Project 1-3: Implementing DML

Due: 2023/5/16 (Tue), 11:59 PM

0. Project Overview

Brief)

이번 프로젝트의 목표는 프로젝트 1-1 및 프로젝트 1-2에서 구현한 프로그램에 기능을 추가하여 DML을 처리할 수 있도록 하는 것이다.

- INSERT
- DELETE
- SELECT

프로젝트 1-2와 마찬가지로 Berkeley DB를 이용해서 구현하며, 테이블 데이터는 파일에 저장되어 프로그램이 종료되어도 사라지지 않아야 한다.

Prerequisites)

본 프로젝트를 시작하기 위한 선행 조건으로 프로젝트 1-2에서 구현한 CREATE 구문이 동작할 수 있어야 한다. 따라서, 아래 CREATE 예제 구문들이 정상적으로 동작해야 한다.

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)
	);

단, 본 프로젝트에서 CREATE 구문들을 평가하지 않는다. 따라서 이전 프로젝트에 정의된 CREATE 구문 에러들은 여기서 고려하지 않아도 된다.

Assumptions)

본 프로젝트를 진행하기 위해 숙지해야 할 가정들이며, 아래 가정을 벗어나는 case들은 고려하지 않고 구현하면 된다.

1) **INSERT/DELETE 구문 입력을 받을 때, Primary key/Foreign key constraint에 위배되는 입력은 들어오지 않는다.**
2) **where절이 포함된 DELETE/SELECT 구문 입력을 받을 때, where 절의 condition은 최대 4개까지 입력된다.**
   select * from *table_name* where *condition1* and *condition2* and (*condition3* or *condition4*);
3) **SELECT 구문의 from은 최대 3개의 테이블까지 적용된다. ('[]'는 생략 가능을 의미)**
   select * from *table_name_a, table_name_b, table_name_c* [where_clause];

본 프로젝트를 시작하기에 앞서 위 사항들을 반드시 확인하고 구현에 임하도록 한다. 본 문서에서 제공되는 요구사항, 개발 환경 조건, 제출 및 성적 기준 안내 등을 숙지하여 진행하면 된다.

1. 요구 사항

- 프로젝트 1-1 및 프로젝트 1-2에서 구현한 프로그램을 이용하여야 한다.
- 2장에 나열된 모든 DML 구문을 처리할 수 있어야 한다. (3장은 optional)
  - 이전 프로젝트에서 정의된 definition과 다른 부분이 있으면, 본 문서에 정의된 definition을 기준으로 구현하도록 한다.
- 테이블 데이터를 파일(단일 파일 혹은 여러 개의 파일)에 저장하여야 한다.
  - 저장 디렉토리 요구사항은 1-2와 동일하다.
  - DBMS 콘솔을 종료한 후 다시 실행하더라도 데이터가 남아 있어야 한다.
  - Berkeley DB를 이용하여 구현한다.
- 2장에는 실행 예시와 기본 가정 및 각 상황 별로 출력해야 하는 메시지의 종류가 나열되어 있다. 메시지의 내용은 메시지 정의 파일 (2023_1-3_Messages doc 파일)을 참조한다.

2. SQL

2.1 INSERT

- Definition

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

- 실행 예시

```
DB_2023-12345> insert into account values(9732, 'Perryridge');
DB_2023-12345> The row is inserted
```

- 튜플(tuple) 삽입에 성공한다면, 테이블에 값을 삽입하고 `InsertResult`에 해당하는 메시지를 출력한다.
- 입력한 쿼리에 오류가 있다면, 오류에 맞는 적절한 에러 메시지를 출력한다.
  - 테이블이 존재하지 않을 경우, `NoSuchTable`에 해당하는 메시지 출력
  - 타입이 맞지 않는다면, `InsertTypeMismatchError`에 해당하는 메시지 출력
    - 컬럼과 값의 수나 타입이 맞지 않는 경우, 컬럼을 명시하지 않았을 때 값의 수가 해당 테이블의 attribute 수와 같지 않은 경우에도 이 에러에 해당됨
  - `null` 값을 가질 수 없는 컬럼에 `null` 값을 삽입하려고 할 경우, `InsertColumnNonNullableError(#colName)`에 해당하는 메시지 출력
  - 존재하지 않는 컬럼에 값을 삽입하려고 할 경우, `InsertColumnExistenceError(#colName)`에 해당하는 메시지 출력
- 튜플을 삽입할 때에는 다음과 같은 가정을 따른다.
  - `char` 컬럼 타입에 명시된 최대 길이보다 긴 문자열을 삽입하려 할 때는, 에러를 발생하지 않고 길이에 맞게 자른(truncate) 문자열을 삽입한다.
  - 테이블의 컬럼 이름은 중복되지 않는다.

