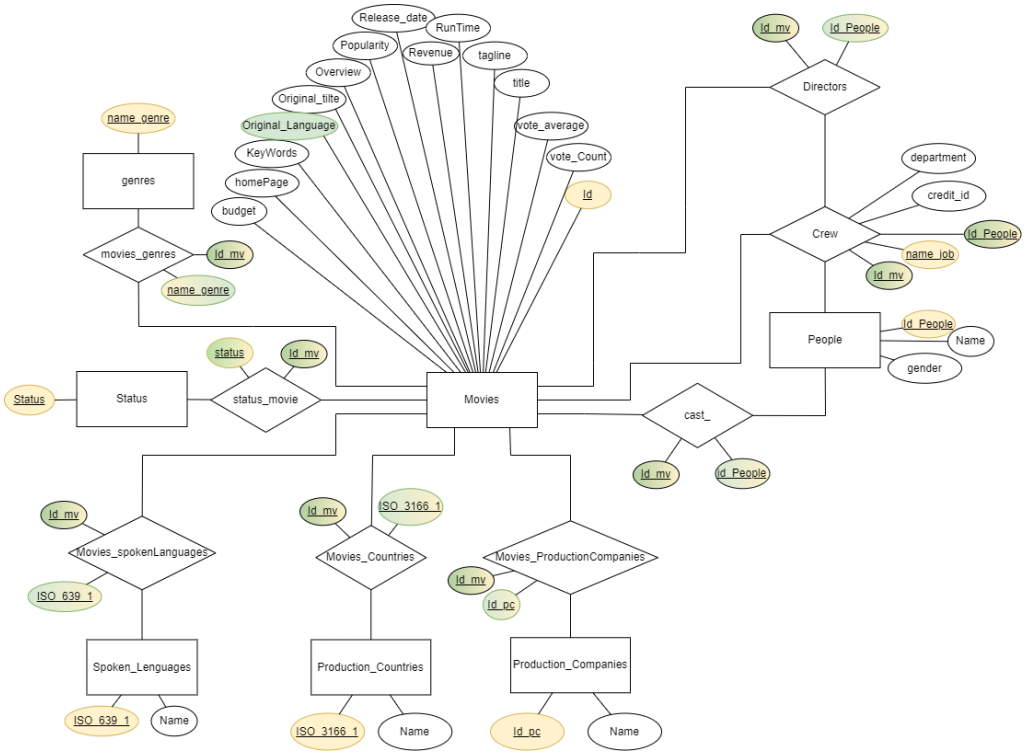
PROYECTO MOVIE_DATASET

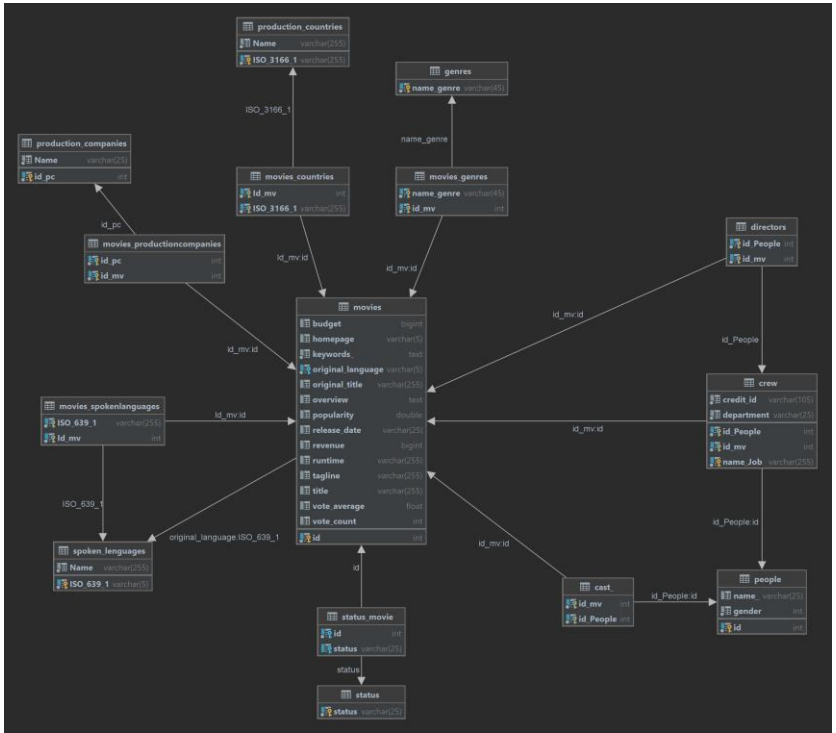
Fundamentos de Base de Datos

Edgar Santiago Espinoza Velásquez

MAPA CONCEPTUAL

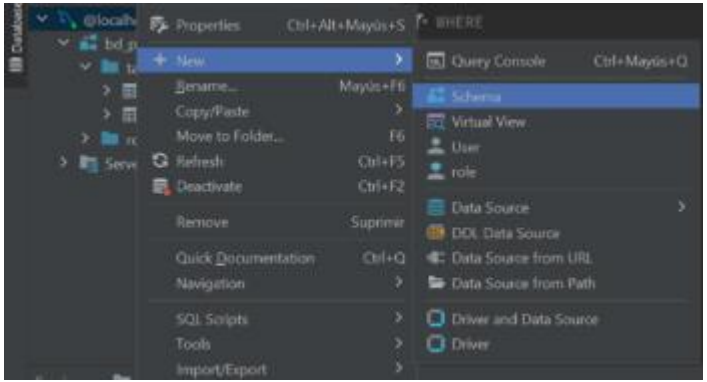


MAPA LÓGICO

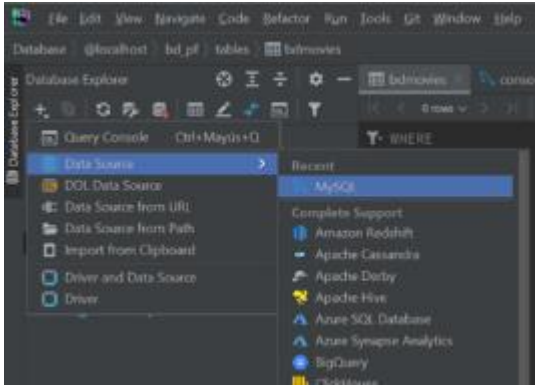

production_countries
Name varchar(255)
ISO_3166_1 varchar(255)
genres
name_genre varchar(45)
ISO_3166_1
name_genre
production_companies
Name varchar(25)
id_pc int
movies_countries
id_mv int
ISO_3166_1 varchar(255)
movies_genres
name_genre varchar(45)
id_mv int
id_pc
movies_productioncompanies
id_pc int
id_mv int
id_mv:id
id_mv:id
directors
id_People int
id_mv int
movies
budget bigint
homepage varchar(5)
keywords text
original_language varchar(3)
original_title varchar(255)
overview text
popularity double
release_date varchar(25)
revenue bigint
runtime varchar(255)
tagline varchar(255)
title varchar(255)
vote_average float
vote_count int
id int
id_mv:id
id_People
id_mv:id
movies_spokenlanguages
ISO_639_1 varchar(255)
id_mv int
id_mv:id
crew
credit_id varchar(105)
department varchar(25)
id_People int
id_mv int
name_Job varchar(255)
id_mv:id
ISO_639_1
original_language:ISO_639_1
id_People:id
spoken_languages
Name varchar(255)
ISO_639_1 varchar(5)
id_mv:id
id
cast
id_mv int
id_People int
id_People:id
people
name_ varchar(25)
gender int
id int
status_movie
id int
status varchar(25)
status
status
status varchar(25)

