Database Project 1-2 Report

2013-11431 정현진

0. 개발 환경

다음과 같은 환경 속에서 프로젝트를 진행하였다.

- OS: Windows 10 (Version 1909, Build 18363.778)
- IDE: Eclipse IDE Version 2020-03
- Java: Java 14
- BerkeleyDB: BerkeleyDB Java Edition 7.5.11

1. 핵심 모듈

본 프로젝트에서는 기본으로 주어진 .jj 파일 외에, 다음과 같은 클래스들을 정의하였다:

1. Message
2. ForeignKey
3. Column
4. Table
5. Schema

1-1. Message

프로젝트에서 사용되는 메시지들을 정의하고 출력하기 위한 클래스이다.

1-2. ForeignKey

Foreign key relationship 을 가지고 있는 객체이다. 테이블 이름과 연관된 칼럼의 이름을 가지고 있으며, referential constraints 를 구현하기 위해 사용한다.

1-3. Column

Column 과 관련된 변수 및 메소드들을 가지고 있다. 자신이 참조하는 ForeignKey 객체를 가지고 있고 자신을 참조하고 있는 ForeignKey 의 목록을 HashSet 으로 가지고 있다.

1-4. Table

Table 과 관련된 변수와 그와 연관된 메소드를 담고 있다. 테이블이 가지고 있는 column 들의 생성 순서를 보존하기 위해 LinkedHashMap 을 사용하여 column 들을 저장하였다.

1-5. Schema

BerkeleyDB 환경 설정 및 조작, 스키마 구현, 에러 처리 등 대부분의 기능을 담당하고 있는 클래스이다. Schema 는 단 하나만 존재해도 무방하므로 Singleton 패턴을 사용하여 구현하였으며, 스키마에 존재하는 table 들의 순서를 보존하기 위해 LinkedHashMap 을 이용해서 table 을 가지고 있다.

2. 구현 내용 및 알고리즘

클래스 및 파일들 간의 역할을 분명하게 나누는 것을 목표로 구현했다. MVC 구조와 비슷하게, javaCC 파일은 단순 파싱만을 담당하고, Table, Column, ForeignKey 클래스들은 model 의 역할을 해 해당 객체와 관련된 변수 및 메소드만을 가지고 있다. 그리고 스키마 구현과 관련된 모든 부분을 Schema 에서 처리하게 해, controller 의 역할을 하게 하였다. 마지막으로 쿼리를 처리하면서 출력이 필요한 경우 Message 클래스를 통하도록 했다.

Model 역할을 하는 클래스들은 모두 Serializable 을 implement 해, 해당 객체들을 byte 형태로 직렬화 할 수 있도록 하였다. Column 및 ForeignKey 클래스는 모두 Table 내부에서 사용되므로, BerkeleyDB 에 스키마를 저장할 때는 Table 객체를 직렬화 한 뒤 저장하고, 반대로 불러올 때는 저장된 데이터를 Table 객체로 역직렬화 한 뒤 사용하였다.

프로젝트를 구현하면서 가장 어려웠던 부분은 referential constraints 를 구현하는 부분이었다. 처음에는 Column 에서 모든 과정을 처리했는데, BerkeleyDB 와 연동하는 과정에서 참조키 관계가 없어지는 문제가 발생했는데 해결하지 못했다. 이 문제를 해결하기 위해 ForeignKey 클래스를 따로 만들어 가장 간단한 정보인 테이블과 칼럼의 이름만을 저장했다. 그 뒤 Column 클래스에 referencing 과 referenced 를 ForeignKey 객체를 이용해 저장한 뒤, Schema 에서 referential constraints 를 처리하였다.

마지막으로 쿼리를 처리하는 과정을 간략히 설명하면 다음과 같다.

1. 프로그램이 실행되면 Schema 에서 BerkeleyDB 를 실행하고 저장되어 있는 스키마들을 불러온 뒤 table schema 를 tables 변수에 저장한다.
2. javaCC 파일을 통해 쿼리를 파싱하고, 파싱하는 도중 필요한 값들은 변수로 저장한다. 쿼리의 파싱이 끝나면 Schema 에서 해당 쿼리를 실행한다.
3. Schema 에서는 제약 조건들을 확인하고, 위반하면 에러를 발생시켜 Message 클래스를 통해 에러 메시지를 출력한다. 정상적으로 입력된 경우 해당 쿼리와 관련된 메소드를 실행한다. 이 때 Table, Column, ForeignKey 클래스를 이용한다.
4. Schema 에서 처리가 정상적으로 종료되면, javaCC 파일에서 Message 클래스를 통해 성공 메시지를 출력한다.
5. EXIT; 나 EOF 가 입력되면 Schema 에서 BerkeleyDB 를 종료시키고 프로그램을 종료한다.

3. 구현하지 못한 내용

테스트를 많이 해보지는 못했지만 프로젝트 명세서에 기재되어 있는 내용 중 구현하지 못한 부분은 없는 것 같다.

4. 가정한 것들

프로젝트 1-1 에서는 에러 처리를 잘못된 입력이 들어오면 바로 처리한다고 가정했는데, 이번 프로젝트에서는 쿼리 하나를 모두 입력 받은 뒤 스키마를 처리하기 때문에, syntax error 를 제외하고 스키마와 관련된 에러는 쿼리를 모두 입력한 뒤 처리하도록 하였다.

5. 컴파일과 실행 방법

컴파일은 Eclipse 를 이용하여 진행하였다. Project 1-1 튜토리얼의 내용 그대로 환경 설정을 진행하였으며, je-7.5.11.jar 파일은 프로젝트의 lib/ 폴더에 저장한 뒤 프로젝트 Build Path 에 추가하였다.

.jar 파일을 실행하기 위해서는 .jar 파일과 같은 폴더 내에 /db 폴더가 생성되어 있어야 한다. 해당 폴더는 BerkeleyDB 저장에 사용되는 폴더로, 스키마들을 저장한다. 실행은 CLI 환경에서 “java -jar PRJ1-2_2013-11431.jar” 명령어를 이용하여 실행한다.

6. 느낀 점

구현 방법이 어느 정도 정형화되어 있던 프로젝트 1-1 과 다르게, 이번 프로젝트에서는 스키마 저장 방식을 다양하게 구현할 수 있어 많은 고민과 시행착오를 겪으며 프로젝트의 구조를 점진적으로 수정해 나갔다. 그 결과 완벽하진 않지만 어느 정도 만족스러운 결과물이 나온 것 같다.