Profesor:	**Pablo Manuel Vigara Gallego**	**Grupo**	**85**
Alumno/a:	**Fernando López de Olmedo Rodríguez-Solano**	**NIA:**	**100495977**
Alumno/a:	**Héctor Herráiz Díez**	**NIA:**	**100499734**
Alumno/a:	**Jorge Mejías Donoso**	**NIA:**	**100495807**
Alumno/a:	**Javier Moyano San Bruno**	**NIA:**	**100495884**

1. Introducción

El presente documento constituye la memoria del trabajo, centrándose en el diseño, implementación y carga de una base de datos para un comercio online. Este proyecto responde a la necesidad de gestionar eficientemente los productos, artículos, proveedores y demás elementos esenciales para el funcionamiento de una plataforma de este tipo, para brindar una experiencia óptima a los usuarios.

El propósito fundamental de este trabajo es abordar la gestión de información en un entorno digital como un comercio online, proponiendo una solución mediante la construcción de una base de datos relacional. Para ello, se estructura el documento en tres apartados principales que exploran diferentes aspectos del proyecto: diseño relacional, implementación de la estática relacional en SQL y carga de datos.

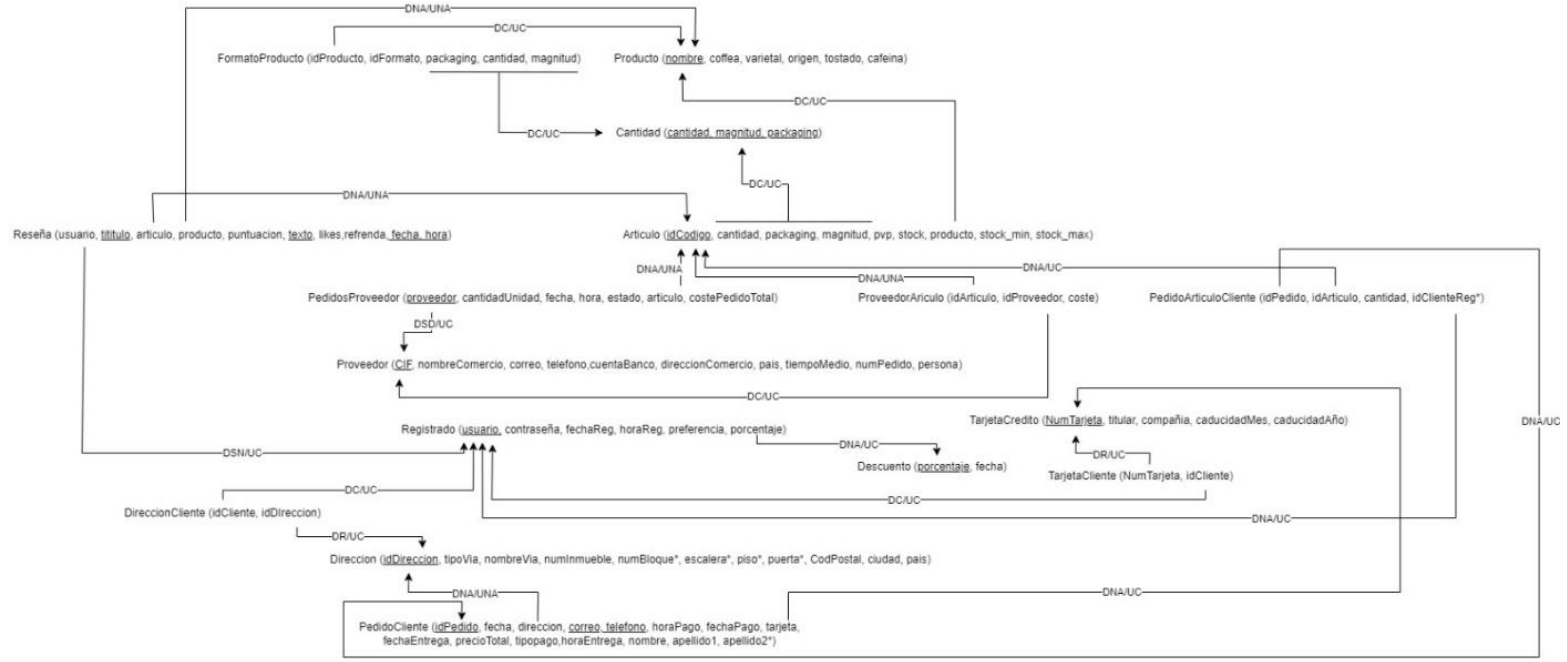
En el apartado del diseño relacional, se presenta el esquema relacional elaborado, definiendo la estructura de las tablas, las relaciones entre ellas y los atributos pertinentes. Además, se examinan aspectos relacionados con la semántica implícita y explícita, así como las posibles implicaciones no contempladas en el diseño original.

