CSE3207-002 데이터베이스 Project1 report

12130919 강 혁

1. 구현 사항

1번 테이블 생성 및 초기 데이터 Insertion

2번 (2.1 ~ 2.8) Insertion

3번 (3.1 ~ 3.5) customerRate 테이블 관련 Update 및 avg() aggregation 함수 이용한 movie 테이블 Update

4번, 5번, 6번 nested subquery를 이용한 Select

7번, 8번, 9번 의존성 있는 테이블 데이터 Deletion 및 테이블 Drop

모두 구현했습니다.

1. 구현 설명

Java Source Code :

error가 어느 query에서 발생하는지를 찾기 위해 각 query처리하는 프로세스를 각각의 try-catch문으로 묶어서 코드를 작성했습니다.

Query 1 :

다른 테이블에서 참조할 key를 가진 테이블 director, actor, movie, award, customer, genre를 먼저 CREATE TABLE ~ VALUES(…)문으로 생성 후 이 테이블들을 참조하는 reference key를 가진 테이블들을 생성했고, 그 후 INSERT INTO ~ VALUES(…) 구문을 통해 초기 데이터를 삽입했습니다.

Query 2 :

award 테이블에 들어갈 초기데이터에 상에 관한 정보가 없었으므로 각 Statement마다 주어진 상들의 정보를 award에 Insertion합니다. 그 후 Statement에 따라 directorObtain, movieObtain, actorObtain 테이블에도 주어진 수상 정보를 Insertion했습니다.

Query 3:

먼저 customer 테이블에서 customerName을 이용해 customerID를 가져오고, movie 테이블에서는 주어진 정보를 이용해 subquery를 구성하고 해당하는 movieID들을 가져옵니다. 그 ID들을 이용해 customerRate에 주어진 Statement에 따라 각 영화마다 평점 정보를 Insertion 합니다. 그 후 avg() 함수를 이용해 movie 테이블의 avgRate column을 Update해줍니다.

Query 4, 5, 6:

주어진 Statement에 따라 묘사하는 movie, director에 대한 정보들을 이용해 subquery로 각 movieID, directorID를 가져오고 movieName과 directorName을 Select문을 통해 가져옵니다. 여기서 6번에서 묘사한 common genre의 경우 count() 함수를 이용해 genre의 수를 계산하고 max() 함수로 가장 많은 영화가 가진 genre를 common genre로 간주해 문제를 해결했습니다.

Query 7:

not exists keyword를 이용해 movieObtain, directorObtain, actorObtain을 check하고 수상된 정보가 없는 상을 award 테이블에서 제거하는 쿼리를 작성했습니다. 해당 문제에서는 모든 상이 수상되어 결과적으로는 삭제된 데이터는 없습니다.

Query 8:

customerRate 테이블의 데이터를 먼저 Deletion하고, 결과에 따라 movie 테이블의 avgRate column도 Update해줍니다. 그 후 의존성을 모두 해결 했으므로 customer 테이블의 데이터를 모두 Deletion 해주었습니다.

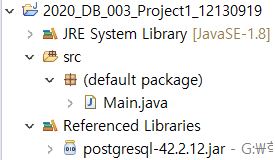
Query 9:

DROP Table ~ 문을 이용해 테이블과 그 테이블에 저장된 데이터를 한 번에 제거했습니다. DROP의 순서는 CREATE TABLE의 역순으로 referencing table먼저 DROP 후 referenced table을 나중에 제거하는 순으로 query를 작성했습니다.

