# 트랜잭션(Transaction) 이란?

트랜잭션(Transaction)의 사전적 의미는 거래이고,
컴퓨터 과학 분야에서의 트랜잭션(Transaction)은 "더이상 분할이 불가능한 업무처리의 단위"를 의미한다.

이것은 하나의 작업을 위해 더이상 분할될 수 없는 명령들의 모음, 즉, 한꺼번에 **수행되어야 할 일련의 연산모음**을 의미한다.

다음과 같은 상황이 있다고 가정하자.

1. A는 매달 부모님에게 생활비를 송금받는다.
   어느 날, 부모님이 A에게 생활비를 송금해 주기 위해 ATM을 이용했고 여느날처럼 A의 계좌로 생활비를 송금했다.
2. 그러나 모종의 이유로 인하여 부모님의 계좌에선 생활비가 차감되었는데, A의 계좌에는 생활비가 입금되지 않았다.

계좌이체 행위를 풀어 쓴 것이다.
보다시피 계좌이체라는 행위는 **인출**과 **입금** 두 과정으로 이루어진다.

위와 같이 만약 인출에는 성공했는데, 입금에 실패하면 치명적인 결과가 나온다.
따라서 **이 두 과정은 동시에 성공하던지 동시에 실패**해야 한다. (하나로 묶음으로써 Atomic 함을 의미한다)
이 과정을 동시에 묶는 방법이 바로 트랜잭션이다.

**sql**

```
START TRANSACTION
    -- 이 블록안의 명령어들은 마치 하나의 명령어 처럼 처리됨
    -- 성공하던지, 다 실패하던지 둘중 하나가 됨.
    A의 계좌로부터 인출;
    B의 계좌로 입금;
```

| COMMIT |
|---|

이와 같이, 데이터베이스와 어플리케이션의 **데이터 거래(Transaction)에 있어서 안전성을 확보하기 위한 방법**이 **트랜잭션**인 것이다.
따라서 데이터베이스에서 테이블의 데이터를 읽어 온 후 다른 테이블에 데이터를 입력하거나 갱신, 삭제하는 도중에 오류가 발생하면, 결과를 재반영 하는 것이 아니라 모든 작업을 원상태로 복구하고, 처리 과정이 모두 성공하였을 때만 그 결과를 반영한다.

---

## MYSQL 트랜잭션

MySQL에서 트랜잭션은 **데이터베이스를 상태를 바꾸는 일종의 작업 단위**이다.
사실 우리가 MySQL의 입력하는 모든 쿼리 명령어들은 각각 하나의 트랜잭션이라고 할 수 있다.
INSERT, DELETE, UPDATE 등의 SQL 명령문을 통해 데이터를 상태를 바꿀 때마다 내부적으로 자동적으로 Commit을 실행하여 변경된 내역을 데이터베이스의 반영하는 것이다.

| Tip<br>git의 commit과 유사하다고 볼 수 있다. |
|---|

즉 여태까지 입력한 명령어들은 MySQL에서 자동 Commit을 통해 쿼리 입력과 동시에 처리하여 데이터베이스에 반영되게 한 것이다.

그런데 착각하지 말아야 할 것은, **작업의 단위**는 **질의어 한 문장이 아니라는 점**이다.
**작업단위란 많은 질의어 명령문들을 사람이 정하는 기준에 따라 정하는 것을 의미한다.**

> Info
> 게시판을 예로 들어보자.
>
> 1) 게시판 사용자는 게시글을 작성하고, 올리기 버튼을 누른다.
> 2) 그러면 글 올리기가 처리되고 자동으로 다시 게시판에 돌아오게 된다.
> 3) 게시판에서 자신의 글이 포함된 업데이트된 게시글들을 볼 수 있다.

이러한 상황을 데이터베이스 작업으로 옮기면, 사용자가 올리기 버튼을 눌렀을 시, Insert 문을 사용하여 사용자가 입력한 게시글의 데이터를 넣는다.
그 후에, 게시판을 구성할 데이터를 다시 Select 하여 최신 정보로 유지한다.

여기서 **작업의 단위는 insert 문과 select 문 둘 다를 합친 것**이다.
이러한 작업단위를 하나의 트랜잭션이라 한다.

