



UNIVERSIDAD TÉCNICA PARTICULAR DE LOJA

La Universidad Católica de Loja

Entrega Preliminar del Proyecto Final De Fundamentos De Base De Datos

Autor:

Piedra Narváez José David

Octubre 2022 – febrero 2023

1. Introducción.....	3
2. Desarrollo del Componente.....	3
2.1. Análisis de los datos que contiene el movie dataset.	3
3. Diseño y Modelado de la Base de datos	5
3.1. Dependencias Funcionales.....	5
3.2. Diseño Conceptual	6
3.3. Diseño Físico	7
3.4. Diseño Lógico	8
4. Desarrollo.....	9
4.1. Eliminación de valores nulos.....	9
4.2. Columna con valores redundantes	10
4.3. Columnas con valores no atómicos.....	12
4.4. Columnas con datos en formato JSON	15
5. Consultas SQL	20
6. Conclusiones	23
7. Aprendizaje.....	26
8. Bibliografía	27

1. Introducción

El proyecto presentado por la Ingeniera Audrey Romero en conjunto con los docentes de las materias de programación funcional y reactiva, y base de datos trató acerca de analizar un archivo CSV, este archivo poseía varios errores, por ejemplo columnas no atómicas, columnas en formato JSON, o columnas con valores nulos, nosotros los estudiantes tuvimos que aplicar nuestros conocimientos para poder realizar acciones como limpieza, transformación y modelado de los datos que poseíamos para poder tener una base de datos ideal para la lectura y su posterior análisis.

2. Desarrollo del Componente.

2.1. Análisis de los datos que contiene el movie dataset.

El movie dataset contenía columnas con varios datos, en total existían un total de 24 columnas, de estas columnas se analizó el significado y el contenido que poseían, y el resultado fue el siguiente:

- Index: número entero, que se auto incrementa
- Budget: presupuesto de la película

- Genres: (no atómica) géneros de la película
- Homepage: dirección URL para más información de la película, contiene varios valores nulos
- Id: (clave primaria) número para identificar la película
- Keywords: (texto) palabras claves para identificar la película
- Original language: lenguaje en el que esta la película
- Original Title: nombre de la película (titulo) en su lenguaje original.
- Overview: (texto) resumen de la película
- Popularity: popularidad de la película
- Production Companies: (JSON) nombre de las productoras en las que trabajan y su id para identificarlas
- Production Countries: (JSON) nombre de los países donde se trabajo y su id para poder identificarlos
- Release date: fecha en la que se estrenó la película
- Revenue: ganancias de la película
- Runtime: tiempo de duración de la película
- Spoken Languages: (JSON) que contiene el id (siglas del lenguaje) y el nombre del idioma
- Status: es un campo atómico que contiene la condición en la que se encuentra la película, contiene varios valores repetidos
- Tagline: es el slogan de la película, contiene varios valores nulos
- Title: Es el título de la película en el idioma de distribución / comercialización
- Vote average: Es el promediado de la votación
- Vote count: La cantidad de votos que tiene la película

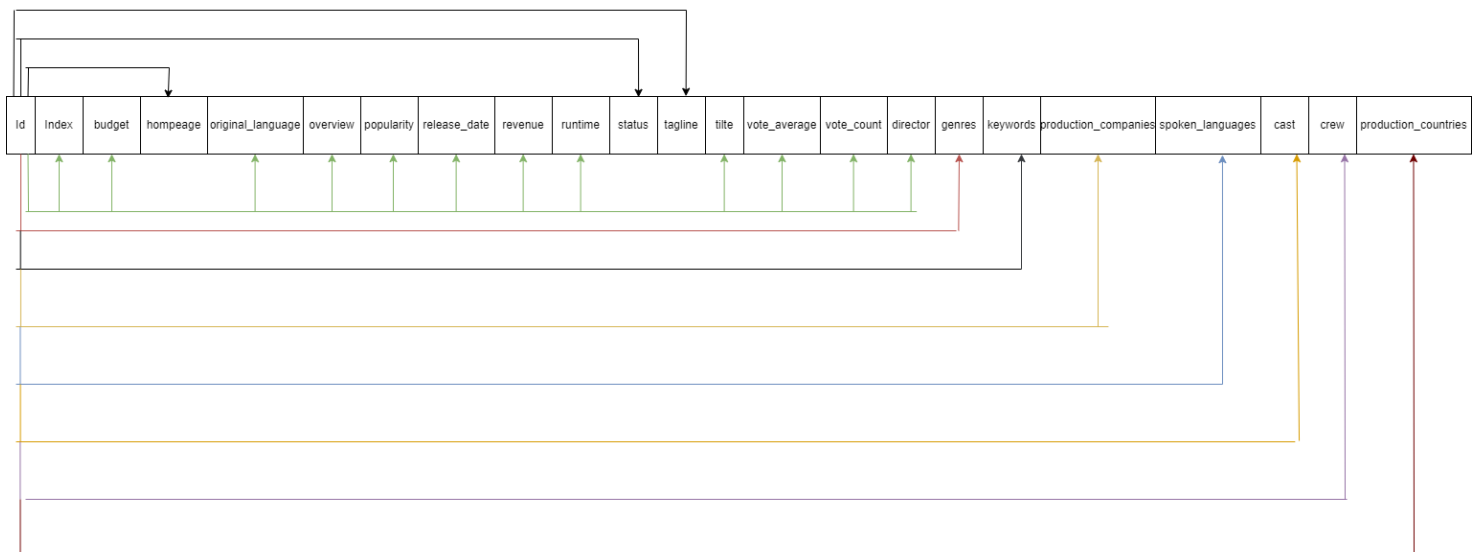
- Cast: Es el elenco principal de la película
- Crew: (JSON) Son los empleados que desarrollan tareas específicas para la producción de la película
- Director: Es el director de la película.

