Proyecto de diseño de una

Base de Datos

Servicio de video streaming



Vlad Comanescu

Ruymán Rodríguez Martín

Diego Machín Guardia

Índice

1.Definición de la Base de Datos

2.Objetivos de la Base de Datos

3.Modelo Conceptual ER

4.Modelo Lógico Relacional

5.Modelo Lógico Objeto-Relacional(UML)

6.Scripts generados

7.Carga de datos

8.Triggers implementados

**1.Definición de la base de datos**

La nueva aplicación OAN nos ha pedido que implementemos su nueva base de datos para la puesta en marcha de su aplicación a la que la base de datos ha de dar soporte. OAN es una empresa de reproducción de vídeo en streaming, en la que se alojan películas, series, documentales… similar a otras plataformas del mercado como Netflix,HBO,Hulu,etc. En la reunión con los directivos de la empresa se ha llegado a las siguientes ideas que se han de llevar a cabo:

La base de datos ha de contener los distintos usuarios que usarán la plataforma para tener a cada uno identificado. Además, es necesario que se guarde la información personal de los usuarios para poder realizar la facturación así como ponerse en contacto o elegir los siguientes títulos a añadir al servicio en función del tipo de perfil de los usuarios.

La aplicación obtendrá sus beneficios en función del pago de los usuarios por el servicio prestado, es por ello que ofrecerá distintos tipos de servicios o suscripciones, por lo que cada usuario tendrá que tener asociado un tipo de suscripción que podrá cambiar a lo largo del tiempo y que tiene que ser pagada. Los cobros se permitirán mensuales o anuales en función de la oferta que se elija qué será de una cuantía diferente en función del tipo de suscripción que se elija entre las disponibles (básica,premium). Se encuentra también una opción disponible de pago bajo demanda en la cual se paga por cada título que se vea de la página por lo que los pagos se realizarán antes de comenzar cada reproducción.

Las suscripciones traerán asociadas un número de perfiles disponibles y cada uno de los perfiles consta como un usuario diferente con sus propias listas, recomendaciones y amigos lo que es una gran ventaja ya que se puede compartir una cuenta entre varias personas o crear una cuenta familiar.

Otro servicio que se les ofrece a los usuarios es el marcado de títulos, es decir, los usuarios han de poder ser capaces de marcar los distintos títulos para añadirlos a su lista de favoritos o pendientes, consultar el historial de sus reproducciones para poder continuar con su título favorito el cual no acabaron de ver o que quieren recordar el título. Finalmente también disponen de la lista de reproducción donde se encuentran los títulos empezados pero que no se han llegado a finalizar, los títulos finalizados se incluyen en la categoría de vistos así se puede llevar un control de lo que se ha visto en caso de que se tenga duda de si se ha visto o no un título o por qué capítulo de la serie se encuentra el perfil.

Los usuarios podrán compartir en sus redes sociales los títulos que han visto así como recomendar a amigos títulos por medio de éstas, es por ello que los perfiles de usuario se pueden conectar con facebook o con twitter. Los usuarios también podrán dejar críticas o comentarios de lo que les ha parecido una película para ayudar a elegir a otros usuarios que estén consultando el título o que quieran compartir opiniones al finalizar la reproducción.

Por otro lado la plataforma ha de almacenar los títulos que contiene y el número de referencia para referenciarlo así como toda la información del título, un enlace para poder reproducirlo dentro de la plataforma. Los títulos podrán ser de distinto tipo como películas, series y documentales y las series estarán contenidas en temporadas con los capítulos disponibles para su visualización. Cada capítulo se podrá visualizar por separado y debido a que esta plataforma busca la expansión y abrirse a nuevos mercados internacionales tendrá cada título en varios idiomas y con subtítulos en varios idiomas. También para la realización de un servicio eficaz y pensando en el usuario la plataforma quiere ofertar distintas calidades que podrán ser escogidas para mejorar la velocidad de reproducción del usuario en caso de que este disponga de mala conexión por lo que los títulos estarán disponibles en varias calidades. Además de que se ha de almacenar la información básica de cada título que será el nombre, la categoría, el año y una breve descripción. Los actores más relevantes de cada película también tendŕan que ser almacenados para poder hacer búsquedas en base a un actor accediendo a todas sus películas así como buscar las películas más famosas de un actor.

