



저장 객체

컴퓨터과학과 정재화

학습목차

— ① 저장 프로시저

— ② 함수

— ③ 트리거



01

저장 프로시저

- 데이터베이스 언어의 특징
- 저장 객체의 이해와 종류
- 저장 프로시저의 이해
- 저장 객체의 활용



데이터베이스 언어의 특징

▷ SQL

- + DBMS에 대한 강력한 작업 지시 기능을 제공
- + 인간의 언어와 매우 유사하고 간단, 명료
- + 비절차적(선언형) 언어, 필요한 데이터만 기술

▷ 비절차적 언어(non-procedural language)

- + 목표만 명시하고 수행에 대한 절차는 기술하지 않음
- + 높은 가독성과 동작 순서에 대한 구체적 기술이 없어 오류가 상대적으로 적은 점이 장점
- + 프로그램의 성능 최적화, 디버깅, 오류 추적 및 복잡한 로직 구현이 한계


저장 객체를 통해 SQL의 단점을 보완

저장 객체의 이해

▶ 저장 객체의 정의

- + SQL 문을 확장하여 절차적으로 처리하기 위한 기능을 제공하는 언어
- + SQL/PSM(Persistent Stored Module) 기반의 확장 언어

▶ 저장 객체의 구조

 CREATE PROCEDURE do_repeat (param1 INT)

```
1 BEGIN
2   SET @x = 0;
3   REPEAT
4     SET @x = @x+1;
5   UNTIL @x > param1
6   END REPEAT;
7 END;
```

형식 정의 부분

기능 정의 부분



저장 객체의 장단점

▷ 저장 객체의 장점

- + 네트워크 전송 효율: 클라이언트와 DBMS 사이에 트래픽 감소
- + 효율적 실행 속도: 저장 객체는 컴파일되어 캐시에 저장
- + 모듈화: 대규모 작업의 시간 단축, 프로그램 크기의 축소
- + 보안성 향상: 사용자별, 작업 단위별로 권한 부여 가능

▷ 저장 객체의 단점

- + 처리 성능 저하: 서버 메모리 사용량이 증가로 인한 성능 저하
- + 어려운 디버깅: 저장 객체를 위한 디버깅 환경 미지원



저장 객체의 종류

▷ 저장 프로시저

- ⊕ 자주 사용되거나 복잡한 과정을 거치는 SQL문을 저장하여 하나의 개체로 관리

▷ 함수

- ⊕ 저장 프로시저의 유사, 결과값을 반환하는 기능 제공
- ⊕ SELECT 문에 포함되어 실행

▷ 트리거

- ⊕ 데이터 갱신 시 지정된 애플리케이션이 동작 또는 다른 테이블의 데이터를 변경



저장 프로시저의 이해

▷ 저장 프로시저의 정의

- + 응용 작업 중 자주 사용되거나 다수의 SQL 문으로 구성되는 복잡한 과정이 저장된 개체
- + 데이터를 주고받는 여러 개의 SQL 문을 묶어서 단계별로 실행

▷ 저장 프로시저의 특징

- + 함수와는 달리 이름을 이용하여 값을 반환하지 않음
- + 매개변수를 받아들이거나 반환
- + 프로시저 내에서 또 다른 프로시저 호출 가능
- + 실행과 관련된 제어값과 메시지를 반환

☆ 저장 프로시저의 생성

▷ 생성 구문형식



```
CREATE [DEFINER {사용자 | 현재사용자} ]  
  PROCEDURE 프로시저명 ( [매개변수[, ...]] )  
BEGIN  
  SQL문  
  ⋮  
END
```

▷ 매개변수 정의



```
[ IN | OUT | INOUT ] 매개변수명 데이터 타입
```

☆ 저장 프로시저의 활용 1

▷ 저장 프로시저 생성

```
1 DELIMITER $$  
2 CREATE PROCEDURE GetStudentByGender( )  
3  
4 BEGIN  
5     SELECT 성별, COUNT(*) FROM 학생  
6     GROUP BY 성별;  
7 END $$  
8 DELIMITER ;
```

▷ 문장 구분자

- + DELIMITER는 개별 SQL 문 구분하는 구분자 정의
- + BEGIN-END 블록 내에 SQL 문과 기능 정의 부분의 구분자 구별 필요

☆ 저장 프로시저의 호출

▷ 호출 구문형식



```
CALL 저장 프로시저명([매개변수[, ...]]);  
또는  
CALL 저장 프로시저명[()];
```

▷ 호출 예



```
CALL GetStudentByGender;
```

학생번호	학생이름	성별	생년월일	나이	전화번호
201831-331215	김마리아	여	1991-06-18	29	010-0000-0002
201834-021216	유관순	여	1902-12-16	118	010-0000-0001
201926-880215	지청천	여	1988-02-15	32	010-0000-0005
:	:	:	:	:	:



성별	COUNT(*)
여	5
남	7

✨ 저장 프로시저의 수정과 삭제

▷ 저장 프로시저의 수정

- + ALTER PROCEDURE 명령을 사용
- + 보안 및 동작 방식에 대한 속성을 변경할 때만 사용
- + 삭제 후 재생성 → 모든 저장 객체 동일

▷ 저장 프로시저의 삭제



DROP PROCEDURE 프로시저명;



DROP PROCEDURE GetStudentByGender;



매개변수의 사용

▷ 매개변수의 역할

- + 외부로부터 주어지는 값이나 기본값을 매개변수를 통해 저장 프로시저로 전달
- + 저장 프로시저를 실행한 결과값을 반환

▷ IN, OUT, INOUT 세 타입의 매개변수를 제공

- + IN: 기본 타입, 외부로부터 값을 전달
- + OUT: 반환값을 저장
- + INOUT: IN과 OUT의 목적으로 모두 사용

매개변수의 활용 - IN

```
1 DELIMITER $$
2 CREATE PROCEDURE GetBalancebyNameBirth(
3     IN name VARCHAR(20), IN birth DATE)
4 BEGIN
5     SELECT 잔액 FROM 학생, 계좌
6     WHERE 학생.학생번호 = 계좌.학생번호 AND
7           학생.학생이름 = name AND 학생.생년월일 = birth;
8 END $$
9 DELIMITER ;
```

```
CALL GetBalancebyNameBirth('안중근', '1979-09-02')
```

잔액
100000

★ 매개변수의 활용 - OUT

```
1 DELIMITER $$
2 CREATE PROCEDURE GetPhoneByStudentID(
3     IN sid CHAR(13),
4     OUT phone CHAR(15))
5 BEGIN
6     SELECT 전화번호 INTO phone FROM 학생
7     WHERE 학생번호 = sid;
8 END $$
9 DELIMITER ;
```

```
CALL GetPhoneByStudentID('202034-596541', @phone);
SELECT @phone;
```

@phone

010-0000-0008

변수의 사용

- ◇ 저장 객체에서 처리하는 응용 과정에서 발생하는 임시적인 값을 저장하고 재사용하는 기능 제공
 - + 사용자 정의 변수: @ 시작, 별도의 선언 없이 사용
 - + 로컬 변수: 저장 객체내에서만 사용, DECLARE로 선언
- ◇ 구문형식



DECLARE 변수명[, ...] 데이터 타입[(크기)] [DEFAULT 기본값];

- + 데이터 타입(크기): 변수가 가질 수 있는 데이터 타입과 크기를 지정
- + DEFAULT 기본값: 변수가 선언될 때 초깃값을 지정

변수값 할당

▷ 선언된 변수에 값을 저장(할당)하는 방법

+ SET 명령

```
DECLARE total_count INT DEFAULT 0;  
SET total_count = 10;
```

+ SELECT INTO 문

- 숫자 타입만 가능

```
DECLARE total_student INT DEFAULT 0;  
SELECT COUNT(*) INTO total_student FROM 학생;
```

선택 구조 - IF

- ◇ 주어진 조건이나 식의 값에 따라 SQL문의 실행 여부를 제어
- ◇ 구문형식



```
IF 조건식 THEN
    SQL문
[ELSEIF 조건식 THEN
    SQL문]
[ELSE
    SQL문]
END IF;
```



IF 문의 활용

```
1 DELIMITER $$
2 CREATE PROCEDURE GetGradeByCredit(
3     IN sID CHAR(13), -- 학생번호
4     OUT nGrade TINYINT) -- 학년
5 BEGIN
6     DECLARE nTotalCredit SMALLINT;    -- 총 이수학점
7     SELECT SUM(이수학점) INTO nTotalCredit FROM 전공
8     WHERE 학생번호 = sID;
9     IF nTotalCredit > 120 THEN
10        SET nGrade = 4;
11    ELSEIF (nTotalCredit > 80 AND nTotalCredit < 120) THEN
12        SET nGrade = 3;
13    ELSEIF (nTotalCredit > 40 AND nTotalCredit < 80) THEN
14        SET nGrade = 2;
15    ELSE
16        SET nGrade = 1;
17    END IF;
18 END$$
19 DELIMITER ;
```