3. Diseño y Modelado de la Base de datos

3.1. Dependencias Funcionales

Lo primero que se realizó, fue realizar un gráfico para poder visualizar las dependencias funcionales de las columnas de nuestro archivo, para eso utilizamos herramientas externas como lo son drawio, que nos permitió visualizar de mejor manera los campos que teníamos.

El resultado fue el siguiente:



Los resultados que obtuvimos tras la comparación de nuestras columnas nos dieron como resultado las siguientes dependencias:

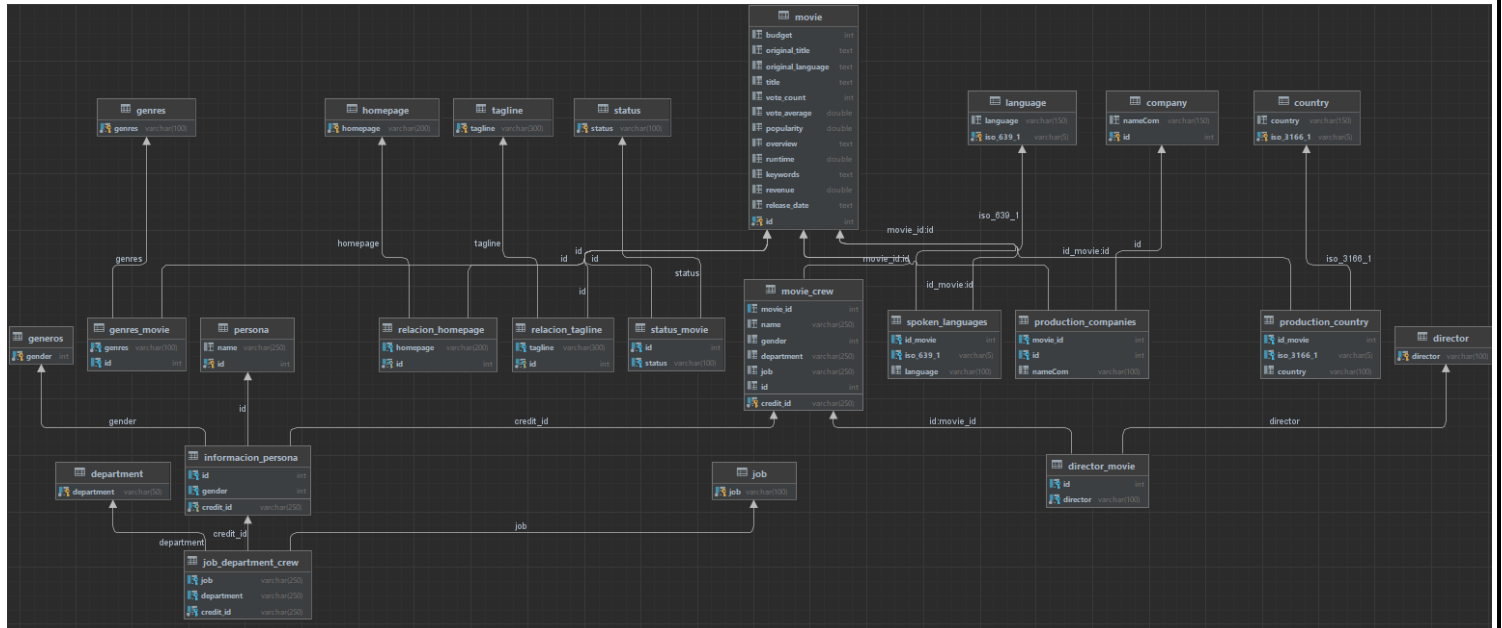
- id -> {index,index, budget, original_language, overview, popularity, release_date, revenue, runtime, title, vote_average, vote_count, director }
- id -> genres
- id -> keywords
- id -> production_companies
- id -> spoken_language
- id -> cast
- id -> crew
- id -> production_countries
- id -> homepage
- id -> status
- id -> tagline

3.2.Diseño Conceptual

Una vez entendidas las dependencias de nuestras columnas, procedimos a diseñar nuestro modelo conceptual, para generar una infraestructura de soporte de la base de datos a un nivel fácil de comprender para el desarrollador y para el usuario final.

3.4.Diseño Lógico

Una vez que tenemos el diseño físico, en el modelo lógico determinamos las entidades y sus claves primarias y claves foráneas, al igual que sus relaciones, su objetivo es obtener una representación más eficiente de los recursos que tenemos para poder mejorar las restricciones que vamos a utilizar.



4. Desarrollo

4.1. Eliminación de valores nulos

Lo primero que analizamos fueron las columnas que cuentan con una gran cantidad de valores nulos, en este caso nosotros encontramos 2 columnas que cuentan con varios valores nulos, en este caso son homepage y tagline, para poder resolver estas columnas nosotros realizamos el siguiente procedimiento:

```
53 DROP TABLE IF EXISTS homepage;
54 CREATE TABLE homepage(homepage VARCHAR(200)) AS
55     SELECT DISTINCT homepage
56     FROM movie_dataset.movie_dataset;
57
58 DELETE
59     FROM homepage
60     WHERE homepage IS NULL;
61
62 ALTER TABLE homepage
63     ADD PRIMARY KEY (homepage);
64
65
66 CREATE TABLE relacion_homepage (id INT NOT NULL PRIMARY KEY, homepage VARCHAR(200)) AS
67     SELECT DISTINCT id, homepage
68     FROM movie_dataset.movie_dataset
69     WHERE homepage != '' OR homepage IS NOT NULL;
70 -- Agregamos las claves foráneas de la relacion_homepage a movie y a homepage según corresponda.
71 ALTER TABLE relacion_homepage
72     ADD FOREIGN KEY (id)
73     REFERENCES movie(id),
74     ADD FOREIGN KEY (homepage)
75     REFERENCES homepage(homepage);
76
```

Lo primero que se realiza para trabajar con columnas con valores nulos es realizar un Distinct para poder obtener las filas únicas de nuestro archivo, de esta forma obtenemos una nueva tabla con los valores de homepage únicos, una vez que tenemos esta tabla nosotros

procedemos a eliminar dentro de esta todos los valores que se posean un null, en este caso solo es 1 ya que la tabla se encuentra normalizada.

