UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

GESTIÓN DE DATOS

Trabajo Práctico

2° Cuatrimestre 2019

Frba Ofertas

Integrantes:

Reynosa, León Lautaro
Vetere Portillo, Giuliana Nicole
Olivera, Lautaro Agustín
Kim, Brian Doyel Ezequiel

UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Índice

Estrategia

Migración

Para la migración, primero analizamos los datos que nos proveía el sistema viejo en la tabla “maestra”, que con conjunto a los requisitos del nuevo sistema, nos permitieron identificar las diferentes tablas. Cabe destacar, que para migrar los datos, no hicimos uso de ningún objeto de base de datos (triggers, funciones, procedures, vistas).
Para comenzar con la migración, empezamos con los datos y las tablas más básicas. En orden de ejecución según el script (ver comentario “Inserts”, a partir del cual empieza la migración):

- **Rubro:** Entidad básica que no depende de ninguna otra entidad, que especifica a qué tipo negocio se dedica el proveedor. Seleccionamos todos los rubros haciendo un “Select DISTINCT provee_rubro”, para obtener los rubros distintos, sin repeticiones.
- **Rol:** Se cargaron solo 3 roles: Proveedor, Cliente y Administrativo
- **Proveedor:** Hicimos un “Select DISTINCT” de la tabla maestra, para seleccionar las diferentes combinaciones de datos de los proveedores, considerando que cada combinación diferente de estos (“Provee_CUIT as cuit, Provee_Dom as direc, Provee_Ciudad as localidad”), implica un nuevo proveedor. Además, le asignamos el rubro previamente cargado en el nuevo sistema usando un sub-select (y cada uno tiene un único rubro).
  Para completar los datos restantes del proveedor, hicimos uso de un cursor para completar la tabla **“Dirección”** con la dirección del proveedor, asignando el cursor a un select con los datos de la dirección más el CUIT del proveedor para identificarlo unívocamente dentro de nuestro sistema, a la hora de persistir los datos. Asi, recorremos y por cada fila, creamos la dirección y su asignación a un proveedor.
  Adicionalmente, por cada iteración del cursor, creamos un **usuario** a cada proveedor, ya que el sistema viejo no contaba con esta funcionalidad, poniendo como usuario y contraseña su propio CUIT, aplicando función hash “SHA-256” a las contraseñas para mayor seguridad. Por último, le asignamos a cada usuario proveedor, su respectivo Rol, usando una tabla intermedia **(RolXUsuario)** entre un usuario y el rol, para romper la relación muchos a muchos (que podría contemplarse a futuro, siendo que podría ser posible que un proveedor pueda ser cliente en un futuro).
- **Cliente:** Al igual que con el proveedor, hicimos un “Select DISTINCT” a los distintos datos del cliente (nombre, apellido, dni, mail unico, teléfono único, fecha de nacimiento). Asignamos también, su respectivo usuario (usando como usuario y contraseña el dni, hasheando la contraseña con SHA-256), rol y dirección. A la hora de persistir el monto, decidimos que todos los usuarios nuevos en el sistema empiezen con 200 de saldo, ya sean que sean

nuevos registrándose(como pedía el enunciado) o que estén migrando del sistema viejo.

- **Tipo Pago:** Consideramos importante hacer una entidad para los tipos de pago, para ser más flexibles a la hora de elegir diferentes métodos de pago. Se identificaron 2 que son efectivo y crédito.
- **Carga:** Por cada carga y respectiva fecha, que aparecían en la tabla "Maestra" (consideramos que cada carga representaba una fila correspondiente donde no fuera null esos campos), persistimos esos datos, utilizando dos sub-select para obtener el código, en nuestro nuevo sistema, del cliente que la realizó; y el código del tipo de pago.
- **Oferta:** Consideramos que cada "Oferta_codigo" diferente en la tabla "Maestra" del sistema viejo, identifica unívocamente a cada oferta realizada, indistintamente de la similitud de descripciones, y caracteres en el código, que en muchos casos solo se diferenciaban en el último número del código. Cada uno de estos códigos los persistimos, considerando que el atributo "Oferta_cantidad" hacía referencia a la "cantidad máxima que podía comprar un usuario de esa oferta" y "el stock disponible", por lo tanto asignando a la cantidad (disponible) de cada oferta, un valor igual a la cantidad máxima.

- **Factura:** Seleccionamos de la tabla maestra las diferentes facturas realizadas buscando los distintos números de factura y sus respectivas fechas. Además, hicimos uso de dos subselect. El primero para sacar el codigo del proveedor en nuestro nuevo sistema, y el segundo para sacar el total de la factura, calculandolo con los datos de la tabla maestra, como la sumatoria del precio de la oferta, de las filas que tuvieran al número de factura requerido (cláusula where).
- **Cupón:** Consideramos que cada fila que tuviera una oferta_fecha_de_compra era una compra, con una cantidad de 1 producto, ya que tomamos el Oferta_cantidad como la cantidad máxima que se podía comprar de dicha oferta (como se dijo en la sección oferta). Hicimos uso de un subselect en el from para facilitar los datos, teniendo solo la fecha de compra, el código de la oferta y el dni del cliente. Con el código y el dni hicimos dos subselect para encontrar dicho cliente y oferta dentro del nuevo sistema. Además, hicimos un tercer subselect para asignarle la correspondiente factura (si es que se facturó el cupón)
- **Entrega:** Hicimos un select de la "Oferta_entrega_fecha", y un subselect para encontrar el cupón entregado dentro del nuevo sistema, comparando los datos de la tabla maestra con los datos de la nueva tabla correspondiente a cupón.