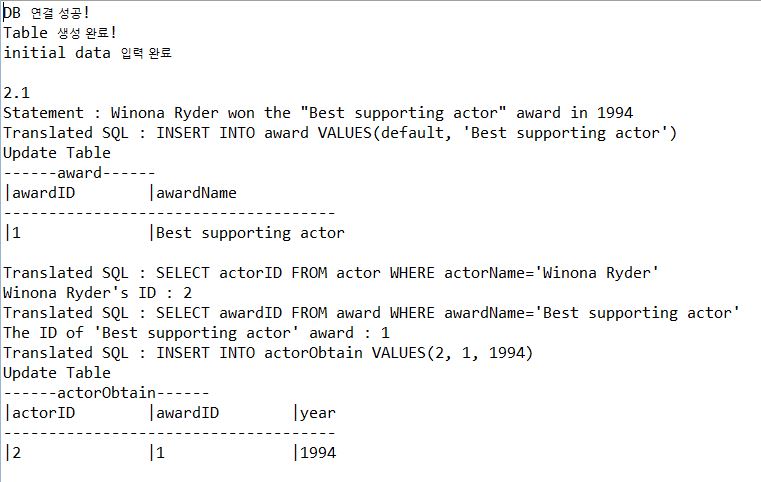
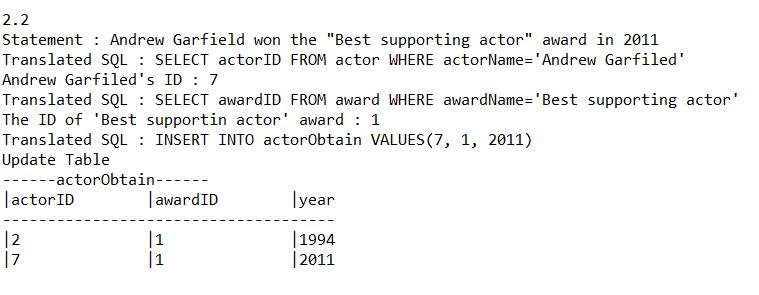
1. 컴파일 환경 및 실행 환경

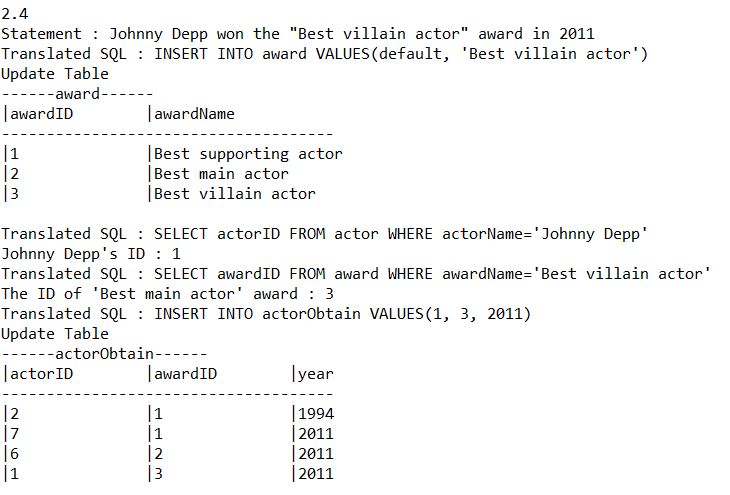
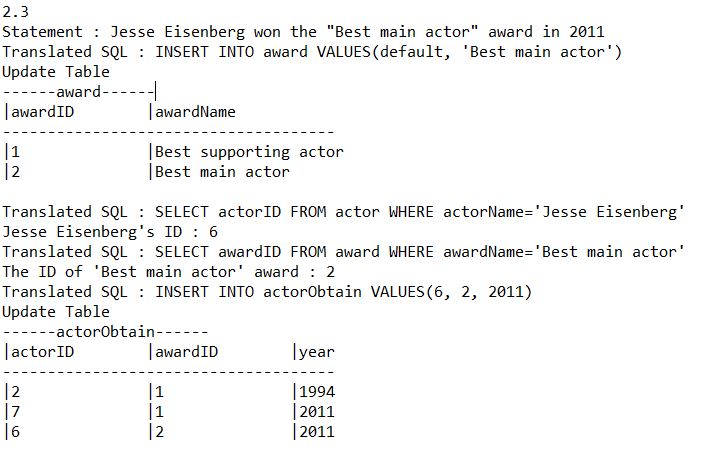
Eclipse IDE for Enterprise Java Developers – 2019 -12 ver