Una vez con la tabla de homepage ya normalizada nosotros creamos la tabla que va a servir como relación de los datos que tenemos; En esta situación lo primero que realizamos es crear los atributos que va a poseer nuestra tabla, el id que sirve para poder relacionar con la tabla general y el homepage para poder relacionarlo con los datos que tiene la tabla homepage, dentro de esta relación nosotros vamos a insertar con un distinct que en este caso seria el id y el homepage, una vez que nosotros tenemos esta relación, procedemos a ubicar las claves foráneas para verificar que estamos realizando todo de forma correcta.

De esta forma en nuestro caso se eliminaron los datos nulos de la columna tagline que contaba con mas de 800 valores nulos y homepage que contaba con mas de 3000 valores nulos.

La razón por la cual nuestro grupo decidió realizar estos cambios fue debido a que consideramos que es oportuno limpiar las columnas con nulo y así poder almacenar mas espacio dentro de nuestra memoria.

4.2.Columna con valores redundantes

En nuestro caso, nosotros ubicamos que la columna status es una columna que cuenta con valores que se repiten de forma constante, los valores que tenemos que se repiten son los siguientes: post production, released y rumores, para poder solucionar este error nosotros realizamos el siguiente procedimiento.

```

9  -- ----- Normalización de Status -----
10 DROP TABLE IF EXISTS status;
11 CREATE TABLE status (status VARCHAR(100)) AS SELECT DISTINCT status
12 FROM movie_dataset.movie_dataset;
13
14 ALTER TABLE status
15 ADD PRIMARY KEY (status);
16
17 -- Creación de la relación movie-status
18 CREATE TABLE status_movie (id INT NOT NULL,
19                               status VARCHAR(100)) AS SELECT id, st.status AS status
20 FROM movie_dataset.movie_dataset movie_status, status st
21 WHERE movie_status.status = st.status;
22
23 ALTER TABLE status_movie
24 ADD FOREIGN KEY (id) REFERENCES movie(id),
25 ADD FOREIGN KEY (status) REFERENCES status(status);
26

```

El proceso que nosotros realizamos con los datos que tiene valores redundantes es bastante similar al proceso con datos nulos, lo primero que nosotros realizamos es crear una tabla donde vamos a almacenar los valores únicos que tiene nuestra columna status, en este caso los valores únicos son 3 por lo que la tabla solo cuenta con 3 valores, aquí algo interesante que nosotros realizamos es que nosotros asignamos una Primary key a la tabla con los valores reducidos.

Luego de tener esta tabla nosotros creamos la relación para poder relacionar la tabla con los valores reducidos y la tabla general donde nosotros tenemos nuestros valores atómicos, esta tabla relacional tienes dos valores los géneros y el id, esta tabla va a tener unas claves foráneas que nos van a ayudar a relacionarla con las otras tablas que tenemos, la primera clave foránea es el id donde su relación se va a mantener con la tabla general de movies, y la otra clave foránea va a estar en status donde su relación se va a mantener con la tabla que posee los 3 valores de status, nosotros insertamos dentro de esta tabla intermediaria los valores de la tabla general de movies id y status para poder poblar esta tabla, y de esta forma nosotros tenemos las tablas totalmente relacionales, nosotros realizamos este procedimiento con todas las columnas que consideramos que podemos reducir por ejemplo:

	gender
1	0
2	1
3	2

	status
1	Post Production
2	Released
3	Rumored

	job
1	24 Frame Playback
2	2D Artist
3	2D Supervisor
4	3D Animator
5	3D Artist
6	3D Coordinator
7	3D Modeller
8	3D Supervisor

	department
1	Actors
2	Art
3	Camera
4	Costume & Make-Up
5	Crew
6	Directing
7	Editing
8	Lighting
9	Production
10	Sound
11	Visual Effects
12	Writing

4.3.Columnas con valores no atómicos

Nosotros encontramos columnas que poseen valores que no son atómicos, en pocas palabras una columna que tiene varios valores, en este caso las columnas que tienen estos errores son los siguientes, genres y cast, para poder solucionar este problema nosotros realizamos el siguiente procedimiento:

Para el desarrollo de 'cast' tuvimos que prestarle mucha atención a la data presentada debido a que existían actores los cuales habían sido ingresados con un solo nombre, dos, tres o incluso su apodo. Por lo cual tuvimos que realizar cambios mediante 'Replace' para así poder establecer que cada actor tenga dos nombres y sea más fácil el procesamiento.

Como primer punto ejecutamos cierta consulta que nos permitía saber la longitud del texto sin espacios, la cantidad de espacios y de palabras. Por lo que con este paso podíamos determinar que caracteres eran necesarios de cambiar o de adecuar para su correcta implementación.

```
341 SELECT id, cast, length(cast) as longitud,  
342 length(REPLACE(cast, ' ', '')) as longitudSinEspacios,  
343 length(cast) - length(REPLACE(cast, ' ', '')) as CantidadEspacios,  
344 length(cast) - length(REPLACE(cast, ' ', '')) + 1 as CantidaDePalabras  
345 FROM cast ;
```

id	cast	longitud	longitudSinEspacios	CantidadEspacios	CantidaDePala...
3891	72/66 Edward Burns Kerry Bisn\ouuey Marsha Uietlein Caitlin Fitzg...	88	71	9	10
3892	231617 Eric Mabius Kristin Booth Crystal Lowe Geoff Gustafson Benj...	77	68	9	10
3893	126186 Daniel Henney Eliza Coupe Bill Paxton Alan Ruck Zhu Shimao	58	49	9	10
3894	25975 Drew Barrymore Brian Herzlinger Corey Feldman Eric Roberts ...	72	63	9	10
3895	49529 Taylor Kitsch Lynn Collins Samantha Morton Willem Dafoe Tho...	75	65	10	11
3896	559 Tobey Maguire Kirsten Dunst James Franco Thomas Haden Churc...	73	63	10	11