Los usuarios han de poder buscar los títulos en función a la categoría que quieran acceder y poder realizar el filtrado de los títulos en función del tipo (pelicula,serie o documental). A la hora de las búsquedas así como en la visualización de los títulos se ha de tener en cuenta la clasificación de la película y la edad del perfil de usuario para asegurar que el usuario puede reproducir la película y no mostrarla en caso de que no deba ser visualizada.

La plataforma ofrece un servicio de recomendaciones por lo que el usuario recibirá recomendaciones tanto en función de los títulos vistos y la información que se haya realizado tanto como recomendaciones que recibirá de otros usuarios, en una lista de títulos privada.

En la plataforma existirán un tipo de usuario que serán los empleados, éstos tendrán que estar a parte y podrán acceder a la información de los usuarios. Su trabajo consistirá en solucionar los problemas técnicos de los reportes de los usuarios así como la comprobación de que todos los usuarios están a la orden de pago. También podrán acceder al ranking de películas más vistas y de las categorías más vistas para poder elegir los siguiente títulos a introducir en la plataforma. A la hora de introducir nuevos títulos se introducirán también en la lista de recién añadidos que contendrá las últimas 10 adquisiciones. Por el contrario los títulos que vayan a ser eliminados de la página han de mostrar su fecha de eliminación para que el usuario sepa que van a dejar de estar disponibles y los visualice antes de la fecha de expiración.

Muchos de los títulos de la plataforma están disponibles en servicio bajo descarga, es decir un usuario puede tenerlos descargados para su visualización sin conexión, pero para ello habŕa que comprobar el número de descargas actuales y el número de descargas permitidas dado por el tipo de suscripción. Estas descargas tienen una fecha de caducidad ya que no pueden estar de manera indefinida descargadas por el usuario, una vez se haya realizado la visualización completa o se haya pasado el tiempo límite el título descargado será eliminado.

**2.Objetivo de la base de datos**

El objetivo principal de la base de datos es ofrecer un medio de almacenamiento de los diferentes usuarios que van a utilizar la aplicación de video streaming por una parte, y por otra parte tener una base de datos en donde se almacenen todos los datos relacionados con el contenido de la página en sí.Con esto se intenta que se le de cierta consistencia al funcionamiento general de nuestra aplicación y mantener un cierto registro de todas los procesos/transacciones que ocurren en nuestro aplicativo.

Todos los resultados del proyecto, los que aparecen en este documento también se pueden consultar en el siguiente repositorio:[Proyecto OAN](https://github.com/Zanuro/BDD)

**3.Modelo Conceptual ER**

Se ha creado y mejorado el modelo conceptual Entidad-relación de nuestra supuesta base de datos siguiendo las directrices y consejos dados tras la primera revisión:

[**Modelo Conceptual ER OAN**](https://github.com/Zanuro/BDD/blob/master/OAN/E-R-OAN.png)

**4.Modelo Lógico Relacional**

Luego hemos pasado al diseño del modelo Lógico-Relacional,utilizado para el modelado y gestión de la base de datos.En este modelo podemos identificar las diferentes relaciones entre las distintas tablas de nuestra base de datos,identificando en cada una de las tablas los atributos que debería tener,el tipo de estos atributos,y las relaciones que existen entre las distintas tablas,es decir,hemos identificado aquellos atributos que deberían actuar como clave primaria,clave foránea,aquellos atributos que son atributos multivaluados,atributos calculados,etc.

[**Modelo Lógico Relacional-OAN**](https://github.com/Zanuro/BDD/blob/master/OAN/OAN_logico_relacional.png)

**5.Modelo Lógico Objeto-Relacional(UML)**

**6.Scripts generados**

El script generado a partir del modelo Lógico Relacional diseñado en SQL-Workbench y generado para que sea compatible con PostgreSQL:

[Script creación base de datos y triggers.](https://github.com/Zanuro/BDD/blob/master/OAN/OAN.sql)

En este se han creado las diferentes tablas anteriormente descritas con sus atributos.También se han definido las diferentes relaciones(referenciales entre las distintas tablas), restricciones(restricciones sobre la actualización,borrado de los datos),etc.

