Database Project 1-3 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

프로젝트 1-3 에서는 1-2 을 기반으로, 다음과 같은 주요 모듈들을 새로 정의하였다.

1. Type
2. ColumnInTable 및 Value
3. ThreeValuedLogic
4. Record
5. Where
6. SelectUtil

1-1. Type

Column 및 comparable value 에서 사용되는 Type 클래스이다. 기존에는 Column 클래스 안에 Type 에 대한 상수만 정의되어 있었으나, 프로젝트 1-3 에서 Type 의 재사용성이 높아져 따로 분리하였다.

1-2. ColumnInTable 및 Value

프로젝트 데이터베이스 문법의 Comparable Operand 를 구현하는 클래스이다. ColumnInTable 은 프로젝트 1-1 에서 변경한 구조를 반영하였다.

1-3. ThreeValuedLogic

이번 프로젝트에서 Where clause 를 계산하는 데 사용되는 three valued logic 의 구현체이다. Or, and, not 등 각 명령의 계산 방법은 수업의 슬라이드를 따랐다.

1-4. Record

Record 들을 저장하기 위한 클래스이다. BerkeleyDB 에 저장하기 위해 직렬화를 지원하는 Serializable 을 implement 하였다.

1-5. Where

데이터베이스 문법의 WHERE 절을 구현한 클래스이다. Where clause 를 처리하면서 구조를 저장하고, 최종적으로 가장 상위 클래스인 BooleanValueExpression 에서 Three Valued Logic 으로 조건을 계산한다. 구체적인 구조는 해당 파일에 주석으로 기록하였다.

1-6. SelectUtil

기존의 Schema 에서 모든 작업을 처리하는 구조로는 SELECT query 를 처리하기가 불가능해, SELECT query 및 WHERE clause 를 효율적으로 token parsing 단계에서 계산하기 위해 생성한 클래스이다.

2. 구현 내용 및 알고리즘

지난 프로젝트 1-2 에서 MVC 구조를 모방해서 구현하고, 에러 처리는 쿼리를 모두 입력 받은 후에 메시지를 출력하도록 구현했는데, 이번 프로젝트에서는 그 구조가 많이 흐트러졌다.

Insert 의 경우 지난 프로젝트와 같은 구조로 작업이 가능했지만, DELETE 와 SELECT query 는 WHERE 절의 존재 때문에 해당 방법으로 처리하기 어려워, SelectUtil 을 활용해 token 을 parsing 하는 과정에 WHERE 문을 형성하고, WHERE 문을 형성하는 과정에서 또 에러 처리의 일부를 parsing 과정에서 처리하게 하였다.

SELECT 문에서는 여러 테이블을 FROM clause 에 입력했을 때 join 을 어떻게 할 지에 대한 고민을 많이 했다. 초기에는 직관적인 구현으로, 테이블에 row 들을 추가하고, cartesian product 를 구현해서 하나하나 비교하는 방법을 생각했지만 기존에 구현하던 방향이 해당 방법과는 달라 프로젝트 전체 구조를 바꿔야 하는 위험이 있어 선택하지 못했다. 최종적으로 backtracking 을 이용해서 Record 를

join 한 경우의 List 를 생성한 다음, 그 List 를 WHERE 클래스에서 계산해 유효한 경우의 수인지 파악하고, 유효한 경우 출력하는 방법을 택했다.

특이점으로 Record 클래스에서 중복된 value 를 가진 여러 개의 Record 를 저장하기 위해 임의의 숫자를 생성해 _id 값을 클래스 내부 변수로 저장하였다. 해당 값이 없을 때 값은 값을 가지는 Record 들을 직렬화한 값이 같아 중복된 값을 가지는 Record 를 여러 번 insert 해도 한 개만 저장되는 문제가 있었다.

3. 구현하지 못한 내용

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

4. 가정한 것들

프로젝트 1-2 에서 syntax error 를 제외하고 스키마와 관련된 에러는 쿼리를 모두 입력한 뒤 처리하도록 했는데, 이번 프로젝트에서는 WHERE clause 에 대한 구문을 토큰 파싱 과정에서 형성하도록 구현했기 때문에 몇몇 에러는 쿼리를 모두 입력하기 전 처리되는 경우가 있다.

또한 프로젝트 명세에서는 select 문에서 FROM clause 에 중복된 alias 가 입력되는 경우의 에러 처리가 없었는데, MySQL 에서 이를 에러로 처리하기 때문에 해당 경우를 에러 처리하였다. 또한 Foreign key 관계에서 참조하는 Record 의 foreign key 중 하나라도 NULL 값이 있을 경우, MySQL 에서와 같이 다른 column 들은 type 만 같으면 모두 유효한 경우라고 판단하였다.

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. .jar 파일의 실행은 CLI 환경에서 “java -jar PRJ1-3_2013-11431.jar” 명령어를 이용하여 실행한다.

6. 느낀 점

프로젝트를 하며 어려웠던 점은, 처음부터 모든 구조를 설계하고 구현한 것이 아니라 INSERT, DELETE, 그리고 SELECT 순으로 구현을 했기 때문에 기존에 구현하던 구조가 맞지 않는 점이 있으면 구조를 바꿔야 한다는 점이었다. 이번 프로젝트를 하며 구조를 두 세번 정도 바꾼 것 같은데, 처음부터 프로젝트 명세를 보며 설계를 잘 하고 구현을 시작해야 한다는 것을 다시금 느낀 것 같다.

마지막으로 간단하지만 데이터베이스 구현을 직접 하며 데이터베이스에 대한 이해도가 높아질 수 있었던 프로젝트였던 것 같다.