Antes de cualquier 'REPLACE' realizábamos un pequeño testeo de los cambios futuros, esto mediante una consulta usando 'SELECT'.

```
2  
3 SELECT id, cast, REPLACE(cast, 'Love E-40', 'Charlie Hustle')  
4 FROM cast WHERE cast LIKE '%Love E-40%';  
5  
6
```

Una vez que ya teníamos claro la data a modificar procedimos a ejecutar un 'UPDATE' a 'cast'.

```
337 UPDATE cast
338 SET cast = REPLACE(cast, ' G.W.Bailey', ' George Bailey' );
339
```

Y así continuamos realizando los ‘REPLACES’ necesarios:

```
136 UPDATE cast
137 SET cast = REPLACE(cast, ' Emma Eliza Regan', ' Emma Regan');
138
139 UPDATE cast
140 SET cast = REPLACE(cast, ' Taylor Scott Olson', ' Taylor Olson');
141
142 UPDATE cast
143 SET cast = REPLACE(cast, ' Audrey Lynn Tennent', ' Audrey Tennent');
144
145 UPDATE cast
146 SET cast = REPLACE(cast, 'Mackenzie E.Danny', 'Mackenzie E. Danny');
147
148 UPDATE cast
149 SET cast = REPLACE(cast, ' Lalaine', ' Lalaine Vergara-Paras');
150
151 UPDATE cast
152 SET cast = REPLACE(cast, ' Mary Kate Wiles', ' Mary Wiles');
```

```

312
313 UPDATE cast
314 SET cast = REPLACE(cast, ' T.I.', ' Clifford Harris');
315
316
317 UPDATE cast
318 SET cast = REPLACE(cast, 'Eminem', ' Marshall Mathers');
319
320 UPDATE cast
321 SET cast = REPLACE(cast, ' Litefoot', ' Gary Davis');
322
323 UPDATE cast
324 SET cast = REPLACE(cast, ' Sinbad', ' David Adkins');
325
326 UPDATE cast
327 SET cast = REPLACE(cast, 'Cher Christina', 'Cherilyn Sarkisian Christina');
328
329 UPDATE cast
330 SET cast = REPLACE(cast, ' Orianthi', ' Orianthi Retta');
331
332 UPDATE cast
333 SET cast = REPLACE(cast, ' Miyavi', ' Takamasa Ishihara');

```

4.4.Columnas con datos en formato JSON

En el desarrollo de nuestro trabajo, nosotros encontramos cuatro columnas que tienen valores en formato JSON, estas columnas tenían singularidades, por ejemplo habían 4 columnas que tenían datos parecidos por lo que pudimos reutilizar código, en cambio existía un JSON que tenían datos distintos por lo que tuvimos que modificar el código reutilizado, para poder trabajar con estas columnas nosotros realizamos lo siguiente:

En el caso de los JSON que mantenían una gran similitud eran 3, Spoken_languages, production_companies y production_countries, para estas 3 columnas se utilizaron códigos parecidos al siguiente:

```

401  ----- Normalización de spoken Languages -----
402
403  # [{"iso_639_1": "en", "name": "English"}]
404
405  DROP PROCEDURE IF EXISTS spoken_languages;
406  DELIMITER $$
407  CREATE PROCEDURE spoken_languages()
408  BEGIN
409      DECLARE done INT DEFAULT FALSE ;
410      DECLARE jsonData json ;
411      DECLARE jsonId varchar(250) ;
412      DECLARE jsonLabel varchar(250) ;
413      DECLARE movieId INT;
414      DECLARE i INT;
415

```

Lo primero que realizamos es crear un procedimiento, dentro de este procedimiento nosotros declaramos variables, estas variables son las siguientes:

El jsonData es donde se va a almacenar todos los datos de JSON, el jsonId es el id que tiene el JSON, el jsonLabel que es el nombre del json y por último el movieId que es el id de la película y por ultimo el i que es para poder trabajar dentro del ciclo while.

```

416  -- Declarar el cursor
417  DECLARE myCursor
418  CURSOR FOR
419  SELECT id, JSON_EXTRACT(CONVERT(spoken_languages USING UTF8MB4), '$[*]')
420  FROM movie_dataset.movie_dataset;
421  -- Declarar el handler para NOT FOUND
422
423  DECLARE CONTINUE HANDLER
424  FOR NOT FOUND SET done = TRUE ;
425
426  -- Abrir el cursor
427  OPEN myCursor ;
428  cursorLoop: LOOP
429
430      FETCH myCursor INTO movieId, jsonData;
431      SET i = 0;
432      IF done THEN
433          LEAVE cursorLoop ;
434      END IF ;

```

Luego en la siguiente parte del código, nosotros declaramos el cursor, el cursor está leyendo línea por línea los json y nos sirve para poder leer cada row de nuestro JSON, el

cursor entra en un loop donde lee los datos y va extrayendo con la función JSON_Extract la información que contiene el JSON, luego este loop termina.

```
436 WHILE(JSON_EXTRACT(jsonData, CONCAT('[', i, ']'))) IS NOT NULL DO
437     SET jsonId = IFNULL(JSON_EXTRACT(jsonData, CONCAT('[', i, '].iso_639_1')), '');
438     SET jsonLabel = IFNULL(JSON_EXTRACT(jsonData, CONCAT('[', i, '].name')), '');
439     SET @sql_text = CONCAT('INSERT INTO spoken_languages VALUES (' , movieId, ', ', REPLACE(jsonId, '\\', ''), ', ', jsonLabel, '); ');
440
441     PREPARE stmt FROM @sql_text;
442     EXECUTE stmt;
443     DEALLOCATE PREPARE stmt;
444     SET i = i + 1;
445 END WHILE;
446
447 END LOOP ;
448
449 CLOSE myCursor ;
450
451 END$$
452 DELIMITER ;
```

