트랜잭션(Transaction) 이란

트랜잭션(Transaction)이란 데이터베이스의 상태를 변환시키는 하나의 논리적 기능을 수행하기 위한 작업의 단위 또는 한꺼번에 모두 수행되어야 할 일련의 연산들을 의미한다.

데이터베이스의 상태를 변경시킨다는 것은 SELECT, UPDATE, INSERT, DELETE 와 같은 행동을 뜻한다.

트랜잭션은 상황에 따라 여러 개가 만들어질 수 있다.
하나의 트랜잭션은 Commit (저장) 되거나 Rollback (철회)될 수 있다.

트랜잭션(Transaction) 이란

예를들어 A 계좌에서 B 계좌로 만원을 송금 했다고 가정했을때 A계좌에서는 만원이 인출되고 B계좌에서는 만원이 입금되어야 한다. 계좌이체라는 행위는 인출과 입금 두 과정으로 이루어진다.

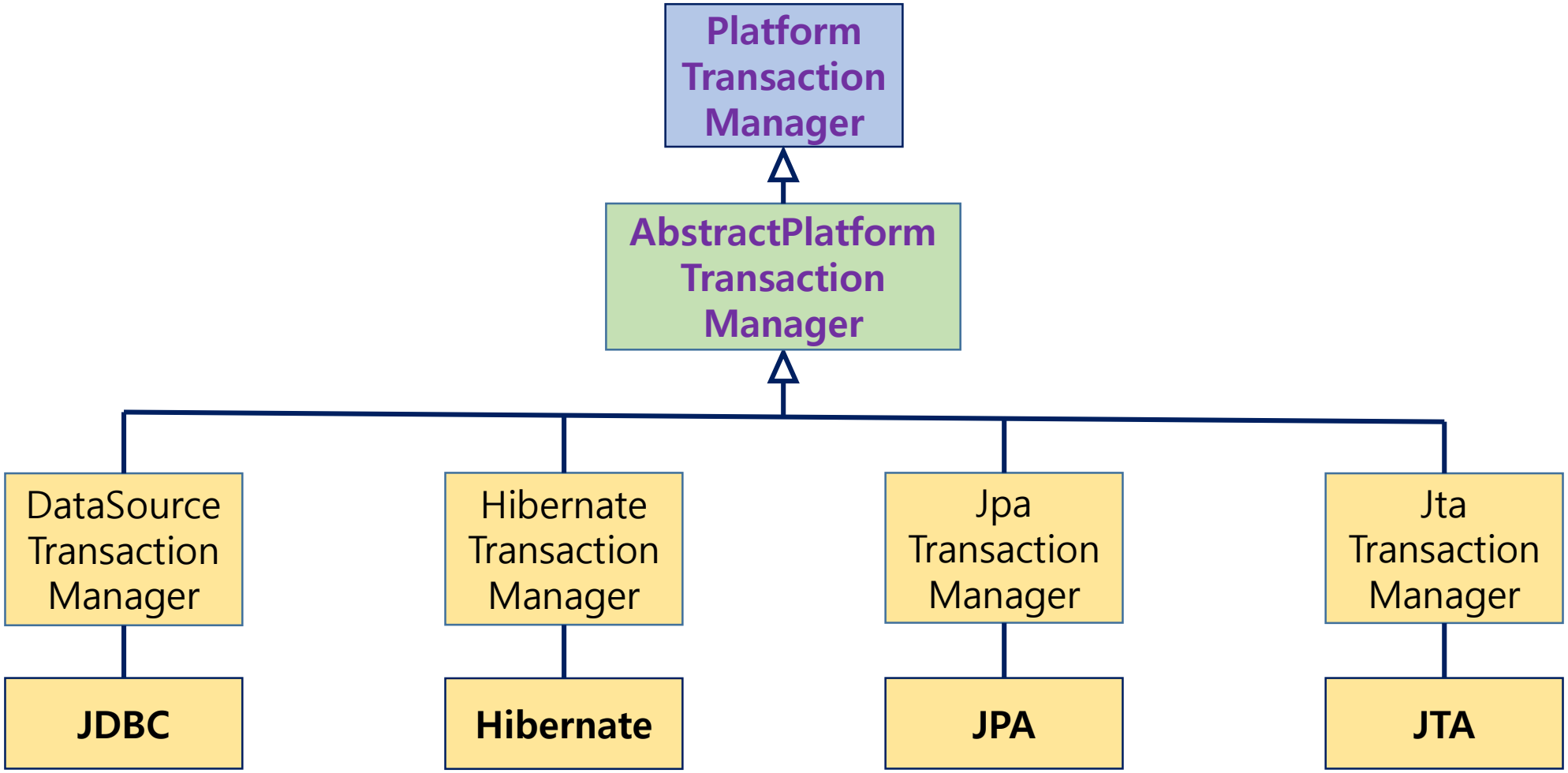
근런데 시스템상의 이유로 인출은 성공했는데, 입금에 실패했다면?.. 치명적인 결과가 나오게 된다. 따라서 이 두 과정은 동시에 성공 또는 실패하여야 한다.