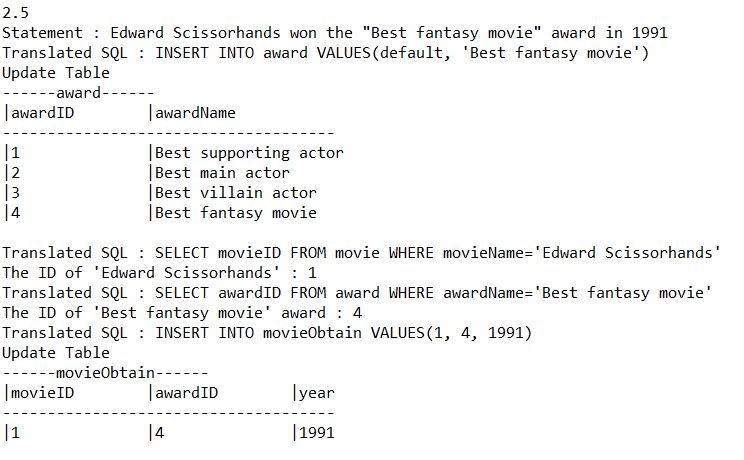
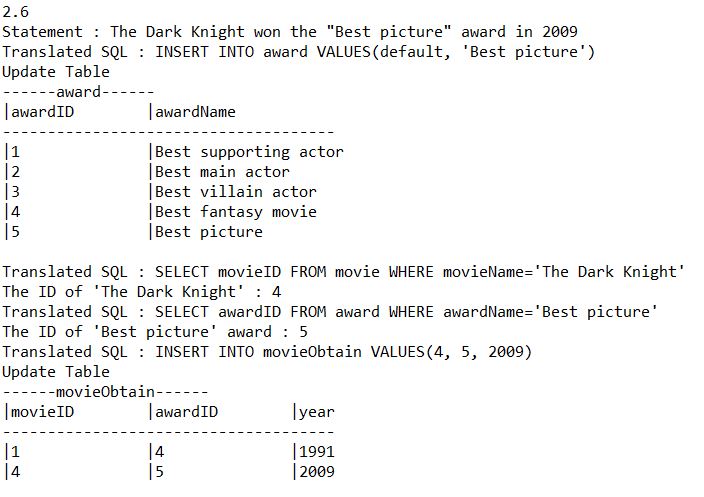
Java version openjdk 1.8 (java 8)

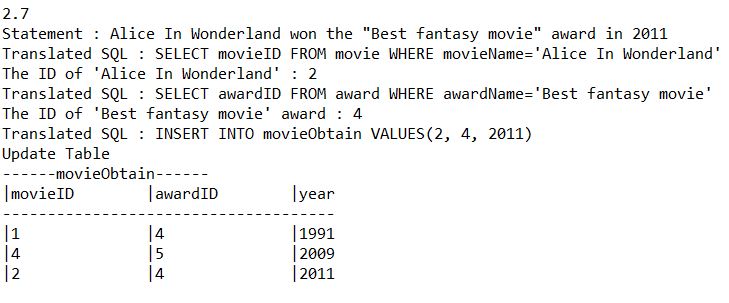
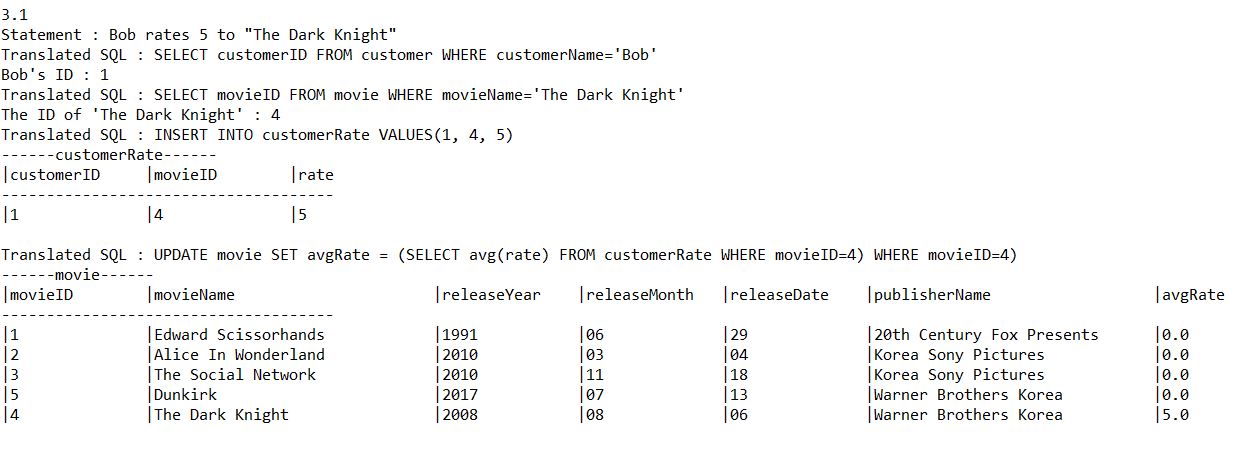