Luego nosotros entramos en el ciclo while, donde vamos a ir insertando los valores que tenemos:

Lo primero que realizamos dentro del ciclo es extraer el ID del Json y almacenarlo en JsonId, luego extraemos la cadena que acompaña al id que es otro valor distinto con el JsonLabel, en este caso dentro del Json la clave tiene el nombre de name, una vez extraído estos dos valores vamos a crear un texto donde vamos a insertar los valores que estamos almacenando, en este caso vamos a ir almacenando dentro de una tabla el id de la película para poder relacionarlo con otras tablas, luego vamos a insertar el id del Json y el nombre del Json, esta orden es un texto, por lo que en la siguiente línea de código especificamos que el texto es una orden y esta lista para ejecutarse, luego de ejecutarse vamos a aumentar nuestro contador para poder seguir trabajando dentro del ciclo while hasta que no encuentre valores para extraer dentro del JSON.

Una vez terminado todo este procedimiento cerramos nuestro cursor, ya que no existen más datos.

También marcamos donde va a terminar el procedimiento para luego ser llamado por la función call().

```
454 DROP TABLE IF EXISTS spoken_languages;
455 CREATE TABLE spoken_languages (
456     id_movie INT,
457     iso_639_1 VARCHAR(5),
458     language VARCHAR(100));
459
460 CALL spoken_languages();
461
462
463 CREATE TABLE language (
464     iso_639_1 VARCHAR(5) NOT NULL PRIMARY KEY,
465     language VARCHAR(150)) AS
466     SELECT DISTINCT iso_639_1, language FROM spoken_languages;
467
468 SELECT * from language WHERE language = '';
469
470 ALTER TABLE spoken_languages
471     ADD FOREIGN KEY (id_movie) REFERENCES movie(id),
472     ADD FOREIGN KEY (iso_639_1) REFERENCES language(iso_639_1);
473
474
```

Lo último que nosotros realizamos es dropear en caso de que exista la tabla donde vamos a almacenar los 3 valores, luego la creamos, para poder llamar al procedimiento nosotros hacemos uso de la función call y el nombre del procedimiento.

Creamos las relaciones de nuestro archivo Json donde en este caso vamos a crear una tabla que es una entidad llamada lenguaje, esta tabla va a almacenar sin olvidar el distinct los datos que contenía el JSON, luego alteramos la tabla donde insertamos los tres valores de forma que la relación nos queda que en la tabla con los 3 valores relacionamos de forma

que el id de la movie se va a relacionar con el id de la película de la tabla general y el id del Json se va a relacionar con el id de del Json de la tabla que contiene el json con el distinct.

De esta forma nosotros trabajamos con los 3 tipos de Json mencionados, la cuestión es que el código sufrió ligeros cambios con la columna de crew, con esta columna se modificó lo siguiente:

```
552 -- [{"name": "Stephen E. Rivkin", "gender": 0, "department": "Editing", "job": "Editor", "credit_id": "52fe48009251416c750aca23", "id": 1721}
553 DROP PROCEDURE IF EXISTS crew;
554 DELIMITER $$
555 CREATE PROCEDURE crew()
556 BEGIN
557     DECLARE done INT DEFAULT FALSE ;
558     DECLARE jsonData json ;
559     DECLARE jsonName varchar(250);
560     DECLARE jsonGender INT;
561     DECLARE jsonDepartment VARCHAR(250);
562     DECLARE jsonJob VARCHAR(250);
563     DECLARE jsonCredit_id VARCHAR(250);
564     DECLARE jsonId INT;
565     DECLARE movieId INT;
566     DECLARE i INT;
```

Se declararon mas variables ya que el Json contenía más datos en forma de clave: valor

```
588 WHILE(JSON_EXTRACT(jsonData, CONCAT('$[', i, ']')) IS NOT NULL) DO
589     SET jsonId = IFNULL(JSON_EXTRACT(jsonData, CONCAT('$[', i, '].id')), '');
590     SET jsonName = IFNULL(JSON_EXTRACT(jsonData, CONCAT('$[', i, '].name')), '');
591     SET jsonGender = IFNULL(JSON_EXTRACT(jsonData, CONCAT('$[', i, '].gender')), '');
592     SET jsonDepartment = IFNULL(JSON_EXTRACT(jsonData, CONCAT('$[', i, '].department')), '');
593     SET jsonJob = IFNULL(JSON_EXTRACT(jsonData, CONCAT('$[', i, '].job')), '');
594     SET jsonCredit_id = IFNULL(JSON_EXTRACT(jsonData, CONCAT('$[', i, '].credit_id')), '');
595     SET @sql_text = CONCAT('INSERT INTO movie_crew VALUES (' , movieId, ', ',
596         jsonName, ', ', jsonGender, ', ', jsonDepartment, ', ', jsonJob, ', ', jsonCredit_id,
597         ', ', jsonId, '); ');
```

También en el ciclo While nosotros agregamos vamos sets, ya que tenemos que extraer los datos de las distintas claves que contiene el Json.

Nota: En este apartado nosotros teníamos que limpiar el archivo Json antes de trabajarlo ya que se encontraba mal formado, en este caso existían comillas dobles donde solo debían estar comillas simples, entre otros errores, para poder solucionar este error nuestro profesor nos ayudó facilitándonos el código, el código utilizado fue el siguiente:

```

542 UPDATE movie_dataset.movie_dataset SET crew = CONVERT (
543 REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(crew,
544 ' ', '\ '),
545 '{', '{ '),
546 '\': '\', ': ' '),
547 '\', '\', ', '),
548 '\': ', ': '),
549 ', \', ', ')
550 USING UTF8mb4 );

```

El UPDATE en este caso nos sirve para actualizar los datos que tiene una columna en específico.