Adicionalmente, agregamos:

- **Funcionalidad**: Ingresamos las distintas funcionalidades que tiene el sistema, como comprar oferta, facturar a proveedor, entregar oferta, entre otros. Cada una de estas, puede pertenecer a uno o más roles. Para esto hicimos la tabla intermedia **"FuncionalidadXRol"** para romper la relación muchos a muchos entre funcionalidad y rol, cada dato representa una funcionalidad dentro de un rol.

- **Usuario administrador:** Agregamos un usuario administrador bajo el usuario y contraseña “admin”. No permitimos que se registren más usuarios administradores, siendo este el único. Le agregamos su correspondiente funcionalidades.

Aplicación Desktop

En cuanto a la aplicación desktop, en el cual utilizamos el lenguaje de programación C# con Visual Studio 2012 y Framework .NET 4.5. Nos encargamos del diseño y la usabilidad del sistema. Para dicho desarrollo nos encargamos de las siguientes pantallas:

- **Login**
- **Registro de Usuarios**
- **Menú de Inicio**
- **ABM Roles**
  - **Alta Rol**
  - **Listado Roles**
  - **Modificar Roles**
- **ABM Clientes**
  - **Alta Cliente**
  - **Listado Clientes**
  - **Modificar Clientes**
- **ABM Proveedores**
  - **Alta Proveedores**
  - **Listado Proveedores**
  - **Modificar Proveedores**
- **Confección y publicación de Ofertas**
- **Compra de Ofertas**
  - **Listado de Ofertas**
  - **Compra Oferta**
- **Carga de Saldo**
- **Listados Estadísticos**
- **Facturación Proveedor**
  - **Listado de Facturación**
  - **Facturación a Proveedor**
- **Entrega/Consumo de Oferta**
- **Modificación de Contraseña**
- **Administración de Usuarios**
  - **Selección Usuario**
  - **Modificación Contraseña**

Login:
El login el la primera pantalla que nos encontramos cuando iniciamos el sistema, en ella tendremos la opción de *Registrar Usuario* para crear uno nuevo, o de ingresar un username ya existente y su respectiva contraseña para así dirigirnos a la pantalla de *Menú Inicio.*
Cabe destacar que si no ingresamos un usuario y contraseña registrado en el sistema, no tendremos el acceso al menú, y si un usuario ingresa tres veces mal su contraseña, este será inhabilitado.

Los Clientes y Proveedores que existían previos a la migración, tendrán disponible un Usuaria y contraseña, el cual será :

Usuario: DNI y Contraseña: DNI para aquellos usuarios que eran Clientes.
Usuario Cuit y Contraseña: CUIT para aquellos usuarios que eran Proveedores.

Registro de Usuarios:
La pantalla de registro de usuarios se mostrará cuando en la pantalla de *Login* se seleccione la opción “Registrar Usuario” o cuando un usuario administrador decida registrar un nuevo usuario gracias a sus funcionalidades. Este se compone principalmente de dos pantallas, *Registro Usuario* y de *Alta Cliente* ó *Alta Proveedor* . Primero se ingresará un usuario que no exista en el sistema aún y luego se deberá seleccionar entre uno de los dos roles para redireccionar al alta correspondiente al rol elegido, donde se ingresarán los datos completos.

Menú de Inicio:
Una vez ingresado nuestro usuario y contraseña nos encontraremos con la pantalla de *Menú de Inicio* donde habrá una botonera correspondientes a las diferentes funcionalidades existentes en el sistema, y dependiendo del usuario que se ingreso, tendrá diferentes funcionalidades con respecto al rol/es que posea. Aquellas funcionalidades que no le correspondan al usuario no se mostrarán visibles ni se tendrá acceso.

ABM Roles:
La ABM (explicación detallada de ABM en la guía de ABM de la cátedra) de roles nos permite manipular los roles existentes en el sistema, esto se refiere a que podremos crear nuevos roles, modificarlos y eliminarlos. Esta funcionalidad en un principio le pertenece al rol administrativo y se compone de tres pantallas para una mejor usabilidad;

- Listado de Roles: Esta pantalla nos permite ver los distintos roles existentes en el sistema y buscarlos mediante distintos filtros para así eliminarlo o modificarlo. Cabe destacar que al eliminar un rol este realizará una baja lógica y esto ocasionará que todos aquellos usuarios que tenían dicho rol, ya no lo tengan.
- Alta Rol: Nos permite crear un nuevo rol, ingresando las funcionalidades que va a poseer y su nombre característico.
- Modificación Rol: Una vez seleccionado un rol se permitirá modificar tanto sus funcionalidades como su nombre desde esta pantalla.

