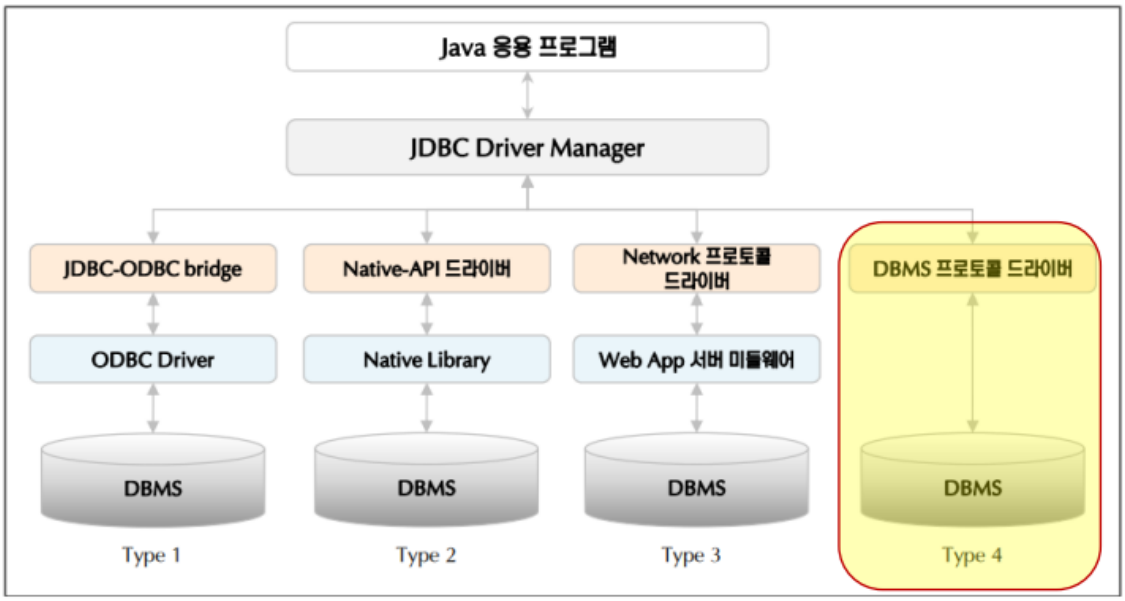
6월 13일 DB 연동수업

▼ JDBC

▼ JDBC 드라이버 종류

✓ **JDBC 드라이버 종류**
- JDBC-ODBC Bridge Driver
- Native API Driver
- Network Protocol Driver
- DBMS Protocol Driver – JDBC 표준에 따라 순수 자바 코드로 작성된 드라이버를 주로 사용한다



JDBC 드라이버의 종류

type 1) MsSQL을 연결할때에는 ODBC Driver를 사용하여야하는데 이건 C언어로 만들어졌기때문에 중간 역할인 JDBC-ODBC Bridge Driver를 사용하여야한다 / 그냥 알아두기만 하기

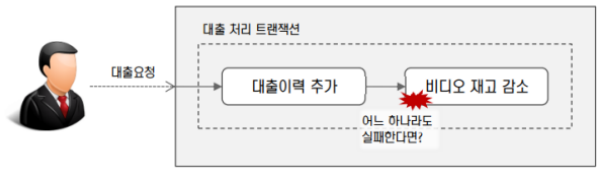
type 2) Native Library도 C언어로 만들어졌기때문에 Native API Driver로 연결해주어야한다 / 타입 1과 거의 유사함

type 3) Wep App 서버 미들웨어를 다운받아서 연결을 하고 네트워크로 드라이버를 사용할 수 있다 주로 네트워크로 연결해서 사용한다

type 4) DBMS Protocol Driver 순수 자바 코드로 작성된 드라이버로 대부분 DBMS Protocol Driver를 많이 사용한다
단점으로는 로컬쪽에서 사용해야하기때문에 드라이버도 설치파일에 같이 포함시켜야 함, 드라이버가 수정될때마다 재배포해야함

