8장

트랜잭션(Transaction)

동시성 제어(Concurrency Control)

회복(Recovery)

데이터베이스응용: 503.825(02)

한문석

8장 트랜잭션, 동시성 제어, 회복

데이터베이스응용

(12W-1) 2024-11-18(목)

한문석

학습목표

- 트랜잭션 개념 이해
  - 데이터베이스에서 필요한 이유
- 트랜잭션 실행 시 동시성 제어 필요 이유
  - 락킹(Locking)을 이용한 동시성 제어 기법
- 트랜잭션 고립 수준
  - 락킹(Locking)보다 완화된 방법
  - 트랜잭션의 동시성을 높임
- DB 시스템에 문제가 생길 때 복구 방법

목차

트랜잭션(Transaction)

학습내용

- 트랜잭션의 개념
- 트랜잭션의 성질
- 트랜잭션과 DBMS

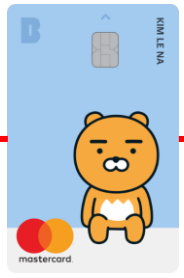
트랜잭션의 개념

A logical unit that is **independently executed for data retrieval or updates**.

트랜잭션의 개념

은행 연금인출기에서 출금거래(Debit Transaction)

카드 삽입



거래유영 선택

거래 취소(ROLLBACK)

거래 완료(COMMIT)

비밀번호 인증

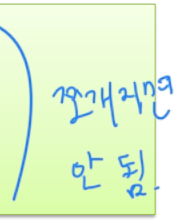
트랜잭션의 개념

- 트랜잭션(transaction): unit of work
  - DBMS에서 데이터를 다루는 논리적인 작업 단위
  - 쪼갤 수 없는 업무처리의 단위
- 데이터베이스에서 트랜잭션을 정의하는 이유
  - 데이터베이스에서 장애가 일어날 때
    - 데이터를 복구하는 작업의 단위
  - DB에서 여러 작업이 동시에 같은 데이터 다룰 때
    - 이 작업을 서로 분리하는 단위가 됨.

트랜잭션의 개념

- 트랜잭션은 전체가 수행되거나 또는 전혀 수행되지 않아야 함(all or nothing).
  - 예) 은행 업무를 보는데 A 계좌(박지성)에서 B 계좌(김연아)로 10,000원을 이체할 경우

BEGIN
① A 계좌(박지성)에서 10,000원을 인출하는 SQL UPDATE 문
② B 계좌(김연아)로 10,000원을 입금하는 SQL UPDATE 문
END



트랜잭션의 개념: 문제 발생과 해결

- 만약 ①번 SQL 문이 수행된 다음 시스템에 문제가 생기거나 다른 UPDATE 문이 끼어들어 A 계좌에서 돈을 동시에 인출하면, A 계좌와 B 계좌의 잔액이 의도하지 않은 값이 될 수 있음
- ①번 SQL 문과 ②번 SQL 문은 모두 수행되거나 아예 수행되지 않아야 함
- 만약 ①번 SQL 문을 수행한 다음 문제가 생겨 ②번 SQL 문을 수행할 수 없는 경우라면 ①번 SQL 문의 수행을 취소해야 함

트랜잭션의 개념: 문제 발생과 해결

- DBMS에 ①번과 ②번의 SQL 문이 하나의 수행 단위라는 것을 알리기 위해 START TRANSACTION 문과 COMMIT 문을 사용하여 트랜잭션의 시작과 끝을 표시함

```
START TRANSACTION
    ① A 계좌(박지성)에서 10,000원을 인출하는 SQL UPDATE 문
    ② B 계좌(김연아)로 10,000원을 입금하는 SQL UPDATE 문
COMMIT;
```