La implementación de la estática relacional en SQL constituye el segundo apartado, donde se detallan las sentencias SQL utilizadas para crear y gestionar la base de datos. Se exploran aspectos como la semántica explícita re-incorporada, semántica implícita y semántica excluida, con el objetivo de comprender cómo se materializan las decisiones de diseño en el lenguaje de manipulación de datos.

Por último, el apartado de carga de datos describe el proceso de carga de información desde las tablas desnormalizadas proporcionadas, haciendo uso del fichero de carga (NEWload.sql). Se abordan aspectos prácticos relacionados con la transformación y carga de datos en el contexto específico del comercio online.

2. Diseño Relacional

- Esquema relacional:



- Semántica implícita:

Sup_id	Mecanismo	Descripción
I_1	ID	Generamos ID porque nos facilita tener claves primarias en ciertas tablas.
I_2	FormatoProducto	Hacemos una tabla intermedia dada la relación N/N entre Cantidad y Producto. Cada producto puede tener distintos formatos mientras que un formato se puede reflejar en varios productos.
I_3	ProveedorAtrículo	Hacemos una tabla intermedia dada la relación N/N entre Proveedor y Artículo, Un proveedor puede suministrar varios artículos, y un artículo puede ser suministrado por varios proveedores.
I_4	DirecciónCliente	Hacemos una tabla intermedia dada la relación N/N entre Dirección y Registrado. Puede haber más de un cliente en cada dirección y un cliente puede hacer pedidos o estar relacionado con diferentes direcciones.
I_5	TarjetaCliente	Hacemos una tabla intermedia dada la relación N/N entre TarjetaCrédito y Registrado. Un cliente puede contar con distintas tarjetas bancarias y una tarjeta puede ser utilizada por varios clientes.
I_6	PedidoArticuloCliente	Hacemos una tabla intermedia dada la relación N/N entre Artículo y PedidoCliente. Un pedido puede estar formado por varios artículos mientras que un artículo puede aparecer en varios pedidos.

I_7	Relaciones	Se han añadido flechas entre la clave foránea y la clave primaria de diferentes tablas
I_8	Descripción de relaciones	Cada relación del apartado anterior esta descrita por un tipo de DELETE y UPDATE

Tabla 1: Semántica implícita

- Semántica explícita no contemplada en el diseño:

Sup_id	Descripción
S_1	No se ha implementado la actualización de las unidades por cada compra
S_2	No se ha implementado la reposición automática cuando la cantidad disponible es inferior al stock mínimo.
S_3	No se ha implementado que el stock máximo se al menos 10 unidades por encima del stock mínimo.
S_4	No se establece un proceso automático de formalización de pedidos de reposición a los proveedores, pero se contemplan las casuísticas de stocks por debajo de lo debido o que no sean inferiores a 0.
S_5	Cuando el precio de un artículo cambia no se puede actualizar de la manera correcta.
S_6	No se ha podido establecer un límite de un pedido por referencia y día, mientras esté en estado 'draft' o 'placed'.
S_7	No se puede establecer la reposición automática al agregar una nueva referencia para alcanzar el stock máximo del producto.
S_8	No se cubre el estado de gestión del pedido en caso de ausencia de proveedores disponibles.
S_9	No se puede garantizar la elección del proveedor de menor coste y tiempo de entrega más rápido en caso de empate.
S_{10}	No se han podido actualizar y eliminar los borradores de pedidos, con restricciones para pedidos en proceso y completados.
S_{11}	No se cambia nombre del proveedor eliminado por 'desconocido' en los pedidos relacionados, se queda en blanco en esa sección.
S_{12}	Cuando un cliente realiza un pedido en la misma fecha y a la misma dirección el pedido no se consolida como uno si no que se tratan como pedidos diferentes.
S_{13}	Si un cliente compra más cantidad del mismo producto en una misma fecha y dirección los productos no se añaden los unos a los otros, también se tratan como pedidos independientes.
S_{14}	Cuando el stock es insuficiente no se manda un mensaje de error que indique que hay que comprar más unidades.
S_{15}	No se ha podido implementar el dominio del campo de la tabla formatoProducto siendo este con 3 posibilidades.
S_{16}	No podemos convertir las compras de usuarios registrados que se dan de baja en compras de clientes no registrados.
S_{17}	No se puede asegurar la aplicación del descuento de bono solo válido por 30 días y basado en el histórico de compras, aunque las tablas de descuento estén creadas y bien conectadas.
S_{18}	Aunque se disponga la información sobre el número de teléfono del cliente no se establece una prioridad con respecto al correo electrónico a la hora de

