Titulación: GRADO INGENIERIA INFORMATICA
Año Académico: 2023/24 – 2º cuatrimestre
Asignatura: Ficheros y bases de datos (2º curso)
Memoria de Prácticas 1: Diseño Relacional y Carga de Datos

uc3m | Universidad Carlos III de Madrid

Profesor:	**Pablo Manuel Vigara Gallego**	**Grupo**	**85**
Alumno/a:	**Iván Sebastián Loor Weir**	**NIA:**	**100448737**
Alumno/a:	**Álvaro González Fúnez**	**NIA:**	**100451281**
Alumno/a:	**Arturo Jiménez Tajuelo Pérez**	**NIA:**	**100451161**

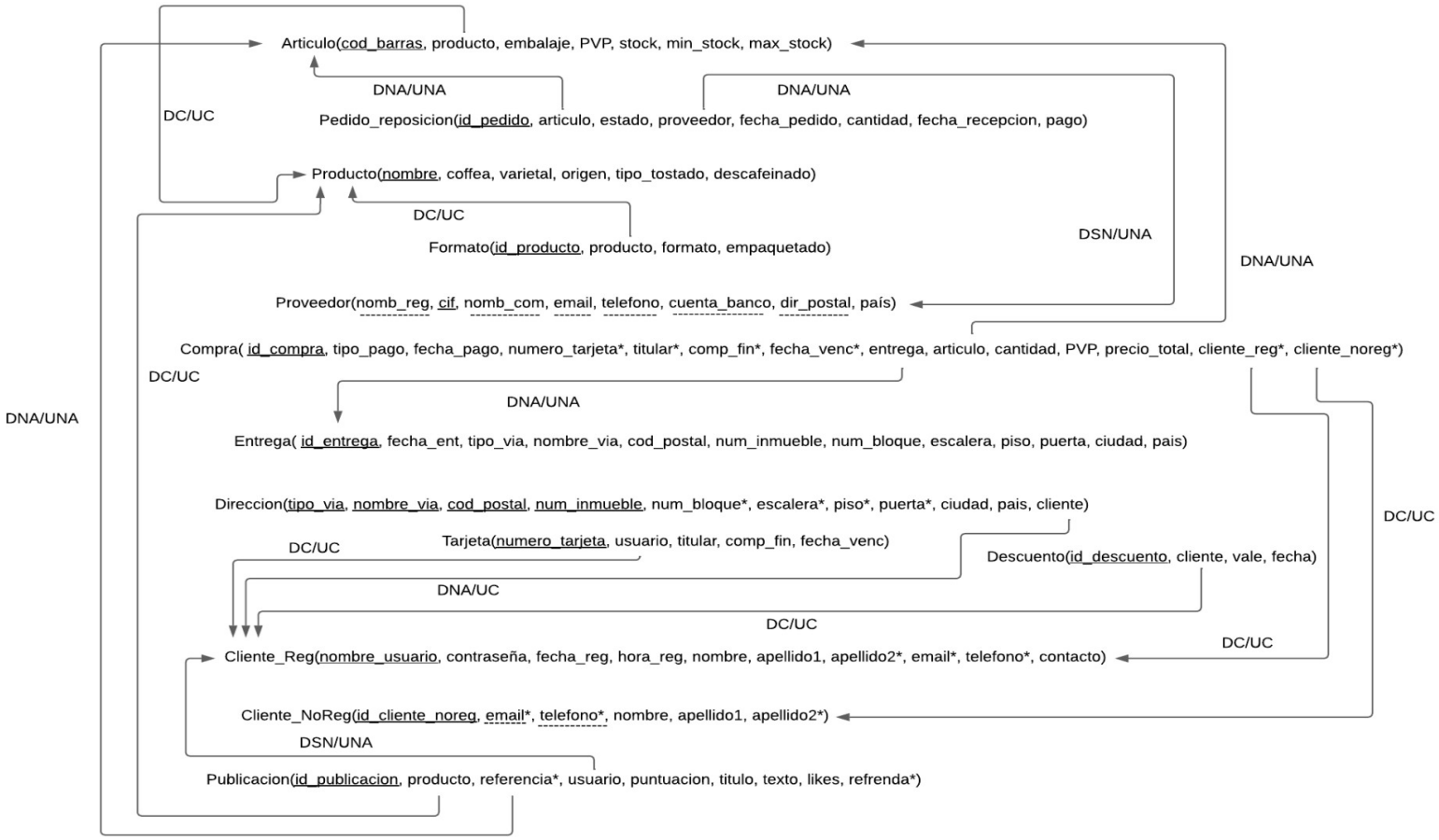
1. Introducción

Este proyecto consiste en el diseño e implementación de una nueva base de datos para el registro y administración de los datos de CaffeineManiacs TM Inc, una empresa de comercio electrónico. En este proceso, hemos elaborado una especificación detallada que abarca los servicios ofrecidos, los proveedores asociados y la información de los clientes.
En primer lugar, hemos creado un esquema relacional, incorporando su semántica implícita y la semántica explícita contemplada. Posteriormente, creamos las tablas nuevas con sus restricciones correspondientes en SQL y finalmente insertamos datos de las antiguas tablas en las nuevas.

2. Diseño Relacional

Esta sección se subdivide en tres apartados:

- Esquema relacional: diseño completo, con la notación de grafo/esquema relacional vista en clase (puede entregarse hecho a mano o con las herramientas de dibujo proporcionadas por Microsoft Office). Es importante que el grafo se visualice claramente.



- Semántica implícita: supuestos semánticos que, por referirse a información ausente en la descripción explícita (es decir, no se encuentran en el enunciado), es necesario añadir para completar el diseño.

Sup_id	Mecanismo	Descripción
I_1	Clave primaria	El pedido de reposición se identifica mediante un id de pedido.
I_2	Clave primaria	El formato del producto se identifica por un id.
I_3	Clave primaria	La compra se identifica mediante un id de compra.
I_4	Clave primaria	La entrega se identifica mediante un id de entrega.
