Project 1-1 Report

핵심 모듈과 알고리즘에 대한 설명

- `grammar.lark`
  - Lark 라이브러리가 query를 파싱할 수 있도록 돕는 grammar 파일이다.
  - DBMS에 주어질 command들이 어떤 형태를 띌지 재귀적으로 정의되어 있다.
  - 우리는 아직 구현되지 않은 쿼리들(insert, drop table, describe, desc, show tables, update tables)을 쿼리의 형태에 맞게 재귀적으로 정의해주면 된다.
- `run.py`
  - `MyTransformer`
    - Lark 라이브러리에서 주는 `Transformer` 클래스를 상속받으며 파싱된 쿼리를 처리할 수 있도록 하는 클래스이다.
    - 아직 쿼리를 직접 실행하는 과정은 필요하지 않으므로, 각 함수에는 해당 형태의 쿼리가 호출 되었다는 print만 실행해주면 된다.
  - `DatabaseManagementSystem`
    - grammar loading, prompt 실행 등 전체적인 DBMS의 실행에 관련된 함수들이 들어있다. 굳이 클래스에 들어있을 필요는 없지만 정돈된 코드를 위해 정의하였다.
    - 사용자에게서 커멘드를 받고 쿼리를 파싱 및 실행하는 함수를 실제로 호출하는 클래스이다.

구현한 내용에 대한 간략한 설명

- `grammar.lark`
  - 과제 명세에 어떤 형태로 query 문법이 정의되는지 잘 나와있다. 해당 정의에 맞게 grammar 파일에 쿼리를 정의해주면 된다.
- `run.py`
  - `MyTransformer`
    - 각 종류의 쿼리 변수명과 동일한 함수명을 주면 상위 클래스에서 파싱을 해준 뒤 우리가 정의한 함수를 호출해준다. 우리는 해당 쿼리를 처리했다는 print문

만 넣어주면 된다.

- `DatabaseManagementSystem`
  - 가장 처음 사용자에게 쿼리를 받을 때 세미콜론을 기준으로 문자열을 split한다.
    - 만약 가장 마지막 블록에 있는 문자열이 스페이스바를 제외하고 빈 문자열이라면 다음 프로세스를 진행하고, 아니라면 계속 입력을 받아준다.
  - 그렇게 잘라진 쿼리 리스트를 가지고 각각 파싱을 진행한다.
    - 만약 진행하다 파싱 에러를 만난다면 파싱을 그 자리에서 멈춘다.
  - 파싱된 리스트의 원소들을 돌면서 transform을 진행한다.
    - 만약 파싱 에러를 만났었다면 syntax error를 출력한다.
    - 만약 command를 transform한 결과가 EXIT이라면 입력받기를 종료한다.

구현하지 못한 내용

- 없다.

프로젝트를 하면서 느낀 점 및 기타사항

- Lark파일에 쿼리 문법을 재귀적으로 정의해보면서, 어떤식으로 쿼리를 작성해야 하는지에 대한 감을 잡은 것 같다.
- 또한 DBMS있기 이전에 사용자가 얼마나 불편했을지 체감했고, DBMS가 데이터베이스 시스템의 어떤 부분을 abstract하고 있는지 알 수 있었다.