5. Consultas SQL

-- ¿Cuáles son las películas más recientes?

```
SELECT * FROM movie ORDER BY release_date DESC;
```

	id	budget	keywords	original_language	original_title	overview
1	388097	0		en	America Is Still the Place	1971 pos
2	426469	0		en	Growing Up Smith	In 1979,
3	325373	0	small town lovers bear north pole	en	Two Lovers and a Bear	Set in a
4	374461	8800000	cook friendship	en	Mr. Church	A unique
5	339488	8500000	slavery	en	The Birth of a Nation	Nat Turn
6	385736	0	blow job cigarette smoking illegal drugs smoking weed shoes	en	Kicks	When his
7	332285	35000000	drug abuse experiment television conspiracy alien abduction	en	Antibirth	In a des
8	184341	20000000		en	Hands of Stone	The lega
9	271969	100000000	betrayal vengeance	en	Ben-Hur	A false
10	294272	65000000	feral child remake dragon orphan 1980s	en	Petes Dragon	Pete is
11	297761	175000000	dc comics shared universe anti hero secret mission villain	en	Suicide Squad	From DC
12	315011	15000000	monster godzilla giant monster destruction kaiju	ja	シン・ゴジラ	From the
13	340611	0	based on novel jewish life ohio 1950s	en	Indignation	In 1951,
14	376659	26000000	alcohol bar party divorce family	en	Bad Moms	When the

-- ¿Cuáles son las películas con una duración superior a Y minutos?

```
SELECT * FROM movie WHERE runtime > 160;
```

	id	budget	keywords	original_language	original_title	overview
1	111	25000000	miami corruption capitalism cuba prohibition	en	Scarface	After g...
2	120	93000000	elves dwarves orcs middle-earth (tolkien) hobbit	en	The Lord of the Rings: The Fellowship of the Ring	Young h...
3	121	79000000	elves orcs middle-earth (tolkien) hobbit based on novel	en	The Lord of the Rings: The Two Towers	Frodo ai...
4	122	94000000	elves orcs middle-earth (tolkien) based on novel suspicion	en	The Lord of the Rings: The Return of the King	Aragorn...
5	197	72000000	individual scotland in love with enemy legend independence	en	Braveheart	Enraged...
6	238	6000000	italy love at first sight loss of father patriarch organized c...	en	The Godfather	Spanning...
7	240	13000000	italo-american cuba vororte melancholy praise	en	The Godfather: Part II	In the i...
8	242	54000000	italy christianity new york assassination italo-american	en	The Godfather: Part III	In the i...
9	254	207000000	film business screenplay show business film making film produc...	en	King Kong	In 1933...
10	285	300000000	ocean drug abuse exotic island east india trading company love...	en	Pirates of the Caribbean: At Worlds End	Captain...
11	297	90000000	life and death love at first sight broken engagement fireworks...	en	Meet Joe Black	When thi...
12	311	30000000	life and death corruption street gang rape sadistic	en	Once Upon a Time in America	A former...
13	334	37000000	farewell regret parents kids relationship suicide attempt beco...	en	Magnolia	An epic...
14	335	5000000	showdown bountyv bountyv hunter loss of brother sadness	it	C'era una volta il West	This cl...

-- ¿Cuáles son las películas dirigidas por Z?

SELECT * FROM movie, director

WHERE director.id = movie.id

AND director = 'James Cameron';

	movie_id	budget	keywords	original_language	original_title	overview
1	19995	237000000	culture clash future space war space colony society	en	Avatar	In the 22nd century, a pa...
2	597	200000000	shipwreck iceberg ship panic titanic	en	Titanic	84 years later, a 101-yea...
3	280	100000000	cyborg shotgun post-apocalyptic dystopia moral ambiguity	en	Terminator 2: Judgment Day	Nearly 10 years have passe...
4	36955	115000000	spy terrorist florida gun kidnapping	en	True Lies	Harry Tasker is a secret...
5	2756	70000000	ocean sea diving suit flying saucer nuclear missile	en	The Abyss	A civilian oil rig crew is...
6	679	18500000	android extraterrestrial technology space marine spaceman cryog...	en	Aliens	When Ripley's Lifepod is f...
7	218	6400000	saving the world artificial intelligence rebel cyborg shotgun	en	The Terminator	In the post-apocalyptic fu...

-- ¿Cuáles son las películas con un actor/actriz específica en el reparto?

SELECT * FROM movie WHERE cast LIKE '%Roberth Downey Jr%';

¿Cuáles son las películas más recientes?			
id	title	runtime	cast
1	Iron Man	1726	126 Robert Downey Jr. Terrence Howard Jeff Bridges Shaun Toub Gwyneth Paltrow
2	Zodiac	1949	157 Jake Gyllenhaal Robert Downey Jr. Mark Ruffalo Anthony Edwards Brian Cox
3	Lucky You	1950	124 Eric Bana Drew Barrymore Robert Duvall Debra Messing Robert Downey Jr.
4	A Scanner Darkly	3589	100 Keanu Reeves Winona Ryder Woody Harrelson Robert Downey Jr. Mitch Baker
5	Gothika	4970	98 Halle Berry Robert Downey Jr. Charles S. Dutton Penelope Cruz John Carroll Lynch
6	Kiss Kiss Bang Bang	5236	183 Robert Downey Jr. Val Kilmer Michelle Monaghan Corbin Bernsen Dash Mihok
7	Tropic Thunder	7446	107 Ben Stiller Jack Black Robert Downey Jr. Nick Nolte Steve Coogan
8	Charlie Bartlett	8669	97 Anton Yelchin Robert Downey Jr. Hope Davis Kat Dennings Tyler Hilton
9	Home for the Holidays	9089	103 Holly Hunter Robert Downey Jr. Anne Bancroft Charles Durning Claire Danes
10	The Shaggy Dog	10067	98 Tim Allen Kristin Davis Danny Glover Spencer Breslin Robert Downey Jr.
11	Iron Man 2	10138	124 Robert Downey Jr. Gwyneth Paltrow Don Cheadle Scarlett Johansson Mickey Rourke
12	Sherlock Holmes	10528	128 Robert Downey Jr. Jude Law Rachel McAdams Mark Strong Eddie Marsan
13	The Soloist	17332	109 Robert Downey Jr. Jamie Foxx Catherine Keener Nelsan Ellis Michael Bunin
14	Fun: An Imaginary Portrait of Diane Arbus	18615	122 Nicole Kidman Robert Downey Jr. Ty Burrell Harris Yulin Jane Alexander

