Project 1-2: Implementing DDL & Basic DML

Due: 2023/4/25 (Tue), 11:59 PM

0. Project Overview

이번 프로젝트의 목표는 프로젝트 1-1에서 구현한 SQL 파서 프로그램을 확장하여 DBMS를 구현함으로써, 스키마를 저장하고 스키마에 접근할 수 있도록 하는 것이다.

이번 프로젝트에서 구현한 프로그램은 아래 나열된 **DDL 구문**과 **DML 구문** 일부를 처리할 수 있어야 한다.

1. **DDL 구문**: `CREATE TABLE, DROP TABLE, EXPLAIN/DESCRIBE/DESC, SHOW TABLES`
2. **DML 구문** (일부): `INSERT TABLE, SELECT`

이번 프로젝트부터는 Berkeley DB 라이브러리를 사용해서 DBMS를 구현하게 된다. Berkeley DB는 다음과 같은 특징을 지닌 라이브러리이다.

- 현재 Oracle에서 제공 중인 라이브러리형 DB
- 초기에는 임베디드 DB SW를 제공하는 것을 목적으로 개발되었음
- key-value pair를 byte array 형태로 저장하는 방식
- 본래는 C언어로 작성되었으나 현재는 C++, Java, Python 등 다양한 언어의 API로도 지원되고 있음

Python에서 지원되는 Berkeley DB API 관련 자료는 6장의 Reference를, 활용 예시는 본 문서와 함께 제공된 OT자료(Project 1-2 Tutorial.pptx)를 참고하도록 한다.

Berkeley DB는 relational system이 아니기 때문에 스키마 설계 및 데이터 관리 방법을 자유롭게 고안해서 DBMS를 구현할 수 있다.

예)

1. 하나의 DB파일에 하나의 스키마를 관리하는 방법 (One DB-One Schema)
2. 하나의 DB파일에 복수의 스키마를 관리하는 방법 (One DB-Multi Schema)
3. 스키마의 메타데이터를 별도의 DB파일에 저장, 관리하는 방법 (Metadata Schema) 등

프로젝트 1-2에서 구현한 DBMS 코드는 추후 프로젝트 1-3에서도 계속해서 사용될 예정이니 본 문서에서 제공되는 요구사항, 개발 환경 조건, 제출 및 성적 기준 안내 등을 숙지하여 진행하면 된다. 추후 프로젝트 1-3에서는 DML 구문들(INSERT/DELETE/SELECT)을 구현할 예정이므로, 이를 고려하면서 구현하도록 한다.

1. 요구 사항

- 프로젝트 1-1에서 구현한 SQL 파서 프로그램을 이용하여야 한다.
- 2장에 나열된 모든 DDL 구문들과 DML 구문들을 처리할 수 있어야 한다.
- 스키마를 파일(단일 파일 혹은 여러 개의 파일)에 저장하여야 한다.
  - Berkeley DB 라이브러리를 이용하여 스키마를 저장, 관리한다.
  - Berkeley DB 11g이상의 버전을 사용하는 경우 SQL을 지원하는 API 사용은 금지된다.
  - 프로그램을 종료한 후 다시 실행하더라도 저장된 DB 파일에 스키마가 남아 있어야 한다.
- **메시지 정의 문서(2023_1-2_Messages.docx)를 참고해서 처리해야 하는 에러 유형을 숙지하고, 그에 대응하는 에러 메시지를 아래처럼 입력 프롬프트 옆에 출력한다.**
  - 예를 들어 에러 유형이 NoSuchTable인 경우, 아래처럼 출력하면 된다.

    ```
    DB_2023-12345> No such table
    ```
- **CREATE TABLE, DROP TABLE을 구현할 때 Foreign key constraint를 고려해야 한다.**

2. SQL

2.1 CREATE TABLE

- Definition

```
create table table_name (
      column_name data_type [not null],
      …
      [primary key(column_name1, column_name2, …),]
      [foreign key(column_name3) references table_name1(column_name4),]
      [foreign key(column_name5) references table_name2(column_name6)]
      …
)
```

- 실행 예시

```
DB_2023-12345> create table account
(
      account_number int not null,
      branch_name char(15)
);
DB_2023-12345> 'account' table is created
```

- 입력한 쿼리가 올바르다면, 테이블 정보를 저장하고 `CreateTableSuccess(#tableName)`에 해당하는 메시지를 출력한다.
- 입력한 쿼리에 오류가 있다면, 오류에 대응되는 에러 메시지를 출력한다.