▼ 트랜잭션

✓ 트랜잭션은 여러 개의 관련 있는 소단위 작업들을 하나의 처리단위로 묶어 놓은 것을 의미한다
✓ 하나의 트랜잭션은 모든 데이터가 정상적으로 반영되거나 반영되지 않음을 보장한다(All or Nothing)
✓ 애플리케이션에서 한번의 요청이 여러 테이블의 데이터들을 동시에 수정한다면 하나의 트랜잭션으로 처리되어야 한다
✓ 트랜잭션이 적용되지 않으면 애플리케이션의 데이터를 신뢰 할 수 없다



- 대출이 일어날 경우
  1) 대출 테이블에 대출 이력에 대한 행이 추가됨과 동시에
  2) 영화 테이블의 재고 수량과 수정일자 값이 갱신되어야 합니다.
- 만약 두 작업 중 하나라도 이루어지지 않는다면 데이터의 무결성이 보장이 안됩니다.
  따라서 두 작업은 하나의 Transaction으로 처리되어야 합니다.

[대출 테이블]

고객ID	영화등록번호	대여일자	대여비
C0001	M0001	2015-03-03	500
C0002	M0002	2015-02-26	500
C0003	M0003	2015-03-02	1000
C0003	**M0003**	**2015-03-09**	**1000**

① 대출 테이블에 새로운 행이 추가됨

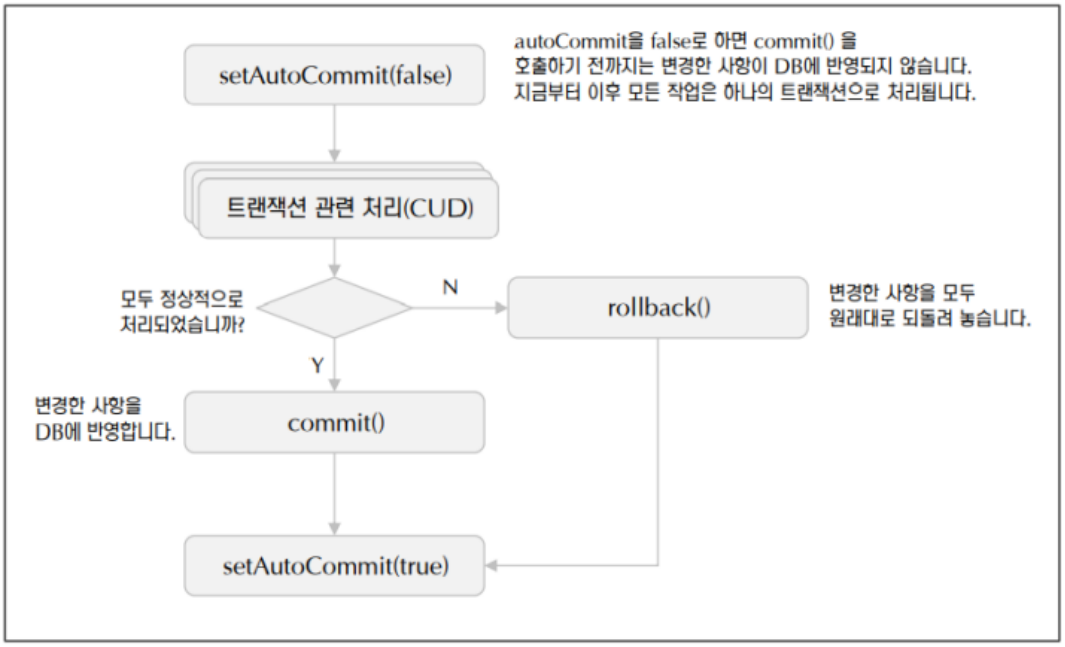
[영화 테이블]

영화등록번호	영화제목	재고	수정일자
M0001	변호인	10	2015-03-03
M0002	또 하나의 약속	8	2015-02-26
M0003	어벤져스 2	5	2015-03-02

② 재고량 감소 (5 → 4)
② 수정일자 갱신 (→ 2015-03-09)

▼ JDBC 트랜잭션 관리

✓ JDBC의 Connection 인터페이스는 트랜잭션 관리를 위한 메소드를 제공한다
✓ 여러 Statement를 한번에 반영함으로써 트랜잭션을 보장한다
✓ connection.commit() 또는 connection.rollback() 메소드로 수행된 Statement를 반영하거나 취소한다
✓ 트랜잭션을 관리하기 위해서는 connection.setAutoCommit(false)를 수행하여야 한다(기본값 : true)



