




10장. Stored Procedure

- ☐ stored procedure
- ☐ user defined function

개요

- ❑ **Stored Procedure** 란? *☞ 미리 저장된 SQL 쿼리 "프로그램"*
 - 데이터베이스 내에 **미리 컴파일되어 저장된 SQL 코드** (**프로그램**)
 - **사용자 프로그램에서** 호출, 또는 **다른 stored procedure** 에서 호출 *(한번 컴파일 후 계속 사용)*
- ❑ **장점** *캐시 이용*
 - 데이터베이스 내에서 SQL 명령을 컴파일 할 때 **캐시**를 이용할 수 있으므로 처리가 매우 빠르다.
 - 한번 컴파일 후 계속 사용* 어플리케이션마다 복수의 SQL 문을 ~~가~~할 필요없이 이미 만들어진 **저장 프로시저**를 사용하면 다른 어플리케이션을 수정하여 컴파일 할 필요가 없다. *☞ 저장 프로시저만 바뀐다면 여러번 저장할 필요 X* 
 - DBA 에러의 감소 및 보안 메커니즘의 적용
- 매번 체크하고, 에러 발생 X (한번 아예 계속 아)* **저장 프로시저**는 항상 만들어지는 순간에 구문이 검사되므로 DBA의 에러를 감소 시킬 수 있고 (**실행전** 검사) *SQL은 실행 X 인 사람이*
- (한번 아예 계속 아)* SQL 문을 직접 실행시킬 수 없는 사용자들도 **저장 프로시저만 실행**시킬 수 있는 **권한**을 가질 수 있도록 할 수 있다. *SQL X 여는 저장 프로시저 실행 권한*

개요

□ Stored Procedure의 단점

- 접하기가 어려움 (배워야 함)
- (JSP, ASP, PHP)에서 호출 방식이 틀려 처음 저장 프로시저 사용하는 경우에는 거부감이 생김
- 앞의 두 경우는 새로운 언어를 배우는 정도의 부담에서 끝나지만 저장 프로시저를 너무 남발하는 경우에는 프로젝트 관리나 유지 보수가 어렵다



- DBMS 제품마다 문법이 다르다 (비표준화)

참조 : <https://recoveryman.tistory.com/186>

개요

DBMS



작업
과정

□ Stored Procedure 를 사용하는 이유

	일반 SQL	저장 프로시저
<div>계속 반복</div> <div>Table 등 확인</div> <p>만들 때</p>	<p>1. 키워드 분리 및 문법 검사</p> <p>2. 각 개체의 이름을 확인</p> <p>3. 권한 및 보안의 점검</p> <p>4. 옵티마이징 (최적화시킨다)</p> <p>5. 컴파일의 실행</p>	<p>1. 키워드 분리 및 문법 검사</p> <p>2. 각 개체의 이름을 확인</p> <p>3. 권한 및 보안의 점검</p> <p>4. 옵티마이징 (최적화시킨다)</p> <p>5. 결과를 서버에 저장</p>
처음 실행 시	컴파일된 것을 실행한다	<p>컴파일하고, 이를 어떻게 실행할 것인지에 대한 실행계획을 만들어 캐쉬에 저장하고 난 뒤 실행한다.</p>
이후 실행 시	위의 두 과정을 다시 또 수행	<p>캐쉬를 확인해서 이미 실행계획이 있다면 그를 사용하며, 캐쉬에 실행계획이 없다면 위의 과정을 수행한다.</p>

Stored Procedure 만들기

□ (1) SQL 명령어로 만들기

무조건 이 안에
넣어야
함!!

```
DELIMITER //
```

```
CREATE PROCEDURE p_emp_sel (id int)
```

```
BEGIN
```

```
select * from emp  
where empno = id ;
```

```
END
```

```
//
```

```
DELIMITER ;
```

