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/

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.

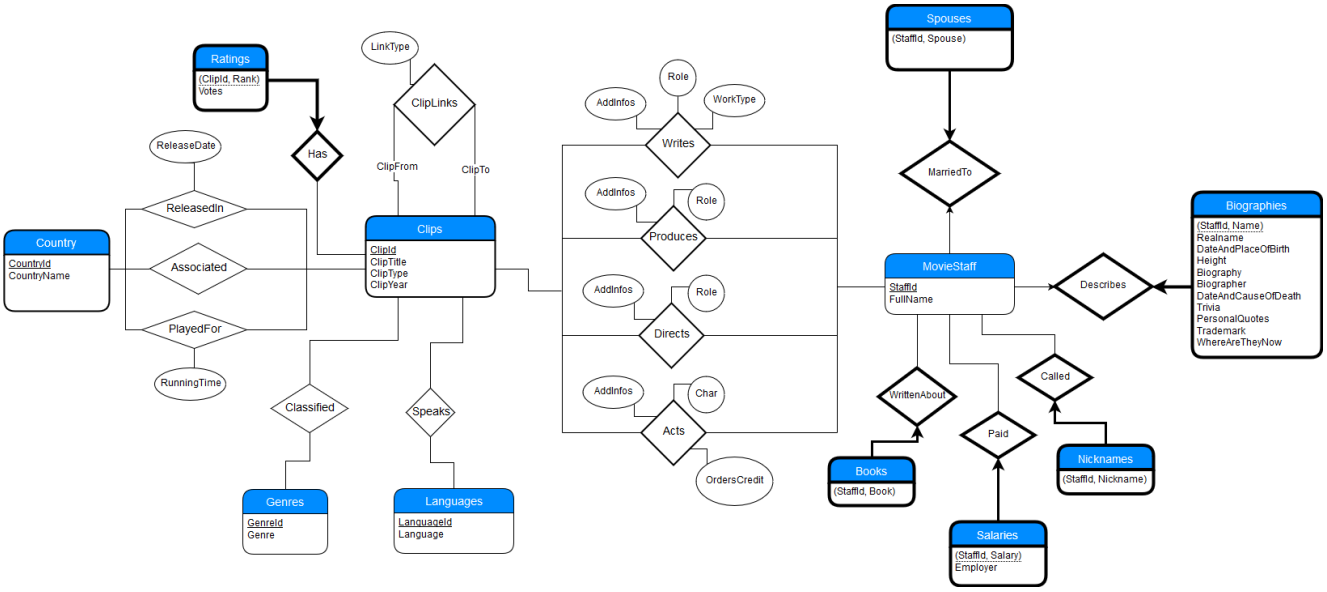
Finally, in this relationship scheme, we imply that a movie staff can be in the data base without being associated to a movie (i.e. the directors' first movie is not released yet or simply isn't in our database)

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 be classified by **Genres**, spoken in **Languages**.

Ratings is a weak entities of **Clip** following the same reasoning since a rating without a clip to rate makes no sence.

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

Whats new

Countries, **Languages** and **Genres** are strong entities now and are identified by their appropriate ids as primary keys. This extremely reduces their sizes and speeds up the database. Moreover they are linked with integers which speeds up even more the searches.
Runnings times and **Released countries** which were weak entities became relationships between countries and clips.
We extracted **Salaries** from **Bibliographies** as weak entities.

Relational Schema

ER schema to Relational schema

Since we only have 5 (old had 2) owner entities: Countries, Languages, Genres, Clips and Movie Staff (the rest being weak entities).
Defining the keys was straightforward:

> Everything has its' own unique Id: CountryId, LanguageId, GenreId, ClipId and StaffId respectively
>
> Every weak entity has a pair of values as their secondary key, which is always the main attribute of the weak entity and the Id linked to this weak entity(ex.: **Nicknames** is related to **MovieStaff**, thus StaffId and Nickname). 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 MovieStaff (

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

```
        StaffId SERIAL,
        FullName VARCHAR(128),
        PRIMARY KEY (StaffId)
);

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

CREATE TABLE Nicknames (
        StaffId INT,
        Nickname VARCHAR(128),
        PRIMARY KEY (StaffId, Nickname),
        FOREIGN KEY (StaffId) REFERENCES MovieStaff ON DELETE CASCADE
);

CREATE TABLE Salaries (
        StaffId INT,
        Salary VARCHAR(256),
        Employer VARCHAR(128),
        PRIMARY KEY (StaffId, Salary),
        FOREIGN KEY (StaffId) REFERENCES MovieStaff ON DELETE CASCADE
);

CREATE TABLE Books (
        StaffId INT,
        Book TEXT,
        PRIMARY KEY (StaffId, Book),
        FOREIGN KEY (StaffId) REFERENCES MovieStaff ON DELETE CASCADE
);

CREATE TABLE Spouses (
```