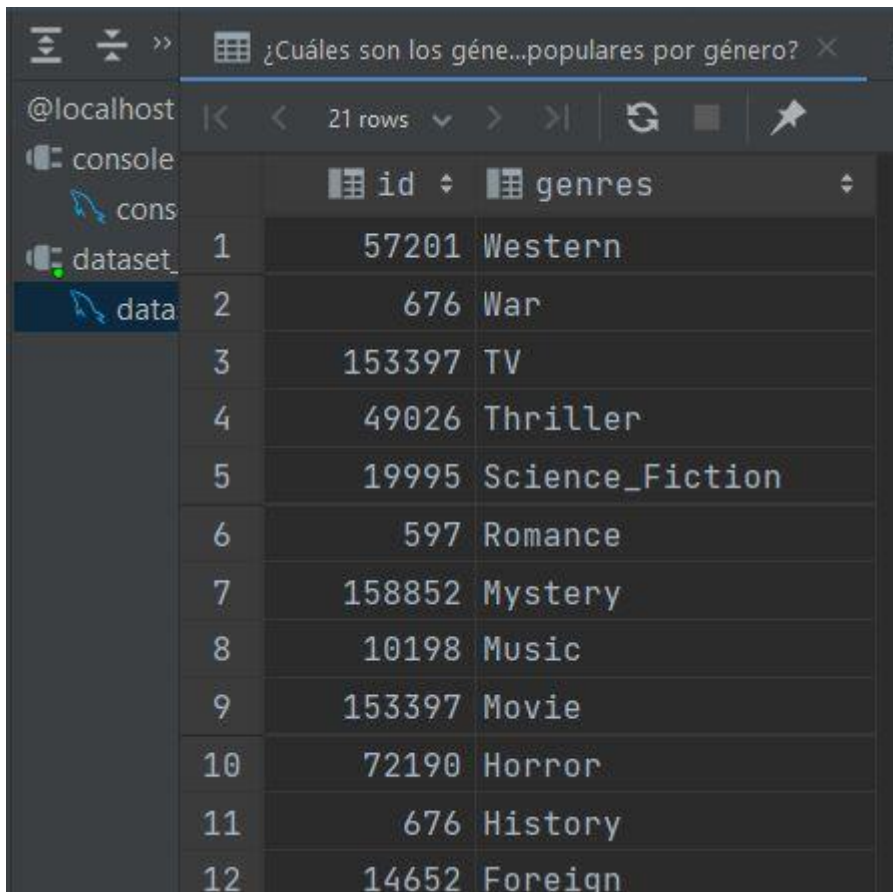
-- ¿Cuáles son los géneros más populares por género?

SELECT id, genres

FROM movie, genres

WHERE movie.id = genres.id_movie

GROUP BY genres ORDER BY genres DESC;



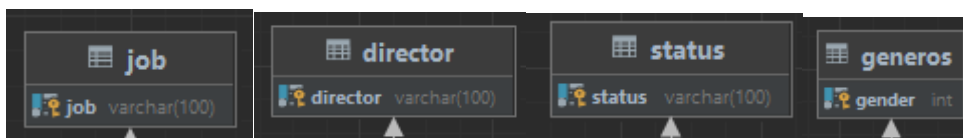
The screenshot shows a database interface with a query result table. The table has two columns: 'id' and 'genres'. The data is as follows:

	id	genres
1	57201	Western
2	676	War
3	153397	TV
4	49026	Thriller
5	19995	Science_Fiction
6	597	Romance
7	158852	Mystery
8	10198	Music
9	153397	Movie
10	72190	Horror
11	676	History
12	14652	Foreign

6. Normalización

Los resultados de nuestra tabla normalizada fueron los siguientes:

- Las tablas que nosotros tenemos con un valor único y que solo contienen una columna son las siguientes: status, genero, department, tagline, genres, homepages, director y job:



The diagram shows four separate table boxes, each representing a normalized table with a single column:

job
job varchar(100)

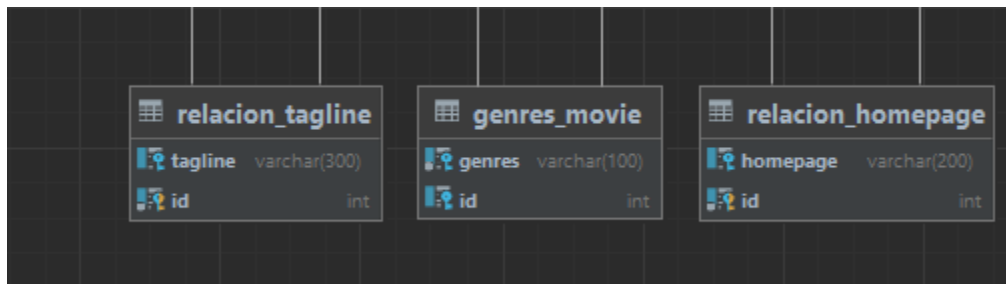
director
director varchar(100)

status
status varchar(100)

