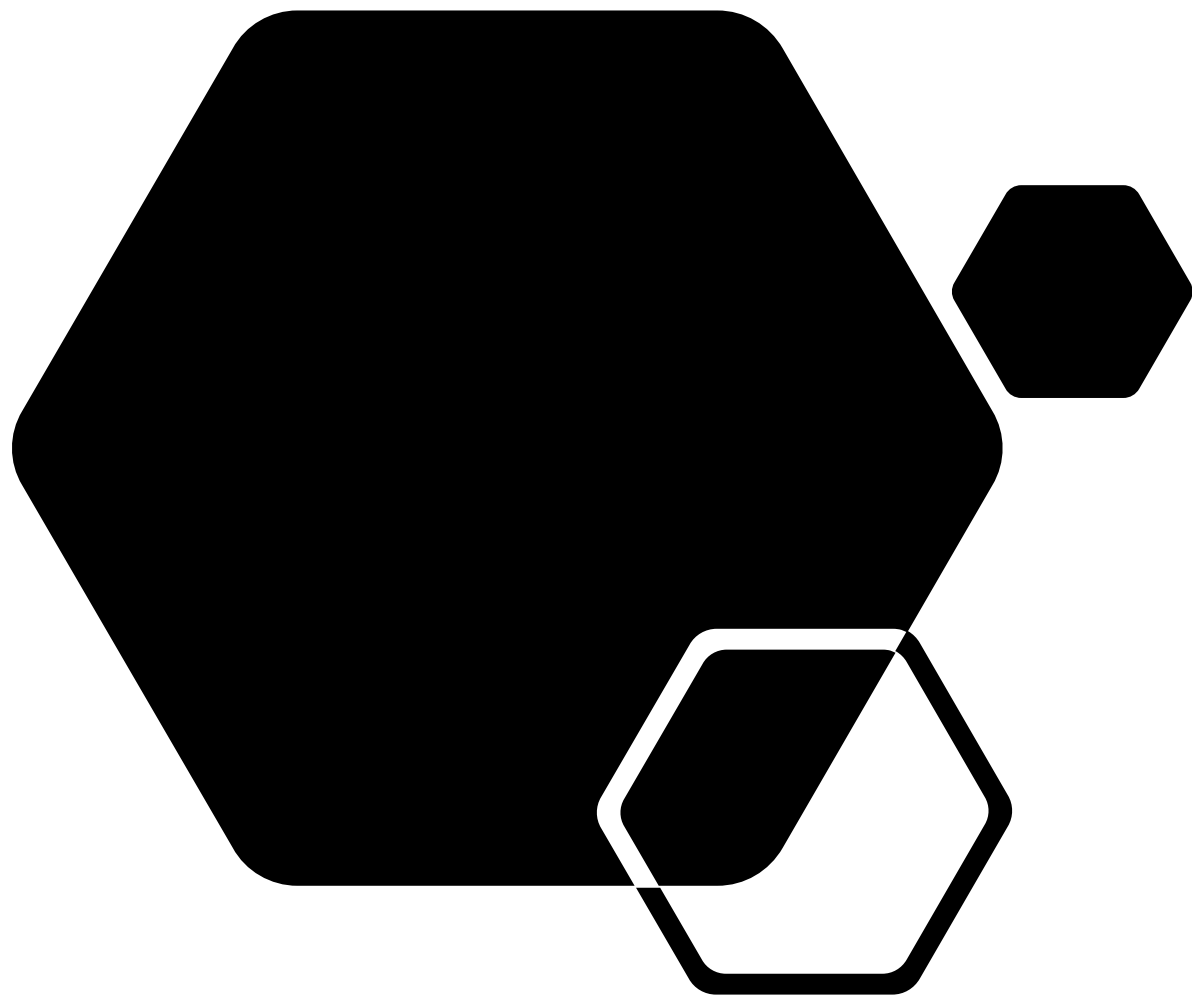
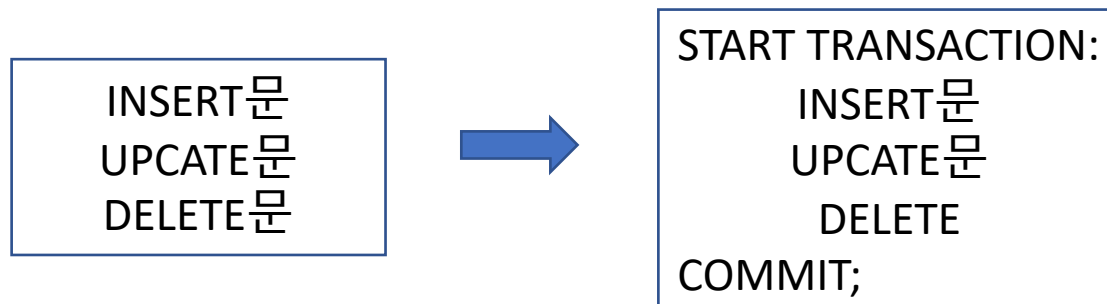


