Project 1-2: Implementing DDL & Basic DML

본 프로젝트는 SQL 파서 프로그램을 확장하여, 스키마를 저장하고 접근하는 DBMS 의 기본적인 처리를 구현하는 것이 목표이다.

핵심 모듈

- **grammar.lark 파일**: EBNF (Extended Backus-Naur Form) 형식으로 SQL 문법을 정의한다. Lark API 를 사용하면 이 파일에서 정의된 문법을 기준으로 SQL 쿼리를 파싱하고 그 결과를 AST (Abstract Syntax Tree)로 변환할 수 있게 된다.
- **Sql_transformer**.py 파일: Transformer 클래스로는 이를 상속한 SQLTransformer 클래스를 만들어 파서에서 생성한 AST 를 처리한다. 입력한 구문에서 핵심이 되는 테이블이나 레코드 정보는 dictionary 에 저장하여 반환한다.
- **Data_model**.py 파일: 스키마와 레코드의 자료구조가 정의되어 있고, BerkeleyDB 의 DB 객체를 조작하는 wrapper 와 이를 처리하는 DBMS 클래스가 구현되어 있다. DBMS 클래스는 CREATE TABLE, DROP TABLE, EXPLAIN/DESCRIBE/DESC, SHOW TABLES, INSERT TABLE, SELECT 구문을 처리한다.
- **Messages**.py 파일: DBMS 클래스가 SQL 명령을 수행할 때 성공을 알리거나 에러를 띄우는 등 메시지들을 모아두었다.
- **run.py 파일**: 쿼리 시퀀스를 여러 개의 구문으로 나누고, 각 쿼리별로 동작을 수행한다. Lark 라이브러리로부터 Lark 클래스를 불러와 grammar.lark 파일에서 정의된 문법을 기준으로 하는 파서를 생성하고, SQLTransformer 를 불러와 파서가 생성한 AST 를 해석하여 필요한 데이터를 추출한다. 이때 가져온 데이터로 DBMS 클래스에서 해당하는 함수들을 호출한다.

구현 내용 및 알고리즘

- **Sql_transformer**.py 파일: Transformer 클래스는 AST 를 bottom-up 순서로 순회하므로, 리프 노드를 발견한 경우에는 이를 파싱한 결과물을 반환하고, 상위 노드, 특히 쿼리를 직접적으로 식별하는 노드에서는 하위 노드들로부터 수합한 데이터를 구문, 테이블, 레코드 정보로 분류하여 dictionary 에 저장한다. 그리고 가장 상위 노드에 다다를 때 이 dictionary 를 반환한다.
- **Data_model**.py 파일: 스키마의 메타데이터를 별도의 DB 파일에 저장, 관리하고 (Metadata schema), 하나의 DB 파일에 하나의 테이블의 레코드를 모두 담는 (one DB-one schema) 방식을 사용했다. 그 이유는 BerkeleyDB 는 하나의 DB 에 데이터를 key-value pair 형식으로 저장하는데, DB 안에 테이블 키와 레코드 키가 혼재할 경우 둘 중 하나만을 탐색하고자 할 때 비효율성이 발생하기 때문이다. 따라서 DBMS 에서 메타데이터만을 관리하는 DB 인스턴스를 지속적으로 관리하고, 새로운 테이블을 생성하는 등 개별 테이블에 접근이 필요할 때 해당 DB 를 생성하거나 열어서 관리한다. 이때 테이블 메타데이터는 Table 클래스, 레코드는 Record 클래스로 구현하여 객체지향성을 만족시켰다.
- run.py
  - **main 함수**:
    1. grammar.lark 파일에서 정의된 문법을 기준으로 Lark 로부터 파서를 생성한다.
    2. SQLTransformer 의 인스턴스를 생성한다.
    3. exit 변수가 참이 아닐 때,
       i. 쿼리(시퀀스)를 입력받고 process_query_sequence 함수에서 이를 쿼리의 목록으로 처리한다.
       ii. (목록 내 각) 쿼리에 대해 parse 함수를 실행하여 쿼리로부터 필요한 데이터를 추출한다.
       iii. 메시지가 "exit"일 경우에는 exit 변수를 거짓으로 바꾸고 루프를 종료하고, 그렇지 않을 경우에는 구문이 지시하는 대로 스키마와 데이터를 관리한다.
       iv. parse 함수를 실행하는 과정에서 틀린 구문이 발견될 경우 에러 메시지를 출력하고, 쿼리들이 남아있을 경우에는 더이상 처리하지 않는다.
       v. A 부터 반복한다.

느낀 점 및 기타사항

데이터베이스를 설계하는 과정을 경험할 수 있어 많은 것들을 고려하게 되었다.