I_5	Clave primaria	La dirección se identifica mediante el tipo de vía, el nombre de la vía, el código postal y el número de inmueble.
I_6	Clave primaria	El descuento se identifica mediante un id de descuento.
I_7	Clave primaria	El cliente no registrado tiene un identificador para poder ser encontrado (no podemos usar teléfono o correo electrónico como claves primarias ya que en el enunciado se indica que una de las dos puede ser nula ya que el usuario no

		registrado debe aportar al menos una, pero no las dos).
I_8	Clave primaria	Las publicaciones se identifican mediante sus propias id.
I_9	Clave primaria	En la tarjeta utilizamos como clave primaria el número de tarjeta.
I_{10}	Semántica	Si se elimina un producto, se eliminan los artículos asociados a este. Así mismo, si se modifica un producto, también se actualizarán los artículos del producto.
I_{11}	Semántica	Por otra parte, si se elimina un producto, las publicaciones asociadas a este se eliminarán y se actualizarán en cascada.
I_{12}	Semántica	Si un usuario registrado se borra de la base de datos, establecemos el valor de usuario en la relación publicaciones como NULL, lo cual equivale a que sus comentarios permanezcan.
I_{13}	Semántica	Si un usuario registrado se borra de la base de datos, sus tarjetas de crédito también serán eliminadas. Si se actualiza un usuario, también lo harán las tarjetas.
I_{14}	Semántica	Si un usuario registrado es borrado, los descuentos de este usuario se borrarán. En el caso de actualizar un usuario, se modifican también los descuentos.
I_{15}	Semántica	La actualización de un usuario registrado afecta también a sus direcciones.
I_{16}	Semántica	Si se borra un usuario registrado, también se eliminan sus compras. Si se actualiza, también se actualizan sus compras.

Tabla 1: Semántica implícita

- Semántica explícita no contemplada en el diseño: supuestos semánticos indicados en el enunciado que no han podido representarse en el esquema relacional. Para cada uno de los supuestos, crea una fila en la tabla presentada a continuación.