Por último, se han definido diferentes triggers explicados más en detalle en el último apartado.

**7.Carga de datos**

Tras la creación de la base de datos hemos continuado introduciendo algunos datos en la base de datos para que luego podamos comprobar su correcto funcionamiento.Cabe destacar que estos son unos datos ficticios.

[Carga de datos en la base de datos](https://github.com/Zanuro/BDD/blob/master/OAN/OAN_datos.sql)

**8.Triggers implementados**

Hemos definido algunos triggers implementados para el buen y correcto funcionamiento de la base de datos.

Los diferentes triggers implementados estarán en el script de creación de la base de datos a partir de la línea 521:

[Triggers implementados-Línea 521.](https://github.com/Zanuro/BDD/blob/master/OAN/OAN.sql)

Realizaremos las pruebas en orden alfabético de la tabla que es utilizada.

**Triggers sobre la tabla descargas:**

- Impedir que el total de descargas de descargas realizado por todos los perfiles de un usuario supere el número de descargas máximas permitidas para este.

- Al insertar/editar/eliminar actualizar el número de descargas activas del perfil

**Trigger sobre la tabla perfil:**

- Impedir asignar más perfiles de los que permite la suscripción actual del usuario.

Premium y Contenido: 4. Básica: 1

**Trigger sobre la tabla perfil\_comenta\_titulo:**

**-** Impedir que se comente un título que no haya sido visualizado

**Trigger sobre la tabla perfil\_perfil:**

- El perfil no puede ser amigo de sí mismo

**Triggers sobre la tabla perfil\_visualiza\_titulo:**

- Al insertar/modificar un registro comprueba si el momento\_actual coincide aproximadamente con la duración del título. Si es así, lo marca como visto.

- Al marcar un título como visto, si está en descargas del usuario, eliminarlo.

**Trigger sobre la tabla suscripción:**

- Impedir que un usuario tenga más de 1 suscripción activa

**Trigger sobre la tabla título:**

- Impedir la modificación del tipo de un título, si anteriormente era serie y aún sigue asociada a una serie.

**Trigger sobre la tabla título\_serie:**

- Comprobar que el título que se desea asociar a una serie es del tipo serie.

Las pruebas realizadas se hicieron con el propósito de comprobar el correcto funcionamiento de los distintos triggers y se realizarán en el orden descrito anteriormente.Es importante que se cargue el fichero de datos y estos datos no se modifiquen.

[Pruebas sobre el funcionamiento de la base de datos](https://github.com/Zanuro/BDD/blob/master/OAN/OAN_pruebas.sql)

En la primera prueba:

Tenemos el usuario antonio.gutierrez1984@gmail.com con 2 descargas activas.Cada cuenta tiene un número máximo de 2 descargas independientemente del número de perfiles que tenga.Por lo tanto si intentamos añadir otra descarga activa asociada a uno de los perfiles de esta cuenta nos debería dar un error:

|  |
| --- |
| Insertandolo con su perfil 1: |
| INSERT INTO descargas (idTitulo, fecha\_caducidad, idPerfil) VALUES (1, CURRENT\_DATE + 1, 1); |
|  |

|  |
| --- |
| Insertandolo con su perfil 2: |
| INSERT INTO descargas (idTitulo, fecha\_caducidad, idPerfil) VALUES (1, CURRENT\_DATE + 1, 2); |
|  |

Vemos como el trigger comprueba que el usuario no puede tener otra descarga activa y salta el trigger.

En la segunda prueba:

En esta prueba hemos probado insertar/actualizar/borrar el número de descargas activas de un perfil.