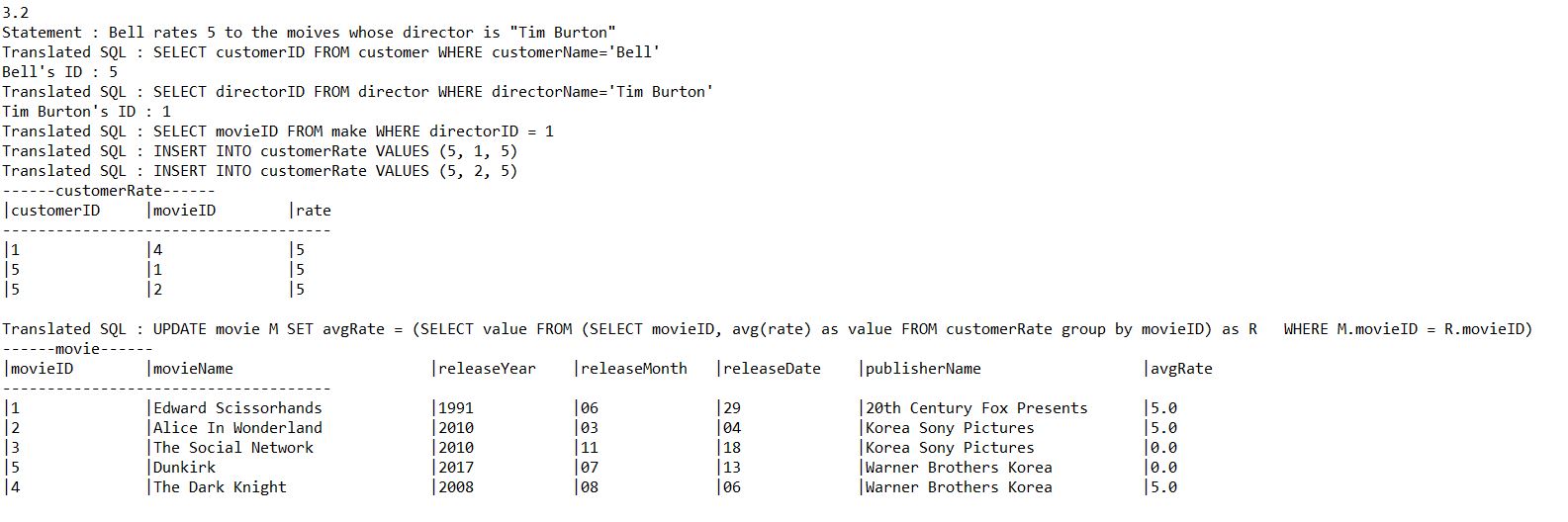
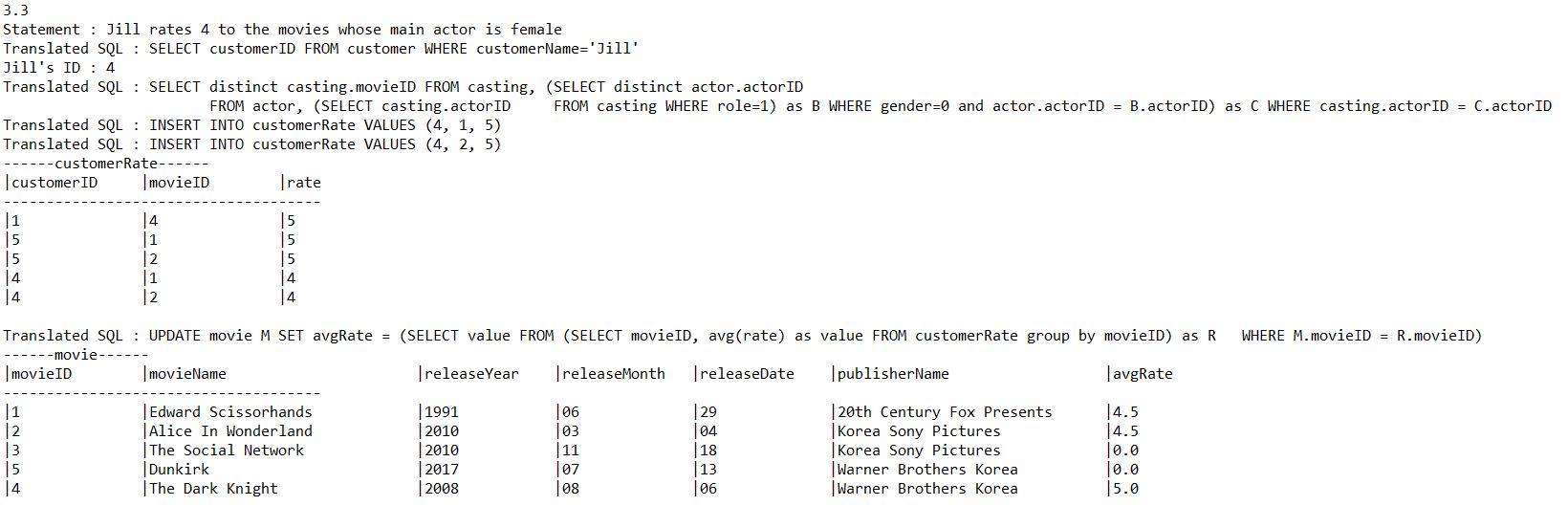