IF 문의 활용

**CALL GetGradeByCredit('202026-590930', @grade);
SELECT @grade**



@grade
4

선택 구조 - CASE

▷ 다중 IF ELSE 문을 간편하게 사용하기 위한 명령어



```
CASE 변수  
  WHEN 비교변숫값1 THEN SQL문  
  [WHEN 비교변숫값2 THEN SQL문]  
  
  [ELSE SQL문] ...  
END CASE;
```




```
CASE  
  WHEN 조건식1 THEN SQL문  
  [WHEN 조건식2 THEN SQL문]  
  
  [ELSE SQL문] ...  
END CASE;
```

CASE 문의 활용

```
1 DELIMITER $$
2 CREATE PROCEDURE GetRoomSize(
3     IN sClassCode CHAR(5), -- 과목코드
4     OUT sClassSize VARCHAR(20)) -- 강의실 규모
5 BEGIN
6     DECLARE nClassVolumn INT; -- 신청자수
7     SELECT COUNT(*) INTO nClassVolumn FROM 수강
8     WHERE 과목코드 = sClassCode;
9     CASE
10        WHEN nClassVolumn > 4 THEN
11            SET sClassSize = '대강의실';
12        WHEN (nClassVolumn > 2 AND nClassVolumn < 4) THEN
13            SET sClassSize = '중강의실';
14        ELSE
15            SET sClassSize = '소강의실';
16    END CASE;
17 END$$
18 DELIMITER ;
```

CASE 문의 활용

.....



```
CALL GetRoomSize('COM12',@size);  
SELECT @size;
```



@size
대강의실

반복 구조

- ▷ MySQL은 특정 SQL문을 주어진 조건이 만족하는 동안 특정 구간을 반복시킬 수 있는 세가지 구문을 제공
 - ⊕ WHILE / REPEAT / LOOP
- ▷ 반복문 안에서 반복 작업을 제어하기 위해 LEAVE와 ITERATE 문을 사용
 - ⊕ LEAVE는 반복 종료
 - ⊕ ITERATE는 현재 반복 구간을 중단하고 조건을 재검사하여 반복 여부는 판별



WHILE 문의 활용

```
1 CREATE PROCEDURE GetFirstCourse(  
2   IN sCourseCode CHAR(5), OUT sPrerequisite CHAR(5) )  
3 BEGIN  
4   DECLARE sInputCourse CHAR(5);  
5   SET sInputCourse = sCourseCode;  
6   pre: WHILE true DO  
7     SELECT 선수과목 INTO sPrerequisite FROM 과목  
8       WHERE 과목코드 = sCourseCode;  
9     IF sPrerequisite IS NOT NULL THEN  
10      SET sCourseCode = sPrerequisite;  
11      SET sPrerequisite = NULL;  
12    ELSE  
13      LEAVE pre;  
14    END IF;  
15  END WHILE;  
16  IF sCourseCode = sInputCourse THEN  
17    SET sPrerequisite NULL;  
18  ELSE  
19    SET sPrerequisite = sCourseCode;  
20  END IF;  
21 END$$
```

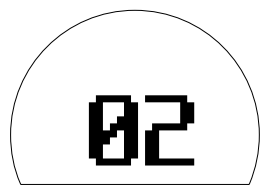
WHILE 문의 활용

```
CALL GetFirstCourse('COM31', @code);  
SELECT @code;
```



@code
COM12





함수

- 함수의 이해
- 함수의 생성과 활용

함수의 이해

- ▷ DBMS는 문자함수, 숫자함수, 날짜함수, 집계함수 등 많은 편리한 함수를 제공
- ▷ 사용자가 프로그래밍 언어의 기능을 사용하여 직접 함수를 정의하는 사용자 정의 함수 기능을 제공
- ▷ 저장 프로시저와 함수의 차이
 - ⊕ 저장 프로시저는 OUT 매개변수를 통해 결과를 반환, 함수는 RETURN문에 의해 특정 값을 반환
 - ⊕ 저장 프로시저는 CALL 명령에 의해 실행되나, 함수는 SELECT문에 포함되어 실행(호출)

함수의 생성

구문형식



```
CREATE
  [DEFINER {사용자|현재사용자} ]
  FUNCTION 함수명( [매개변수[, ...]] )
    RETURNS 데이터 타입
    [Characteristic]
BEGIN
  SQL문
END
```

- + 매개변수: IN 타입의 매개변수만 사용 가능
- + Characteristic: SQL 문의 특성을 기술
 - NO SQL/READ SQL DATA/MODIFIES SQL DATA

