Project 1-3: Implementing DML

Due: 2017/12/5 (TUE), 11:59 PM

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

구현한 프로그램은 3개의 DML 구문(insert, delete, select)을 처리할 수 있어야 한다.

테이블 데이터는 파일에 저장되어 프로그램이 종료되어도 사라지지 않아야 한다. 데이터 저장에는 Berkeley DB를 이용한다.

1. **요구 사항**
   * 프로젝트 1-1 및 프로젝트 1-2에서 구현한 프로그램을 이용하여야 한다.
   * 2장에 나열된 모든 DML 구문을 처리할 수 있어야 한다.
   * 테이블 데이터를 파일(단일 파일 혹은 여러 개의 파일)에 저장하여야 한다.
     + DBMS 콘솔을 종료한 후 다시 실행하더라도 데이터가 남아 있어야 한다.
     + 테이블 데이터 저장에는 Berkeley DB를 이용한다.

2장에는 실행 예시와 기본 가정 및 각 상황 별로 출력해야 하는 메시지의 종류가 나열되어 있다. 메시지의 내용은 메시지 정의 파일을 참조한다.

1. **SQL**

**2.1 INSERT // 클래스 객체넣어서 거기에 ArrayList<ArrayList<String>> records 선언해서 했다.**

|  |
| --- |
| DB\_2017-12345> insert into account values(9732, 'Perryridge');  DB\_2017-12345> The row is inserted |

* 실행 예시
* 튜플(tuple) 삽입에 성공한다면, 테이블에 값을 삽입하고 InsertResult에 해당하는 메시지를 출력한다.
* 입력한 쿼리에 오류가 있다면, 오류에 맞는 적절한 에러 메시지를 출력한다.
  + 테이블이 존재하지 않을 경우, NoSuchTable에 해당하는 메시지 출력
  + 타입이 맞지 않는다면, InsertTypeMismatchError에 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
  + 실행 결과가 primary key 제약에 위배된다면, InsertDuplicatePrimaryKeyError에 해당하는 메시지 출력 –

primary key가 정의 안되있는 테이블 : 전체가 프라이머리 키

* + 실행 결과가 foreign key 제약에 위배된다면, InsertReferentialIntegrityError에 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 // where절 Set 이용해서 OR는 set union, And는 ....**

|  |
| --- |
| DB\_2017-12345> delete from account  where branch\_name = 'Perryridge';  DB\_2017-12345> 5 row(s) are deleted |

* 실행 예시
* 입력한 쿼리가 올바르다면, 조건에 해당되는 튜플을 삭제하고 DeleteResult(#count)에 해당하는 메시지를 출력한다.
  + 실행 결과, referential integrity 때문에 일부 튜플은 삭제할 수 없었다면 DeleteReferentialIntegrityPassed(#count)에 해당하는 메시지를 추가로 출력
    - DeleteResult(#count)와 DeleteReferentialIntegrityPassed(#count) 모두 출력하여야 한다.
    - Referential interegrity와 관련된 자세한 사항은 “About Referential Integrity Constraints” 문서를 참조
* 입력한 쿼리에 오류가 있다면, 오류에 맞는 적절한 에러 메시지를 출력한다.
  + 테이블이 존재하지 않을 경우, NoSuchTable에 해당하는 메시지 출력
  + 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
  + //리네임하면 리네임한것만 봄 지난 이름은 버림 as로 리네임 해놓고 리네임 하기 전에 테이블 이름으로 참조하는경우 WhereTableNotSpecified
  + //여러 테이블이 as로 리네임을 같은 이름으로 할경우, 원래있는 이름으로 리네임 할경우 - 모호함 체크

//// 테이블, 컬럼은 케이스인센시티브, 레코드는 케이스 센시티브

/// 세미콜론 찍기전에 에러나도 되네 ㅡㅡ 조교코드

/// 스트링 비교는 사전순 비교로 앞에 나오는게 작은거라고 생각했따

/// null 벨류는 unknown 처리 했다 – where not (val = 3) 이런거하면 null 벨류 인 애들 삭제 안되게

* 튜플을 삭제할 때에는 다음과 같은 가정을 따른다.
  + 아래 경우에 해당될 때에만 두 값을 비교할 수 있다.
    - char 타입의 값은 길이와 상관 없이 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 값과 다른 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**

* 실행 예시

|  |
| --- |
| DB\_2017-12345> 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\_2017-12345> 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.
* 입력한 쿼리에 오류가 있다면, 오류에 맞는 적절한 에러 메시지를 출력한다.
  + from 절에 있는 테이블 중 존재하지 않는 테이블이 있다면, SelectTableExistenceError(#tabName)에 해당하는 메시지를 출력

///// from절에 a as b, b 이런식으로 as를 from절에 존재하는 다른 테이블 이름으로 해버릴경우 = 여기서 처리하지 않고 웨어절에서 쓰려고 하면 모호함 에러 뜨게. 겹치는 리네임 자체는 허용

* + select 뒤에 나오는 컬럼 이름을 해석하는 데에 문제가 있다면, SelectColumnResolveError(#colName)에 해당하는 메시지를 출력
    - 컬럼이 존재하지 않거나 컬럼 이름이 모호한 경우
* select 쿼리를 처리할 때에는 다음과 같은 가정을 따른다.
  + where 절에 대한 조건은 delete 쿼리와 같다.
  + where 절이 없다면 모든 튜플을 출력한다.
* **본 문서에 정의되지 않은 애매한 경우가 있다고 판단되면 MySQL을 기준으로 구현할 것!**

1. **개발 환경**
   * Java
   * Eclipse
   * Oracle Berkeley DB API
2. **제출**
   1. Runnable jar 파일
      * 파일명: PRJ1-3\_학번.jar (예: PRJ1-3\_2017-12345.jar)
   2. Eclipse 프로젝트
      * 소스 파일은 반드시 적절한 주석을 포함하여야 함
   3. 리포트
      * 파일명: PRJ1-3\_학번.pdf (예: PRJ1-3\_2017-12345.pdf)
      * 반드시 pdf 포맷으로 제출
      * 포함되어야 하는 내용
        + 핵심 모듈과 알고리즘에 대한 설명
        + 구현한 내용에 대한 간략한 설명
        + (제시된 요구사항 중 구현하지 못한 부분이 있다면) 구현하지 못한 내용
        + 가정한 것들
        + 컴파일과 실행 방법
        + 프로젝트를 하면서 느낀 점
   * 위의 3가지 파일을 압축하여 [lecture@europa.snu.ac.kr](mailto:lecture@europa.snu.ac.kr)로 제출
     + 파일명: PRJ1-3\_학번.zip (예: PRJ1-3\_2017-12345.zip)
     + 메일 제목: [DB Project1-3] 학번, 이름 (예: [DB Project1-3] 2017-12345, 홍길동)
3. **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

1. **References**
   * **Oracle Berkeley DB**
     + <http://www.oracle.com/technetwork/database/berkeleydb/overview/index.html>