Tarea 3

Creación y Modelación de una una Base de Datos para una Empresa de Servicio de Películas mediante Streaming

Grupo 12:

Sebastián Hernández

Franco Curotto
Pablo Polanco

CC3201 Bases de Datos
Sección 1
Profesor Jorge Pérez
4 de diciembre de 2013

Índice

1. Introducción

2. Descripción del Problema

En esta entrega se pretende crear una base de datos para una empresa que ofrece servicios de películas mediante *streaming*. Para lograr esto se deben integrar las tres bases de datos creadas de manera individual por los autores. Las bases de datos originales corresponden a las siguientes: una base de datos de Red Social, en donde los usuarios pueden comentar y poner "me gusta" a los recursos, una base de datos de Películas, que contiene toda la información de películas, actores y directores, y una base de datos de Empresas, que contiene información de los servicios que ofrecen las empresas y los usuarios que los han contratado.

Para lograr la integración se deben crear vistas que obtengan información interesante de las tres bases de datos. Se generarán 6 vistas, 2 por cada par de base de datos: Social/Películas, Películas/Empresas Empresas/Social. Luego, utilizando las vistas creadas, se deben hacer 3 consultas que obtengan información importante de los datos. La idea es crear consultas que obtengan información relevante desde el punto de vista de la empresa, para la "toma de decisiones".

Por último se debe crear una interfaz web simple en PHP, desde la cual se puedan ejecutar las consultas creadas. La interfaz también debe considerar la variación de parámetros en las consultas, por ejemplo, cambiar el genero de una película, o cambiar el número de seguidores pedidos para un usuario.

3. Solución

3.1. Integración

El primer problema que se debió resolver es cómo se realizó la integración de las tres bases de datos. En este informe no se describirán en detalle los esquemas para realizar las bases de datos, pero se presenta un diagrama de las tres bases de datos en el anexo. Además, por si se tiene alguna duda sobre la bases de datos, se adjuntan los informes anteriores de las bases de datos individuales.

Para realizar la integración se debió encontrar una forma de relacionar los tres pares de base de datos, intentando evitar en la mayor medida posible, conflictos de duplicidad, pero con la restricción de no modificar los esquemas actuales, y no agregar nuevas tablas. La relación entre la base de datos Social/Películas se hizo considerando que las películas también pueden ser recursos, a los que se las puede comentar y dar "me gusta". Para lograr esto se decidió hacer que los recursos que representan películas tengan la misma id en el atributo `id_source` que la id de la película. Usualmente la `id_source` apunta al recurso que se está comentando, pero en este caso es el vínculo con la tabla películas. Para diferenciar ese recurso con los otros, se debe colocar en su atributo `type` el caracter 'm' de *movie*.

La relación entre Empresa y Red Social se hizo a través de los usuarios de ambas bases de datos. Debido a que en la Red Social la llave primaria de los usuarios era una id, y en Empresas era un RUT, no se podían relacionar por ese atributo. En cambio se decidió relacionar ambas tabla por el email de los usuarios. En principio los mails no son llave primaria, sin embargo para lograr este vínculo se necesario imponer la restricción de un mail único por usuario. Para conservar esta

restricción, ésta se impone a la hora de ingresar los datos.

Por último, para relación entre Películas y Empresas se hizo que un conjunto de los servicios básicos que ofrecen las empresas sea cada una de las películas (por separado) en un estilo de *pay per view*. De esta manera el vínculo entre ambas bases de datos es el título de la película con el nombre del servicio base. Un problema que se tuvo con este enfoque es que algunas veces el *string* que contenía el título de la película, que es un VARCHAR(100), era demasiado largo y con alcanzaba en el atributo del nombre de servicio, que es un VARCHAR(20). Para solucionar eso simplemente se ingresó en servicios el título de la película hasta donde alcanzara en el atributo. Luego, para hacer un join, se usa el operador `like` en vez de la igualdad. Se considera que el fragmento del título de la película tomada es el suficiente para que no haya ambigüedad entre las películas, pero no se puede asegurar.