트랜잭션의 개념: 계좌이체 트랜잭션

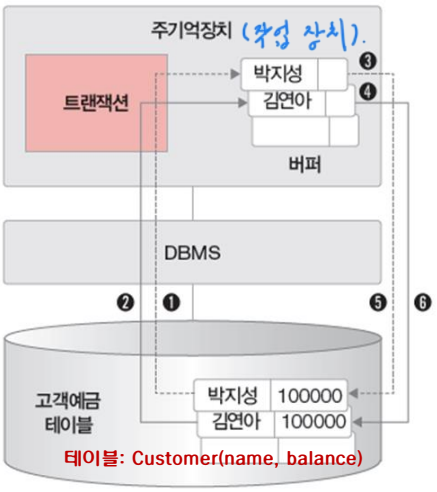
```
START TRANSACTION
① /* 박지성 계좌를 읽어온다 */
② /* 김연아 계좌를 읽어온다 */
   /* 잔고 확인 */

③ /* 예금인출 박지성 */
UPDATE Customer
SET balance=balance-10000
WHERE name= '박지성' ;
```

```
④ /* 예금입금 김연아 */
UPDATE Customer
SET balance=balance+10000
WHERE name= '김연아' ;

COMMIT /* 부분완료 */
⑤ /* 박지성 계좌를 기록한다 */
⑥ /* 김연아 계좌를 기록한다 */
```


트랜잭션의 개념: 트랜잭션 수행 과정



① A 계좌(박지성)의 값을 하드디스크(데이터베이스)에서 주기억장치 버퍼로 읽어온다

② B 계좌(김연아)의 값을 하드디스크(데이터베이스)에서 주기억장치 버퍼로 읽어온다.

③ A 계좌(박지성)에서 10,000원을 인출한 값을 저장한다.

④ B 계좌(김연아)에 10,000원을 입금한 값을 저장한다.

⑤ A 계좌(박지성)의 값을 주기억장치 버퍼에서 하드디스크(데이터베이스)에 기록한다.

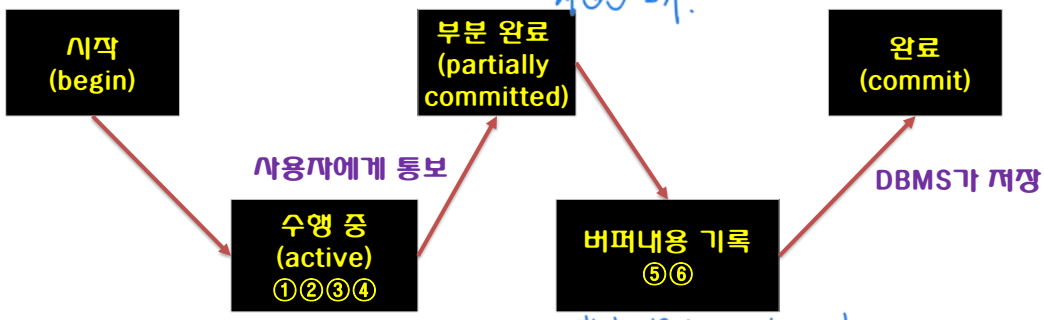
⑥ B 계좌(김연아)의 값을 주기억장치 버퍼에서 하드디스크(데이터베이스)에 기록한다.

트랜잭션 종료(COMMIT) 알림 방법

[방법 1] ①－②－③－④－COMMIT－⑤－⑥

[방법 2] ①－②－③－④－⑤－⑥－COMMIT

→ DBMS는 사용자에게 빠른 응답성을 보장하기 위해 [방법 1]을 선택한다.