- 컬럼의 이름이 중복될 경우, `DuplicateColumnDefError`에 해당하는 메시지 출력
- Primary key 정의가 여러 번 입력된 경우, `DuplicatePrimaryKeyDefError`에 해당하는 메시지 출력
- Foreign key의 타입과 foreign key가 참조하는 컬럼의 타입이 서로 다른 경우, `ReferenceTypeError`에 해당하는 메시지 출력
- Foreign key가 primary key가 아닌 컬럼을 참조한다면, `ReferenceNonPrimaryKeyError`에 해당하는 메시지 출력

  (Foreign key가 composite primary key의 일부만을 reference하는 경우에도 `ReferenceNonPrimaryKeyError`에 해당하는 메시지 출력)
- Foreign key가 존재하지 않는 컬럼을 참조한다면, `ReferenceColumnExistenceError`에 해당하는 메시지 출력
- Foreign key가 존재하지 않는 테이블을 참조한다면, `ReferenceTableExistenceError`에 해당하는 메시지 출력
- 존재하지 않는 컬럼을 primary key로 정의한 경우, `NonExistingColumnDefError(#colName)`에 해당하는 메시지 출력
- 존재하지 않는 컬럼을 foreign key로 정의한 경우에도 `NonExistingColumnDefError(#colName)`에 해당하는 메시지 출력
- 이미 같은 이름의 테이블이 존재할 경우, `TableExistenceError`에 해당하는 메시지 출력
- `char` 타입의 길이를 1보다 작게 지정한 경우, `CharLengthError`에 해당하는 메시지 출력

- 테이블을 생성할 때에는 다음과 같은 가정을 따른다.
  - Primary key로 지정된 컬럼은 자동적으로 `not null`이 된다.
  - Foreign key는 다른 테이블의 primary key를 참조하여야 한다.
  - Foreign key와 foreign key가 참조하는 컬럼의 타입은 서로 같아야 한다.
    - `char` 타입의 경우, 길이가 다르다면 서로 다른 타입으로 본다.
  - `null` 값을 가질 수 있는 컬럼도 foreign key가 될 수 있다.
  - 하나의 테이블은 여러 개의 foreign key를 가질 수 있다.
  - Foreign key는 자신과 같은 테이블에 있는 컬럼을 참조할 수 없다.
  - `char` 타입의 길이는 0보다 커야 한다.
  - 테이블 이름과 컬럼 이름은 대소문자를 구분하지 않는다. (case insensitive)

2.2 DROP TABLE

- Definition

```
drop table table_name;
```

- 실행 예시

```
DB_2023-12345> drop table account;
DB_2023-12345> 'account' table is dropped
```

- 입력한 쿼리가 올바르다면, 테이블 정보를 삭제하고 `DropSuccess(#tableName)`에 해당하는 메시지를 출력한다.
- 입력한 쿼리에 오류가 있다면, 오류에 대응되는 에러 메시지를 출력한다.
  - 테이블이 존재하지 않을 경우, `NoSuchTable`에 해당하는 메시지 출력
  - 다른 테이블이 참조하고 있는 테이블을 삭제하려고 할 경우, `DropReferencedTableError(#tableName)`에 해당하는 메시지 출력

2.3 EXPLAIN / DESCRIBE / DESC

- Definition

```
explain table_name;
describe table_name;
desc table_name;
```

- 실행 예시

```
DB_2023-12345> explain account;
-------------------------------------------------------------------
table_name [account]
column_name          type          null          key
account_number       char(10)      N             PRI
branch_name          char(15)      N             FOR
balance              int           Y
-------------------------------------------------------------------
```

- 테이블 정보를 두 점선 사이에 출력한다.
  - 컬럼 이름, 타입, `null` 값 허용 여부, key 정보(primary key, foreign key)를 포함하여야 함
  - 출력 형식은 위와 같이 테이블 이름이 먼저 출력되고, 각 컬럼 이름 (column_name, type, null, key), 그 다음으로 각 값들이 출력되어야 한다.
    - 필드 사이의 공백은 원하는 대로 정의하면 된다.
    - 또한 데이터 출력 순서 관계없이 모든 데이터가 정상적으로 출력되기만 하면 된다.

- 테이블이 존재하지 않는다면 `NoSuchTable`에 해당하는 메시지를 출력한다.
- **describe, desc** 명령어에 대해서도 동일한 결과물을 출력할 수 있어야 한다.

2.4 SHOW TABLES

- Definition

```
show tables;
```

- 실행 예시

```
DB_2023-12345> show tables
------------------------
branch
customer
loan
borrower
account
depositor
------------------------
```

- DB에 존재하는 모든 테이블의 이름을 출력한다.
- 아무 테이블도 존재하지 않는다면 두 점선만 출력한다.