	establecer contacto con él. Se supone que si hay número de teléfono hay que hacerlo por SMS.
S_{19}	Cuando un usuario registrado ha publicado una reseña y posteriormente se borra la cuenta desde la que se ha publicado el comentario no podemos mandar la reseña a un usuario anónimo o borrarla.
S_{20}	No se han podido representar restricciones de tipo numéricas o por palabras como tipoTostatado, tipoPago...
S_{21}	No se ha implementado que el stock máximo se al menos 10 unidades por encima del stock mínimo.

Tabla 2: Semántica explícita no contemplada

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

Semántica explícita re-incorporada:

Sup_id	Descripción de la solución
S_3	Restricción CHECK (*ck_articulo_stockMax*) a la tabla <*articulo*>. Hemos puesto que el stock máximo sea mayor que 10. El enunciado indica que el stock máximo debe ser 10 por encima del mínimo, pero hemos puesto 10 ya que cubrir 15 supondría un vacío.
S_{21}	Restricción CHECK (*ck_articulo_stockMin*) a la tabla <*articulo*>. Hemos puesto que el stock mínimo sea mayor que 5 por defecto.
S_{15}	Restricción CHECK (*ck_formato*) a la tabla <*formatoProducto*>. Hemos limitado los formatos a los indicados en el enunciado con una lista de palabras.
S_{20}	Restricción CHECK (*ck_articulo_stockMin*) a la tabla <*articulo*>. Hemos establecido que el stock mínimo sea mayor que 5.
S_{20}	Restricción CHECK (*ck_producto_tostado*) a la tabla <*producto*>. Hemos limitado los tipos de tostado a los nombrados en el enunciado con una lista de palabras.
S_{20}	Restricción CHECK (*ck_pedidosProveedor_estado*) a la tabla <*pedidosProveedor*>. Hemos limitado los tipos de estado de un pedido de proveedor a una lista de palabras.
S_{20}	Restricción CHECK (*ck_registrado_preferenciaContacto*) a la tabla <*registrado*>. Hemos limitado los tipos de preferencia de contacto a los indicados en el enunciado.
S_{20}	Restricción CHECK (*ck_pedidoCliente_tipoPago*) a la tabla <*pedidoCliente*>. Hemos limitado los tipos de pago a tarjeta de crédito, transferencia bancaria o COD.
S_{20}	Restricción CHECK (*ck_pedidoArticuloCliente_cantidad*) a la tabla <*pedidoArticuloCliente*>. Hemos impuesto que la cantidad de artículos pedidas por un cliente sea mayor que 0
S_{20}	Restricción CHECK (*ck_review_meGusta*) a la tabla <*review*>. Hemos limitado el número de me gustas de una review de un usuario entre 0 y mil millones
S_{20}	Restricción CHECK (*ck_review_puntuacion*) a la tabla <*review*>. La puntuación de una review ira de 1 a 5.

Tabla 3: Semántica explícita re-incorporada

Semántica implícita:

Sup_id	Mecanismo	Descripción
I_9	Tipo de atributo	Hemos establecido un tipo VARCHAR2, NUM, CHAR o DATE para representar los atributos de las diferentes tablas
I_{10}	Unión de atributos	Algunos atributos como Fecha y Hora se han unido en FechaHora como un atributo DATE en SQL.
I_{11}	UNIQUE key	Para los atributos que son únicos añadimos un CONSTRAINT que los califique como unique keys.
I_{12}	DELETE	Hemos implementado los tipos de delete en las diferentes claves foráneas.
I_{13}	Clave Foránea	Se han clasificado las claves foráneas con un CONSTRAINT y el delete anteriormente comentado
I_{14}	Not NULL	Para aquellos datos que no pueden ser nulos introducimos la condición Not NULL
I_{15}	PRIMARY key	Aquellas claves primarias en las tablas se han clasificado con un CONSTRAINT
I_{16}	DROP table	Para que no haya errores al crear las tablas se borran las que tengan su mismo nombre de SQL antes de crearlas

Tabla 1(cont.): Semántica implícita

Semántica excluida:

Sup_id	Descripción semántica	Motivo	Explícita/ Implícita
E_1	Las modificaciones en cascada definidas en el grafo relacional	Oracle no contempla esta regla de integridad, solamente el DELETE CASCADE	Implícita
E_2	Atributo hora en las tablas	Se ha resumido como FechaHora como un ya que Oracle no ha permitido añadir TIME	Implícita
E_3	Actualizaciones en la base de datos	Necesitamos un sistema para programar las funciones de actualización de los diferentes datos de la base. “UPDATE”.	Explícita
E_4	Funciones por implementar	Al igual que las actualizaciones, hay funciones como la reposición de productos que no podemos implementar ya que necesitarían un código de programación.	Explícita
E_5	Atributo BASE_PRICE no se utiliza	En la base de datos antigua hay un atributo llamado base_price que no encaja con ninguno de los	Explícita

		atributos de las tablas que tenemos en nuestra nueva base de datos por lo que es el único atributo de la anterior base de datos que no hemos implementado y no ha sido por fallo de los datos que contiene	

Tabla 4: Semántica excluida en la creación de tablas

4. Carga de datos (LMD)

Dentro de la tabla producto toda la migración de datos que se realizan vienen de la tabla obsoleta fsdb.catalogue, todos los datos de nuestra nueva tabla 'producto' quedan cubiertos por los antiguos datos.

En la tabla cantidad encontramos el primer gran problema ya que nosotros buscamos clasificar los datos que en la antigua base de datos vienen comprimidos en un solo atributo (packaging) en 3 atributos dentro de nuestra nueva tabla. El problema es que los datos dentro de 'packaging' están todos en una misma cadena de texto, pero separadas por espacios, siguen esta estructura: tipo de paquete, peso en número, magnitud del peso. Con el código en SQL conseguimos separar los tres datos dentro de la cadena porque vienen separados por un espacio. Nos servimos del uso de la función substr para poder lograr separar los datos y mandar cada uno a uno de nuestros nuevos atributos (packaging, cantidad, magnitud).

A partir de esta tabla, siempre que queramos servirnos de los datos que hacen referencia a la cantidad del producto tenemos que volver a incluir todo el código que hemos empleado para traer los datos de la base antigua y poder migrarlos a nuestras nuevas tablas.

Para nuestra tabla 'formatoProducto' usamos estos datos que se encontraban dentro de packaging y almacenamos los datos sobre el formato y el producto en unos nuevos atributos a los que llamamos como id, aunque estos no sean id creados por nosotros con secuencias como haremos más adelante.

En la tabla 'articulo' volvemos a necesitar los datos que hemos obtenido en cantidad, además de un idCodigo en el que se almacenan los datos del barcode de ese mismo artículo en la base de datos antigua, para ser más exactos en la tabla catalogue.

Algo importante que destacar es que hay datos que se repiten en varias de las tablas de la base de datos antigua como por ejemplo barcode al que acabamos de hacer referencia. Si hay que migrar esos datos a una de nuestras nuevas tablas no es necesario pasar uno a uno los datos de las tres tablas ya que en ese caso por ejemplo nos interesa que los datos de articulo solo los obtengamos de la tabla catalogue ya que contiene todos los datos que ya tienen las otras dos además de otros datos que no aparecen en las otras.

Después de articulo tratamos la tabla proveedor donde sí que podemos obtener todos sus datos de la antigua base de datos y cubrimos sus atributos, dentro de esta tabla nos encontramos con un nuevo atributo denominado persona al que no se hacía referencia en el enunciado pero que vemos que es un dato que existía en la antigua base de datos por lo que vamos a mantenerlo en la nueva porque puede ser una información útil más adelante para alguien que quiera consultar información sobre los responsables de diferentes pedidos.

Tenemos que destacar que entre la tabla proveedor y la siguiente en nuestra base de datos hay una tabla llamada 'pedidosProveedor', lo destacable de esta tabla es que no se le migra ningún dato de la base vieja ya que todos los datos acerca del proveedor ya están cubiertos en nuestras otras tablas que cubren a los proveedores. Esta tabla comienza vacía y se llenará según se actualizan los pedidos con la nueva base de datos.

Una vez hemos migrado de manera correcta todos los atributos de proveedor podemos ocuparnos de otra de nuestras tablas que está relacionada con esta que es 'proveedorArticulo', dentro de esta volvemos a mover los datos de barcode y prov_taxid a sus respectivos atributos en nuestras nuevas tablas y aparece coste que traemos de la antigua base de datos fsdb.catalogue donde se le hace referencia como cost_price.

Ahora migramos los datos sobre la tarjeta de crédito a nuestra nueva tabla tarjetaCredito, estos atributos han sido fáciles de clasificar en nuestra base porque estaban clasificados como card_ esto los hace fácilmente identificables y cuadran con los que teníamos creados en nuestra tabla.

