

**UNIVERSIDAD TÉCNICA PARTICULAR DE LOJA**

*La Universidad Católica de Loja*

[**Informe**](http://dspace.utpl.edu.ec/handle/20.500.11962/23888) **Proyecto Final**

**Fundamentos de Bases de Datos**

**Autor**: Isaac Matias Quezada Alvarez

Octubre 2022 – Febrero 2023

[Introducción 3](#_Toc126830817)

[Modelos 4](#_Toc126830818)

[Modelo Conceptual 4](#_Toc126830819)

[Modelo Lógico 6](#_Toc126830820)

[Dependencias Funcionales -Tabla Universal- 7](#_Toc126830821)

[Modelo Físico 9](#_Toc126830822)

[Normalización 10](#_Toc126830823)

[First Normal Form (1NF) 10](#_Toc126830824)

[Second Normal Form (2NF) 11](#_Toc126830825)

[Third Normal Form (3NF) 11](#_Toc126830826)

[Pasos Para Realizar el Proyecto 17](#_Toc126830827)

[Limpieza De Crew **¡Error! Marcador no definido.**](#_Toc126830828)

[Consultas 25](#_Toc126830829)

[Conclusiones 31](#_Toc126830830)

# Introducción

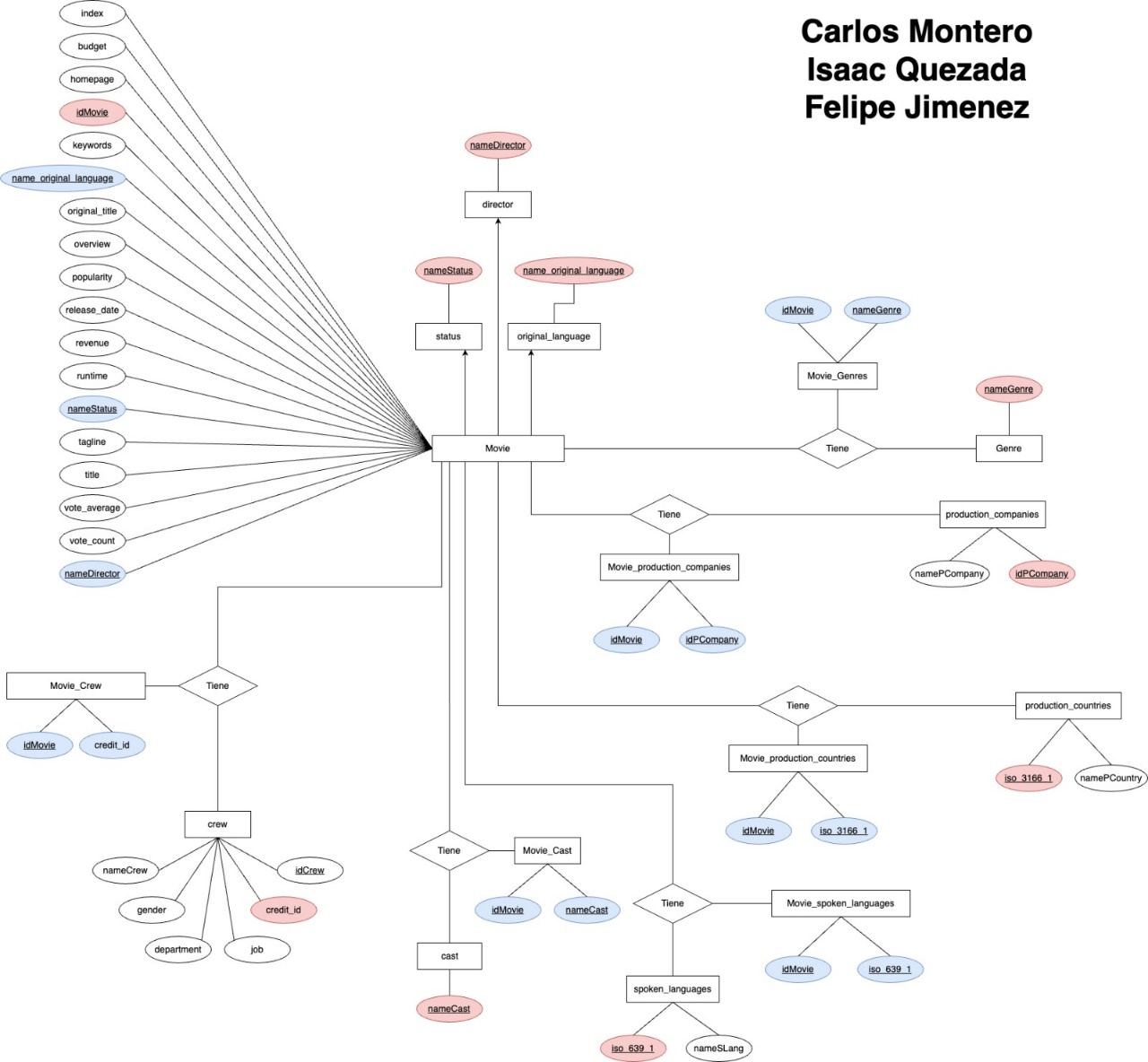
Las bases de datos han demostrado a lo largo del tiempo que son esenciales en muchos sistemas informáticos ya que nos da la opción de ingresar, manipular, almacenar y/o eliminar grandes cantidades de información, este presente informe tiene como objetivo aplicar todos los conocimientos obtenidos en todo este tiempo de estudio con base a la materia de Base de Datos, aplicando la importación del CSV desde una base de datos usando el sistema de gestión de base de datos MySQL, además de pasar por distintas fases, las cuales incluyen la inserción de datos, la limpieza de dichos datos, la carga y sobre todo la explotación de datos. Igualmente, podemos diferenciar los métodos empleados al realizar todas las fases propuestas en este trabajo.  
 Este informe busca producir resultados orientados a la gestión y uso eficiente de la información relevante y de interés, a partir del conjunto de datos de trabajo. Además, nos permitió recordar varios temas estudiados en la materia tales como diagramas de Entidad/Relación, modelos conceptuales, lógicos y físicos, la creación de instructivos SQL, entre muchos más temas que se muestran a lo largo del informe.