3.2. Vistas

Las vistas que se implementaron para extraer información de las tres bases de datos son las siguientes (el código para crear las vistas es se encuentra en "vistas todas.sql):

- Vistas Social/Películas:

  1. La cantidad de "me gustan" y "no me gustan" de cada película. Atributos: película, n° de "me gusta", n° de "no me gusta".

pelicula	n° likes	n° de dislikes

Cuadro 1: Vista 1

  2. Tags usados por usuarios que sea el nombre de un actor, y la cantidad de "me gusta" y "no me gusta" de ese tag. Atributos: tag, n° de "me gusta", n° de "no me gusta".

tag	n° likes	n° de dislikes

Cuadro 2: Vista 2

- Vistas Películas/Empresa:

  1. Los directores con sus películas mas caras (ofrecida por una empresa). Atributos: director, película, precio.

director	película	precio

Cuadro 3: Vista 3

  2. La cantidad de película de cierto genero que ofrecen los proveedores. Atributos: proveedor, genero, n° películas.

- Vistas Empresa/Social:

proveedor	genero	n° películas

Cuadro 4: Vista 4

usuario 1	usuario 2	servicio	proveedor

Cuadro 5: Vista 5

1. Los usuarios que son amigos y que contratan uno o más servicios del mismo proveedor. Atributos: usuario 1, usuario 2, servicio, proveedor.
2. Los usuarios con al menos un seguidor, la cantidad de seguidores y los servicios que contratan. Atributos: usuario, n° seguidores, servicios.

usuario	n° followers	servicios

Cuadro 6: Vista 6

3.3. Interfaz Web

La interfaz web creada se encuentra en el dominio `http://anakena.dcc.uchile.cl/~cc320125/`.

4. Consultas SQL

Para esta tarea se implementaron 3 consultas que podrían ser de interés para la empresa, y que obtuvieran información cruza da las bases de datos. También se intentó aprovechar al máximo las vistas implementados. Las consultas presentan parámetros que son fijoc cuando fueron creada, pero mediante al PHP se pueden variar. El código para hacer la consultas se encuentra en el archivo "consultas todas.sql". Las consultas creadas son:

1. Las películas que tienen más n "no me gusta" y menos de m "me gusta" y que corresponden a la película más cara de cierto director. Esta consulta podría ser interesante ya que se podría identificar directores que cobra muy caro por las películas, pero éstas no son muy populares.

```
select l.pelicula, d.Director, l.dislikes as n_dislikes, d.Precio
from cc320114_db.likeo_peliculas as l
join cc320114_db.Directores_pelicula as d on l.pelicula = d.Pelicula
where l.dislikes > 2 and l.likes < 3;
```

pelicula	Director	n° dislikes	n° Precio

Cuadro 7: Resultado consulta 1

2. Los usuarios con más de n seguidores, los servicios que contratan y el precio de esos servicios. Con esta consulta se puede identificar a la gente más popular, de manera de hacer rebajas en los servicios que contratan y provocar que los usuarios comuniquen la noticia.

```
select f.nombre, f.n_followers, f.servicio, s.Precio
from cc320114_db.followers_servicios as f
join cc320151_db.Servicios_base as s on f.servicio = s.Nombre_servicio
where n_followers > 4;
```

nombre	n_followers	servicio	Precio

Cuadro 8: Resultado consulta 2

3. Las películas cuya cantidad de "likes totales" (likes - dislikes) es menor a n, y tal que la cantidad de "likes totales" que tienen los actores que actuaron en esas películas en los tags sea mayor a m. De esta manera se identifica a los "malos directores", que a pesar de tener un elenco popular, su película fue mal evaluada.

