**TRABAJO PRACTICO**

**GESTION DE DATOS**

**PRIMER CUATRIMESTRE**

**2017**

**Curso:** K3114

**Nombre del grupo:** CRAZYDRIVER

**Integrantes:**

Nombre: Ruben Maier Enzler

Legajo: 1506420

Email responsable: ruben.maier@hotmail.com

Nombre: Norman Perrin

Legajo: 1472150

Nombre: Santiago Nicolás Maresco

Legajo: 1522790

Nombre: Nicolas Matías Pfeger

Legajo: 1402547

***Criterios tomados en la migración:***

**Usuarios:**

Consideramos que un usuario será todo aquel que tenga acceso al sistema, en este caso hasta ahora son los choferes, clientes y administradores. Como en la base maestra existen choferes y clientes, a la hora de migrarlos les asignamos un usuario a cada uno con el username igual a su DNI (único) y la password la evaluamos por default igual a la contraseña definida para el administrador por requisito del enunciado del trabajo practico (w23e). Consideramos que la misma podría ser modificada a futuro si se implementara un sistema de configuración para el usuario donde podría modificar su perfil propiamente.

**Doble Facturación:**

Consideramos que hay facturas impagas y que anteriormente a que el sistema sea relevado, se pretendía manipular la información impaga y paga duplicando los datos.

Query para validar que hay duplicados:

SELECT Auto\_Patente, Chofer\_Dni, Cliente\_Dni, Viaje\_Fecha, Viaje\_Cant\_Kilometros, Turno\_Descripcion, Turno\_Precio\_Base, Turno\_Valor\_Kilometro, Factura\_Nro, Rendicion\_Nro, Rendicion\_Importe, count(\*)

FROM gd\_esquema.Maestra

WHERE Factura\_Nro IS NOT NULL AND Rendicion\_Nro IS NULL

GROUP BY Auto\_Patente, Chofer\_Dni, Cliente\_Dni, Viaje\_Fecha, Viaje\_Cant\_Kilometros, Turno\_Descripcion, Turno\_Precio\_Base, Turno\_Valor\_Kilometro, Factura\_Nro, Rendicion\_Nro, Rendicion\_Importe

HAVING count(\*) > 1

ORDER BY Chofer\_Dni, Auto\_Patente, Cliente\_Dni, Viaje\_Fecha, Turno\_Descripcion, Factura\_Nro, Rendicion\_Nro

**Doble Rendición:**

Consideramos que las rendiciones son únicas, pero sin embargo hay rendiciones duplicadas:

Query para validar que hay duplicados:

SELECT Auto\_Patente, Chofer\_Dni, Cliente\_Dni, Viaje\_Fecha, Viaje\_Cant\_Kilometros, Turno\_Descripcion, Turno\_Precio\_Base, Turno\_Valor\_Kilometro, Factura\_Nro, Rendicion\_Nro, Rendicion\_Importe, count(\*)

FROM gd\_esquema.Maestra

WHERE Factura\_Nro IS NULL AND Rendicion\_Nro IS NOT NULL

GROUP BY Auto\_Patente, Chofer\_Dni, Cliente\_Dni, Viaje\_Fecha, Viaje\_Cant\_Kilometros, Turno\_Descripcion, Turno\_Precio\_Base, Turno\_Valor\_Kilometro, Factura\_Nro, Rendicion\_Nro, Rendicion\_Importe

HAVING count(\*) > 1

ORDER BY Chofer\_Dni, Auto\_Patente, Cliente\_Dni, Viaje\_Fecha, Turno\_Descripcion, Factura\_Nro, Rendicion\_Nro

Ejemplo:

select \* from gd\_esquema.Maestra where Chofer\_Dni = 48584942 and Cliente\_Dni = 18911370 and Rendicion\_Nro = 22937

Pero nótese que para cada doble rendición existen 4 facturaciones y a su vez 2 viajes idénticos sin facturación ni rendición, por lo que consideramos que son dos viajes distintos.

