지원

LEAVE문

프로그래밍 언어의 BREAK와 비슷한 역할

LOOP, REPEAT, WHILE문에서 사용

지정한 label이 정의된 반복문을 종료

```
[begin label :] REPEAT
    IF condition THEN
        LEAVE label;
    END IF;
    statements;
UNTIL condition
END REPEAT [end label];
```

```
[begin label :] WHILE
condition DO
 IF condition THEN
        LEAVE label;
    END IF;
    statements;
END WHILE [end label];
```

Thank You :)