데이터베이스에서 데이터를 읽어 온 후 다른 테이블에 데이터를 입력, 갱신, 삭제하는 도중에 오류가 발생하면, 결과를 재반영 하는 것이 아니라 모든 작업을 원상태로 복구하고, 처리 과정이 모두 성공하였을때만 그 결과를 반영하도록 한다.

트랜잭션 성질

원자성(Atomicity)	일관성(Consistency)	독립성(Isolation)	지속성(Durability)
한 트랜잭션 내의 실행 작업은 하나로 간주한다. (즉 모두 성공 또는 모두 실패를 의미)	트랜잭션은 일관성 있는 데이터베이스 상태를 유지한다. (제약 조건이나 데이터 규칙에 위반하지 않는 일관성을 의미)	동시에 실행되는 트랜잭션들이 서로 영향을 미치지 않도록 해야 한다. (한 동작이 독립적으로 처리해야 한다는 원칙)	트랜잭션을 성공적으로 마치면 결과가 항상 저장되어야 한다.

Spring 트랜잭션(Transaction)



PlatformTransactionManager

스프링에서는 PlatformTransactionManager 인터페이스로 트랜잭션 처리를 추상화를 했다. 구현 클래스는 설정 파일에 Bean으로 등록하고, DI를 받아 사용한다.
PlatformTransactionManager는 TransactionManager의 최상위 인터페이스로, 인터페이스에 각자의 환경에 맞는 TransactionManager 객체를 주입하여 사용한다.

예를 들어, JDBC 및 MyBatis 등의 JDBC 기반 라이브러리로 데이터베이스에 접근하는 경우에 DataSourceTransactionManager를 트랜잭션 관리자로 사용하고, 하이버네이트는 HibernateTransactionManager, JPA는 JpaTransactionManager 를 트랜잭션 관리자로 사용하여 해당 객체를 주입한다.

JDBC 기반 트랜잭션 처리

```
<bean id="tm" class="org.springframework.jdbc.datasource.DataSourceTransactionManager" >
    <property name="dataSource" ref="ds"></property>
</bean>
```

```
@Autowired
PlatformTransactionManager tm;
```

```
TransactionDefinition definition = new DefaultTransactionDefinition();
TransactionStatus status = tm.getTransaction(definition);
```

```
tm.commit(status);
```

```
tm.rollback(status);
```

스프링 설정파일에 DataSourceTransactionManager 클래스로 빈 객체를 생성한 후 Data Access Object 클래스에서는 PlatformTransactionManager 한테 주입하여 COMMIT 메소드와 ROLLBACK 메소드로 트랜잭션 처리를 한다.

TransactionTemplate

PlatformTransactionManager 를 사용하여 개발자가 트랜잭션의 시작 및 종료 시점을 명시적으로 결정할 수 있다. 하지만 PlatformTransactionManager 보다 더 간편한 방법으로는 TransactionTemplate 을 사용하는 방법이 있다.

```
<bean id="transactionTemplate" class="org.springframework.transaction.support.TransactionTemplate">
    <property name="transactionManager" ref="transactionManager" />
</bean>
```

스프링 설정파일에 TransactionManager 객체를 주입받는 TransactionTemplate 빈 객체를 생성을 해준다.

TransactionTemplate

```
@Autowired
private TransactionTemplate tranTemplate;
```

```
tranTemplate.execute(new TransactionCallbackWithoutResult() {

    @Override
    protected void doInTransactionWithoutResult(TransactionStatus status) {

        // 실행할 소스코드 작성

    }
});
```

TransactionTemplate 의 execute 메소드로 TransactionCallback<T> 타입의 객체를 넣어주고 해당하는 추상 메소드를 오버라이딩 하여 메소드 안에 실행할 소스코드를 작성 한다.