ESTRATEGIA
“PALCO NET”

Materia: Gestión de Datos

Grupo Nro.: 17

Nombre del grupo: COMPUMUNDOHIPERMEGARED

Alumnos:

- Fiorella Denise Altobelli, 159.092-3
- Martín Gabriel Mallea Santagada, 159.093-5
- Lautaro Gomez Odriozola, 156.607-6

Descripción:

En esta breve guía pasaremos a dar un paneo general de la implementación de las tablas fundamentando el porqué de cada relación establecida, a su vez se enunciaron y describirán las procedures utilizadas, funciones, triggers,vistas, secuencias y hablaremos un poco sobre características generale y particulares de cada abm y función a implementar

Objetos de la base de datos **3**
Tablas y sus relaciones 3
Usuario 3
Rol_Usuario 3
Rol 3
Rol_Funcionalidad Y Funcionalidad 3
Cliente Y Empresa 4
Puntos_Canje, Canje, Puntos Y Premio_Disponible. 4
Compra, Item_Factura, Factura y Tarjeta 5
Tarjeta 5
Ubicación 5
TipoUbicacion 5
Espectáculo 6
Rubro 6
Publicación 6
Grado 6
Sector 7
Relación entre cliente y tarjeta 7
Funciones 7
Procedures 8
Triggers 9
Vistas 9
Secuencias 10

Consideraciones en la migración **10**
Usuarios creados 10
Publicaciones 10
Rubros 10
Compras y ubicaciones 10
Puntos 11
Clientes y empresas 11
Clientes 11

Consideraciones generales sobre las funcionalidades implementadas en el TP **11**
Roles y Funcionalidades 11
Cambio de clave 12
ABM Clientes 12
Sobre las tarjetas 12
Validaciones al crear un cliente 12
ABM de Empresas 13
Validaciones al crear una empresa 13
Publicaciones 13

Para la tabla de publicación y el modo en que el cliente genera las publicaciones 13
Otras consideraciones 14
ABM grado 14
Canje de Puntos 15
Compras 15
Consideraciones para las publicaciones mostradas al momento de comprar 16
Listado estadístico 16
Rendición de comisiones 16
Habilitación/Inhabilitación de Usuarios 17
Configuración de fecha y hora del sistema 17
Der 18

Objetos de la base de datos

Tablas y sus relaciones

Usuario

La tabla usuario nos permite registrar principalmente el username y el password asociado a un Cliente o Empresa. Ademas la tabla cuenta con una PK para identificarla llamada id_usuario.
La tabla posee un campo de validación de intentos de log-in, que consta de un tinyint seteada inicialmente en 3. Se le permite al usuario una cantidad de ingresos erróneos de su password, la cual decrece por cada uno de ellos. Al llegar a 0, el procedure que valida el login actualiza el estado mediante el campo habilitado haciéndolo pasar de habilitado a deshabilitado y vuelve los intentos a 0 para que cuando un administrador lo vuelva a habilitar el usuario pueda volver a logearse.
Tanto habilitado como eliminado están compuestos por un bit permitiéndonos definir dos estados
Variable habilitado: se refiere a la habilitación/deshabilitación por ingresos fallidos
habilitado=1 / deshabilitado=0
Variable eliminado: está para contemplar la posibilidad de que un administrador pueda dar de baja a un usuario
eliminado=1 / activo= 0
Diferenciamos el concepto de usuario (como medio para identificarse en el sistema) de rol/tipo de usuario (cliente, empresa, administrador u otros posibles). Cuando un usuario se registre puede elegir hacerlo como cliente o como empresa de manera inmutable, pero la base de datos igualmente permite agregar ilimitados roles a esta cuenta.

Rol_Usuario

La creación de la tabla rol usuario surgió como una tabla intermedia a partir de detectar que la relación entre usuarios y roles era de muchos a muchos, la misma está creada con una PK que corresponde al código de Id_Rol_Usuario. A su vez posee referencia a ambas tablas por medio de dos FK Usuario_Id y Rol_Id.

Rol

Posee el campo “habilitado” el cual nos permite definir una baja lógica. Al ser deshabilitados se eliminan todas sus asignaciones a usuarios en la tabla Rol_Usuario.

Rol_Funcionalidad Y Funcionalidad

Así como un usuario puede tener muchos roles, un rol puede tener muchos usuarios y viceversa. Esto nos generaba una relación de muchos a muchos solucionada mediante una tabla intermedia denominada Rol_Funcionalidad, cuya PK es particular (propia de la tabla) y

posee una FK a cada tabla de la relación.

Cliente Y Empresa

Las tablas Cliente y Empresa son la base principal de los usuarios, ya que los mismos pueden ser o cliente o empresa.
El Domicilio de clientes y empresas se definió como datos separados y embebidos. Esta decisión se tomó ya que por ejemplo el domicilio de la calle en Clientes figura como un nvarchar(255) y para Empresas es un nvarchar(50) teniendo una inconsistencia en el tipo de dato.
Además en el modelo de nuestra aplicación no realizamos consultas referidas al domicilio ni lo utilizamos de manera separada a cliente o empresa no vemos que tener una tabla para el domicilio sea algo importante. Por ej: no estamos interesados en saber cuántos ni qué clientes viven en el mismo domicilio.
Se agregó el atributo Ciudad al cliente ya que la Empresa lo tenía y este no.
En la tabla de empresas no permitimos que se repita el cuit mediante una constraint de unique.
En la tabla de clientes no pudimos realizar esto ya que en la tabla maestra no había datos de cuil. Sin embargo, podemos tener unique para el DNI. Y la forma de validar que no va a haber repetidos en el cuil es validar la correspondenci entre DNI y cuil en la app, por lo que al no permitir insertar DNIs repetidos y al corresponder los CUILs con los DNIs, estos tampoco podrán repetirse.
El cliente tiene un tipo_documento que lo representamos con un char(1) porque creemos que es la forma más eficiente de hacerlo. Y además agregamos una constraint de check para validar que los datos almacenados sean consistentes. Es bastante improbable que se agreguen más tipos de DNIs por lo que no nos interesa mantenerlo normalizado.
A cada Cliente/Empresa se la asigna un usuario único que se valida en el registro del usuario.

