DIAS: Data-Intensive Applications and Systems Laboratory
School of Computer and Communication Sciences
Ecole Polytechnique Fédérale de Lausanne
Building BC, Station 14
CH-1015 Lausanne
URL: http://dias.epfl.ch/

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Databases Project – Spring 2018

Team No: 7

Names:

Da Rocha Rodrigues David Joaquim, Torres Da Cunha Pedro Filipe , Justinas Sukaitis

Contents

DIAS: Data-Intensive Applications and Systems Laboratory
School of Computer and Communication Sciences
Ecole Polytechnique Fédérale de Lausanne
Building BC, Station 14
CH-1015 Lausanne
URL: http://dias.epfl.ch/

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

Deliverable 1

Assumptions

We assumed that a clip could be in development, which implies that it might not have yet a released date/country, a rating, defined genres, defined languages and associated countries.

We also assumed that a movie staff (i.e. actor, director, producer or writer) doesn’t necessarily have a bibliography.

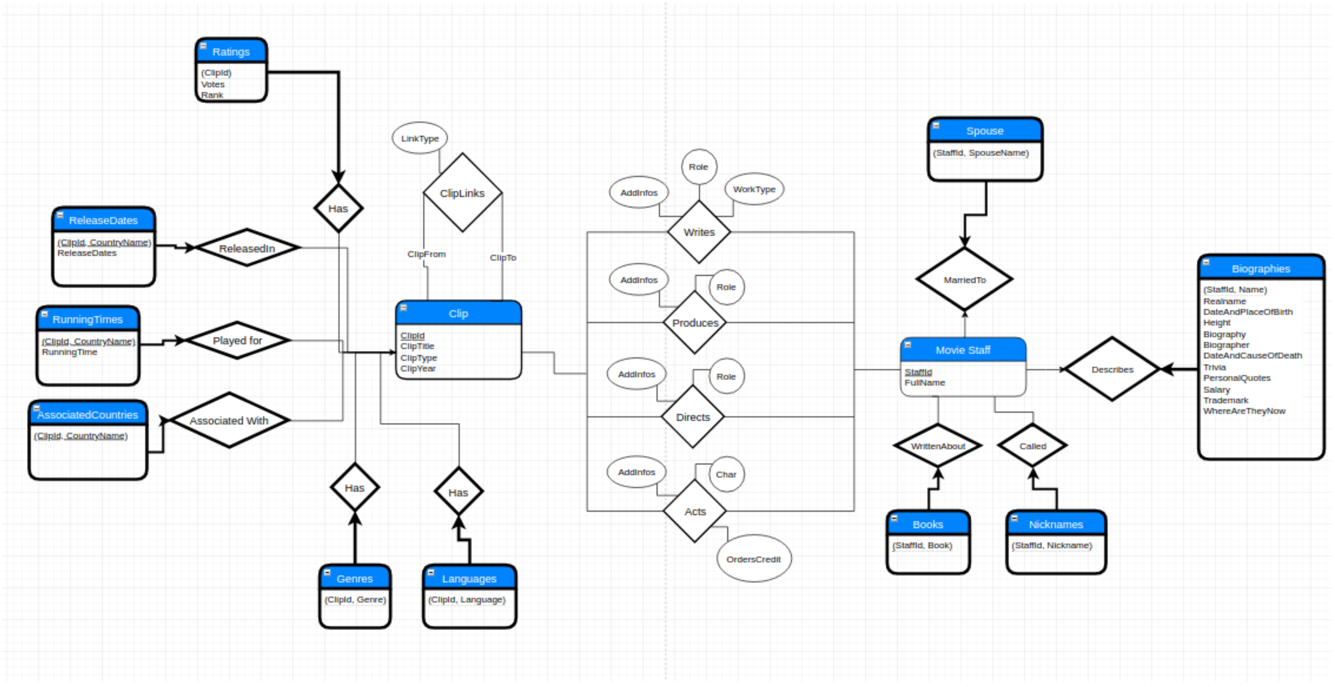
Finally, in this relationship scheme, we imply that a movie staff can be in the data base without being associated to a movie (i.e. the directors' first movie is not released yet or simply isn't in our database)

DIAS: Data-Intensive Applications and Systems Laboratory
School of Computer and Communication Sciences
Ecole Polytechnique Fédérale de Lausanne
Building BC, Station 14
CH-1015 Lausanne
URL: http://dias.epfl.ch/

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Entity Relationship Schema

Schema



Description

We regrouped the actors, producers, directors and writers into a single entity **Movie Staff** whose instances are identified by a unique *StaffId*.

A **Movie Staff** can be described by at most one **Biography**. A **Biography** describes exactly one **Movie Staff** which makes it a weak entity.

A **Movie Staff** can write, produce, direct or act in **Clip**s.