함수의 생성

```
1 DELIMITER $$
2 CREATE FUNCTION GetInterestRate(nBalance INT)
3 RETURNS DECIMAL(4, 2)
4 READS SQL DATA
5 BEGIN
6 DECLARE fIR DECIMAL(4, 2);
7
8 IF nBalance >= 500000 THEN
9     SET fIR = 0.03;
10 ELSEIF (nBalance >= 100000 AND nBalance < 500000) THEN
11     SET fIR = 0.02;
12 ELSEIF (nBalance < 100000) THEN
13     SET fIR = 0.01;
14 ELSE
15     SET fIR = 0.0;
16 END IF;
17 RETURN (fIR);
18 END$$
19 DELIMITER ;
```

함수의 활용

```
SELECT 학생번호, 계좌번호, 잔액, GetInterestRate(잔액) AS 이율
FROM 계좌
WHERE 잔액 >= 300000;
```

학생번호	계좌번호	잔액
201831-331215	123434-222222	800000
201834-021216	123434-111111	600000
201926-880215	123434-555555	300000
201931-781109	123434-444444	400000
201934-080621	123434-777777	300000
201934-790902	123434-666666	100000
202034-596541	123434-888888	1200000
202031-816515	123434-999999	150000
⋮	⋮	⋮

함수의 활용

```
SELECT 학생번호, 계좌번호, 잔액, GetInterestRate(잔액) AS 이율
FROM 계좌
WHERE 잔액 >= 300000;
```

학생번호	계좌번호	잔액	이율
201831-331215	123434-222222	800000	0.03
201834-021216	123434-111111	600000	0.03
201978-610408	123434-333333	400000	0.02
201931-781109	123434-444444	400000	0.02
201926-880215	123434-555555	300000	0.02
201934-080621	123434-777777	300000	0.02
202034-596541	123434-888888	1200000	0.03
⋮	⋮	⋮	

03

트리거

- 트리거의 이해
- 트리거의 종류
- 트리거의 생성과 활용

트리거의 이해

▷ 데이터 변경 시 자동으로 지정된 애플리케이션이 동작하거나 다른 테이블의 데이터를 변경하도록 설계된 저장 프로시저의 특별한 형태

▷ 트리거의 역할

- + 전체적인 데이터베이스의 무결성과 일관성을 유지하도록 동작
- + 조건에 따른 데이터 적합성 검사 가능
- + 연속적 트리거 실행 시 다른 데이터베이스나 서버에 대한 광범위한 참조 무결성 검사도 가능

트리거의 종류

▷ 트리거 이벤트와 트리거 시점으로 구분

- ⊕ 트리거 이벤트: 자동으로 반응하는 SQL 문
- ⊕ 트리거 시점: 트리거가 실행될 시점을 명시

트리거 이벤트	실행시점	기능
INSERT	BEFORE	테이블에 데이터가 입력되기 전에 실행
	AFTER	테이블에 데이터가 입력된 후에 실행
UPDATE	BEFORE	테이블의 데이터가 수정되기 전에 실행
	AFTER	테이블의 데이터가 수정된 후에 실행
DELETE	BEFORE	테이블의 데이터가 삭제되기 전에 실행
	AFTER	테이블의 데이터가 삭제된 후에 실행

트리거의 생성

▷ 구문형식



```
CREATE TRIGGER 트리거이름  
    트리거시점 트리거이벤트 ON 테이블이름  
    FOR EACH ROW  
BEGIN  
    SQL문  
END
```

▷ OLD와 NEW 키워드

- + OLD: 해당 테이블에 변경이 가해지기 전의 레코드
- + NEW: 해당 테이블에 변경이 가해진 후의 레코드

트리거의 활용

```
1 DELIMITER $$
2 CREATE TRIGGER before_과목_update
3 BEFORE UPDATE ON 과목
4 FOR EACH ROW
5
6 BEGIN
7     IF NEW.학점 < 1 THEN
8         SET NEW.학점 = 1;
9     ELSEIF NEW.학점 > 3 THEN
10        SET NEW.학점 = 3;
11    END IF;
12 END $$
13 DELIMITER ;
```

트리거의 활용

UPDATE 과목

SET 학점 = 5

WHERE 과목코드 = 'COM34';

SELECT * FROM 학생;



과목코드	과목명	학점	선수과목	이수구분	교수번호
COM11	컴퓨터의 이해	3		교양	...
COM12	파이썬 프로그래밍 기초	3		교양	...
COM24	자료구조	3	COM12	전공필수	...
COM31	데이터베이스 시스템	3	COM24	전공필수	...
COM34	알고리즘	3	COM24	일반선택	...
COM44	클라우드 컴퓨팅	3		전공필수	...
ECE24	놀이지도	3		전공필수	...
ECE31	유아언어교육	3	ECE31	전공필수	...
⋮	⋮	⋮	⋮	⋮	⋮

다음 시간



데이터 저장과 파일