|  |
| --- |
| PROYECTO FINAL  SQL |
|  |
| 2022  SERGIO EMANUEL PRELLEZO  Coderhouse |

**Índice**

[Introducción 3](#_Toc109340609)

[Objetivo 3](#_Toc109340610)

[Situación problemática 3](#_Toc109340611)

[Modelo de negocio 4](#_Toc109340612)

[Diagrama de entidad relación 5](#_Toc109340613)

[Objetos de la base de datos 7](#_Toc109340614)

[Tablas 7](#_Toc109340615)

[Listado 7](#_Toc109340616)

[Descripción 8](#_Toc109340617)

[Script de creación de tablas 13](#_Toc109340618)

[Esquemas relacionales - *Reverse Engineer* 13](#_Toc109340619)

[Inserción de datos 15](#_Toc109340620)

[Script de inserción de datos 20](#_Toc109340621)

[Vistas 20](#_Toc109340622)

[Descripción 20](#_Toc109340623)

[Script de creación de vistas 25](#_Toc109340624)

[Funciones 25](#_Toc109340625)

[Descripción 25](#_Toc109340626)

[Script de creación de funciones 26](#_Toc109340627)

[Procedimientos almacenados 26](#_Toc109340628)

[Descripción 27](#_Toc109340629)

[Script de creación de procedimientos almacenados 30](#_Toc109340630)

[Triggers 30](#_Toc109340631)

[Script de creación de triggers 32](#_Toc109340632)

[Usuarios 32](#_Toc109340633)

[Script de creación de usuarios y permisos 33](#_Toc109340634)

[Transacciones 33](#_Toc109340635)

# Introducción

Este informe constituye el proyecto final para el curso de SQL dictado por Coderhouse.

Recientemente acabo de terminar la carrera de “Desarrollo Full Stack” y para la misma realicé un proyecto de ecommerce, por lo que decidí tomar como referencia ese trabajo para la elección de mi situación problemática y modelo de negocio.

# Objetivo

Crear una base de datos, en la cual se implementará el modelo relacional para representar procesos basados en un modelo de negocio propio, con dataset ficticio y derivado de un proyecto anterior de ecommerce.

Se desarrollarán objetos e implementarán los procesos técnicos que requiere el mantenimiento de una base de datos.

También se implementarán consultas SQL que permitan la generación de informes de diferente índole y alcance.

# Situación problemática

Nos encontramos ante un comercio electrónico, más específicamente, una tienda de bicicletas y artículos de ciclismo llamada **Mammoth**.

Se plantea la necesidad de dar soporte a la app de la empresa generando la persistencia de la información en una base de datos SQL. En mi caso utilizaré a MySQL como sistema gestor de base de datos (**SGBD**).

Las principales necesidades del comercio son:

* guardar información de contacto de sus clientes
* guardar preferencias de los clientes tales como favoritos y suscripciones con el fin de poder conocer mejor a los usuarios y generar análisis que den origen a potenciales oportunidades de venta.
* almacenar los carritos de compra activos
* ofrecer su catálogo de productos
* capacidad de administrar los productos y controlar su stock
* guardar y llevar un control de los pedidos y sus respectivos estados
* guardar los detalles de facturación y posibilitar diversos análisis sobre la misma.

# Modelo de negocio

Como comenté anteriormente, estamos frente a una tienda online (**Mammoth**).

La plataforma permite visualizar el catálogo de productos junto con la posibilidad de ver en detalle cada uno de ellos, junto a su actual precio de venta y su disponibilidad de stock.

Los productos pertenecen a determinadas categorías del comercio (cada producto se asocia con una única categoría), tales como bicicletas, accesorios, indumentaria, etc. También se informa el país de procedencia de los mismos y quién es el proveedor en cada caso (cada producto tiene asociado un único proveedor), aunque esta última información es solo visible por los miembros de la empresa.

Para proceder a interactuar con la plataforma en cuanto a la selección de productos, selección de favoritos y suscripciones, es necesario que el usuario complete un registro brindando información de contacto y facturación. En un principio el comercio va a trabajar con clientes locales, por lo que la información precargada en las tablas de ciudades y provincias corresponden a la República Argentina.

El usuario puede elegir distintos productos y cantidades de los mismos para su carrito de compras, el cuál va a ser almacenado por la empresa durante un periodo de tiempo que se considere razonable. Carritos más antiguos que dicho período procederán a borrarse por cuestiones de optimización de almacenamiento del servidor.

Si se elige más de una unidad de un determinado producto, se agrupa dicha cantidad, dando origen a un detalle de carrito o pedido optimizado sin repeticiones de un mismo artículo.

Una vez que el usuario confirma un pedido, se procede a cargar su detalle en base al detalle del carrito que da origen a la orden de venta y posteriormente si todo ocurre correctamente, se procede a eliminar dicho carrito ya que carece de sentido mantenerlo persistente. De esta manera cada usuario puede tener un único carrito activo a la vez.

También previo a generar la orden de venta, se chequea la disponibilidad de stock y de ser correcta, el producto actualiza su stock.