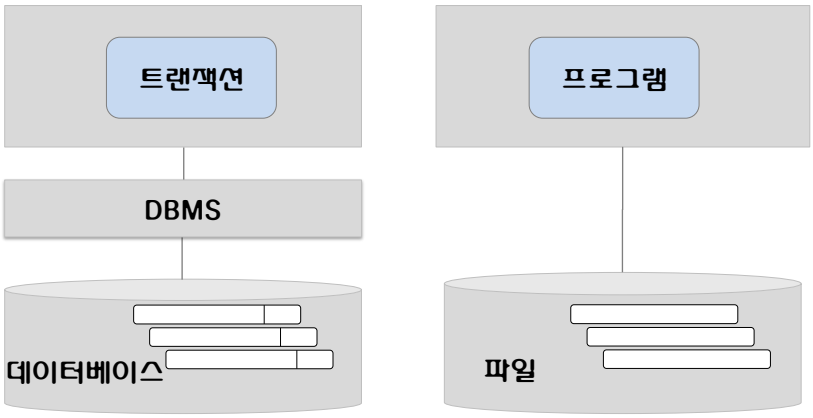
트랜잭션의 성질

- 트랜잭션과 프로그램의 차이점

구분	트랜잭션	프로그램
프로그램 구조	BEGIN TRANSACTION … COMMIT	main() { … }
운영 데이터	데이터베이스 저장된 데이터	파일에 저장된 데이터
번역기	DBMS	컴파일러
성질	원자성, 일관성, 고립성, 지속성	–

트랜잭션의 성질

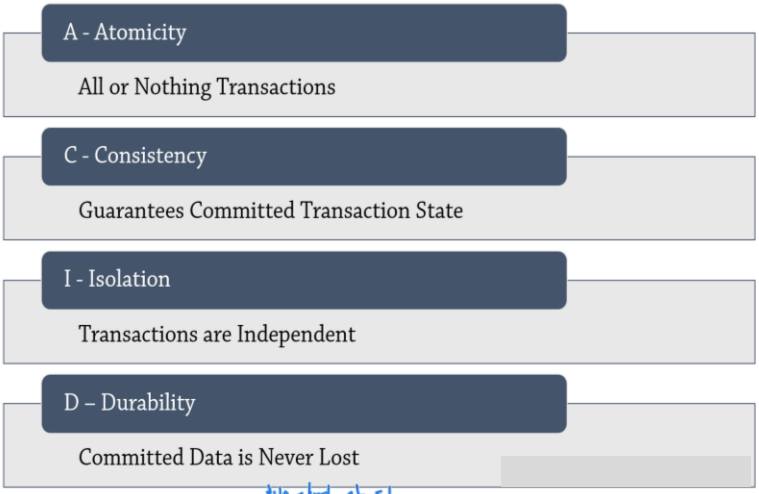
- 컴퓨터 시스템 내의 트랜잭션과 프로그램



트랜잭션의 ACID 성질

- 원자성(Atomicity)
  - 트랜잭션에 포함된 작업은 전부 수행되거나 아니면 전부 수행되지 않아야(all or nothing) 함.
- 일관성(Consistency)
  - 트랜잭션을 수행하기 전이나 수행한 후나 데이터베이스는 항상 일관된 상태를 유지해야 함.
- 고립성(Isolation)
  - 수행 중인 트랜잭션에 다른 트랜잭션이 끼어들어 변경 중인 데이터 값을 훼손하는 일이 없어야 함.
- 지속성(Durability)
  - 수행을 성공적으로 완료한 트랜잭션은 변경한 데이터를 영구히 저장해야 함.

트랜잭션의 ACID 성질



훼손되면 안 됨.

원자성(Atomicity)

- 트랜잭션이 원자처럼 더 이상 쪼개지지 않는 하나의 프로그램 단위로 동작해야 한다는 의미
- 일부만 수행되는 일이 없도록 전부 수행하거나 아예 수행하지 않아야(all or nothing) 함.

