Nutflux

Nicolas Di Costanzo 21209060
nicolas.dicostanzo@ucdconnect.ie

Use a splash page image here [optional]

Use *LaTeX* if you wish, but use the general spacing and font/style you find here (1.5 spacing, 12 point font for text, etc.).

Be sure to submit a PDF (not a .DOC file) as your report. Overall it should be **30 pages or more**, including diagrams and screenshots. A significant portion of the report should be textual. Do not rely on images to write your report for you.

Remember, your project this year concerns a database for a streaming platform. Your database is intended to support a data-driven, knowledge-based approach to content selection. As such, identify the place of the database in the overall platform, and tell us how you would support its operations at the SQL level.

What to submit: This report, as a PDF, *and* the necessary SQL files to allow us to examine your database constructs and test your queries.

Contents

1. Introduction

This project will try to describe, at the data level, what a visual content database could be for different kind of target audience. With this database design, I will try to capture all relevant information for a visual content application. A website and/or a mobile application that has two main objectives:
- to provide content to the user, movies, series, shows etc...
- to provide information about this content.

The first point is not described in the database. Indeed, it is not a data modelling problem. However, each entity that represents a content would be linked to an audio-visual content present on a server. But the existence of this audio-visual content is not captured by the database design itself although it should be understood that each entity in the CONTENT table represents, in addition to the information recorded by the database, the corresponding audio-visual file.

The second point concerns the information contained by the application. This is where the work on the database modelling can make the application relevant (or not). Relevant modelling can not only lead users to find the content they are looking for if they have a specific idea in mind, but also to make relevant suggestions based on the content they have previously consumed. Also, the addition of relatively unusual information, such as quotes and information about the private lives of people related to a piece of content, can help to engage the user and make them more active in the content they are consuming.

There are two main audiences targeted by this application. “Movie nuts” (pro-users) and casual viewers (standard-users). The base must contain information that are relevant for both. All data relevant to standard-users will be relevant to pro-users but all data relevant to pro-users will not necessarily be relevant to standard-users. However, the database must contain both set of data. The difference will be managed in the way the information appears to users. And this information and the way it is displayed will depend on the status of the user (standard or pro user). The database design does not make any difference

Commented [NDC1]: Introduce your vision of the project here. Describe the domain of the project, and the intended application. Characterize the nature and scale of the data you are working with. What role does your database design play within the larger system (as the database will not do everything, please say what you think it *should* do). Do not skimp on this section: it provides an important foundation for the project, and will be marked as such.

between pieces of information that are specifics to a certain type of user: this will be managed on the application itself. However, in this report (in the *Database Views* section) I will show which kind of information will be available to which kind of user.

This database will be used for a public application (pro-users are also part of the general public) so it does not need to store very specific information that would only be useful for professionals, for example highly technical data such as standards related to cinematography.

I would like to propose a database which, in addition to giving the classical and necessary information on a video content (date, name, actors etc...) would give information on the ethics of the people involved in its production. For example, if one of the actors has been accused of sexual or physical assault or embezzlement. How far these accusations have gone: whether there were only rumors, or whether there was a trial or conviction.

More and more people are interested in this aspect of industrial and cultural production. We can see this with the development of organic and fair-trade labels on food products for example. In the world of culture, the numerous controversies that take place each time a work by Roman Polanski is released are good examples of this tendency. The director was convicted in 1977 by the American justice system for the rape of a 13-year-old girl and is still considered a fugitive by the USA. As a result, many people boycott his productions and protest when he receives awards.

More recently, the Weinstein affair has had an international impact. It has brought to light practices of intimidation, abuse of power and sexual assault in the film industry. As a result, some moviegoers have decided to boycott Weinstein productions and even more broadly Hollywood productions, as Hollywood is seen as a central player in the trivialization of these practices.

Of course, ethics encompasses many different parameters such as the inclusion of people from minorities, who are gender fluid, disabled etc... But in this work, I will not take these aspects into account, I will only focus on rumors, accusations, and convictions of crimes as it is sufficient to showcase the idea's potential.

2. Database Plan: A Schematic View

Figure 1: Database design (only primary keys are shown)

Commented [NDC2]: In this section offer a high-level view of the database and its design. State what you think the principal entities are, as well as their main attributes and the key relations that connect them. Provide an E-R diagram (entities and tables) that illustrates your plan. Motivate your design – state why this way and not another. The project will give you certain leeway to define your own information system, so it is crucial that your motivate whatever you include, and what you leave out.

