# 트리거

- 자동으로 수행하여 사용자가 추가 작업을 잊어버리는 실수를 방지해줌(INSERT, UPDATE, DELETE 작업이 발생하면 실행되는 코드)

- 트리거를 사용하면 데이터에 오류가 발생하는 것을 막을 수 있음 이런 것을 데이터의 무결성이라고 부르기도 함

- 예를 들어 회원이 탈퇴하면 간단히 회원 테이블에서 탈퇴 회원의 정보를 삭제하면 됩니다. 그런데 나중에 탈퇴한 사람의 정보를 알려고 할 때 이미 데이터베이스에 존재하지 않기 때문에 알 수 있는 방법이 없습니다. 이를 해결하는 방법은 삭제 전에 내용을 다른 곳에 복사해 놓는 방법이 있는데 수작업으로 할 경우, 잊어버리는 경우가 발생할 수 있습니다 이를 방지하기 위해 데이터 변화가 일어나기 전에 다른 곳에 자동으로 저장해주는 기능이 트리거입니다.

**\* 트리거의 기본 작동**

- 테이블에서 DML(Data Manipulation Language)문 (UPDATE, INSERT, DELETE)등의 이벤트가 발생할 때 작동

- 테이블에 미리 부착되는 프로그램 코드라고 생각하면 됨

- 스토어드 프로시저와 문법이 비슷하지만, CALL 문으로 직접 실행시킬 수는 없고 오직 테이블에서 INSERT, UPDATE, DELETE등의 이벤트가 발생할 경우에만 자동으로 실행됨. 또한 IN, OUT 매개변수를 사용할 수 없음

Ex)

USE market\_db;

CREATE TABLE IF NOT EXISTS trigger\_table (id INT, txt VARCHAR(10));

INSERT INTO trigger\_table VALUES(1, '레드벨벳');

INSERT INTO trigger\_table VALUES(2, '잇지');

INSERT INTO trigger\_table VALUES(3, '블랙핑크');

DROP TRIGGER IF EXISTS myTrigger;

DELIMITER $$

CREATE TRIGGER myTrigger -- 트리거 이름

AFTER DELETE -- 삭제후에 작동하도록 지정

ON trigger\_table -- 트리거를 부착할 테이블

FOR EACH ROW -- 각 행마다 적용시킴

BEGIN

SET @msg = '가수 그룹이 삭제됨' ; -- 트리거 실행시 작동되는 코드들

END $$

DELIMITER ;

SET @msg = '';

INSERT INTO trigger\_table VALUES(4, '마마무');

SELECT @msg; // INSERT문을 실행했을 경우 트리거가 사용되었는지 확인

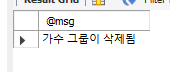
UPDATE trigger\_table SET txt = '블핑' WHERE id = 3;

SELECT @msg; // UPDATE문을 실행했을 경우 트리거가 사용되었는지 확인

==> DELETE문 트리거이기 때문에 INSERT, UPDATE문을 실행했을 경우에는 트리거가 작동하지 않음



DELETE FROM trigger\_table WHERE id = 4;

SELECT @msg; // DELETE문을 실행했을 경에는 트리거가 작동함  


# 트리거 활용

- 테이블에 입력/수정/삭제되는 정보를 백업하는 용도로 활용할 수 있음

Ex)

USE market\_db;

CREATE TABLE singer (SELECT mem\_id, mem\_name, mem\_number, addr FROM member);

-> 실제 사용 테이블

DROP TABLE IF EXISTS backup\_singer;

CREATE TABLE backup\_singer

( mem\_id CHAR(8) NOT NULL ,

mem\_name VARCHAR(10) NOT NULL,

mem\_number INT NOT NULL,

addr CHAR(2) NOT NULL,

modType CHAR(2), -- 변경된 타입. '수정' 또는 '삭제'

modDate DATE, -- 변경된 날짜

modUser VARCHAR(30) -- 변경한 사용자

); -> 백업 테이블 생성

// UPDATE 트리거 생성

DROP TRIGGER IF EXISTS singer\_updateTrg;

DELIMITER $$

CREATE TRIGGER singer\_updateTrg -- 트리거 이름

AFTER UPDATE -- 변경 후에 작동하도록 지정

ON singer -- 트리거를 부착할 테이블

FOR EACH ROW

BEGIN

INSERT INTO backup\_singer VALUES( OLD.mem\_id, OLD.mem\_name, OLD.mem\_number,

OLD.addr, '수정', CURDATE(), CURRENT\_USER() ); // OLD에 변경되기 전 데이터가 잠깐 들어가게 된다

END $$

DELIMITER ;

// DELETE 트리거 생성

DROP TRIGGER IF EXISTS singer\_deleteTrg;

DELIMITER $$

CREATE TRIGGER singer\_deleteTrg -- 트리거 이름

AFTER DELETE -- 삭제 후에 작동하도록 지정

ON singer -- 트리거를 부착할 테이블

FOR EACH ROW

BEGIN

INSERT INTO backup\_singer VALUES( OLD.mem\_id, OLD.mem\_name, OLD.mem\_number,

OLD.addr, '삭제', CURDATE(), CURRENT\_USER() );

END $$

DELIMITER ;

===> 테이블 당 트리거는 하나만 하는게 아니라 여러 개를 부착시켜도 상관없음

UPDATE singer SET addr = '영국' WHERE mem\_id = 'BLK';

DELETE FROM singer WHERE mem\_number >= 7;

SELECT \* FROM backup\_singer;