Sup_id	Descripción
S_1	No hemos implementado un stock mínimo diferente para cada referencia (5 unidades, por defecto).
S_2	No hemos tenido en cuenta los términos “min_stock” y “max_stock” para la actualización de los pedidos, aunque se encuentran representados en el diseño.
S_3	No hemos contemplado el proceso de reposición automática que llenará el stock hasta el umbral máximo.
S_4	A la hora de seleccionar proveedores, no los comparamos por costes, tiempos de entrega ni cantidad de pedidos.
S_5	No actualizamos la cantidad al máximo ni generamos un error cuando el stock de una determinada referencia no es suficiente.

S_6	No hemos considerado la validez de la fecha de los descuentos, así como, no implementamos su cálculo en el grafo relacional.
S_7	No podemos asegurar que no se pueda realizar más de un pedido por referencia y día.
S_8	No se puede ilustrar el cambio de estado en los pedidos por referencia.
S_9	No se pueden aplicar las reglas de integridad referencial en los estados del pedido debido a que el estado es simplemente un atributo.
S_{10}	No podemos garantizar que si un cliente registrado se elimina, todas sus compras se conviertan en compras de clientes no registrados.
S_{11}	No podemos asegurar que en la preferencia de contacto se opte por utilizar una de las varias opciones que tiene.
S_{12}	No se puede asegurar que las publicaciones tengan una puntuación compuesta de números naturales (1-5).
S_{13}	No podemos establecer un valor por defecto en el atributo likes ni mostrar su acumulación propia.
S_{14}	No podemos garantizar que las publicaciones/comentarios que se etiqueten con una refrenda provengan de un cliente registrado que ya haya consumido ese producto o artículo (referencia) anteriormente.

Tabla 2: Semántica explícita no contemplada

3. Implementación de la Estática Relacional en SQL (LDD)

Esta sección complementa al fichero con el script de creación de la base de datos (NEWcreation.sql). Añadiendo los siguientes apartados:

Semántica explícita re-incorporada: Incluir aquellos supuestos de la Tabla 2 que se han podido contemplar con las sentencias de definición de SQL.

Sup_id	Descripción de la solución
S_2	CONSTRAINT ck_articulo_3 CHECK (stock>=min_stock AND stock<=max_stock) -> Comprobamos que el stock se encuentre entre los valores mínimo y máximo
S_8	CONSTRAINT ck_pedido CHECK (estado IN ('draft', 'placed', 'fulfilled')) -> Comprobamos que el estado del pedido solamente pueda tener esos tres valores
S_{12}	CONSTRAINT ck_publicacion_2 CHECK (puntuacion BETWEEN 1 AND 5) -> Comprobamos que la puntuación de una publicación sea un número entre el 1 y el 5
S_{13}	likes NUMBER(9) DEFAULT 0 not null, CONSTRAINT ck_publicacion CHECK (likes>=0) -> Comprobamos que la publicación no tenga un número de likes negativos

Tabla 3: Semántica explícita re-incorporada

Semántica implícita: (continúa la numeración donde terminó en la tabla 1)