Here are the main tables and their attributes' description:

Figure 2: Content table

Table which stores a visual content (movie or series instance for example).

- synopsis: content synopsis
- id_studio: studio's ID which produced the content
- id_content_category: content category's ID (if it is comedy, action, horror…)
- name_content: content's name
- year_content: released year

Figure 3: Peron table

Table which stores each person present in the database. A person can be an actor, a director, a writer or have several roles at the same time. This is the reason why it is not an *actor* table for example.

- birth_date: date of birth
- id_nationality: nationality's ID
- gender: gender (is 0 for a male, 1 for a woman)
- person_name: person's full name

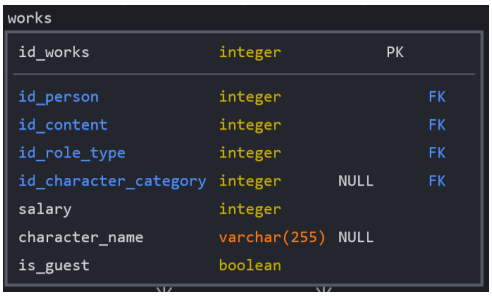

Figure 4: works table

Table which stores the relationship between the *content* and *person* tables. *works* table will captures all relevant information about someone's work on a piece of content. Since a person can have several roles on a same content, it has to be a separate table from the *person*'s table.

- id_person: ID of the person's instance about whom this work is about
- id_content: content's ID on which the person has worked
- id_role_type: role type's ID (actor, director, voice actor etc…)
- id_character_category: if it is an acting job, which type of character have been player? Spy, detective, villain etc…
- salary: salary perceived for this work
- character_name: character's name if there is one.
- is_guest: if this person worked as a guest on this content.

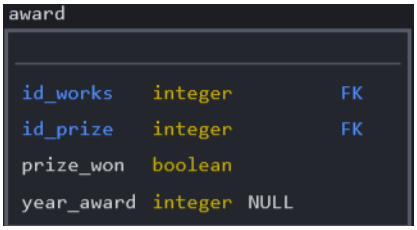

Figure 5: award table

Table which stores relevant information about an award. This table is a relationship table between *works* and *prize* table.

- id_works: work related to that award
- id_prize: prize's id
- prize_won: is 1 if the prize has been won, is 0 if it has just been a nomination.
- year_award: nomination year

