INDICE

Introducción:

El objetivo de este documento es poder conocer más acerca del trabajo práctico realizado durante el cuatrimestre y contar la estrategia utilizada para resolver el caso planteado.

Lo primero que hicimos fue estudiar detalladamente el problema y sus reglas de negocio mediante un análisis profundo de la tabla maestra y del enunciado suministrado por la cátedra. Luego hicimos un primer modelo de diagrama entidad relación, el cual fuimos refinando a medida que estudiamos en detalle cada caso puntual. Para ello fue clave realizar todo tipo de consultas sobre la tabla maestra, para deducir la relación de los datos ya existentes. Se nos fueron presentando situaciones no especificadas en el enunciado del trabajo práctico, para las cuales como grupo tuvimos que tomar decisiones, hasta llegar a un modelo final. Una vez diagramadas las tablas y sus relaciones establecimos los atributos (claves y no claves) y los tipos de datos de dichos atributos de acuerdo a los de la tabla maestra. Luego establecimos reglas de integridad semánticas, constraints, necesarias en cada caso particular para validar adecuadamente los datos.

Con todo el análisis terminado, procedimos a crear las tablas de acuerdo al modelo final establecido, y luego la migración de todos los datos existentes en la tabla maestra. Cabe aclarar que utilizamos Transact SQL.

Una vez cargados todos los datos en las tablas, comenzamos con la aplicación desktop, creando todos los formularios de acuerdo a las funcionalidades requeridas en el trabajo practico y a la guía de ABM's provista por la cátedra. Teniendo todos los formularios necesarios creados, comenzamos a programarlos uno por uno, creando en paralelo todas las Funciones, Stored Procedures y Triggers que se fueron necesitando según cada funcionalidad.

Una vez desarrolladas todas las funcionalidades de la aplicación desktop, comenzamos con las pruebas y validaciones. Luego se agregaron funcionalidades extra, para mejorar la interacción entre el usuario final y la aplicación, tales como accesos rápidos.

Decisiones estratégicas tomadas:

Algunas de las decisiones tomadas por el grupo durante el desarrollo del trabajo práctico, las cuales no figuraban en el enunciado, son:

- En una misma factura se puede "facturar" más de una publicación. Es decir, un vendedor puede rendir más de una publicación en una factura (se elige el número de publicaciones vencidas o finalizadas que se quieren rendir). El sistema no deja elegir de forma libre al vendedor qué publicaciones desea rendir. Las publicaciones que se pueden facturar están ordenadas primero por la fecha de vencimiento (o finalización) y luego por la fecha de la primer compra.

- Al facturar una publicación, se facturan al mismo tiempo todas las compras realizadas sobre esa publicación que aparezcan en la tabla compras. Esto se ha supuesto así ya que se podría llegar a dar el caso en el que se bloquee a un vendedor sólo por recibir un total de 10 compras (lo cual puede suceder en un período muy corto de tiempo). Entonces se ha tomado como regla de negocio que un vendedor no puede acumular un número mayor al de 10 publicaciones vencidas o finalizadas de manera simultánea. El vendedor tiene que ir rindiendo dichas publicaciones sin superar ese número. Esto, a su vez, deja que el vendedor pueda controlar mejor su situación de débito para con el sistema y es pura y exclusivamente su decisión deber al sistema publicaciones no rendidas.

- A los usuarios migrados de la tabla maestra, al no contar con usuario ni contraseña, se procedió a generar un usuario automático y con uno de los siguientes formatos, según corresponda a un cliente o a una empresa:
  - be_sharps_CX
  - be_sharps_EX

  Siendo X es un número, arrancando de 1 y que se incrementa en uno por la inserción de cada usuario.

- En el caso de las contraseñas, se establece para dichos usuarios la contraseña “besharps” (ya encriptada bajo el algoritmo de encriptación SHA256).

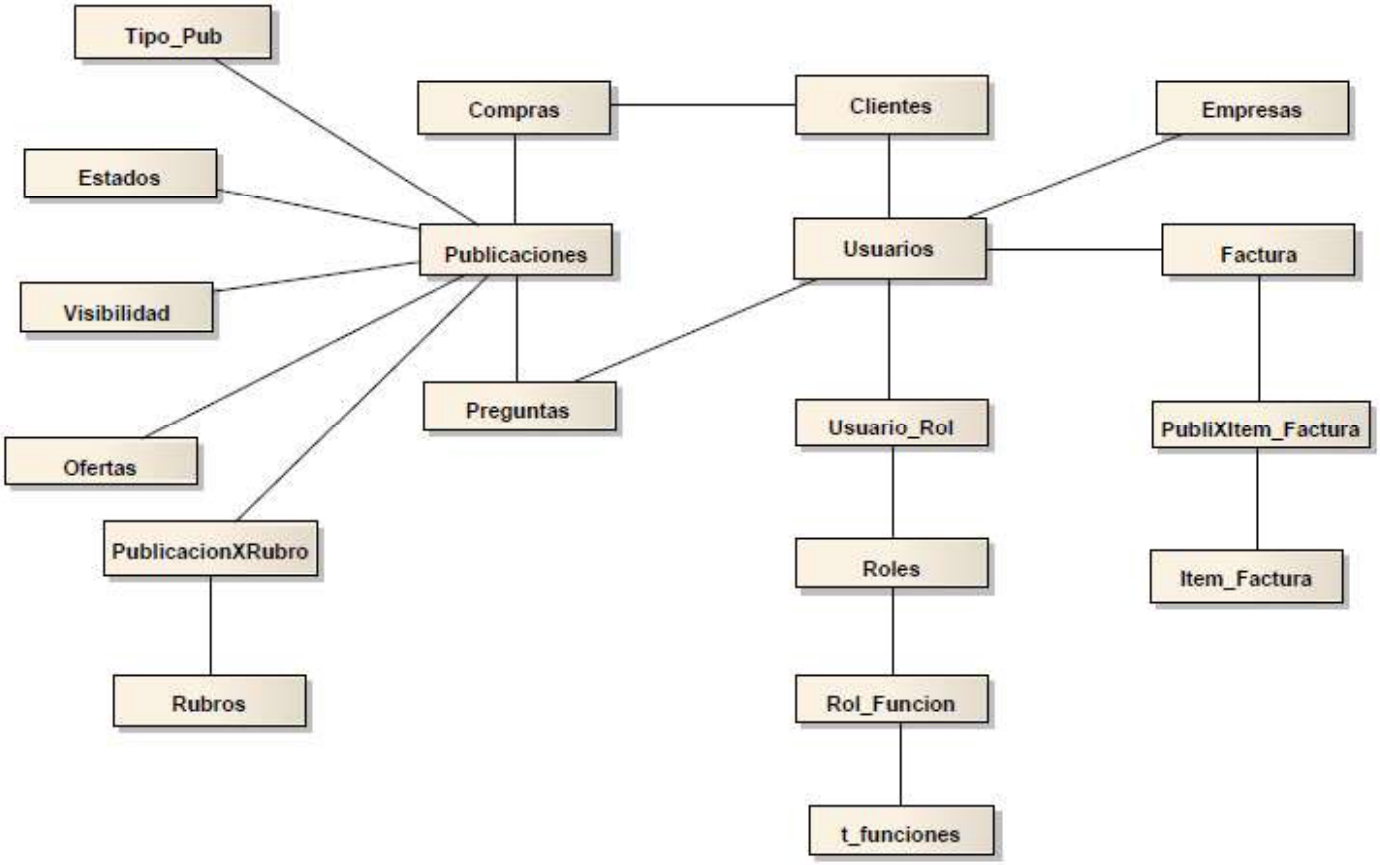
- Debido a que los clientes no poseen teléfono y los valores del campo CLI_TELEFONO no se pueden repetir ni tampoco pueden ser NULL, se les puso por defecto su número de DNI como teléfono. Esto sucede únicamente con los usuarios cuyo origen es la tabla maestra provista por la catedra.