이름
배경변수
이름 타입

SCHEMAS

Filter objects

- ▶ bonus
- ▶ customer
- ▶ dept
- ▶ dummy
- ▶ emp
- ▶ item
- ▶ middle
- ▶ mydept
- ▶ myemp
- ▶ ord
- ▶ price
- ▶ product
- ▶ salgrade

- ▶ Views
- ▼ Stored Procedures
 - ▶ p_emp_sel
- ▶ Functions

□ Stored Procedure 실행

```
call p_emp_sel (7566) ;
```

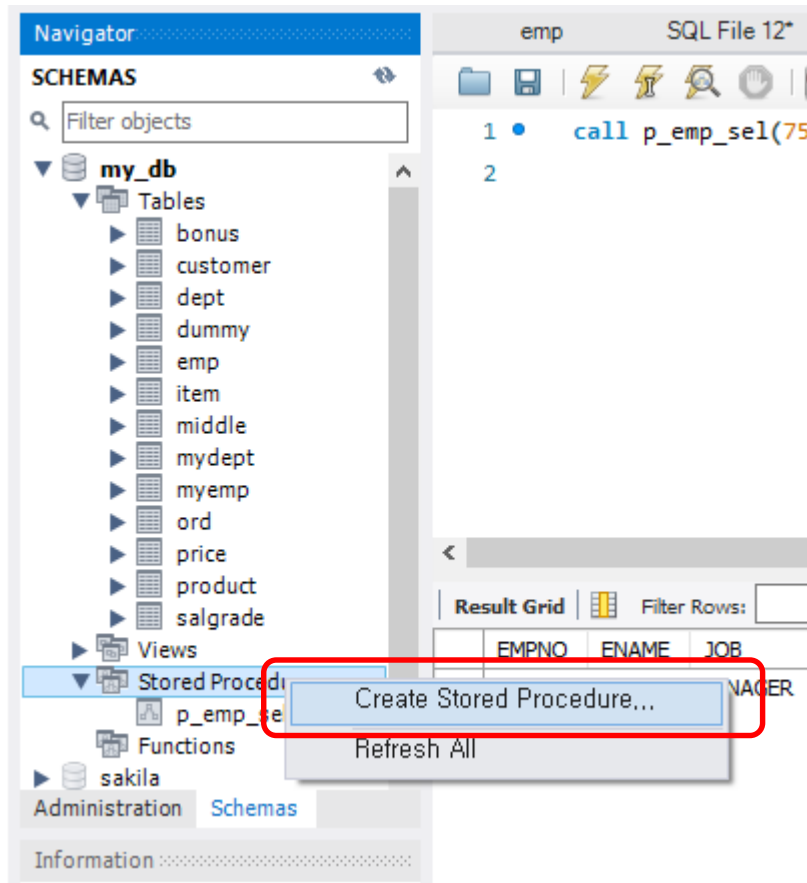
10 • `call p_emp_sel(7566) ;`

11

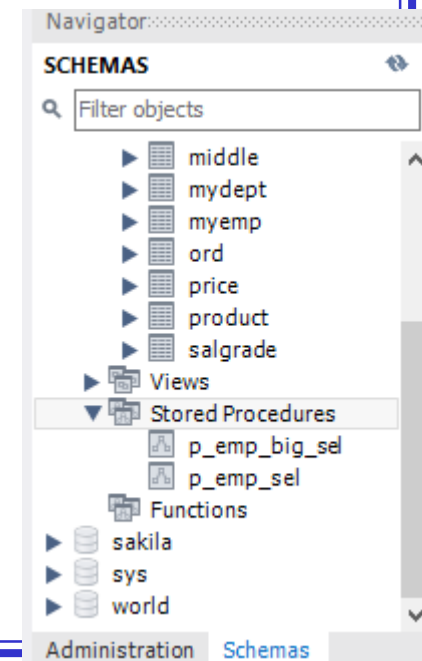
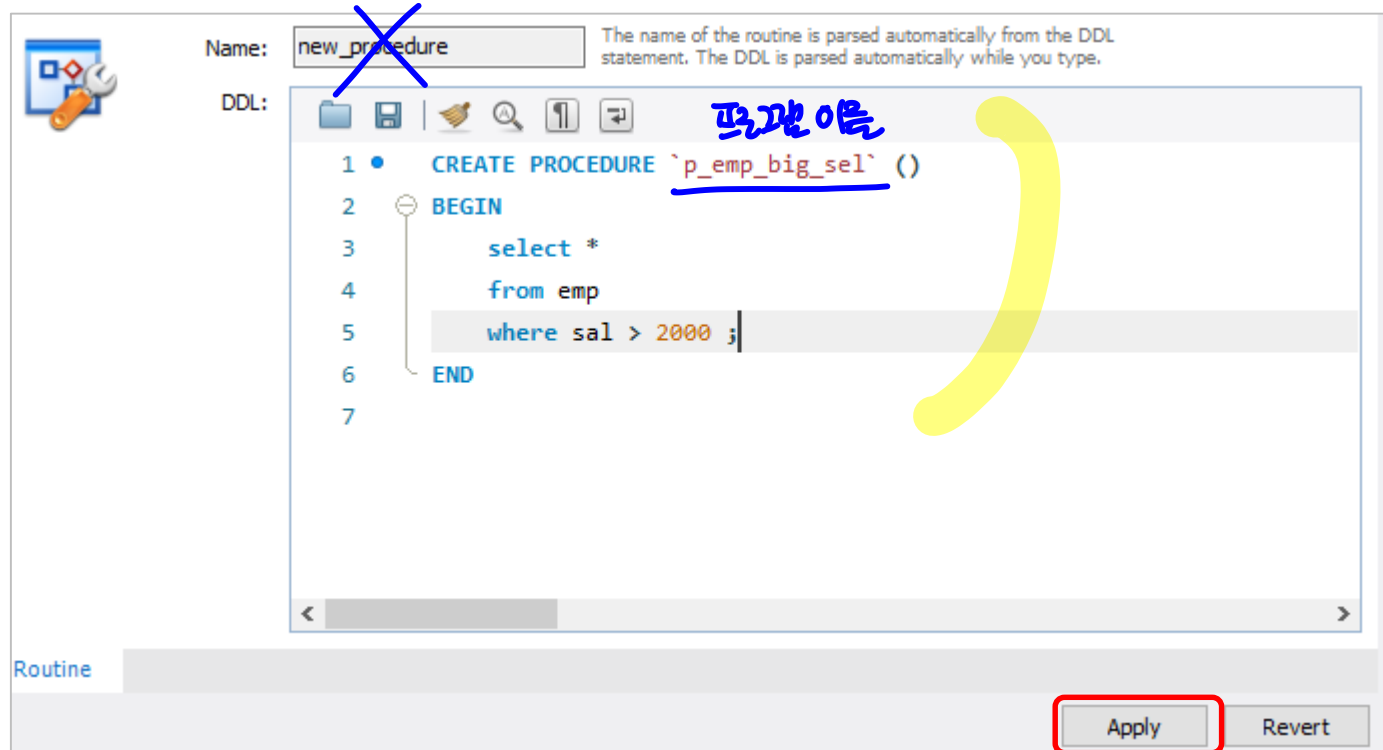
Result Grid								
		Filter Rows:		Export:		Wrap Cell Content:		
	EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
▶	7566	JONES	MANAGER	7839	1981-04-02	2975.00	NULL	20

Stored Procedure 만들기

❑ (2) Mysql Workbench 이용하기



Stored Procedure 만들기



Stored Procedure 만들기

❑ (2) Mysql Workbench 이용하기

The screenshot shows the MySQL Workbench interface. The top toolbar includes icons for file operations, execution, and viewing. The SQL editor contains the query: `1 call p_emp_big_sel ;`. A red arrow points from the text below to the semicolon at the end of the query. Below the SQL editor, the 'Result Grid' tab is active, displaying a table of employee data. A red arrow points from the text below to the 'MGR' column in the table. The table has columns: EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, and DEPTNO. The data rows are as follows:

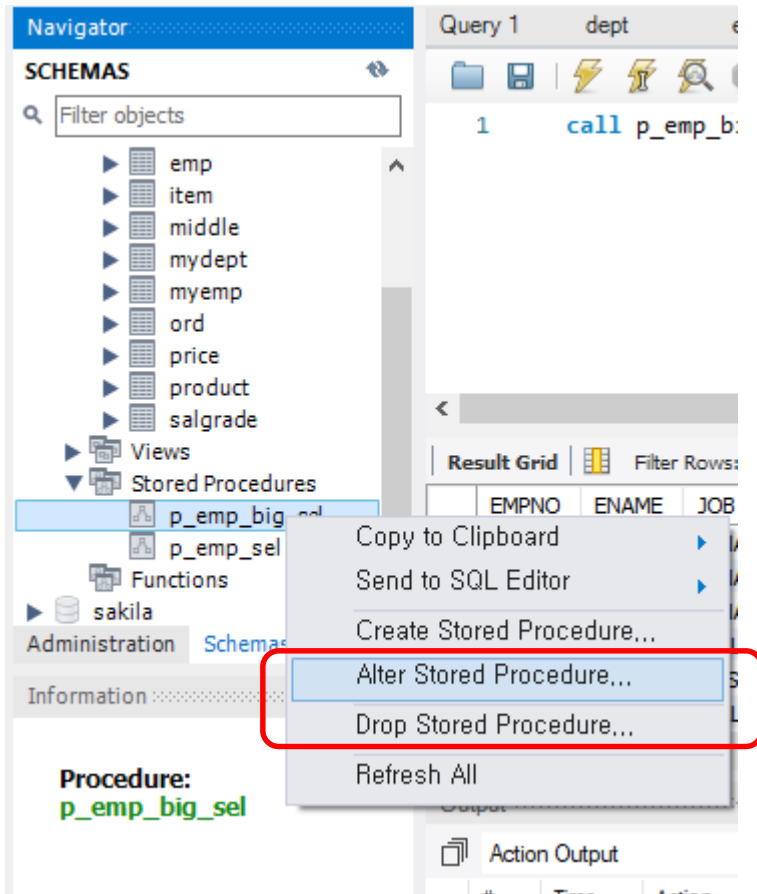
EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7566	JONES	MANAGER	7839	1981-04-02	2975.00	NULL	20
7698	BLAKE	MANAGER	7839	1981-05-01	2850.00	NULL	30
7782	CLARK	MANAGER	7839	1981-06-09	2450.00	NULL	10
7788	SCOTT	ANALYST	7566	1982-12-09	3000.00	NULL	20
7839	KING	PRESIDENT	NULL	1981-11-17	5000.00	NULL	10
7902	FORD	ANALYST	7566	1981-12-03	3000.00	NULL	20

매개변수가 없는 경우는 () 없이 호출해도 된다.

실행 결과

Stored Procedure 만들기

❑ 만들어진 Stored Procedure 의 수정, 삭제

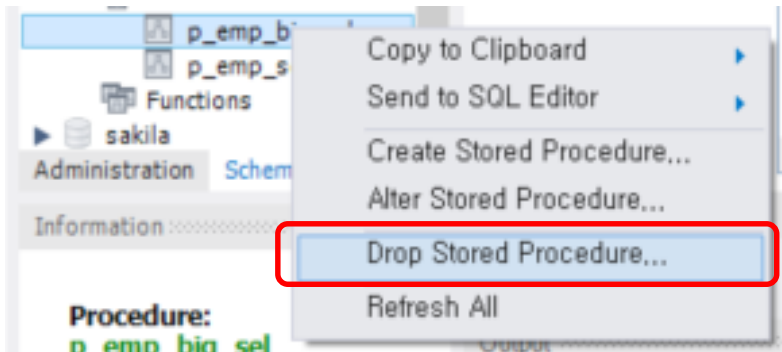


Stored Procedure 삭제하기

❑ SQL 명령문 이용

```
DROP PROCEDURE 저장_프로시저_이름 ;
```

❑ Mysql workbench 이용



[실습과제 1]