Sup_id	Mecanismo	Descripción
I_{17}	Check (Tabla Cliente_NoReg)	Comprobamos que el email corresponde al formato <texto>@<texto>
I_{18}	Check (Tabla Cliente_NoReg)	Comprobamos que el número tiene 9 dígitos exactamente
I_{19}	Check (Tabla Cliente_NoReg)	El email o el telefono son al menos uno de los dos no nulo
I_{20}	Check (Tabla Cliente_NoReg)	Aunque la clave primaria es id_cliente_noreg, establecemos email y telefono como unique para que no se repitan
I_{21}	Check (Tabla Cliente_Reg)	El email es <texto>@<texto>
I_{22}	Check (Tabla Cliente_Reg)	La longitud del atributo número es de 9 dígitos
I_{23}	Check (Tabla Cliente_Reg)	El email o el telefono son al menos uno de los dos no nulo
I_{24}	Check (Tabla Cliente_Reg)	Limitamos los valores de la variable contacto a los especificados en la tabla
I_{25}	Check (Tabla Producto)	Los valores de descafeinado son si (yes) o no
I_{26}	Check (Tabla Producto)	El tipo de tostado es natural, high-roast o blend
I_{27}	Check (Tabla Proveedor)	La longitud del cif del proveedor es de 10 caracteres
I_{28}	Check (Tabla Proveedor)	El correo electronico tiene el formato <texto>@<texto>
I_{29}	Check (Tabla Proveedor)	La longitud del telefono es de 9 dígitos
I_{30}	Check (Tabla Proveedor)	La longitud de la cuenta bancaria es de 17 digitos
I_{31}	Check (Tabla Tarjeta)	La longitud del número de tarjeta es 20
I_{32}	Check (Tabla Formato)	La expresion de empaquetado se ajusta a <texto> <numero> <unidad de medida>.
I_{33}	Check (Tabla Articulo)	La longitud del código de barras es de 15 dígitos exactos
I_{34}	Check (Tabla Articulo)	La expresion del embalaje corresponde al formato <texto> <numero> <unidad de medida>.
I_{35}	Check (Tabla Articulo)	El stock actual está en el rango [min_stock, man_stock]
I_{36}	Check (Tabla Publicacion)	La longitud de referencia (código de barras) es de 15 caracteres
I_{37}	Check (Tabla Articulo)	El valor de la puntuacion se encuentra entre 1 y 5
I_{38}	Check (Tabla	El número de likes de una publicaciones

	Articulo)	es igual o mayor a 0
I_{39}	Check (Tabla Publicación)	Refrenda solo pueda tomar valores TRUE y FALSE
I_{40}	Check (Tabla Pedido_reposicion)	El estado solo está en una de esas 3 formas
I_{41}	Check (Tabla Compra)	La compra pertenece a un cliente registrado o a un cliente no registrado
I_{42}	Clave primaria (Tabla Cliente_NoReg)	Aunque la clave primaria es id_cliente_noreg, establecemos email y telefono como unique para que no se repitan
I_{43}	Clave primaria (Tabla Formato)	Clave primaria es un id de formato incremental
I_{44}	Clave primaria (Tabla Publicación)	Clave primaria es una id generada de forma incremental
I_{45}	Clave primaria (Tabla Comentario)	Clave primaria generada mediante una id en forma ascendente
I_{46}	Clave primaria (Tabla Entrega)	Clave primaria generada mediante una id en forma ascendente
I_{47}	Clave primaria (Tabla Direccion)	Clave primaria compuesta por 4 valores siempre presentes en las direcciones
I_{48}	Clave primaria (Tabla Pedido_reposicion)	Clave primaria generada mediante una id en forma ascendente
I_{49}	Clave primaria (Tabla Compra)	Clave primaria generada mediante una id en forma ascendente

Tabla 1(cont.): Semántica implícita

Semántica excluida: Al crear la base de datos en SQL específico del SGBD Oracle puede que no se hayan podido contemplar algunas restricciones semánticas explícitas (tabla 2 – tabla 3), o implícitas que no han podido incorporarse (tabla 1).

Sup_id	Descripción semántica	Motivo	Explícita/ Implícita
E_1	No se ha podido incorporar aumentar el % de descuento por cada 10 euros de compra	Harían falta triggers para poder implementarlo correctamente.	Explícita
E_2	No se ha podido implementar la actualización de cantidad al maximo si no hay stock suficiente, ni generar su error asociado	Se deberían usar vistas o triggers para evaluar en versiones más recientes la cantidad de stock disponible.	Explícita
E_3	No podemos garantizar que si un cliente registrado se elimina, todas sus compras se conviertan en compras de clientes no registrados.	Creemos que con los triggers sería posible esta opción.	Explícita
E_4	No podemos asegurar que no se pueda realizar más de un pedido por referencia y día.	Se deberían usar triggers o vistas para que solo se puedan hacer inserciones de pedido por día.	Explícita

