**IMDB Project**

**Milestone 1 partial report**

**0. Summary**

[1. Introduction 3](#_Toc413839304)

[1.1. Technologies 3](#_Toc413839305)

# Introduction

The goal of this report is to present our work for the project due to EPFL’s CS-322 “Introduction to database systems” course. Realized application consists of a database designed and implemented on the basis of simple data files and short description given at the beginning of the project, coupled to a simple web application to manage and query this database. The stored data corresponds to some movies and series with simple relations such as actors, extracted from the *IMDB movies database[[1]](#footnote-1)*.

## Technologies

As we had full choice about the tools and software we use for this project, here is a list of the main components we used and the reason of “why these and not others ?”.

### Database management system (DBMS)

We were offered to use some dedicated *Oracle[[2]](#footnote-2)* server, internal to EPFL. The major inconvenient of this solution was that we won’t be able to work on this project in locations where no easy Internet access is provided, like trains, and we would become dependent of the availability of the server.

Instead, we chose to locally install and use *MySQL Community Edition[[3]](#footnote-3)*, with the following argument about this choice.

* It is totally free and wide-used, essentially in the world of web applications, which kind of application we chosed to build (see 1.1.2) ;
* it comes with *MySQL Workbench[[4]](#footnote-4)*, a powerful dedicated tool to model schemas, manage graphically the DBMS parameters, check queries are working and so on ;
* it is available on all major OS, which can be useful as we have different kinds of computers with different OS families installed.

More precisely, we use the following versions of the DBMS and tools.

* MySQL 5.6.23
* MySQL Workbench 6.2.4
* MySQL Utilities 1.5.3

### Graphical application development

We had full choice about the programming language and API technologies we want to use to develop the graphical application that has to come out as a result for this project.

We chosed to develop a web application written in PHP, potentially using Javascript to add more interactivity. The exact tools like webserver, PHP version and other details are not decided yet, because this is not part of the first milestone for this project.

# Conceptual design

In this chapter, we analyse the data and metadata that are given at the start point of the project and mix them together into a Entity-Relational (ER) schema and a set of out-of-model constraints.

## Metadata and data analysis

Here we do a hypothesis-based analysis of parts of the data and their metadata (in fact, just column names) we were given. This will be used in 2.2 to justify parts of the ER schema.

### “PERSON” and “ALTERNATIVE\_NAME”

This 2 files are obviously linked by the fact that a person can have multiple alternative names, which is confirmed by the presence of the *“person\_id”* field in *“ALTERNATIVE\_NAME”*. In *“PERSON”* as well as in *“ALTERNATIVE\_NAME”*, the *“name”* field seems to be always constructed in the same way, starting by the last name, followed by a comma, and then the first name(s). Sometimes, there is only one name without coma, suggesting it’s just the “main name”, let’s say last name. This field is to be split into 2 distinct fields “lastname” and “firstname” by parsing during the data import.

We also remark that the field *“gender”* in *“PERSON”* contains always values “m” and “f”, obviously describing if the person is a man or a woman. This is a candidate field to be exported as a separate entity set in ER schema.

### “PRODUCTION” and “ALTERNATIVE\_TITLE”

This 2 files are also linked in a similar way as “PERSON” and “ALTERNATIVE\_NAME” (see above).

By digging into “PRODUCTION”, we see that there are 3 types of productions to consider.

* The single movies, that are registered once with all their informations ;
* the series, that are registered as a grouping element for episodes that they contain ;
* the episodes, linked to particular serie through the *“series\_id”* field, and also registering a particular *“season\_number”* and *“episode\_number”* relative to the serie it belongs to.

This repartition of productions is a clear candidate to be modelled as a “ISA” hierarchy in the ER schema. More importantly, the *season* concept which is only present through the *“season\_number”* field is a perfect candidate to be exported as a separate entity set in the ER schema, adding a level of hierarchy and thus helping to ensure consistency of the data.

The *“gender”* field of *“PRODUCTION”* is also a good candidate for such export, as it contains highly-repeated values that cleary consist of a finite set of genders of movies.

### “COMPANY” and “PRODUCTION\_COMPANY”

The *“COMPANY”* file contains listing of companies. By digging into it, we can rapidly show the following particularities.

* The *“country\_code”* field is highly-repeating, so it is a candidate to be exported in a separate entity set in the ER schema ;
* the *“company\_type”* field is in a similar situation, thus also a candidate for such export.

With more attention, a more complex structure can be seen. Firstly, we remark that some keys of the field *“id”* are repeating with the same name of enterprise, only changing the *“company\_type”* field. This is an indication that a single company can have multiple types, giving an indication of the cardinality of the corresponding relation in the ER schema. Then, we see that a company can be present in multiple countries, but with changing *“id”*. A good example of all these variations can be found in the first 100 rows of the file, which is presented below.

**Id country\_code name company\_type**

71 [be] Sony Pictures Home Entertainment distributors

72 [nl] Sony Pictures Home Entertainment distributors

72 [nl] Sony Pictures Home Entertainment production companies

73 [us] Sony Pictures Home Entertainment distributors

73 [us] Sony Pictures Home Entertainment production companies

The *“PRODUCTION\_COMPANY”* file acts as an associative relationship between *“PRODUCTION”* and *“COMPANY”*. By digging into it, we easily see that a company can be involved in multiple productions, and that a production needs the participation of multiple companies, thus giving a N:N relationship in the incoming ER schema.

### “PRODUCTION\_CAST” and “CHARACTER”

The *“PRODUCTION\_CAST”* file acts as an associative relationship between *“PRODUCTION”*, “PERSON” and *“CHARACTER”*, with a supplementary *“role”* value. This value is clearly repeating, making it a good candidate to be modelled as a separate entity set in the ER schema. By exploring the file, we also see that the *“production\_id”*, *“person\_id”* and *“role”* are always filled, thus always telling that a person participated in a production as a particular role. This is to be modelled as a ternary relationship set in the ER model. But because the *“character\_id”* is not always filled, we need to be careful and think about modelling it as a second relationship set, linking the aggregation of our first relationship set to the entity set corresponding to the file *“CHARACTER”*.

Finally, the *“CHARACTER”* file is just a listing of the characters played in movies. By making a quick test with the search function of a basic text editor into *“PRODUCTION\_CAST”*, it seems that some characters appears in multiple productions, either referenced by the exact same key or by having multiple instances of it in the *“CHARACTER”* file with slightly different names (e.g. “James Bond”, “James Bond 007”, “007 James Bond”, …). This will be hard to make this proper, but this is not the purpose of the first milestone of this project.

1. <http://www.imdb.com> [↑](#footnote-ref-1)
2. <http://www.oracle.com/fr/database/overview/index.html> [↑](#footnote-ref-2)
3. <http://dev.mysql.com/downloads/> [↑](#footnote-ref-3)
4. <http://dev.mysql.com/downloads/workbench/> [↑](#footnote-ref-4)