- Al crear una publicación que se encuentre en estado de "Publicada" o "Pausada", no se podrá modificar el precio de dicha publicación. Esto lo hemos supuesto así, ya que se podría dar el caso en el que un vendedor cambie el precio de la publicación por uno mucho menor justo antes de rendir esa publicación, lo que sería una manera de violar el sistema de porcentajes de las visualizaciones. El precio de una publicación solo se puede modificar si ésta se encuentra en estado "Borrador".

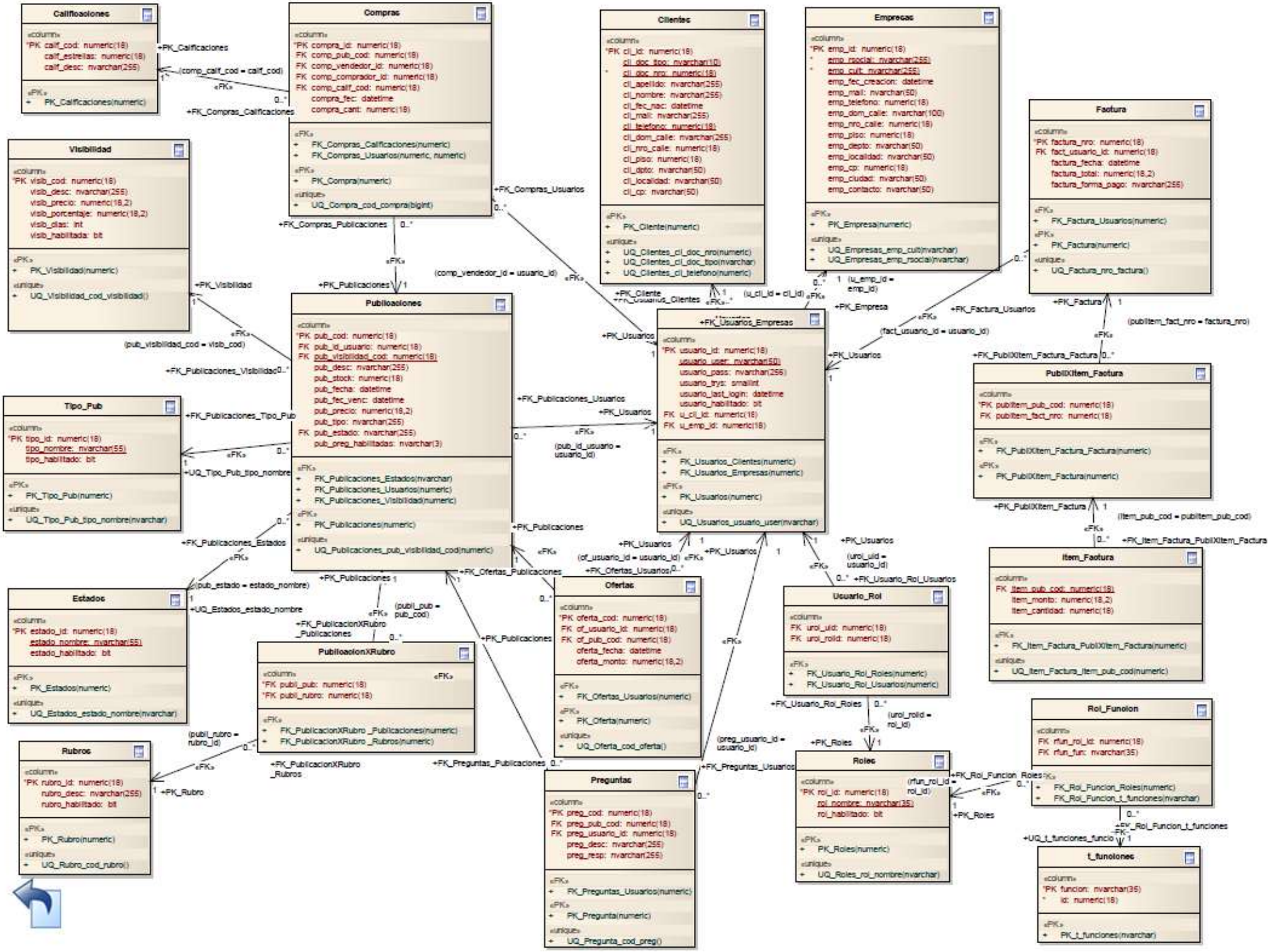
- Las publicaciones provenientes de la tabla maestra del tipo subasta que tienen una compra registrada, son tomadas como publicaciones finalizadas y facturadas (migrando un registro a la tabla de Compras y otro registro a la tabla de Facturas).

- Respecto a las bonificaciones, se decidió que lo que se bonificará será el precio por publicar y no las comisiones, ya que si se bonificasen las comisiones, podría dar lugar a fraudes por parte del usuario, como por ejemplo, si planea vender algo muy caro, primero hacer 9 ventas de escaso valor y como numero 10 la venta del producto caro.

- A continuación podemos ver el DER simplificado, a modo de entender cómo se relacionan cada una de las tablas modeladas:



A continuación se puede visualizar el DER completo que utilizamos de modelo:


Calificaciones
«column»
*PK calif_cod: numeric(18)
calif_estrellas: numeric(18)
calif_desc: nvarchar(255)
«PK»
+ PK_Calificaciones(numeric)
+PK_Calificaciones
(comp_calif_cod = calif_cod)
«FK»
1
0..*
+FK_Compras_Calificaciones
Compras
«column»
*PK compra_id: numeric(18)
FK comp_pub_cod: numeric(18)
FK comp_vendedor_id: numeric(18)
FK comp_comprador_id: numeric(18)
FK comp_calif_cod: numeric(18)
compra_fec: datetime
compra_cant: numeric(18)
«FK»
+ FK_Compras_Calificaciones(numeric)
+ FK_Compras_Usuarios(numeric, numeric)
«PK»
+ PK_Compra(numeric)
«unique»
+ UQ_Compra_cod_compra(bigint)
+FK_Compras_Usuarios
+FK_Compras_Publicaciones
Clientes
«column»
*PK cl_id: numeric(18)
cl_doc_tipo: nvarchar(10)
cl_doc_nro: numeric(18)
cl_apellido: nvarchar(255)
cl_nombre: nvarchar(255)
cl_fec_nac: datetime
cl_mail: nvarchar(255)
cl_telefono: numeric(18)
cl_dom_calle: nvarchar(255)
cl_nro_calle: numeric(18)
cl_piso: numeric(18)
cl_dpto: nvarchar(50)
cl_localidad: nvarchar(50)
cl_cp: nvarchar(50)
«PK»
+ PK_Cliente(numeric)
«unique»
+ UQ_Clientes_cl_doc_nro(numeric)
+ UQ_Clientes_cl_doc_tipo(nvarchar)
+ UQ_Clientes_cl_telefono(numeric)
Empresas
«column»
*PK emp_id: numeric(18)
emp_rsocial: nvarchar(255)
emp_cuit: nvarchar(255)
emp_fec_creacion: datetime
emp_mail: nvarchar(50)
emp_telefono: numeric(18)
emp_dom_calle: nvarchar(100)
emp_nro_calle: numeric(18)
emp_piso: numeric(18)
emp_depto: nvarchar(50)
emp_localidad: nvarchar(50)
emp_cp: numeric(18)
emp_ciudad: nvarchar(50)
emp_contacto: nvarchar(50)
«PK»
+ PK_Empresa(numeric)
«unique»
+ UQ_Empresas_emp_cuit(nvarchar)
+ UQ_Empresas_emp_rsocial(nvarchar)
Factura
«column»
*PK factura_nro: numeric(18)
FK fact_usuario_id: numeric(18)
factura_fecha: datetime
factura_total: numeric(18,2)
factura_forma_pago: nvarchar(255)
«FK»
+ FK_Factura_Usuarios(numeric)
«PK»
+ PK_Factura(numeric)
«unique»
+ UQ_Factura_nro_factura()
Visibilidad
«column»
*PK visib_cod: numeric(18)
visib_desc: nvarchar(255)
visib_precio: numeric(18,2)
visib_porcentaje: numeric(18,2)
visib_dias: int
visib_habilitada: bit
«PK»
+ PK_Visibilidad(numeric)
«unique»
+ UQ_Visibilidad_cod_visibilidad()
+PK_Visibilidad
(pub_visibilidad_cod = visib_cod)
+FK_Publicaciones_Visibilidad
Publicaciones
«column»
*PK pub_cod: numeric(18)
FK pub_id_usuario: numeric(18)
FK pub_visibilidad_cod: numeric(18)
pub_desc: nvarchar(255)
pub_stock: numeric(18)
pub_fecha: datetime
pub_fec_venc: datetime
pub_precio: numeric(18,2)
pub_tipo: nvarchar(255)
FK pub_estado: nvarchar(255)
pub_preg_habilitadas: nvarchar(3)
«FK»
+ FK_Publicaciones_Estados(nvarchar)
+ FK_Publicaciones_Usuarios(numeric)
+ FK_Publicaciones_Visibilidad(numeric)
«PK»
+ PK_Publicaciones(numeric)
«unique»
+ UQ_Publicaciones_pub_visibilidad_cod(numeric)
+PK_Publicaciones
(comp_vendedor_id = usuario_id)
+PK_Publicaciones
(pub_id_usuario = usuario_id)
+FK_Publicaciones_Usuarios
+PK_Usuarios
+FK_Usuarios_Empresas
«column»
*PK usuario_id: numeric(18)
usuario_user: nvarchar(50)
usuario_pass: nvarchar(256)
usuario_try: smallint
usuario_last_login: datetime
usuario_habilitado: bit
FK u_cl_id: numeric(18)
FK u_emp_id: numeric(18)
«FK»
+ FK_Usuarios_Clientes(numeric)
+ FK_Usuarios_Empresas(numeric)
«PK»
+ PK_Usuarios(numeric)
«unique»
+ UQ_Usuarios_usuario_user(nvarchar)
+PK_Cliente
+FK_Usuarios_Clientes
(u_cl_id = cl_id)
+PK_Empresa
(u_emp_id = emp_id)
+PK_Usuarios
(fact_usuario_id = usuario_id)
+FK_Factura_Usuarios
+PK_Factura
(pubiItem_fact_nro = factura_nro)
+FK_PubliXItem_Factura_Factura
PubliXItem_Factura
«column»
*PK pubiItem_pub_cod: numeric(18)
FK pubiItem_fact_nro: numeric(18)
«FK»
+ FK_PubliXItem_Factura_Factura(numeric)
«PK»
+ PK_PubliXItem_Factura(numeric)
+PK_PubliXItem_Factura
(item_pub_cod = pubiItem_pub_cod)
+FK_Item_Factura_PubliXItem_Factura
Item_Factura
«column»
FK item_pub_cod: numeric(18)
item_monto: numeric(18,2)
item_cantidad: numeric(18)
«FK»
+ FK_Item_Factura_PubliXItem_Factura(numeric)
«unique»
+ UQ_Item_Factura_item_pub_cod(numeric)
Tipo_Pub
«column»
*PK tipo_id: numeric(18)
tipo_nombre: nvarchar(55)
tipo_habilitado: bit
«PK»
+ PK_Tipo_Pub(numeric)
«unique»
+ UQ_Tipo_Pub_tipo_nombre(nvarchar)
+FK_Publicaciones_Tipo_Pub
+UQ_Tipo_Pub_tipo_nombre
Estados
«column»
*PK estado_id: numeric(18)
estado_nombre: nvarchar(55)
estado_habilitado: bit
«PK»
+ PK_Estados(numeric)
«unique»
+ UQ_Estados_estado_nombre(nvarchar)
+FK_Publicaciones_Estados
(pub_estado = estado_nombre)
+UQ_Estados_estado_nombre
Ofertas
«column»
*PK oferta_cod: numeric(18)
FK of_usuario_id: numeric(18)
FK of_pub_cod: numeric(18)
oferta_fecha: datetime
oferta_monto: numeric(18,2)
«FK»
+ FK_Ofertas_Usuarios(numeric)
«PK»
+ PK_Oferta(numeric)
«unique»
+ UQ_Oferta_cod_oferta()
+FK_Ofertas_Publicaciones
+FK_Ofertas_Usuarios
(of_usuario_id = usuario_id)
Usuario_Rol
«column»
FK urol_uid: numeric(18)
FK urol_rolid: numeric(18)
«FK»
+ FK_Usuario_Rol_Roles(numeric)
+ FK_Usuario_Rol_Usuarios(numeric)
+FK_Usuario_Rol_Usuarios
(urol_uid = usuario_id)
+FK_Usuario_Rol_Roles
(urol_rolid = rol_id)
PublioacionXRubro
«column»
*FK publi_pub: numeric(18)
*FK publi_rubro: numeric(18)
«FK»
+ FK_PublicacionXRubro _Publicaciones(numeric)
+ FK_PublicacionXRubro _Rubros(numeric)
+FK_PublicacionXRubro _Publicaciones
(publi_pub = pub_cod)
+FK_PublicacionXRubro _Rubros
(publi_rubro = rubro_id)
Rubros
«column»
*PK rubro_id: numeric(18)
rubro_desc: nvarchar(255)
rubro_habilitado: bit
«PK»
+ PK_Rubro(numeric)
«unique»
+ UQ_Rubro_cod_rubro()
+PK_Rubro
Preguntas
«column»
*PK preg_cod: numeric(18)
FK preg_pub_cod: numeric(18)
FK preg_usuario_id: numeric(18)
preg_desc: nvarchar(255)
preg_resp: nvarchar(255)
«FK»
+ FK_Preguntas_Usuarios(numeric)
«PK»
+ PK_Pregunta(numeric)
«unique»
+ UQ_Pregunta_cod_preg()
+FK_Preguntas_Publicaciones
+FK_Preguntas_Usuarios
(preg_usuario_id = usuario_id)
Roles
«column»
*PK rol_id: numeric(18)
rol_nombre: nvarchar(35)
rol_habilitado: bit
«PK»
+ PK_Roles(numeric)
«unique»
+ UQ_Roles_rol_nombre(nvarchar)
+PK_Roles
(rfun_rol_id = rol_id)
+FK_Rol_Funcion_Roles
Rol_Funcion
«column»
FK rfun_rol_id: numeric(18)
FK rfun_fun: nvarchar(35)
«FK»
+ FK_Rol_Funcion_Roles(numeric)
+ FK_Rol_Funcion_t_funciones(nvarchar)
+FK_Rol_Funcion_t_funciones
+UQ_t_funciones_funcio
t_funciones
«column»
*PK funcion: nvarchar(35)
id: numeric(18)
«PK»
+ PK_t_funciones(nvarchar)

Modelado de tablas:

Las tablas modeladas, con sus respectivas constraints fueron las siguientes:

Nota: El equipo decidió colocarle el nombre de la tabla a cada atributo, al comienzo de los mismos. En algunos casos el nombre completo y en otros una abreviación del mismo.