2.2 DELETE

- Definition

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

- 실행 예시

```
DB_2023-12345> delete from account where branch_name = 'Perryridge';
DB_2023-12345> 5 row(s) are deleted
```

- 입력한 쿼리가 올바르다면, 조건에 해당되는 튜플을 삭제하고 `DeleteResult(#count)`에 해당하는 메시지를 출력한다.
- 입력한 쿼리에 오류가 있다면, 오류에 맞는 적절한 에러 메시지를 출력한다.
  - 테이블이 존재하지 않을 경우, `NoSuchTable`에 해당하는 메시지 출력
- 입력한 쿼리 중 where clause에 오류가 있다면, 그에 맞는 에러 메시지를 출력한다. **(SELECT 구문 구현할 때도 이 부분을 참고하도록 한다)**

- where 절에서 비교 연산자(>, <, =, !=, >=, <=)로 비교할 수 없는 값들을 서로 비교할 경우, WhereIncomparableError에 해당하는 메시지 출력
- where 절에서 명시되지 않은 테이블을 참조할 경우, WhereTableNotSpecified에 해당하는 메시지 출력
  - 예: 실행 예시의 where 절에서 account 테이블이 아닌 다른 테이블을 참조할 경우
- where 절에서 존재하지 않는 컬럼을 참조할 경우, WhereColumnNotExist에 해당하는 메시지 출력
- where 절에서 참조가 모호한 경우, WhereAmbiguousReference에 해당하는 메시지 출력

- 튜플을 삭제할 때에는 다음과 같은 가정을 따른다.
  - 아래 경우에 해당될 때에만 두 값을 비교할 수 있다.
    - char 타입의 값은 길이와 상관없이 다른 char 타입의 값과 비교할 수 있다.
    - int 타입의 값은 다른 int 타입의 값과 비교할 수 있다.
    - date 타입의 값은 다른 date 타입의 값과 비교할 수 있다.
    - null 값은 다른 모든 타입의 값과 비교할 수 있다.
  - null 값을 비교할 경우, 비교 결과는 true, false, unknown 중 하나이다.
    - 자세한 사항은 교재 및 슬라이드를 따른다.
  - where 절이 없다면 모든 튜플을 삭제한다.

2.3 SELECT

- Definition

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

- 실행 예시

```
DB_2023-12345> 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_2023-12345> 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   |
+---------------+-------------+--------+
```

- 입력한 쿼리가 올바르다면, 결과를 예시와 같은 형식으로 출력한다.
  - 점선, 필드 간의 공백은 원하는 대로 정의하면 되며, 출력되어야 하는 데이터가 모두 결과물에 정상적으로 포함되어 있어야 한다.
- 입력한 쿼리에 오류가 있다면, 오류에 맞는 적절한 에러 메시지를 출력한다.
  - `from` 절에 있는 테이블 중 존재하지 않는 테이블이 있다면, `SelectTableExistenceError(#tableName)`에 해당하는 메시지를 출력
  - `select` 뒤에 나오는 컬럼 이름을 해석하는 데에 문제가 있다면, `SelectColumnResolveError(#colName)`에 해당하는 메시지를 출력
    - 컬럼이 존재하지 않거나 컬럼 이름이 모호한 경우 이 에러에 해당된다.
- `select` 쿼리를 처리할 때에는 다음과 같은 가정을 따른다.
  - `where` 절에 대한 조건은 `delete` 쿼리와 같다.
  - `where` 절이 없다면 모든 튜플을 출력한다.