Obtener el número de descargas de un perfil:

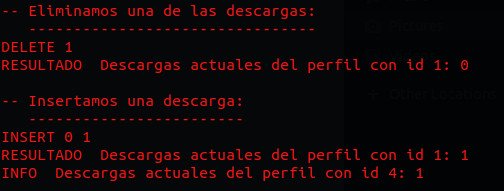
|  |  |
| --- | --- |
|  |  |
| SELECT num\_descargas\_actuales AS ndescargas FROM perfil WHERE idperfil=1  \echo 'INFO Descargas actuales del perfil con id 1:' :ndescargas |  |
|  |  |
|  |  |
| Resultado: |  |



Borrar o insertar descargas:

|  |
| --- |
| \echo '-- Eliminamos una de las descargas:' |
| \echo ' --------------------------------' |
| SELECT idTitulo AS titulo FROM descargas WHERE idperfil=1 LIMIT 1 |
| \gset |
| DELETE FROM descargas WHERE idTitulo=:titulo AND idPerfil = 1; |
| SELECT num\_descargas\_actuales AS ndescargas FROM perfil WHERE idperfil=1 |
| \gset |
| \echo 'RESULTADO Descargas actuales del perfil con id 1:' :ndescargas |
| \echo |
| \echo '-- Insertamos una descarga:' |
| \echo ' ------------------------' |
| INSERT INTO descargas (idTitulo, fecha\_caducidad, idPerfil) VALUES (3, CURRENT\_DATE + 1, 1); |
| SELECT num\_descargas\_actuales AS ndescargas FROM perfil WHERE idperfil=1 |
| \gset |
| \echo 'RESULTADO Descargas actuales del perfil con id 1:' :ndescargas |
| SELECT num\_descargas\_actuales AS ndescargas FROM perfil WHERE idperfil=4 |
| \gset |
| \echo 'INFO Descargas actuales del perfil con id 4:' :ndescargas |

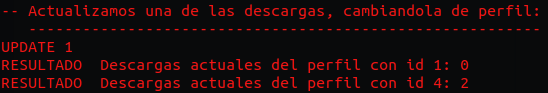
Resultado:



Por último. probamos las actualización cambiando el perfil de la descarga:

|  |
| --- |
| \echo '-- Actualizamos una de las descargas, cambiandola de perfil:' |
| \echo ' ---------------------------------------------------------' |
| SELECT idTitulo AS titulo FROM descargas WHERE idperfil=1 LIMIT 1 |
| \gset |
| UPDATE descargas SET idPerfil=4 WHERE idTitulo=:titulo AND idPerfil=1; |
| SELECT num\_descargas\_actuales AS ndescargas FROM perfil WHERE idperfil=1 |
| \gset |
| \echo 'RESULTADO Descargas actuales del perfil con id 1:' :ndescargas |
| SELECT num\_descargas\_actuales AS ndescargas FROM perfil WHERE idperfil=4 |
| \gset |
| \echo 'RESULTADO Descargas actuales del perfil con id 4:' :ndescargas |
| \echo |

Resultado:

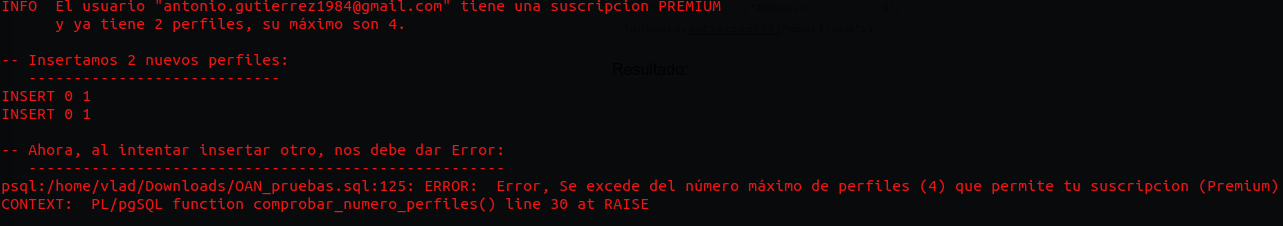


En la tercera prueba:

En esta prueba se impide que una cuenta tenga más perfiles asignados de los permitidos:Max. Premium y Contenido: 4. Básica: 1

|  |
| --- |
|  |
| \echo ' PERFIL: Impedir asignar más perfiles de los que permite la suscripción actual del usuario.' |
|  |
| \echo 'INFO El usuario "antonio.gutierrez1984@gmail.com" tiene una suscripcion PREMIUM' |
| \echo ' y ya tiene 2 perfiles, su máximo son 4.' |
| \echo |
| \echo '-- Insertamos 2 nuevos perfiles:' |
| \echo ' ----------------------------' |
| INSERT INTO perfil (nombre, email) VALUES ('Antonio 2', 'antonio.gutierrez1984@gmail.com'); |
| INSERT INTO perfil (nombre, email) VALUES ('Antonio 3', 'antonio.gutierrez1984@gmail.com'); |
| \echo |
| \echo '-- Ahora, al intentar insertar otro, nos debe dar Error:' |
| \echo ' -----------------------------------------------------' |
| INSERT INTO perfil (nombre, email) VALUES ('Antonio 4', 'antonio.gutierrez1984@gmail.com'); |

Resultado:

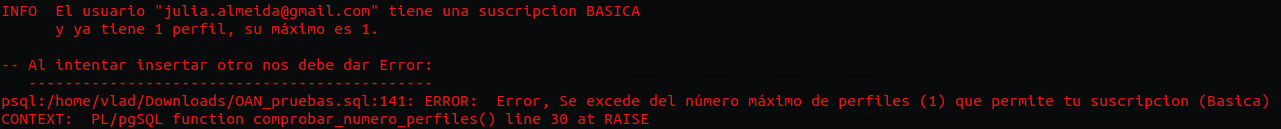


El usuario tenía 2 perfiles de un máximo de cuatro,se han introducido otros 2 y al intentar introducirse otros salta el trigger.

Otro ejemplo:

|  |
| --- |
| \echo 'INFO El usuario "julia.almeida@gmail.com" tiene una suscripcion BASICA' |
| \echo ' y ya tiene 1 perfil, su máximo es 1.' |
| \echo |
| \echo '-- Al intentar insertar otro nos debe dar Error': |
| \echo ' ---------------------------------------------' |
| INSERT INTO perfil (nombre, email) VALUES ('Julia 2', 'julia.almeida@gmail.com'); |

Resultado:

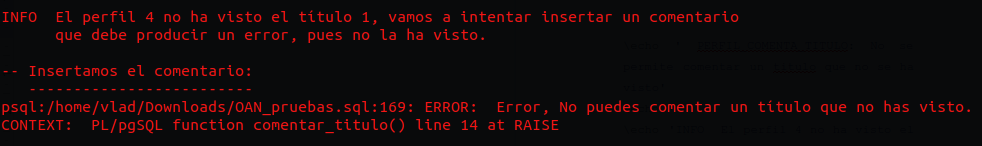


Cuarta prueba:

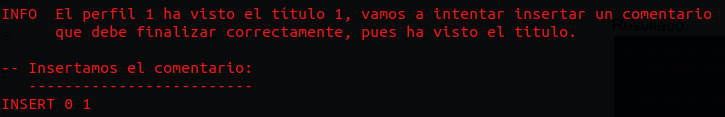
En esta prueba vamos a impedir que un usuario pueda comentar un título que no haya visto.

|  |
| --- |
|  |
| \echo ' PERFIL\_COMENTA\_TITULO: No se permite comentar un titulo que no se ha visto' |
| \echo |
| \echo 'INFO El perfil 4 no ha visto el título 1, vamos a intentar insertar un comentario' |
| \echo ' que debe producir un error, pues no la ha visto.' |
| \echo |
| \echo '-- Insertamos el comentario:' |
| \echo ' -------------------------' |
| INSERT INTO perfil\_comenta\_titulo (idPerfil, idTitulo, fecha, comentario) |
| VALUES (4, 1, CURRENT\_TIMESTAMP, 'No me ha gustado nada la película'); |

Resultado:



Otro ejemplo:

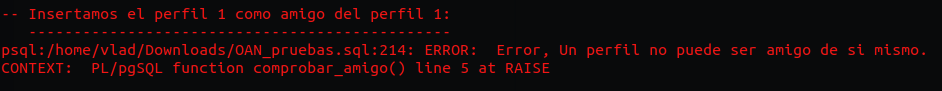


Quinta prueba:

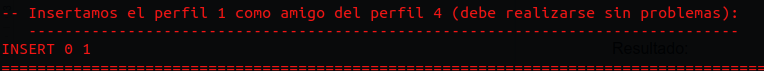
En esta prueba vamos a impedir que un perfil sea amigo de sí mismo.

|  |
| --- |
|  |
| \echo ' PERFIL\_PERFIL: No se permite que un perfil sea amigo de si mismo' |
| \echo |
| \echo '-- Insertamos el perfil 1 como amigo del perfil 1:' |
| \echo ' -----------------------------------------------' |
| INSERT INTO perfil\_perfil (idPerfil, idPerfilAmigo) VALUES (1, 1); |

Resultado:



Otro ejemplo:

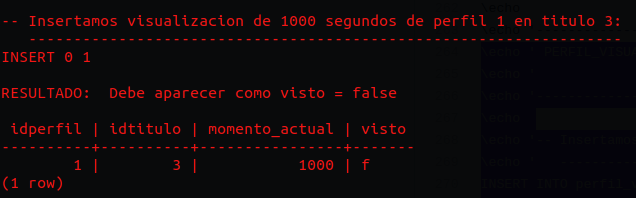


Sexta prueba:

En esta prueba hacemos que un usuario que haya visto un titulo, y la duracion del titulo visto es aproximadamente igual que la duracion del titulo se marca como visto ese titulo en ese perfil.

|  |
| --- |
| \echo ' PERFIL\_VISUALIZA\_TITULO: Si el momento actual coincide aproximadamente con' |
| \echo ' la duración del título, se marca como visto. ' |
| \echo '---------------------------------------------------------------------------' |
| \echo |
| \echo '-- Insertamos visualizacion de 1000 segundos de perfil 1 en titulo 3:' |
| \echo ' ------------------------------------------------------------------' |
| INSERT INTO perfil\_visualiza\_titulo (idPerfil, idTitulo, momento\_actual) |
| VALUES (1, 3, 1000); |
| \echo |
| \echo 'RESULTADO: Debe aparecer como visto = false' |
| \echo |
| SELECT \* FROM perfil\_visualiza\_titulo WHERE idPerfil=1 AND idTitulo=3; |

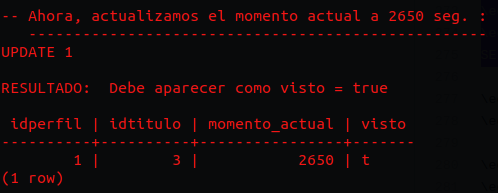
Resultado:



Y luego modificamos el tiempo:

|  |
| --- |
|  |
| \echo '-- Ahora, actualizamos el momento actual a 2650 seg. :' |
|  |
| UPDATE perfil\_visualiza\_titulo SET momento\_actual = 2650 WHERE idPerfil=1 AND idTitulo=3; |
| \echo |
| \echo 'RESULTADO: Debe aparecer como visto = true' |
| \echo |
| SELECT \* FROM perfil\_visualiza\_titulo WHERE idPerfil=1 AND idTitulo=3; |

Resultado:

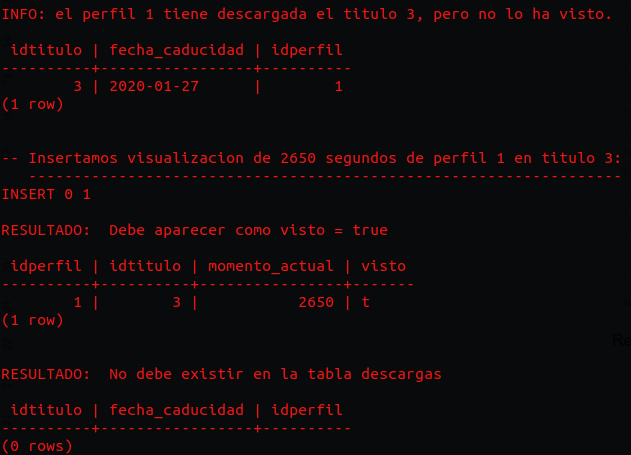


Séptima prueba:

En esta prueba comprobamos que si un usuario tiene marcado un título como visto pero lo tiene en descargas este título se elimina.

|  |
| --- |
| \echo ' -- PERFIL\_VISUALIZA\_TITULO: Si se marca un título como visto y el usuario lo tiene en descargas, se elimina.' |
| \echo |
| \echo 'INFO: el perfil 1 tiene descargada el titulo 3, pero no lo ha visto.' |
| \echo |
| SELECT \* FROM descargas WHERE idPerfil=1 AND idTitulo=3; |
| \echo |
| \echo '-- Insertamos visualizacion de 2650 segundos de perfil 1 en titulo 3:' |
| \echo ' ------------------------------------------------------------------' |
| INSERT INTO perfil\_visualiza\_titulo (idPerfil, idTitulo, momento\_actual) |
| VALUES (1, 3, 2650); |
| \echo |
| \echo 'RESULTADO: Debe aparecer como visto = true' |
| \echo |
| SELECT \* FROM perfil\_visualiza\_titulo WHERE idPerfil=1 AND idTitulo=3; |
| \echo |
| \echo 'RESULTADO: No debe existir en la tabla descargas' |
| \echo |
| SELECT \* FROM descargas WHERE idPerfil=1 AND idTitulo=3; |

Resultado:



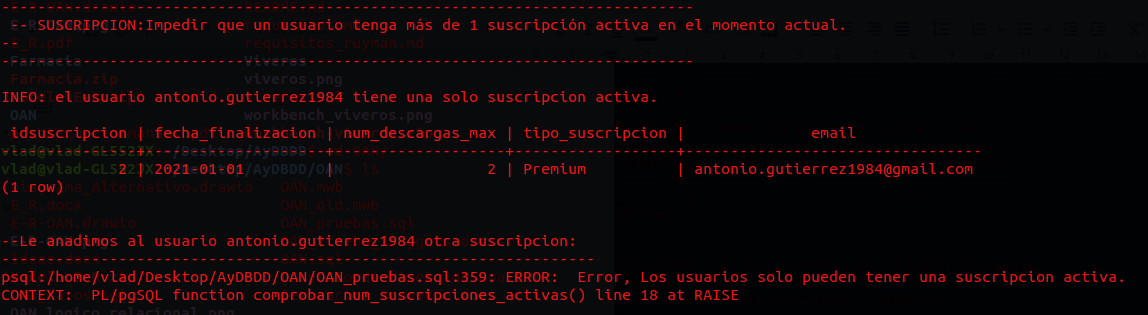
El perfil 1 tenia el titulo con id 3 en descargas pero al modificar la tabla perfil\_visualiza\_titulo añadiendo el tiempo y marcándose como visto se borra de la tabla descargas para ese perfil ese título.

Octava prueba:

En esta prueba tenemos que impedir tenga más de una suscripción activa en el momento actual.

|  |
| --- |
|  |
| \echo ' -- SUSCRIPCION:Impedir que un usuario tenga más de 1 suscripción activa en el momento actual. ' |
| \echo ' -- SUSCRIPCION:Impedir que un usuario tenga más de 1 suscripción activa en el momento actual. ' |
| \echo ' -- SUSCRIPCION:Impedir que un usuario tenga más de 1 suscripción activa en el momento actual. ' |
| \echo 'INFO: el usuario antonio.gutierrez1984 tiene una solo suscripcion activa.' |
| \echo |
| SELECT \* FROM suscripcion WHERE email='antonio.gutierrez1984@gmail.com' and fecha\_finalizacion >= CURRENT\_DATE; |
| \echo |
| \echo '--Le anadimos al usuario antonio.gutierrez1984 otra suscripcion:' |
| \echo '------------------------------------------------------------------' |
| INSERT INTO suscripcion (idsuscripcion, fecha\_finalizacion, num\_descargas\_max, tipo\_suscripcion, email) |
| VALUES (5, '2020-05-10', 1, 'Basica', 'antonio.gutierrez1984@gmail.com'); |