- **Clientes**

  Contiene los datos de los Clientes.

  CLI_ID [PK | IDENTITY (1,1)]: código identificador de un cliente
  CLI_DOC_TIPO [UNIQUE | DEFAULT 'DNI']: tipo de documento de identidad del cliente. Por defecto toma valor 'DNI'
  CLI_DOC_NRO [UNIQUE | NOT NULL]: número de documento de identidad del cliente.
  CLI_APELLIDO: apellido del cliente
  CLI_NOMBRE: nombre del cliente
  CLI_FEC_NAC: fecha de nacimiento del cliente
  CLI_MAIL: dirección de mail del cliente
  CLI_TELEFONO [UNIQUE]: teléfono del cliente
  CLI_DOM_CALLE: nombre de la calle del domicilio del cliente
  CLI_NRO_CALLE: altura de la calle del domicilio del cliente
  CLI_PISO: piso del domicilio del cliente
  CLI_DPTO: departamento del domicilio del cliente
  CLI_LOCALIDAD: localidad del domicilio del cliente
  CLI_CP: código postal del domicilio del cliente

- **Empresas**

  Contiene los datos de las empresas.

  EMP_ID [PK | IDENTITY (1,1)]: código identificador de la empresa
  EMP_RSOCIAL [UNIQUE | NOT NULL]: razón social de la empresa
  EMP_CUIT [UNIQUE | NOT NULL]: número de CUIT correspondiente a la empresa
  EMP_FEC_CREACION: fecha de registro de la empresa en el Sistema
  EMP_MAIL: mail de la empresa
  EMP_TELEFONO: teléfono de la empresa
  EMP_DOM_CALLE: nombre de la calle del domicilio de la empresa
  EMP_NRO_CALLE: altura de la calle del domicilio de la empresa
  EMP_PISO: piso del domicilio de la empresa
  EMP_DEPTO: departamento del domicilio de la empresa
  EMP_LOCALIDAD: localidad del domicilio de la empresa
  EMP_CP: código postal del domicilio del cliente
  EMP_CIUDAD: ciudad del domicilio del cliente

EMP_CONTACTO: contacto de la empresa

- **Usuarios**

  Contiene los datos de los usuarios del Sistema

  USUARIO_ID [PK | IDENTITY (1,1)]: código identificador de un usuario
  USUARIO_USER [UNIQUE]: nombre identificador de usuario
  USUARIO_PASS: contraseña del usuario
  USUARIO_TRYS [DEFAULT 3 | CHECK (0, 1, 2, 3)]: cantidad de intentos de logeo fallidos restantes
  USUARIO_LAST_LOGIN [DEFAULT NULL]: fecha de último logeo al Sistema
  USUARIO_HABILITADO [DEFAULT 1]: estado de habilitación de usuario. Si es válido toma valor 1 y si es inválido toma valor 0
  U_CLI_ID [FK]: número identificador de cliente
  U_EMP_ID [FK]: número identificador de la empresa a la que corresponde

- **Visibilidad**

  Contiene los datos las diferentes opciones de visibilidad del sistema.

  VISIB_COD [PK]: número identificador de la visibilidad
  VISIB_DESC: descripción de la visibilidad
  VISIB_PRECIO: precio correspondiente a la visibilidad
  VISIB_PORCENTAJE: porcentaje correspondiente a la visibilidad
  VISIB_DIAS [DEFAULT 7]: cantidad de días de visibilidad
  VISIB_HABILITADA [DEFAULT 1]: estado de habilitación de visibilidad. Si es válido toma valor 1 y si no es válido 1.

- **Rubros**

  Contiene los datos los rubros a los que puede pertenecer una publicación.

  RUBRO_ID [PK | IDENTITY (1,1)]: número identificador del rubro
  RUBRO_DESC: nombre del rubro
  RUBRO_HABILITADO [DEFAULT 1]: estado de habilitación del rubro. Si es válido toma valor 1 y si no es válido 0.

- **Estados**

  Contiene los estados posibles de una publicación.

  ESTADO_ID [PK | IDENTITY (1,1)]: número identificador del estado
  ESTADO_NOMBRE [UNIQUE]: nombre del estado de la publicación

ESTADO_HABILITADO [DEFAULT 1]: estado de habilitación del estado. Si es válido toma valor 1 y si no es válido 0.

- **Tipo_pub**

  Contiene los tipos de una publicación.

  TIPO_ID [PK | IDENTITY (1,1)]: número identificador del tipo de publicación
  TIPO_NOMBRE [UNIQUE]: nombre del tipo de la publicación
  TIPO_HABILITADO [DEFAULT 1]: estado de habilitación del tipo. Si es válido toma valor 1 y si no es válido 0.

- **Publicaciones**

  Contiene los datos de las publicaciones.

  PUB_COD [PK | IDENTITY (1,1)]: número identificador de la publicación
  PUB_ID_USUARIO [FK]: código identificador de usuario
  PUB_VISIBILIDAD_COD [FK]: número identificador de la visibilidad
  PUB_DESC: descripción de la publicación
  PUB_STOCK [CHECK >-1]: cantidad de unidades a la venta. El valor deberá ser mayor que -1.
  PUB_FECHA: fecha de la publicación
  PUB_FEC_VENC: fecha de vencimiento de la publicación
  PUB_PRECIO: precio de la venta
  PUB_TIPO [FK]: tipo de publicación.
  PUB_ESTADO [FK]: estado de la publicación.
  PUB_PREG_HABILITADAS [DEFAULT 'NO' | CHECK ('NO, 'YES')]: determina si se permiten realizar preguntas o no en la publicación. Los valores posibles son 'YES' y 'NO'

- **PublicacionXRubro**

  Contiene los datos de la relación publicación-rubro.

  PUBLI_PUB [FK | NOT NULL]: número identificador de la publicación
  PUBLI_RUBRO [FK | NOT NULL]: número identificador del rubro

- **Calificaciones**

  Contiene los datos de las calificaciones de los usuarios.

  CALIF_COD [PK | IDENTITY (1,1)]: número identificador de la calificación
  CALIF_ESTRELLAS: cantidad de estrellas asignadas

CALIF_DESC: descripción de la calificación

- **Compras**

  Contiene los datos de las compras realizadas.

  COMPRA_ID [PK | IDENTITY (1,1)]: código identificador de la compra
  COMP_PUB_COD [FK]: número identificador de la publicación
  COMP_VENDEDOR_ID [FK]: código identificador de un usuario
  COMP_COMPRADOR_ID [FK]: código identificador de un usuario
  COMP_CALIF_COD [FK]: código identificador de una calificación
  COMPRA_FEC: fecha de la operación
  COMPRA_CANT: cantidad de unidades

- **Preguntas**

  Contiene los datos correspondientes a las preguntas realizadas y contestadas en las publicaciones

  PREG_COD [PK | IDENTITY (1,1)]: número identificador de una pregunta
  PREG_PUB_COD [FK]: número identificador de la publicación
  PREG_USUARIO_ID [FK]: código identificador de un usuario
  PREG_DESC: consulta realizada
  PREG_RESP: respuesta a la consulta

- **Ofertas**

  Contiene los datos de las ofertas.