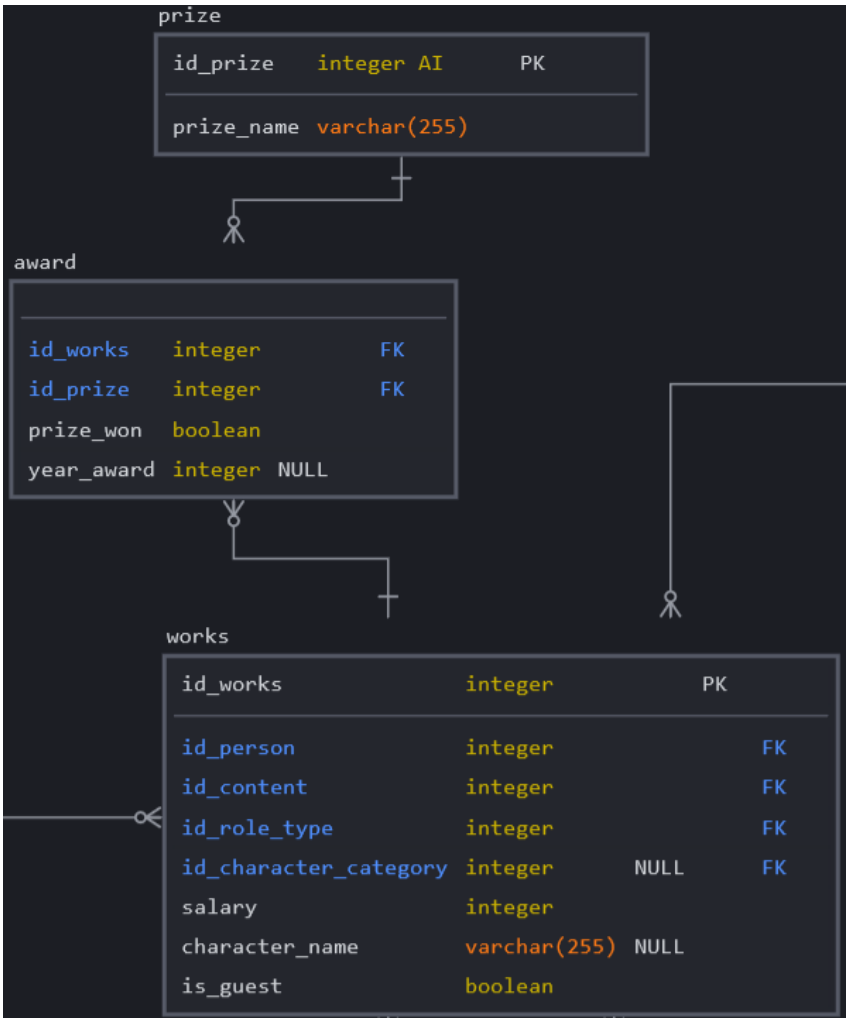

Figure 6: relationship between a work and a prize

An award is linked to a person's work and not directly to the person. Also, a person can receive several prizes and/or nominations for a same work as well as a prize can concerned different nominations. That is why a relationship table is required to link *prize* and *works* table and why this relationship table has references to the person and prize's ID.

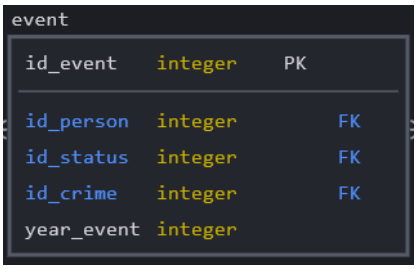

Figure 7: event table

The *event* table is a link between a person instance and an event. An even is a condemnation, a rumor, an acquittal… for a crime.

- id_person: person's ID linked to that event
- id_status: event status. If it is a rumor, if the person has been condemned, acquitted, if charges has been dropped etc…
- id_crime: type of crime's ID
- year_event: when the event occurred

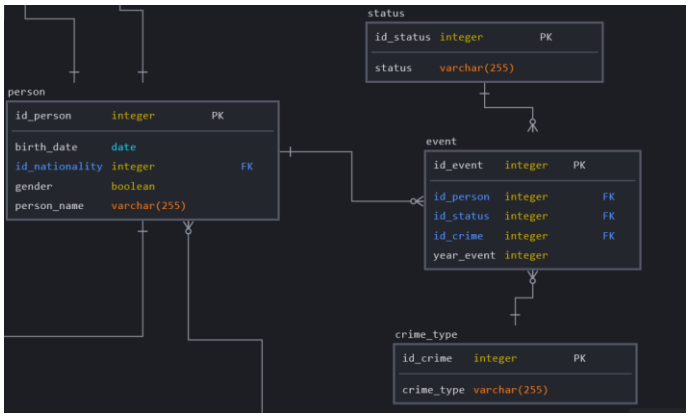

Figure 8: Modelling of the relationship between a crime and a person

A person instance can have several events associated with, so we need a relationship table between a person and a crime type. This relationship table has to store the type of crime, the "status" of the event and the person. At first, the primary key of this table was a concatenation of these three foreign keys. However, I realized that a same person can be linked the same way to a same type of crime. Therefore, this relationship table needs a dedicated primary key.

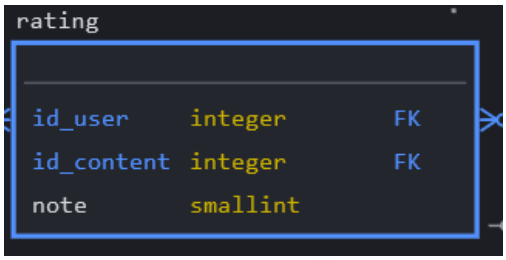

Figure 9: rating table

A content can be rated several times and a user can rates several contents. However, a user is only able to rate each content once. So, we need a relationship table between a content and a user.