관리자나 개발자가 하나의 트랜잭션 설계를 잘하는 것이 데이터를 다루는 것에 많은 이점이 있다.

---

**트랜잭션 특징**

트랜잭션의 특징은 크게 4가지로 구분된다.

| | |
|---|---|
| 원자성 (Atomicity) | 원자성은 **트랜잭션이 데이터베이스에 모두 반영되던가, 아니면 전혀 반영되지 않아야 한다는 것**이다.<br><br>트랜잭션은 사람이 설계한 논리적인 작업 단위로서, 일 처리는 작업단위 별로 이루어 져야 사람이 다루는데 무리가 없다.<br><br>만약 트랜잭션 단위로 데이터가 처리되지 않는다면, 설계한 사람은 데이터 처리 시스템을 이해하기 힘들 뿐만 아니라, 오작동 했을 시 원인을 찾기가 매우 힘들 |

| | |
|---|---|
| | 어질것이다. |
| 일관성<br>(Consistency) | 일관성은 **트랜잭션의 작업 처리 결과가 항상 일관성이 있어야 한다는 것**이다.<br><br>트랜잭션이 진행되는 동안에 데이터베이스가 변경 되더라도<br>**업데이트된 데이터베이스로 트랜잭션이 진행되는것이 아니라**, **처음에 트랜잭션을 진행하기 위해 참조한 데이터베이스로 진행**된다.<br><br>이렇게 함으로써 각 사용자는 일관성 있는 데이터를 볼 수 있는 것이다. |
| 독립성<br>(Isolation) | 독립성은 둘 이상의 트랜잭션이 동시에 실행되고 있을 경우<br>**어떤 하나의 트랜잭션이라도, 다른 트랜잭션의 연산에 끼어들 수 없다는 점을 가리킨다.**<br><br>하나의 특정 트랜잭션이 완료될 때까지, 다른 트랜잭션이 특정 트랜잭션의 결과를 참조할 수 없다. |
| 영구성<br>Durability) | 지속성은 트랜잭션이 성공적으로 완료됐을 경우, **결과는 영구적으로 반영되어야 한다는 점**이다. |

---

## 트랜잭션 상태

트랜잭션의 개념은 서술한 바와 같이 데이터베이스의 상태를 변환시키는 하나의 논리적 기능을 수행하기 위한 작업의 단위 혹은 데이터베이스 시스템에서 복구 및 병행 수행 시 처리되는 작업의 논리적 단위이다.


트랜잭션의 연산과정의 단계를 도식화하고 각각의 단계의 상태를 그림과 같이 정리할 수 있는데, 개별 상태에 대한 상세는 다음과 같다.

**Info**

**1. 활성(Active)**: 트랜잭션이 정상적으로 실행 중인 상태를 의미한다.
트랜잭션이 시작되면, 해당 트랜잭션의 상태는 활동(Active)상태가 된다.
해당 상태는 설계자가 설계한 대로 연산들이 정상적으로 실행 중인 상태를 의미한다.

**- 작업 성공시,**

**2-1. 부분 완료(Partially Committed)**: 트랜잭션의 마지막까지 실행되었지만, Commit 연산이 실행되기 직전의 상태
**2-2. 완료(Committed)**: 트랜잭션이 성공이 종료되어 Commit 연산을 실행한 후의 상태

설계된 트랜잭션대로 명령을 성공적으로 수행하면 그 다음 상태는 부분적 완료(Partially Committed)상태가 된다.
설계된 작업대로 작업이 성공하였다고 하여 무조건 반영하는 것이 아니라, 설계자의 최종 승인(Commit)이 있을 때까지 실제 데이터베이스에 작업 내용을 반영하지 않고 기다리고 있는 상태이다.
설계자가 작업 결과에 대하여 반영을 승인(Commit)한다면 트랜잭션이 성공적으로 종료된다(Committed)

**- 작업 실패시,**

**2-1. 실패(Failed)**: 트랜잭션 실행에 오류가 발생하여 중단된 상태.
**2-2. 철회(Aborted)**: 트랜잭션이 비정상적으로 종료되어 Rollback 연산을 수행한 상태.

트랜잭션을 수행하는 중간에 모종의 원인으로 인하여 오류가 발생하여 실행이 중단된 상태를 실패(Failed)상태라고 한다.
이때 트랜잭션이 비정상적으로 종료되었으니, 설계되어있는 트랜잭션 내부의 작업을 다시 수행 이전의 상태로 돌리는 (ROLLBACK) 연산을 수행하면 그 상태를 철회(Aborted)라고 한다.

