데이터베이스 입문

Stored Procedure

㈜ 코드스쿼드 Honux

GOAL

스토어드 프로시저를 사용할 수 있다.

스토어드 프로시저 문법 이해
스토어드 프로시저 실습

SQL의 기본 명령은 선언적 명령입니다.

데이터베이스에서 절차적 명령을 수행하기 위해서는 **스토어드 프로그램**을 사용해야 합니다.

스토어드 프로그램의 종류

스토어드 함수

스토어드 프로시져

트리거

이벤트 핸들러

스토어드 프로그램의 장점

응용 프로그램의 성능 향상!

네트워크 트래픽 감소!

보안성 향상!

개발 업무의 구분

스토어드 프로그램의 단점

유지 보수가 매우 매우 어렵다!

Git에서 관리가 쉽지 않다.

문제가 생겼을 때 rollback은?

명령 자체의 성능 감소?

스토어드 프로시져를 사용하는 곳

게임분야 : 많이 사용

웹분야 : case by case 인데 사용이 줄어들고 있다고 함

Stored Function vs Stored Procedure

함수는 쿼리 내에 사용할 수 있지만 프로시져는 불가능

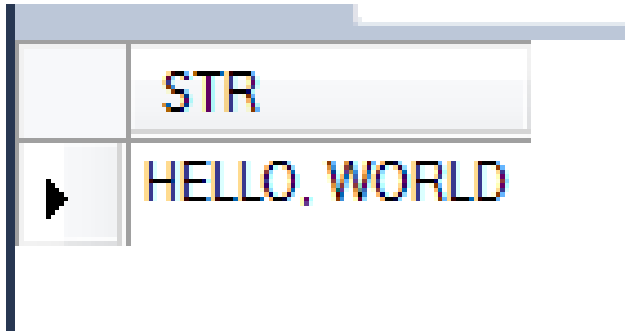
함수는 대신에 제약사항이 많음

우리는 프로시저만 다룸

Hello World

```sql
DROP PROCEDURE IF EXISTS SP_HELLO;
DELIMITER $$
CREATE PROCEDURE SP_HELLO()
    BEGIN
        DECLARE STR CHAR(20) DEFAULT 'POPI';
        SET STR = 'HELLO, WORLD';
        SELECT STR;
    END $$
DELIMITER ;

CALL SP_HELLO();
```



기본 변수 사용하기

기본 변수 선언

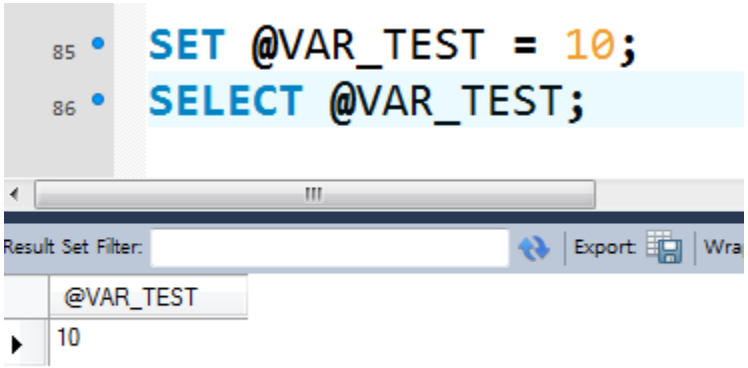
```
DELCARE 변수명 타입 [DEFALUT 기본값]
```

기본 변수는 프로시져 내에서만 유효합니다.

세션 변수 선언 및 사용

```
SET @변수명 = 값;
```

세션 변수는 말 그대로 세션 내에서 계속 유효합니다.



기본 변수에 값 넣기

SET 사용

```
14  DROP PROCEDURE IF EXISTS SP_TEST1;
15  DELIMITER $$
16  CREATE PROCEDURE SP_TEST1()
17      BEGIN
18          DECLARE A INT;
19          SET A = 10;
20          SET A = A * 2;
21          SELECT A;
22      END $$
23  DELIMITER ;
24
25  CALL SP_TEST1();
```

SELECT INTO 사용

쿼리의 결과를 변수에 넣을 수 있음

쿼리의 결과값이 스칼라 값일 경우만 가능

```
DROP PROCEDURE IF EXISTS SP_TEST2;
DELIMITER $$
CREATE PROCEDURE SP_TEST2()
    BEGIN
        DECLARE A INT;
        SELECT COUNT(UID) INTO A FROM USER_INFO;
        SELECT A;
    END $$
DELIMITER ;

CALL SP_TEST2();
```

결과값을 테이블에 넣기

```
CREATE TABLE TEST3(NUM INT);
INSERT INTO TEST3 VALUES (1);
DROP PROCEDURE IF EXISTS SP_TEST3;
DELIMITER $$
CREATE PROCEDURE SP_TEST3()
    BEGIN
        DECLARE A INT DEFAULT 1;
        SELECT MAX(NUM) INTO A FROM TEST3;
        SET A = A + 1;
        INSERT INTO TEST3 VALUES (A);
        SELECT * FROM TEST3;
    END $$
DELIMITER ;
```

```
59  CALL SP_TEST3();
60  CALL SP_TEST3();
61  CALL SP_TEST3();
62  SELECT * FROM TEST3;
```

