2015 fall 4190.310-001 Database

Project 1-2: Implementing DLL

2013-11395 김희훈

1. Introduction

이번 프로젝트에서는 Berkeley DB를 이용, 테이블 스키마를 생성, 불러오기, 수정, 삭제하는 기능을 구현하여 CREATE TABLE, DROP TABLE, DESC 구문에 대해 실제 동작을 하도록 SimpleDBMS를 개선하였다.

2. Implementation

파서 동작만 JavaCC로 구현했던 이전 프로젝트와 달리, 이번 프로젝트는 Table과 Column 정보를 메모리 상에 저장하고 이를 바이트 스트림으로 바꾸어 Berkeley DB에 저장하는 등의 동작이 필요했다. 그래서 여러 클래스를 작성하였다.

A. Column

column 이름, 데이터 타입, char 타입인 경우 길이, nullable한지, primary key인지, foreign key인지, 어떤 column을 참조하는지, 어떤 column에 의해 참조 당하는지를 저장한 클래스이다. 참조 정보는 Foreign 클래스에 담았으며, 여러 column에 의해 참조 당할 수 있으므로 set을 써서 관리하였다.

B. ColumnBinding

Berkeley DB는 바이트 기반 key value DB이므로 column을 저장하기 위해서는 바이트로 인코딩 및 디코딩 해주는 Class가 필요하여 TupleBinding을 상속하여 구현하였다. Column에서 언급한 순서대로 인코딩하였으며, Column명이 제일 앞에 있기 때문에 Berkeley DB에서 바이트 기준 사전 순 정렬을 했을 때 자연스럽게 Column 이름으로 정렬된다. 그래서 (tableName)으로 검색하거나 (tableName, columnName)으로 검색하는 게 모두 가능하다. DB 구조는 다음과 같다.

key	value								
table name	column name	data type	data length	is notnull	is primary	is foreign	referencing column (if foreign)	referenced count	referenced columns (가변길이)

C. Foreign

참조 정보를 담기 위해 테이블명, 칼럼명만 가지고 있는 단순 클래스이다. 사전 순 정렬이 용이하게 Comparable 인터페이스만 맞춰주었다.

D. Table

테이블명과 테이블에 속한 Column 정보를 갖고 있는 클래스이다. column명으로 column을 빠르게 접근할 수 있도록 set으로 관리하였다. addColumn, addPrimaryKey, addForeignKey 메소드가 구현되어 파싱하면서 메모리 상에 테이블 스키마를 구성하는 역할을 한다. 자신을 관리하는 SchemaManger에 대한 레퍼런스도 갖고 있어서, 다른 테이블의 특정 칼럼이 primary 인지 확인하거나, 마지막에 테이블을 Berkeley DB에 저장하는 등 DB 접근이 필요한 작업에 이용한다.

E. SchemaManager

Berkeley DB를 이용하는 모든 작업을 담당하는 클래스이다. 테이블이 존재하는지 확인하는 tableExists, 테이블 한 개 또는 모두를 삭제하는 dropTable[s], 테이블을 불러오는 getTable[s], 외부 키를 추가/삭제하는 add/removeForeign, 칼럼 한 개를 추가 또는 불러오기 하는 put/getColumn 메소드가 구현되어 있다.

F. Message

메시지 스펙이 복잡해졌기 때문에 메시지 코드와 printMessage 함수를 빼내어 클래스로 구성하였다. 메시지 코드, 테이블 이름 등을 찍기 위해 필요한 추가 스트링, 프롬프트 출력 여부를 저장하여 전달할 수 있으며, static 메소드로도 메시지 출력이 가능하다.

3. 가정한 것들

- 일관성을 위해 테이블이 없을 때 DESC *; 실행 시 가로줄이 출력되도록 하였다.
- DESC 구문에서 칼럼명은 22자까지만 출력되도록 하였다.
- Column definition보다 key constraint가 먼저 온 경우 NonExistingColumnDefError로 처리하였다.
- 같은 칼럼에 대한 foreign key constraint가 두 번 정의된 경우 DuplicateColumnDefError로 처리하였다.

4. 실행 방법

Java –jar 옵션으로 runnable jar 파일을 실행하면 된다. 단, db 폴더가 jar 파일과 같은 폴더에 존재해야 한다.

5. 느낀 점

Schema 같은 정보를 Berkeley DB와 같은 제약된 key value DB에 생성, 수정, 삭제가 가능한 형태로 넣을 수 있도록 설계하는 것이 쉽지 않았다. 여러 개의 DB로 나누거나, E-R model을 흉내 내는 등 다양한 방법을 시도해 보았으나, 결국에는 table name - column으로 구성된 한 개의 DB에 모두 넣고 Berkeley DB의 default ordering이 바이트 사전 순이라는 것을 이용한 방식을 취하게 되었다.

또한 Schema를 메모리상에 저장할 때 여러 시행착오가 있었다. 처음엔 AST(Abstract Syntax Tree) 노드를 구성하면서 Bottom-up 방식으로 정보를 전달했는데, DB 동작보다 오히려 파서 동작에 묶여 구현이 쉽지 않았다. 그래서 새로운 시도로 상위 노드에서 객체를 생성한 후, 하위 노드로 넘겨주고 아래쪽에서 setter 등으로 정보를 입력하는 Top-down 방식을 사용하게 되었다. C-like language 컴파일러를 작성할 때는 전자가 더 편했었는데, application마다 알맞은 구현에 차이가 있는 것 같다.