데이터 베이스 프로젝트 2 보고서

컴퓨터공학부

2012-11258

유근국

- 핵심 모듈과 알고리즘에 대한 설명

Java를 이용하여 DB에 접속, DB에 저장된 레코드들을 조작할 수 있도록 하는 응용 프로그램을 구현하였다. 모든 구현은 TicketDB.java 소스파일에 구현하였으며 mariaDB 라이브러리를 자바 프로젝트에 추가하여 내부 함수들을 사용하였다.

구현한 응용 프로그램의 기본적인 알고리즘은 다음과 같다.

1. 제공된 DB에 접속하기 위해 connection 변수들을 초기화한다.
2. 사용자로부터 동작 번호(action number, 1~15)를 입력을 받는다.
3. 입력된 번호가 유효한 동작 번호인지 판단한다.
4. 유효한 동작 번호라면 connection 변수를 통해 DB와 상호작용 하여 해당 번호에 맞는 동작을 수행, 결과를 출력하고, 유효하지 않다면 에러 메세지를 출력한다.
4-1. 동작을 수행하는 중에 에러가 확인되면 즉시 동작을 멈추고 에러 메시지를 출력한다.
5. 2~4의 과정을 종료(15) 입력이 들어올 때까지 반복한다.

위 알고리즘에서 에러인 경우는 스펙문서에 명시된 case 및 “가정한 것들”에 본인이 명시한 case이다.

- 구현한 내용에 대한 간략한 설명

본인은 스펙에서 제시한 세가지 테이블 building, performance, audience 에 더불어 assignment, booking 테이블을 추가하여 테이블 스키마를 구성하였다.

Assignment 테이블은 공연과 공연장의 할당 관계를 저장하는 테이블이다. 따라서 assignment테이블의 column은 building_id, performance_id으로 구성되며, 각각은 building과 performance 테이블을 참조하는 포린키이다. 또한 한 공연장에 여러 공연을 할당할 수 있고, 한 공연이 여러 공연장에 할당될 수 없으므로 프라이머리키를 performance_id로 정하였다.

Booking 테이블은 공연과 관객의 예약 관계를 저장하는 테이블이다. 따라서 Booking 테이블의 column은 performance_id, audience_id, seat_number로 구성되며, performance_id

는 assignment 테이블을, audience_id는 audience 테이블을 각각 참조하는 포린키이다. 이때, 한 공연의 특정 좌석에 대해 중복해서 예약이 불가능 하므로 (performance_id, seat_number) 집합을 프라이머리키로 정하였다.

테이블 스키마를 정의하는데 있어서는 편의를 위해 HeidiSQL 프로그램을 사용하였다.

제시된 스펙 중 데이터 타입 체크는 DB 스키마 단계에서 constraint를 주는 방식으로 할 수도 있으나 본인은 스키마 단계가 아닌 응용프로그램 단계에서 처리하였다. 즉, DB에 새로운 데이터를 저장할 때, 유저에게 받은 인풋을 그대로 query로 요청하지 않고 string은 200자로 자르고, int는 범위를 검사 한 후 query로 요청하였다.

데이터 타입 체크가 아닌 다른 에러 체크의 경우에는 크게 1. SQL constraint violation exception catch, 2. Query response table 확인, 3. 적용 행 수 확인 등의 방법을 사용하였다.

예를 들어 10번 공연배정 구현에서 공연이 이미 다른 공연장에 배정되었다면 에러메세지를 출력해야 하는데 위에서 말하였듯 본인은 assignment 테이블을 두고 performance_id를 프라이머리키로 지정하였으므로 중복된 공연의 할당은 SQL constraint violation exception을 raise 하게 된다. 따라서 이를 catch 하여 해당 에러 메시지를 출력하도록 하였다. 또한 대부분의 레코드의 존재 여부를 확인하는 에러 체크의 경우에는 select query를 보내어 그 결과가 있는지 없는지를 통해 판단하였다.

마지막으로 정의한 모든 포린키는 딜리트 옵션을 cascade로 주어 building, performance, audience 테이블에서 삭제가 발생하면 assignment, booking 테이블에서도 자동으로 삭제되도록 하였다.

- 가정한 것들

- 2번과 12번 동작에서 출력해야 하는 예매자 수(booked)는 예매한 사람 수가 아닌 예매된 좌석 수로 세었다.

- 입력으로 받는 정수 데이터들이 integer 범위를 넘어가면 에러메세지를 띄우도록 처리하였다. (ex, capacity, action number 등)

- 기존의 building, performance, audience 테이블의 프라이머리키는 모두 각각의 id로 정하였다. 즉, id만 다르면(AUTO_INCREMENT 옵션으로 인해 자동으로 다르게된다.) 나머지 column이 모두 같아도 다른 레코드로 인식한다. (ex, 같은 이름, 같은 타입, 같은 가격을 가진 공연이 여러 개 존재할 수 있다.)

- 스펙에는 명시되어 있지 않지만 관객 테이블에 없는 관객아이디로 예매 요청을 하거나

공연 테이블에 없는 공연아이디로 할당 요청을 하는 등의 경우도 에러메세지를 띄우도록 처리하였다. (본인의 테이블 스키마에서 foreign key constraint violation에 해당하는 경우)

- 12번과 13번 동작에서 “해당 공연장이 없다면” 과 “해당 공연이 없다면” 을 말 그대로 해당 공연장이나 공연이 각각 building 과 performance 테이블에 없을 경우로 받아들였다. 즉, “해당 공연장에 할당된 공연이 없다면” 혹은 “해당 공연을 예매한 관객이 없다면” 의 경우에는 에러가 아니라 비어있는 표를 출력하게 하였다.

- 컴파일과 실행 방법

  이클립스 상에서 프로젝트를 열고 TicketDB.java 파일을 src로 등록 한 후 TicketDB를 메인 class로 하여 실행한다.

  혹은 리눅스 쉘 상에서 PRJ2_2012-11258.jar 파일을 바로 java –jar PRJ2_2012-11258.jar 명령을 통해 실행할 수 있다.

- 프로젝트를 하면서 느낀 점

  프로젝트를 시작하기전에 이번 프로젝트2는 프로젝트1보다 훨씬 쉬울거라는 소문을 듣고 굉장히 만만하게 생각했었는데 안일한 판단이었다. 물론 프로젝트1-3을 통해 단련될 대로 단련된 상태라 아주 고생하지는 않았지만 절대로 쉽다고 말할만한 프로젝트는 아니었다.

  이번 프로젝트 뿐만 아니라 이번 데이터베이스 과목을 들으며 수행한 모든 프로젝트에서 공통적으로 느낀 것은 에러처리의 중요성과 어려움이다. 모든 구현의 80프로 이상이 에러처리를 위한 것 이었다고 해도 과언이 아닐 만큼 에러처리가 굉장히 중요한 사안이었다. 하지만 중요한만큼 그것을 모두 커버할 수 있도록 구현하는 것은 정신적으로 너무 큰 스트레스였다. 내가 아무리 잘 생각해서 코드를 짜도 모든 경우를 커버한다고 확신할 수 없다는 것이 가장 힘들었다.

  그래도 한학기동안 실제로 쿼리를 처리하는 간단한 DBMS도 만들어보고 그런 DBMS를 이용하는 응용프로그램까지 만들어보고나니 뭔가 아주 얕게나마 데이터베이스의 활용모습을 체험해본 것 같아서 좋았다. 이번 학기동안 경험한 것들이 부디 훗날 조금이나마 도움이 될 수 있으면 좋겠다.