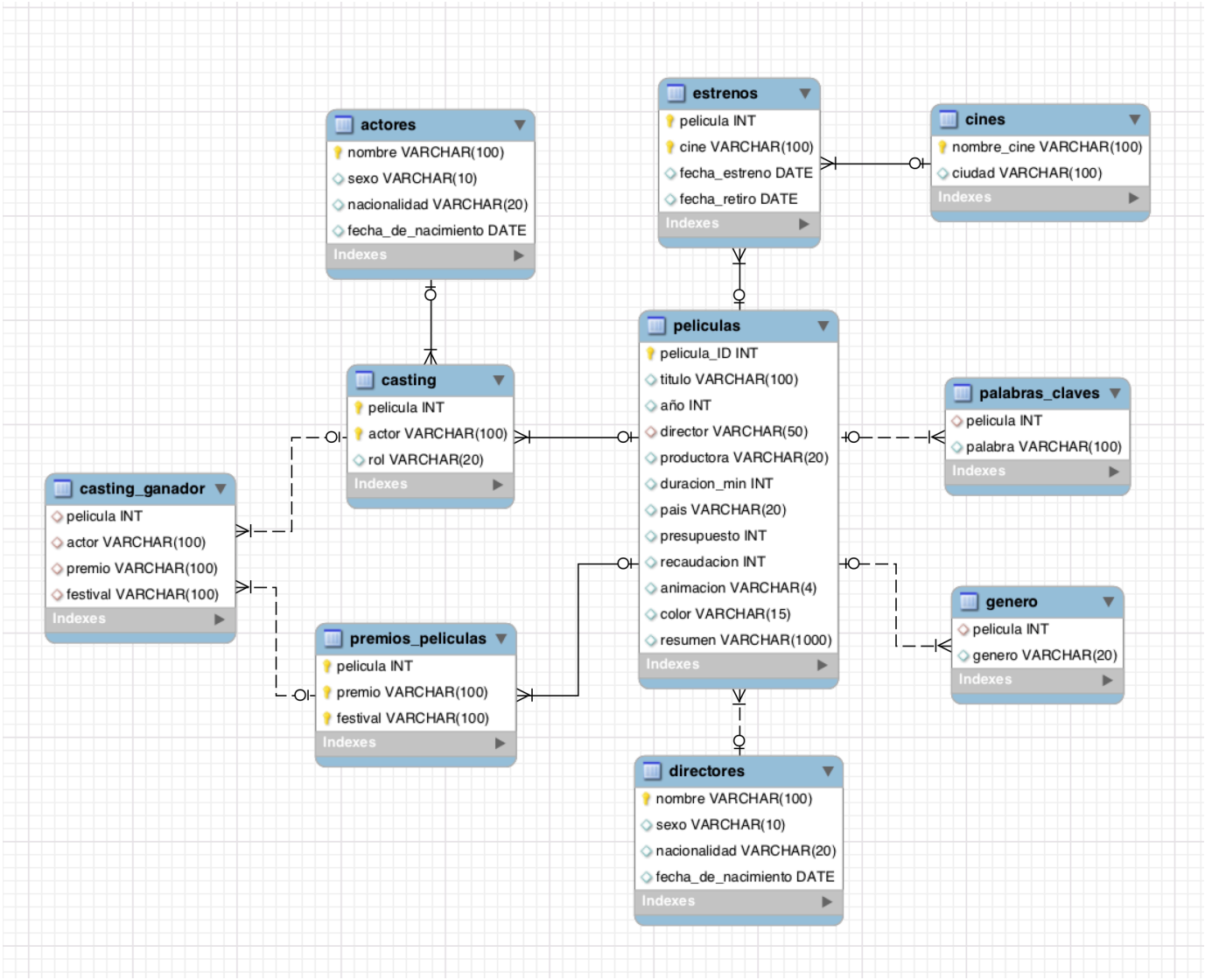
```
select s.pelicula, s.director, s.total_likes_pelicula, s.total_likes_elenco
from(
select p.titulo as pelicula, p.director as director,
l.likes - l.dislikes as total_likes_pelicula,
sum(t.Likes - t.Dislikes) as total_likes_elenco
from cc320114_db.peliculas as p
join cc320114_db.likeo_peliculas as l on p.titulo = l.pelicula
join cc320114_db.casting as c on p.pelicula_ID = c.pelicula
join cc320114_db.Tag_de_actores as t on c.actor = t.Actor
group by l.pelicula) as s
where total_likes_pelicula < 2 and total_likes_elenco > 2;
```