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/

```
        StaffId INT,
        Spouse TEXT,
        PRIMARY KEY (StaffId, Spouse),
        FOREIGN KEY (StaffId) REFERENCES MovieStaff ON DELETE CASCADE
);

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

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

CREATE TABLE Produces (
        StaffId INT,
        ClipId INT,
        Role VARCHAR(128),
        AddInfos TEXT,
        PRIMARY KEY (StaffId, ClipId),
        FOREIGN KEY (StaffId) REFERENCES MovieStaff ON DELETE CASCADE,
        FOREIGN KEY (ClipId) REFERENCES Clips ON DELETE CASCADE
);

CREATE TABLE Directs (
        StaffId INT,
        ClipId INT,
        Role VARCHAR(128),
        AddInfos TEXT,
        PRIMARY KEY (StaffId, ClipId),
        FOREIGN KEY (StaffId) REFERENCES MovieStaff ON DELETE CASCADE,
        FOREIGN KEY (ClipId) REFERENCES Clips ON DELETE CASCADE
);
```

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

```
CREATE TABLE Acts (
        StaffId INT,
        ClipId INT,
        Char TEXT,
        OrdersCredit INT,
        AddInfos TEXT,
        PRIMARY KEY (StaffId, ClipId),
        FOREIGN KEY (StaffId) REFERENCES MovieStaff ON DELETE CASCADE,
        FOREIGN KEY (ClipId) REFERENCES Clips ON DELETE CASCADE
);

CREATE TABLE ClipLinks(
        ClipFrom INT,
        ClipTo INT,
        LinkType VARCHAR(32),
        PRIMARY KEY (ClipFrom, ClipTo, LinkType),
        FOREIGN KEY (ClipFrom) REFERENCES Clips ON DELETE CASCADE,
        FOREIGN KEY (ClipTo) REFERENCES Clips ON DELETE CASCADE
);

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

CREATE TABLE Languages (
        LanguageId SERIAL,
        Language VARCHAR(64),
        PRIMARY KEY (LanguageId)
);

CREATE TABLE Speaks (
        ClipId INT,
        LanguageId INT,
        PRIMARY KEY (ClipId, LanguageId),
        FOREIGN KEY (ClipId) REFERENCES Clips ON DELETE CASCADE,
        FOREIGN KEY (LanguageId) REFERENCES Languages ON DELETE CASCADE
);

CREATE TABLE Genres (
        GenreId SERIAL,
        Genre VARCHAR(32),
```

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/

```
        PRIMARY KEY (GenreId)
);

CREATE TABLE Classified (
        ClipId INT,
        GenreId INT,
        PRIMARY KEY (ClipId, GenreId),
        FOREIGN KEY (ClipId) REFERENCES Clips ON DELETE CASCADE,
        FOREIGN KEY (GenreId) REFERENCES Genres ON DELETE CASCADE
);

CREATE TABLE Country (
        CountryId SERIAL,
        CountryName VARCHAR(64),
        PRIMARY KEY (CountryId)
);

CREATE TABLE ReleasedIn (
        ClipId INT,
        CountryId INT,
        ReleaseDate TEXT,
        PRIMARY KEY (CountryId, ClipId),
        FOREIGN KEY (CountryId) REFERENCES Country ON DELETE CASCADE,
        FOREIGN KEY (ClipId) REFERENCES Clips ON DELETE CASCADE
);

CREATE TABLE Associated (
        ClipId INT,
        CountryId INT,
        PRIMARY KEY (CountryId, ClipId),
        FOREIGN KEY (CountryId) REFERENCES Country ON DELETE CASCADE,
        FOREIGN KEY (ClipId) REFERENCES Clips ON DELETE CASCADE
);



CREATE TABLE PlayedFor (
        ClipId INT,
        CountryId INT,
        RunningTime INT,
        PRIMARY KEY (CountryId, ClipId),
        FOREIGN KEY (CountryId) REFERENCES Country ON DELETE CASCADE,
        FOREIGN KEY (ClipId) REFERENCES Clips ON DELETE CASCADE
);
```

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

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/

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

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

Since the ER changed, we needed to change the queries

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, P.RunningTime
FROM Clips C
NATURAL JOIN PlayedFor P
NATURAL JOIN Country C2
WHERE C2.CountryName = 'France'
AND P.RunningTime IS NOT NULL
ORDER BY P.RunningTime DESC
LIMIT 10
```

Query b:

Description of logic:

Compute the number of clips released per country in 2001.

SQL statement