ABM Clientes:
La ABM de clientes nos permite manipular los roles existentes en el sistema, esto se refiere a que podremos crear nuevos clientes, modificarlos y eliminarlos. Esta funcionalidad en un principio le pertenece al rol administrativo y se compone de tres pantallas para una mejor usabilidad;

- Listado de Clientes: Esta pantalla nos permite ver los distintos clientes existentes en el sistema y buscarlos mediante distintos filtros para así eliminarlo o modificarlo. Hay que tener en cuenta que la eliminación de un cliente es lógica, y no física, por lo que una vez eliminado, su campo de “clie habilitado” estará en 0.

- Alta Cliente: Nos permite crear un nuevo cliente, pero para esto, primero deberemos elegir un usuario que aun no sea cliente, una vez seleccionado tendremos que ingresar su información correspondiente.
- Modificación Cliente: Una vez seleccionado un cliente se permitirá modificar su información.

ABM Proveedores:
La ABM de proveedores nos permite manipular los proveedores existentes en el sistema, esto se refiere a que podremos crear nuevos proveedores, modificarlos y eliminarlos. Esta funcionalidad en un principio le pertenece al rol administrativo y se compone de tres pantallas para una mejor usabilidad;

- Listado de Proveedores: Esta pantalla nos permite ver los distintos proveedores existentes en el sistema y buscarlos mediante distintos filtros para así eliminarlo o modificarlo. Hay que tener en cuenta que la eliminación de un proveedores lógica, y no física, por lo que una vez eliminado, su campo de “prov_habilitado” estará en 0.
- Alta Proveedores: Nos permite crear un nuevo proveedor, pero para esto, primero deberemos elegir un usuario que aun no sea proveedor, una vez seleccionado tendremos que ingresar su información correspondiente.
- Modificación Proveedor: Una vez seleccionado un proveedor se permitirá modificar su información.

Confección y Publicación de ofertas:
La confección y publicación de ofertas en principio es una de las funcionalidades pertenecientes al Proveedor. Se encarga principalmente de crear Ofertas para que así, los usuarios que sean del tipo cliente puedan comprarlas. Se deberá completar la oferta con una descripción de la misma, un precio de oferta, precio de lista, fecha de publicación, fecha de vencimiento, cantidad disponible y cantidad máxima para comprar. Una vez ingresado los campos se creará la oferta y los usuarios clientes podrán comprarla.

Compra de Ofertas:
La compra de ofertas es una funcionalidad que inicialmente corresponde a los Clientes, está se encargara de comprar las ofertas disponibles en la base de datos siempre y cuando todavía no se hayan vencido, teniendo en cuenta la fecha estipulada en el archivo de config. Para la compra de ofertas se dispone de dos pantallas:

- Listado de Ofertas: En el listado de ofertas nos encontraremos con todas las ofertas comprendidas entre la fecha de publicación de la oferta y la fecha de vencimiento, teniendo en cuenta la fecha del archivo de config. Se deberá seleccionar una para luego pasar a la pantalla de *Compra Oferta.*
- Compra Oferta: En la pantalla de compra oferta podremos ingresar la cantidad de esa oferta que queramos comprar, siempre y cuando no supere la cantidad disponible que haya de la misma.

Carga de Saldo:
La carga de saldo en un principio corresponde a las funcionalidades del Cliente, esta pantalla es la encargada de poder incrementar el saldo correspondiente al Cliente, para esto

se deberá elegir el método de pago (crédito o efectivo) y el monto a cargar. Si se eligiera “Crédito” como método de pago, se deberá ingresar la información de la tarjeta. Una vez con todos los campos ingresado, y aceptando, se procederá a crear una nueva fila en la tabla “Carga” y se le subirá el saldo al cliente que cargó.

Listados Estadísticos:
En la pantalla de listados estadísticos, tendremos que ingresar el año el cual queremos evaluar, el semestre , y tendremos la opción de elegir entre dos tipos de listados diferentes; “Mayor porcentaje descuento ofrecido en las ofertas” o “Mayor facturación”. Una vez completado estos campos se pondrá aceptar y se podrá ver el top 5 de los Proveedores para cada tipo de listado.

Facturación a Proveedor:
La facturación al proveedor es la funcionalidad encargada de generar las facturas a un proveedor determinado comprendiendo dos fechas. Esta funcionalidad corresponde principalmente al usuario administrativo. Se compone de dos pantallas para su uso:

- Facturación a Proveedor: Para esta parte se ingresaran dos fechas en las cuales va a comprender la facturación, y se escogerá un Proveedor de los que haya en la base de datos. Una vez elegido pasará a la siguiente pantalla
- Listado de Facturación: Una vez seleccionados los datos anteriores, se mostrarán los cupones que se facturará. Al ingresar generar, la facturación habrá terminado y estarán disponibles las facturas en la base de datos.