트랜잭션 제어 명령어(TCL)

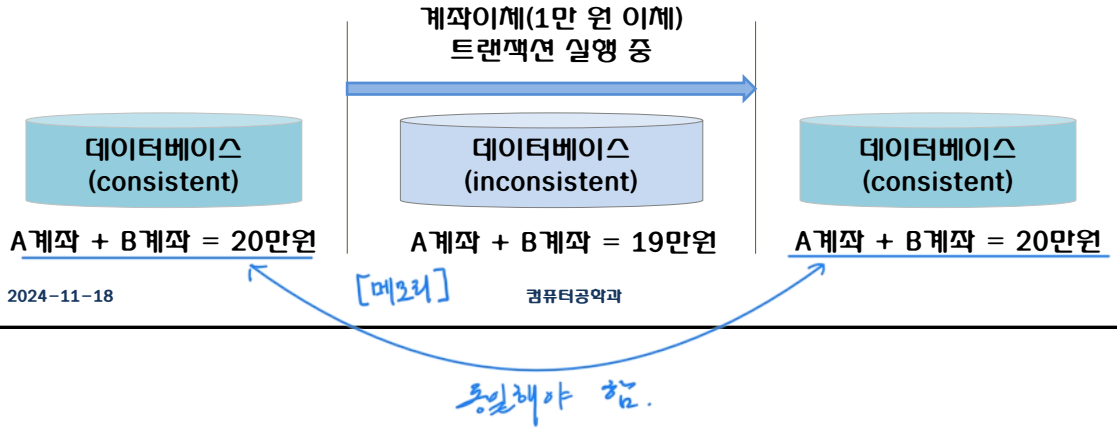
표준 명령어	데이터베이스 문법	설명
START TRANSACTION	SET TRANSACTION NAME <이름>	트랜잭션의 시작
COMMIT	COMMIT	트랜잭션의 종료
ROLLBACK	ROLLBACK {TO <savepoint>}	트랜잭션을 전체 혹은 <savepoint>까지 무효화시킴
SAVEPOINT	SAVEPOINT <identifier>	<savepoint>를 만듦

어느 지점.

일관성(Consistency)

- 트랜잭션은 데이터베이스의 일관성을 유지해야 함.
- 일관성은 테이블 생성 시
  - CREATE 문과 ALTER 문의 무결성 제약조건을 통해 명시됨.

데이터베이스 변경 중과 변경 후의 일관성

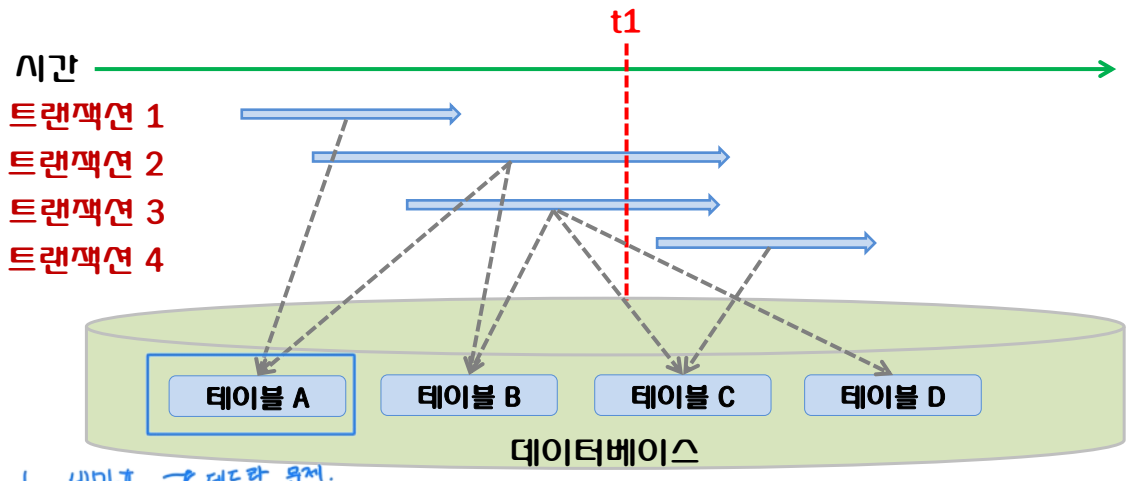


고립성(Isolation)

- 데이터베이스는 공유가 목적
  - 여러 트랜잭션이 동시에 수행됨.
- 동시에 수행되는 트랜잭션은 상호 존재를 모름
- 독립적으로 수행되는데, 이를 고립성이라고 함.
- 고립성을 유지하기 위해서는 트랜잭션이 변경 중인 임시 데이터를 다른 트랜잭션이 읽고 쓸 때 제어가 필요함.
  - 동시성제어

세마포?