2.5 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)을 삽입할 때에는 다음과 같은 가정을 따른다.
  - `char` 컬럼 타입에 명시된 최대 길이보다 긴 문자열을 삽입하려 할 때는, 에러를 발생시키지 않고 길이에 맞게 자른(truncate) 문자열을 삽입한다.
  - 테이블의 컬럼 이름은 중복되지 않는다.
  - 이번 프로젝트에서는 **primary/foreign key가 없고 '유효한 튜플'만이 삽입**되는 상황만 가정하고 구현하도록 한다.

'유효한 튜플'이란 테이블 정의에 위배되지 않는, 즉 아래 조건을 모두 충족하는 튜플을 의미한다.

- ◆ 테이블 컬럼 개수와 입력된 value 개수가 일치
- ◆ 각 컬럼의 자료형과 입력된 value 들의 자료형이 모두 일치
- ◆ not null에 해당하는 컬럼에는 null 값이 들어가지 않음

- 튜플 삽입에 성공한다면, 테이블에 값을 삽입하고 `InsertResult`에 해당하는 메시지를 출력한다.
- 입력한 쿼리에 오류가 있다면, 오류에 대응되는 에러 메시지를 출력한다.
  - ■ 테이블이 존재하지 않을 경우, `NoSuchTable`에 해당하는 메시지 출력
  - ■ Key referential constraint 및 INSERT 구문 처리 과정에서 처리가 필요한 다른 오류들은 다음 프로젝트 (1-3)에서 구현하게 될 예정이다.

2.6 SELECT (이번 프로젝트에서는 'select all'만 필요)

- Definition

```
select * from table_name
```

- 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     |
+----------------+-------------+---------
```

- 입력한 쿼리가 올바르다면, 결과를 예시와 같은 형식으로 출력한다.
  - ■ 점선, 필드 간의 공백은 원하는 대로 정의하면 된다.
  - ■ 또한 데이터 출력 순서 관계없이 모든 데이터가 정상적으로 출력되기만 하면 된다.
- 입력한 쿼리에 오류가 있다면, 오류에 대응되는 에러 메시지를 출력한다.
  - ■ `from` 절에 있는 테이블이 존재하지 않는다면, `SelectTableExistenceError(#tabName)`에 해당하는 메시지를 출력

- 이번 프로젝트에서는 컬럼 선택, where 절 등을 배제하고 한 테이블에 저장된 전체 데이터를 조회하는 기능만 구현한다.
  본 기능이 구현되어야 **2.5 INSERT**장에 요구된 기능을 구현하고 동작 여부를 테스트할 수 있다.
  INSERT 구문과 마찬가지로 SELECT 구문은 다음 프로젝트 (1-3)에서 완성할 예정이다.

본 문서에서 정의된 오류 유형 이외의 유형이 추가로 필요하다고 판단될 경우 해당 유형의 이름과 메시지를 직접 정의하고 보고서에 명시하면 된다.

단, 본 문서에 정의된 오류 유형들에 대한 처리 여부만 채점 기준에 해당된다. 본 문서에 정의되지 않은 유형에 대해서는 채점하지 않기 때문에 본 문서에 정의된 오류 유형을 우선 처리하는 것을 권장한다.

3. 개발 환경

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

4. 제출

1. 프로젝트 디렉토리
   - grammar.lark 파일
   - run.py 파일
     - 추가적인 소스코드 파일 및 서브 디렉토리를 함께 제출해도 된다. 단, python run.py로 프로그램이 구동되도록 해야 하며, run.py는 프로젝트의 최상위 디렉토리에 배치해야 한다.
   - 소스코드 파일은 반드시 적절한 주석을 포함하여야 함
   - 프로젝트 디렉토리 이름: PRJ1-2_학번 (예: PRJ1-2_2023-12345)
2. 리포트
   - run.py와 동일한 위치에 있어야 함
   - 파일명: PRJ1-2_학번.pdf (예: PRJ1-2_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
- ◆ 리포트는 1장 recommend, 2장 이내로 작성

- Optional
  - ◆ 본 문서에 정의된 오류 유형 외 추가로 정의한 오류 유형

- 위 '프로젝트 디렉토리'를 압축하여 etl로 제출
  - 파일명: PRJ1-2_학번.zip (예: PRJ1-2_2023-12345.zip)

5. 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 않았거나 주석이 없는 경우 감점

6. 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