Puntos_Canje, Canje, Puntos Y Premio_Disponible.

Puntos canje es una tabla intermedia que se genera a partir de la relación entre Canje y Puntos, que genera una relación de muchos a muchos, la misma no posee ninguna PK, ya que no se consideró necesario, pero si contiene como FK las PK de canje y puntos respectivamente.
Paralelamente la tabla puntos se asocia a un Cliente_Id para determinar de qué usuario son, y a una Compra_id para saber en qué compra se acreditaron dichos puntos. Con la compra se puede saber además de eso, la cantidad de puntos que se ganaron, la fecha, y cuando vencerán dichos puntos. Esta última fecha se calcula al momento de sumar los puntos y es 3 meses posterior a la fecha de compra.
Incluida en estas operaciones tenemos la tabla Canje asociada con PremioDisponible esto se realiza a través de la FK id_premio, que nos permite relacionar un canje de puntos con el premio por el que se realizó el canje.

Compra, Item_Factura, Factura y Tarjeta

En el circuito de compras se engloba varias tablas relacionadas entre sí, cuya principal es Compra, la misma está relacionada a un usuario determinado a través de la FK Cliente_Id y a un medio de pago, a través de la FK Tarjeta_Id. El campo Tarjeta_Id fue implementado debido a que se debia establecer un medio de pago “electronico”. Y además porque se debe conservar la relación entre la compra y la tarjeta con la que fue realizada.
Además cada compra posee una PK que es su Id_Compra, el cual se relaciona con la tabla Item_Factura y con Factura. Lo que sucedía era que la relación entre compra y factura generaba una relacion “muchos a muchos” por eso tuvimos que optar por utilizar una tabla intermedia que de los ítems por cada factura.
En la Tabla Item_Factura poseemos como PK una estructura doble que apunta tanto al Factura_id como a la Compra_id siendo estos a su vez FK. En la misma se especifica la descripcion,cantidad y monto.
Con el Id_Factura se sigue la relación hacia la tabla Factura la cual deja constancia de la relacion entre toda la información recopilada en Item__Factura sobre el cliente, los productos (espectáculos), la cantidad y el monto, con la empresa proveedora de dicho servicio, a través del Empresa_id .

Tarjeta

Podemos ver como el tipo de la tarjeta está modelado con un char(1) agregando una constraint de check. Nos parece importante acá mantener la selección acotada.
Además no se permiten nulos ni en el número de tarjeta ni el tipo ni ccv ni fecha de vencimiento. Y el cliente_id tampoco será nulo ya que se completa en el procedure que asigna las tarjetas.

Ubicación

La tabla ubicación posee tres relaciones principales.
La primera de ellas es la Compra_id, ya que cuando se registra la compra de un espectáculo se debe determinar de alguna manera que ubicación fue adquirida en dicha compra. Nuestra manera de modelarlo fue que la ubicación contenga una FK que nos permita determinar a qué compra pertenece. Al estar la FK del lado de ubicación estamos permitiendo que la relación de compra-ubicación permita que haya varias ubicaciones por compras.
A su vez la Ubicación posee una relación (FK) con la Publicación a través de publicacion_id. Esto es debido a que una publicacion puede contener muchas ubicaciones ditintas, por lo que nos pareció lógico modelar dicha estructura del lado de la ubicación, es decir como hicimos con la compra, que sea la ubicación la encargada de tener una FK a que publicación corresponde

TipoUbicacion

La compra se relaciona con su tipo de ubicación de la FK tipo_ubicacion_id ya que las ubicaciones pueden ser de distinto tipo como ser: palco o platea, esa “clasificación” la modelamos a través de una tabla cuya referencia se encuentra en Ubicación. Nos parece

adecuado tener normalizada esta información para evitar la repetición de datos y facilitar tanto la búsqueda como el ingreso de datos (ya que estos son acotados).

Espectáculo

Como las empresas se dedican a los espectáculos decidimos crear una tabla para persistir en ella toda la información acerca de estos. Cada espectáculo se identifica a través de su id_Espectaculo manteniendo a su vez una relación con la empresa que es propietaria de dicho espectáculo a través del campo empresa_id. Los datos restantes corresponden al domicilio donde se llevará a cabo dicho espectáculo.
Nos interesa separar el concepto de espectáculo del de ubicación ya que un espectáculo podría ser publicado varias veces y esto nos evita repetir información.

Rubro

En cuanto al rubro, cada espectáculo debe pertenecer a uno, por lo tanto decidimos que se posea una FK denominada rubro_id que nos hace referencia al código del mismo y nos permite acceder a su descripción en la tabla Rubro. Nos interesa mantener normalizada esta información ya que, por más que no esté implementado, si se hiciera un ABM de Rubro sería la mejor forma de mantenerla consistente tras realizar altas, bajas y modificaciones.

Publicación

Las publicaciones se corresponden a espectáculos es por eso que un espectáculo puede tener más 1 o más publicaciones. Dicha relación la marcamos mediante una FK en la tabla Publicación denominada espectáculo _id.
Cada publicación posee:

- fecha_creacion: que es el momento en la cual se publicó.
- fecha_vencimiento: que es el momento en el cual la publicación pasó al estado finalizada.
- fecha_espectaculo: es la fecha en la cual se lleva a cabo el mismo.