```
SELECT COUNT(DISTINCT R.ClipId)
FROM ReleasedIn R
WHERE R.ReleaseDate LIKE '2001'
GROUP BY R.CountryId
```

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

Query c:

Description of logic:

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

SQL statement

```
SELECT COUNT(DISTINCT R.ClipId)
FROM Classified C
NATURAL JOIN ReleasedIn R
NATURAL JOIN Country C2
WHERE C2.CountryName = 'USA'
AND ( R.ReleaseDate LIKE '2014'
        OR R.ReleaseDate LIKE '2015'
        OR R.ReleaseDate LIKE '2016'
        OR R.ReleaseDate LIKE '2017'
        OR R.ReleaseDate LIKE '2018')
GROUP BY C.GenreId
```

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
NATURAL JOIN Acts A
GROUP BY S.StaffId
ORDER BY COUNT(DISTINCT A.ClipId) DESC
LIMIT 1
```

Query e:

Description of logic:

Print the maximum number of clips any director has directed

SQL statement

```
SELECT COUNT(DISTINCT D.ClipId)
FROM Directs D
GROUP BY D.StaffId
ORDER BY COUNT(DISTINCT D.ClipId) DESC
LIMIT 1
```

Query f:

Description of logic:

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

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

SQL statement

```
SELECT DISTINCT FullName FROM (
        SELECT staffid FROM (
                SELECT DISTINCT staffid, clipid FROM acts
                UNION ALL
                SELECT DISTINCT staffid, clipid FROM directs
                UNION ALL
                SELECT DISTINCT staffid, clipid FROM produces
                UNION ALL
                SELECT DISTINCT staffid, clipid FROM writes
        ) L GROUP BY staffid, ClipId HAVING SUM(1) >= 2
) R NATURAL JOIN MovieStaff
```

Query g:

Description of logic:

Print the 10 most common clip languages.

SQL statement

```
SELECT L.Language
FROM Speaks S
NATURAL JOIN Languages L
GROUP BY L.LanguageId
ORDER BY COUNT(DISTINCT S.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

```
SELECT S.FullName
FROM Acts A
NATURAL JOIN MovieStaff S
NATURAL JOIN Clips C
WHERE C.ClipType = 'V'
GROUP BY S.StaffId
ORDER BY COUNT(DISTINCT A.ClipId) DESC
LIMIT 1
```

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

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

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/

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Deliverable 3

Assumptions

We keep the same assumptions as in the previous deliverables. We assume stacked values need to be seperated with '|', with the exception to books in bibliographies where they are separated with '.|'.
We also assume Languages, Genres and Countries can exist in the database without having an associated clip.
In the queries when asked to compute X per Y and print X, we do not print Y since it would have been specified otherwise.

Query Implementation

Query a:

Description of logic:

Print the names of the top 10 actors ranked by the average rating of their 3 highest-rated clips that where voted by at least 100 people. The actors must have had a role in at least 5 clips (not necessarily rated).

We solve this in three steps: first we filter the actors that played in at least 5 clips, then we filter the actors what played in at least 3 clips with more than 100 votes, then for each staff we attribute to each clip an index depending on its rank (lowest index means highest rank) and do the average over the indices smaller than 3 (the 3 highest rated clips).

SQL statement

```
WITH CAL1 AS (
    -- filter the actors that played in at least 5 clips
    SELECT staffid FROM Acts A
        GROUP BY staffid
        HAVING COUNT(1) >= 5
), CAL2 AS (
    -- filter the actors that played in at least 3 clips with more than 100 votes
    SELECT staffid FROM CAL1
        NATURAL JOIN Acts
        NATURAL JOIN Ratings
        WHERE Votes >= 100
        GROUP BY StaffId
        HAVING COUNT(1) >= 3
), CAL3 AS (
    SELECT StaffId, Rank, ROW_NUMBER() OVER (PARTITION BY StaffId ORDER BY Rank
DESC) AS NUM
```

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/

```
    FROM ACTS
    NATURAL JOIN Ratings R
    WHERE EXISTS(
        SELECT 1 FROM CAL2
        WHERE CAL2.StaffId = Acts.StaffId)
    AND Votes >= 100
)

SELECT FullName FROM CAL3
NATURAL JOIN MovieStaff
WHERE NUM <= 3
GROUP BY FullName
ORDER BY AVG(Rank) DESC
LIMIT 10
```

Query b:

Description of logic:

Compute the average rating of the top-100 rated clips per decade in decreasing order.

From the clip year, we get the decade by dividing it by 10. Then for each decade, we assign an index to each clip of the decade depending on its rank (lowest index means highest rank) and do the average over the indices smaller than 100 (the 100 highest rated clips of the decade).

SQL statement