# Modelos

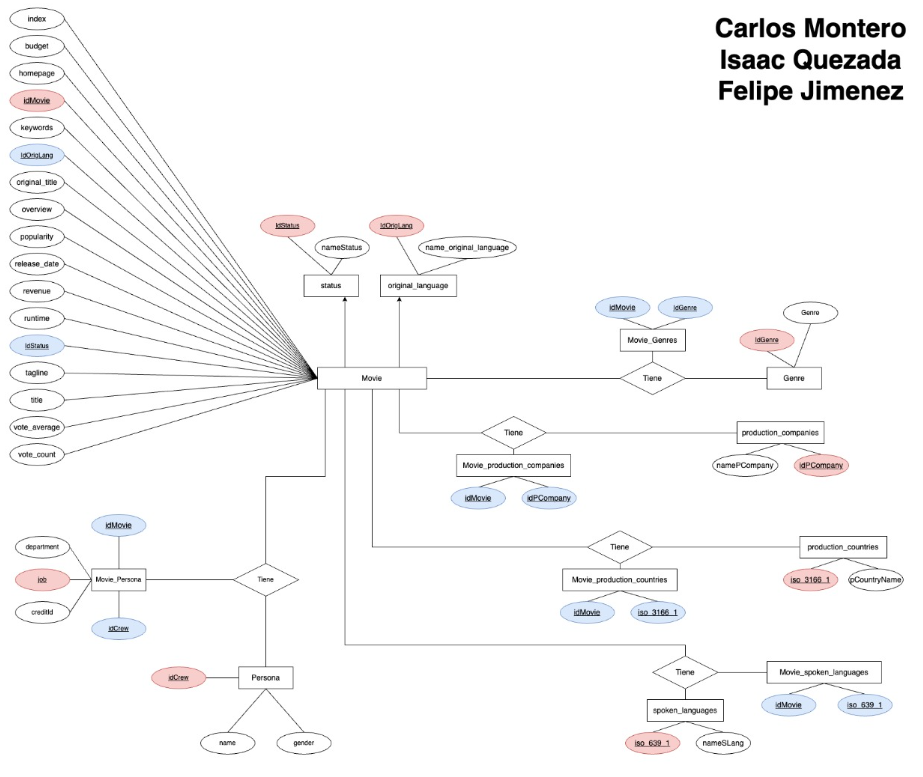
## Modelo Conceptual

Después de haber establecido los datos mediante la tabla universal, normalizamos los mismos utilizando la metodología Entidad-Relación, identificando los atributos correspondientes a cada relación. Para ello, utilizamos la herramienta draw.io, lo que nos permitió crear diagramas de flujo efectivamente.