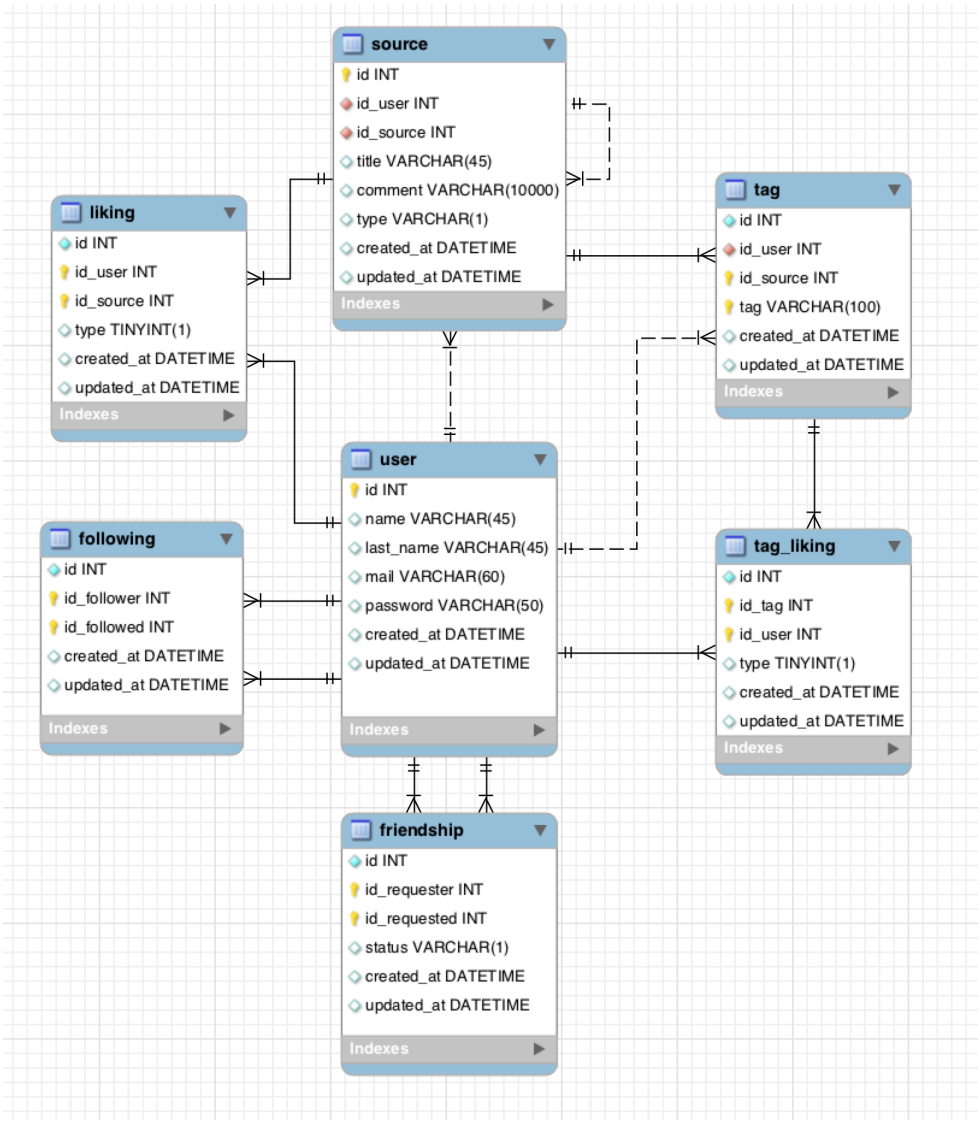
pelicula	director	total_likes_peliculas	total_likes_actores