  OFERTA_COD [PK | IDENTITY (1,1)]: número identificador de una oferta
  OF_USUARIO_ID [FK]: número identificador de un usuario
  OF_PUB_COD [FK]: número identificador de la publicación
  OFERTA_FECHA: fecha de la oferta
  OFERTA_MONTO: cantidad de dinero ofertada

- **Factura**

  Contiene los datos de las facturas

  FACTURA_NRO [PK | IDENTITY (1,1)]: número identificador de la factura
  FACT_USUARIO_ID [FK]: código identificador de un usuario
  FACTURA_FECHA: fecha de la factura
  FACTURA_TOTAL: total de la factura
  FACTURA_FORMA_PAGO: forma de pago de la factura

Al incluir el campo FACTURA_TOTAL estamos aumentando la redundancia y la complejidad espacial debido a que tenemos todos los datos para calcular el monto total de la factura pero también estamos disminuyendo la complejidad computacional debido a que cuando se requiera dicho dato no es necesario calcular la sumatoria de los montos de todos los ítems que componen la factura.

- **PubliXItem_Factura**

  Contiene los datos de la relación publicación-item_factura.

  PUBLITEM_PUB_COD [PK | FK]: número identificador de la publicación
  PUBLITEM_FACT_NRO [FK]: número identificador de la factura

- **Item_Factura**

  Contiene los datos de los ítems de las facturas

  ITEM_PUB_COD [FK]: número identificador de la publicación
  ITEM_MONTO: precio del ítem
  ITEM_CANTIDAD [DEFAULT 1]: cantidad de unidades del ítem

- **Roles**

  Contiene los datos de los roles para las funcionalidades del sistema.

  ROL_ID [PK | IDENTITY (1,1)]: número identificador del rol
  ROL_NOMBRE [UNIQUE]: nombre descriptivo del rol
  ROL_HABILITADO: indica si es válido 1 o invalido 0

- **t_funciones**

  Contiene las funciones de la aplicación.

  ID [IDENTITY (1,1) | NOT NULL]: código identificador de la función
  FUNCION [PK]: nombre de la función

- **Rol_Funcion**

  Contiene los datos de la relación rol-usuario.

  RFUN_ROL_ID [FK]: número identificador del rol
  RFUN_FUN [FK]: descripción de la función de usuario

- **Usuario_Rol**

  Contiene los datos de la relación usuario-rol.

  UROL_UID [FK]: código identificador de un usuario
  UROL_ROLID [FK]: número identificador del rol

Carga de datos básicos:

Una vez creadas las tablas, continuamos cargando ciertos datos que no se encuentran en la tabla maestra, y son necesarios para llevar a cabo la migración. Estos son por ejemplo todas las funciones.

A continuación se detallan las tablas y los datos cargamos manualmente por código desde el script inicial:

- **t_funcion:** Debido a que el enunciado establece que las funcionalidades son fijas, es decir, no se pueden agregar, ni quitar, ni modificar una funcionalidad, hubo que agregarlas por código una por una. Estas son:
  - ABM Cliente
  - ABM Empresa
  - ABM Rol
  - ABM Visibilidad
  - Listado estadistico
  - Buscar/Comprar
  - Generar publicacion
  - Editar publicacion
  - Facturar publicaciones
  - Gestor de preguntas
  - Historial
  - Calificar compras
  - Facturar cliente
  - Editar perfil

- **Clientes:** Se creó un cliente para luego asociarlo con el usuario a crear según el enunciado. Este se crea con los siguientes detalles:
  - ID: 1
  - Tipo de Documento: “DNI”
  - Número de Documento: 99999999
  - Apellido: “Sharps”
  - Nombre: “Be”
  - Fecha de Nacimiento: 2014-01-01
  - Mail: besharps@gmail.com
  - Teléfono: 99999999
  - Calle: “AvenidaSiempreViva”
  - Altura: 999
  - Piso: 9
  - Departamento: “Z”
  - Localidad: “CABA”
  - Código Postal: 9999

- **Empresas:** Se creó una empresa para luego asociarla con el usuario a crear según el enunciado. Este se crea con los siguientes detalles:
  - ID: 1
  - Razón Social: BeSharps
  - CUIT: 9-99999999-9
  - Fecha de Creación: 2014-01-01
  - Mail: besharps@gmail.com
  - Teléfono: 90000009
  - Calle: “AvenidaSiempreViva”
  - Altura: 999
  - Piso: 9
  - Departamento: “Z”
  - Localidad: “CABA”
  - Código Postal: 9999
  - Ciudad: “BuenosAires”
  - Contacto: “Jaime”

- **Usuarios:** El enunciado pide la creación de un usuario que posea el rol de Administrador general, que será el que se usará para poder dar de alta a otros usuarios, quienes manejarán la aplicación para sus distintas actividades. El usuario se crea con los siguientes detalles:
  - Usuario: “admin”
  - Contraseña: “w23e”
  - Cantidad de intentos disponibles de logueo: 3
  - Usuario habilitado
  - ID Cliente: 1
  - ID Empresa: 1
  - Último ingreso al sistema: fecha de ejecución del script

- **Roles:** El enunciado del trabajo práctico define un rol inicial a crear llamado *Administrador General*.

  Adicionalmente se crean los siguientes roles:
  - Administrador
  - Cliente
  - Empresa

- **Rol_Funcion:** El enunciado del trabajo práctico define que el rol *Administrador General* deberá contener todas las funciones del Sistema. Estas fueron insertadas anteriormente en la tabla T_FUNCION.

En cuanto a los otros roles creados, la asignación de funciones para cada uno es la siguiente:

- Administrador:
  - ABM Cliente
  - ABM Empresa
  - ABM Rol
  - ABM Visibilidad
  - Listado Estadístico
  - Facturar cliente
- Cliente:
  - Buscar/Comprar
  - Generar publicación
  - Editar publicación
  - Facturar publicaciones
  - Gestor de preguntas
  - Historial
  - Calificar compras
  - Editar perfil
- Empresa:
  - Generar publicación
  - Editar publicación
  - Facturar publicaciones
  - Gestor de preguntas
  - Historial
  - Editar perfil

- **Usuario_Rol:** El enunciado del trabajo práctico define que el usuario *Admin* deberá contar con el rol *Administrador General*.

- **Tipo_Pub:** Se crearon los tipos de publicaciones posibles. Estos son los siguientes:
  - Compra Inmediata
  - Subasta

- **Estados:** Se crearon los estados de publicaciones posibles. Estos son los siguientes:
  - Borrador
  - Publicada
  - Pausada
  - Finalizada

Migración:

Una vez creadas las tablas y cargados los datos básicos, procedimos a migrar los datos de la tabla maestra provista por la cátedra, a todas las tablas.

