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: 17

Names: Cho Hyun Jii, Poopalasingam Kirusanth, Reetz Florian

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/

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

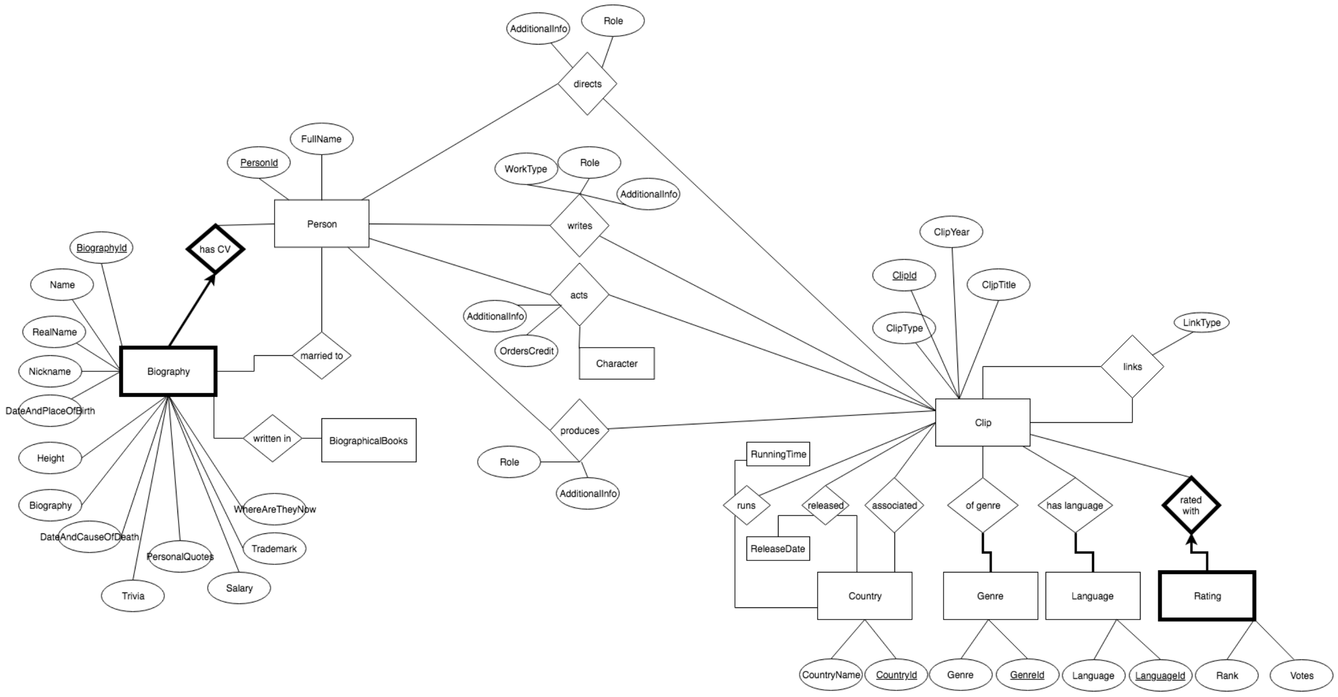
We make the listed design decisions based on the following assumptions:

- *Actors, Writers, Producers, Directors* are not modelled as entities but as relations between entity *Person* and entity *Clip* because one *Person* can perform multiple jobs and we assume that queries become easier if the jobs are relations.
- *ReleaseDate* and *RunningTime* have been moved to a single entity since both describe additional information about a clip released in a country.
- *Biography* has been modelled as one entity since we don't know the exact data yet which could justify a subdivision.
- We model *Country* as a separate entity to be able to describe other relations to *Country* like we do with the *released* relation.
- *Rating* is modelled as a weak entity since it is only associated to a single *Clip* (a *Rating* without an existing *Clip* does not make sense)
- *Biography* is also modelled as a weak entity (we cannot have a *Biography* for a non-existing *Person*)
- We consider the attribute "language" of entity *Language* a primary key because it is a unique and necessary attribute which cannot change.