NUM
1
2
3
4
5

매개변수 사용하기

IN parameter : 입력에 사용

```
DROP TABLE IF EXISTS TEST4;
CREATE TABLE  TEST4 (
    ID INT PRIMARY KEY AUTO_INCREMENT,
    NAME VARCHAR(20)
);
INSERT INTO TEST4 VALUES(NULL, 'Amuro');
INSERT INTO TEST4 VALUES(NULL, 'Char');
INSERT INTO TEST4 VALUES(NULL, 'Kamille');
INSERT INTO TEST4 VALUES(NULL, 'Four');
```

```sql
DROP PROCEDURE IF EXISTS SP_TEST4;
DELIMITER $$
CREATE PROCEDURE SP_TEST4(IN pname varchar(10) )
    BEGIN
        DECLARE A INT DEFAULT 1;
        SELECT * FROM TEST4 WHERE NAME = pname;
    END $$
DELIMITER ;

CALL SP_TEST4('Four');
```

ID	NAME
4	Four

OUT 매개변수 사용하기

```
DROP PROCEDURE IF EXISTS SP_TEST5;
DELIMITER $$
CREATE PROCEDURE SP_TEST5(IN pname varchar(10),
    OUT pid int)
    BEGIN
        DECLARE A INT DEFAULT 1;
        SELECT id into pid
            FROM TEST4
            WHERE NAME = pname;
    END $$
DELIMITER ;

CALL SP_TEST5('Four', @var_ret);
select @var_ret;
```

IF 사용하기

IF도 사용이 가능합니다.

```
IF if_expression THEN commands
   [ELSEIF elseif_expression THEN commands]
   [ELSE commands]
   END IF;
```

END IF; 로 끝나야 한다는 점을 주의해야 합니다.

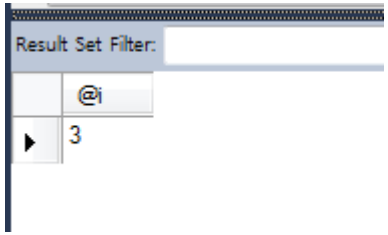
```sql
DROP TABLE IF EXISTS TEST6;
CREATE TABLE TEST6(NUM INT);

DROP PROCEDURE IF EXISTS SP_TEST6;
DELIMITER $$
CREATE PROCEDURE SP_TEST6(OUT RET INT)
    BEGIN
        DECLARE A INT DEFAULT 1;
        SELECT MAX(NUM) INTO A FROM TEST6;
        IF A IS NULL THEN
            SET A = 1;
        ELSE
            SET A = A + 1;
        END IF;
        INSERT INTO TEST6 VALUES (A);
        SET RET = A;
    END $$
DELIMITER ;
```

```
CALL SP_TEST6(@i);
SELECT @i;

CALL SP_TEST6(@i);
SELECT @i;

CALL SP_TEST6(@i);
SELECT @i;
```



WHILE 사용하기

WHILE 도 됩니다.

```
1 WHILE expression DO
2     Statements
3 END WHILE
```

역시 **END WHILE;** 을 조심하세요.

기타

case (switch case 와 같은 용도)

REPEAT ~ UNTIL (do while과 같은 용도)

실습1

디비로 별 찍기 ㅋㅋ

```
mysql> call sp_star(10);
Query OK, 1 row affected (0.05 sec)

mysql> select * from star;
+-----+------------+
| LNO | LINE       |
+-----+------------+
|   1 | *          |
|   2 | **         |
|   3 | ***        |
|   4 | ****       |
|   5 | *****      |
|   6 | ******     |
|   7 | *******    |
|   8 | ********   |
|   9 | *********  |
|  10 | ********** |
+-----+------------+
10 rows in set (0.00 sec)
```

실습2

테이블 스키마 생성

USER

ID	NAME	LAST_VISIT	MONEY	NUM_TRADE

TRADE

TNUM	SELLER	ITEM_NAME	MARKETID

SELLER --> USER(ID)

MARKETID --> MARKET(ID)

MARKET

ID	NAME	STATUS	MANAGER	NUM_TRADE

MANAGER --> USER(ID)

테이블의 특정 컬럼을 이용해서 값 업데이트하기

```
UPDATE USER
        INNER JOIN
        (SELECT SELLER, COUNT(*) AS USER_TRADE
         FROM TRADE
        GROUP BY SELLER) T ON USER.ID = T.SELLER

SET USER.NUM_TRADE = USER_TRADE;
```

물품 등록 PROCEDURE 만들기
등록과 동시에 USER와 MARKET의 NUM_TRADE 값 증가

USER

ID	NAME	LAST_VISIT	MONEY	NUM_TRADE

TRADE

TNUM	SELLER	ITEM_NAME	MARKETID

SELLER --> USER(ID)

MARKETID --> MARKET(ID)

MARKET

ID	NAME	STATUS	MANAGER	NUM_TRADE

MANAGER --> USER(ID)

THANK YOU!!!