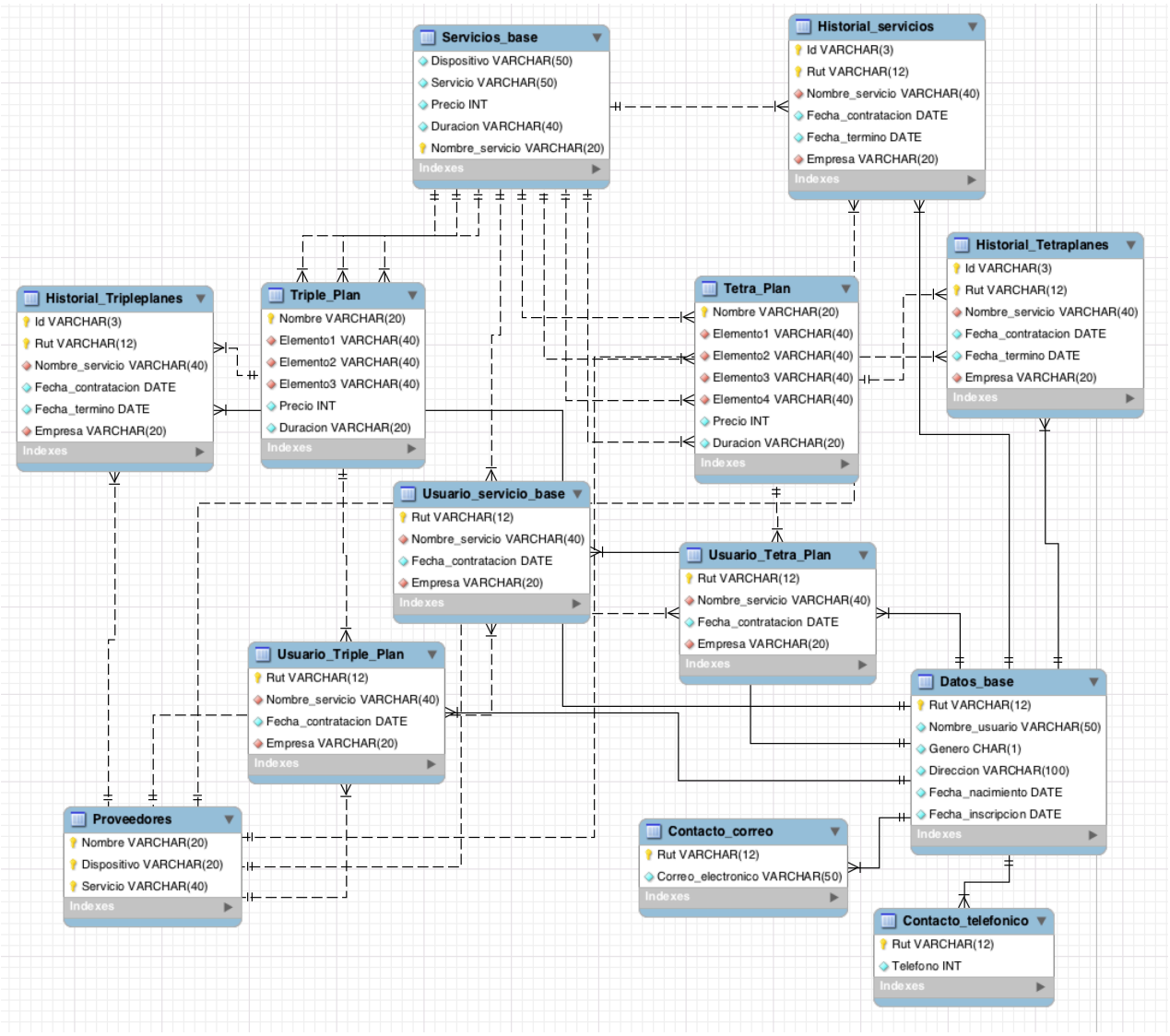
Cuadro 9: Resultado consulta 3

5. Anexos

5.1. Diagrama de las bases de datos


actores
nombre VARCHAR(100)
sexo VARCHAR(10)
nacionalidad VARCHAR(20)
fecha_de_nacimiento DATE
Indexes
estrenos
pelicula INT
cine VARCHAR(100)
fecha_estreno DATE
fecha_retiro DATE
Indexes
cines
nombre_cine VARCHAR(100)
ciudad VARCHAR(100)
Indexes
casting
pelicula INT
actor VARCHAR(100)
rol VARCHAR(20)
Indexes
peliculas
pelicula_ID INT
titulo VARCHAR(100)
año INT
director VARCHAR(50)
productora VARCHAR(20)
duracion_min INT
pais VARCHAR(20)
presupuesto INT
recaudacion INT
animacion VARCHAR(4)
color VARCHAR(15)
resumen VARCHAR(1000)
Indexes
palabras_claves
pelicula INT
palabra VARCHAR(100)
Indexes
casting_ganador
pelicula INT
actor VARCHAR(100)
premio VARCHAR(100)
festival VARCHAR(100)
Indexes
premios_peliculas
pelicula INT
premio VARCHAR(100)
festival VARCHAR(100)
Indexes
genero
pelicula INT
genero VARCHAR(20)
Indexes
directores
nombre VARCHAR(100)
sexo VARCHAR(10)
nacionalidad VARCHAR(20)
fecha_de_nacimiento DATE
Indexes