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/

Entity Relationship Schema

Schema



Description

We apply the following constraints

Key Constraints:

- One *Biography* is only associated with one *Person*. A *Person* can have many *Biographies*. (One-to-Many)
- One *Rating* is associated to one *Clip*. A *Clip* can have many *Ratings*. (One-to-Many)
- Every other relation is modelled as many-to-many (e.g. every *Person* can direct, write, act or produce in many *Clips*. A *Clip* can have many producer, director, actor or writer. Every *Clip* can have many *Countries* associated. A *Country* can be associated with many *Clips*.)
- In principle, a *Clip* should have at least one *Country* and at least one *Language* but we do not want to enforce this because we do not know the data yet.

Participation Constraints:

- Each *Rating* is assigned to exactly one *Clip*.
- Each *Biography* is assigned to exactly one *Person*
- Each *Language* and *Genres* must have at least one *Clip* to be relevant to this database.

- Every other relation participates partially (e.g. *Country*: A *Country* does not need to be associated with a *Clip* if it is a *Country* where a *Clip* is released. A *Clip* does not need to have a *Country* association)

Relational Schema

ER schema to Relational schema

The following describes how we translate the constraints

- Translating the many-to-many relationships "directed, acted, produced, wrote" we did not use a superkey as primary key (i.e. primary key(person_id,clip_id)) because this key does not allow for one person to act two different roles in the same clip.
- In contrast, translating the many-to-many relationships "clip_country, clip_genre, clip_language" we use a superkey as primary key (e.g. primary key(clip_id,genre_id))
- The key constraints in "clip_rating" and "biography" are enforced by setting the foreign key NOT NULL.

The following attributes are defined as unique:

- Country(country_name)
- Genre(Genre)