A **Clip** can have any number of ClipLinks with other **Clip**s.

A **Clip** can have **Genres**, have **Languages**, be associated to **Countries**, have **Release Countries** and have a **Rating**.

A **Genre** instance is a pair composed of a *ClipId* and a *Genre* so each instance is linked to exactly one **Clip**, this makes **Genres** be a weak entity of **Clip**.

Ratings, **Release Country**, **Country** and **Languages** are also all weak entities of **Clip** following the same reasoning.

Relational Schema

ER schema to Relational schema

Since we only have 2 owner entities: Clips and Movie Staff (the rest being weak entities). Defining the keys was straightforward:

Clips and Movie Staff have a unique Id for each of them: ClipId and StaffId respectively

Most of the weak entities have a pair of values as their primary key, which is always the main attribute of the weak entity and one of the 2 Ids, depending on to which entity we're relating to (ex.: Languages is related to Clips, thus ClipId and Language). As for Ratings and Biographies, there is at most one rating for a clip and at most one biography for a movie staff, so respectively the clip id and the staff id are enough for the primary keys.

Relationships that link 2 non-weak entities have their own tables with foreign keys showing what these relationships link.

DDL

```
CREATE TABLE Biographies ( StaffId INT,
                           Name VARCHAR(64),
                           Realname VARCHAR(64),
                           DateAndPlaceOfBirth VARCHAR(128),
                           Height VARCHAR(16),
                           Biography TEXT,
                           Biographer VARCHAR(40),
                           DateAndCauseOfDeath VARCHAR(128),
                           Trivia TEXT,
                           PersonalQuotes TEXT,
                           Salary TEXT,
                           Trademark VARCHAR(256),
                           WhereAreTheyNow TEXT,
                                        PRIMARY KEY (StaffId),
                                        FOREIGN KEY (StaffId) REFERENCES
MovieStaff ON DELETE CASCADE );

CREATE TABLE MovieStaff ( StaffId SERIAL,
                                       FullName VARCHAR(128),
                                       PRIMARY KEY (StaffId));

CREATE TABLE Writes ( StaffId INT,
                                 ClipId INT,
                                 AddInfos VARCHAR(128),
                                 Role VARCHAR(128),
                                 WorkType TEXT,
                                 PRIMARY KEY (StaffId, ClipId)
                                 FOREIGN KEY (StaffId) REFERENCES MovieStaff,
                                 FOREIGN KEY (ClipId) REFERENCES Clip );

CREATE TABLE Produces ( StaffId INT,
                                      ClipId INT,
                                      AddInfos VARCHAR(100),
                                      Role TEXT,
                                      PRIMARY KEY (StaffId, ClipId),
                                      FOREIGN KEY (StaffId) REFERENCES
MovieStaff,
                                      FOREIGN KEY (ClipId) REFERENCES Clip );

CREATE TABLE Directs ( StaffId INT,
                                  ClipId INT,
                                  AddInfos VARCHAR(100),
                                  Role TEXT,
                                  PRIMARY KEY (StaffId, ClipId),
                                  FOREIGN KEY (StaffId) REFERENCES MovieStaff,
                                  FOREIGN KEY (ClipId) REFERENCES Clip );

CREATE TABLE Acts ( StaffId INT,
                ClipId INT,
                Char TEXT,
                OrdersCredit INT,
                AddInfos TEXT,
                    PRIMARY KEY (StaffId, ClipId),
                FOREIGN KEY (StaffId) REFERENCES MovieStaff,
                FOREIGN KEY (ClipId) REFERENCES Clip);

CREATE TABLE Clip ( ClipId INT,
                               ClipTitle TEXT,
                               ClipYear INT,
                               ClipType VARCHAR(64),
                               PRIMARY KEY (ClipId));

CREATE TABLE ClipLinks( ClipFrom INT,
                                      ClipTo INT,
                                      LinkType VARCHAR(32),
                                      PRIMARY KEY (ClipFrom, ClipTo, LinkType),
                                      FOREIGN KEY (ClipFrom) REFERENCES Clip,
                                      FOREIGN KEY (ClipTo) REFERENCES Clip);

CREATE TABLE Ratings ( ClipId INT,
                                  Votes INT,
                                  RANK REAL,
                                  PRIMARY KEY (ClipId),
                                  FOREIGN KEY (ClipId) REFERENCES Clip ON
DELETE CASCADE);

CREATE TABLE RunningTime ( ClipId INT,
                                        ReleaseCountry VARCHAR(32),
                                        RunningTime INT,
                                        PRIMARY KEY (ClidId, ReleaseCountry),
                                        FOREIGN KEY (ClipId) REFERENCES Clip ON
DELETE CASCADE );

CREATE TABLE ReleaseDates ( ClipId INT,
                                              ReleaseCountry varchar(32),
                                              ReleaseDate VARCHAR(32),
                                              PRIMARY KEY (ClipId, ReleaseCountry),
                                              FOREIGN KEY (ClipId) REFERENCES Clip
ON DELETE CASCADE);

CREATE TABLE AssociatedCountry ( ClipId INT,
                                  CountryName VARCHAR(40),
                                  PRIMARY KEY (ClipId, CountryName),
                                  FOREIGN KEY (ClidId) REFERENCES Clip ON
DELETE CASCADE );

CREATE TABLE Languages (ClipId INT,
                                      Language VARCHAR(64),
                                      PRIMARY KEY (ClipId, Language),
                                      FOREIGN KEY (ClipId) REFERENCES Clip ON
DELETE CASCADE );

CREATE TABLE Genres (ClipId INT,
```