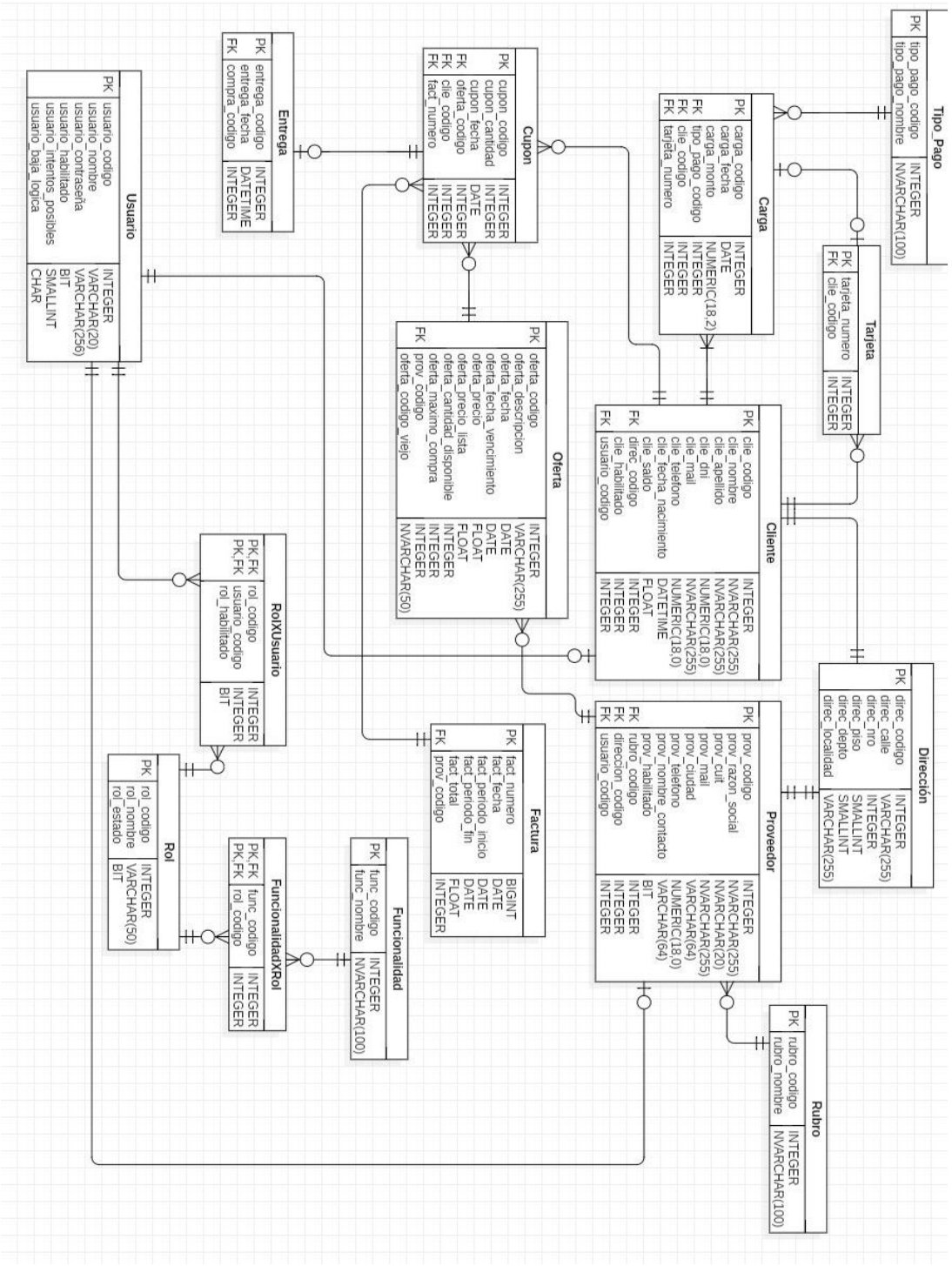
Entrega/Consumo de Oferta:
Para esta sección, nos encargaremos de consumir un cupón que un Cliente haya comprado para este proveedor, se deberá ingresar el codigo del cupon y si el cupón corresponde al proveedor, este se consumirá y se generará su respectiva factura.

Modificación de Contraseña:
Esta sección está habilitada para todos los usuarios, es para permitirnos poder modificar nuestra contraseña. Para esto, deberemos introducir la contraseña vieja que el usuario tenía e introducir una nueva para realizar el cambio.

Administración de Usuario:
Esta funcionalidad es exclusiva para los administrativos, dentro de esta tendremos dos pantallas:

- Selección de Usuario: Tendremos un listado para seleccionar un usuario que haya en el sistema. Una vez elegido podremos modificarlo.
- Modificar Contraseña: Una vez elegido el usuario podremos modificar su contraseña desde esta sección.

Modelo de Datos



Usuario

Entidad:
La entidad de *Usuario* es aquella que nos va a permitir los distintos logins que haya en el sistema, para esto cada usuario deberá loguearse con su respectivo username y contraseña, los cuales van a estar en la base de datos para verificar que coincidan. El username será un campo identificatorio único para el usuario.
Campos:

- usuario_codigo (Primary Key) : es el campo numerico identificativo del usuario, por lo cual cada usuario tendrá un código distinto. Elegimos que el código sea la primary key y no el usuario_nombre, debido a que este último va a ser un String, y no es recomendable que un string actúe como primary key.
- usuario_nombre: Este campo es para identificar al usuario por un nombre, este debe ser único para cada usuario y no se puede repetir.
- usuario contraseña: se refiere a la contraseña ingresada a la hora de crear un usuario, ésta se almacenará en la base de datos de forma encriptada utilizando el algoritmo de encriptación SHA 256.
- usuario_habilitado: Este campo es del tipo BIT, y nos permitirá saber si el usuario fue inhabilitado por un usuario administrador o por haber ingresado mal la contraseña 3 veces al intentar loguearse.
- usuario_intentos_posibles:

Procedures:

loginYSeguridad(@usuario varchar(20), @contraseña_ingresada varchar(256))
valor de retorno:
1 si el usuario está habilitado, e ingresó correctamente su usuario y contraseña (sin llegar a los 3 intentos fallidos). Puede acceder al sistema, y reinicia el contador de intentos.
0 si la contraseña ingresada es incorrecta. Se resta un intento de inicio de sesión.
-1 si no está habilitado o tuvo tres intentos fallidos de inicio de sesión.
-2 si el usuario no se encuentra registrado.

cambiarContraseniaUsuario(@usuario_id INT, @usuario_nueva_contrasenia varchar(256))
Este procedure le modifica la contraseña al usuario pasado por parámetro.

darDeBajaUsuario(@usuario_id INT)
Dar de baja un usuario implica su baja lógica por lo que se setea su char de baja lógica en ‘S’ (SÍ).

Funciones:

verificarUsuario(@usuario varchar(20)) RETURNS INT
Al momento de registrar el usuario, retorna:
0 si ya existe el usuario
1 si puede crearse

sosUsuarioAdministrador(@usuario_codigo int) RETURNS INT
En el registro, el valor de retorno será:
1 si el usuario tiene rol administrativo
-1 si no lo tiene

compararContrasenias(@usuario_id int, @contrasenia_prevista varchar(256)) RETURNS INT
Esta función verifica que la contraseña ingresada sea igual a la contraseña guardada anteriormente. Retorna:
1 si la contraseña es correcta
0 si no lo es

Cliente

Entidad:
La entidad Cliente es aquella que se encargará de persistir y organizar los Usuarios del tipo Cliente que existan en el sistema, este tendrá un código de cliente único e identificatorio el cual será su Primary Key, está se generará automáticamente al crear un nuevo Cliente. Éste tambien posee dos Foreign Keys, una hacia la entidad *Dirección* y otra hacia *Usuario,* siendo la entidad Dirección apuntada, la que contenga la información de donde ubicar al cliente y la entidad Usuario tendrá el usuario con el cual el Cliente se loguea.
Campos:

- clie_codigo(Primary Key): Es la clave identificativa del Cliente, para esto se autogenera el código cada vez que se ingrese un Cliente.
- clie_nombre: Es el nombre del Cliente, este es distinto al nombre de usuario.
- clie_apellido: Es el apellido del Cliente.
- clie_dni: Documento único de identidad del Cliente.
- clie_mail : Es el mail que nos brindará el Cliente.
- clie_telefono: Es el telefono de contacto brindado por el Cliente.
- clie_fecha_nacimiento: Fecha de nacimiento del Cliente.
- clie_saldo: Saldo vigente del Cliente, este se restará si se compra una oferta o sumará si se realiza una carga.
- clie_habilitado: Campo del tipo BIT que nos indica si el Cliente está habilitado.
- direc_codigo (Foreign Key): Este campo es la relación directa con una Dirección la cual contendrá la información de donde ubicar al Cliente.
- usuario_codigo (Foreign Key): Este campo es la relación con el usuario correspondiente con el Cliente, este nos ayudará para identificar las funcionalidades correspondientes al usuario y su información como cliente.

Procedures:

insertarCliente(@usuario_nombre VARCHAR(20), @usuario_contraseña VARCHAR(256), @clie_nombre NVARCHAR(255), @clie_apellido NVARCHAR(255), @clie_dni

NUMERIC(18,0), @clie_mail NVARCHAR(255), @clie_telefono NUMERIC(18,0), @clie_fecha_nacimiento DATETIME, @clie_saldo FLOAT, @direc_codigo INT)
Crea un nuevo cliente poniéndole como datos los que le son pasados por parámetro. Estos son nombre, contraseña, dni, mail, entre otros.

bajaLogicaCliente(@usuario_codigo_bajacliente INT)
Al realizar la baja lógica de un cliente, se cambia a 0 su bit de habilitado junto con el del rol por usuario.

modificarCliente(@cliente_codigo_modif INT, @clie_nombre_modif NVARCHAR(255), @clie_apellido_modif NVARCHAR(255) , @clie_dni_modif NUMERIC(18,0), @clie_mail_modif NVARCHAR(255), @clie_telefono_modif NUMERIC(18,0) , @clie_fecha_nac_modif DATETIME, @clie_habilitado BIT, @clie_calle_modif VARCHAR(255) , @clie_localidad_modif VARCHAR(255) , @clie_nro_modif INT , @clie_depto_modif SMALLINT , @clie_piso_modif SMALLINT)
Se realiza una modificación en los datos del cliente. Estos pueden ser su código, nombre, apellido, dni, mail, teléfono, fecha de nacimiento, etc.

Proveedor

Entidad:
La entidad proveedor se encarga de persistir los datos de los usuario que tenga el rol de proveedor. Cada proveedor tiene un código único con el que se lo identifica unívocamente (Primary key). Además posee los campos:

- prov_razon_social : Es el nombre del proveedor
- prov_cuit : clave que identifica inequívocamente a un proveedor
- prov_mail : decidimos que pueda tener solo un mail por proveedor
- prov_ciudad: decidimos desnormalizar este valor para tenerlo en el proveedor y a su vez en la entidad dirección, en caso de necesitarlo y agilizar consultas.
- prov_telefono: decidimos que pueda tener solo un teléfono por proveedor
- prov_nombre_contacto: nombre de contacto del proveedor
- prov_habilitado: Indica si el proveedor está habilitado o no
- rubro_codigo: Foreign key que referencia el rubro al cual se dedica el proveedor
- direc_codigo: Foreign key que referencia a la dirección que ingresó el proveedor
- usuario_codigo: Foreign key que referencia al usuario que es propietario de los datos de ese proveedor.

