Proyecto Final Base de Datos

Alumno: Gastón A. López Duarte

Comisión: 31270

Contenido:

Proyecto final My Anime List

1 – Introducción

Para llevar a cabo el proyecto se utilizó un dataset sacado de la siguiente url:
https://www.kaggle.com/datasets/azathoth42/myanimelist
Este conjunto de datos pretende ser una muestra representativa de la comunidad otaku de Internet para el análisis demográfico y las tendencias dentro de este grupo. Contiene información sobre usuarios (género, ubicación, fecha de nacimiento, etc.), sobre anime (fecha de emisión, géneros, productor…) y listas de anime.

2 – Objetivo

El objetivo de este proyecto es tener un catálogo ordenado de los animes contenidos en el dataset previamente comentado. Con el fin de poder sacar estadísticas reales para la comunidad.

3 – Situación Problemática

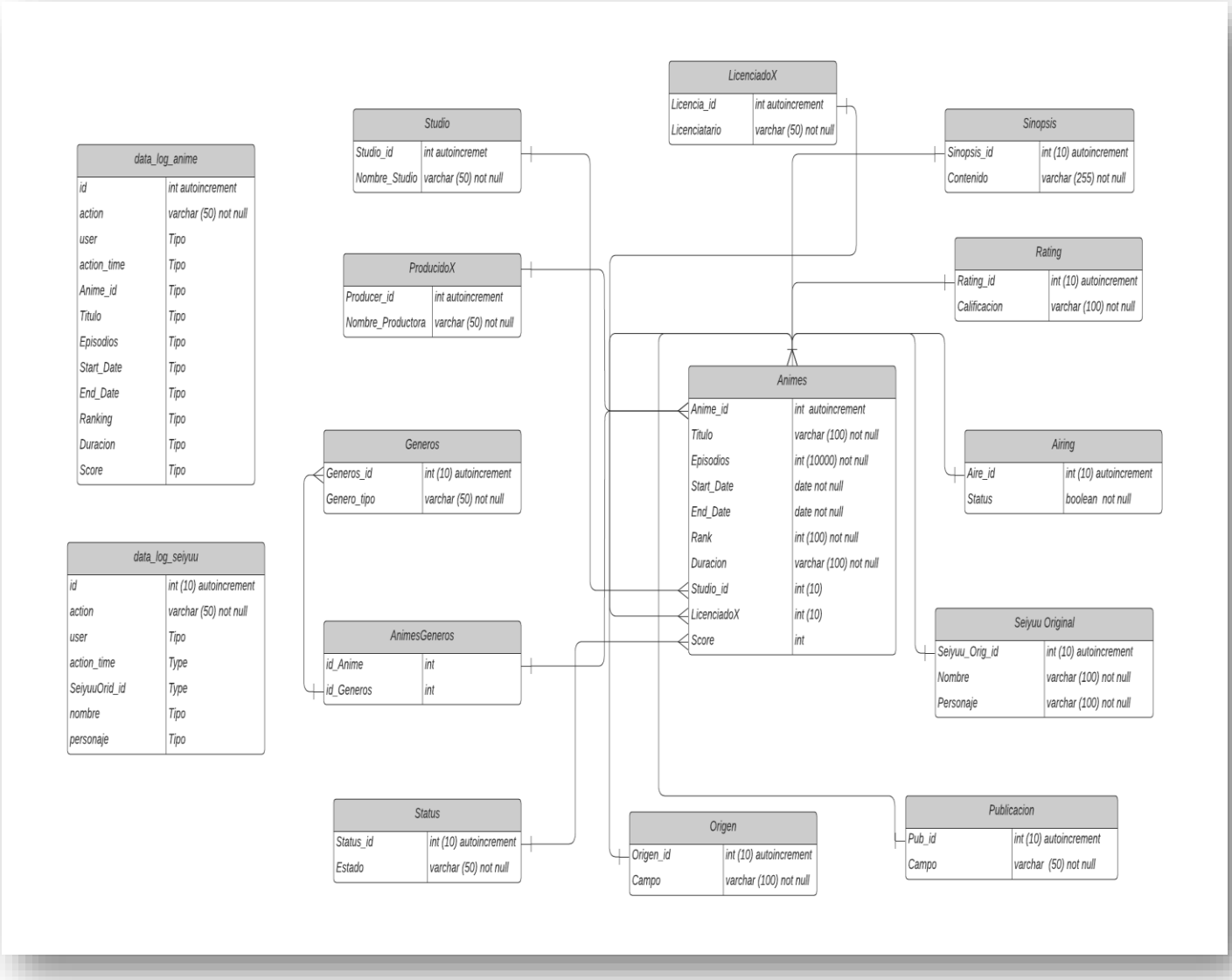
El dataset de MyAnimeList cuenta con más de 9GB de datos almacenados. Los mismos necesitan ser normalizados para su correcto uso.