Genre VARCHAR(32),
PRIMARY KEY (ClipId, Genre),
FOREIGN KEY (ClipId) REFERENCES Clip ON DELETE CASCADE);

General Comments

Firstly we tried to think individually about the logical possible relationships between actors, directors, clips etc.
After having agreed on a version, we added constraints by looking at the data and using the course proposed book and course material.

Finally after the scheme was rectified, we created the DDL provided above.

We usually work together throughout the week, so there were no difficulties regarding meet ups and keeping a constant progress on the project.

DIAS: Data-Intensive Applications and Systems Laboratory
School of Computer and Communication Sciences
Ecole Polytechnique Fédérale de Lausanne
Building BC, Station 14
CH-1015 Lausanne
URL: http://dias.epfl.ch/

Deliverable 2

Assumptions

During this deliverable, we finished cleaning the data. We made the following assumptions on the
attributes:

- Even though countries do exist independently of related clips in the real world, in our database there is no need for a country to exist if it is not related to any clip
- We assume a computer adds a ClipId next to a user entry and thus the ClipId's are inherently correct and have no need to be cleaned or modified.
-

Data Loading

Query Implementation

Query a:

Description of logic:

Print the name and length of the 10 longest clips that were released in France.

SQL statement

```
SELECT C.cliptitle, R.runningtime
FROM Clip C, runningtime R
WHERE (R.releasecountry = 'France' AND C.clipid = R.clipid)
ORDER BY (R.runningtime) desc
LIMIT 10
```

Query b:

Description of logic:

Compute the number of clips released per country in 2001.

SQL statement

```
SELECT COUNT(R.ClipId)
FROM releasedates R
Group By R.ReleaseCountry
```

Query c:

Description of logic:

Compute the numbers of clips per genre released in the USA after 2013

SQL statement

```
SELECT COUNT(ClipId)
FROM Genre G, Releasedates R
WHERE R.ReleaseCountry = 'USA'
        AND R.ReleaseDate > 2013
Group By G.genre
```

Query d:

Description of logic:

Print the name of actor/actress who has acted in more clips than anyone else.

SQL statement

```
Select S.fullname
FROM moviestaff S
WHERE S.staffid = (Select A.staffid
        FROM acts A
        GROUP BY A.staffid
        ORDER BY COUNT(A.clipid)
        desc LIMIT 1)
```

Query e:

Description of logic:

Print the maximum number of clips any director has directed

SQL statement

```
Select Count(A.clipid)
FROM directs A
GROUP BY A.staffid
ORDER BY COUNT(A.clipid) desc
LIMIT 10
```

Query f:

Description of logic:

Print the names of people that had at least 2 different jobs in a single clip.

SQL statement

<The SQL statement>

Query g:

Description of logic:

Print the 10 most common clip languages.

SQL statement

```
Select A.language
FROM languages A
GROUP BY A.language
ORDER BY COUNT(A.clipid)
desc LIMIT 10
```

Query h:

Description of logic:

Print the full name of the actor who has performed in the highest number of clips with a user-specified type.