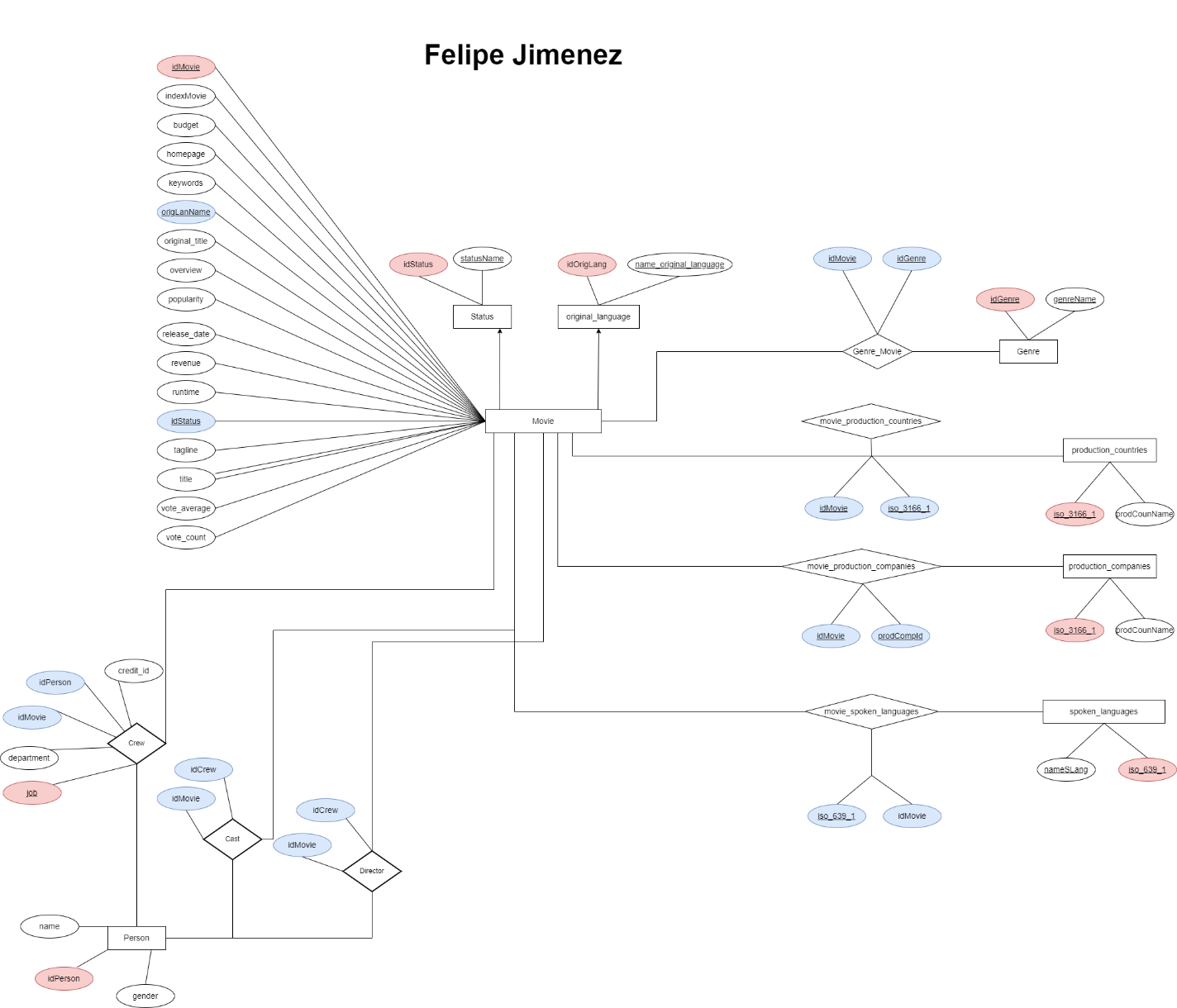
**VERSION 1**



**VERSION 2**



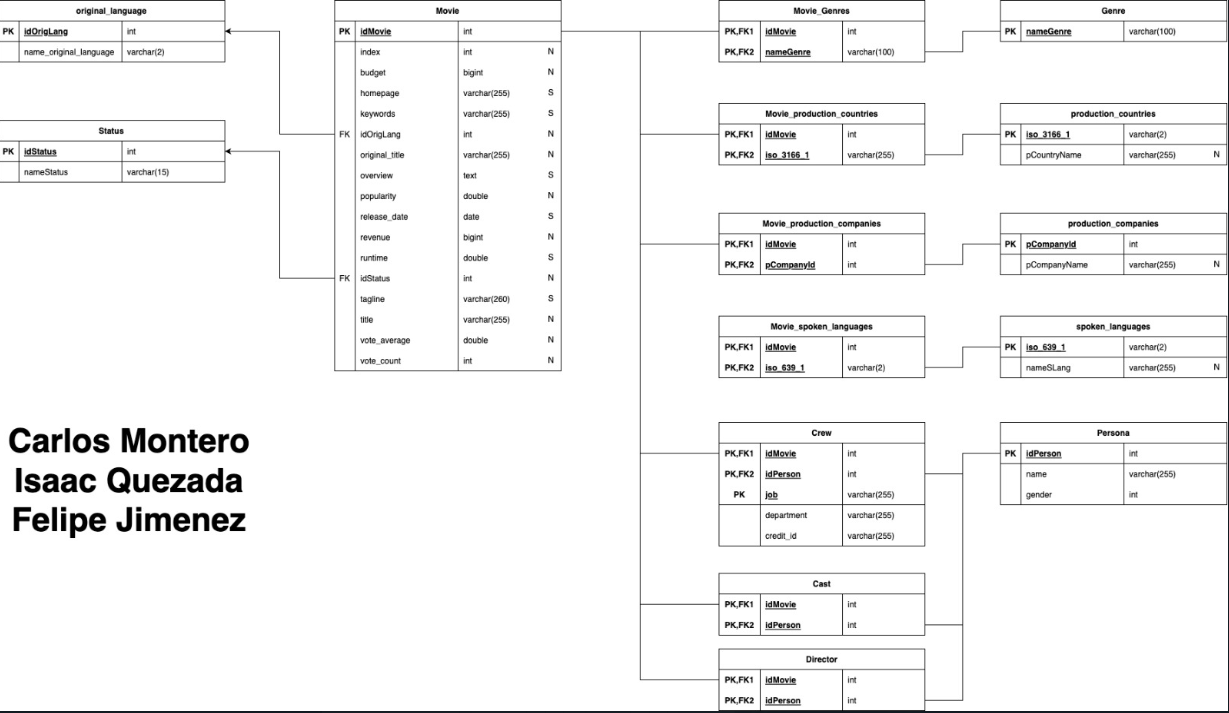
**VERSION FINAL**

****

## Modelo Lógico

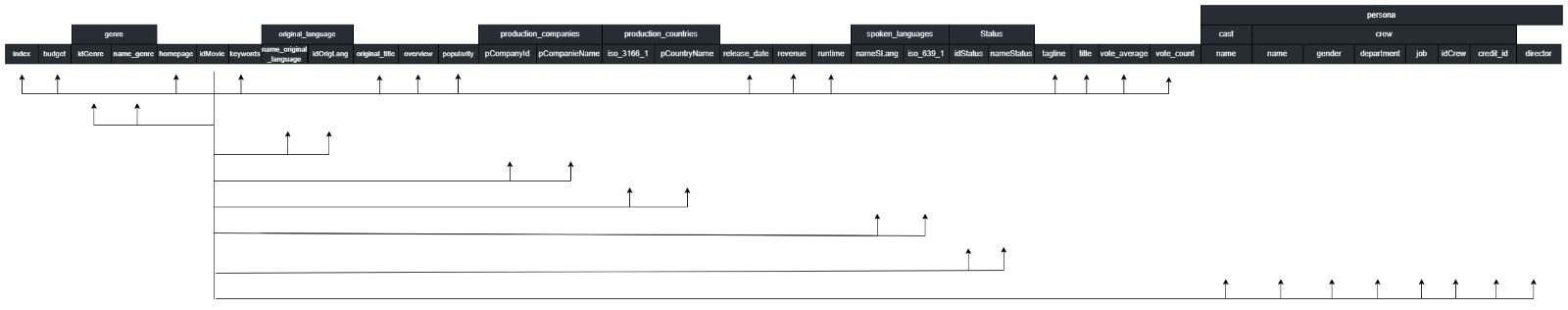
Con el modelo conceptual ya realizado, se seleccionan los atributos apropiados para cada tabla y se crea un prototipo de estas basado en las entidades y relaciones identificadas. En cada tabla, se declara una clave primaria y, en su caso, una clave foránea, esto con el objetivo de tener una base sólida para desarrollar el esquema de la base de datos.