Resultado:



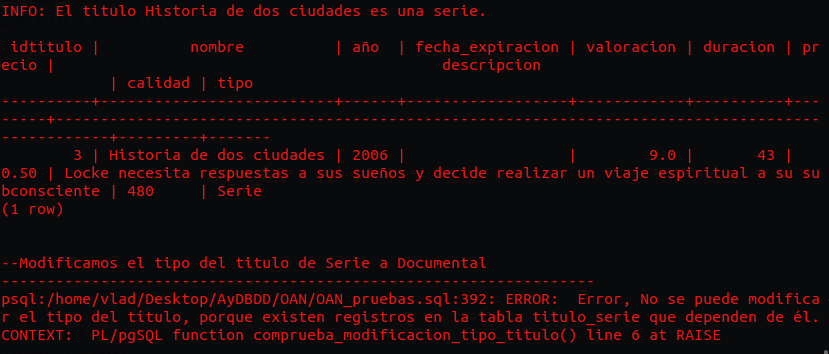
El usuario antonio.gutierrez1984 no puede tener más de una suscripción activa.

Novena prueba:

En esta prueba tenemos que impedir la modificación del tipo de un título, si anteriormente era serie y aún sigue asociada a una serie.

|  |
| --- |
|  |
| \echo ' -- TITULO:Impedir la modificación del tipo de un título, si anteriormente era serie y aún sigue asociada a una serie' |
| \echo |
| \echo 'INFO: El titulo Historia de dos ciudades es una serie.' |
| \echo |
| SELECT \* FROM titulo WHERE nombre='Historia de dos ciudades' and tipo='Serie'; |
| \echo |
| \echo '--Modificamos el tipo del titulo de Serie a Documental' |
| \echo '------------------------------------------------------------------' |
| UPDATE titulo SET tipo='Documental' WHERE nombre='Historia de dos ciudades' and tipo='Serie'; |

Resultado:

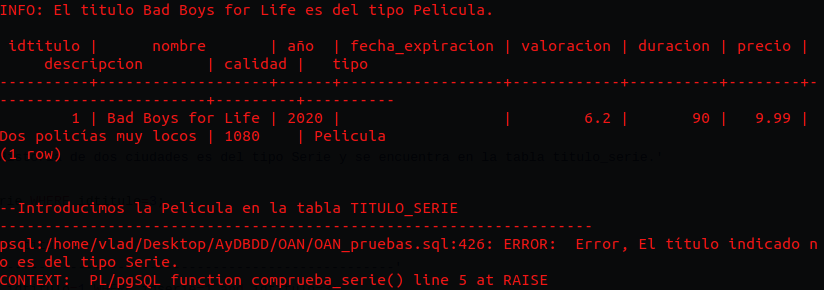


Décima prueba:

En esta prueba tenemos que comprobar que el título que se desea asociar a una serie es del tipo serie.

|  |
| --- |
|  |
| \echo ' -- TITULO\_SERIE:Comprobar que el título que se desea asociar a una serie es del tipo serie' |
| \echo 'INFO: El titulo Bad Boys for Life es del tipo Pelicula.' |
| \echo |
| SELECT \* FROM titulo WHERE nombre='Bad Boys for Life' and tipo='Pelicula'; |
| \echo |
| \echo '--Introducimos la Pelicula en la tabla TITULO\_SERIE' |
| \echo '------------------------------------------------------------------' |
| INSERT INTO titulo\_serie (idtitulo, capitulo, temporada, idserie) |
| VALUES (1, 1, 2, 2); |

Resultado:



Otro ejemplo(cambiando el idserie a otro idtitulo que no sea una Serie):

