데이터베이스 Project 1-3 Report

공과대학 컴퓨터공학부

2016-12155

이진호

1. 프로그래밍 환경

a) IDE 및 컴파일러

- Eclipse Neon

b) OS

- MS windows 10 Pro

2. 핵심 모듈과 알고리즘 및 코드에 대한 설명

이번 프로젝트는 1-1 프로젝트, 1-2 프로젝트에서는 구현한 프로그램에 3 개의 DML 구문(`insert, delete, select`)을 추가로 구현할 것이다.

먼저 DML 패키지를 만들고 안에 크게 Insert, Delete Select 클래스를 만들고 각 쿼리를 처리하도록 하였다. 또한 각 Value 를 저장하기위해 Value 클래스를 만들고 int 를 이용해 1 은 integer, 2 는 char, 3 은 date, 4 는 null 을 가르키게 하였고 String 으로 value 를 저장하도록 하였다. Record 클래스는 Value 들이 저장된 Arraylist 를 갖고 참조되는지 여부를 알수 있는 ForeignRecordData 정보를 ArrayList 로 갖고 있다.

Value 는 실제로 파일에 저장될 때 type#value 형식으로 저장된다. 예를 들어 type 이 char 이고 'test'를 value 로 가지면 2#test 가 저장된다.

Record 는 저장될 때 Value 들을 '(Single Quote)를 통해 구분한다. 예를 들어 다음과 같다.

[2#test'1#12312'3#2016-11-12'2#sfefsfesf#/@']

Char 는 #을 가질 수 있지만 먼저 '(Single Quote)를 이용해 Value 들을 구분해낸 뒤 처음나오는 #을 이용해 분리해서 값들을 정확히 인식 할 수 있었다. 이는 char 값이 '(Single Quote)를 가질 수 없음을 이용해 구현하였다.

Insert class 는 Insert 구문이 실행되면 tableName, ColumnNameList, ValueList 를 입력 받는다. 먼저 tableName 을 이용해 tableName 의 DBSchema 가 있는지 확인한다. 또한 Value 들의 type 과 Schema 에서 정의된 type 을 비교하고 오류가 있다면 적절한 오류메세지를 출력한다.

또한 foreign key 는 reference 하는 table 이 실제로 그 값들을 primary key 가 있는 지 확인한다. 만약 없으면 오류를 출력하고 있다면 그 레코드에 referencingTableName, primaryKey, notNull, indexList 정보를 "(Double Quote)를 이용해서 구분하여 넣어줬다.

예를 들어 table1 이라는 table 에 2#referenced'1#123'3#2016-11-11' 이라는 record 가 있었고 2#referenced 가 primary key 라고 가정해보자.

이때 table2 에 1#321'2#refereneced'3#2017-11-11'이라는 1#321 을 primary key 로 갖고 2#referenced 를 foreign key 로 갖는 record 가 들어오면 참조 당한 2#referenced'1#123'3#2016-11-11'의 레코드 뒤에 다음과 같이 추가로 ForeignRecordData 의 정보가 쓰여진다.

예) 2#referenced'1#123'3#2016-11-11'"table2"1#321"0"1"

이와 같이 참조당한 record 들에 추가로 정보를 써 주었다.

Delete 문은 `ReferentialIntegrity` 를 체크하고 지울 수 있는 경우 해당 `Record` 를 `Referencing` 하는 `Record` 의 `foreign key` 를 모두 `null` 로 바꾸고 자신이 `referencing` 하는 `record` 들을 다 찾아가 ForeignRecordData 에 해당되는 정보를 지웠다.

Where 문도 따로 패키지와 클래스를 만들고 구현하였다. Parser 에서 입력받는로 BooleanFactor, BooleanTerm, CompOperand 등 클래스를 모두 만들었다. BooleanTest 는 execute 를 메소드를 갖는

추상클래스로 만들고 이를 Predicate 라는 추상클래스와 BoolValExp 클래스가 상속받게 하였다. 또한 Predicate 는 ComparisionPredicate, NullPredicate 가 상속받게 하였다.

Where 가 execute 되면 하위에 해당되는 클래스들이 차례로 execute 가 된다. 그 후 제일 밑바닥에서 에러가 있으면 -1, false 면 0, true 면 1, unknown 이면 2 로 int 타입으로 리턴하게 하였다.

Select 의 경우 from 에 들어온 table 마다 cursor 를 열고 ArrayList 로 관리한다. 이들 table 마다 레코드를 한 개씩 돌면서 Where 절을 체크한다. 그리고 Where 절에서 참으로 나온 레코드들을 주어진 column 에 맞게 출력한다.

3. 구현한 내용에 대한 간략한 설명

Select 문에서 출력할 때 각 column 마다 가장 긴 길이를 갖는 Value 를 기준으로 글자수를 세 "-" 개수를 출력해 더 보기 쉽도록 하였다.

테이블 이름과 컬럼 이름은 대소문자를 구분하지 않는다고 되어 있으므로 모두 lowerCase 로 바꿔서 저장하고 처리하였다.

4. 가정한 것들

두가지 이상의 오류가 동시에 발생하지 않는다고 가정하였다.

Insert 에서 들어오는 테이블의 컬럼 이름은 중복되지 않는다

5. 에러 처리 방식에 관하여

1) Insert Query 에서 column name list 의 크기와 Table Schema 의 Column Definition List 크기 다른경우

-> Column name list 순서대로 Value 들을 집어 넣고 Column name list 에 명시되지 않은

값들에는 null 값을 넣었다.

2) SelectTableExistenceError

- select * from none as n; 라고 존재하지 않는 table 을 참조한 경우 none(as 앞에 tablename) 을 이용해 오류를 출력하였다.

3) SelectColumnResolveError

- select none as no from tableOne; 이라는 쿼리로 SelectColumnResolveError 가 발생할 경우 none(as 앞에 쓴 columnName)을 이용해 오류를 출력하였다.

6. 컴파일과 실행 방법

eclipse 에서 컴파일 하였으며 윈도우 cmd 에서 java –jar 명령어를 통해 실행시켰다. Exit;을 통해 프로그램을 종료한다.

7. 느낀점

Insert 문에서 referential constraint 를 어떻게 체크하는지 고민해볼 수 있었다.

Where 절에서 또한 오류가 없는지 체크하는 과정을 통해 실제 DBMS 에서는 어떻게 구현되어 있을지 고민해 보았다.

프로젝트를 하며 전반적으로 DBMS 안에 구현방식에 대해 더 고민해볼 수 있었다.