*Figura 2: Representación gráfica del Modelo Lógico*

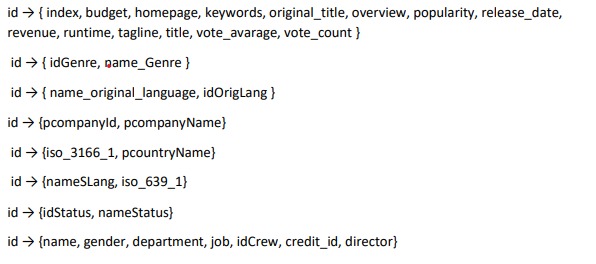


## Dependencias Funcionales -Tabla Universal-

Una vez establecido el modelo lógico y conceptual se creó las dependencias funcionales existentes en la tabla universal.



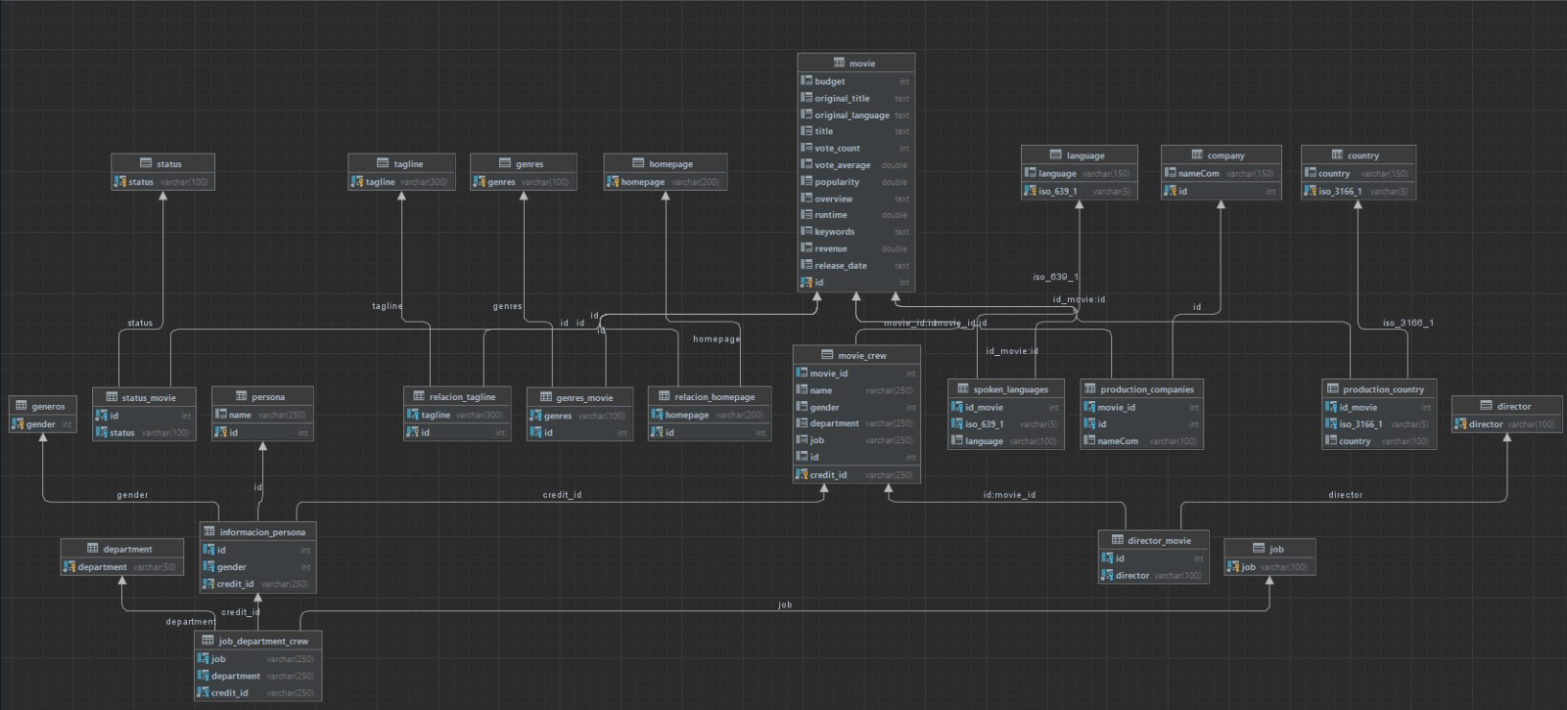
*Figura 3: La imagen muestra una descripción gráfica de las dependencias existentes en las columnas del dataset*



*Figura 4: Dependencias Funcionales textualizadas para un mejor entendimiento*

## Modelo Físico

Una vez terminado el modelo lógico pasamos a la fase final que es la creación del modelo físico donde se especificó las tablas, columnas, claves principales y a su vez sus claves externas o foráneas con sus respectivas relaciones. Con este modelo podemos pasar a generar las respectivas sentencias DDL



*Figura 5: Representación gráfica del Modelo Físico*

# Normalización

Antes de entrar a este tema debemos saber su concepto y lo que conlleva aplicarlo en este proyecto, este proceso se realiza en la estructura de las bases de datos para poder así mejorar en temas de eficiencia, organización e integridad, con el objetivo de eliminar los tipos de redundancia y dependencia funcional. Otro concepto mayor mente conocido por los usuarios es la división de información en diferentes tablas y hacemos relaciones entre sí.

Seguimos los siguientes pasos para normalizar los datos del archivo CSV:

## First Normal Form (1NF)

* Una relación en donde la intersección de cada fila y columna contiene un y solo

un valor.

* La fase antes de 1NF es Unnormalized Form (UNF) la cual es una tabla que

contiene uno más grupos repetidos.