Se realizó de la siguiente manera:

- **Clientes:** Se migran a la tabla CLIENTES, los datos de las columnas CLI_DOC_NRO, CLI_APELLIDO, CLI_NOMBRE, CLI_FEC_NAC, CLI_MAIL, CLI_TELEFONO, CLI_DOM_CALLE, CLI_NRO_CALLE, CLI_PISO, CLI_DPTO y CLI_CP, correspondientes a las siguientes columnas de la tabla maestra respectivamente CLI_DNI, CLI_APELLIDO, CLI_NOMBRE, CLI_FECHA_NAC, CLI_MAIL, CLI_DNI, CLI_DOM_CALLE, CLI_NRO_CALLE, CLI_PISO, CLI_DEPTO y CLI_COD_POSTAL. Cabe aclarar que sobre los datos de origen se toman únicamente los registros diferentes, utilizando la sentencia DISTINCT.

  Para realizar la migración solo se verifica que el valor del campo CLI_DNI no carezca de valores. En caso de que el campo sea NULL no se inserta en la tabla CLIENTES.

  En el caso del campo CLI_TELEFONO, se le carga el valor del campo CLI_DNI ya que el mismo se definió como UNIQUE, y este último no se repite entre registros.

- **Empresas:** Se migran a la tabla EMPRESAS, los datos de las columnas EMP_RSOCIAL, EMP_CUIT, EMP_FEC_CREACION, EMP_MAIL, EMP_DOM_CALLE, EMP_NRO_CALLE, EMP_PISO, EMP_DEPTO y EMP_CP, correspondientes a las siguientes columnas de la tabla maestra respectivamente PUBL_EMPRESA_RAZON_SOCIAL, PUBL_EMPRESA_CUIT, PUBL_EMPRESA_FECHA_CREACION, PUBL_EMPRESA_MAIL, PUBL_EMPRESA_DOM_CALLE, PUBL_EMPRESA_NRO_CALLE, PUBL_EMPRESA_PISO, PUBL_EMPRESA_DEPTO y PUBL_EMPRESA_COD_POSTAL.

  Para realizar la migración solo se verifica que el valor del campo PUBL_EMPRESA_CUIT no carezca de valores. En caso de que el campo sea NULL no se inserta en la tabla EMPRESAS.

- **Usuarios:** La tabla se completa mediante los registros insertados anteriormente en las tablas CLIENTES y EMPRESAS.

  En primer lugar se insertan en la tabla USUARIOS datos en las columnas U_CLI_ID, USUARIO_USER y USUARIO_PASS. El campo U_CLI_ID se completa con el valor del campo CLI_ID de la tabla CLIENTES. En el caso del campo USUARIO_USER, este se obtiene mediante la función *generarUsuario* a la cual se le pasan 2 parámetros:

- El valor del campo CLI_ID
- Valor constante 'C'

En cuanto al campo USUARIO_PASS, este se completa con un valor fijo que se encuentra encriptado por el algoritmo SHA256. Este es "1b397769df4d42ebb18176c0bca10e5a17d81d07ef0f949866f7954b53a1d403" y el cual equivale a "besharps".

Luego se insertan en la misma tabla los datos de las columnas U_EMP_ID, USUARIO_USER y USUARIO_PASS. El campo U_EMP_ID se completa con el valor del campo EMP_ID de la tabla EMPLEADOS. En el caso del campo USUARIO_USER, este se obtiene mediante la función *generarUsuario* a la cual se le pasan 2 parámetros:

- El valor del campo EMP_ID
- Valor constante 'E'

En cuanto al campo USUARIO_PASS, este se completa de la misma forma que se hizo con los clientes en esta misma tabla.

En ambos casos se insertan los usuarios cuyo ID de Cliente / Empresa sea distinto de 1 ya que estos corresponden al Cliente / Empresa creado anteriormente para el usuario Admin.

- **Usuario_Rol:** Finalizada la carga de la tabla USUARIOS, se procede a asignarle a dichos usuarios los roles correspondientes. Para esto se inserta en la tabla USUARIO_ROL un registro con cada ID de usuario y con cada ID de rol. Para esto se sigue la siguiente lógica:
  - En caso de que el usuario no contenga valores en el campo U_CLI_ID → Se asigna el valor 3 que corresponde al rol Empresa.
  - En caso de que el usuario contenga valores en el campo U_CLI_ID → Se asigna el valor 2 que corresponde al rol Cliente.

  En ambos casos se valida que el ID sea distinto de 1 ya que los mismos corresponden al Cliente / Empresa creado para el usuario Admin.

- **Visibilidad:** Se migran a la tabla VISIBILIDAD, los datos de las columnas VISIB_COD, VISIB_DESC, VISIB_PRECIO y VISIB_PORCENTAJE, correspondientes a las siguientes columnas de la tabla maestra respectivamente PUBLICACION_VISIBILIDAD_COD, PUBLICACION_VISIBILIDAD_DESC, PUBLICACION_VISIBILIDAD_PRECIO y PUBLICACION_VISIBILIDAD_PORCENTAJE.

A su vez se cargó en el campo VISIB_DIAS de acuerdo a la siguiente lógica:

- En caso de que el valor del campo PUBLICACION_VISIBILIDAD_COD sea = a 10006 → 7
- En caso de que el valor del campo PUBLICACION_VISIBILIDAD_COD sea <> a 10006 → 30

Para realizar la migración solo se verifica que el valor del campo PUBLICACION_VISIBILIDAD_COD no carezca de valores. En caso de que el campo sea NULL no se inserta en la tabla VISIBILIDAD.

A su vez, solo se insertan los registros cuyos valores son distintos. Para esto se utilizó la sentencia DISTINCT.

Antes de realizar la inserción de registros, se activa el IDENTITY_INSERT sobre la tabla VISIBILIDAD para que el motor de la Base de Datos permita insertar valores en el campo VISIBILIDAD_COD que posee la constraint IDENTITY (1,1).

Luego de realizar la migración de dicha tabla, se deshabilita el IDENTITY_INSERT para que de ahí en adelante a todas las inserciones de registros que se hagan, se les asigne de forma automática un número de identidad.

Cabe aclarar que los registros que se insertan se hacen de forma ordenada según los valores del campo PUBLICACION_VISIBILIDAD_COD.

- **Rubros:** Se migran a la tabla RUBROS, los datos de la columna RUBRO_DESC cuyo origen es la columna PUBLICACION_RUBRO_DESCRIPCION de la tabla maestra.

  Para realizar la migración solo se verifica que el valor del campo PUBLICACION_RUBRO_DESCRIPCION no carezca de valores. En caso de que el campo sea NULL no se inserta en la tabla RUBROS.

  A su vez, solo se insertan los registros cuyos valores son distintos. Para esto se utilizó la sentencia DISTINCT.

- **Publicaciones:** Se migran a la tabla PUBLICACIONES, los datos de las columnas PUB_COD, PUB_DESC, PUB_STOCK, PUB_FECHA, PUB_FEC_VENC, PUB_ESTADO, PUB_TIPO, PUB_PRECIO y PUB_VISIBILIDAD_COD, correspondientes a las siguientes columnas agrupadas de la tabla maestra respectivamente PUBLICACION_COD, PUBLICACION_DESCRIPCION, PUBLICACION_STOCK, PUBLICACION_FECHA, PUBLICACION_FECHA_VENC, PUBLICACION_ESTADO, PUBLICACION_TIPO, PUBLICACION_PRECIO y PUBLICACION_VISIBILIDAD_COD.

El campo PUB_ID_USUARIO se completa mediante la función obtenerIDusuario a la cual se le pasa como parámetro el valor 'DNI' y el número de documento y retorna el número identificador del usuario.