SQL statement

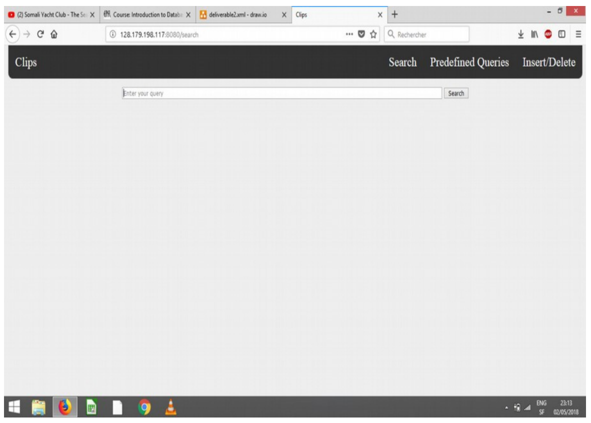
<The SQL statement>

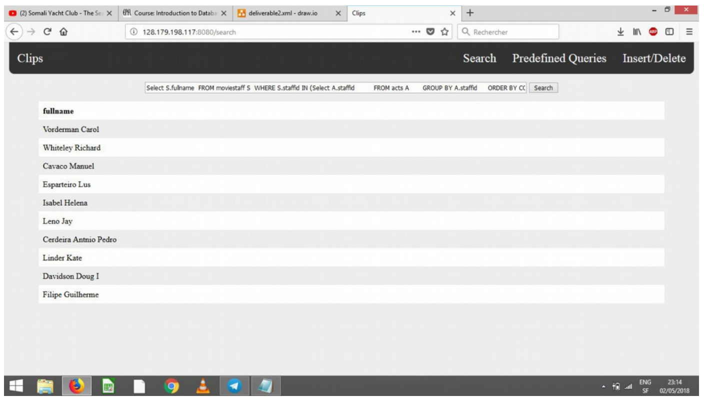
Interface

Design logic Description

For our database interface, we decided to use nodejs (as both the backend and frontend) and render the result locally on the browser (localhost). For now, for the search, we have to use sql syntax queries but on the upside, we can search for anything in the database. Insertion is not completely done and Deletion is not implemented.

Screenshots

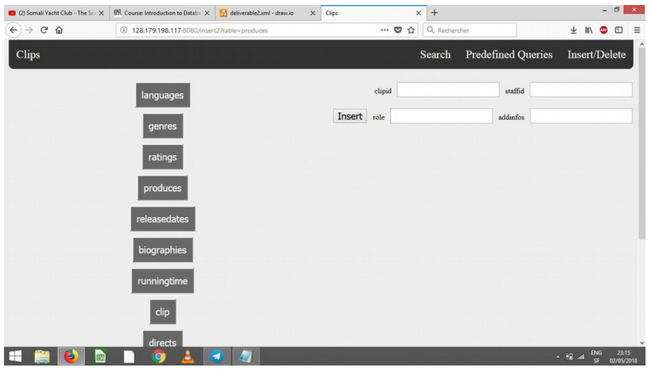

(2) Somali Yacht Club - The Se
Course: Introduction to Databa
deliverable2.xml - draw.io
Clips
128.179.198.117:8080/search
Rechercher
Clips
Search
Predefined Queries
Insert/Delete
Enter your query
Search
ENG SF
23:13
02/05/2018


(2) Somali Yacht Club - The Se
Course: Introduction to Databa
deliverable2.xml - draw.io
Clips
128.179.198.117:8080/search
Rechercher
Clips
Search
Predefined Queries
Insert/Delete
Search
ENG SF
23:14
02/05/2018

```
Select S.fullname  FROM moviestaff S  WHERE S.staffid IN (Select A.staffid      FROM acts A      GROUP BY A.staffid     ORDER BY CC
```

fullname
Vorderman Carol
Whiteley Richard
Cavaco Manuel
Esparteiro Lus
Isabel Helena
Leno Jay
Cerdeira Antnio Pedro
Linder Kate
Davidson Doug I
Filipe Guilherme

DIAS: Data-Intensive Applications and Systems Laboratory
School of Computer and Communication Sciences
Ecole Polytechnique Fédérale de Lausanne
Building BC, Station 14
CH-1015 Lausanne
URL: http://dias.epfl.ch/



General Comments

We didn’t finish everything up because of our poor schedule planning but we hope to get some feedback on the work we did even though we didn’t do everything that was required.

DIAS: Data-Intensive Applications and Systems Laboratory
School of Computer and Communication Sciences
Ecole Polytechnique Fédérale de Lausanne
Building BC, Station 14
CH-1015 Lausanne
URL: http://dias.epfl.ch/

Deliverable 3

Assumptions

<In this section write down the assumptions you made about the data. Write a sentence for each assumption you made>

Query Implementation

<For each query>

Query a:

Description of logic:

<What does the query do and how do I decide to solve it>

SQL statement

<The SQL statement>

Query Analysis

Selected Queries (and why)

Query 1

<Initial Running time:
Optimized Running time:
Explain the improvement:
Initial plan
Improved plan>

Query 2

<Initial Running time:
Optimized Running time:
Explain the improvement:
Initial plan
Improved plan>

Query 3

<Initial Running time:
Optimized Running time:
Explain the improvement:
Initial plan
Improved plan>

DIAS: Data-Intensive Applications and Systems Laboratory
School of Computer and Communication Sciences
Ecole Polytechnique Fédérale de Lausanne
Building BC, Station 14
CH-1015 Lausanne
URL: http://dias.epfl.ch/

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Interface

Design logic Description

<Describe the general logic of your design as well as the technology you decided to use>

Screenshots

<Provide some initial screen shots of your interface>

General Comments

<In this section write general comments about your deliverable (comments and work allocation between team members>