El proyecto busca generar un set de datos utilizable en forma de POC (proof of concetp) para luego llevarlo a gran escala. Creando nuevas tablas y relaciones.

4 – Modelo de Negocios

Esta base de datos será implementada y utilizada por la comunidad argentina de anime para la organización y catálogo de los animes vistos por comunidad.

5 – Diagrama E.R



6 – Listado de tablas con descripción de estructura

6 – 1 Tabla Animes.

Esta tabla contiene los nombres de los Animes que se van agregando constantemente.

Animes	
Anime_id	*int autoincrement*
Titulo	*varchar (100) not null*
Episodios	*int (10000) not null*
Start_Date	*date not null*
End_Date	*date not null*
Rank	*int (100) not null*
Duracion	*varchar (100) not null*
Score	*int*
LicenciadoX	*int*

6 – 1.1 Campos que componen la tabla:

Anime_id: es la clave primaria e indica el id del anime.
Título: Nombre del anime.
Episodios: Cantidad de episodios emitidos.
Start_Date: Fecha de inicio de emisión.
End_Date: Fecha de término de emisión.
Rank: Ranking publico.
Duración: total de minutos que dura la emisión.
LicenciadoX: Nombre del licenciatario.
Score: Score generado por la comunidad.

PK	COLUMN	TYPE	NOT NULL	UNIQUE	LEN
TRUE	Anime_id	INT	TRUE	TRUE	
	Titulo	VARCHAR	TRUE		100
	Episodios	INT	TRUE		10000
	Start_Date	date	TRUE		
	End_Date	date	TRUE		
	Rank	int	TRUE		100
	Duracion	VARCHAR	TRUE		100
	LicenciadoX	INT			10

6 -2 Tabla Géneros.

Esta tabla contiene los géneros (drama-acción-etc) que pueden tener los distintos animes.

Generos	
Generos_id	*int (10) autoincrement*
Genero_tipo	*varchar (50) not null*

6 – 2.1 Campos que contienen las tablas:

Generos_id: clave primaria y id de los géneros.
Generos_tipo: Distintos tipo de genero.

PK	COLUMN	TYPE	NOT NULL	UNIQUE	LEN	NOTES
TRUE	Generos_id	INT	TRUE	TRUE		
	Generos_tipo	VARCHAR	TRUE		50	

6 – 3 Tabla Status.

Esta tabla nos comenta si continua o no el anime.

Status	
Status_id	*int (10) autoincrement*
Estado	*varchar (50) not null*

6 – 3.1 Campos que contienen las tablas:

Status_id: clave primaria y contiene el id del status.
Estado: Distintos tipos de estado.

PK	COLUMN	TYPE	NOT NULL	UNIQUE	LEN
TRUE	Status_id	INT	TRUE	TRUE	
	Estado	VARCHAR	TRUE		50

6 – 4 Tabla Origen.

Esta tabla nos comenta si es una novela, manga…

Origen	
Origen_id	*int (10) autoincrement*
Campo	*varchar (100) not null*

6 – 4.1 Campos que contienen las tablas.

Origen_id: clave primaria que contiene el id de origen.
Campo: nos comenta si es novela-manga-original.