고립성(Isolation)



1. 세마포 → 데드락 문제.
2.

트랜잭션의 동시 수행과 데이터 공유

지속성(Durability)

- 트랜잭션이 정상적으로 완료(commit) 혹은 부분완료(partial commit)한 데이터는 DBMS가 책임지고 데이터베이스에 기록하는 성질

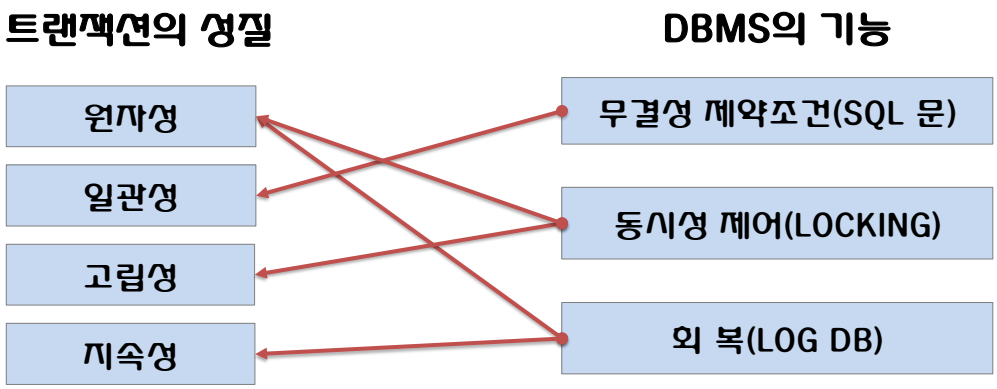
트랜잭션의 상태도

트랜잭션과 DBMS

- DBMS는 원자성을 유지하기 위해 회복(복구) 관리자 프로그램을 작동시킴.
- DBMS는 일관성을 유지하기 위해 무결성 제약조건을 활용함.
- DBMS는 고립성을 유지하기 위해 일관성을 유지하는 것과 마찬가지로 동시성 제어 알고리즘을 작동시킴.
- DBMS는 지속성을 유지하기 위해 회복 관리자 프로그램을 이용함.

undo, redo 내역 생성.

트랜잭션과 DBMS



트랜잭션의 성질과 DBMS의 기능

MySQL 트랜잭션 실습

- 마당DB 초기화
  - demo_madang_init.sql 실행
- 트랜잭션 시작
  - 표준 명령
    - START TRANSACTION
  - SET TRANSACTION
    - 트랜잭션의 특성을 지정
  - 데이터 변경문 실행 시
- 트랜잭션 끝
  - COMMIT/ROLLBACK문
  - DDL문 실행 시

MySQL 트랜잭션 실습

- SET TRANSACTION
  - 트랜잭션의 특성을 지정
  - 특성값
    - Transaction Isolation Levels
    - Transaction Access Mode
    - Transaction Characteristic Scope

MySQL 트랜잭션 실습

```
SET [GLOBAL | SESSION] TRANSACTION
    transaction_characteristic [,transaction_characteristic] ...

transaction_characteristic: { ISOLATION LEVEL level |
access_mode }

level: { REPEATABLE READ | READ COMMITTED | READ
UNCOMMITTED |       SERIALIZABLE }

access_mode: {      READ WRITE | READ ONLY
}
```


MySQL 트랜잭션 실습

- **COMMIT**
  - Commit the current transaction, making its changes permanent.
- **ROLLBACK**
  - Roll back the current transaction, canceling its changes.