* Para transformar la tabla no normalizada a la primera forma normal, identificamos

y eliminamos los grupos que se repiten dentro de la tabla.

* + Un grupo repetitivo es un atributo, o grupo de atributos, dentro de una tabla que ocurre con múltiples valores para una sola ocurrencia de los atributos clave designados para esa tabla.

## Second Normal Form (2NF)

* + Una relación que está en la primera forma normal y cada atributo que no es de clave principal depende funcionalmente de la clave principal.
  + La normalización de las relaciones 1NF a 2NF implica la eliminación de dependencias parciales. Si existe una dependencia parcial, eliminamos los atributos parcialmente dependientes de la relación colocándolos en una nueva relación junto con una copia de su determinante.

## Third Normal Form (3NF)

* + Una relación que está en primera y segunda forma normal y en la que ningún atributo que no sea de clave principal depende transitivamente de la clave principal.
  + La normalización de las relaciones 2NF a 3NF implica la eliminación de las dependencias transitivas.
  + Si existe una dependencia transitiva, eliminamos los atributos transitivamente dependientes de la relación colocando los atributos en una nueva relación junto con una copia del determinante.

Primero identificamos y eliminamos grupos repetitivos dentro de la tabla. Un grupo repetitivo es un atributo, o grupo de atributos, dentro de una tabla que ocurre con múltiples valores para una sola ocurrencia de los atributos clave designados para esa tabla.

Tabla Uno a Muchos (1 : N)

Status es una columna con tres posibles valores (released, rumored, post-production)

* + Una película puede tener un Status, pero un Status puede representar a muchas películas.
* Para solucionar, se crea una tabla separada que se llame Status.
* La llave primaria es statusName.

Como solución, utilizamos el statusName como llave foránea en Movie.Director es una columna con el nombre de un único director.

* Una película puede tener un Director, pero un Director puede tener

muchas películas.

* Para solucionar, se crea una tabla separada que se llame Director.
* La llave primaria es directorName.
* Como solución, utilizamos el directorName como llave foránea en Movie.

Original\_language es una columna con el nombre del lenguaje original de la Movie.

* + Una película tiene un Original\_language, pero un Original\_language puede tener muchas películas.
* Para solucionar, se crea una tabla separada que se llame original\_language.
  + La llave primaria es origLanName.
  + Como solución, utilizamos el origLanName como llave foránea en original\_language.

Tabla Muchos a Muchos (N : N)

En nuestro caso, ninguna de las 4803 entradas se repite, cada película es diferente (única). Cada Movie tiene un index y un id diferente. Como llave primaria utilizamos id el cual nombramos idMovie para no confundirlo con otros id que existan en otras columnas.

* Genres contiene un String de géneros que puede contener 0 a n géneros.
* Para solucionar, cada columna debe contener un solo valor.
* En este caso tenemos una lista de géneros. Existen múltiples valores.
* La solución es crear una tabla separada que se llame Genres.
* La llave primaria es genreName
  + Una Movie puede tener muchos géneros, y un género puede estar en muchas Movie.
  + La relación (muchos a muchos) se soluciona con una tabla llamada Movie\_Genres utilizando la llave primaria de cada uno.
* Keywords contiene un String de keywords que puede contener (0 a N) keywords.
* Para solucionar, cada columna debe contener un solo valor.
* En este caso tenemos una lista de keywords. Existen múltiples valores.
* La solución es crear una tabla separada que se llame Keywords.
* La llave primaria es keywordName
  + Una Movie puede tener muchos keywords, y un keyword puede aparecer en muchas Movie.
  + La relación (muchos a muchos) se soluciona con una tabla llamada Movie\_Keywords utilizando la llave primaria de cada uno.
* Production\_companies contiene un String de JSON que contiene un name y un id que puede contener 0 a n production\_companies.
* Para solucionar, cada columna debe contener un solo valor.
  + - En este caso tenemos una lista de production\_companies. Existe múltiples valores.
    - La solución es crear una tabla separada que se llame Production\_companies.
    - Tiene dos atributos, name e id los cuales los nombramos prodCompName y prodCompId el cual es la primary key.
    - Una Movie puede tener muchos production\_companies, y un production\_companies puede aparecer en muchas Movie.
    - La relación (muchos a muchos) se soluciona con una tabla llamada Movie\_ Production\_companies utilizando la llave primaria de cada uno.
  + Production\_countries contiene un String de JSON que contiene un iso\_3166\_1 y un name que puede contener 0 a n production\_countries.
* Para solucionar, cada columna debe contener un solo valor.
  + - En este caso tenemos una lista de production\_countries. Existen múltiples valores.
* La solución es crear una tabla separada que se llame Production\_countries.
  + - Tiene dos atributos, iso\_3166\_1 y name los cuales los nombramos, iso\_3166\_1 el cual es la primary key y prodCompName.
    - Una Movie puede tener muchos production\_countries, y un production\_countries puede aparecer en muchas Movie.
    - La relación (muchos a muchos) se soluciona con una tabla llamada Movie\_ Production\_countries utilizando la llave primaria de cada uno.
* spoken\_languages contiene un String de JSON que contiene un iso\_639\_1 y un name que puede contener 0 a n spoken\_languages.
* Para solucionar, cada columna debe contener un solo valor.
* En este caso tenemos una lista de spoken\_languages. Existen múltiples valores.
* La solución es crear una tabla separada que se llame spoken\_languages.
* Tiene dos atributos, iso\_639\_1 y name los cuales los nombramos, iso\_639\_1 el cual es la primary key y spokLangName.
* Una Movie puede tener muchos spoken\_languages, y un spoken\_languages puede aparecer en muchas Movie.
* La relación (muchos a muchos) se soluciona con una tabla llamada Movie\_ Spoken\_languages utilizando la llave primaria de cada uno.
* ***‘Cast’, ‘Crew’*** y ***‘Director’*** contienen los nombres de integrantes que participan de alguna forma en la película. Estos necesitan ser normalizados en una tabla con datos comunes, en este caso llamada ***Persona.*** 
  + Se crea una tabla Persona que incluya los atributos comunes a todas las entidades, entre los cuales están: gender, idCrew y name.
  + Ya que la relacion Persona-Movie es muchos a muchos, por consecuencia se necesita otra tabla que contenga el idMovie e idCrew.
  + Es momento de crear tablas de ***‘Cast’, ‘Crew’*** y ***‘Director’,*** para la migracion de datos hacia la tabla persona.
  + Antes de migrar los datos a la tabla Persona, es importante realizar una limpieza de datos para garantizar que los datos sean precisos y coherentes.