PK	COLUMN	TYPE	NOT NULL	UNIQUE	LEN
TRUE	Origen_id	INT	TRUE	TRUE	
	Origen	VARCHAR	TRUE		100

6 – 5 Tabla Publicación.

Esta tabla nos muestra si es un OVA-TV-Movie

Publicacion	
Pub_id	*int (10) autoincrement*
Campo	*varchar (50) not null*

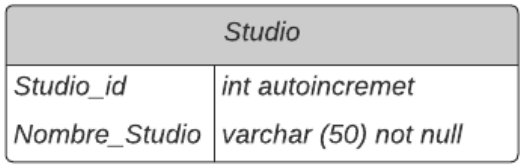
6 – 5 – 1 Campos que contienen las tablas:

Pub_id: clave primaria que contiene el id.
Campo: nos indica si es un OVA o un reléase para TV.

PK	COLUMN	TYPE	NOT NULL	UNIQUE	LEN
TRUE	Pub_id	INT	TRUE	TRUE	
	Publicacion	VARCHAR	TRUE		50

6 – 6 Tabla Studio.

Esta tabla tienen los Estudios encargados de la producción.



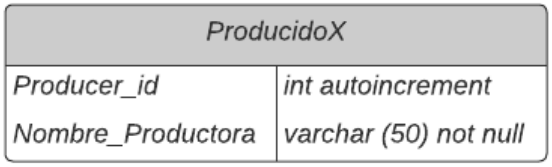
6 – 6.1 Campos que contienen las tablas:

Studio_id: Clave primaria.
Nombre_Studio: nombre de los estudios.

PK	COLUMN	TYPE	NOT NULL	UNIQUE	LEN
TRUE	Studio_id	INT	TRUE	TRUE	
	Nombre_Studio	VARCHAR	TRUE		50

6 – 7 Tabla ProducidoX.

Esta tabla contiene los nombres de las productoras.



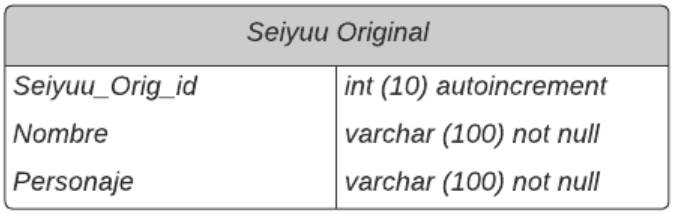
6 – 7.1 Campos que contienen las tablas:

Producer_id: clave primaria.
Nombre_Productora: Nombres de las productoras.

PK	COLUMN	TYPE	NOT NULL	UNIQUE	LEN
TRUE	Producer_id	INT	TRUE	TRUE	
	Nombre_Productora	VARCHAR	TRUE		50

6 – 8 Tabla SeiyuuOriginal.

Esta tabla contiene los nombres de los actores de voz.



6 – 8.1 Campos que contienen la tabla:

Seiyuu_Orig_id: clave primaria.
Nombre: Nombre del actor.
Personaje: Nombre del personaje.

PK	COLUMN	TYPE	NOT NULL	UNIQUE	LEN
TRUE	Seiyuu_Orig_id	INT	TRUE	TRUE	
	Nombre	VARCHAR	TRUE		100
	Personaje	VARCHAR	TRUE		100

6 – 9 Tabla Airing.

Tabla que nos indica si esta al aire o no.

Airing	
Aire_id	*int (10) autoincrement*
Status	*boolean not null*

6 – 9.1 Campos que contienen la tabla:

Aire_id: Clave primaria.

Status: Si o No está al aire.

PK	COLUMN	TYPE	NOT NULL	UNIQUE	LEN
TRUE	Aire_id	INT	TRUE	TRUE	
	Status	BOOLEAN	TRUE		

6 – 10 Tabla Rating.

Esta tabla nos muestra el rating que tiene este anime.

Rating	
Rating_id	*int (10) autoincrement*
Calificacion	*varchar (100) not null*

6 – 10.1 Campos que contiene la tabla:

Rating_id: clave primaria .
Calificación: ranking generado x la comunidad.

