Databases Project – Spring 2018

Team No: 17

Names: Cho Hyun Jii, Poopalasingam Kirusanth, Reetz Florian

Contents

Contents 1

Deliverable 1 2

Assumptions 2

Entity Relationship Schema 2

Schema 2

Description 2

Relational Schema 2

ER schema to Relational schema 2

DDL 3

General Comments 3

Deliverable 2 4

Assumptions 4

Data Loading 4

Query Implementation 4

Query a: 4

Description of logic: 4

SQL statement 4

Interface 4

Design logic Description 4

Screenshots 4

General Comments 4

Deliverable 3 5

Assumptions 5

Query Implementation 5

Query a: 5

Description of logic: 5

SQL statement 5

Query Analysis 5

Selected Queries (and why) 5

Query 1 5

Query 2 5

Query 3 5

Interface 6

Design logic Description 6

Screenshots 6

General Comments 6

# 

# Deliverable 1

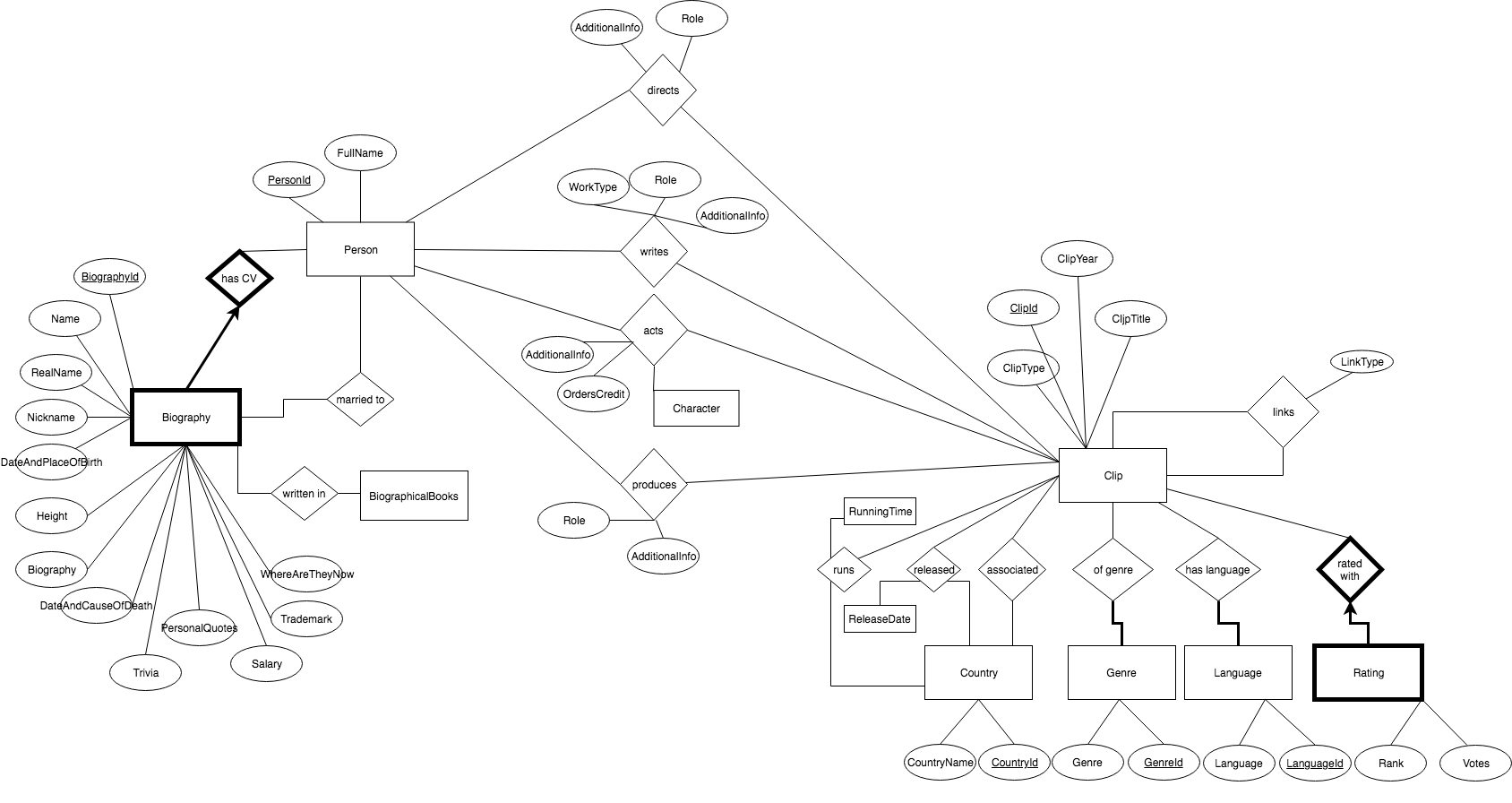
## Assumptions

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.

## 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

(clip\_from\_id INTEGER,

clip\_to\_id INTEGER,

link\_type CHAR(255),

PRIMARY KEY (clip\_from\_id,clip\_to\_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));

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) REFERENCES Language(language));

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),

spouse\_id INTEGER,

person\_id INTEGER NOT NULL,

FOREIGN KEY (person\_id) REFERENCES Person(person\_id) ON DELETE CASCADE,

FOREIGN KEY (spouse\_id) REFERENCES Clip(person\_id),

PRIMARY KEY (biography\_id));

CREATE TABLE BiographicalBooks  
 (book\_id INTEGER,  
 biography\_id INTEGER NOT NULL,

title CHAR(100),

FOREIGN KEY (biography\_id) REFERENCES Clip(biography\_id),

PRIMARY KEY (book\_id));

## General Comments

Work allocation  
Cho : DDL commands

Poopalasingam : ER model

Reetz : revision and comments

# Deliverable 2

## Assumptions

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

## Data Loading

## Query Implementation

<For each query>

### 

### Query a:

#### Description of logic:

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

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

#### SQL statement

**SELECT** *\****FROM** (  
 **SELECT  
 C**.**CLIP\_ID**,  
 **C**.**CLIP\_TITLE**,  
 *sum*(R.**RUNNING\_TIME**)  
 **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**)  
**WHERE** *ROWNUM* < 10;

### Query b:

#### Description of logic:

Compute the number of clips released per country in 2001

#### SQL statement

**SELECT** c2.**COUNTRYNAME**,  
 *count*(*\**)  
**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**;

### Query c:

#### Description of logic:

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

#### SQL statement

**SELECT** G.**GENRE**,  
 *count*(*\**)  
**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**;

### Query d:

#### Description of logic:

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

#### SQL statement

**SELECT** m.**FULLNAME  
FROM** (  
 **SELECT** *\** **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** ) d  
 **ORDER BY** d.nb\_acts **DESC** ) m  
**WHERE** *ROWNUM* < 1;

### Query e:

#### Description of logic:

#### Print the maximum number of clips any director has directed.

#### SQL statement

**SELECT** m.**FULLNAME  
FROM** (  
 **SELECT** *\** **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** ) d  
 **ORDER BY** d.nb\_acts **DESC** ) m  
**WHERE** *ROWNUM* < 1;

### 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

**TBD**

### Query g:

#### Description of logic:

#### Print the 10 most common clip languages

#### SQL statement

**SELECT** m.**LANGUAGE  
FROM** (  
 **SELECT** *\** **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** ) d  
 **ORDER BY** d.count **DESC** ) m  
**WHERE** *ROWNUM* < 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

**TBD**

## 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>

# 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>

# 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>