The *rating* table is the table which describes the relationship between a content and a user. In this database design, this is the only link which exists between these two tables. A user will be able to rate a content and each note has to capture three pieces of information:

- id_user: user who rates
- id_content: content rated
- note: note (on 10) given by the user to the content

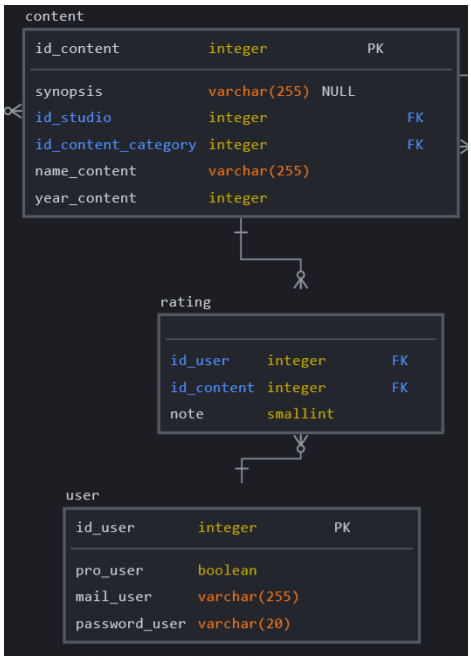

Figure 10: Modelling of the relationship between a content and a user

Another important piece of information to capture in the database is the relationship between persons. A relationship can be a love, family, friendship, or professional relationship between two persons. So, we need a relationship table between two persons instances. That is why there are two foreign keys from the *person* table. Also, information is needed on the nature of the relationship, that is what relationship_type_id does by referencing the ID of the relationship table in the relationship_type table. Lastly, we would like to know when the relationship has started and, if it is not an actual relationship, when it ended.

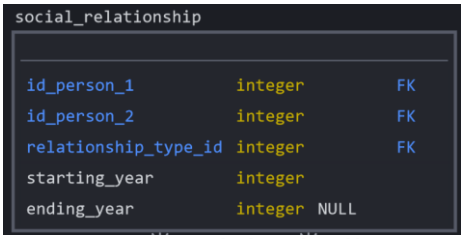

Figure 11: social_relationship table

- id_person1: First reference to the *person* table
- id_person2: Second reference to the *person* table
- relationship_type_id: Reference to the *relationship_type* table
- starting_year: relationship starting year
- ending_year: relationship ending year (can be NULL if the relationship has not yet ended or cannot end)

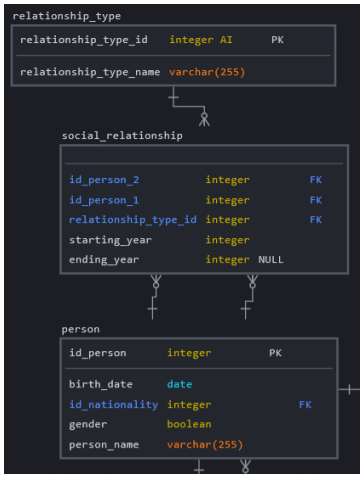

Figure 122: Relationship modelling between two persons and the status of their relationship

3. Database Structure: A Normalized View

Commented [NDC3]: Describe the main tables in your database and the role played by each. Show that your database meets the definitions of 1NF, 2NF and 3NF normal forms. Is your database in BCNF normal form? If so, explain how and why. If not, justify this.

1NF

For a database to be in first normal form, it has to follow these four rules:

- Each column has to store only one value.
- All the values in a column have to be of the same type.
- Each column's name has to be unique.
- Every column has to relate to the key.

Let's look at our main tables to check how they respect these rules.

"Each column has to store only one value" rule

The database has obviously a "content" table which registers the main information about a content. Its name and its released year for example. Another interesting piece of information about content is the identity and role of the people working on it. However, this is not a relevant information to store in a column. If I had to store this information in a column named "actors" for example, I would have to put every actor's name in it and separate them with commas. But it would violate one of the 1NF rule since only one value has to be stored in a column. To solve this problem, I have a "person" and a "works" tables that link the content with every single person that worked on it. Each instance of the person table stores information about only one person, and each works' table instance stores information about the nature of the functions that the person had on this content.