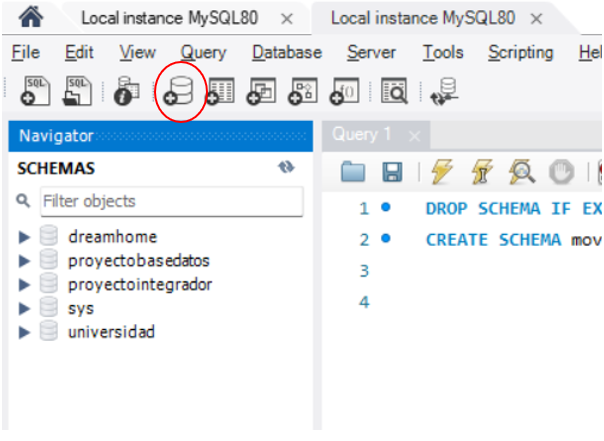
# Pasos Para Realizar el Proyecto

1. Creación de un “Schema”

Para la creación del “Schema” se podía realizar de dos distintas maneras, la creación automática o por sentencias SQL.

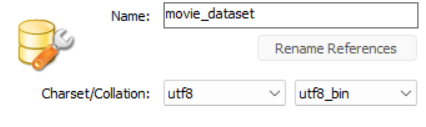
**OPCIÓN 1 – MEDIANTE LA CREACIÓN AUTOMÁTICA**

Para la creación automática del Schema dentro del lenguaje de Base de Datos MySql, primero se debe entrar se debe ubicar en la parte superior izquierda, donde se encuentra los íconos, señalar la que es para crear un Schema como se puede ver en el siguiente Anexo:



*Figura 6: Representación de la barra de opciones perteneciente a Workbench*

Subsecuentemente se le daría nombre al Schema en este caso “movie\_dataset”, donde se podría modificar varios aspectos, en nuestro Schema utilizamos un Charset (codificación de datos) de “utf8”:

*Figura 7: Representación de la funcionalidad de creación de Schema automáticamente*

**OPCIÓN 2 – MEDIANTE SENTENCIAS SQL**

Para la creación del “Schema” o base de datos usando sentencias SQL, es necesario hacer lo siguiente: 

Es necesario especificarle la codificación de caracteres utf8 ya que en el archivo CSV se usa caracteres especiales y esta misma codificación los transforma.

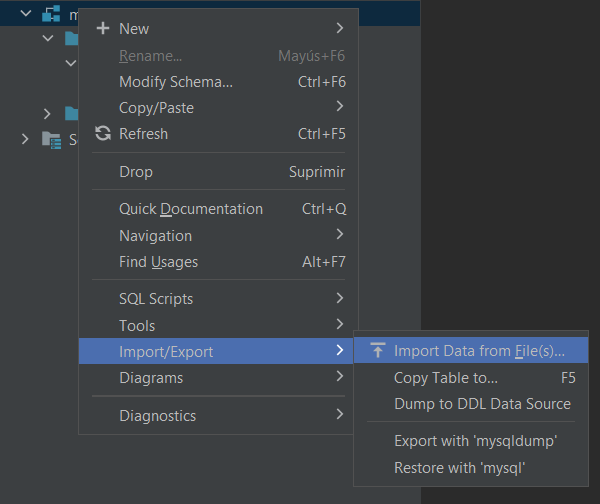
1. Conexión del “Schema” a la Base De Datos.

Para poderse conectar, por así decirlo, es necesario hacerlo mediante sentencias SQL, la sentencia es la siguiente:



1. Importación del CSV

DataGrip es un IDE de JetBrains para el manejo de base de datos. Dentro de este existe una herramienta facilitadora de importación de varios tipos de archivos a tablas MySQL. A continuación, se muestran los pasos seguidos en la siguiente imagen:



*Figura 8: Representación de la funcionalidad de importación de archivos en DataGrip*

Una vez entrado a este apartado, se habilita una interfaz de importación en donde especificamos que el separador es la coma, y que la primera fila son los títulos de las columnas