JDBC를 이용한 트랜잭션 처리과정

▼ 트랜잭션 예외처리

트랜젝션은 기본적으로 AutoCommit이 됐다
따라서 예외처리를 할땐 AutoCommit을 꺼줘야함

✓ JDBC를 사용해서 DB와 연동할 때 접속실패, SQL오류 등의 예외가 발생할 수 있다
✓ 예외는 SQLException과 이를 상속받은 SQLWarning, DataTruncation, BatchUpdateException등이 제공된다
✓ 트랜잭션 처리 중 예외가 발생하는 경우 catch절을 통해서 rollback()을 수행하도록 한다
✓ 예외처리가 적절히 수행되지 않는다면 데이터의 무결성을 보장 할 수 없다

```
try {
// DB 연결객체 생성
con = DriverManager.getConnection(url, user, password);

// 트랜잭션 시작
con.setAutoCommit(false);

// 트랜잭션 관련 처리1
String sql = "INSERT INTO ... VALUES (?,?,?)";
pstmt = conn.prepareStatement(sql);

// 트랜잭션 관련 처리2
conn.commit();
  } catch (SQLException e) {
    e.printStackTrace();
  try {
    conn.rollback();
```

```
  } catch (SQLException e1) { }
  } finally {
    try {
      pstmt.close();
      conn.close();
    } catch (SQLException e) { }
}
```

```
try{
  con.setAutoCommit(false);
  ...
  con.commit();
} catch (SQLException) {
  con.rollback();
}
// 오류가 발생하면 중간에 롤백해준다
```

▼ CallableStatement

✓ PL/SQL 프로시저 호출

```
StringBuilder sb = new StringBuilder();
sb.append("{call findemployeebyno(?, ?, ?, ?)}");
Connection con = null;
CallableStatement cstmt = null;
try {
    Class.forName(driver);
    con = DriverManager.getConnection(url, userid, password);
    cstmt = con.prepareCall(sb.toString());
    cstmt.setInt(1, 110);
    cstmt.registerOutParameter(2, Types.INTEGER);
    cstmt.registerOutParameter(3, Types.VARCHAR);
    cstmt.registerOutParameter(4, Types.VARCHAR);
    cstmt.execute();
    int id = cstmt.getInt(2);
    String name = cstmt.getString(3);
    int salary = cstmt.getInt(4);
    System.out.println(id + "\t" + name + "\t" + salary);
}finally {
    try {
        if(cstmt != null) cstmt.close();
        if(con != null)   con.close();
    }catch (Exception e) {}
}
```

PreparedStatement와 똑같이 setXXX(,)를 사용하면 된다
대신 반환받을땐 registerOutParameter(, Types.Object)를 사용하면 된다
실행할땐 execute()를 사용하면 결과가 들어온다
호출한 결과를 받은 값을 getXXX()에서 받아 사용한다

```
-- SQLPlus*에서 프로시저 실행
-- 반환값을 받기 위한 변수선언
/*
VAR b_id NUMBER;
VAR b_name VARCHAR2(20);
VAR b_salary NUMBER;

EXECUTE findEmployeeByNo(100, :b_id, :b_name, :b_salary);
-- 변수값 출력
print b_id;
*/
```

▼ 용어 정리

▼ 리플렉션

리플렉션을 사용하면 **생성자, 메소드, 필드 등 클래스에 대한 정보를 아주 자세히 알아낼 수 있다**

Class 클래스

리플렉션의 가장 핵심은 `Class` 클래스이다. `Class` 클래스는 `java.lang` 패키지에서 제공된다. 어떻게 특정 클래스의 `Class` 인스턴스를 획득할 수 있을까?

Class 객체 획득 방법

```
Class<Member> aClass = Member.class; // (1)

Member member1 = new Member();
Class<? extends Member> bClass = member1.getClass(); // (2)

Class<?> cClass = Class.forName("hudi.reflection.Member"); // (3)
```

▼ BOF/EOF

BOF (Before of File) :

EOF (End of File) :

▼

isEmpty() : 비어있는 배열 / 비어있으면 true 존재하면 false

Map<>() : 키와 값으로 저장할때 사용한다,