La publicación tiene un estado el cual lo representamos con un char(1), que puede ser ‘P’, ‘F’ o ‘B’ para Publicada, Finalizada y Borrador respectivamente y está validado con una constraint de check que eso está en estos valores. Hicimos esto porque vemos poco probable agregar nuevos estados a la publicación. De hecho, no tenemos una funcionalidad que permita ni agregar ni sacarle estados a la publicación. Tampoco permitimos el nulo en este campo porque una publicación debería estar obligatoriamente en alguno de esos estados.
La publicación también tiene una FK a la tabla de grados.

Grado

Tiene una comisión, descripcion y a su vez comprobar si el grado está eliminado a través del bit eliminado permitiendo así una baja lógica.
Además consideramos a la comisión como el atributo para ordenar publicaciones por grado de visibilidad. Ya que las que pagan más comisión deberían tener mejor prioridad.

Sector

La tabla Sector posee la característica de no tenes una PK pero si posee dos FK las mismas apuntan al espectáculo Id_Espectaculo y al tipo de ubicacion Tipo_Ubicacion_Id. en el sector se estable el precio de las filas justamente por “sectores” es decir desde x fila hasta otra x fila.
Se utiliza para persistir la información de qué ubicaciones van a querer crearse para cierta publicación mientras esta se encuentra en estado borrador. Permitiendo así que el usuario no necesite ingresar las ubicaciones una por una sino “describir” cómo quiere generar las ubicaciones.
La tabla guarda información de sectores tanto numerados como no numerados ya que hubiera sido más costoso separarlos en dos tablas ya que siempres se piden todos juntos. Se valida desde la app que los datos se carguen consistentemente para un sector numerado o no numerado.

Relación entre cliente y tarjeta

Entre estas dos tablas puede observarse que hay una doble relación. Esto es porque el cliente siempre tendrá una tarjeta considerada su tarjeta actual, con la cual realizará las compras. Pero nunca se podrán modificar los registros de la tabla de tarjetas ya que estos podrían estar relacionados a compras ya realizadas. Por esto, modificar la tarjeta del cliente implica siempre crear un nuevo registro en la tabla de tarjetas y asignarlo como tarjeta actual. Mientras que los registros anteriores deben seguir relacionados con el cliente y por eso la FK desde tarjeta hacia cliente.

Funciones

- *DescripcionOrElse:* Se usa para migrar las publicaciones que no tienen ningún rubro asignándoles el rubro ‘Sin rubro’.
- *find_funcionalidades_de_rol*: Devuelve una tabla con las funcionalidades disponibles para cierto rol. Se lo llama desde la función *find_funcionalidades_de_usuario*.
- *find_funcionalidades_de_usuario*: se utiliza para obtener todas las funcionalidades disponibles para un usuario a partir de su id. Se utiliza desde la app para cargar las funcionalidades a las que un usuario puede acceder y de esta forma mostrarle sólo las opciones que son válidas (en el combobox del selector de funcionalidades de la app).
- *get_espectaculo_id_de_publicacion*: se utiliza para obtener a partir de una publicación el id del espectáculo y reutilizar esta lógica al momento de generar las ubicaciones para una publicación y actualizar los datos de una publicación borrador.
- *puntosDeCliente*: se usa para obtener los puntos de un cliente en cierta fecha. Se llama la función desde la funcionalidad de consultar puntos de la app. Realiza la sumatoria de los puntos que no estén vencidos y que tengan una cantidad de puntos restante.
- *StockDePublicacion*: Calcula las ubicaciones disponibles restantes para una publicación. Se utiliza al realizar la compra para verificar si debe finalizar automáticamente la publicación.

- *EmpresasConMenosVentas:* Función que se utiliza para mostrar su resultado en el listado estadístico. Devuelve una tabla con las empresas con menos ventas para un año, un mes y un grado de publicación.
- *GetTipoDocumento*: función que se utiliza para convertir el tipo de documento de la forma en que se almacena en la tabla (char(1)) a una cadena legible para el usuario.
- *ClientesConMasPuntosVencidos*: también se utiliza para el listado estadístico. Devuelve una tabla con los clientes con más puntos vencidos en cierto año y trimestre. Los puntos a sumar son los que tienen su fecha de vencimiento dentro del período indicado. Y la cantidad de puntos a sumar son las que no se hayan usado, es decir *cantidad total - cantidad usada*. Si esa cantidad es mayor a 0, entonces esos puntos se vencieron, ya que su fecha de vencimiento ya pasó. Nunca podría ser menor a 0 porque no se permite usar más puntos que la cantidad total.
- *ClientesConMasComprasDeEmpresa*: se utiliza en el listado estadístico. Devuelve una tabla con los clientes que más compras realizaron a cierta empresa en un cierto período trimestre/año. Se está utilizando el criterio de mostrar primero al que más compras tenga y no necesariamente al que más ubicaciones haya comprado. Por ejemplo: Cliente A compró 5 ubicaciones a una empresa en una sóla compra y Cliente B le compró a la misma empresa dos ubicaciones en dos compras distintas. Entonces el Cliente B se mostrará antes que el Cliente A.
- *ComprasDeCliente:* Función utilizada para mostrar las compras que realizó un cliente cuando él consulta su historial.
- *TopComprasDeEmpresa:* devuelve las primeras compras de una empresa ordenadas por fecha. Se utiliza para la funcionalidad de rendición de comisiones, porque son estas compras las que serán facturadas. Excluye a las compras a las cuales ya se les realizó factura para evitar que se facturen 2 veces.
- *UbicacionesDeCompra:* sirve para mostrar las ubicaciones de una compra cuando son solicitadas en la funcionalidad del listado de compras.