```
WITH Prep AS (
    SELECT Rank, ClipYear/10 AS Year, ROW_NUMBER() OVER (PARTITION BY ClipYear/10
ORDER BY Rank DESC) AS NUM
    FROM clips NATURAL JOIN Ratings
)

SELECT Year*10, AVG(Rank)
FROM Prep
WHERE NUM <= 100
GROUP BY Year
ORDER BY Year
```

Query c:

Description of logic:

For any video game director, print the first year he/she directed a game, his/her name and all his/her game titles from that year.

We first compute the minimal year for each video game developer and we just get the entries from directs with the same year.

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

SQL statement

```
WITH prep AS (
    SELECT StaffId, MIN(clipYear) as Year FROM Directs
    NATURAL JOIN Clips
    WHERE ClipType = 'VG'
    GROUP BY StaffId)

SELECT FullName, Year, ClipTitle
FROM prep
NATURAL JOIN Directs
NATURAL JOIN MovieStaff
NATURAL JOIN Clips C
WHERE C.ClipType = 'VG'
AND C.ClipYear = Year
```

Query d:

Description of logic:

For each year, print the title, year and rank-in-year of top 3 clips, based on their ranking.

For each year, we attribute an index to each clip depending on its rank and we print the clips with the 3 lowest indices (3 highest ranks).

SQL statement

```
SELECT ClipYear, ClipTitle, Rank FROM (
    SELECT ClipYear, ClipTitle, Rank, ROW_NUMBER() OVER (PARTITION BY ClipYear ORDER
BY Rank DESC) AS NUM
    FROM Clips
    NATURAL JOIN Ratings) L
WHERE NUM <= 3
```

Query e:

Description of logic:

Print the names of all directors who have also written scripts for clips, in all of which they were additionally actors (but not necessarily directors) and every clip they directed has at least two more points in ranking than any clip they wrote.

We first get the staffs where each (staff, clip) in writes matches those in acts and compute the maximum rank among those clips. We then get the clips directed by the staff and check if every rank is higher than 2 + the previously computed rank.

SQL statement

```
WITH Prep AS (
    SELECT w.staffid, MAX(R.Rank) FROM Writes W
```

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/

```
    NATURAL LEFT JOIN Ratings R
    GROUP BY W.StaffId
    HAVING EVERY(EXISTS(
        SELECT 1 FROM Acts A WHERE W.StaffId = A.StaffId AND W.ClipId = A.ClipId
    ))
)

SELECT S.FullName FROM Directs D
NATURAL LEFT JOIN Ratings R
NATURAL JOIN Prep P
NATURAL JOIN MovieStaff S
GROUP BY S.FullName
HAVING EVERY(CASE WHEN R.Rank IS NULL THEN 0 ELSE R.Rank END >= (P.Max + 2))
```

Query f:

Description of logic:

Print the names of the actors that are not married and have participated in more than 2 clips that they both acted in and co-directed it.

We first get all the rows (staff, clip) from acts that are also included in directs with role ‘co-director’ and where the staff is single. We then do a Cartesian product with itself and check if the clips match and group the pairs and count the number of rows for each pair, if more than 2 matches then the query is satisfied

SQL statement

```
WITH ActAndCodirect AS (
    SELECT A.StaffId, A.ClipId FROM Acts A
    INNER JOIN Directs D ON A.StaffId = D.StaffId AND A.ClipId = D.ClipId
    WHERE Role = 'co-director'
    AND NOT EXISTS(SELECT 1 FROM Spouses S WHERE S.StaffId = A.StaffId)
)

SELECT S1.FullName, S2.FullName
FROM ActAndCodirect AAC1 INNER JOIN MovieStaff S1 ON AAC1.StaffId = S1.StaffId,
ActAndCodirect AAC2 INNER JOIN MovieStaff S2 ON AAC2.StaffId = S2.StaffId
WHERE AAC1.StaffId < AAC2.StaffId
AND AAC1.ClipId = AAC2.ClipId
GROUP BY S1.FullName, S2.FullName
HAVING(Count(1) > 2)
```

Query g:

Description of logic:

Print the names of screenplay story writers who have worked with more than 2 producers.

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

We get the rows of directs where the worktype is screenplay and we try to find rows in produces with a matching clipid, we then group by staff and count the number of rows for each of them, this is the number of producers.

SQL statement