DDL

```
CREATE TABLE Person
(
  person_id INTEGER,
  fullname  CHAR(20),
  PRIMARY KEY (person_id)
);

CREATE TABLE Clip
(
  clip_id    INTEGER,
  clip_type  CHAR(20),
  clip_year  DATE,
  clip_title CHAR(20),
  PRIMARY KEY (clip_id)
);

CREATE TABLE Directs
(
  person_id       INTEGER,
  clip_id         INTEGER,
  additional_info CHAR(200),
  role            CHAR(20),
  PRIMARY KEY (person_id, clip_id),
  FOREIGN KEY (person_id) REFERENCES Person (person_id),
  FOREIGN KEY (clip_id) REFERENCES Clip (clip_id)
);

CREATE TABLE Acts
(
  person_id       INTEGER,
  clip_id         INTEGER,
  additional_info CHAR(200),
  orders_credit   CHAR(20),
  character       CHAR(20),
  PRIMARY KEY (person_id, clip_id, character),
  FOREIGN KEY (person_id) REFERENCES Person (person_id),
  FOREIGN KEY (clip_id) REFERENCES Clip (clip_id)
);

CREATE TABLE Produces
(
  person_id       INTEGER,
  clip_id         INTEGER,
  additional_info CHAR(200),
  role            CHAR(20),
  PRIMARY KEY (person_id, clip_id),
  FOREIGN KEY (person_id) REFERENCES Person (person_id),
  FOREIGN KEY (clip_id) REFERENCES Clip (clip_id)
);

CREATE TABLE Writes
(
  person_id       INTEGER,
  clip_id         INTEGER,
  additional_info CHAR(200),
  work_type       CHAR(20),
  role            CHAR(200),
  PRIMARY KEY (person_id, clip_id),
  FOREIGN KEY (person_id) REFERENCES Person (person_id),
  FOREIGN KEY (clip_id) REFERENCES Clip (clip_id)
);

CREATE TABLE ClipLinks
(
  -- we cannot use clip_from_id and clip_to_id as PK, since it's not unique
  cliplink_id  INTEGER,
  clip_from_id INTEGER,
  clip_to_id   INTEGER,
  link_type    CHAR(255),
  PRIMARY KEY (cliplink_id),
  FOREIGN KEY (clip_from_id) REFERENCES Clip (clip_id),
  FOREIGN KEY (clip_to_id) REFERENCES Clip (clip_id)
);

CREATE TABLE Country
(
  country_id  INTEGER,
  countryname CHAR(100),
  PRIMARY KEY (country_id),
  constraint ux_country unique (countryname)
);

CREATE TABLE Genre
(
  genre_id INTEGER,
  genre    CHAR(20),
  PRIMARY KEY (genre_id),
  constraint ux_genre unique (genre)
);

CREATE TABLE Language
(
  language_id INTEGER,
  language    CHAR(20),
  PRIMARY KEY (language_id),
  constraint ux_language unique (language)
);

CREATE TABLE Clip_country
(
  clip_id    INTEGER,
  country_id INTEGER,
  PRIMARY KEY (clip_id, country_id),
  FOREIGN KEY (clip_id) REFERENCES Clip (clip_id),
  FOREIGN KEY (country_id) REFERENCES Country (country_id)
);

CREATE TABLE Clip_genre
(
  clip_id  INTEGER,
  genre_id INTEGER,
  PRIMARY KEY (clip_id, genre_id),
  FOREIGN KEY (clip_id) REFERENCES Clip (clip_id),
  FOREIGN KEY (genre_id) REFERENCES Genre (genre_id)
);

CREATE TABLE Clip_language
(
  clip_id     INTEGER,
  language_id INTEGER,
  PRIMARY KEY (clip_id, language_id),
  FOREIGN KEY (clip_id) REFERENCES Clip (clip_id),
  FOREIGN KEY (language_id) REFERENCES Language (language_id)
);

CREATE TABLE Clip_rating
(
  clip_id   INTEGER NOT NULL,
  rating_id INTEGER,
  rank      NUMBER(10),
  votes     NUMBER(10),
  PRIMARY KEY (rating_id),
  FOREIGN KEY (clip_id) REFERENCES Clip (clip_id)
    ON DELETE CASCADE
);

CREATE TABLE Released
(
  clip_id      INTEGER,
  country_id   INTEGER,
  release_date DATE,
  PRIMARY KEY (clip_id, country_id),
  FOREIGN KEY (clip_id) REFERENCES Clip (clip_id),
  FOREIGN KEY (country_id) REFERENCES Country (country_id)
);

CREATE TABLE Runs
(
  clip_id      INTEGER,
  country_id   INTEGER,
  running_time NUMBER(10),
  PRIMARY KEY (clip_id, country_id),
  FOREIGN KEY (clip_id) REFERENCES Clip (clip_id),
  FOREIGN KEY (country_id) REFERENCES Country (country_id)
);

CREATE TABLE Biography
(
  biography_id      INTEGER,
  name              CHAR(20),
  realname          CHAR(20),
  nickname          CHAR(20),
  birth_date        DATE,
  birth_place       CHAR(20),
  height            CHAR(20),
  biography         CHAR(400),
  biographer        CHAR(20),
  death_date        DATE,
  death_place       CHAR(20),
  trivia            CHAR(200),
  biographicalbooks CHAR(100),
  personalquotes    CHAR(200),
  salary            CHAR(20),
  trademark         CHAR(20),
  wherenow          CHAR(200),
  person_id         INTEGER NOT NULL,
  FOREIGN KEY (person_id) REFERENCES Person (person_id)
    ON DELETE CASCADE,
  PRIMARY KEY (biography_id)
);

CREATE TABLE BiographicalBooks
(
  book_id      INTEGER,
  title        CHAR(100),
  biography_id INTEGER NOT NULL,
  FOREIGN KEY (biography_id) REFERENCES Biography (biography_id),
  PRIMARY KEY (book_id)
);

CREATE TABLE Married_to
(
  married_id  INTEGER NOT NULL,
  biography_id  INTEGER NOT NULL,
  person_id INTEGER NOT NULL,
  date CHAR(50),
  state CHAR(20),
  children CHAR(20),
  PRIMARY KEY (married_id)
  FOREIGN KEY (biography_id) REFERENCES Biography (biography_id),
  FOREIGN KEY (person_id) REFERENCES Person (person_id)
);
```

General Comments

Work allocation

Cho : DDL commands
Poopalasingam : ER model
Reetz : revision and comments

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

- All strings are properly represented by encoding them as 'utf-8'.
- There is a pair of ClipId's (719315,719344) which are equal in all attributes. We do not treat it as duplicate because we take the ClipId's as given.
- Languages with/without brackets and languages with additional descriptions like 'version' are considered one language. This is not assumed for 'subtitles' or unfamiliar languages.
- Dates given in 'release_date' and 'biography' are assumed to have the format '01 January 1999' and converted into the oracle date format. If days or months are not given, we assume first day of the month and the first month of the year.
- References to many "clips" in actors, directors, etc. is given in the format '[info1 | info2]'. Each bracket entry is written into a separate row.
- Persons are not given with a unique identifier. Thus, we assume the FullName to be unique. Unique names are taken from the combined tables of biographies, actors, producers, writers, directors and they are associated with an integer ID ordered along the alphabetical order of names.
- Biographies provides information with string characters. We convert this information into numerical information for birth/death date (see above) and for height. The latter is given in either >># cm<< or >>#' # ½"<< which are converted into "cm" units.
- Spouses are given with names (enclosed by ' ') and additional information. The names are added to the person relation.