Al realizar la compra, el usuario puede escoger entre distintos tipos de envío y también diferentes opciones de pago. Existe una relación biunívoca entre la orden de venta y la factura emitida, por lo que el detalle de la factura es exactamente el mismo que la orden de venta relacionada y por ende se toma de ella sin redundar información.

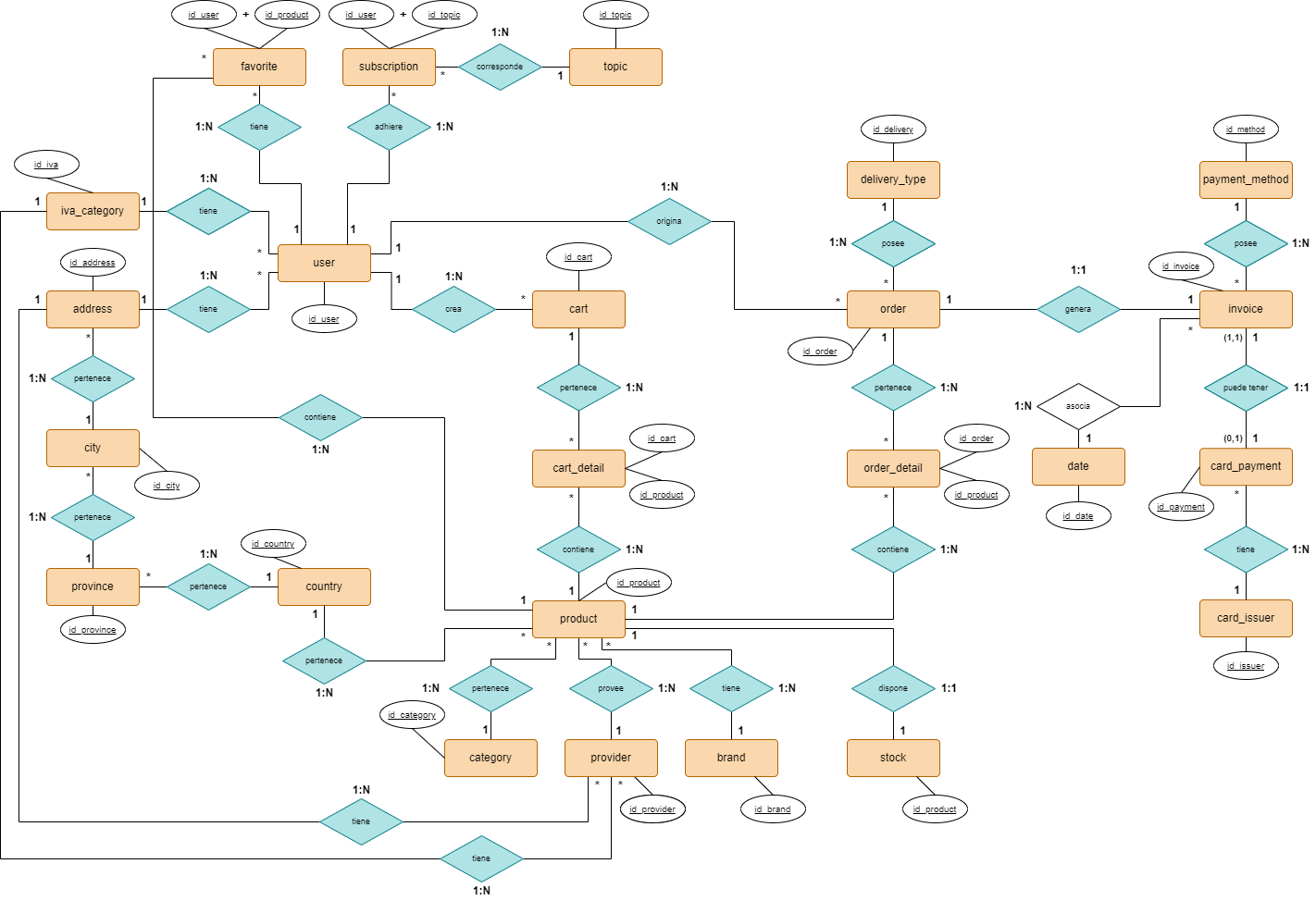
Finalmente, un usuario registrado puede tener múltiples productos favoritos y también se puede suscribir a múltiples temáticas de interés.

# Diagrama de entidad relación

En la siguiente hoja se muestra el diagrama de entidad relación (**DER**) correspondiente al modelo de base de datos desarrollado **mammoth**.

El diagrama posee la siguiente terminología estándar:

* Rectángulos: representan conjuntos de entidades.
* Elipses: representan atributos. En general se colocan solamente los nombres subrayados de los campos que son llaves primarias (**PK**).
* Rombos: representan acciones que permiten relaciones entre tablas.
* Líneas: unen atributos a conjuntos de entidades y conjuntos de entidades a conjuntos de relaciones. Representan la unión entre acciones y entidades
* Cardinalidad: Representada en cada extremo de la relación y también sobre cada acción.