IMPORTACIÓN CSV

1. CREACIÓN DEL SCHEMA

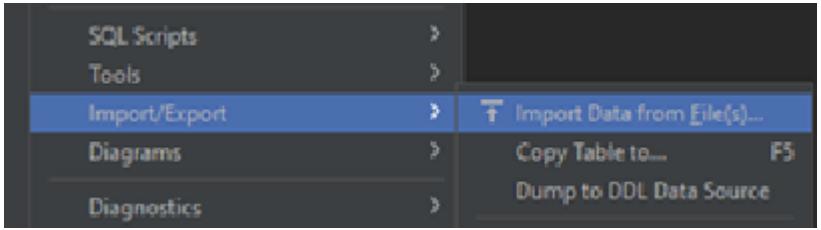


2. CONEXIÓN A LA BASE DE DATOS

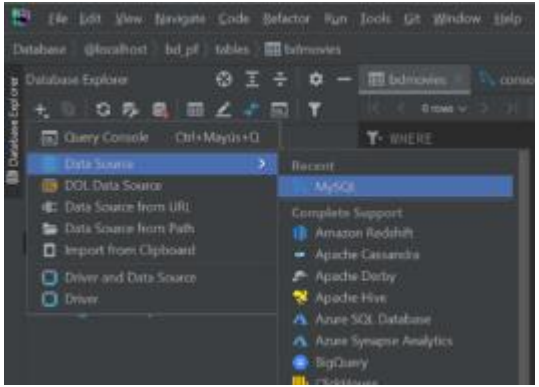


IMPORTACIÓN CSV

3. IMPORTACIÓN CSV



4. PREVIEW IMPORTACIÓN



Creación Tablas Temporales

```
DROP PROCEDURE IF EXISTS creacion_tablas ;
DELIMITER $$
CREATE PROCEDURE creacion_tablas ()
BEGIN
    DROP TABLE IF EXISTS Production_Countries_temp;
    SET @sql_text = 'CREATE TABLE Production_Countries_temp (
        iso_3166_1 varchar(10) NOT NULL,
        Name varchar(155) NOT NULL
    );';
    PREPARE stmt FROM @sql_text;
    EXECUTE stmt;
    DROP TABLE IF EXISTS production_Companies_temp;
    SET @sql_text = 'CREATE TABLE Production_Companies_temp (
        id_pc int NOT NULL,
        Name varchar(155) NOT NULL
    );';
    PREPARE stmt FROM @sql_text;
    EXECUTE stmt;
```

```
    DROP TABLE IF EXISTS spoken_languages_temp;
    SET @sql_text = 'CREATE TABLE spoken_languages_temp (
        iso_639_1 varchar(10) NOT NULL,
        Name varchar(155) DEFAULT NULL
    );';
    PREPARE stmt FROM @sql_text;
    EXECUTE stmt;
    DROP TABLE IF EXISTS People_temp;
    SET @sql_text = 'CREATE TABLE `People_temp` (
        `id` int NOT NULL,
        `name_` varchar(155) DEFAULT NULL,
        `gender` int NOT NULL
    );';
    PREPARE stmt FROM @sql_text;
    EXECUTE stmt;
    DEALLOCATE PREPARE stmt;

END$$
DELIMITER ;
CALL creacion_tablas();
```

Tablas Movies - Insert

```
DROP PROCEDURE IF EXISTS Procedure_Movie ;
DELIMITER $$
CREATE PROCEDURE Procedure_Movie()
BEGIN
      DROP TABLE IF EXISTS Movies;
        CREATE TABLE `Movies` (
      `budget` bigint DEFAULT NULL,
      `homepage` varchar(255) DEFAULT NULL,
      `id` int NOT NULL,
      `keywords_` text DEFAULT NULL,
      `original_language` varchar(5) NOT NULL,
      `original_title` varchar(255) DEFAULT NULL,
      `overview` text,
      `popularity` double DEFAULT NULL,
      `release_date` varchar(25) DEFAULT NULL,
      `revenue` bigint DEFAULT NULL,
      `runtime` varchar(255) DEFAULT NULL,
      `tagline` varchar(255) DEFAULT NULL,
      `title` varchar(255) DEFAULT NULL,
      `vote_average` float DEFAULT NULL,
      `vote_count` int DEFAULT NULL,
      PRIMARY KEY (`id`)
    );
```

```
      INSERT INTO Movies (budget, homepage,id,keywords_,original_language,original_title,
                          overview,popularity,release_date, revenue,runtime,
                          tagline,title,vote_average,vote_count)
        SELECT budget, homepage,id,keywords,original_language,original_title,
               overview,popularity,release_date,revenue,runtime,tagline,title,
               vote_average,vote_count
        FROM movie_dataset;
END $$
DELIMITER ;
CALL Procedure_Movie();
```

Tablas Status – Status_Movie

```
DROP PROCEDURE IF EXISTS Procedure_Status ;
DELIMITER $$
    CREATE PROCEDURE Procedure_Status()
    BEGIN
        DROP TABLE IF EXISTS Status;
    CREATE TABLE Status (status VARCHAR(25)) AS SELECT DISTINCT status
    FROM movie_dataset;

    ALTER TABLE Status
    ADD PRIMARY KEY (status);
        DROP TABLE IF EXISTS Status_movie;
    #Creación e INSERTS movie-status
    CREATE TABLE Status_movie (id INT NOT NULL,
                               status VARCHAR(25))AS SELECT id, st.status AS status
    FROM movie_dataset movie_status, Status st
    WHERE movie_status.status = st.status;

    ALTER TABLE status_movie
    ADD FOREIGN KEY (id) REFERENCES Movies(id),
        ADD FOREIGN KEY (status) REFERENCES Status(status);

END $$
DELIMITER ;
CALL Procedure_Status();
```