트랜잭션



- MySQL은 시스템이 안정적으로 운영되기 위해서는 트랜잭션과 락의 사용이 필수적
- 트랜잭션 명령문
  - **START TRANSACTION**(또는 **BEGIN TRANSACTION**): 새로운 트랜잭션을 시작
  - **COMMIT**: 현재 트랜잭션에서 변경한 내역을 영구적으로 시스템에 반영
  - **SAVEPOINT identifier**: identifier라는 이름의 저장 시점을 만든다. ROLLBACK 시 지정된 시점에서의 복귀를 강제로 지정
  - **ROLLBACK [to identifier]**: 현재 트랜잭션에서 변경한 내역들을 모두 취소하여 트랜잭션을 시작하기 이전 상태로 돌린다. identifier를 지정하여 특정 시점에서의 복귀가 가능.
  - **SET autocommit**: 현재 세션에 대한 'autocommit 모드'(SQL문 실행 시 변경 내역이 자동으로 commit되어 시스템에 영구적으로 반영되는 모드)를 적용하거나 해제.

- 트랜잭션 사용
  - 자동 커밋 트랜잭션
    - MySQL 서버는 기본적으로 모든 실행되는 질의에 대해 실행 후 사용자의 요청 없이 자동으로 커밋하는 기능을 사용
      - 개별적으로 실행되는 모든 INSERT, UPDATE, DELETE문은 내부적으로 자동적으로 커밋 트랜잭션 처리



- 트랜잭션 사용
  - 자동 커밋 모드를 활성화 또는 비활성화시키기
    - 자동 커밋 비활성화
      - SET autocommit = 0;
    - 자동 커밋 활성화
      - SET autocommit = 1;

- 명시적 트랜잭션

- 명시적 트랜잭션(explicit transaction)이란 자동 커밋 트랜잭션과 달리 트랜잭션의 시작 시점과 커밋 또는 롤백 시점을 사용자나 응용 프로그램이 직접 결정하는 트랜잭션

- 구문 형식

```
START TRANSACTION
    [WITH CONSISTANT SNAPSHOT]
    [READ WRITE]
    [READ ONLY]
    SQL STATEMENTS
    ...
COMMIT or ROLLBACK
```

- 트랜잭션 구문을 적용시켜 보기 위해 '정용민' 학생이 '안중근' 학생에게 500,000원을 계좌이체하는 작업을 가정



계좌이체의 단위 작업

- 계좌 테이블 생성

```
CREATE TABLE 계좌 (  
    학생이름 CHAR(13) PRIMARY KEY,  
    계좌번호 CHAR(15) NOT NULL,  
    잔액 INT NOT NULL DEFAULT 0  
);
```

```
INSERT INTO 계좌 VALUES ('정용민', '123435-333333', 450000);
```

```
INSERT INTO 계좌 VALUES ('안중근', '123434-666666', 100000);
```

## ■ 트리거 생성 전에

-- 정용민 계좌에서 안중근 계좌로 50000원 송금

UPDATE 계좌 SET 잔액 = 잔액-50000 WHERE 학생이름 = '정용민';

UPDATE 계좌 SET 잔액 = 잔액+50000 WHERE 학생이름 = '안중근';

-- 정용민 계좌에서 안중근 계좌로 500000원 송금

UPDATE 계좌 SET 잔액 = 잔액-500000 WHERE 학생이름 = '정용민';

UPDATE 계좌 SET 잔액 = 잔액+500000 WHERE 학생이름 = '안중근';