Procedures

- *crearNuevoRol*: Se utiliza para registrar un nuevo rol en el sistema. El cual puede tener varias funcionalidades. Se utiliza en la migración para crear lo roles preexistentes como Cliente, Administrador, Empresa y Administrador General. Se utiliza también en el ABM de Rol cuando el usuario crea un nuevo rol.
- *actualizarRol:* Se utiliza cuando se realiza un cambio en las funcionalidades de un rol desde la app.
- *crear_nuevo_usuario:* Se utiliza para compartir la lógica de la creación de un usuario con cierto rol. Y después reutilizarlo cuando se registra un cliente o una empresa (el rol podría ser tanto cliente como empresa).
- *intentar_logear:* Se utiliza cuando se realiza el login desde la app. Mantiene actualizado el estado de la cantidad de veces que un usuario intentó ingresar insatisfactoriamente y lo deshabilita si ingresó mal 3 veces. El procedure lanza un error si el login es insatisfactorio para informarle a la app (y luego esta al usuario) qué es lo que pasó.
- *crear_usuario_cliente:* Crea un usuario llamando al procedure de *crear_nuevo_usuario* y le agrega también los datos del cliente en la tabla de clientes.

- *crear_usuario_empresa:* Crea un usuario llamando al procedure *crear_nuevo_usuario* y lo asocia a los datos de la empresa que se está registrando.
- *eliminar_rol:* Procedure llamado desde la app desde el ABM de rol cuando se baja un rol. Da la baja lógica de ese rol y quita las asignaciones de ese rol para los usuarios (Rol_Usuario). Las asignaciones las elimina con un delete ya que cuando el rol se habilita de vuelta, de todas maneras no se recuperarán.
- *crear_borrador:* Procedure utilizado para crear un nuevo borrador. Se llama en la funcionalidad de generar publicaciones desde la app.
- *generar_ubicaciones_de*: Procedure utilizado para crear y asignar todas las ubicaciones a una publicación a partir de las zonas descriptas por el usuario.
- *update_datos_borrador:* Actualiza los datos de una publicación borrador. Se llama desde la app en la funcionalidad de editar publicaciones.
- *publicar_fecha:* Genera una publicación de cierto espectáculo para una fecha en particular.
- *realizarCanje:* Es un procedure llamado desde la funcionalidad de Canje de Puntos, el cual canjea los puntos de un cliente por un premio seleccionado.
- *AsignarTarjetaA:* Crea una nueva tarjeta y se la asigna como tarjeta actual a un cliente.
- *ComprarUbicaciones:* Registra la compra de un conjunto de ubicaciones para un cliente. También actualiza la cantidad de ubicaciones compradas y luego valida si la publicación se quedó sin stock y si es así la finaliza.
- *RegistrarPuntosDeCompra:* Se llama desde la app luego de que el cliente realice la compra satisfactoriamente. Además los puntos que se sumaron quedan relacionados con la compra que los generó.
- *RendirComisionesDeEmpresa:* Utilizado en la funcionalidad de rendición de comisiones para generar las facturas por la comisión cobrada a las empresas.

Aclaración: @fecha_creacion en crear_usuario_cliente, crear_usuario_empresa y publicar_fecha; @fecha_actual en realizarCanje y RendirComisionesDeEmpresa, y @fecha en ComprarUbicaciones son parámetros utilizados para pasar la fecha del sistema de la app y usarla en la base de datos.

Triggers

- *PubliTrigger:* se encarga de validar que una publicación no pueda realizar una transición de estados inválidos. Es decir, que no pueda pasar de Publicada a Borrador ni de Finalizada a cualquier otro estado.

Vistas

- *PublicacionesView:* Se utiliza para agrupar todos los datos referidos a una publicación, como los del espectáculo, los del rubro y su grado. Se utiliza tanto en la funcionalidad de listar publicaciones de la empresa como en la de comprar cuando el cliente elige la publicación de la cuál va a comprar entradas.

Secuencias

- *UbicacionSequence:* Utilizada para autogenerar las PKs de las ubicaciones.
- *CompraSequence:* Utilizada para autogenerar las PKs de las compras.

Consideraciones en la migración

Usuarios creados

Para permitir su acceso desde el sistema:

- Las empresas fueron migradas creando usuarios en las que tanto su username y password son su cuil.
- Los clientes fueron migrados creando usuarios en los que tanto su username como su password son su número de documento.

Creamos un usuario administrador con rol de administrador ya que no había ninguno con ese rol. Username: ‘administrador’. Password: ‘administrador’.
Creamos al usuario administrador general con un sólo rol que tiene todas las funcionalidades. Username: 'admin' Password: ‘'w23e'

Publicaciones

Al migrar se calculan tanto su cantidad de ubicaciones totales como sus ubicaciones vendidas para mantener consistente el modelo a cómo lo usaremos luego de migrar. Adicionalmente, se marcarán como finalizadas las que hayan vendido todas sus ubicaciones.
Se les está asignando a las publicaciones el grado “Alta” ya que las comisiones correspondientes a las compras facturadas que encontramos en la tabla maestra son aproximadamente del 10%. Además necesitamos este dato, ya que sino no se podrían realizar nuevas compras de esas publicaciones ni luego poder rendirlas.

Rubros

Consideramos que un string vacío no representa un valor válido como un rubro. Por esto todos los espectáculos con un rubro sin descripción fueron migrados como sin rubro. Esto nos facilita poder encontrarlos más tarde cuando se realicen búsquedas al comprar. Ya que si no al filtrar por rubro no nos sería posible encontrar estos espectáculos.
Como no tenemos ABM de rubro agregamos algunos rubros como ejemplo para poder ser utilizados desde la aplicación y generar publicaciones con esos rubros.

Compras y ubicaciones

Si bien detectamos que por los datos que había en la tabla maestra la relación compra-ubicación era 1 a 1. Nos parece importante migrar esta relación con la FK del lado

de ubicación hacia compra para permitir posteriormente que las compras tengan muchas ubicaciones. Y además restringir a que una ubicación sea comprada una sóla vez.