A colación de la tabla de la tarjeta tratamos unos datos que han conllevado algunos problemas, que son los relacionados con los descuentos, el migrarlos ha sido sencillo porque de los dos atributos que tenemos en nuestra tabla (fecha y descuento) solo tenemos los descuentos en la base antigua por lo que podemos establecer el valor de fecha como null. Además, los datos que había dentro del atributo discount en la base antigua eran todos 0%, así que si sucediera algún problema a la hora de migrarlos tampoco serían datos claves en nuestra nueva base datos.

Después de migrar los datos en las anteriores tablas llegamos a la tabla dirección, el problema que encontramos al realizar la migración a esta tabla es que hay dos tipos de datos dliv_ y bill_ que corresponden a los mismos atributos en nuestra tabla dirección, para poder meter los datos de dos atributos diferentes en un solo dato en la nueva base necesitamos usar la función UNION, con ella metemos todos los datos de dliv_ primero, después realizamos el UNION y por último volvemos a migrar los datos pero esta vez de bill_ de la misma forma que lo hemos estado haciendo en las demás tablas. Realizando esta sucesión no saltan errores de violación de convenio.

Ahora podemos migrar los datos necesarios a una de las tablas más importantes de nuestra nueva base de datos que es 'registrado', aquí podemos cubrir todos los datos de manera normal menos preferenciaContacto que tampoco existía en la base antigua y que controlamos con un check en nuestra creación de tablas. Pero hay otro dato que nos trae un nuevo problema que es FechaHoraEntrega, siempre que aparece algún dato de fechahora en nuestra nueva base se va a tratar como un único atributo, pero en el diseño relacional separamos la fecha en dos. Para poder

migrar este tipo de datos de la forma DATE de manera correcta nos servimos de la función TO_DATE que acompañada de algo como esto (reg_date || ' ' || reg_time, 'YYYY / MM / DD HH:MI:SS AM') esto nos ayuda a almacenar las fechas que aparecen en la base antigua de la forma correcta en la nueva base.

Para la tabla direccionCliente podemos migrar los datos de manera sencilla como en tablas anteriores, pero se emplea INNER JOIN que junta datos de varias tablas diferentes para poder migrar todos los datos en un único insert y no obliga a usar distintos que después acarrean errores en SQL.

La migración de datos a 'tarjetaCliente' es parecida a la que realizamos en tarjetaCredito y en las demás tablas, los datos que necesitamos en esta nueva tabla aparecen todos en la base de datos que nos proporcionaba el enunciado.

Después de estas tablas procedemos a migrar los datos a pedidoCliente que es una de las tablas que más extensión tiene, aunque se vuelve fácil de entender si comprendes como hemos migrado los datos, ya que son iguales que otros que han aparecido anteriormente en otras tablas, para la mayoría de los atributos basta con pasarlos directamente de las otras tablas cuyos datos hemos unificado con INNER JOIN los datos restantes son fechas, como: fechaHoraPago o fechaHoraEntrega. Estos datos los hemos pasado con TO_DATE y la estructura que hemos enseñado previamente en la que podemos dividir los datos de la fecha en los que se llega a especificar hasta los segundos.

La última tabla relacionada directamente con los pedidos es 'pedidoArticuloCliente', en esta migración podemos obtener los datos necesarios de la base antigua, además volvemos a utilizar INNER JOIN y TO_DATE para cuadrar todos los datos que han sido necesarios en las otras tablas que tenían esta relación directa con el pedido .

Para facilitar todo este paso de datos que hemos realizado en estas últimas tablas y simplificar el código en si hemos usado abreviaciones que se indican junto al INNER JOIN, el motivo de usar estas abreviaturas es que si no habría que escribir fsdb.trolley por ejemplo, y lo sustituimos por troll. Como se puede apreciar las abreviaturas son fácilmente reconocibles y además ahorran un gran espacio a la hora de escribir el código y hacen más fácil su lectura y comprensión.

La última tabla que necesitamos migrar es review, en ella se contienen los datos del usuario que comenta además del producto y artículo al que se está escribiendo la reseña. De la antigua base de datos traemos toda la información que había en reviews de la antigua base de datos, su título, texto, likes, puntuación.... Para traer los datos sobre el producto y los artículos aquí también necesitamos usar el INNER JOIN y con esto tener unificados todos los datos que puede requerir la base a la hora de almacenar la información sobre el producto del que se está escribiendo la reseña.