데이터베이스 Project 1-1

<SQL Parser>

2018-13627 김기덕

1. Introduction

   본 Project는 python shell로 입력받은 sql 형태의 query를 parse 해보는 것이다.
   Python에서 제공하는 lark를 사용해서 parser를 구현하였다.
   추후에 schema와 data를 관리하는 프로그램을 만들기 위한 기본 과정이라고 볼 수 있다.

2. 핵심 모듈

   SQL Parser에서 핵심적으로 만들어야 될 것을 나열해보면 첫째로 input을 받는 부분, 둘째로 받은 input을 parse하는 부분, 마지막으로 parse된 input을 분석하여 어떤 종류의 query인지 출력하는 부분으로 나눌 수 있다.

3. 알고리즘 및 구현 방식

   A. Input 받기

      Input의 끝이 ';' 이 아니면 계속 받아야 하므로 Input의 끝을 확인해서 ';'이 아니면 계속해서 Input을 받는 것으로 구현하였다. 그 과정에서 이전 Input과 공백으로 연결해 주도록 하였다.

   B. Input Parse 하기

      A에서 받은 Input을 ';'까지로 끊어서 n개의 query string으로 나눈 후 각각에 대해 grammar.lark에 저장되어있는 파일로 Lark를 선언해 내장되어있는 parse 함수로 parsing을 진행하였다.

   C. Query 출력하기

      B에서 query string을 parse해서 Lark에서 제공하는 Tree의 형태로 바꿔놓았다. 이렇게 만든 Tree들을 직접 만든 MyTransformer를 통해 어떤 query인지 찾아 출력을 할 수 있도록 만들었다. MyTransformer는 Lark에서 제공하는 Transformer를 따르는데, 어떤 data를 찾아 그에 대한 일을 수행하고 싶다면, data의 이름대로 함수를 선언해서 내부를 구현하면 된다.

4. 전제 조건

   구현하는 과정에서 여러 가정을 하였는데, 우선 input에서 엔터를 칠 때, 위의 문장과 아래의 문장은 그냥 이어서 보는 것이 아니라 띄어쓰기가 하나 된 상태로 생각을 하였다. 예를 들어 윗줄에 select, 아랫줄에 id를 치면 select id 와 같은 문장으로 간주를 하였다. 또한, 현 상황까지 봐도 충분히 어긋난 sql 문장이지만, ';'로 끝내기 전까지는 굳이 판단하지 않기로 하였다.

5. 느낀 점

   Lark를 사용하지 않고 직접 custom하게 만들어서 Parsing을 했으면 경우의 수도 많아 길고 복잡한 코드가 예상되었는데, Lark 문법을 사용해서 간단하게 해결을 하는 방법을 새로 알게 되어서 유익한 경험을 했다는 느낌이 들었다.
   이번 프로젝트는 직접 shell을 구현하는 느낌이었기 때문에 입출력에 항상 shell처럼 'DB_student_id > ' 와 같은 것들이 기본적으로 등장해야 한다. 분명 반복되는 행동이라 방법이 있을 것 같기는 하지만 현재 구현한 상태로는 매번 입출력시마다 'DB_student_id> '를 직접 출력해주었다. 틀린 것은 아니지만 매번 이렇게 출력하기에는 효율이 떨어지기 때문에 더 간단한 새로운 방법을 알고 싶다는 생각이 들었다.