PK	COLUMN	TYPE	NOT NULL	UNIQUE	LEN
TRUE	Rating_id	INT	TRUE	TRUE	
	Calificacion	VARCHAR	TRUE		100

6 – 11 Tabla Sinopsis.

Esta tabla contiene una breve reseña del anime.

Sinopsis	
Sinopsis_id	*int (10) autoincrement*
Contenido	*varchar (255) not null*

6 – 11.1 Campos que contiene la tabla:

Sinopsis_id: clave primaria.
Contenido: breve reseña de cada anime.

PK	COLUMN	TYPE	NOT NULL	UNIQUE	LEN
TRUE	Sinopsis_id	INT	TRUE	TRUE	
	Contenido	VARCHAR	TRUE		254

7 – Script de creación de objetos de la DB e inserción de datos

El script de creación de la DB y objetos y datos se encuentra guardado en la web de Github.

Para poder utilizarlo uno debe descargarse el archivo *.sql El que le generara automáticamente toda la base. Como así también las vistas, Stored Procedures, Funciones y usuarios necesarios para el correcto funcionamiento.

La url para bajar dicho archivo es la siguiente:

https://github.com/tonga2508/Coder-SQL/blob/main/TP21072022

Para poder correr correctamente todas las funciones de la base uno debe tener creado el siguiente usuario. De todas maneras, en el script ya se encuentra añadido.

```
CREATE DATABASE  IF NOT EXISTS `codertp` /*!40100 DEFAULT CHARACTER SET utf8mb4 COLLATE utf8mb4_0900_ai_ci */ /*!80016 DEFAULT ENCRYPTION='N' */;
USE `codertp`;
CREATE USER IF NOT EXISTS Gaston IDENTIFIED BY 'colegio123';
GRANT ALL PRIVILEGES ON codertp.* TO 'Gaston'@'%';
```

También uno puede descargarse la base por partes. Es decir, uno puede bajar el archivo que contiene solamente la estructura de la DB y por otro lado puede bajar el archivo con la inserción de datos. Las url son las siguientes:

https://github.com/tonga2508/Coder-SQL/blob/main/Estructura
https://github.com/tonga2508/Coder-SQL/blob/main/DataOnly

8 – Script de inserción de datos

La inserción de datos también se encuentra en el mismo archivo que el anterior. Así que al correr ese script uno cuenta con la DB totalmente pupulada y lista para ser usada.

9 – Informes Generados

Descripción de Stored Procedures, Functions, Views y Triggers

Stored Procedures:
Para agregar 1 anime:
call insert_Anime(62,'Jaspion','555','2009-10-22','2010-11-11','10','24 min. per ep.','8');
Para borrar 1 anime especificando id:
call Borrar_Anime(60);
Al realizar estas consultas se genera un trigger que updatea la tabla data_log_anime.
Existen otros 2 SP que realizan 2 búsquedas:
selector realiza búsqueda por anime_id
La consulta para el anime con id 30 es: call selector(30);
El segundo es selector2 que realiza una búsqueda por género (acción-drama-etc).
La consulta para utilizarlo para buscar por acción es: call selector2(acción);

Funciones:
Para calcular el resultado de la hipotenusa usando la función Pitágoras: select pitagoras(2,2) as ResultHipotenusa;
Donde (2,2) hace referencia a (alto,largo) se pueden utilizar distintos valores.

Para utilizar la función Fu_AnimeLevel se debe utilizar la vista: animelevel de la siguiente manera: SELECT * FROM codertp.animelevel;
La misma le otorga un valor (platinum-gold-silver) según el ranking.

Views:
100caps está view nos da como resultado los anime con más de100 capítulos.
animelevel está view utiliza la función FU_AnimeLevel para darles distintos valores según el ranking.
generosytitulos muestra el título del anime con su correspondiente género.
ranking muestra el título y el ranking.
sinopsys muestra el título y sinopsis correspondiente.
sinopyreating nos muestra título, sinopsis y su calificación.

