Project 1-3: Implementing DML

Due: 2024/5/21 (TUE), 11:59 PM

1. Project Overview

이번 프로젝트의 목표는 프로젝트 1-1 및 1-2에서 구현한 DBMS를 확장하여 다음 DML 구문들을 처리할 수 있도록 하는 것이다.

- INSERT
- DELETE
- SELECT

2. Requirements

이하의 조건들을 만족하도록 run.py 파일을 작성한다.

- 2.2장에 명시된 모든 구문들을 받아 올바르게 처리할 수 있어야 한다.
- 2.3장의 구현은 **필수가 아니나**, 올바르게 구현 시 프로젝트 1-1 및 1-2에서의 감점을 만회할 수 있는 추가 점수가 **최대 10점** 부여된다.
- 메시지 정의 문서(2024_1-3_Messages.docx)는 DBMS가 사용할 메시지의 종류와 그 내용을 정리한 문서이다. 이를 참고하여 상황에 맞는 메시지를 프롬프트 뒤에 표시해야 한다.
  - ✓ 예시) DB_2024-12345> No such table
- 정의되지 않은 요소의 구현
  - ✓ 메시지 정의 문서에서 정의된 오류 유형 이외의 유형이 추가로 필요하다고 판단될 경우 해당 유형의 이름과 메시지를 직접 정의하고 보고서에 명시한다.
  - ✓ 그 외 본 문서에 정의되지 않은 요소를 구현할 경우 **MySQL**을 기준으로 구현하고 보고서에 명시한다.
- 기타 유의 사항
  - ✓ 여러 오류 유형에 포함되는 구문의 경우 해당하는 오류 메시지 중 1개를 출력한다
  - ✓ 정답 외 불필요한 출력이 추가로 나올 경우 감점된다.
  - ✓ Berkeley DB의 **SQL API 사용은 금지**된다.

2.1. Prerequisites

프로젝트 1-3에서는 이미 생성된 table 내에서 데이터를 조작하는 기능을 구현한다. 따라서 본 과제에서 사용될 DBMS는 CREATE 문으로 table을 올바르게 생성할 수 있어야 한다. 아래 4개의 구문이 문제없이 실행되어 4개의 스키마가 만들어져야 한다. **아래 구문 외의 CREATE 구문은 평가하지 않는다.**

create table students (
  id char (10) not null,
  name char (20),
  primary key (id)
);
	create table lectures (
	  id int not null,
	  name char (20),
	  capacity int,
	  primary key (id)
	);

create table ref (
  id int,
  foreign key (id) references lectures (id)
);
	create table apply (
	  s_id char (10) not null,
	  l_id int not null,
	  apply_date date,
	  primary key (s_id, l_id),
	  foreign key (s_id) references students (id),
	  foreign key (l_id) references lectures (id)
	);

2.2. SQL Queries

본 장에서는 각 구문에 대한 정의, 처리 방법과 입출력 예제를 제공한다.

INSERT문과 DELETE문의 **Primary key/Foreign key constraint 위반에 대한 오류 처리는 필수가 아니며,** 이에 대한 자세한 사항은 2.3장에서 설명한다.

2.2.1. INSERT

정의

```
insert into table_name [(col_name1, col_name2, … )] values(value1, value2, …);
```

실행 예시

```
DB_2024-12345> insert into account values(9732, 'Perryridge');
DB_2024-12345> 1 row inserted
```

구현 요구사항

- 올바른 INSERT 문의 실행
  - ✓ 적절한 컬럼에 입력된 값을 삽입하고, `InsertResult` 메시지를 출력한다.
  - ✓ `char` 컬럼 타입에 허용하는 최대 길이보다 긴 문자열을 삽입하려 할 경우, 길이에 맞게 자른 (truncate) 문자열을 삽입한다.
- 쿼리 오류에 대한 종류 별 메시지 출력
  - ✓ 삽입할 테이블이 존재하지 않을 경우, `NoSuchTable` 메시지를 출력한다.
  - ✓ 삽입할 데이터의 타입이 맞지 않는 경우, `InsertTypeMismatchError` 메시지를 출력한다.
    - 지정된 컬럼과 값의 개수가 다른 경우
    - 지정된 컬럼과 값의 타입이 맞지 않는 경우
    - 컬럼을 명시하지 않았는데, 입력 값 개수와 해당 테이블의 attribute 수가 다른 경우
  - ✓ `null` 값을 가질 수 없는 컬럼에 `null`을 삽입하는 경우, `InsertColumnNonNullableError(#colName)` 메시지를 출력한다.
  - ✓ 존재하지 않는 column에 값을 삽입하는 경우, `InsertColumnExistenceError(#colName)` 메시지를 출력한다.