Procedures:

bajaLogicaProveedor(@usuario_codigo_bajaproveedor INT)
La baja lógica del proveedor implica que se cambia a 0 (inhabilitado) el bit de proveedor habilitado y el del rol por usuario.

modificarProveedor(@prov_codigo_modif INT, @prov_razon_social_modif NVARCHAR(255), @prov_cuit_modif NVARCHAR(20), @prov_mail_modif

NVARCHAR(255), @prov_telefono_modif NUMERIC(18,0) , @prov_nombre_contacto_modif VARCHAR(64), @prov_habilitado_modif BIT, @prov_calle_modif VARCHAR(255) , @prov_localidad_modif VARCHAR(255) , @prov_nro_modif INT , @prov_depto_modif SMALLINT , @prov_piso_modif SMALLINT)
Se encarga de modificar los datos del proveedor, tales como su código, razón social, cuit, mail, teléfono, nombre de contacto, entre otros. Esto sólo puede hacerlo el usuario con rol administrativo.

insertarProveedor(@usuario_nombre_prov VARCHAR(20), @usuario_contraseña_prov VARCHAR(256), @razon_social_prov NVARCHAR(255), @cuit_prov NVARCHAR(20), @mail_prov NVARCHAR(255),@ciudad_prov VARCHAR(64) , @telefono_prov NUMERIC(18,0), @nombre_contacto_prov VARCHAR(64), @rubro_codigo_prov INT, @direc_codigo_prov INT)
Crea un nuevo proveedor con los datos pasados por parámetro (nombre, contraseña, razón social, cuit, etc).

Función:

obtenerCodigoProveedor(@usuario_id INT)
Pasándole un id de usuario, me devuelve el código del proveedor asociado al mismo.

Dirección

Entidad:
Entidad que almacena las direcciones generadas por los usuarios del sistema, estos tendrán un campo identificatorio direc código el cual se genera automáticamente cuando se inserta una nueva dirección.
Campos:

- direc_codigo(Primary Key): Codigo identificatorio que que nos permite reconocer unequivocamente cada una de las dirección.
- direc_localidad: Campo que hace referencia a la localidad de la dirección.
- direc_calle: Campo que hace referencia a la calle de la dirección
- direc_nro: Campo numérico que hace referencia al número de la calle de la direccion.
- direc_piso: Campo numérico que determina si la dirección es en un departamento y determina en qué piso del departamento es.
- direc_depto: Campo numérico que determina si la dirección es en un departamento y determina qué número de departamento es.

Rol

Entidad:
Esta entidad es la encargada de almacenar principal el nombre y el estado de cada rol, pero también posee una Primary Key el cual es el rol_codigo para identificar inequívocamente. Esta información se puede modificar con la ABM de Roles.

Campos:

- rol_codigo (Primary Key) : Es el campo para identificar inequívocamente al Rol, este campo se autogenera una vez creado el Rol.
- rol_nombre: Es el nombre que se le asigna al rol para identificarlo.
- rol_estado: Este campo del tipo Bit, nos permitirá saber si el rol está habilitado, si se deshabilita un rol, se eliminaran todas las relaciones con los Usuarios de ese rol, en la tabla de **RolXUsuario**.

Procedure:

insertarRolNuevo(@rol_nombre VARCHAR(50))
Con este procedure se crea un nuevo rol con el nombre pasado por parámetro. Por default el bit de estado se pone en 1 y su código se autogenera.

eliminarRol(@codigo_rol_eliminar INT)
Eliminar un rol implica la baja lógica del mismo por lo que se pone en 0 su bit de estado pero sí se elimina el rol por usuario(en la tabla RolXUsuario).

insertarFuncionalidadPorRol(@func_codigo INT, @rol_codigo INT)
Se utiliza para agregar una funcionalidad a un rol determinado. Para eso se toma por parámetro la funcionalidad elegida y el rol al cual se la quiere agregar.

Trigger:

elimininacionRol ON S_QUERY.Rol
Al hacer un Update en la tabla Rol, y que este se inhabilite, poniendo su rol_estado en 0, se activará este Trigger para así eliminar toda aquella relación que haya entre los usuarios y el rol dado de baja, en la tabla **RolXUsuario**.

RolXUsuario

Entidad:
Esta entidad nos ayudará para romper la relación de muchos a muchos entre el Rol y el Usuario, ya que esta entidad nos permitirá identificar con certeza, qué rol corresponde a que usuario.

- rol_codigo(Primary Key, Foreign Key) : Nos permite saber que rol es el que se encuentra en la relación.
- usuario_codigo(Primary Key, Foreign Key): Nos permite saber que usuario es el que se encuentra en la relación.
- rol_habilitado: nos permite saber si esa relación de rol para ese usuario, se encuentra disponible.