Data Loading

The data has been cleaned and sorted using the python library "pandas". A connection the Oracle SQL server was set up using the library "sqlalchemy".

In order to tests the imports and save time we use SQLite locally instead of Oracle SQL server. For

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.CLIP_ID,
  C.CLIP_TITLE,
  sum(R.RUNNING_TIME) as runtime
FROM CLIP c
  JOIN CLIP_COUNTRY cc ON cc.CLIP_ID = c.CLIP_ID
  JOIN COUNTRY C2 ON cc.COUNTRY_ID = C2.COUNTRY_ID
  JOIN RUNS R ON c.CLIP_ID = R.CLIP_ID
WHERE C2.COUNTRYNAME = 'France'
GROUP BY c.CLIP_ID, c.CLIP_TITLE
order by runtime desc, c.CLIP_ID, C.CLIP_TITLE
FETCH FIRST 10 ROWS ONLY;
```

Query b:

Description of logic:

Compute the number of clips released per country in 2001

SQL statement

```
SELECT
  c2.COUNTRYNAME,
  count(*) as nb_of_clips
FROM CLIP c
  JOIN RELEASED R ON c.CLIP_ID = R.CLIP_ID
  JOIN COUNTRY C2 ON R.COUNTRY_ID = C2.COUNTRY_ID
WHERE extract(YEAR FROM r.RELEASE_DATE) = 2001
GROUP BY c2.COUNTRYNAME
order by c2.COUNTRYNAME;
```

Query c:

Description of logic:

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

SQL statement

```
SELECT
  G.GENRE,
  count(*) as nb_of_clips
FROM CLIP c
  JOIN CLIP_GENRE CG ON CG.CLIP_ID = C.CLIP_ID
  JOIN RELEASED R ON c.CLIP_ID = R.CLIP_ID
  JOIN COUNTRY C2 ON R.COUNTRY_ID = C2.COUNTRY_ID
  JOIN GENRE G ON CG.GENRE_ID = G.GENRE_ID
WHERE extract(YEAR FROM r.RELEASE_DATE) > 2013
      AND C2.COUNTRYNAME = 'USA'
GROUP BY G.GENRE
order by g.GENRE;
```

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 d:

Description of logic:

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

SQL statement

```
SELECT d.FULLNAME
FROM
  (SELECT
     P.FULLNAME,
     count(*) AS nb_acts
   FROM PERSON P
     JOIN ACTS A2 ON P.PERSON_ID = A2.PERSON_ID
     JOIN CLIP C2 ON A2.CLIP_ID = C2.CLIP_ID
   GROUP BY P.FULLNAME
   ORDER BY nb_acts DESC
   FETCH FIRST 1 ROW ONLY
  ) d;
```

Query e:

Description of logic:

Print the maximum number of clips any director has directed.

SQL statement

```
SELECT d.FULLNAME
FROM
  (
    SELECT
      P.FULLNAME,
      count(*) AS nb_acts
    FROM PERSON P
      JOIN DIRECTS D2 ON P.PERSON_ID = D2.PERSON_ID
      JOIN CLIP C2 ON D2.CLIP_ID = C2.CLIP_ID
    GROUP BY P.FULLNAME
    ORDER BY nb_acts DESC
    FETCH FIRST 1 ROWS ONLY
  ) d;
```

Query f:

Description of logic:

Print the names of people that had at least 2 different jobs in a single clip. For example, if X has both acted, directed and written movie Y, his/her name should be printed out. On the other hand, if X has acted as 4 different personas in the same clip, but done nothing else, he/she should not be printed.

SQL statement