```
SELECT S.FullName
FROM Writes W NATURAL JOIN MovieStaff S
INNER JOIN Produces P ON W.ClipId = P.ClipId AND W.StaffId != P.StaffId
WHERE W.WorkType = 'screenplay'
GROUP BY S.FullName
HAVING COUNT(DISTINCT P.StaffId) > 2
```

Query h:

Description of logic:

Compute the average rating of an actor's clips (for each actor) when she/he has a leading role (first 3 credits in the clip).

We get the rows of acts with ordersCredit smaller than 3 and group by staff and aggregate with the average function

SQL statement

```
<The SQL statement>
SELECT AVG(Rank)
FROM Ratings
NATURAL JOIN Acts
WHERE OrdersCredit <= 3
GROUP BY StaffId
```

Query i:

Description of logic:

Compute the average rating for the clips whose genre is the most popular genre.

We first the get the most popular genre by counting the grouped rows for each genre from Classified and we get the rows from the same table where the genre matches the one found and we do the average.

SQL statement

```
SELECT AVG(Rank) FROM Classified C NATURAL JOIN Ratings R
WHERE C.GenreId = (SELECT C.genreId FROM Classified C
    GROUP BY C.genreId
    ORDER BY Count(*) DESC
    LIMIT 1)
```

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

Query j:

Description of logic:

Print the names of the actors that have participated in more than 100 clips, of which at least 60% where short but not comedies nor dramas, and have played in more comedies than double the dramas. Print also the number of comedies and dramas each of them participated in.

We first prepare 3 result tables: the comedy clips, the drama clips and the short clips. We then get the acts rows and group by staff and for each staff we count the needed values and filter with the help of the 3 result tables and some control/condition flow (Exists and Case). We then print the surviving staffs with the requested infos.

SQL statement

```
WITH Comedies AS (

    SELECT ClipId FROM Classified

    NATURAL JOIN Genres

    WHERE genre = 'Comedy'

), Dramas AS (

    SELECT ClipId FROM Classified

    NATURAL JOIN Genres

    WHERE genre = 'Drama'

), Shorts AS (

    SELECT ClipId FROM Classified

    NATURAL JOIN Genres

    WHERE genre = 'Short')


SELECT S.FullName,

    SUM(CASE WHEN EXISTS(SELECT 1 FROM Comedies C WHERE C.ClipId = A.ClipId) THEN
1 ELSE 0 END) AS Comedies,
```

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/

```
    SUM(CASE WHEN EXISTS(SELECT 1 FROM Dramas D WHERE D.ClipId = A.ClipId) THEN
1 ELSE 0 END) AS Dramas

FROM Acts A NATURAL JOIN MovieStaff S

GROUP BY S.StaffId

HAVING COUNT(*) > 100

AND 0.6*COUNT(*) <= SUM(

    CASE WHEN EXISTS(

        SELECT 1 FROM Shorts WHERE Shorts.ClipId = A.ClipId)

        AND NOT EXISTS(SELECT 1 FROM Comedies C WHERE C.ClipId = A.ClipId)

        AND NOT EXISTS(SELECT 1 FROM Dramas D WHERE D.ClipId = A.ClipId)

    THEN 1 ELSE 0 END)

AND SUM(CASE WHEN EXISTS(SELECT 1 FROM Comedies C WHERE C.ClipId = A.ClipId)
THEN 1 ELSE 0 END) >

    2*SUM(CASE WHEN EXISTS(SELECT 1 FROM Dramas D WHERE D.ClipId = A.ClipId)
THEN 1 ELSE 0 END)
```

Query k:

Description of logic:

Print the number of Dutch movies whose genre is the second most popular one.

We first get the second most popular genre from Classified with the OFFSET primitive and filter the clips that match the calculated genre and with language ‘Dutch’. We then count the number of rows.

SQL statement

```
SELECT COUNT(*) FROM Classified C

NATURAL JOIN Speaks S

NATURAL JOIN Languages L

WHERE C.genreId = (SELECT C.GenreId
```

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/

```
    FROM Classified C
    GROUP BY C.GenreId
    ORDER BY COUNT(*) DESC
    LIMIT 1 OFFSET 1)
AND L.language = 'Dutch'
```

Query l:

Description of logic:

Print the name of the producer whose role is coordinating producer, and who has produced the highest number of movies with the most popular genre.

We get the most popular genre from Classified and get the produces rows matching the genre and the role ‘coordinating producer’. We then group by staff (producers) and count the rows and print the producer with the highest number of rows.

SQL statement