2.2.2. DELETE

정의

```
delete from table_name [where clause];
```

실행 예시

```
DB_2024-12345> delete from account where branch_name = 'Perryridge';
DB_2024-12345> 5 row(s) deleted
```

DELETE 문 구현 요구사항

- 올바른 DELETE 문의 실행
  - ✓ 테이블에서 조건에 맞는 튜플을 삭제하고 DeleteResult(#count)에 해당하는 메시지 출력한다.
  - ✓ WHERE 절이 있다면, WHERE 절의 조건을 만족하는 튜플을 삭제한다.
  - ✓ WHERE 절이 없다면, 모든 튜플을 삭제한다.
- 쿼리 오류에 대한 종류 별 메시지 출력
  - ✓ 삭제 요청한 테이블이 존재하지 않는 경우, NoSuchTable에 해당하는 메시지 출력
  - · WHERE 절 내에서 발생한 오류 처리는 아래 WHERE 절 구현 요구사항을 따른다.

WHERE 절 구현 요구사항

- WHERE 절에서는 아래 조건에 부합하는 경우에만 비교연산자로 두 값을 비교할 수 있다.
  - ✓ char 타입의 값은 길이와 상관없이 다른 char 타입의 값과 비교(=, !=)할 수 있다.
  - ✓ int 타입의 값은 다른 int 타입의 값과 비교(>, <, =, !=, >=, <=)할 수 있다.
  - ✓ date 타입의 값은 다른 date 타입의 값과 비교(>, <, =, !=, >=, <=)할 수 있다.
  - ✓ null 은 다른 모든 타입의 값과 비교(is, is not)할 수 있다.
- WHERE 절 내에서 발생한 오류 처리
  - ✓ WHERE 절에서 비교 연산자로 비교할 수 없는 값들을 비교할 경우, WhereIncomparableError 메시지 출력
  - ✓ FROM 절에 명시되지 않은 테이블을 WHERE 절에서 참조할 경우, WhereTableNotSpecified 메시지 출력
  - ✓ WHERE 절에서 존재하지 않는 컬럼을 참조할 경우, WhereColumnNotExist에 해당하는 메시지 출력

- ✓ `WHERE` 절에서 참조하는 컬럼이 어느 테이블에 속해 있는지 모호한 경우, `WhereAmbiguousReference` 메시지 출력

- [where clause]를 구성하는 condition은 2개 이하로 구성된 경우만을 평가한다. 이때 괄호로 묶인 복잡한 condition은 고려하지 않아도 된다.

  - ✓ ex) 1 condition: where *condition1*
    2 conditions: where *condition1* and/or *condition2*

2.2.3. SELECT

정의

```
select [table_name.]column_name, …
from table_name, …
[where clause];
```

실행 예시

```
DB_2024-12345> select * from account;
+----------------+-------------+---------+
| account_number | branch_name | balance |
+----------------+-------------+---------+
| A-101          | Downtown    | 500     |
| A-102          | Perryridge  | 400     |
| A-201          | Brighton    | 900     |
| A-215          | Mianus      | 700     |
| A-217          | Brighton    | 750     |
| A-222          | Redwood     | 700     |
| A-305          | Round Hill  | 350     |
+----------------+-------------+---------+
DB_2024-12345> select customer_name, borrower.loan_number, amount
from borrower, loan
where borrower.loan_number = loan.loan_number and branch_name = 'Perryridge';
+---------------+-------------+--------+
| customer_name | loan_number | amount |
+---------------+-------------+--------+
| Adams         | L-16        | 1300   |
| Hayes         | L-15        | 1500   |
+---------------+-------------+--------+
```

구현 요구사항

- 올바른 SELECT문의 실행

  - ✓ WHERE 절이 있다면, 조건에 맞는 튜플에서 요청된 컬럼에 해당하는 값을 출력한다.

  - ✓ WHERE 절이 없다면, 선택한 테이블 내 모든 튜플에서 요청된 컬럼에 해당하는 값을 출력한다.

- ✓ 출력 포맷은 실행 예시를 따르되, 점선, 필드 간의 공백은 개수나 모양에 제약을 두지 않는다.

- 쿼리 오류에 대한 종류 별 메시지 출력
  - ✓ WHERE 절 내에서 발생한 오류 처리는 2.2.2 DELETE의 WHERE 절 구현 요구사항을 따른다.
  - ✓ FROM 절에 존재하지 않는 테이블이 포함되어 있다면, SelectTableExistenceError(#tableName) 메시지 출력
  - ✓ SELECT 대상이 되는 컬럼이 어느 테이블에 속한 것인지 모호하거나, 해당 컬럼이 없는 경우 SelectColumnResolveError(#colName) 메시지 출력
- SELECT 구문의 FROM 절은 **3개 이하**의 테이블을 포함하는 경우만을 평가한다.