E_5	No podemos garantizar que los comentarios que se etiqueten con una refrenda provengan de un usuario registrado que ya haya consumido ese producto/referencia anteriormente	No se pueden verificar en el tiempo los productos que ya haya consumido tal cliente y por eso serían mejor los triggers. para dar con la solución del problema.	Explícita
E_5	El formato de correo electrónico de las tablas Proveedor, Cliente_Reg, Cliente_NoReg es válido.	Se utilizó la función REGEXP_LIKE(email, '^[a-zA-Z0-9]+(@).+$') para que la expresión sea <texto>@<texto>, a pesar de que sabemos que sigue siendo muy permisiva.	Implícita

Tabla 4: Semántica excluida en la creación de tablas

4. Carga de datos (LMD)

Esta sección describirá la carga de datos realizada desde las tablas desnormalizadas entregadas junto con la entrega del fichero de carga (NEWload.sql). A tal efecto, se analizará el problema de la carga y se describirá la solución, haciendo hincapié en:

- El orden de tablas que se adopta para volcar en ellas los datos (justificado).
- Los problemas que surgen (campos obligatorios sin valor, defectos en los datos originales, conversiones de datos, etc) y las soluciones que se adoptan para superarlos.

El orden de creación de tablas se ha decidido en base a las restricciones de clave ajena. Estas restricciones impiden que algunas tablas se creen o se borren antes que otras. Las tablas sin estas restricciones deben ser las primeras en crearse y rellenarse, y las últimas en borrarse.
Es por eso que comenzamos con la tabla sencilla de Producto, seguida de la tabla Articulo que lo referenciaba. Luego, seguimos con Proveedor y Pedido_Reposicion que lo referenciaba junto a Articulo. Después, continuamos con Formato, Cliente_NoReg, Cliente_Reg, Entrega y Compra que referenciaba a los dos clientes, la entrega y el artículo. Finalmente, terminamos con las últimas tablas que contenían restricciones variadas como Direccion, Descuento, Tarjeta y Publicacion.

Los problemas encontrados se irán explicando tabla por tabla:

- Cliente_NoReg: Esta tabla sirve para almacenar los clientes no registrados, ya que carecen de suficientes datos como para pertenecer a una tabla global de clientes y hallar una forma de diferenciarlos, además de no tener los clientes no registrados un identificador adecuado para su uso. El motivo por el que no se halló ninguna clave primaria es debido a que tanto el teléfono como el email podían ser nulos (a pesar de ser únicos) y el nombre no es muy distintivo como para serlo. Entonces se adoptó la medida de crear un id que se incrementase por cada inserción metida en la tabla. Luego, el

teléfono lo transformamos como un número de 9 dígitos exactos, dejamos el último apellido como que podía ser nulo y verificamos que el email tuviera el siguiente formato para ser aceptado: <texto>@<texto>. Para ese formato utilizamos la función REGEXP_LIKE que devuelve un booleano después de realizar la coincidencia del patrón establecido. Es por esto, que la inserción de datos no tuvo grandes percances y solo tuvimos que enfocarnos en insertar los clientes no registrados (sin username).

- Cliente_Reg: Esta tabla sirve para almacenar los clientes registrados. Se utilizó el nombre de usuario como clave primaria y se dejaron los atributos homónimos a la tabla no registrada con el mismo tipo de datos. Para la contraseña decidimos dejarla como varchar2, pero para la fecha de registro utilizamos el tipo de dato TIMESTAMP que guarda fecha y hora, teniendo un 0 como parámetro para olvidarnos de las fracciones de segundo. Decidimos implementar así la fecha para ahorrarnos atributos de más y porque era complicado guardar en otro atributo solo la hora sin que apareciera la fecha. En cuanto a la preferencia de contacto, dejamos como valor por defecto 'sms' y comprobamos al hacer la inserción que si existe el correo lo cambie a email y en cualquier otro caso lo deje como estaba. Chequeamos también que la preferencia de contacto solo pueda tener las palabras reservadas ('sms', correo, 'facebook', etc) así como las demás restricciones que tenía Cliente_NoReg. Al insertar los datos solo tuvimos que modificar la forma de cómo insertamos la fecha y decidimos utilizar TO_TIMESTAMP para juntar los dos atributos de fecha y hora bajo el mismo formato en nuestro atributo ya anteriormente inicializado.