```
SELECT p.FULLNAME
FROM (
       SELECT
         c.CLIP_ID,
         P.PERSON_ID,
         count(a.CLIP_ID) AS acts,
         count(d.CLIP_ID) AS directs,
         count(w.CLIP_ID) AS writes
       FROM CLIP C, PERSON P
         LEFT JOIN ACTS a ON a.PERSON_ID = p.PERSON_ID
         LEFT JOIN DIRECTS d ON d.PERSON_ID = p.PERSON_ID
         LEFT JOIN WRITES w ON w.PERSON_ID = p.PERSON_ID
       WHERE
         a.CLIP_ID = c.CLIP_ID AND
         d.CLIP_ID = c.CLIP_ID AND
         w.CLIP_ID = c.CLIP_ID
       GROUP BY C.CLIP_ID, P.PERSON_ID
       HAVING (count(a.CLIP_ID) > 0 AND count(d.CLIP_ID) > 0) OR
              (count(d.CLIP_ID) > 0 AND count(w.CLIP_ID) > 0) OR
              (count(a.CLIP_ID) > 0 AND count(w.CLIP_ID) > 0)
     ) m
  JOIN PERSON p ON p.PERSON_ID = m.PERSON_ID;
```

Query g:

Description of logic:

Print the 10 most common clip languages

SQL statement

```
SELECT d.LANGUAGE
FROM
  (
    SELECT
      L.LANGUAGE,
      count(*) AS count
    FROM CLIP_LANGUAGE
      JOIN LANGUAGE L ON CLIP_LANGUAGE.LANGUAGE_ID = L.LANGUAGE_ID
    GROUP BY L.LANGUAGE
    ORDER BY d.count DESC
    FETCH FIRST 10 ROWS ONLY
  ) d
;
```

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 p.FULLNAME
FROM (
       SELECT
         a.PERSON_ID,
         count(*) AS count
       FROM ACTS A
         JOIN CLIP C2 ON A.CLIP_ID = C2.CLIP_ID
       WHERE C2.CLIP_TYPE = 'V'
       GROUP BY a.PERSON_ID
     ) b
  JOIN PERSON p ON p.PERSON_ID = b.PERSON_ID
ORDER BY b.count DESC;
```