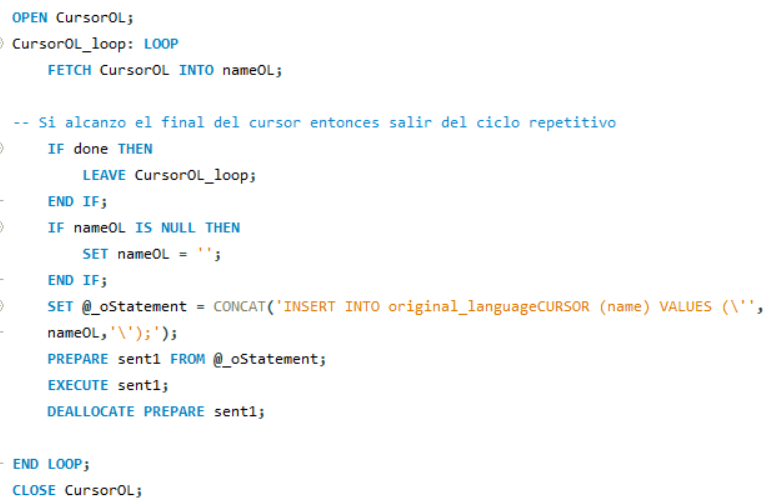
Captura de pantalla de computadora

Descripción generada automáticamente

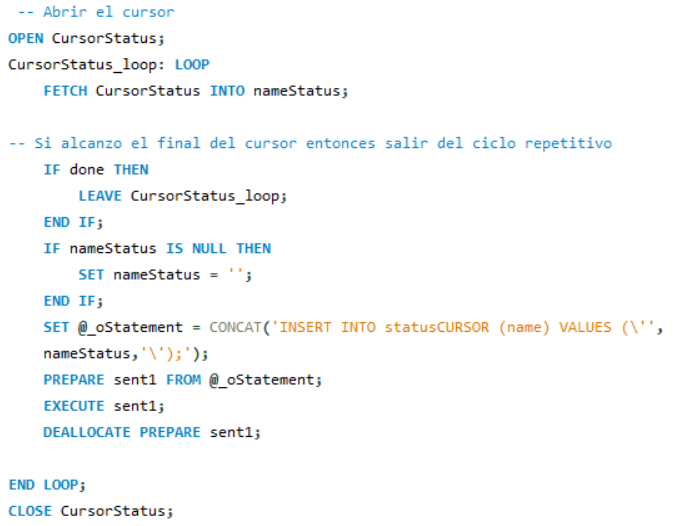
*Figura 8: Representación de la interfaz de importación de archivos en DataGrip*

1. Creación de funciones que Permitan Extraer y Limpiar los Datos.

**Original\_language.** Para la integridad de las tablas, original language tuvo que ser creada primero junto a status, ya que estas no dependen de ninguna clave foránea y en cambio movies dependen de estas. Se hizo el siguiente procedimiento



**Status.** Este sigue la lógica de original\_language para la creación de la tabla. Se hizo un procedimiento para este.



**Cast.** Dentro de cast, no pudimos encontrar un patrón claro en el que podamos separar los nombres de manera coherente. Es por esto y por falta de tiempo que se decidió

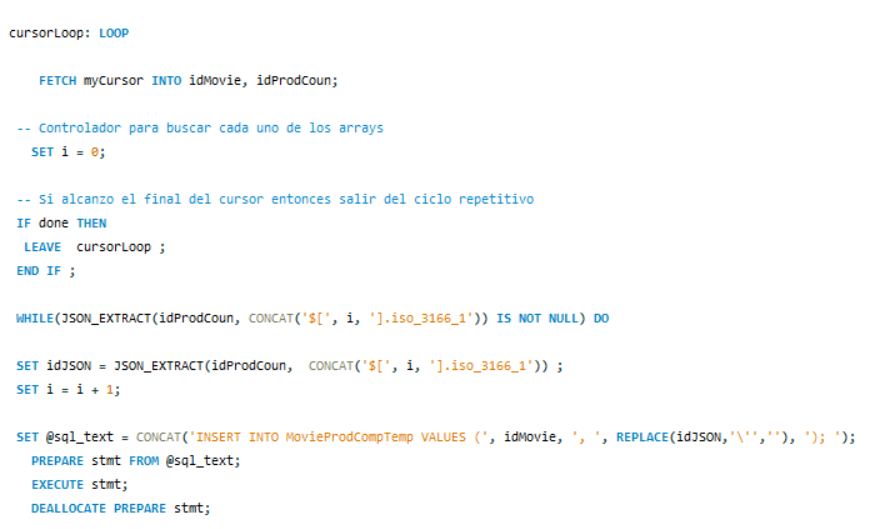
**Movie.** Una vez creadas las tablas de relaciones uno a muchos, es momento de poblar la tabla principal, la cual es movie



**Production\_companies.** Al ser este un JSON, es necesario usar la función JSON\_EXTRACT para separar los objetos del JSON y pasarlos a la tabla



**Production\_countries.**



**Crew.** Crew tiene una particularidad, y es que tendría que ser un JSON, sin embargo tiene un error de formato que necesita ser tratado mediante el siguiente código:

SELECT JSON\_EXTRACT(CONVERT(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE

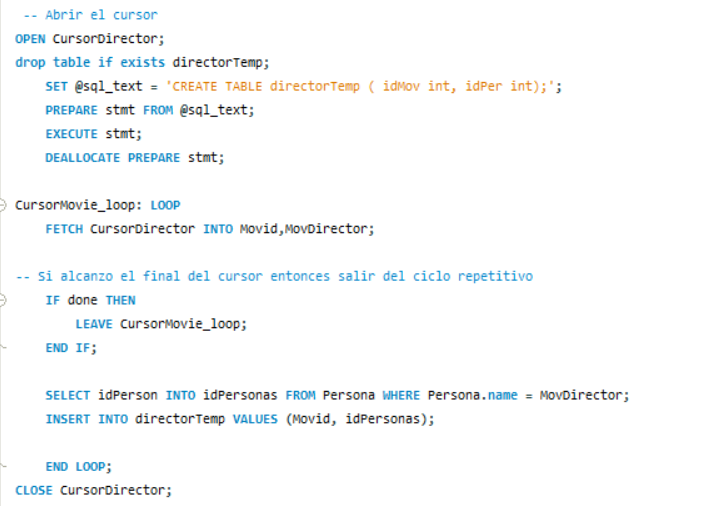
(REPLACE(crew, '"', '\''), '{\'', '{"'),

'\': \'', '": "'),'\', \'', '", "'),'\': ', '": '),', \'', ', "')