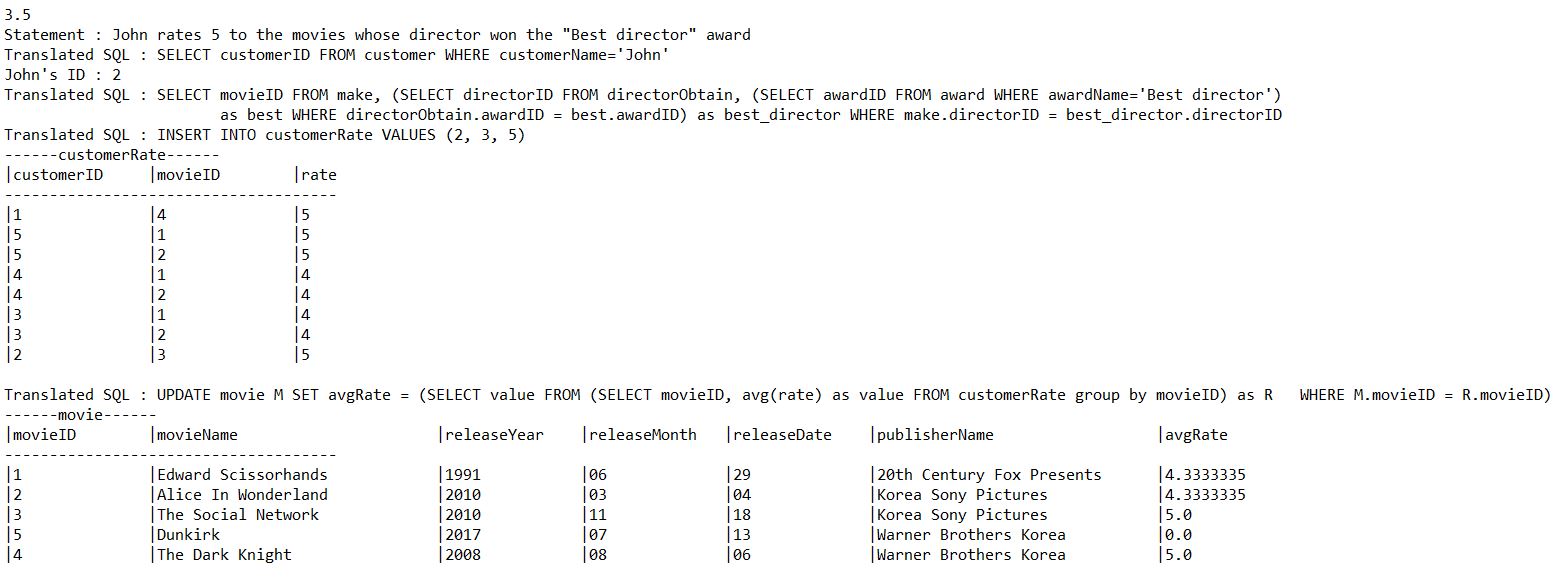
프로젝트 구조 –