Tablas Status – Status_Movie

```
DROP PROCEDURE IF EXISTS Procedure_Status ;
DELIMITER $$
    CREATE PROCEDURE Procedure_Status()
    BEGIN
        DROP TABLE IF EXISTS Status;
    CREATE TABLE Status (status VARCHAR(25)) AS SELECT DISTINCT status
    FROM movie_dataset;

    ALTER TABLE Status
    ADD PRIMARY KEY (status);
        DROP TABLE IF EXISTS Status_movie;
    #Creación e INSERTS movie-status
    CREATE TABLE Status_movie (id INT NOT NULL,
                               status VARCHAR(25))AS SELECT id, st.status AS status
    FROM movie_dataset movie_status, Status st
    WHERE movie_status.status = st.status;

    ALTER TABLE status_movie
    ADD FOREIGN KEY (id) REFERENCES Movies(id),
        ADD FOREIGN KEY (status) REFERENCES Status(status);

END $$
DELIMITER ;
CALL Procedure_Status();
```

Tablas Directors

```
#---------------------------------------------PROCEDURES Directors ----------------------
DROP PROCEDURE IF EXISTS Procedure_Directors ;
DELIMITER $$
    CREATE PROCEDURE Procedure_Directors()
    BEGIN
        DROP TABLE IF EXISTS Directors;
    CREATE TABLE `Directors` (
      `id_People` int NOT NULL,
      `id_mv` int NOT NULL,
      PRIMARY KEY (`id_People`,`id_mv`),
      FOREIGN KEY (`id_People`) REFERENCES `Crew` (`id_People`),
      FOREIGN KEY (`id_mv`) REFERENCES `Movies` (`id`)
    );
        INSERT IGNORE INTO Directors(id_People, id_mv)
            SELECT  c.id_People, c.id_mv FROM Crew c,People p,movie_dataset m
            WHERE  c.name_Job = 'Director' AND p.id = c.id_People AND m.id = c.id_mv;

END $$
DELIMITER ;
CALL Procedure_Directors();
```

Tabla Cast_

```
#-----------------------------------------------PROCEDURES Cast ---------
DROP TABLE IF EXISTS cast_;
CREATE TABLE `cast_` (
  `id_mv` int NOT NULL,
  `id_People` int NOT NULL,
  PRIMARY KEY (`id_mv`,`id_People`),
  FOREIGN KEY (`id_mv`) REFERENCES `Movies` (`id`),
  FOREIGN KEY (`id_People`) REFERENCES `People` (`id`)
);
```

Tabla Genres

```sql
DROP PROCEDURE IF EXISTS Procedure_Genres ;
DELIMITER $$
    CREATE PROCEDURE Procedure_Genres()
    BEGIN
        DROP TABLE IF EXISTS genres;
    CREATE TABLE genres(genres VARCHAR(100)) AS
    SELECT DISTINCT (
    SUBSTRING_INDEX(SUBSTRING_INDEX(genres,' ', 5), ' ', -1)) AS genres
    FROM movie_dataset;
    DELETE
    FROM genres
    WHERE genres IS NULL;
```

Tabla Movies_Genres

```
    ALTER TABLE genres
        ADD PRIMARY KEY (genres);
    DROP TABLE IF EXISTS Movies_genres;
    CREATE TABLE Movies_genres AS
        SELECT tg.genres, id
        FROM genres tg, movie_dataset mv
        WHERE INSTR(mv.genres, tg.genres )>0;

    ALTER TABLE Movies_genres
    ADD FOREIGN KEY (id)
        REFERENCES Movies(id),
        ADD FOREIGN KEY (genres)
            REFERENCES genres(genres);
END $$
DELIMITER ;
CALL Procedure_Genres();
```

Tablas JSON

```
DROP PROCEDURE IF EXISTS Procedurejson_spokenLenguages ;
DELIMITER $$
CREATE PROCEDURE Procedurejson_spokenLenguages ()
BEGIN
 DECLARE done INT DEFAULT FALSE ;
 DECLARE jsonData json ;
 DECLARE jsonId varchar(250) ;
 DECLARE jsonLabel varchar(250) ;
 DECLARE i INT;

 -- Declarar el cursor
 DECLARE myCursor
  CURSOR FOR
   SELECT JSON_EXTRACT(CONVERT(spoken_languages USING UTF8MB4), '$[*]') FROM movie_dataset;

 -- Declarar el handler para NOT FOUND (esto es marcar cuando el cursor ha llegado a su fin)
 DECLARE CONTINUE HANDLER
  FOR NOT FOUND SET done = TRUE ;
 -- Abrir el cursor
 OPEN myCursor  ;
```