empho

1. 사원번호를 매개변수로 입력 받아 사원번호, 이름, 담당업무, 연봉, 소속 부서명을 보여주는 stored procedure 를 작성하시오 (p_emp_sel_1)
2. 부서번호, 부서명, 위치를 매개변수로 입력 받아 새로운 부서 정보를 생성하는 stored procedure 를 작성하시오 (p_dept_insert_1)
3. 사원번호, 사원 이름을 매개변수로 입력 받아 이름을 수정하는 stored procedure 를 작성하시오 (p_emp_update_1)

delimiter //

Create procedure p_emp_sel_1 (no int)

(start select ename, job, sal, dname from emp e, dept d

end where emp

// delimiter ;

where e.deptno = d.deptno and empno = no;

문제별로 Stored procedure 의 내용과 Stored procedure 의 테스트 실행 화면을 파일에 저장하여 제출하시오

주석문

- ❑ Stored procedure 에서는 두가지 형태의 주석문 사용 가능



Name: p_emp_big_sel

The name of the routine is parsed automatically from the DDL statement. The DDL is parsed automatically while you type.

DDL:

```
1 • CREATE DEFINER='root'@'localhost' PROCEDURE `p_emp_big_sel`()
2 BEGIN
3 ① # 주석문은 이렇게 넣는다
4 ② -- 이렇게 넣어도 된다
5      select *
6      from emp
7      where sal > 2000 ;
8 END
```

변수의 선언, if문

- 사원번호, 급여를 입력받고, 급여액이 최고 급여액 이하이면 해당사원의 급여를 입력한 급여액으로 갱신한다.

```
CREATE PROCEDURE up_emp_sel(  
    id int,  
    new_sal decimal(7,2)  
BEGIN  
    DECLARE max_sal decimal(7,2) ;  
    (select max(sal) into max_sal  
    from emp ;  
  
    IF (new_sal < max_sal) THEN  
        update emp  
        set sal = new_sal  
        where empid = id ;  
    END IF ;  
END
```

Handwritten annotations in the image:

- Red text above `new_sal decimal(7,2)`: "급여액", "중간값", "최대값", "변수"
- Green text above `DECLARE`: "변수선언"
- Blue text above `DECLARE`: "변수선언"
- Green text above `max_sal`: "데이터 타입"
- Blue text above `max_sal`: "변수선언"
- Green text above `update`: "갱신"
- Green text above `set`: "변수선언"
- Green text above `where`: "조건문"
- Green text above `END IF`: "if문"

Programming Structure

- ❑ Stored Procedure 안에 우리가 알고 있는 프로그래밍 문법의 사용이 가능하다
 - Operators
 - Looping structures
 - Control-of-flow statements
 - SQL 문

[실습과제 2]

1. 사원번호를 매개변수로 입력 받아 사원의 담당업무가 'CLERK' 이면 급여를 20% 올리고, 아닌 경우는 10%를 올리는 stored procedure 를 작성하시오 (p_emp_update_2)
2. 사원번호를 매개변수로 입력 받은 후에 그 사원이 속한 부서 사람들의 연봉 합계를 구하여 출력하는 stored procedure 를 작성하시오 (p_emp_sel_2)
3. 사원번호를 매개변수로 입력 받은 후에 사원의 급여가 평균급여 이상 이면 해당 사원의 근무지를 보이고, 그렇지 않으면 사원의 직무를 보이는 stored procedure 를 작성 하시오. (p_emp_sel_3)