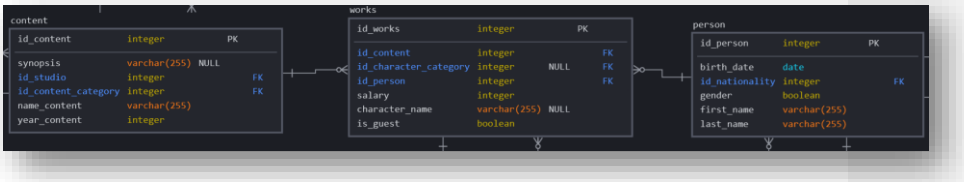

Figure 13: content, works and person table relationship

"All the values in a column have to be of the same type" rule

Let's take the content table as an example again. The two most important information about a content is its name and its released year since it is thanks to these two properties that we manage to uniquely identify a content. But these two pieces of information have a different type since one if a number and the other one is one or several words. So, we cannot store them in the same column. That is why there is one column of type *varchar* to store the name and one of the type *integer* to store its released year.

"Each column's name has to be unique." Rule

Some columns have the same meaning within a table. It is the case for the *year* property which appears in several different entities. To differentiate them, I simply add the table column after the property name such as follow:

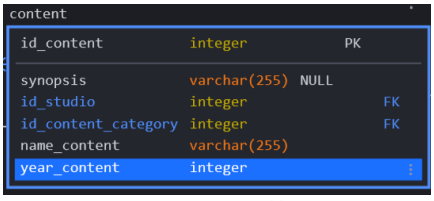

Figure 14: content table

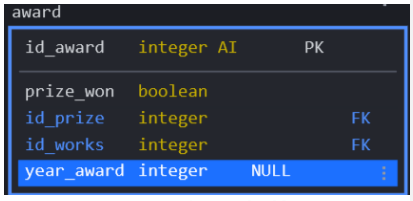

Figure 135: award table

"Every column has to relate to the key" rule

id_content	synopsis	id_studio	id_content_category	name_content	year
0	A Polish Jewish musician struggles to survive the ...	0	1	The pianist	2002

Figure 14: instance of the "content" table

As we can see in this instance of the *content* table there is a reference to the IDs of the studio and the content's category. I could have directly stored these two pieces of information as *varchar* but then these columns would not have been connected to a key since several instances of the "content" table can share a same value for these properties. Instances sharing the same value for this

property would not have been connected to each other in the database. This is the case if we reference the IDs of the instances of these values.

2NF

For a database to be in second normal form, it should not contain any partial dependency. Partial dependency happens when a non-prime attribute only depends on part of a prime key and not on the whole prime key. To remove Partial dependency, we can divide a table, remove the attribute, which is causing partial dependency, and move it to some other table where it fits in well.

Figure 17: relationship between a user and a content

The relationship between these tables gives an example on how I manage the second normal form. If I had recorded the note as a property (and not as a foreign key) of the *user* table, it would not have relevant since it is not a sufficient information to know what the rating is. Indeed, a rating is dependent on the user ID AND on the content ID. So, we need to store these two pieces of information in a dedicated table.

3NF

For a table to be in third normal form it needs to be in second normal form, and it should not have transitive dependency. Transitive dependency happens when an attribute depends on some non-prime attribute and not on the prime attribute. Let's take again the example above. When I added the *note* property, I could have added it to the *content* table. But *note* does not dependent on the content ID solely. It also depends on the user ID. If I had done that, we would have an instance of each content for each note given. Since *note* property is an attribute in the *rating* table, each content has only one instance in the *content*

table and several ratings can be associated to each of these content instances.

BCNF

For a table to be in Boyce-Codd Normal Form (BCNF) it must be in the third normal form and for any dependency *A->B*, *A* should be a super key. This database is in BCNF since no non-prime attribute.

4. Database Views

For the sake of visual clarity, the next two pages only show the four views I will then talk about.