**차례대로 하나씩 수행해 보세요!!!**



- 트리거 생성

- 계좌 테이블에 잔액이 0보다 작을 수 없다는 제약조건

```
DELIMITER $$
CREATE TRIGGER balance_check_insert BEFORE INSERT ON 계좌 FOR EACH ROW
BEGIN
    IF (new.잔액 < 0)then
        SIGNAL SQLSTATE '45000'
        SET message_text='잔액은 음수가 될 수 없습니다.';
    END IF;
END $$
DELIMITER;
```

```
DELIMITER $$
CREATE TRIGGER balance_check_insert BEFORE UPDATE ON 계좌 FOR EACH ROW
BEGIN
    IF (new.잔액 < 0)then
        SIGNAL SQLSTATE '45000'
        SET message_text='잔액은 음수가 될 수 없습니다.';
    END IF;
END $$
DELIMITER;
```

## ■ 트리거 생성 후에

-- 정용민 계좌에서 안중근 계좌로 50000원 송금

UPDATE 계좌 SET 잔액 = 잔액-50000 WHERE 학생이름 = '정용민';

UPDATE 계좌 SET 잔액 = 잔액+50000 WHERE 학생이름 = '안중근';

-- 정용민 계좌에서 안중근 계좌로 500000원 송금

UPDATE 계좌 SET 잔액 = 잔액-500000 WHERE 학생이름 = '정용민';

UPDATE 계좌 SET 잔액 = 잔액+500000 WHERE 학생이름 = '안중근';

**트리거를 생성하고 차례대로 하나씩 수행해 보세요!!!**



**단일 작업**이 두 개의 독립된 데이터베이스 명령으로 수행되기 때문

## ■ 트랜잭션 수행

-- 트랜잭션으로 정용민 계좌(400000)에서 안중근 계좌(150000)로 500000원 송금

START TRANSACTION;

UPDATE 계좌 SET 잔액 = 잔액-500000 WHERE 학생이름 = '정용민';

UPDATE 계좌 SET 잔액 = 잔액+500000 WHERE 학생이름 = '안중근';

ROLLBACK;

- ROLLBACK 문을 수행하여 오류가 발생한 전체 트랜잭션을 취소
- 트랜잭션 시작 이전의 상태로 값이 복원

- 안전한 트랜잭션 수행

-- '정용민'의 잔액 400,000원과 안중근'의 잔액 150,000원인 상태에서, 정용민의 계좌에서 '안중근'의 계좌로 500,000원을 송금하시오. 단, '정용민'의 잔액이 부족할 경우, 송금을 취소하시오.

- WIRE\_TRANSFER 프로시저 생성

```
DELIMITER $$
CREATE PROCEDURE WIRE_TRANSFER(
IN sender CHAR(13),
IN recipient CHAR(15),
IN amount INT)
BEGIN
    DECLARE EXIT HANDLER FOR SQLEXCEPTION ROLLBACK: #SQL예외 이벤트 발생시 ROLLBACK문 수행
    START TRANSACTION;
    UPDATE 계좌 SET 잔액 = 잔액 - amount WHERE 학생이름 = sender;
    UPDATE 계좌 SET 잔액 = 잔액 + amount WHERE 학생이름 = recipient;
    COMMIT;
END$$
DELIMITER;
```

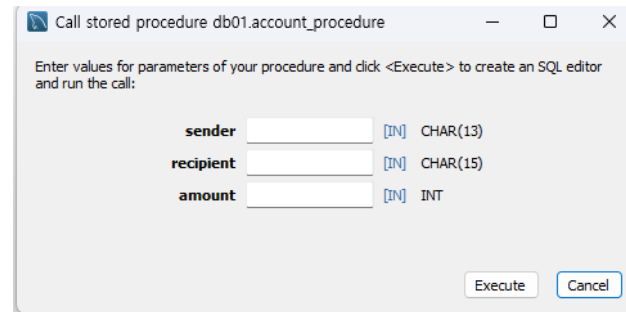
- 안전한 트랜잭션 수행

-- '정용민'의 잔액 400,000원과 안중근'의 잔액 150,000원인 상태에서, 정용민의 계좌에서 '안중근'의 계좌로 500,000원을 송금하시오.

단, '정용민'의 잔액이 부족할 경우, 송금을 취소하시오.

- CALL WIRE\_TRANSFER 프로시저 수행

CALL WIRE\_TRANSFER( ' 정용민 ' , '안중근', 100000)



Call stored procedure db01.account\_procedure

Enter values for parameters of your procedure and click <Execute> to create an SQL editor and run the call:

sender	<input type="text"/>	[IN] CHAR(13)
recipient	<input type="text"/>	[IN] CHAR(15)
amount	<input type="text"/>	[IN] INT

Execute Cancel