Para realizar la migración solo se verifica que el valor del campo PUBLICACION _COD no carezca de valores. En caso de que el campo sea NULL no se inserta en la tabla PUBLICACION.

Antes de realizar la inserción de registros, se activa el IDENTITY_INSERT sobre la tabla PUBLICACIONES para que el motor de la Base de Datos permita insertar valores en el campo PUBLICACION_COD que posee la constraint IDENTITY (1,1).

Luego de realizar la migración de dicha tabla, se deshabilita el IDENTITY_INSERT para que de ahí en adelante a todas las inserciones de registros que se hagan, se les asigne de forma automática un número de identidad.

Cabe aclarar que los registros que se insertan se hacen de forma ordenada según los valores del campo PUBLICACION_COD.

- **PublicacionXRubro:** Se migran a la tabla PUBLICACIONXRUBRO, los datos de las columnas PUBLI_PUB y PUBLI_RUBRO. Los valores del campo PUBLI_PUB se obtienen del campo PUBLICACION_COD correspondiente a la tabla maestra.

  En cuanto a los valores de la columna PUBLI_RUBRO, estos se obtienen mediante la función obtenerIDrubro a la cual se le pasa como parámetro la descripción del rubro obtenida del campo PUBLICACION_RUBRO_DESCRIPCION de la tabla maestra.

  Cabe aclarar que los registros que se insertan se hacen de forma ordenada según los valores del campo PUBLICACION_COD.

  A su vez, solo se insertan los registros cuyos valores son distintos. Para esto se utilizó la sentencia DISTINCT.

- **Calificaciones:** Se migran a la tabla CALIFICACIONES, los datos de las columnas CALIF_COD, CALIF_ESTRELLAS y CALIF_DESC, correspondientes a las siguientes columnas de la tabla maestra respectivamente CALIFICACION_CODIGO, CALIFICACION_CANT_ESTRELLAS y CALIFICACION_DESCRIPCION.

  Para realizar la migración solo se verifica que el valor del campo CALIFICACION_CODIGO no carezca de valores. En caso de que el campo sea NULL no se inserta en la tabla CALIFICACIONES.

  A su vez, solo se insertan los registros cuyos valores son distintos. Para esto se utilizó la sentencia DISTINCT.

Antes de realizar la inserción de registros, se activa el IDENTITY_INSERT sobre la tabla CALIFICACIONES para que el motor de la Base de Datos permita insertar valores en el campo CALIF_COD que posee la constraint IDENTITY (1,1).

Luego de realizar la migración de dicha tabla, se deshabilita el IDENTITY_INSERT para que de ahí en adelante a todas las inserciones de registros que se hagan, se les asigne de forma automática un número de identidad.

- **Compras:** Se migran a la tabla COMPRAS, los datos de las columnas COMPRA_FEC, COMPRA_CANT, COMP_PUB_COD y COMP_CALIF_COD, correspondientes a las siguientes columnas de la tabla maestra respectivamente COMPRA_FECHA, COMPRA_CANTIDAD, PUBLICACION_COD y CALIFICACION_CODIGO.

  En cuanto a los valores de la columna COMP_VENDEDOR_ID, estos se obtienen mediante la función obtenerIDusuario a la cual se le pasa como parámetros el valor 'DNI', el número de Documento y el número de CUIT.

  En cuanto a los valores de la columna COMP_COMPRADOR_ID, estos se obtienen mediante la función obtenerIDusuario a la cual se le pasa como parámetros el valor 'DNI', el número de Documento y NULL.

  Para realizar la migración se verifica que los campos COMPRA_FECHA y CALIFICACION_CODIGO no carezcan de valores. En caso de que alguno de los campos sea NULL no se inserta en la tabla COMPRAS.

  Cabe aclarar que los registros que se insertan se hacen de forma ordenada según los valores de los campos PUBLICACION_COD y VENDEDOR (valor obtenido en la primera llamada a la función obtenerIDusuario).

- **Ofertas:** Se migran a la tabla OFERTAS, los datos de las columnas OF_PUB_COD, OFERTA_FECHA y OFERTA_MONTO, correspondientes a las siguientes columnas de la tabla maestra respectivamente PUBLICACION_COD, OFERTA_FECHA y OFERTA_MONTO.

  En conjunto con los datos anteriores, se completa el campo OF_USUARIO_ID mediante la función obtenerIDusuario a la cual se le pasa como parámetros el valor 'DNI', el número de documento obtenido del campo CLI_DNI y NULL.

  Para realizar la migración solo se verifica que el valor del campo OFERTA_FECHA no carezca de valores. En caso de que el campo sea NULL no se inserta en la tabla OFERTAS.

  Cabe aclarar que los registros que se insertan se hacen de forma ordenada según los valores de los campos PUBLICACION_COD y ID_CLIENTE.

A su vez, solo se insertan los registros cuyos valores son distintos. Para esto se utilizó la sentencia DISTINCT.

- **Factura:** Se migran a la tabla FACTURA, los datos de las columnas agrupadas FACTURA_NRO, FACTURA_FECHA, FACTURA_TOTAL y FACTURA_FORMA_PAGO, correspondientes a las siguientes columnas de la tabla maestra respectivamente FACTURA_NRO, FACTURA_FECHA, FACTURA_TOTAL y FORMA_PAGO_DESC.

  En cuanto a los valores de la columna FACT_USUARIO_ID, estos se obtienen mediante la función obtenerIDusuario a la cual se le pasa como parámetros el valor 'DNI' y el número de documento y retorna el número identificador del usuario.

  Para realizar la migración solo se verifica que el valor del campo FACTURA_NRO no carezca de valores. En caso de que el campo sea NULL no se inserta en la tabla FACTURA.

  Antes de realizar la inserción de registros, se activa el IDENTITY_INSERT sobre la tabla FACTURA para que el motor de la Base de Datos permita insertar valores en el campo PUBLICACION_NRO que posee la constraint IDENTITY (1,1).

  Luego de realizar la migración de dicha tabla, se deshabilita el IDENTITY_INSERT para que de ahí en adelante a todas las inserciones de registros que se hagan, se les asigne de forma automática un número de identidad.

  Cabe aclarar que los registros que se insertan se hacen de forma ordenada según los valores del campo FACTURA_NRO.

- **PubliXItem_Factura:** Se migran a la tabla PUBLIXITEM_FACTURA, los datos de las columnas PUBLITEM_FACT_NRO y PUBLITEM_PUB_COD, correspondientes a las siguientes columnas de la tabla maestra respectivamente FACTURA_NRO y PUBLICACION_COD.

  Para realizar la migración solo se verifica que los campos FACTURA_NRO y PUBLICACION_COD no carezcan de valores. En caso de que alguno de los campos sea NULL no se inserta en la tabla PUBLIXITEM_FACTURA.

  A su vez, solo se insertan los registros cuyos valores son distintos. Para esto se utilizó la sentencia DISTINCT.

  Cabe aclarar que los registros que se insertan se hacen de forma ordenada según los valores del campo FACTURA_NRO.

- **Item_Factura:** Se migran a la tabla ITEM_FACTURA, los datos de las columnas ITEM_PUB_COD, ITEM_CANTIDAD e ITEM_MONTO, correspondientes a las siguientes

columnas de la tabla maestra respectivamente PUBLICACION_COD, ITEM_FACTURA_CANTIDAD y ITEM_FACTURA_MONTO.

Para realizar la migración solo se verifica que los campos FACTURA_NRO y PUBLICACION_COD no carezcan de valores. En caso de que alguno de los campos sea NULL no se inserta en la tabla ITEM_FACTURA.