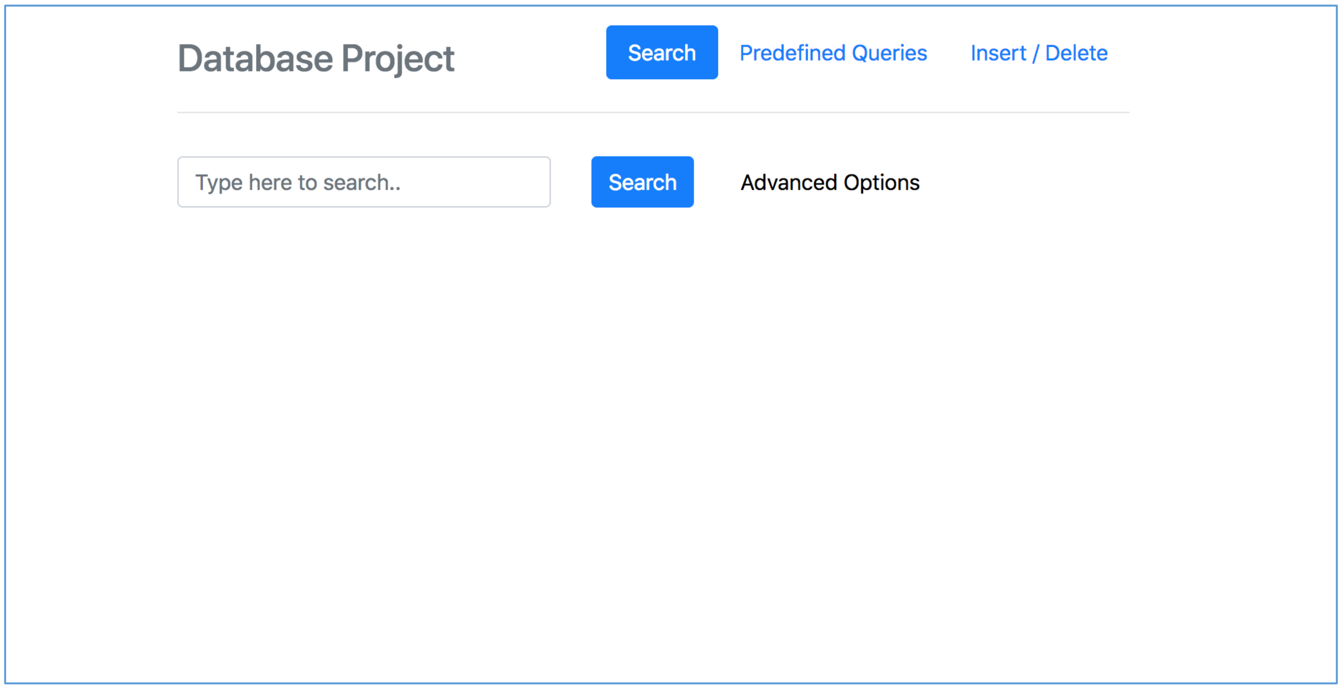
Interface

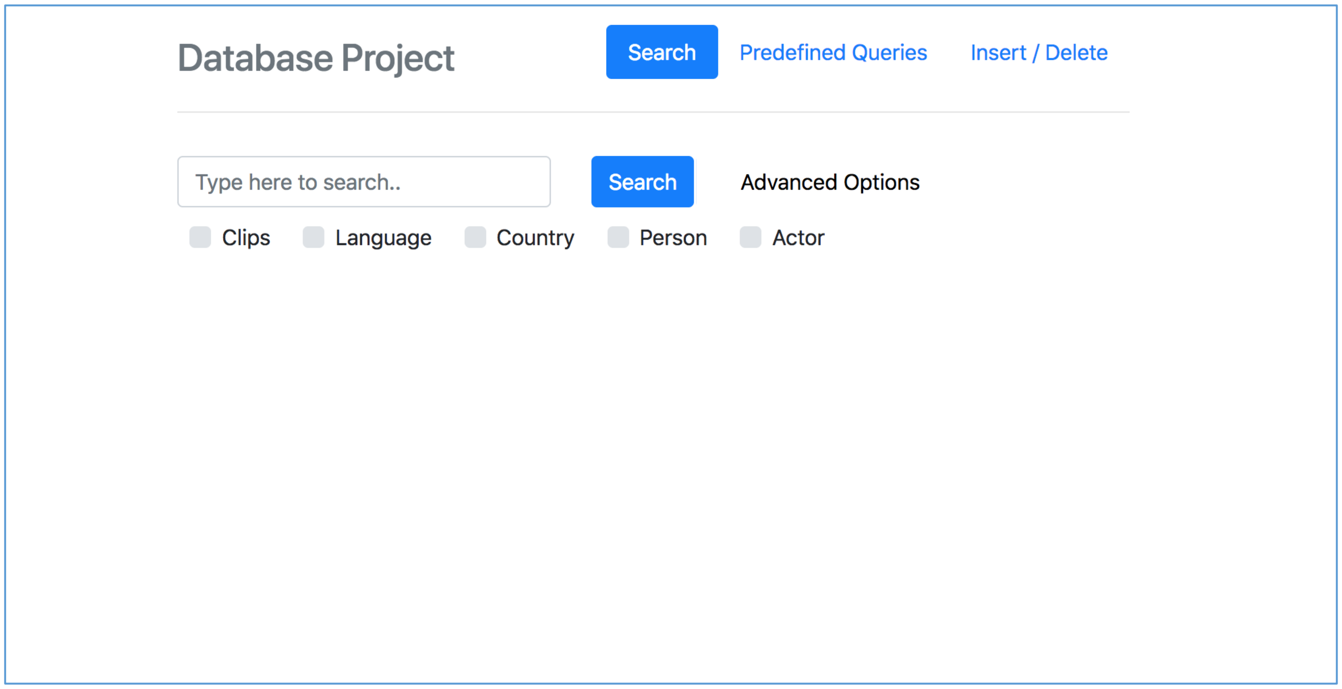
Design logic Description

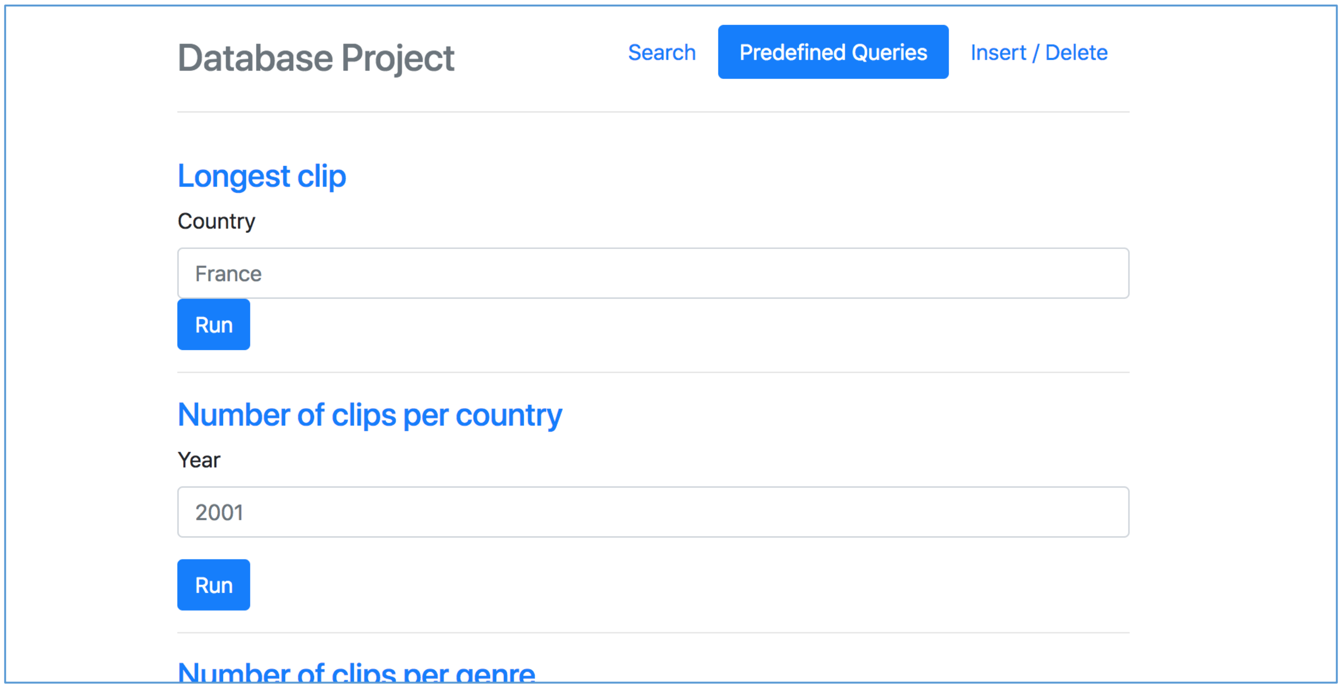
The user interface is a simple web page, where user can search for relevant entities, and see the result. The backend is developed in Python with Flask library. The web page is based on Bootstrap 4 CSS Framework.