Funcionalidad

Entidad:
Esta entidad nos persiste las distintas funcionalidades existentes en el sistema, cada una de estas tendrá una Primary Key la cual será para identificar cada funcionalidad. Las funcionalidades son asignadas/vinculadas con los roles mediante la tabla **FuncionalidadXRol** la cual fue creada para romper la relación muchos a muchos entre las funcionalidades y los roles.
Campos:

- func_codigo (Primary Key) : Este campo nos determina un código inequívoco para cada una de las funcionalidades, este será autogenerado con cada inserción de una funcionalidad.
- func_nombre: Campo que detalla el nombre de la funcionalidad.

FuncionalidadXRol

Entidad:
Esta entidad fue creada principalmente para romper la relación de muchos a muchos entre la tabla Funcionalidad y Rol. ya que un rol puede tener muchas funcionalidades y una funcionalidad puede estar en muchos roles. Para esta relación tendremos dos campos los cuales ambos actuarán como Primary Key y Foreign Key a la vez.
Campos:

- func_codigo (Primary Key, Foreign Key) : Este campo nos indica con cuáles de las funcionalidades se va a relacionar en el campo de rol_codigo.
- rol_codigo (Primary Key, Foreign Key): Este campo nos indica cual de los roles va a poseer la funcionalidad determinada por el campo func_codigo.

Rubro

Entidad:
Esta entidad determina los distintos rubros existentes en el sistema, en un principio habrá solo tres rubros distintos. Tendrá un campo identificatorio que actuará como Primary Key
Campos:

- rubro_codigo (Primary Key) : Campo identificatorio e inequívoco que se autogenera para identificar cada tipo de rubro.
- rubro_nombre : Campo String que permite asignarle un nombre al rubro determinado.

Ofertas

Entidad:
Esta entidad es la encargada de almacenar y organizar las ofertas existentes en el sistema y las que se creen a futuro, para ello contendrá un campo identificatorio con el código de la oferta, que se genera automáticamente y corresponde a su primary key. También contendrá el código del proveedor el cual generó la oferta, y este será una Foreign Key.
Campos:

- oferta_codigo(Primary Key): Es el campo identificatorio para cada oferta, este código se genera automáticamente cada vez que se inserte una nueva oferta
- oferta_descripcion: Es el detalle de cada oferta, describe brevemente la oferta y que producto oferta.
- oferta_fecha: Es la fecha de publicación de la oferta.
- oferta_fecha_vencimiento: Fecha de vencimiento de la oferta, una vez pasada esta, no se podrá ingerir dicha oferta.
- oferta_precio: Es el precio que se ofrece en la oferta por el producto detallado.
- oferta_precio_lista: Precio original del producto antes de la oferta.
- oferta_cantidad_disponible: Cantidad del stock disponible que hay para dicha oferta.
- oferta_maximo_compra: Cantidad máxima que el Usuario puede comprar por cada compra de oferta.
- oferta_codigo_viejo: Código que existía previamente , antes de la migración, este campo será null para aquellas ofertas ingresadas por el nuevo sistema.
- prov_codigo(Foreign Key): Foreign Key que hace referencia al proveedor que ingresó dicha oferta.

Procedures:

insertarOfertaNueva(@oferta_descripcion VARCHAR(255) , @oferta_fecha DATE , @oferta_fecha_vencimiento DATE , @oferta_precio FLOAT, @oferta_precio_lista FLOAT , @oferta_cantidad_disponible INT , @oferta_maximo_compra INT , @user_codigo INT)
Este procedure crea una nueva oferta con los valores que son pasados por parámetros.

ingresarEntregaOferta(@entrega_fecha DATETIME, @cupon_codigo INT, @usuario_codigo INT)

comprarOferta(@cupon_fecha_compra DATE , @cupon_cantidad_compra INT,@clie_codigo_compra INT , @oferta_codigo_compra INT)
Al comprar una oferta se le resta la cantidad comprada a la cantidad disponible de la misma, se crea el cupón correspondiente y se disminuye el saldo del cliente.

Cupón

Entidad:
Entidad que nos permite persistir en la base de datos aquellos cupones generados a partir de la compra de una oferta por un Cliente. Para identificarlo se tendrá una Primary Key que será el cupon_codigo el cual habrá uno por Cupón. Cada Cupón tendrá tres Foreign Keys, los cuales son el código_oferta del cual se crea el cupón, el clie_codigo que determina que cliente compró la oferta y a quien se le entregó el cupón y fact_numero el cual es un campo que en un principio estará en null y una vez facturado el cupón se completará con la factura determinada, este último campo se encuentra del lado del cupón debido a que es una relación de Uno a Muchos, es decir, una factura tendrá muchos cupones.
Campos:

- cupon_codigo (Primary Key): Es el capo inequívoco para identificar cada cupón, este se generará cada vez que se inserte un nuevo cupón.
- cupon_cantidad: Se refiere a la cantidad de ofertas que se compraron por el cliente.
- cupon_fecha: Fecha en la cual se realizó la compra del cupón.
- oferta_codigo (Foreign Key): Codigo que hace referencia a la oferta con la cual se vincula el cupón.
- clie_codigo(Foreign Key): Codigo del cliente que compró la oferta con el cual se relaciona el cupón.
- fact_numero(Foreign Key): Codigo de la factura la cual contempla que se realizó la facturación para este cupón.