2.3. (Optional) Primary key/Foreign key Constraints

본 장에서는 INSERT문과 DELETE문의 Primary key/Foreign key constraint에 대한 오류 처리에 대해 설명한다.

본 장에 대한 Q&A는 제공되지 않는다.

2.3.1 INSERT

- INSERT문의 Primary key constraint에 대한 오류 처리
  - ✓ Primary key에 이미 존재하는 값을 삽입하려고 하는 경우 `InsertDuplicatePrimaryKeyError` 메시지를 출력한다.
- INSERT문의 Foreign key constraint에 대한 오류 처리
  - ✓ 다른 테이블(TABLE_REF)의 primary key(KEY_REF)를 foreign key로 참조하고 있는 컬럼에 삽입할 값이 TABLE_REF.KEY_REF에 포함되지 않은 경우 `InsertReferentialIntegrityError` 메시지를 출력한다.

2.3.2 DELETE

- DELETE문의 Foreign key constraint에 대한 오류 처리
  - ✓ 다른 테이블에서 foreign key로 참조하고 있는 튜플이 삭제 대상에 포함될 경우 `DeleteReferentialIntegrityPassed(#count)` 메시지를 출력한다. 이 때, #count는 삭제 요청된 모든 튜플의 개수를 의미한다.

- ✓ Foreign key constraint를 위반하지 않고 삭제할 수 있는 나머지 튜플도 삭제하지 않는다.

3. 개발 환경

- Python 3.8 ~ 3.12
- Lark API
- Oracle Berkeley DB API

4. 제출

다음 파일들을 PRJ1-3_학번.zip(예: PRJ1-3_2024-12345.zip)으로 압축하여 제출한다.

1. grammar.lark
2. run.py
   - ✓ 프로젝트의 최상위 디렉토리에 위치해야 한다.
   - ✓ 추가적인 소스코드 파일 및 서브 디렉토리를 함께 제출해도 된다. 단, python run.py로 프로그램이 구동될 수 있도록 해야 한다.
   - ✓ 적절한 주석을 포함해야 한다.
3. 리포트
   - ✓ 프로젝트의 최상위 디렉토리에 위치해야 한다.
   - ✓ 반드시 pdf 형식이어야 한다.
   - ✓ 파일명은 PRJ1-3_학번.pdf (예: PRJ1-3_2024-12345.pdf)으로 한다.
   - ✓ 2장 이내로 작성한다(1장을 권장).
   - ✓ 반드시 포함되어야 하는 내용
     - 핵심 모듈과 알고리즘에 대한 설명
     - 구현한 내용에 대한 간략한 설명
     - (제시된 요구사항 중 구현하지 못한 부분이 있다면) 구현하지 못한 내용

- 프로젝트를 하면서 느낀 점 및 기타사항

✓ 추가로 포함할 수 있는 내용

- 정의되지 않은 요소의 구현
- 본 문서에 정의된 오류 유형 외 추가로 정의한 오류 유형

4. (Optional) DB 폴더 및 파일

✓ Database 저장 시 경로는 아래와 같다.

- DB를 단일 파일로 구성할 경우:
  PRJ1-3_2024-12345/myDB.db
- DB를 복수개의 파일로 구성할 경우:
  PRJ1-3_2024-12345/DB/myDB1.db ,
  PRJ1-3_2024-12345/DB/myDB2.db , …

```
# 제출 파일(PRJ1-3_학번.zip) 내부 구조
PRJ1-3_학번
├── PRJ1-3_학번.pdf
├── run.py
├── grammar.lark
└── myDB.db (DB파일이 복수개로 저장되는 경우 DB/myDB1.db, …)
```

5. 성적 관련 사항

- 기한 끝난 후 과제 제출 시 늦은 시간 별 감점 기준
  - ✓ 0~24 시간: 10% 감점
  - ✓ 24~48 시간: 20% 감점
  - ✓ 48시간 이후: 점수 없음
- 부정 행위는 0점 처리
  - ✓ 다른 사람의 코드를 참조하는 행위
  - ✓ 이전에 수강한 사람의 코드를 참조하는 행위
  - ✓ 프로그램을 통해 표절 여부 확인 예정
- 본 문서 상의 출력 양식을 지키지 않을 시 감점

- 소스 코드에 주석이 없는 경우 감점

6. 참고 자료

- **Oracle Berkeley DB**
  - ✓ http://www.oracle.com/technetwork/database/berkeleydb/overview/index.html
- **Python Berkeley DB API Resources**
  - ✓ https://www.jcea.es/programacion/pybsddb.htm
  - ✓ https://docs.jcea.es/berkeleydb/latest/index.html