Puntos

Generamos los puntos correspondientes a las compras que había en el sistema antes de la migración, utilizando el mismo algoritmo con el que se generarán posteriormente (cantidad total de compra / 3). Y también registrando su vencimiento dentro de 3 meses.

Clientes y empresas

Se registra su fecha de creación como la fecha en la cual se realiza la migración.

Clientes

Registramos su tipo de documento como DNI, ya que en la tabla maestra el nombre de la columna se llama *Cli_Dni* por lo que asumimos que se trataban de DNIs.

Consideraciones generales sobre las funcionalidades implementadas en el TP

Roles y Funcionalidades

Un rol es un conjunto de funcionalidades. Las funcionalidades se corresponden con los distintos ABMs que se desarrollan en la aplicación.
No modelamos al registro de usuario como una funcionalidad ya que todos pueden registrarse. Y en realidad quien se está registrando todavía no es un usuario del sistema, por lo que no tiene sentido que un usuario pueda tener la funcionalidad de registrarse.
Las funcionalidades disponibles para un usuario van a ser la unión de las funcionalidades de todos sus roles.

Cuando una empresa o un cliente son eliminados no se les va a permitir acceder a las pantallas de sus funcionalidades tanto de empresa como cliente. Pero eso no los hace estar eliminados a nivel usuario, ya que como un usuario puede tener varios roles, podríamos estar imposibilitando su acceso a otros roles.

Cuando un usuario que tenga las funcionalidades que son referidas a un cliente o a una empresa pero su rol no es de cliente o empresa. Entonces se le solicitará que elija el cliente o empresa para el cual querrá realizar la funcionalidad. Se considera que los roles con permisos sobre funcionalidades que son propias de ser realizada por un cliente o una empresa como comprar o publicar respectivamente, al no tener una empresa o cliente asociado, tienen los permisos para realizar esta acción en nombre de cualquier empresa/cliente del sistema.

Por ejemplo: un administrador general (que tiene todas las funcionalidades) podría querer listar y modificar las publicaciones de una empresa o comprarle una entrada a un cliente.

Si entre todos los roles de un usuario suman una sola funcionalidad, el usuario accede directamente a ella.

Cambio de clave

A los usuarios se les ofrece la opción de cambiar su contraseña. Sin embargo, no es posible que cambien su username.

ABM Clientes

Los clientes se guardan en la tabla de clientes. Cada cliente está asociado a un usuario del sistema (FK).
No se permite tener una eliminación física de los clientes. Pueden ser tanto deshabilitados como habilitados por un administrador.

Sobre las tarjetas

Un cliente sólo va a poder tener una tarjeta “activa” al mismo tiempo. Con esta tarjeta es con la cual realizará sus compras.
No se permite ni la modificación ni la actualización de los registros de la tabla tarjeta, ya que podrían haberse ya realizado compras con esa tarjeta y necesitamos conservar esa información.
Por eso, para que el cliente pueda cambiar los datos de su tarjeta. Se creará un nuevo registro en la tabla de tarjetas, el cual será su nueva tarjeta actual. Sin embargo, se siguen conservando los datos de tarjetas anteriores.
Cuando un usuario administrativo crea un cliente, este es asociado automáticamente a un usuario cuyo username será su dni y su password será autogenerada e informada al administrador para que este se la comunique al cliente.

Validaciones al crear un cliente

Se obliga a ingresar todos los datos marcados con un asterisco. El piso y departamento son opcionales ya que consideramos que alguien podría vivir en una casa (y no tener ni piso ni departamento).
Se valida el formato del email.
Se valida el formato de CUIL con 2 dígitos seguidos de un guión seguidos de 8 dígitos, seguidos de otro guión y otro dígito más.
Se valida la correspondencia de CUIL y DNI, con la salvedad de que no podemos validar los dígitos verificadores del CUIL ya que no tenemos el dato del género del cliente pero tampoco parece relevante en el sistema.

Al modificar un cliente, si hubieran faltantes de estos datos o inconsistencias, se obligará al administrador que lo modifique a rectificarlos.

ABM de Empresas

Cada empresa está asociada con usuario de la tabla de usuarios.
Cuando un administrador da de alta una empresa se autogenera su usuario con el CUIL de la empresa como username y un password generado aleatoriamente. Se le informa al administrador por pantalla esta situación.

Validaciones al crear una empresa

Se deben ingresar todos los campos marcados con asterisco. El piso y departamento son opcionales por la misma razón que la de los clientes.
Se que el e-mail tenga un formato válido.
Se valida que el CUIT tenga el formato *2 dígitos - 8 dígitos - 1 dígito*

Al modificar una empresa, si hubieran faltantes de estos datos o inconsistencias, se obligará al administrador que lo modifique a rectificarlos.

Publicaciones

Para la tabla de publicación y el modo en que el cliente genera las publicaciones

Para que una empresa pueda generar su publicación va a tener que ingresar que ubicaciones va a tener su espectáculo.
Como sería muy difícil que un usuario del sistema pueda ingresar manualmente todas y cada una de las ubicaciones, creemos que la mejor forma sería dividir las ubicaciones por sectores.
Estos sectores pueden ser tanto numerados como no numerados.
En los sectores numerados el usuario debe especificar el rango de filas (ej: ‘A’ a ‘M’) y de asientos (ej: 1 al 10).
También puede elegir un sector no numerado. En el que va a tener que especificar cuántas ubicaciones tiene dicho sector.
Para ambos tipos de sectores debería especificar tanto el precio como qué tipo de ubicaciones tiene el sector.
Esto permitiría definir fácilmente al usuario qué ubicaciones quiere tener. Y hacerlo de manera flexible ya que de esta forma al tener varios sectores puede tener para cada sector un precio y un tipo de ubicación distinto.
Cuando el usuario publica su Publicación, entonces en ese momento se generan todos los registros de la tabla de ubicaciones a partir de los sectores definidos por el usuario.
Todos se van a generar como “no comprados” y desde ese momento van a estar disponibles para comprar y no se van a poder volver a modificar por el usuario empresa.