문제별로 Stored procedure 의 내용과 Stored procedure 의 테스트 실행 화면을 파일에 저장하여 제출하시오

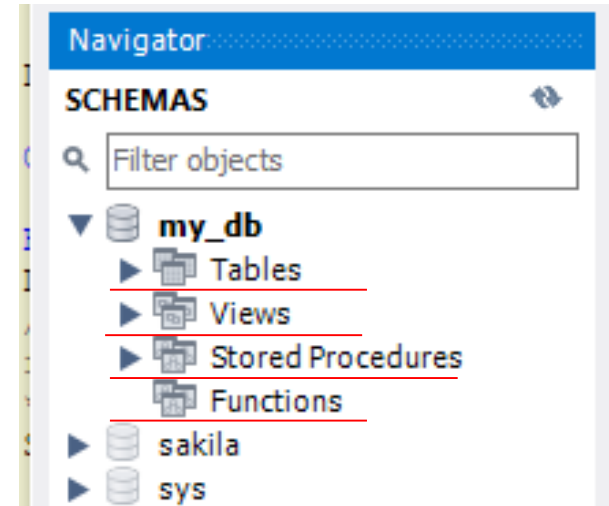
User Defined Function

❑ Stored procedure

- 일반적으로 ~~return~~ 값이 없는 프로그램
- CALL 에 의해서 호출

❑ Function

- return 값이 있는 프로그램
- Max(), min() 과 같이 SQL 문 안에서 사용됨



User Defined Function

- ❑ 함수를 생성하기 전에 먼저 다음과 같이 시스템 환경변수 값을 변경한다

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

사용자 정의 함수를 저장하는 것이 가능하도록 함

User Defined Function

가장 쉬운

```
CREATE FUNCTION f_grade(value decimal(8,2))  
RETURNS vvarchar(40)  
BEGIN  
    DECLARE grade vvarchar(40) ;  
    SET grade = 'Low salary' ;  
    IF (value > 2500) THEN  
        SET grade = 'High salary' ;  
    END IF ;  
    RETURN grade ;  
END
```

이름
배경변수
이름
가장 쉬운
변환의 타입

```
select ename, sal, f_grade(sal)  
from emp ;
```

1000 이하 low high

User Defined Function

The screenshot displays a database management tool interface. On the left, the 'Navigator' pane shows the 'my_db' schema with various objects. The 'Functions' folder is expanded, and the function 'f() f_grade' is highlighted with a red rectangle. The main editor shows a SQL query in the 'p_emp_big_sel - Routine' tab:

```
1 • select ename, sal, f_grade(sal)
2   from emp ;
3
```

Below the query editor, the 'Result Grid' shows the output of the query. A red rectangle highlights the 'f_grade(sal)' column, which contains the results of the UDF. The results are as follows:

ename	sal	f_grade(sal)
ALLEN	1600.00	Low salary
WARD	1250.00	Low salary
JONES	2975.00	High salary
MARTIN	1250.00	Low salary
BLAKE	2850.00	High salary
CLARK	2450.00	Low salary
SCOTT	3000.00	High salary
KING	5000.00	High salary
TURNER	1500.00	Low salary
ADAMS	1100.00	Low salary
JAMES	950.00	Low salary
FORD	3000.00	High salary
MILLER	1300.00	Low salary

[실습과제 3]

(1) 사원이름을 입력하면 사원의 매니저 이름을 리턴하는 함수를 작성하시오 (f_mgr)

다음과 같이 함수를 테스트한다.

```
select empno, ename, f_mgr(ename) as manager  
from emp
```

(2) 부서번호를 입력하면 부서의 위치를 출력하는 함수를 작성하시오 (f_loc)

다음과 같이 함수를 테스트한다.

```
select empno, ename, job, f_loc(deptno) as loc  
from emp
```

Handwritten notes in green ink:

- 4×2
- x^2
- 6^0
- 2^0
- 2^1 (circled)
- 2^2
- 2^3