```
SELECT S.FullName FROM Produces P
NATURAL JOIN MovieStaff S
NATURAL JOIN Classified C
WHERE P.Role = 'coordinating producer'
AND C.genreId = (SELECT C.GenreId
    FROM Classified C
    GROUP BY C.GenreId
    ORDER BY COUNT(*) DESC
    LIMIT 1)
GROUP BY S.StaffId
ORDER BY COUNT(C.ClipId) DESC LIMIT 1
```

Query Analysis

Selected Queries (and why)

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

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>

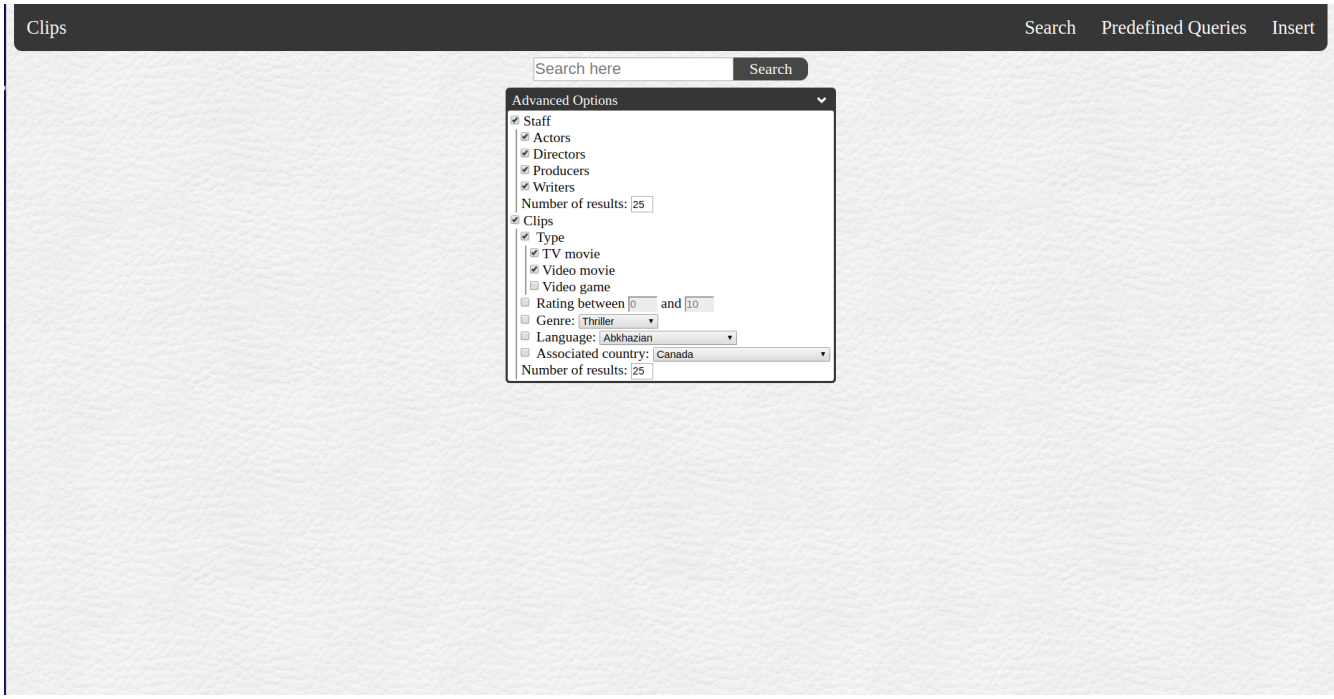
Interface

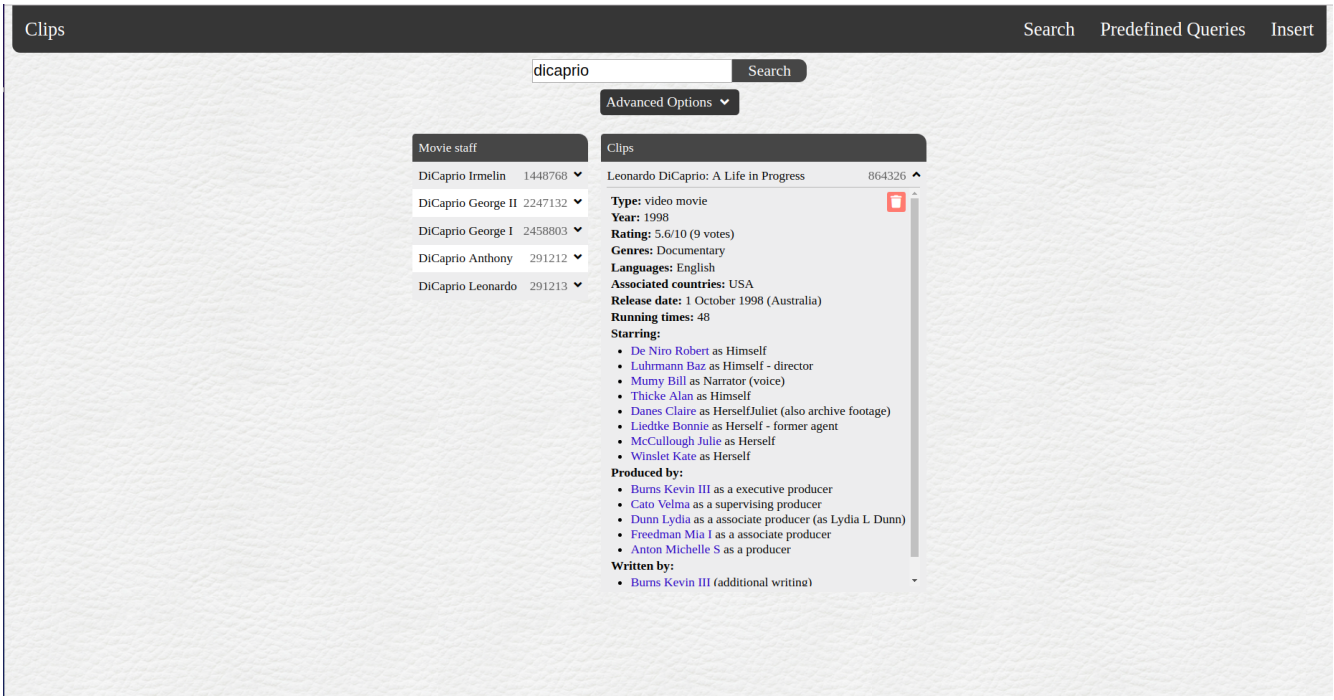
Design logic Description

For our interface we decided to use node.js which provides the necessary web files to be able to use the interface from the browser and also connects to our postgresql database to respond to the needed requests.
