**데이터베이스 PRJ 1-1 보고서 2023-11398 윤석민**

* grammar.lark

drop table, Explain 등의 query는 제공된 스켈레톤의 형식으로 구현했다.   
Update query에서의 EQUAL이 where\_clause의 형식에 포함되어 중복이 발생했다. 그래서 syntax error가 발생하는 문제가 있었는데, SELECT query 아래에 정의된 where\_clause 안의 COMP\_OP 대신 oper : COMP\_OP | EQUAL로 중복을 피할 수 있도록 수정하여 중복 문제를 해결했다.

* run.py

run.py를 입력 / 파싱 / Transformer 세 가지 모듈로 나누어 구성했다.

1. 입력

입력(Reading 함수)의 경우, 한 입력에 온전한(세미콜론으로 끝나는) 하나의 query만 들어오는 것이 아닌 여러 queries가 여러 입력에 거쳐서 들어온다는 점을 고려해야 한다. 이를 위해 빈 List를 만들고 세미콜론으로 끝나는 인풋을 받기 전까지 List에 입력 값을 계속 추가해준다.   
여러 번의 입력이 query 별로 끊어진 게 아닌, query의 구성요소가 중간에 끊어져 입력되는 경우도 있기에 list의 내용을 join을 통해 한 줄로 합치고, 세미콜론을 기준으로 나눈 새로운 list를 만든다. 이 list는 입력 받은 온전한 query를 각 요소로 가지고 있으며, Query 별 내용이 담긴 list가 Reading()의 return값이 된다.

1. 파싱

파싱(Parsing 함수)의 경우, 입력(Reading)에서 반환한 list와 grammar.lark를 읽어 온 sql\_parser, notError를 파라미터로 받는다. 함수 안에서는 입력 받은 list의 요소 하나씩을 처음부터 반복문을 실행한다. .parse()함수로 입력 받은 query를 하나씩 파싱하고 transform()을 통해 그에 해당하는 query의 이름을 가진 Transformer class의 함수를 실행한다. Query sequence를 입력으로 받을 때, 중간에 잘못된 query가 있으면 에러 메세지를 출력하고 파싱이 종료되어야 한다. 이를 위해 UnexpectedInput 에러가 발생되면, notError가 false로 바뀌게 되어 반복문이 종료된다. 새로운 input이 들어올 때는 notError = True로 설정해 다시 실행될 수 있도록 한다.

1. Transformer

Parsing에서 실행한 transform() 함수를 통해, 사용자가 입력한 쿼리의 이름을 가진 함수를 실행한다. 각자의 이름 + requested만 출력하면 되기 때문에, 공통으로 사용할 출력 함수( print\_query(self, query\_name) )를 만들고 query 각각에 print\_query(출력할 쿼리 이름)을 써주어 출력했다.

* 느낀 점

Update query에서 발생한 중복 문제를 해결하는데 시간을 많이 들였다. Update query의 스켈레톤 자체에는 문제가 없을 것이라 생각하여 EQUAL이 중복되는 부분인 where\_clause의 수정이 필요할 것이라 생각해 수정을 통해 해결했다. 문제 없이 잘 작동하지만 기존에 먼저 제공한 Select query에 속한 내용을 수정하는 것이 올바른 구현일까라는 의구심이 들었다. 강의에서 제공하는 표준 답안이 궁금해졌다.