source
id INT
id_user INT
id_source INT
title VARCHAR(45)
comment VARCHAR(10000)
type VARCHAR(1)
created_at DATETIME
updated_at DATETIME
Indexes
liking
id INT
id_user INT
id_source INT
type TINYINT(1)
created_at DATETIME
updated_at DATETIME
Indexes
tag
id INT
id_user INT
id_source INT
tag VARCHAR(100)
created_at DATETIME
updated_at DATETIME
Indexes
user
id INT
name VARCHAR(45)
last_name VARCHAR(45)
mail VARCHAR(60)
password VARCHAR(50)
created_at DATETIME
updated_at DATETIME
Indexes
following
id INT
id_follower INT
id_followed INT
created_at DATETIME
updated_at DATETIME
Indexes
tag_liking
id INT
id_tag INT
id_user INT
type TINYINT(1)
created_at DATETIME
updated_at DATETIME
Indexes
friendship
id INT
id_requester INT
id_requested INT
status VARCHAR(1)
created_at DATETIME
updated_at DATETIME
Indexes


Servicios_base
Dispositivo VARCHAR(50)
Servicio VARCHAR(50)
Precio INT
Duracion VARCHAR(40)
Nombre_servicio VARCHAR(20)
Indexes
Historial_servicios
Id VARCHAR(3)
Rut VARCHAR(12)
Nombre_servicio VARCHAR(40)
Fecha_contratacion DATE
Fecha_termino DATE
Empresa VARCHAR(20)
Indexes
Historial_Tripleplanes
Id VARCHAR(3)
Rut VARCHAR(12)
Nombre_servicio VARCHAR(40)
Fecha_contratacion DATE
Fecha_termino DATE
Empresa VARCHAR(20)
Indexes
Triple_Plan
Nombre VARCHAR(20)
Elemento1 VARCHAR(40)
Elemento2 VARCHAR(40)
Elemento3 VARCHAR(40)
Precio INT
Duracion VARCHAR(20)
Indexes
Tetra_Plan
Nombre VARCHAR(20)
Elemento1 VARCHAR(40)
Elemento2 VARCHAR(40)
Elemento3 VARCHAR(40)
Elemento4 VARCHAR(40)
Precio INT
Duracion VARCHAR(20)
Indexes
Historial_Tetraplanes
Id VARCHAR(3)
Rut VARCHAR(12)
Nombre_servicio VARCHAR(40)
Fecha_contratacion DATE
Fecha_termino DATE
Empresa VARCHAR(20)
Indexes
Usuario_servicio_base
Rut VARCHAR(12)
Nombre_servicio VARCHAR(40)
Fecha_contratacion DATE
Empresa VARCHAR(20)
Indexes
Usuario_Tetra_Plan
Rut VARCHAR(12)
Nombre_servicio VARCHAR(40)
Fecha_contratacion DATE
Empresa VARCHAR(20)
Indexes
Usuario_Triple_Plan
Rut VARCHAR(12)
Nombre_servicio VARCHAR(40)
Fecha_contratacion DATE
Empresa VARCHAR(20)
Indexes
Datos_base
Rut VARCHAR(12)
Nombre_usuario VARCHAR(50)
Genero CHAR(1)
Direccion VARCHAR(100)
Fecha_nacimiento DATE
Fecha_inscripcion DATE
Indexes
Proveedores
Nombre VARCHAR(20)
Dispositivo VARCHAR(20)
Servicio VARCHAR(40)
Indexes
Contacto_correo
Rut VARCHAR(12)
Correo_electronico VARCHAR(50)
Indexes
Contacto_telefonico
Rut VARCHAR(12)
Telefono INT
Indexes