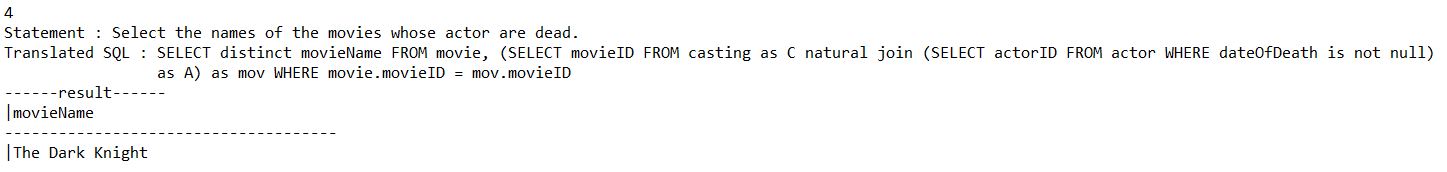
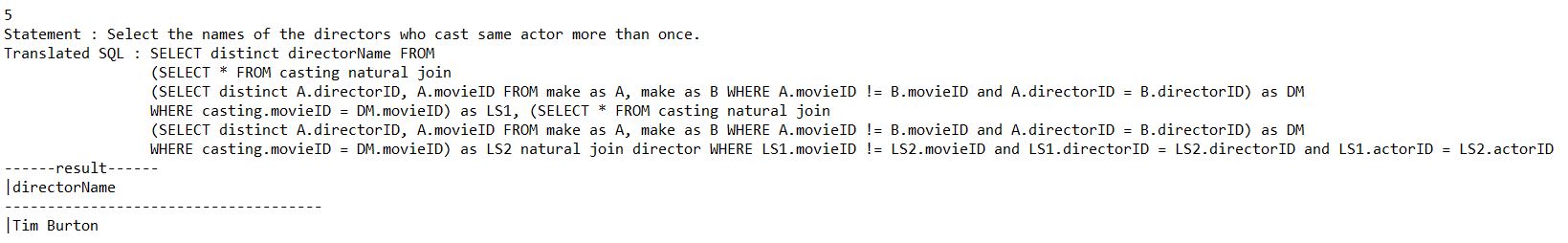
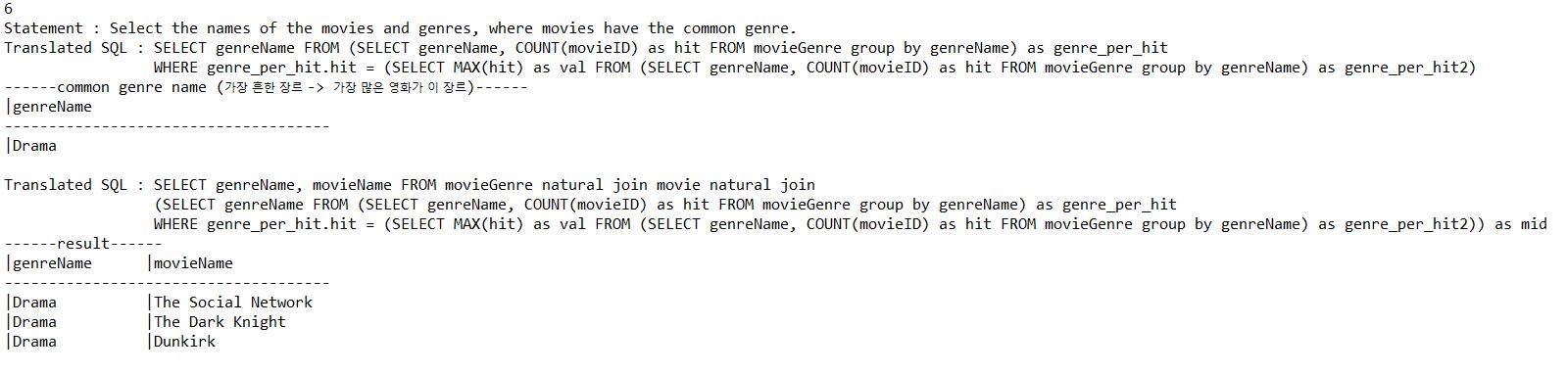