Otras consideraciones

Como tenemos las publicaciones borradores en la misma tabla que las “definitivas” no podemos hacer chequeos de not null. Pero sin embargo, podemos chequear que la publicación sea consistente en el momento en el que pasa al estado publicado. Ya que luego no será modificada.

Además el porcentaje de comisión a cobrar se guarda en la publicación al generarse. Ya que este podría ser cambiado en el futuro siendo que el cliente ya había aceptado que le cobren ese porcentaje (o peor, incluso después de que hayan sido realizadas compras de esa publicación). Por esto no se lo va a permitir modificar.

Se valida mediante un trigger en la tabla de publicaciones que no puedan cambiar de estado de finalizada a cualquier otro, o de publicada a borrador.

Los datos que se ven marcados con un asterisco en el form son necesarios para publicar pero no necesarios para guardarla como borrador, ya que vemos que tiene sentido permitir guardar datos temporalmente inconsistentes si damos la posibilidad de que haya un borrador.

Las publicaciones se generan sin fecha de vencimiento, ya que se considera que su vencimiento será el momento en que realicen su pasaje automático de *Publicada* a *Finalizada* tras haber finalizado su stock o que la empresa lo haya marcado.

Se considera que el usuario responsable de la publicación es el usuario correspondiente a la empresa creadora de la publicación. No se guarda este atributo en la publicación porque es calculable con joins y no se perderá ya que ni el usuario ni la empresa tienen bajas físicas. Además no vemos sentido a mantenerlo denormalizado en la publicación ya que no es algo por lo cual se consulte luego.

Si bien se guarda en la tabla de espectáculos el código de espectáculo que estaba en la tabla maestra para no perder dicha información, este no se sigue utilizando para realizar consultas sino que se usa una PK numérica autoincremental para así mantener consecutivas las numeraciones.

ABM grado

Cada registro de la tabla de grados tiene su descripción, porcentaje de comisión, id y bit de eliminación lógica.
Cuando un grado está dado de baja lógicamente no se va a permitir que se den de alta nuevas publicaciones con dicho grado, ya que no se muestran en la selección acotada que se realiza al dar de alta una publicación.
Sin embargo las publicaciones que ya han sido publicadas y cuyas entradas ya están a la venta conservarán el grado. Esto es porque al ya estar publicadas no se pueden modificar

y, además, la empresa de espectáculos ya había seleccionado el grado y comisión que estaba dispuesta a pagar y sería injusto modificarlo, no al menos sin informarle.
El porcentaje de comisión va a ser el indicador para el orden en que se mostrarán las publicaciones cuando un cliente va a comprar. Mayor porcentaje de comisión implica mayor prioridad de visualización.

Canje de Puntos

Cada vez que un cliente realiza una compra, este gana un punto por cada $3 gastados. Los puntos ganados por el cliente se registran en la tabla de puntos, guardando la fecha en que fueron adquiridos (fecha_creacion), la cual es la fecha en que se realizó la compra.
Se guarda una fk a la compra para no perder la información de a qué compra correspondieron los puntos ganados, pero se guarda denormalizada la fecha para evitar tener que hacer un join con la tabla de compras cuando un cliente consulta sus puntos. Y también se guarda la futura fecha de vencimiento para no tener que calcularla luego.
Los premios disponibles se registran en la tabla de PremiosDisponibles, en la que figura cuántos puntos son requeridos para adquirir ese premio.
Los puntos actuales de un clientes son aquellos puntos que no haya gastado y cuya fecha de creación sea de no más de hace 3 meses.
Cuando un cliente canjea un premio, se añade un registro a la tabla de Canje y se le descuentan puntos empezando por el registro de puntos de mayor antigüedad.
Dentro de cada entrada de la tabla Puntos se guardan tanto los puntos iniciales como los puntos gastados. Los disponibles entonces serán los iniciales menos los gastados.
Entre Canje y Puntos se agrega una tabla intermedia ya que para un canje se pueden requerir los puntos sumados a lo largo de varios registros de la tabla de Puntos y un mismo registro de la tabla de puntos podría participar de más de un canje.

- Si bien la cant_usada de puntos en la tabla de puntos es calculable sumando los puntos_canjeados de los registros de Puntos_Canje que corresponden a ese registro de puntos, mantenemos este atributo denormalizado ya que ahorra la necesidad de hacer un join para calcular los puntos de un cliente. Además, para mantenerlo consistente, sólo se necesita modificarlo al momento de realizar un canje.

Compras

Cuando un cliente realiza un compra, se crea un registro en la tabla de compras.
Este está relacionado con las ubicaciones que compró, con el cliente y con la tarjeta que usó para realizar la compra.
Esto último es por si el cliente cambia la tarjeta luego de realizar la compra.

Al realizar la compra, se verifica que las publicaciones afectadas sigan teniendo stock de ubicaciones. Si se agotaron sus ubicaciones su estado pasa a finalizado.

Además, luego de comprar el cliente suma puntos relacionados a esa compra.

Consideraciones para las publicaciones mostradas al momento de comprar

No mostramos ni borradores ni publicaciones finalizadas, ya que no vemos sentido en mostrar publicaciones de las cuales no se puede comprar entradas en una funcionalidad de comprar.

Las publicaciones se muestran ordenadas por el porcentaje de comisión de su grado, ya que consideramos que esto es su grado de visibilidad o “peso”. Adicionalmente, tuvimos en cuenta la fecha del espectáculo como segundo criterio de ordenamiento, para así mostrar los más próximos primero para facilitar su búsqueda al usuario.