- Producto: En la tabla dejamos el nombre como su identificador y los demás atributos como varchar2, teniendo en cuenta que el descafeinado y el tipo de tostado solo deben tener unos valores específicos. Para verificar que tuvieran tales valores usamos la función trim para borrar los espacios en blanco que poseían, ahora sí insertar los datos y hacer la comparación correctamente.

- Proveedor: Para empezar, usamos el cif como clave primaria y fuimos agregando como únicos todos los atributo excepto el país. Después fuimos chequeando que la longitud de estos fueran constantes para lanzar excepción en caso contrario, junto al formato específico del correo como en las anteriores tablas y sólo necesitamos utilizar la función trim en la cuenta bancaria ya que al principio no se comparaban bien los datos.

- Tarjeta: La clave primaria escogida fue el número de tarjeta y los demás datos se guardaron estableciendo límites de longitud. Al principio, nos dió problemas al guardarlo como número ya que se ponía como número en notación científica y nos daba muchos problemas en la inserción, es por eso que se dejó como varchar2. En lo referente a la fecha, elegimos DATE como el tipo de dato ya que solo se necesita guardar el mes y el año. Luego, en la inserción de la fecha utilizamos TO_DATE con el formato específico para la fecha de expiración proporcionada. Por último, tuvimos que aclarar que el cliente tenía que existir y obviamente el número de tarjeta para introducir todos los datos posibles.

- Formato: Usamos como clave primaria un identificador de formato que se genera de forma automática. Dentro del insert, tendríamos que insertar tres campos de la tabla

fsdb.catalogue: product, format y packaging (aunque la tabla tiene 4 atributos pero la clave primaria se genera automáticamente cada vez que se introduce un formato dentro de la tabla). En este caso, hemos aplicado un trim() a product para poder eliminar los espacios en blanco al final del atributo debido a que es un tipo CHAR y que nos permite poder comparar Product con los productos ya introducidos en la tabla Producto creada previamente

- Artículo: En artículo hemos utilizado el código de barras de referencia como clave primaria. En esta tabla hemos insertado, además del código de barras (barcode en la tabla antigua), el producto al que hace referencia el artículo, el empaquetado de este, el precio de venta, y tanto el stock actual como el stock máximo y mínimo, todos estos datos procedentes de la tabla fsdb.catalogue. Al igual que en formato, usamos trim() en Product para poder referenciar correctamente la tabla Producto, y además transformamos a número con TO_NUMBER los datos referentes al stock y al precio de venta. En este último caso además hemos utilizado la función REGEXP_REPLACE para sustituir la expresión de la tabla antigua en formato "número_decimal ?" por un número que no contenga la interrogación.

- Publicación: Utilizamos como clave primaria un identificador de publicación que se genera de forma automática al insertar datos en la tabla, lo que provoca que en el insert no haga falta incluirlo. Respecto al resto de datos, insertamos de la tabla antigua fsdb.posts los atributos del producto (con el trim() correspondiente para poder referenciar la tabla producto), el código de barras, el nombre de usuario que hace la publicación, la puntuación, el título, el texto, el número de likes y la refrenda. También cabe destacar que con un TO_NUMBER() hemos transformado a número los datos de puntuación y likes. Además de eso, tuvimos problemas visualizando lo que debía ser un comentario. Creamos una tabla comentario que poseía una id y la refrenda como atributo opcional, pero terminamos descartando esta idea e interpretamos que el comentario era igual a una publicación. Para esto, añadimos la refrenda al final de la tabla Publicación como atributo opcional que debía dar un resultado booleano debido a que al tratarse de una cuestión de etiquetación o no (y no contar con datos a los que comparar en la tabla vieja), lo dejamos implementado de esa forma. Por último, también recalcar que hemos usado un inner join para poder introducir las publicaciones sobre productos existentes, ya que al tratar de insertar los datos observamos que una de las publicaciones hacía referencia a un producto que no existía y por tanto descartamos ese caso concreto al ser un producto que no existe.