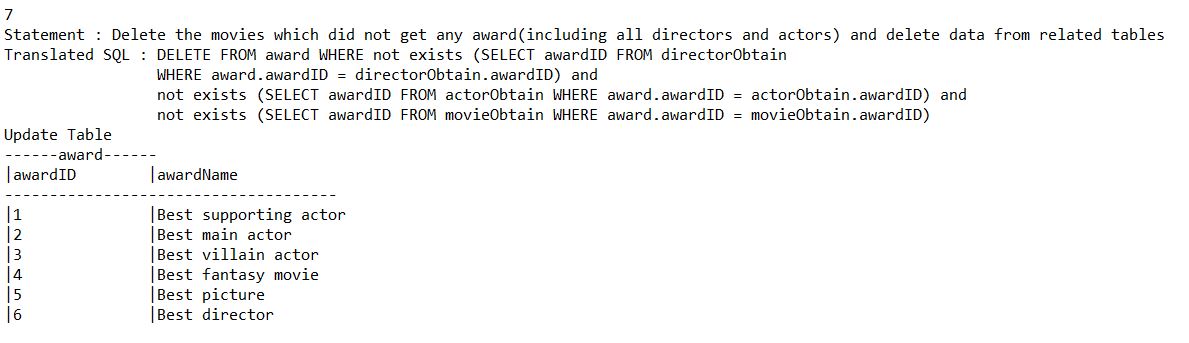
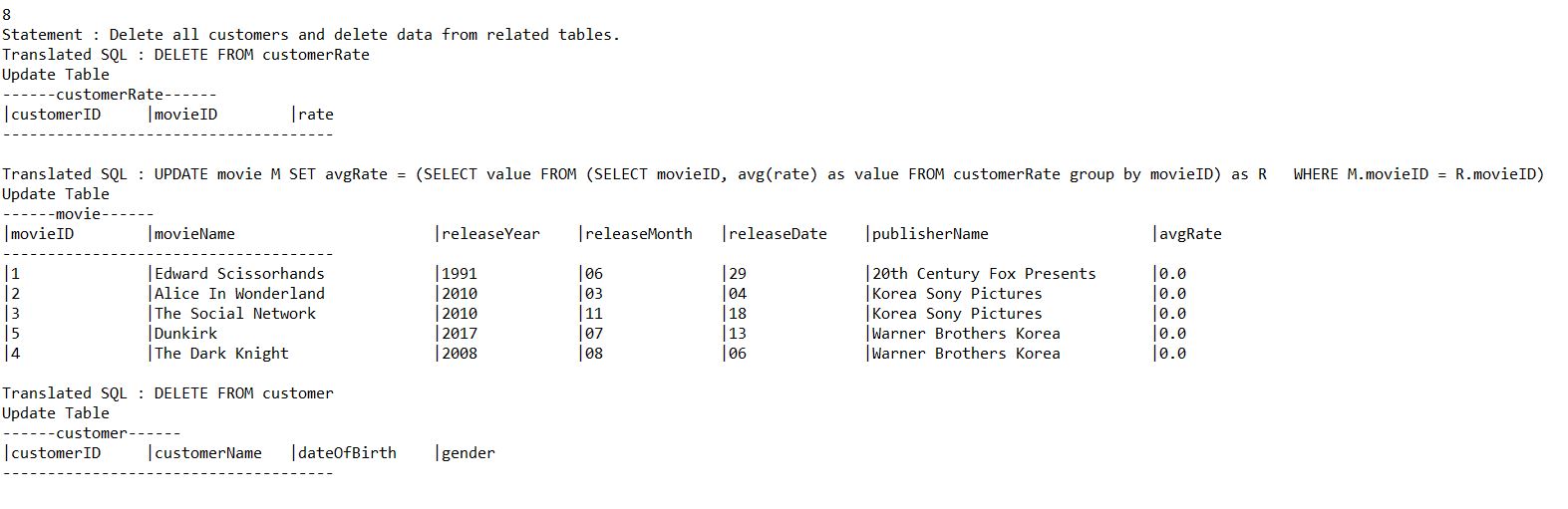
1. Query 및 실행 결과