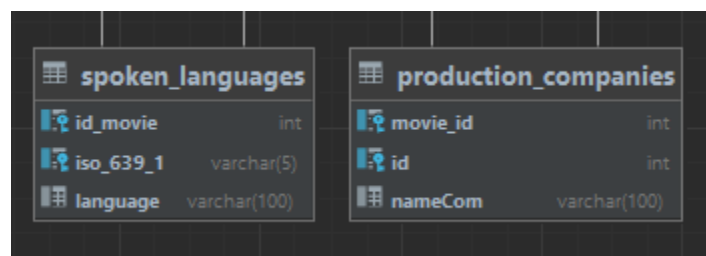
generos
gender int



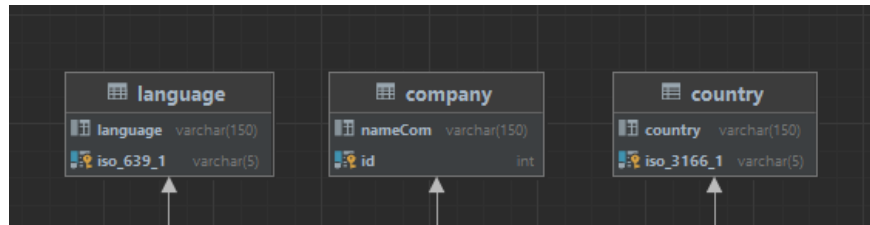
- Para poder normalizar las tablas Homepage y tagline nosotros primero eliminamos todos los datos nulos, por ejemplo en la columna homepage se eliminaron aproximadamente un total de 3100 columnas que tenían nulos, luego nosotros nos dimos cuenta que existían columnas repetidas por lo que usamos un distinct para poder limpiarla y eliminamos nulos luego aplicamos la relación.



- Las tablas relacionales están conectadas a las tablas que tienen una sola columna y un Primary key, y tienen el id de la tabla general para poder encontrar la relación.



- En las tablas que contienen JSON nosotros, les asignamos al momento de realizar el Insert el id de la tabla general para poder crear la relación.



- De los Json sacamos una columna con valores únicos donde asignamos como clave primaria el ID que tenían

7. Conclusiones

- El proyecto hecho ayudo a nuestro aprendizaje integro como próximos profesionales de las materias dadas como lo es Base de datos y programación Funcional y reactiva. La presentación de varios desafíos a lo largo del proyecto ha ayudado a nuestra forma de afrontar grandes retos en base de datos, la aplicación de métodos eficientes para la limpieza ayudo a la visión más integra del proyecto.
- El reto propuesto incentivo de manera directa a nuestra búsqueda de más formas y funciones para poder mejorar la calidad de nuestro proyecto, de tal manera que pudimos deducir nuevas instancias con las cuales pudimos concatenar con los conocimientos dados por nuestros Docentes para poder trabajar grandes volúmenes de data de manera eficiente y organizada.
- A base de este proyecto, ya que se asimila a proyectos más apegados a la realidad laboral, nos indujo formar nuevos conceptos y proponer nuevas estrategias para modelado, inserción, limpieza de datos, esto ayudo de forma variada a nuestra formación para afrontar nuevos retos en los cuales necesitemos aplicar los conocimientos aprendidos en este ciclo.

- Se pudo concluir que el trabajo en equipo ayudo bastante en en la distribución de ideas, El trabajo compartido de este proyecto fue aclarándonos dudas que teníamos en aquellos momentos, aprendiendo de nuestros errores. De tal manera que tuvimos retroalimentación variada por cada integrante ya que cada uno buscaba la manera más eficiente de dar solución a las diferentes cuestiones que se presentaron.

8. Aprendizaje

Dentro de la elaboración de este proyecto se presentaron algunas complicaciones con respecto al manejo de nuevas estructuras de datos, manipulación de caracteres y las relaciones entre entidades. Pero con una fuerte investigación, pudimos sobrellevar estos obstáculos y buscar la solución más adecuada, por lo que:

-Aprendimos que se puede llevar a cabo la limpieza y manipulación de un dataset utilizando fundamentos de bases de datos.

-SQL ofrece varias funciones que nos facilitan varias tareas en el tipado de sentencias y manejo de estructuras de datos como lo son los 'Json'.

-Los procedimientos y cursores fueron fundamentales para el desarrollo, recorrido e implementación de nuestra base de datos, ya que permitían de una forma dinámica recorrer los 'Json' y determinar su 'Id' y proceder con una normalización correcta.

- Los campos nulos, las claves foráneas y las claves primarias son elementos cruciales en la estructuración y organización de una base de datos. Los campos nulos permiten especificar si un valor es opcional u obligatorio en una tabla, lo que asegura la integridad de los datos al evitar registros vacíos. Las claves foráneas establecen relaciones entre tablas, permitiendo la integración de información relacionada y evitando redundancias en la base de datos. Por último, la clave primaria es un valor único e irrepetible que identifica a cada registro en una tabla y garantiza la integridad referencial al relacionar registros en diferentes tablas. En resumen, los campos nulos, las claves foráneas y las claves primarias son elementos esenciales para mantener una base de datos organizada, integrada y coherente.

9. Bibliografía

amazon web services. (06 de Agosto de 2022). *amazon web services*. Obtenido de Qué es la visualización de datos: <https://aws.amazon.com/es/what-is/data-visualization/>

EmpresaActual. (25 de Enero de 2021). *EmpresaActual*. Obtenido de Explotación de datos con ETL: <https://www.empresaactual.com/explotacion-de-datos-con-etl/>

IBM, D. (12 de 1 de 2023). *IBM*. Obtenido de Importación o Extracción, Transformación, Carga (ETL): <https://www.ibm.com/docs/es/license-metric-tool?topic=concepts-import-extract-transform-load-etl>

Jetbrains. (2023). *PyCharm Help*. Obtenido de Code completion: <https://www.jetbrains.com/help/pycharm/auto-completing-code.html>

Lucidchart. (2023). *Lucidchart.com*. Obtenido de Qué es un modelo de base de datos: <https://www.lucidchart.com/pages/es/que-es-un-modelo-de-base-de-datos>

Microsoft. (2023). *Extracción, transformación y carga de datos (ETL)*. Obtenido de Azure Architecture Center: <https://learn.microsoft.com/es-es/azure/architecture/data-guide/relational-data/etl>