- Entrega: Esta tabla almacena las direcciones de entrega a las que van asociadas las compras. Decidimos utilizar como clave primaria un id ascendente debido a que las entregas se podían hacer en las mismas direcciones. Para la fecha de entrega volvimos a usar TIMESTAMP(0) como tipo de dato, juntando la fecha y hora con TO_TIMESTAMP y su formato correspondiente en la inserción de datos. Seguidamente, inicializamos el código postal, el número de bloque y escalera como números enteros y no los demás atributos porque había casos donde tenían letras. También utilizamos la función trim() para insertar bien los datos de escalera igual que en la tabla Direccion ya que al principio nos daba problemas al insertar.

- Dirección: Aquí se guardan las direcciones de los clientes registrados. En este caso, los tipos de datos de los atributos se inicializaron de igual forma que en la tabla Entrega. Cabe recalcar que como aquí cada dirección necesitaba diferenciarse, tuvimos que usar varios atributos como clave primaria. En principio, creímos que el nombre de vía, tipo de vía y código postal bastarían, pero pronto contemplamos problemas de clave violada y después de analizar en profundo los datos de la tabla vieja concluimos que el número de inmueble debía estar también en la clave primaria, a pesar de que no lo decía claramente el enunciado, perteneciendo en consecuencia a la semántica implícita.

- Compra: La tabla compra contiene un id de compra que se genera automáticamente en función de los datos que vayamos introduciendo en la tabla. Posteriormente, introduciremos los datos de las tablas antiguas en compra. Para empezar, de fsdb.trolley introduciremos el tipo de pago, la fecha y hora de pago concatenadas y en formato timestamp en un único atributo, el número, propietario, compañía y fecha de caducidad (formato date 'MM/YY') de la tarjeta de crédito (en caso de que se haya pagado con tarjeta), la cantidad, el precio de base en formato numérico, el precio total (también en formato numérico y haciendo una multiplicación de la cantidad por el precio base) y el usuario que realiza la compra. Además, cogeremos de otras tablas los datos del código de barras del artículo, el id_ entrega, el nombre de usuario del cliente registrado y el id del cliente no registrado (teniendo en cuenta que las compras pueden hacerlas usuarios registrados o no registrados). Finalmente, para esta tabla hay que hacer uso de dos join para juntar los artículos y entregas de las compras y un full outer join para poder incluir compras de usuarios registrados y no registrados.

- Pedido_Reposición: En esta tabla empleamos un identificador progresivo igual que los ya usados en otras tablas anteriormente. La fecha de pedido y de recepción utilizan el mismo tipo de dato TIMESTAMP(0) en la creación de la tabla y luego unen sus fechas y horas correspondientes usando TO_TIMESTAMP unido al formato específico para insertarlo en sus dos atributos creados. Más adelante dejamos la cantidad como un entero, pero el pvp lo dejamos como decimal (haciendo uso de number) así como el precio total a calcular multiplicando el pvp por la cantidad. De paso, chequeamos que el estado del pedido solo pueda comprender tres valores y que solo pueda existir uno de los dos tipos de clientes (registrado o no registrado). En la inserción también utilizamos la función REGEXP_REPLACE junto a su formato numérico para insertar bien el precio de costo y poder ser calculado después en el precio total. Y ya al final partimos de los datos proporcionados por dos tablas viejas y nuestra tabla Articulo haciendo los correspondientes joins para lograr la inserción deseada.