Vistas:

Para el desarrollo del trabajo práctico no utilizamos ninguna vista.

Funciones:

A lo largo de todo el desarrollo de la aplicación fuimos necesitando crear distintas funciones para calcular ciertas cosas. A continuación nombraremos todas las funciones utilizadas y una breve descripción de cada una:

- **obtenerIDrubro:** Función a la cual le paso como parámetro el nombre de un rubro y me devuelve un valor numérico que corresponde al número identificador del rubro. En caso de que no exista el rubro devuelve 9999999.

- **obtenerIDusuario:** Función a la cual le paso como parámetros el tipo y número de documento en caso de ser un cliente, y el CUIT en caso de ser una empresa. La función retorna un valor numérico que corresponde al número identificador del usuario.

- **obtenerIDcliente:** Función a la cual le paso como parámetros el tipo y número de documento del cliente y retorna un valor numérico que corresponde al número identificador del cliente.

- **generarUsuario:** Función a la cual le paso como parámetros un identificador numérico y el tipo de usuario y retorna una cadena alfanumérica identificadora de usuario la cual se compone de acuerdo a la siguiente nomenclatura: *be_sharps_[tipo de usuario][identificador de usuario]*

Stored Procedures:

Durante el desarrollo de la aplicación fuimos necesitando crear distintos Stored Procedures para distintas cosas como por ejemplo para dar altas, bajas y realizar modificaciones. A continuación explicaremos brevemente su gran mayoría:

- **facturarPublicaciones:** lo llama un usuario cuando desea facturar publicaciones que esten finalizadas o vencidas. Lo que hace el procedimiento es recibir una lista de publicaciones para facturar y hace la factura por la cantidad vendida que es la comision que se paga por publicar y la visibilidad de la publicacion.

- **ofertar:** inserta en la tabla OFERTAS un registro con todos los datos correspondientes a la misma.

- **comprar:** inserta en la tabla COMPRAS un registro con todos los datos correspondientes a la misma y actualiza el stock.

- **obtenerReputacion:** retorna la cantidad de estrellas que recibió el vendedor, la cantidad de ventas que realizó, la cantidad de calificaciones que recibió y finalmente su reputación. Esta última se calcula haciendo la cantidad de estrellas que recibió dividido la cantidad de calificaciones que recibió.

- **insertarEmpresa:** inserta una empresa en la tabla EMPRESAS junto con todos sus datos. Luego genera el usuario correspondiente insertando un registro en la tabla USUARIOS.

- **eliminarEmpresa:** modifica el estado de habilitación de los usuarios correspondientes a una empresa recibida como parámetro.

- **modificarEmpresa:** modifica los datos de la empresa recibida como parámetro.

- **crearPublicacion:** inserta en la tabla PUBLICACIONES un registro con todos los datos correspondientes a la publicación.

- **agregarRubroAPublicacion:** asocia una publicación determinada a un rubro determinado.

- **modificarCliente:** modifica los datos del cliente recibido como parámetro.

- **eliminarCliente:** modifica el estado de habilitación del usuario cuyo nombre es el cliente recibido como parámetro.

- **insertarCliente:** inserta un cliente con todos los datos recibidos como parámetro.

- **insertarRol_Funcion:** inserta una función a un rol. Esto se realiza mediante la inserción de un registro en la tabla ROL_FUNCION.

- **obtenerCodigo:** obtiene el ID del rol recibido como parámetro.

- **p_insertar_visib:** inserta un registro en la tabla VISIBILIDAD.

- **insertar_pub:** inserta un registro en la tabla PUBLICACIONES.

- **crear_tablas:** crea todas las tablas detalladas en el diagrama entidad relación. Ejecuta todas las sentencias CREATE TABLE. Junto con esto se crean las constraints necesarias.

- **realizar_migracion:** procesa la tabla maestra provista por la catedra y carga todas las tablas previamente creadas. Basicamente lo que se realiza son sentencias INSERT y ALTER TABLE para activar o desactivar una constraint como IDENTITY.

- **pausarUser:** dado un ID de usuario obtiene todas las publicaciones del mismo que se encuentren en estado 'Publicada' y las recorre mediante un cursor modificándoles el estado a 'Pausada'.

- **modificarPublicacion:** modifica los datos de la publicación recibida como parámetro.

- **tipoUsuario:** dado un ID de usuario devuelve el tipo y el ID del vendedor. El tipo puede ser C (Cliente) o E (Empresa).

- **calificar:** crea una calificación correspondiente a una compra.

Nota: en todos los casos se realizan validaciones de tipo de dato (desde la aplicación C #), para que no se ingresen caracteres inválidos, ni números en los lugares que no corresponde.

Triggers:

Para el desarrollo del trabajo práctico fue necesaria la creación de los siguientes triggers. A continuación explicamos brevemente cada uno:

- **tr_actualizar_permisos:** Se dispara luego de insertarse un registro en la tabla USUARIO_ROL en la cual se le asigna un rol a un usuario que ya tenía dicho rol. En caso de existir se realiza un rollback de la transacción, caso contrario se realiza el commit.

- **tr_finalizarSubasta:** Se dispara cuando se finaliza una subasta y en caso de que haya una subasta ganadora se inserta en la tabla Compras.

- **tr_pubSinStock:** Se dispara cuando se inserta una publicación y crea un cursor que recorre las publicaciones y verifica que las que se encuentren en estado 'Compra Inmediata' tengan stock igual a 0. En dicho caso les modifica el estado a 'Finalizada' ya que significa que se quedó sin stock.

Índices:

Para el trabajo práctico utilizamos los índices que se generan automáticamente al emplear Primary Keys, Foreign Keys y constraints UNIQUE.

Aplicación Desktop en Visual C#:

Los criterios más importantes al momento de programar en C# fueron:

- Utilizamos una única conexión para acceder a la base de datos, la cantidad de veces que sea necesaria.
- Se utiliza try / catch cuando se quiere acceder a la base, para que en caso de que se produzca un error, el programa no termine de manera inesperada.
- Investigación del algoritmo de encriptación SHA-256.
- Validaciones en el ingreso de datos, para evitar inconsistencias.
- Utilización de nombre de objetos representativos, y colocando delante de cada objeto la abreviación del tipo de objeto.

Conclusión del trabajo practico:

Finalizado el trabajo práctico podemos decir que el mismo nos demandó un esfuerzo de investigación importante debido a que ninguno de los cuatro integrantes del grupo conocía SQL ni C#. La temática del trabajo practico nos pareció muy interesante debido a que es un tema que hoy en día todos utilizamos o lo hicimos alguna vez, para adquirir un determinado bien o servicio a través de internet.

Nos resultó muy importante poder relacionar conceptos no solo vistos en clase, sino en otras materias. Pudimos llevar a cabo básicamente el proceso de normalización visto en Diseño de Sistemas, así como el manejo del Diagrama Entidad Relación visto en Análisis de Sistemas. Pudimos también aplicar conceptos sobre el paradigma objetos, visto en Paradigmas de Programación.

Tuvimos que utilizar herramientas muy usadas hoy en día, como lo son Microsoft Visual C# y el SQL Server.

Debido a lo detallado anteriormente, podemos concluir que el trabajo práctico fue muy provechoso para cada uno de los integrantes del grupo ya que los conocimientos adquiridos no solamente nos serán útiles para completar nuestra formación académica sino que también para nuestro desarrollo laboral.