MySQL 트랜잭션 실습: 파일 8_code01.sql

```
START TRANSACTION;
```

START TRANSACTION

```
INSERT INTO Book VALUES(99, '데이터베이스', '한빛', 25000);
```

INSERT INTO Book VALUES(99, '데이터베이스', '한빛', 25000)

```
SELECT bookname 'bookname1' FROM Book  WHERE bookid=99;  /* 데이터베이스 */
```

bookname1
데이터베이스

SELECT bookname 'bookname1' FROM Book WHERE bookid=99 LIMIT 0, 1000

```
SAVEPOINT a;
```

SAVEPOINT a

MySQL 트랜잭션 실습: 파일 8_code01.sql

```
UPDATE Book SET bookname='데이터베이스 개론' WHERE bookid=99;
```

UPDATE Book SET bookname='데이터베이스 개론' WHERE bookid=99

```
SELECT bookname 'bookname2'  FROM Book  WHERE bookid=99;
/* 데이터베이스 개론 */
```

bookname2
데이터베이스 개론

SELECT bookname 'bookname2' FROM Book WHERE bookid=99 LIMIT 0, 1000

```
SAVEPOINT b;
```

SAVEPOINT b

MySQL 트랜잭션 실습: 파일 8_code01.sql

```
UPDATE Book SET bookname='데이터베이스 개론 및 실습' WHERE bookid=99;
```

```
UPDATE Book SET bookname='데이터베이스 개론 및 실습' WHERE bookid=99
```

```
SELECT bookname 'bookname3'  FROM Book  WHERE bookid=99;
/* 데이터베이스 개론 및 실습 */
```

	bookname3
▶	데이터베이스 개론 및 실습

```
SELECT bookname 'bookname3'  FROM Book  WHERE bookid=99 LIMIT 0, 1000
```

```
ROLLBACK TO b;
```

```
ROLLBACK TO b
```


MySQL 트랜잭션 실습: 파일 8_code01.sql

```
SELECT bookname 'bookname4'  FROM Book  WHERE bookid=99;
/* 데이터베이스 개론 */
```

	bookname4
▶	데이터베이스 개론

```
SELECT bookname 'bookname4'  FROM Book  WHERE bookid=99 LIMIT 0, 1000
```

```
ROLLBACK TO a;
```

```
ROLLBACK TO a
```

```
SELECT bookname 'bookname5'  FROM Book  WHERE bookid=99;
/* 데이터베이스 */
```

	bookname5
▶	데이터베이스

```
SELECT bookname 'bookname5'  FROM Book  WHERE bookid=99 LIMIT 0, 1000
```

```
COMMIT;
```

```
COMMIT
```

MySQL 트랜잭션 실습: 파일 8_code01.sql

```
START TRANSACTION;
```

START TRANSACTION

```
UPDATE Book SET bookname='데이터베이스 개론 및 실습2' WHERE bookid=99;
```

UPDATE Book SET bookname='데이터베이스 개론 및 실습2' WHERE bookid=99

```
SELECT bookname 'bookname6'  FROM Book   WHERE bookid=99;
/* 데이터베이스 개론 및 실습2 */
```

bookname6
데이터베이스 개론 및 실습2

SELECT bookname 'bookname6' FROM Book   WHERE bookid=99 LIMIT 0, 1000

```
ROLLBACK;
```

ROLLBACK

MySQL 트랜잭션 실습: 파일 8_code01.sql

```
SELECT bookname 'bookname7'  FROM Book   WHERE bookid=99;
/* 데이터베이스 */
```

bookname7
데이터베이스

SELECT bookname 'bookname7' FROM Book   WHERE bookid=99 LIMIT 0, 1000

```
DELETE FROM  Book WHERE bookid=99;
```

DELETE FROM Book WHERE bookid=99

```
COMMIT;
```

COMMIT


ENJOY YOUR DB!!!