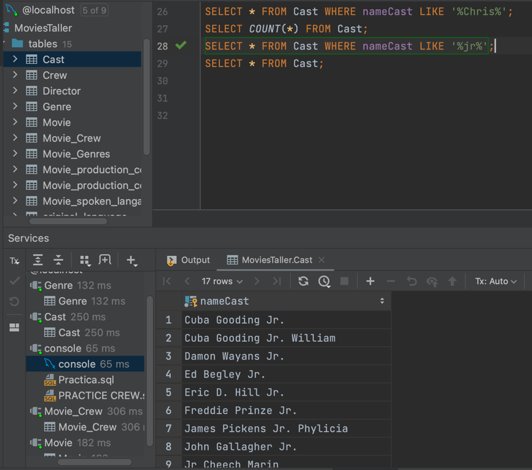
USING UTF8mb4 ), '$[\*]') FROM movie\_dataset

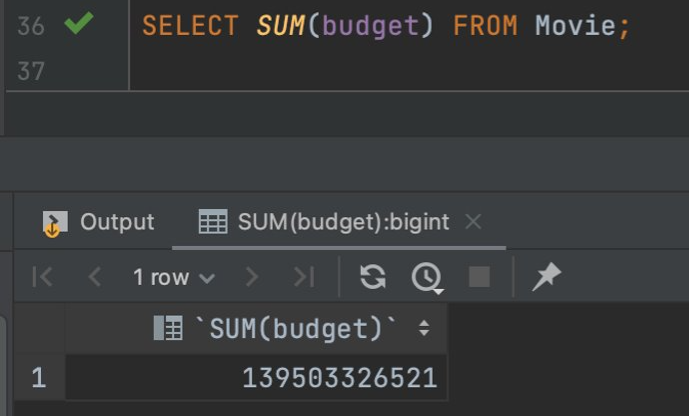


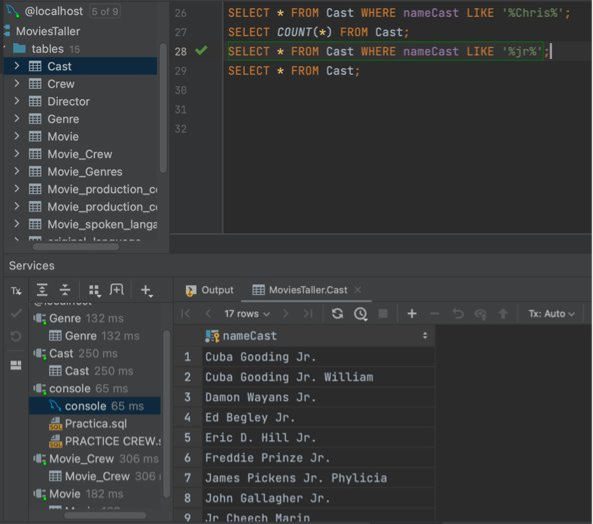
Director.

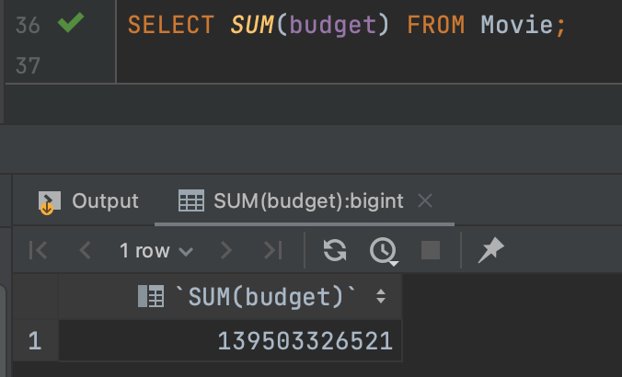


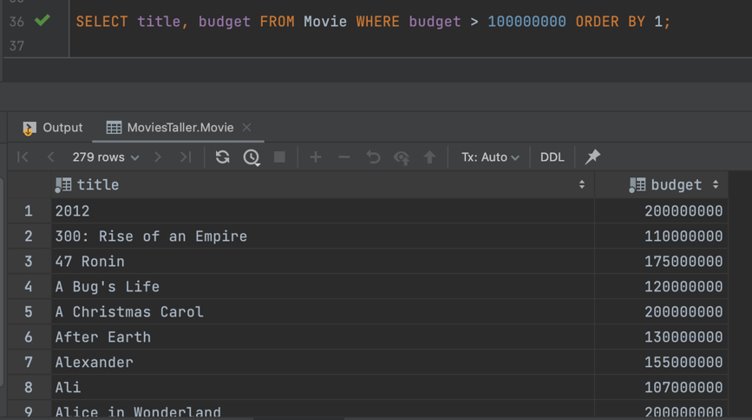
# Consultas











# Conclusiones

Este proyecto fue realizado bajo unas normas de modelado de base de datos, como normalización, diseño conceptual, lógico, etc. Y luego de una ardua tarea por parte de todos los integrantes del grupo, hemos podido cumplir con todos los objetivos planteados al inicio del desarrollo. Cabe resaltar que hemos tenido que afrontar varios desafíos a medida que íbamos progresando en el proyecto, sin embargo, con las pautas de nuestros docentes guías y nuestra creatividad de resolución de problemas, pudimos satisfacer los requerimientos dados.

Una vez realizado todos los pasos correspondientes para culminar nuestro proyecto, hemos podido evidenciar un gran progreso en nuestra toma de decisiones referentes al modelado y población de una base de datos, además de aprender nuevas estrategias de programación aplicables a casos de la vida real. El desarrollo de este proyecto sin duda nos acerca un paso más a ser unos profesionales seguros a la hora de trabajar en proyectos futuros.

Aprendimos que datos, antes de sus respectivos tratamientos, deben de hacer de manera prolija para evitar consistencia y errores. Además, entendemos la importancia de las herramientas al momento de interactuar con el base de datos. Y se puede culminar que todo lo visto este ciclo fue muy importante ya que eso fue lo utilizado en este proyecto integrador cabe recalcar que este proyecto integrador fue un reto para nosotros como estudiantes ya que es nuestra primera vez trabajando con tantos datos.