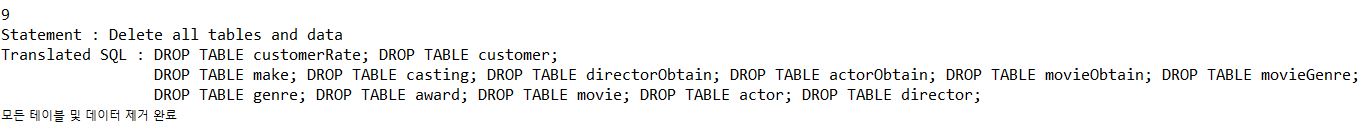












1. 프로젝트를 진행하면서 경험한 것

같은 결과를 내는 쿼리여도 어떤 연산을 이용하는지, 중복을 허용했는지 안했는지 등 여러 조건에 따라 수행속도(성능)의 차이가 생각했던 것보다 많이 난다는 것을 알게 되었습니다.

그렇기에 SELECT 연산을 이용해 DB에 접근해 Data를 가져올 때 join이나 distinct등 cost가 많이 발생하는 연산을 최대한 줄일 수 있게 DB를 설계해야 하는 필요성에 대해서도 생각해 볼 수 있는 기회였던 것 같습니다.

또 High Level에서 기술한 Statement를 SQL로 옮기는 작업 도중에 Subquery의 depth가 별로 차이 나지 않아 보이는 비슷한 Statement일지라도 테이블마다의 특성 때문에(가장 컸던 것이 referenced table인지 referencing table인지) 구현 난이도에 있어서 차이를 보였던 것이 인상적이었습니다. 특히 4번과 5번에서 차이를 느꼈던 것 같습니다.

그 외에도 FROM 절에서 subquery를 이용할 때 모호함을 없애기 위해 aliasing을 꼭 해주어야 하는 작업등 실제 RDBMS를 이용할 때 Application 개발자로서 체크해야 하는 부분에 대해서도 익힐 수 있었던 기회가 된 것 같습니다.

1. 연락처

E-mail: [k941026h@naver.com](mailto:k941026h@naver.com)

Phone: 010-9690-9410