LOOP

```
cursorLoop: LOOP
  FETCH myCursor INTO jsonData;
  -- Controlador para buscar cada uno de lso arrays
    SET i = 0;
  -- Si alcanzo el final del cursor entonces salir del ciclo repetitivo
  IF done THEN
   LEAVE  cursorLoop ;
  END IF ;

  WHILE(JSON_EXTRACT(jsonData, CONCAT('$[', i, ']')) IS NOT NULL) DO

  SET jsonId = IFNULL(JSON_EXTRACT(jsonData,  CONCAT('$[', i, '].iso_639_1')), '') ;
  SET jsonLabel = IFNULL(JSON_EXTRACT(jsonData, CONCAT('$[', i,'].name')), '') ;
  SET i = i + 1;
  SET @sql_text = CONCAT('INSERT INTO spoken_languages_temp VALUES (', REPLACE(jsonId,'\'',''), ', ', jsonLabel, '); ');
PREPARE stmt FROM @sql_text;
EXECUTE stmt;
DEALLOCATE PREPARE stmt;
  END WHILE;

END LOOP ;
```

Tabla Spoken_Languages

```
    DROP TABLE IF EXISTS spoken_languages;
    CREATE TABLE spoken_languages AS
    SELECT Distinct iso_639_1,name
    FROM spoken_languages_temp ;

    ALTER TABLE spoken_languages
    ADD PRIMARY KEY (iso_639_1);
    DROP TABLE spoken_languages_temp;
 CLOSE myCursor ;
END$$
DELIMITER ;

CALL Procedurejson_spokenLenguages ();
```

Tabla Movies_SpokenLanguages

```sql
DROP TABLE IF EXISTS Movies_spokenLanguages;
CREATE TABLE Movies_spokenLanguages (
      `ISO_639_1` varchar(255) NOT NULL,
      `Id_mv` int NOT NULL,
      PRIMARY KEY (`ISO_639_1`,`Id_mv`),
      FOREIGN KEY (`Id_mv`) REFERENCES `Movies` (`id`),
      FOREIGN KEY (`ISO_639_1`) REFERENCES `spoken_Languages` (`ISO_639_1`)
    );
DROP PROCEDURE IF EXISTS ProcedureRelacion_spokenLenguages ;
DELIMITER $$
CREATE PROCEDURE ProcedureRelacion_spokenLenguages ()
BEGIN
 DECLARE done INT DEFAULT FALSE ;
 DECLARE jsonData json ;
 DECLARE jsonId varchar(250) ;
 DECLARE jsonLabel varchar(250) ;
 DECLARE resultSTR LONGTEXT DEFAULT '';
 DECLARE i INT;
 DECLARE idmv INT;
```

```sql
 -- Abrir el cursor
 OPEN myCursor  ;
 cursorLoop: LOOP
  FETCH myCursor INTO idmv ,jsonData;
  -- Controlador para buscar cada uno de lso arrays
    SET i = 0;
  -- Si alcanzo el final del cursor entonces salir del ciclo repetitivo
  IF done THEN
   LEAVE  cursorLoop ;
  END IF ;
  WHILE(JSON_EXTRACT(jsonData, CONCAT('$[', i, ']')) IS NOT NULL) DO

  SET jsonId = IFNULL(JSON_EXTRACT(jsonData,  CONCAT('$[', i, '].iso_639_1')), '') ;
  SET i = i + 1;
  SET @sql_text = CONCAT('INSERT INTO Movies_spokenLanguages VALUES (', REPLACE(jsonId,'\'',''), ', ', idmv, '); ');
    PREPARE stmt FROM @sql_text;
    EXECUTE stmt;
    DEALLOCATE PREPARE stmt;
  END WHILE;
 END LOOP ;
 CLOSE myCursor ;
END$$
DELIMITER ;
CALL ProcedureRelacion_spokenLenguages ();
```

NOTA

En esta presentación se encuentra
solo un ejemplo de tabla a partir de columnas tipo JSON, para evitar la redundancia.

GRACIAS!!