Tip

**Commit**

Commit이란, 모든 작업들을 정상 처리하겠다고 확정하는 명령어로서, 해당 처리 **과정을 DB에 영구 저장**하겠다는 의미이며, Commit을 수행하면 **하나의 트랜잭션 과정이 종료**되는 것이다.

Commit을 수행하면 이전 데이터가 완전히 반영되어 UPDATE된다.

Tip

**Roll-back**

Roll-back은 작업 중 문제가 발생되어 트랜잭션의 처리 과정에서 **발생한 변경사항을 취소**하는 명령어이다.

해당 명령을 트랜잭션에게 하달하면, 트랜잭션은 **시작되기 이전의 상태로 되돌아간다**.

이것은 마지막 Commit을 완료한 시점으로 돌아간다는 말과 상통한다.

즉, Rollback은 **Commit하여 저장한 예전 상태를 복구**하는 것이다

---

**트랜잭션 문법**

명령어들은 MySQL에서 자동 Commit을 통해 쿼리 입력과 동시에 처리하여 데이터베이스에 반영된다.

그렇다면 **Commit을 수동으로 바꿔서 한 트랜잭션 안에 여러 개의 명령문을 넣게 커스텀**해보자

**sql**

```
START TRANSACTION; -- 트랜잭션 시작

select * from members; -- 초기 상태 보여줌
insert into members values(1, '쿠', '크다스', '크라운제과' '?', '대한민국');
-- 데이터 수정
select * from members; -- 수정 상태 보여줌
```

COMMIT -- 트랜잭션을 DB에 적용

select * from members; -- 적용된 결과 조회

```
mysql> select * from members;
+----+----------+-----------+---------+------+----------+
| ID | LastName | FirstName | Address | City | Country  |
+----+----------+-----------+---------+------+----------+
|  2 | 안       | 주중      | 주소1   | 부산 | 대한민국 |
|  3 | 고       | 자원      | 주소2   | 서울 | 대한민국 |
|  4 | 지       | 원        | 주소3   | 속초 | 대한민국 |
+----+----------+-----------+---------+------+----------+
3 rows in set (0.00 sec)

mysql> insert into members values(1,'쿠','크다스','크라운제과','?', '대한민국');
Query OK, 1 row affected (0.00 sec)

mysql> select * from members;
+----+----------+-----------+------------+------+----------+
| ID | LastName | FirstName | Address    | City | Country  |
+----+----------+-----------+------------+------+----------+
|  1 | 쿠       | 크다스    | 크라운제과 | ?    | 대한민국 |
|  2 | 안       | 주중      | 주소1      | 부산 | 대한민국 |
|  3 | 고       | 자원      | 주소2      | 서울 | 대한민국 |
|  4 | 지       | 원        | 주소3      | 속초 | 대한민국 |
+----+----------+-----------+------------+------+----------+
4 rows in set (0.00 sec)
```

```
mysql> commit;
Query OK, 0 rows affected (0.01 sec)

mysql> select * from members;
+----+----------+-----------+------------+------+----------+
| ID | LastName | FirstName | Address    | City | Country  |
+----+----------+-----------+------------+------+----------+
|  1 | 쿠       | 크다스    | 크라운제과 | ?    | 대한민국 |
|  2 | 안       | 주중      | 주소1      | 부산 | 대한민국 |
|  3 | 고       | 자원      | 주소2      | 서울 | 대한민국 |
|  4 | 지       | 원        | 주소3      | 속초 | 대한민국 |
+----+----------+-----------+------------+------+----------+
4 rows in set (0.00 sec)
```

---

다음은 **롤백**과정이다.

테이블에 데이터가 넣어졌음에도 rollback 한순간 예전 상태로 되돌아가는 걸 볼 수 있다.