Our web page is composed of three main parts: search, predefined queries and insert.
The search page allows the user the possibility to search for clips and staffs, the search is based on substring matching (‘Capri’ will match ‘DiCaprio’). The user can also use the ‘advanced options’ button which includes checkboxes and dropdown-menus to make more specific searches. The results are shown below with the name of the clips and staffs corresponding to the search, the user can click on a result to have more detailed information about it. We also decided to include the delete button here instead of the last page because it made more sense to have a look to the detailed informations before erasing the precious content.
The second page ‘predefined queries’ is just a list of buttons (from ‘2.a’ to ‘3.l’) the user can click to show the corresponding query and sql, he can then click on a button to execute the query and display the results.
Finally the last page allows the user to insert a new now in any of our database tables.

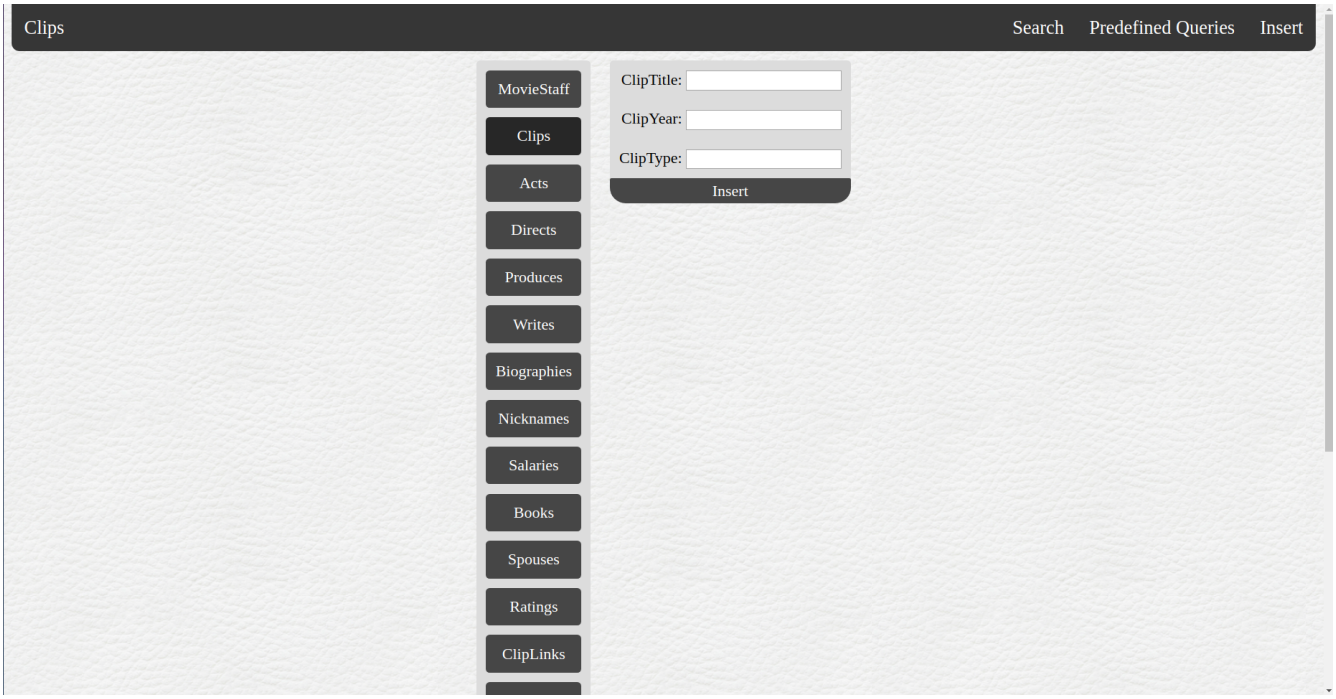
DIAS: Data-Intensive Applications and Systems Laboratory
School of Computer and Communication Sciences
Ecole Polytechnique Fédérale de Lausanne
Building BC, Station 14
CH-1015 Lausanne
URL: http://dias.epfl.ch/

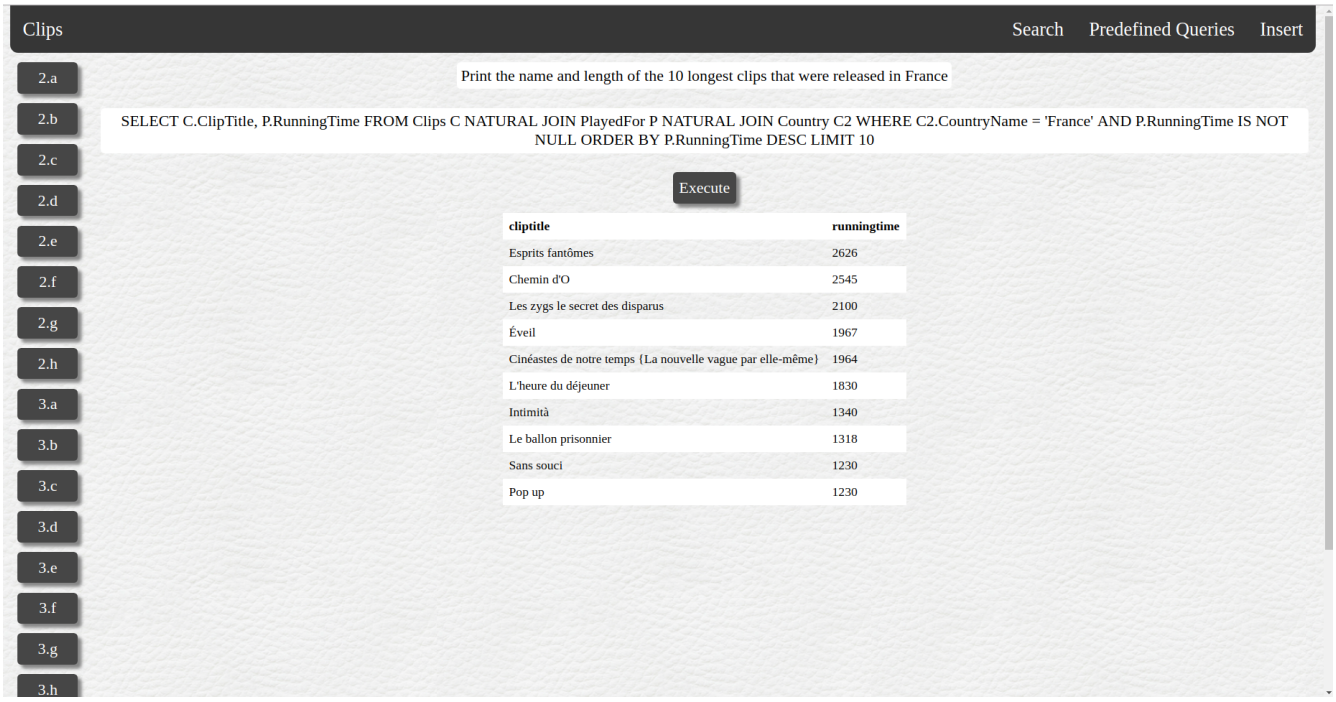
Screenshots





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