Como no tenemos un proceso automático que finalice las publicaciones por tiempo (sí por agotar stock) y además como en la tabla maestra las publicaciones tienen fecha de vencimiento y consideramos la fecha actual a lo que esté configurado en la app. Entonces estamos ocultando las publicaciones que tengan fecha de vencimiento anterior a la fecha de la aplicación. No podemos marcar publicaciones como finalizadas por la fecha en la migración ya que no sabemos qué fecha se utilizará en la app.

Además, tampoco se mostrarán al comprar las publicaciones cuya fecha de espectáculo sea menor a la fecha actual del sistema (app.config), ya que un cliente no podría comprar entradas para un espectáculo que ya ocurrió. Y tampoco se permite en la búsqueda ingresar una fecha anterior a la actual.

Listado estadístico

Para facilitar el listado de las publicaciones menos vendidas se realizó una denormalización en la tabla de publicaciones en la que se mantiene las ubicaciones compradas y las ubicaciones totales de una publicación

Rendición de comisiones

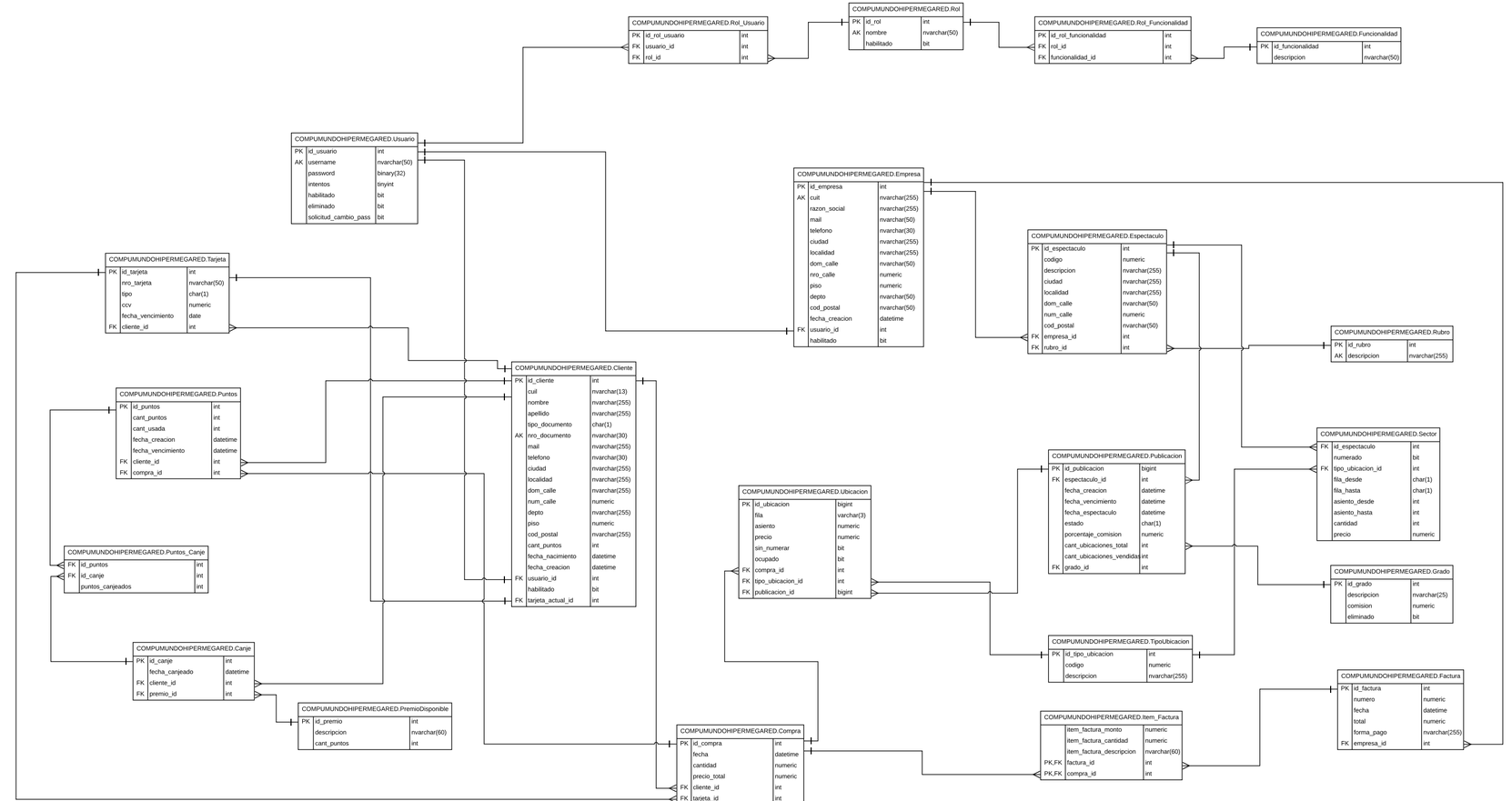
Para rendir comisiones primero se solicita al usuario que elija de qué empresa quiere rendir comisiones y de cuántas compras quiere rendirlas. Luego se muestra qué es lo que se va a rendir y se solicita su confirmación. Si la empresa seleccionada no tuviera compras a rendir se le informará al usuario y si tuviera menos compras a rendir que las solicitadas se le dará la opción de rendir menos compras o cancelar la operación.
Las compras ya facturadas no se pueden volver a facturar y no son mostradas cuando se solicita rendir comisiones. Esto se verifica ya que para las compras facturadas, los registros de compras están relacionados con los registros de la tabla item_factura.
Como para generar facturas es importante que los números sean consecutivos, al hacerlo se calcula siguiente del máximo existente en la tabla. Además se valida la unicidad con una constraint unique.