Entrega

Entidad:
Esta entidad nos permite persistir las entregas realizadas de cupones, esta entrega se inserta una vez consumido un cupón por el cliente. Para identificar cada entrega disponemos de una Primary Key autogenerada para identificar cada entrega y una Foreign Key que hace referencia al cupón que fue consumido.
Campos:

- entrega_codigo (Primary Key): Campo inequívoco que nos permite identificar las diferentes entregas.
- entrega_fecha : Campo de tipo Date que nos determina en qué fecha se realizó el consumo del cupón.
- cupon_codigo (Foreign Key): Campo que hace referencia a la relación de la entrega con el cupón que fue consumido.

Carga

Entidad:
Esta entidad es la encargada de tener constancia de aquellas cargas que fueron realizadas en el sistema, y sobre qué cliente fueron ejecutadas. Para esto disponemos de un campo identificatorio el cual es el código de carga y se genera automáticamente para cada carga.

En cuanto a sus Foreign Keys, posee tres, una para identificar a qué usuario se le aplicó la carga, otra para tener registro de la tarjeta utilizada para la carga (esto será null si se realiza en efectivo) y uno para el tipo de pago, que hace referencia a la tabla **Tipo_Pago** el cual tiene los diferentes medios para pagar.
Campos:

- carga_codigo(Primary Key): Campo para identificar inequívocamente la carga realizada.
- carga_fecha: campo del tipo Date, el cual determina la fecha en la que se realizó la carga.
- carga_monto: Cantidad numérica que nos dice cuánto fue el saldo aumentado para el cliente en la carga.
- clie_codigo(Foreign Key): este campo hace referencia al cliente el cual realizó la carga y el cual se le incrementará el saldo.
- tarjeta_numero(Foreign Key): Este campo hará referencia a una de las filas de la tabla **Tarjeta** la cual contendrá la información de la tarjeta del usuario.
- tipo_pago_codigo(Foreign Key): Este campo contendrá una referencia hacia uno de los tipos de pago de la tabla **Tipo_Pago**

Procedures:

cargarCredito(@fecha_de_carga DATE , @monto numeric(18,2) , @clie_codigo_carga INT , @tarjeta_numero_carga INT , @tipo_pago_carga INT)
Lo que hace este procedure es sumarle crédito a la cuenta del cliente pasado por parámetro. Se debe especificar el monto y el tipo de pago. Si el tipo de carga es 2 (con tarjeta), y el usuario no tiene ninguna tarjeta ingresada, se la crea.

Factura

Entidad:
Esta entidad será la que persista aquellas facturas generadas por el sistema de los proveedores que hayan realizado ventas de cupones. Para esto tendremos como Primary Key un número de factura que será único e unívoco y una Foreign Key la cual nos permitirá relacionarnos con la tabla **Proveedor** y saber que proveedor tiene vinculada la factura.
Cabe destacar que las facturas van a ser realizadas por un rango de fechas, en las cuales se comprenden todos los cupones vendidos dentro de este rango.
Campos:

- fact_numero (Primary Key): Numero inequívoco para identificar la factura determinada.
- fact_fecha: Fecha para identificar cuando se generó la factura.
- fact_periodo_inicio: Fecha la cual indica a partir de que fecha se comenzó a facturar.
- fact_periodo_fin: Fecha la cual indica hasta que fecha se terminó de facturar.
- fact_total: Total acumulado de lo vendido segun lo comprendido entre las fechas estipuladas.
- prov_codigo (Foreign Key): Campo para relacionar a un proveedor perteneciente a la base de datos.

Procedures:

GENERAR_FACTURACION(@FECHA DATETIME, @INICIO DATETIME, @FIN DATETIME, @PROVEEDOR INT)

Función:

FACTURACION_PROVEEDOR(@PROVEEDOR INT, @INICIO DATETIME, @FIN DATETIME)

Listado Estadístico

Funciones:

TOP5_PROVEEDORES_MAYOR_FACTURACION(@ANIO INT, @MES INT)

TOP5_PROVEEDORES_MAYOR_DESCUENTO_OFRECIDO_EN_OFERTAS(@ANIO INT, @MES INT)

Aclaraciones

- A la hora de hacer la migración, los Clientes y Proveedores que existían previamente en el sistema, tendrán creado un Usuario por cada uno, el cual su usuario y contraseña dependerá del rol que tenían:
    - Clientes -> Usuario: DNI y Contraseña: DNI
    - Proveedores -> Usuario: CUIT y Contraseña: CUIT
- El sistema contará con un Usuario administrativo predefinido, éste tendrá como Usuario: ‘admin’ y Contraseña: ‘admin’
- No se podrá crear por la aplicación desktop un usuario con el Rol Administrativo.
- En cuanto a la dirección , el piso y el depto se ingresaran como campos numéricos.
- Al momento de la migración de las ofertas publicadas, la cantidad disponible de cada oferta va a corresponder con su cantidad máxima, ambas sacadas del dato de la columna “Oferta_cantidad” de la tabla maestra ,esto lo hicimos en el caso de que se desee probar con dichas ofertas, ya que si las ponemos en 0, no se podrían utilizar.