3. SQL (OPTIONAL)

- Primary key/Foreign key constraint가 모두 반영된 INSERT/DELETE 구문을 구현하면 프로젝트 1-1, 프로젝트 1-2 성적을 만회할 수 있는 가산점이 부여된다.
- 가산점은 1-1, 1-2 합쳐서 최대 10점까지 받을 수 있다.

3.1 INSERT

- Definition은 '2.1 INSERT' 장과 동일하다.
- 실행 결과가 primary key 제약에 위배된다면, `InsertDuplicatePrimaryKeyError`에 해당하는 메시지 출력
  - 입력 받은 primary key 값이 기존 튜플의 primary key 값과 중복되는 경우
- 실행 결과가 foreign key 제약에 위배된다면, `InsertReferentialIntegrityError`에 해당하는 메시지 출력
  - 입력 받은 foreign key 값이 referenced 테이블의 primary key 값에 존재하지 않는 경우

3.2 DELETE

- Definition은 '2.2 DELETE' 장과 동일하다.
- 실행 결과, referential integrity 때문에 일부 튜플을 삭제할 수 없다면 삭제를 진행하지 않고 `DeleteReferentialIntegrityPassed(#count)`에 해당하는 메시지를 추가로 출력
  - 다른 테이블이 reference 하는 primary key 값을 보유 중인 튜플을 삭제하려는 경우

※주의사항※

- 본 장의 스펙을 구현하려다 본래 구현한 INSERT/DELETE 기능에 오류가 발생하여 2장 스펙 구현에 대한 채점 시 불이익을 받지 않도록 주의한다.
- 필수 과제가 아니기 때문에 해당 범위에 대한 Q&A는 제공되지 않는다.
- 본 장에 정의되지 않은 error에 해당하는 case는 채점에 사용되지 않기 때문에 고려하지 않고 구현하면 된다.

본 문서에 정의되지 않은 애매한 경우가 있다고 판단되면 MySQL을 기준으로 구현하고 보고서에 명시한다.

본 문서에서 정의된 오류 상황 이외의 오류 상황이 있다고 판단될 경우 그 오류의 이름과 메시지를 직접 정의하고 보고서에 명시한다.

(단, 채점은 본 문서에 나오는 오류 상황 처리 여부만 볼 것임. 즉 본 문서에 정의되지 않은 상황에 대해서는 채점하지 않음)

4. 개발 환경

- Python 3.6 ~ 3.9
- Lark
- Oracle Berkeley DB API

5. 제출

1. 프로젝트 디렉토리
   - run.py가 최상위 디렉토리에서 실행될 수 있어야 함
   - 소스 파일은 반드시 적절한 주석을 포함하여야 함
   - 프로젝트 디렉토리 이름 : PRJ1-3_학번 (ex: PRJ1-3_2023-12345)
2. 리포트
   - run.py와 동일한 위치에 있어야 함

- 파일명: PRJ1-3_학번.pdf (예: PRJ1-3_2023-12345.pdf)
- **반드시 pdf 포맷으로 제출**
- **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
  - 분량은 1장 권장, 2장 이내로 작성

- 위 '프로젝트 디렉토리'를 압축하여 ETL로 제출
  - 파일명: PRJ1-3_학번.zip (예: PRJ1-3_2023-12345.zip)

6. 성적 관련 사항

- 제출 기한 이후 24시간 이내 제출시 10% 감점
- 제출 기한 이후 24시간 이후 48시간 이내 제출시 20% 감점
- 제출 기한 48시간 이후에는 점수 없음
- 부정 행위는 0점 처리
  - 다른 사람의 코드를 참조하는 행위
  - 이전에 수강한 사람의 코드를 참조하는 행위
  - 제출한 소스코드에 대해 표절 방지 프로그램을 돌릴 예정
- 본 문서에 명시되어 있는 출력 양식을 지키지 않을 시 감점
- 주석이 없는 경우 감점

7. References

- 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