Habilitación/Inhabilitación de Usuarios

Esta funcionalidad fue agregada para considerar la posibilidad de que un administrador pueda volver a habilitar a un usuario que inició mal 3 veces sesión.
Y además para que pueda dar de baja o volver a habilitar a cualquier usuario del sistema independientemente de esto último.
Separamos esto de la baja de clientes y empresas, ya que de esta forma se permite dar de baja a usuarios que tengan rol que no sea ni cliente o empresa.

Configuración de fecha y hora del sistema

La configuración de fecha y hora del sistema se encuentra en el app.config. La key es *System_DateTime* y el formato requerido para cargar la fecha y hora es *dd/MM/yyyy HH:mm:ss*.


COMPUMUNDOHIPERMEGARED.Rol_Usuario
PK id_rol_usuario int
FK id_usuario_id int
FK rol_id int
COMPUMUNDOHIPERMEGARED.Rol
PK id_rol int
AK nombre nvarchar(50)
habilitado bit
COMPUMUNDOHIPERMEGARED.Rol_Funcionalidad
PK id_rol_funcionalidad int
FK rol_id int
FK funcionalidad_id int
COMPUMUNDOHIPERMEGARED.Funcionalidad
PK id_funcionalidad int
descripcion nvarchar(50)
COMPUMUNDOHIPERMEGARED.Usuario
PK id_usuario int
AK username nvarchar(50)
password binary(32)
intentos tinyint
habilitado bit
eliminado bit
solicitud_cambio_pass bit
COMPUMUNDOHIPERMEGARED.Empresa
PK id_empresa int
AK cuit nvarchar(255)
razon_social nvarchar(255)
mail nvarchar(50)
telefono nvarchar(30)
ciudad nvarchar(255)
localidad nvarchar(255)
dom_calle nvarchar(50)
nro_calle numeric
piso numeric
depto nvarchar(50)
cod_postal nvarchar(50)
fecha_creacion datetime
FK usuario_id int
habilitado bit
COMPUMUNDOHIPERMEGARED.Espectaculo
PK id_espectaculo int
codigo numeric
descripcion nvarchar(255)
ciudad nvarchar(255)
localidad nvarchar(255)
dom_calle nvarchar(50)
num_calle numeric
cod_postal nvarchar(50)
FK empresa_id int
FK rubro_id int
COMPUMUNDOHIPERMEGARED.Rubro
PK id_rubro int
AK descripcion nvarchar(255)
COMPUMUNDOHIPERMEGARED.Tarjeta
PK id_tarjeta int
nro_tarjeta nvarchar(50)
tipo char(1)
ccv numeric
fecha_vencimiento date
FK cliente_id int
COMPUMUNDOHIPERMEGARED.Cliente
PK id_cliente int
cuil nvarchar(13)
nombre nvarchar(255)
apellido nvarchar(255)
tipo_documento char(1)
AK nro_documento nvarchar(30)
mail nvarchar(255)
telefono nvarchar(30)
ciudad nvarchar(255)
localidad nvarchar(255)
dom_calle nvarchar(255)
num_calle numeric
depto nvarchar(255)
piso numeric
cod_postal nvarchar(255)
cant_puntos int
fecha_nacimiento datetime
fecha_creacion datetime
FK usuario_id int
habilitado bit
FK tarjeta_actual_id int
COMPUMUNDOHIPERMEGARED.Puntos
PK id_puntos int
cant_puntos int
cant_usada int
fecha_creacion datetime
fecha_vencimiento datetime
FK cliente_id int
FK compra_id int
COMPUMUNDOHIPERMEGARED.Puntos_Canje
FK id_puntos int
FK id_canje int
puntos_canjeados int
COMPUMUNDOHIPERMEGARED.Canje
PK id_canje int
fecha_canjeado datetime
FK cliente_id int
FK premio_id int
COMPUMUNDOHIPERMEGARED.PremioDisponible
PK id_premio int
descripcion nvarchar(60)
cant_puntos int
COMPUMUNDOHIPERMEGARED.Ubicacion
PK id_ubicacion bigint
fila varchar(3)
asiento numeric
precio numeric
sin_numerar bit
ocupado bit
FK compra_id int
FK tipo_ubicacion_id int
FK publicacion_id bigint
COMPUMUNDOHIPERMEGARED.Publicacion
PK id_publicacion bigint
FK espectaculo_id int
fecha_creacion datetime
fecha_vencimiento datetime
fecha_espectaculo datetime
estado char(1)
porcentaje_comision numeric
cant_ubicaciones_total int
cant_ubicaciones_vendidas int
FK grado_id int
COMPUMUNDOHIPERMEGARED.Sector
FK id_espectaculo int
numerado bit
FK tipo_ubicacion_id int
fila_desde char(1)
fila_hasta char(1)
asiento_desde int
asiento_hasta int
cantidad int
precio numeric
COMPUMUNDOHIPERMEGARED.Grado
PK id_grado int
descripcion nvarchar(25)
comision numeric
eliminado bit
COMPUMUNDOHIPERMEGARED.TipoUbicacion
PK id_tipo_ubicacion int
codigo numeric
descripcion nvarchar(255)
COMPUMUNDOHIPERMEGARED.Compra
PK id_compra int
fecha datetime
cantidad numeric
precio_total numeric
FK cliente_id int
FK tarjeta_id int
COMPUMUNDOHIPERMEGARED.Item_Factura
item_factura_monto numeric
item_factura_cantidad numeric
item_factura_descripcion nvarchar(60)
PK,FK factura_id int
PK,FK compra_id int
COMPUMUNDOHIPERMEGARED.Factura
PK id_factura int
numero numeric
fecha datetime
total numeric
forma_pago nvarchar(255)
FK empresa_id int