select Auto\_Patente, Chofer\_Dni, Cliente\_Dni, Turno\_Descripcion, Viaje\_Cant\_Kilometros, Viaje\_Fecha, Rendicion\_Fecha, Rendicion\_Importe, Rendicion\_Nro, Factura\_Fecha, Factura\_Fecha\_Fin, Factura\_Fecha\_Inicio, Factura\_Nro

from gd\_esquema.Maestra m

where exists

(

select Cliente\_Dni,Chofer\_Dni,Auto\_Patente,Turno\_Descripcion,Rendicion\_Nro,Rendicion\_Fecha,Rendicion\_Importe,Viaje\_Fecha,Viaje\_Cant\_Kilometros, count(Rendicion\_Nro) cantidad

from gd\_esquema.Maestra

where Rendicion\_Nro is not null

and Cliente\_Dni = m.Cliente\_Dni and Chofer\_Dni = m.Chofer\_Dni and Auto\_Patente = m.Auto\_Patente and Turno\_Descripcion = m.Turno\_Descripcion

and Viaje\_Fecha = m.Viaje\_Fecha and Viaje\_Cant\_Kilometros = m.Viaje\_Cant\_Kilometros

group by Cliente\_Dni,Chofer\_Dni,Auto\_Patente,Turno\_Descripcion,Rendicion\_Nro,Rendicion\_Fecha,Rendicion\_Importe,Viaje\_Fecha,Viaje\_Cant\_Kilometros

having count(rendicion\_nro) > 1

)

order by Auto\_Patente, Chofer\_Dni, Cliente\_Dni, Turno\_Descripcion, Viaje\_Cant\_Kilometros, Viaje\_Fecha, Rendicion\_Fecha, Rendicion\_Importe, Rendicion\_Nro, Factura\_Fecha,Factura\_Fecha\_Fin, Factura\_Fecha\_Inicio, Factura\_Nro;

Nótese que también hay rendiciones duplicadas, pero tienen solo 2 facturaciones, por lo que consideramos que estos casos son facturaciones impagas

**Fecha en los viajes:**

Basados en los requisitos que el enunciado solicita a la hora de registrar un viaje pidiéndonos ingresar una fecha de inicio y una fecha de fin del viaje, tomamos como condición de migración utilizar la “fecha” que figuraba en la tabla maestra como “fecha\_inicio” y frente a la eventualidad de no dejar valores nulos en la hora fin también duplicamos este dato ingresando el valor de la columna “fecha” de la tabla maestra en nuestra columna “fecha\_fin”.

**Facturaciones:**

Consideramos que las facturaciones se hacen 1 vez por mes para los clientes, pero permitimos que si el administrador decide se pueda facturar a un cliente antes de que termine el mes, aunque si esto ocurre este cliente no podrá realizar viajes hasta no haber finalizado el mes.

**Rendiciones:**

Consideramos que las rendiciones se efectúan una sola vez por día y al cierre del día, es decir se permite realizar la rendición a partir de las 0 horas, que es el comienzo del día siguiente. Por este motivo, implica a su vez que el chofer solo pueda trabajar en un único turno, aunque el sistema en sí, está modelado para poder llegar a soportar la posibilidad de poder tener choferes que hagan más de un turno de ser necesario.

**Usuarios del sistema:**

Para los administradores definimos que tengan permiso/acceso a todas las funcionalidades, pero para los clientes y choferes solo definimos algunas funcionalidades randoms como para validar que funcionan, pero al no tener un requerimiento de esta condición que nos especifique que funcionalidades puede realizar cada tipo de usuarios, tomamos un criterio propio a fin de poder realizar un testeo de su funcionamiento.

**Automóvil:**

Consideramos que a un automóvil no se le puede modificar ni la marca ni el modelo puesto que esta es una condición de fabricación del vehículo y no algo alterable.

**Criterio de programación:**

* Se decidió trabajar con una arquitectura de capas para facilitar la detección de errores separando así el programa en:
  + **Capa datos:** encargada de prestar servicio a la capa negocio y ser la única capaz de interactuar con la base de datos de manera directa.
  + **Capa negocio:** cumple la función de intermediario entre las necesidades de la interfaz y la solicitud de peticiones a la base de datos.
  + **Capa interfaz:** es la encargada de brindarle servicios solicitados a las distintas ventanas que el sistema implementa. La capa interfaz cumple la función de separar código de lógica de interfaz con respecto a código propiamente de ventanas que sí interactúan estrechamente con el usuario.
  + **Entidades:** es una capa que cumple la función puramente de evitar repetir código innecesario facilitar así la programación, en especial la intercomunicación entre las distintas capas mencionadas anteriormente que solicitan datos que cumplen con cierta “lógica común” o “patrón”. Abstrae las condiciones de las entidades de la base de datos permitiéndonos así poder aplicar criterios de programación orientada a objetos.
  + **UBER FRBA:** Contiene las distintas ventanas con las que el usuario va a interactuar con el programa.
* Tomamos el valor -1 como consideración de “invalidez”, es decir, dato o valor erróneo.