Triggers:
Se crearon 6 triggers. Los mismos funcionan de la siguiente manera:
Los primeros 3 se encuentran en la tabla animes. El primero utiliza un before y los restantes un AFTER
Estos escriben en la tabla data_log_anime. Su función es dejar logueado la creación, borrado y update

en la tabla anime.
Los 3 restantes tienen la misma función que los anteriores, pero escriben en la tabla data_log_seiyuu y
se encuentran en la tabla seiyuuoriginal.

Creación de Usuarios Sentencias sublenguaje TCL

Creación de usuarios:

Usuario Con permisos para crear, insertar, updatear y seleccionar:

```
CREATE USER IF NOT EXISTS Userd1 IDENTIFIED BY 'Popurri1';
GRANT create, insert, select, update ON codertp.* TO 'Userd1'@'%';
```

Usuario read-only:

```
CREATE USER IF NOT EXISTS Userd2 IDENTIFIED BY 'Popurri2';
GRANT select, show view ON codertp.* TO 'Userd2'@'%';
```

En ninguno de los 2 casos estas cuentas pueden borrar.

SENTENCIAS DEL SUBLENGUAJE TCL

Insertamos los siguientes registros en la tabla anime:

```
INSERT INTO `animes` VALUES (60,'Death Note',37,'2006-10-04','2007-06-27',51,'23 min. per ep.',9),
(61,'Death Note',37,'2006-10-04','2007-06-27',51,'23 min. per ep.',9),
(62,'Death Note',37,'2006-10-04','2007-06-27',51,'23 min. per ep.',9),
(63,'Death Note',37,'2006-10-04','2007-06-27',51,'23 min. per ep.',9);
```

Podemos utilizar la siguiente consulta para estar seguros que se sumaron esos 4 registros:

```
SELECT * FROM animes order by Anime_id limit 70;
```

Ahora vamos a iniciar una transacción donde borraremos los últimos 4 registros:

```
START TRANSACTION;
DELETE FROM animes WHERE Anime_id in (60,61,62,63);
```

Ahora utilizando la siguiente sentencia vamos a recuperar esos 4 registros borrados y luego commiteamos:

```
rollback;
commit;
```

Utilizamos la siguiente sentencia para ver lo comentado anteriormente:

```
SELECT * FROM animes order by Anime_id limit 70;
```

Acá iniciamos una transacción, luego sumamos 4 registros y creamos un savepoint. Luego creamos otros 4 y volvemos a crear otro savepoint. Al final nos deshacemos el savepoint insert1.

```
Start transaction;
INSERT INTO `animes` VALUES (64,'Death Note',37,'2006-10-04','2007-06-27',51,'23 min. per ep.',9);
INSERT INTO `animes` VALUES (65,'Death Note',37,'2006-10-04','2007-06-27',51,'23 min. per ep.',9);
INSERT INTO `animes` VALUES (66,'Death Note',37,'2006-10-04','2007-06-27',51,'23 min. per ep.',9);
INSERT INTO `animes` VALUES (67,'Death Note',37,'2006-10-04','2007-06-27',51,'23 min. per ep.',9);
savepoint insert1;
INSERT INTO `animes` VALUES (68,'Death Note',37,'2006-10-04','2007-06-27',51,'23 min. per ep.',9);
INSERT INTO `animes` VALUES (69,'Death Note',37,'2006-10-04','2007-06-27',51,'23 min. per ep.',9);
```

```
INSERT INTO `animes` VALUES (70,'Death Note',37,'2006-10-04','2007-06-27',51,'23 min. per ep.',9);
INSERT INTO `animes` VALUES (71,'Death Note',37,'2006-10-04','2007-06-27',51,'23 min. per ep.',9);
savepoint insert2;
release savepoint insert1;
```

10 – Tecnologías y Herramientas utilizadas.

Las herramientas que se utilizaron para llevar a cabo este proyecto son las siguientes:

- Mysql Workbench para elaborar la DB como asi también los scripts y demás funciones que la componen.
- Microsoft Excel para manipulación de datos.
- Microsoft Word para redactar este T.P.
- NotePad ++ para manipulación de datos.
- LucidApp para generar el modelo de E.R.
- Zoom para las clases de Coder.
- GitHub para guardar los scripts de creación y población de la DB.