Database Project #1-3 Report

2019-18499 김준혁

핵심 모듈과 알고리즘

1. table 구조의 형태

Comparison에서 발생하는 “Unknown” 및 type에 의한 comparison 불가 에러 등을 잡기 위해 table 구조의 형태를 변화시켰다. table을 5개의 list를 원소로 갖는 list형태로 제작하였으며, 그 특성은 다음과 같다.

Index	Type	Value
table[0]	1차원 list	join된 table_name의 list
table[1]	1차원 list	column_name의 list, 0번은 “0,index”로 고정.
table[2]	2차원 list	각 index 및 column에 해당하는 data의 list
table[3]	1차원 list	각 column의 type list
table[4]	1차원 list	각 data의 comparison value list (True, False, Unknown) 이 값이 True일 경우에만 printTable에서 출력됨.

에러를 표시하기 위해, 에러 또한 table 또는 table1과 같은 이름으로 반환되는데, 이는 “@error:”로 시작하는 string 형태로 반환되게 하여 error가 발생함을 확인할 수 있도록 하였다.

2. 추가한 주요 함수들 (여러 쿼리에서)

check_error(value) : value가 table인지, error인지 구분하여 error일 경우 True를 반환하는 함수

where_process(table, where_clause) : where절을 수행하는 함수. 다수의 process 함수를 통해 작동

join_table(table1, table2) : from절을 수행하는 함수. 단순한 join을 실행한다.

select_process(table, where_clause) : select절을 수행하는 함수. 일부 column, data만 저장하는 식

구현한 내용

1. insert

insert_query 함수 내에서 column, value checking 및 type checking 등의 에러를 모두 체크한 이후에 에러가 없을 경우에 대해서 DB에 해당 데이터를 insert하는 식으로 구현하였다.

2. delete

먼저 해당 table의 존재를 확인한 이후, where절이 존재할 경우, where_process 함수를 통해 where 절의 내용을 수행한다. 반환값은 check_error를 통해 error인지 확인, error인 경우 printError를 통해 에러 메시지 출력, 아닌 경우 DB로부터 해당하는 데이터(table[4][idx]가 “True”인 데이터)를 제거

3. select

먼저, select절과 from절의 column과 table을 각각의 list에 따로 저장을 해둔다. 간단한 table 존재 확인 및 joinTable을 통해 from절을 수행하여 table을 join한다. 이후 where절이 존재할 경우, where_process 함수를 통해 수행한다. 에러 체킹을 한 번 한 후에, select 절에 ‘*’이 아닌 입력이 들어올 경우, select_process 함수를 통해 select절을 수행, 에러를 한 번 더 체킹해준다. 이후 printTable 함수를 통해 결과가 “True”인 data만 출력하는 식으로 전체 select 쿼리를 수행한다.

4. 다수의 process 함수

process 함수들이 꽤 많은데, 이들 모두를 설명하기에는 과잉이라고 판단하며, 함수의 이름을 통해 어떤 것이 구현된 함수인지 충분히 알 수 있다. 실행에 대한 추가적인 설명은 없으며, 코드에 넣은 주석으로 충분하다.

느낀 점 및 기타사항

기존에 PRJ1-2에서 만들었던 table 형태에 예상 외의 2개의 value가 더 필요했다. 그래서 추가한 것이 table[3], table[4]에 해당하는, column type과 data available(T/F/U)였다. 그래서 사실 순서상으로는 약간 어거지로 이렇게 붙여야 하나라는 생각이 있었지만, 이를 수정하기에는 이후 생산성에 비해 소모값이 크다고 판단, data를 의미하는 table[2] 이후에 덧붙이는 방식으로 구현하게 되었다. 만약 이들의 필요성을 미리 알았다면, 아니면 조금 더 복잡한 형태라 생산성이 필요할 경우 이 순서들을 조정하고, 더 필요한 값들을 추가했을 것 같다.

where clause가 이번 PRJ에서 가장 골머리를 썩힌 부분이라고 생각한다. 구현 자체는 그렇게 어려운 것이 없었지만, 어떻게 이를 구현할 것인가 즉 구조에 대해 많은 고민을 했다. 이들을 같은 함수 내에서 boolean_XX에서 XX에 따라 수행을 달리할지, 이를 모두 분리해서 각각의 함수로 만들지 고민을 했었다. 특히, 괄호로 둘러싸여 boolean_XX로 회귀할 때가 가장 고민됐다. 하지만 너무 복잡하게 생각하지 않고 그냥 함수를 분리한 덕분에 그나마 구조가 쉬워지게 되었다고 생각한다. 그리고 사실 그것을 구현하면서 에러가 많이 나오는 것에 대해 걱정이 많았는데, 다행히도 fatal한 문제가 발생하지 않아서 천만다행이라고 생각한다.