Introduction to Databases

Coursework 1 (Relational Algebra)

Dr Paolo Guagliardo

v1.0 (05/10/2021)

1 Database Schema

The schema we use in this assignment models a simplified music scenario consisting of the tables below.

`Artists` have the following attributes:

- `Name` (string) is the artist's unique name
- `Type` (string) is either `PERSON` or `BAND`
- `Country` (string) is an ISO alpha-3 code, which consists of exactly 3 uppercase letters characters

Each artist (i.e., each row in the `Artists` table) is uniquely identified by the value of its `Name`, so *there cannot be two artists with the same name.*

`Albums` have the following attributes:

- `Title` (string) is the album's title
- `Artist` (string) is a reference to an artist's name
- `Year` (positive integer) is the year when the album was released
- `Type` (string) is either `STUDIO`, `LIVE`, or `COMPILATION` (all uppercase)
- `Rating` (positive integer) ranges from 1 to 5

Each album (i.e., each row in the `Albums` table) is uniquely identified by the values of its `Title` and `Artist` (taken together as a pair), so *there may be albums with the same title but by different artists.*

`Tracks` have the following attributes:

- `Album` (string) is the title of an album that, together with the value of `Artist` below, references a row in the `Albums` table
- `Artist` (string) is the name of an artist that, together with the value of `Album` above, references a row in the `Albums` table
- `Title` (string) is the track's title
- `Length` (positive integer) is expressed in seconds

Each track (i.e., each row in the `Tracks` table) is uniquely identified by the values of its `Album`, `Artist` and `Title` (taken together as a triple), so *there may be tracks with the same title from different albums.*

2 Assignment

Write the following queries in the Relational Algebra language (under set semantics) supported by the REAL interpreter (v0.5.1). *The names of the columns in the output table are important* (recall also that they are case-sensitive), but their order is not (there is no order).

(01) **Tracks that are at least 9 minutes and 34 seconds long** [10 marks]

Expected column names of the output table: { `Album`, `Artist`, `Title`, `Length` }

(02) **Artists who have released a compilation in the 1990s** [10 marks]

Return name and country of artists who have released a compilation album between 1990 and 1999 (inclusive).

Expected column names of the output table: { `Name`, `Country` }

(03) **Shortest track (or tracks) of each album** [10 marks]

For each album in the `Tracks` table, return its title and artist, and the `Title` and `Length` of its shortest tracks (there may be more than one).

Expected column names of the output table: { `Album`, `Artist`, `Title`, `Length` }

(04) **Artists who have never released a live album** [10 marks]

Return the `Name` of each artist that satisfies the requirement. Artists who have not released any albums (at all, of any type) must be included in the output.

Expected column names of the output table: { `Name` }

(05) **Latest studio albums by each artist** [10 marks]

For each artist, return the title and year of their latest studio albums (that is, those for which no other studio album by the same artist was released in a later year). If an artist has never released a studio album, they will not be in the output.

Expected column names of the output table: { `Artist`, `AlbumTitle`, `Year` }

(06) **Bands who have only released studio albums** [10 marks]

Return the `Name` of each artist that satisfies the requirement.

Expected column names of the output table: { `Name` }

(07) **Artists with non-decreasing album ratings** [10 marks]

These are artists who have never released albums in distinct years such that the later one had a lower rating than the earlier one. Return the name of each such artist. Artists who have released at most one album trivially satisfy the requirement and must be included in the answers.

Expected column names of the output table: { `Name` }

(08) **Bands from USA whose debut studio album was rated 5 stars** [15 marks]

For each artist that satisfies the requirement, return their `Name` and the `Year` of their studio debut.

A *debut* studio album is one for which there is no other studio album released by the same artist in a previous year. USA bands who released more than one studio album on their debut year are returned if at least one of those albums is rated 5 stars.

Expected column names of the output table: { `Name`, `Year` }

(09) **Artists who have never released two consecutive studio albums** [15 marks]

Two albums A and B, by the same artist, are *consecutive* if A was released before B and no album (of any type) was released after A and before B. Observe that the granularity of albums' releases is in years.

Output the name of each artist that satisfies the requirement (recall that we are interested in consecutive *studio* albums). Artists who have released less than two studio albums (which includes artists who have not released any albums at all) will trivially be in the output.

Expected column names of the output table: { `Name` }

3 Test Database

A test instance to work on the assignment is available on Learn in the file `testdb.zip` linked from the RA assignment page. To use the test data, follow the steps below.

1. Download the zip file `testdb.zip` and unpack its contents into a directory of your choice.
2. In a shell, navigate to that directory and launch REAL:

   ```
   java -jar [PATH/TO/REAL/]real-0.5.1.jar
   ```

   where `[PATH/TO/REAL/]` is the directory where the REAL's JAR file is located on your system.
3. At REAL's prompt (`$>`), issue the following commands:

   ```
   $> .add Tracks(Album, Artist, Title, Length) : tracks.csv
   $> .add Artists(Name, Type, Country) : artists.csv
   $> .add Albums(Title, Artist, Year, Type, Rating) : albums.csv
   $> .save schema.json
   ```

   This will create the file `schema.json` in the directory where you extracted the CSV files.

The schema file can be renamed and moved around your file system as you please, but **the corresponding CSV files must remain where you initially placed them** (otherwise you need to recreate, or manually edit, the schema file, which stores the *absolute path* to the CSV file of each table). The test database can then be loaded into REAL at launch from the shell with the `-d` option, as follows:

```
java -jar [PATH/TO/REAL/]real-0.5.1.jar -d [PATH/TO/SCHEMA/FILE/]schema.json
```

or from REAL's prompt (`$>`) with the command:

```
$> .load [PATH/TO/SCHEMA/FILE/]schema.json
```

Please note that **you cannot make any assumptions based on the test data**: your queries will be executed on different instances (of the same schema) on which such assumptions may not hold. The only constraints that are guaranteed to hold on every instance are those explicitly stated in definition of the schema (Section 1).

4 Check Script

You can check whether your query files can be parsed and comply with REAL's syntax using the script `check.sh` available on DICE in the folder `/afs/inf.ed.ac.uk/group/teaching/dbs/2021/cw1/`. The script takes the following three input arguments:

1. The path to REAL's JAR file;
2. The path to a JSON schema file;
3. The path to a directory containing your query files.

So, for example, if your queries are in the directory `ra_queries` inside your home folder, and you copy (or link) the script into the same directory where REAL and the schema file are located, from that directory (in a DICE shell) you would issue the following command:

```
./check.sh real-0.5.1.jar schema.json ~/ra_queries
```

If there is a problem with some of the queries, an error message will be printed.