**sql**

```
START TRANSACTION; -- 트랜잭션 시작

insert into members values(5, '쿠', '크다스 동생', '크라운제과', '?', '대한민국'); -- 데이터 수정
select * from members; -- 수정 상태 보여줌

ROLLBACK -- 트랜잭션을 취소하고 START TRANSACTION 실행 전 상태로 롤백함

select * from members; -- 조회
```

```
mysql> START TRANSACTION;
Query OK, 0 rows affected (0.00 sec)

mysql> insert into members values(5,'쿠','크다스 동생','크라운제과','?', '대한민국');
Query OK, 1 row affected (0.00 sec)

mysql> select * from members;
+----+----------+-------------+------------+------+----------+
| ID | LastName | FirstName   | Address    | City | Country  |
+----+----------+-------------+------------+------+----------+
|  1 | 쿠       | 크다스      | 크라운제과 | ?    | 대한민국 |
|  2 | 안       | 주중        | 주소1      | 부산 | 대한민국 |
|  3 | 고       | 자원        | 주소2      | 서울 | 대한민국 |
|  4 | 지       | 원          | 주소3      | 속초 | 대한민국 |
|  5 | 쿠       | 크다스 동생 | 크라운제과 | ?    | 대한민국 |
+----+----------+-------------+------------+------+----------+
5 rows in set (0.00 sec)

mysql> ROLLBACK;
Query OK, 0 rows affected (0.01 sec)

mysql> select * from members;
+----+----------+-----------+------------+------+----------+
| ID | LastName | FirstName | Address    | City | Country  |
+----+----------+-----------+------------+------+----------+
|  1 | 쿠       | 크다스    | 크라운제과 | ?    | 대한민국 |
|  2 | 안       | 주중      | 주소1      | 부산 | 대한민국 |
|  3 | 고       | 자원      | 주소2      | 서울 | 대한민국 |
|  4 | 지       | 원        | 주소3      | 속초 | 대한민국 |
+----+----------+-----------+------------+------+----------+
4 rows in set (0.00 sec)
```

---

**트랜잭션 예외**

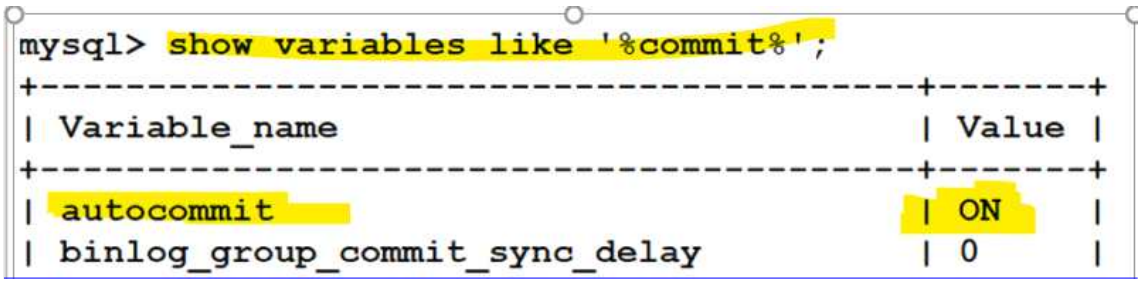
모든 명령어에 대하여 트랜잭션의 롤백 명령이 적용되는 것은 아니다.

**DDL문(CREATE, DROP, ALTER, RENAME, TRUNCATE)**은 transaction의 rollback 대상이 아니다.

**MYSQL 트랜잭션 전역 설정**

MySQL에서는 디폴트로 auto commit이 on으로 설정 되어 있다.

명령어를 한 번, 그러니까 세미콜론을 한 번 찍을 때마다 **DB에서 자동으로 commit**해 주는 시스템이다.




이를, 아예 MySQL 기본 Commit 방식을 바꿀 수 있다.

- 0은 자동 Commit Off
- 1은 자동 Commit On

sql

```
-- 오토커밋 off
SET AUTOCOMMIT = 0;

-- 오토커밋 on
SET AUTOCOMMIT = 1;

-- 오토쿼리 설정 확인
select @@autocommit;
```

트랜잭션 기능은 MySQL의 엔진에 영향을 받는다.

DB 엔진을 InnoDB로 하셔야 트랜잭션을 사용할 수 있다.