project 1-3 report

2023-13751 / 자유전공학부 / 김하람

1. 핵심 모듈 및 알고리즘

(1) SQL 파서 - Lark

SQL 문장을 파싱하기 위해 Lark를 활용하였다. grammar.lark 파일에 SQL의 문법을 정의하고, Lark 파서로 이를 파싱하여 파싱 트리(Tree)를 생성한 뒤, 이를 MyTransformer 클래스에서 처리한다. 이 구조는 SQL 입력 → 파싱 트리 생성 → 실행 흐름으로 이어지는 전통적인 컴파일러 구조와 유사하다. project 1-2와 비교하여, select 문에 where 절과 order by 그리고 join 관련 문법을 추가로 정의하였다.

(2) 메타데이터 및 데이터 저장 - Berkeley DB

Berkeley DB를 통해 테이블 스키마(metadata.db)와 각 테이블의 데이터({table_name}.db)를 저장한다. 키-값 형태로 데이터를 다룰 수 있기 때문에 row ID를 key로, JSON으로 직렬화된 데이터를 value로 저장하는 구조를 사용하였다.

(3) 명령어 실행 흐름 - Transformer

MyTransfotmer 클래스는 Lark.Transformer를 상속바다 SQL 명령어의 트리 구조를 직접 해석하고 실행한다. 주요 명령어는 다음과 같다.

create_table_query	테이블 정의, 기본키/외래키 제약 조건 검증 및 저장
insert_query	값 검증, NOT NULL 및 기본키 중복 검사, row 저장
select_query	WHERE 조건 평가, JOIN 수행, ORDER BY 정렬, 결과 출력
delete_query, truncate_query, drop_table_query	데이터 삭제 관련 로직
rename_query	테이블 이름 변경 및 외래키 참조 업데이트
show_query, explain_query	메타정보 출력

(4) where 절, join 절

where 절의 조건은 where_clause_check, predicate_check 함수를 거쳐 수행되며, 파싱된 트리를 기반으로 구문을 해석하여 비교 연산을 수행한다.

join은 inner_join 함수에서 구현되어 있으며, on 이후의 조건에 따라 중첩시키며 테이블 간 행을 병합한다. 이는 소규모 데이터에 적합한 단순한 Nested Loop join 알고리즘이다.

2. 느낀 점

이번 프로젝트는 SQL 엔진의 내부 동작 장식에 대해 깊이 있게 이해할 수 있는 좋은 기회였다. 직접 SQL 문법을 정의하고 파싱하는 과정은 마치 미니 컴파일러를 구현하는 느낌이었고, 특히 파싱 트리를 순회하며 논리적으로 명령어를 처리하는 부분에서 이론이 실제 코드로 어떻게 연결되는지 알 수 있었다.
JOIN과 WHERE를 처리하는 과정에서 간단해 보이는 SQL문 아래 내부 로직은 얼마나 복잡한지 체감할 수 있었다. 특히 테이블 간 조인을 처리하면서 테이블 스키마 관리, 컬럼 충돌 처리, 외래키 검증 등의 세부 구현이 얼마나 많은 디테일을 필요로 하는지 배울 수 있었다.
Project 1-1에서 시작해 Project 1-3에 이르기까지, 점차 기능을 확장해 나가는 과정을 거치며 전체 구조의 확장 가능성을 염두에 두고 프로그램을 설계하는 것이 얼마나 중요한지 절실히 깨달았다. 과제를 진행하면서 고려해야 할 요소가 많아지고 코드의 길이도 길어지며, 점차 코드가 복잡해지고 얽히는 것을 느꼈고, 결국 중간에 구조를 전면적으로 개편하게 되었다. 이러한 경험을 통해, 처음부터 모듈 간의 역할과 흐름을 명확하게 정의하고 관리하는 것이 유지 보수성과 확장성을 확보하는 데 핵심이라는 점을 배웠다. 과제를 늦게 내어 점수가 깎이는 점이 아쉽지만, 이번 기회를 통해 한 단계 더 성장할 수 있었다.

3. 추가 구현 사항

- inert 시 primary key와 값이 겹치는 경우 아레와 같은 에러 메시지가 출력되도록 하였다.
  Insert has failed: duplicate primary key