Commented [NDC4]: What views do you provide onto your database? Justify each and define them here. Explain what each view is supposed to provide, and to whom it provides it. Is it wise or necessary to represent a certain relation as an SQL view? Be sure to specify your views (about 4 would be sufficient) and give examples of their use.

name_content	synopsis	studio_name	year_content	content_category_name	Unethic score
Shakespeare in Love	The world's greatest ever playwright, William Shakespeare, is young, out of ideas and short of cash, but meets his ideal woman and is inspired to write one of his most famous plays.	The Bedford Falls Company	1998	historical drama	3
Gang of New York	In 1862, Amsterdam Vallon returns to the Five Points area of New York City seeking revenge against Bill the Butcher, his father's killer.	Miramax Films	2002	historical drama	3
House of Cards	A Congressman works with his equally conniving wife to exact revenge on the people who betrayed him.	Netflix	2013	politic	3
Cannibal Holocaust	During a rescue mission into the Amazon rainforest, a professor stumbles across lost film shot by a missing documentary crew.	F.D. Cinematografica	1980	horror	2
Hostel: Part II	Three American college students studying abroad are lured to a Slovakian hostel and discover the grim reality behind it.	Lionsgate	2007	horror	2
Beauty and the Beast	A prince cursed to spend his days as a hideous monster sets out to regain his humanity by earning a young woman's love.	Buena Vista Pictures Distribution	1991	musical	1
Demolition Man	A police officer is brought out of suspended animation in prison to pursue an old ultra-violent nemesis who is loose in a non-violent future society.	Warner Bros.	1993	science fiction	1
Mr. & Mrs. Smith	A bored married couple is surprised to learn that they are both assassins hired by competing agencies to kill each other.	New Regency Pictures	2005	action	1
The Pianist	A Polish Jewish musician struggles to survive the destruction of the Warsaw ghetto of World War II.	Canal+	2002	drama	1
Cleopatra	Queen Cleopatra VII of Egypt experiences both triumph and tragedy as she attempts to resist the imperial ambitions of Rome.	20th Century Fox	1963	historical drama	0

Figure 15: Unethic score view

5. Procedural Elements

6. Example Queries: Your Database In Action

7. Conclusions

Acknowledgements

References

https://www.pastemagazine.com/movies/10-mob-movie-actors-with-actual-organized-crime-ti/

https://www.rosaleslawfirm.com/blog/2019/08/famous-white-collar-crime-cases-from-gangsters-to-actors-and-television-stars/

https://www.independent.co.uk/arts-entertainment/films/news/mark-wahlberg-racist-hate-crimes-wikipedia-history-george-floyd-blm-protests-a9554191.html

https://en.wikipedia.org/wiki/Sexual_abuse_in_the_American_film_industry

https://www.cbr.com/cannibal-holocaust-director-was-charged-with-murder/

https://www.glamour.com/gallery/post-weinstein-these-are-the-powerful-men-facing-sexual-harassment-allegations

Commented [NDC5]: Does your design employ procedural extras such as database triggers (in PL/SQL or the MySQL equivalent format)? If so, describe and motivate each. If your design does not contain procedural extras, explain why, and say how you were able to do without these additions. Most projects have some scope for procedural elements (about 4 would be sufficient here).

Commented [NDC6]: Your database will provide a structure for the data in the application, and means of accessing and viewing that data. In this section show us the database in action, by providing sample queries and their outputs (please do not provide large data sets as outputs; summarize as appropriate). Provide specific queries to test on your database, and tell us what those queries provide to the application. Use your existing database as the basis for your queries. If a query makes reference to additional tables then provide example rows of this table in section 3.
You may use screenshots here but do not overfill your report with screenshots. Ensure that there is a cohesive argument expressed in the text of the report and that it is not simply a bag of diagrams and queries and screenshots. When you include images, make sure they are readable and actually add to the discussion.

Commented [NDC7]: Provide any concluding thoughts here. How might you build on this work for the future? How might your database support future developments?

Commented [NDC8]: Name check any person who helped you with this work. Acknowledge that the work is entirely your own, and that every sentence in this report was written by you and you alone. If you wish to quote another person or piece of work, place the quoted work in quotation marks and cite the author inline. Plagiarism is a very serious infraction that must be dealt with severely. Avoid any ambiguity on this point by citing things carefully!

Commented [NDC9]: List any bibliographical citations here [for people and work that you quote/cite in the main text of your report]. Follow professional citation norms, and do not use partial references. The most impressive citations are to academic papers and books, not to websites and Wikipedia articles.

Commented [NDC10]: Wesley Snipes is another famous actor with tax evasion attached to his name. It was said that Snipes avoided paying $7 million in taxes between 1999 and 2001. He claimed that he was only listening to advice from hired professionals on the matter. It did not go well for him but he fought it in court.