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/







Longest clip

Country: France

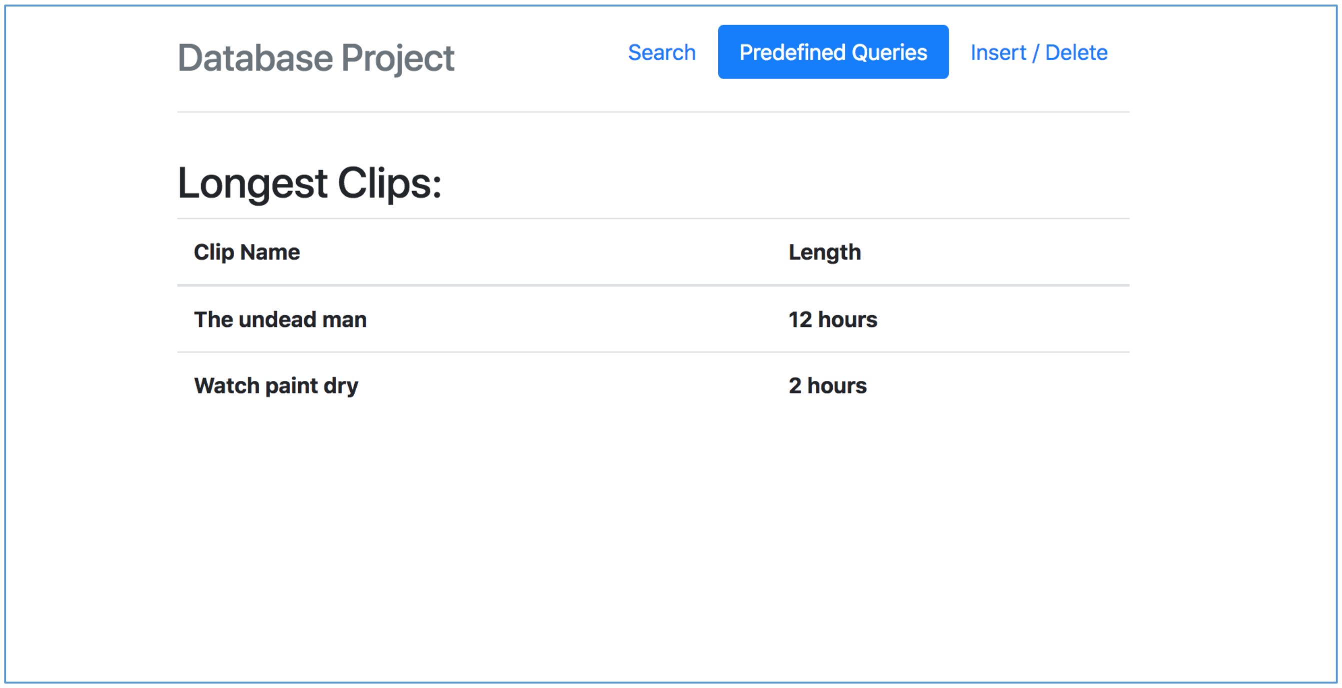
Run

Number of clips per country

Year: 2001

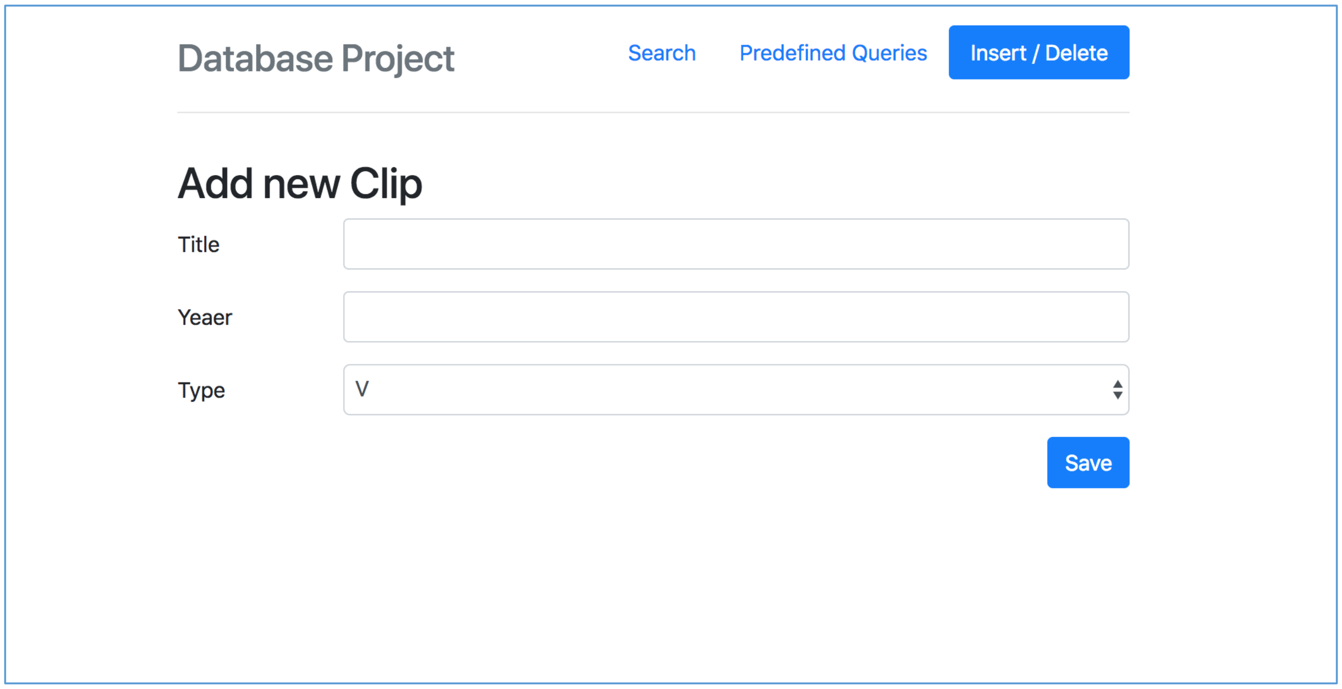
Run

Number of clips per genre



Longest Clips:

Clip Name	Length
The undead man	12 hours
Watch paint dry	2 hours



Add new Clip

Title	
Yeaer	
Type	V

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

<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>