# Objetos de la base de datos

A lo largo de esta documentación se irán describiendo los distintos objetos de la base de datos **mammoth** junto a sus scripts correspondientes. **Para facilitar recrear la BD**, se adjunta el script global mammoth-DB.sql (dentro de la carpeta “scripts”), que reúne todas las sentencias en un único archivo.

# Tablas

A continuación, se encuentra el listado de tablas incluido en la base de datos **mammoth** y luego la descripción de cada una de ellas

## Listado

|  |  |
| --- | --- |
| 1. *user* 2. *subscription* 3. *topic* 4. *favorite* 5. *iva\_category* 6. *address* 7. *city* 8. *province* 9. *country* 10. *cart* 11. *cart\_detail* 12. *producto* 13. *category* | 1. *brand* 2. *provider* 3. *stock* 4. *order* 5. *order\_detail* 6. *delivery\_type* 7. *invoice* 8. *payment\_method* 9. *card\_payment* 10. *card\_issuer* 11. *date* 12. *table\_manipulation\_log* 13. *product\_price\_update* |

## Descripción

















































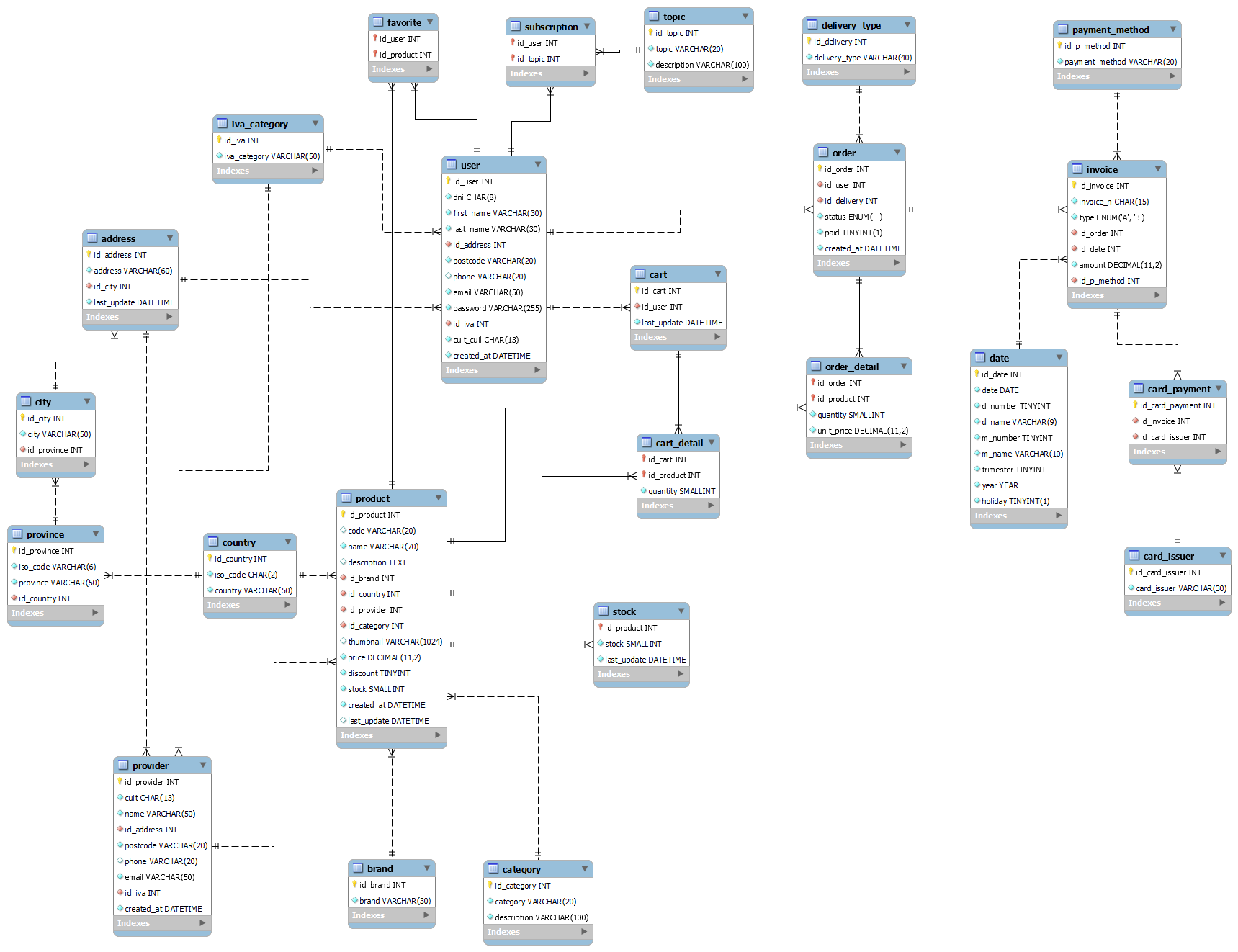
## Script de creación de tablas

Se adjunta a este documento el script de creación de tablas en el archivo de extensión sql llamado mammoth-CREATE.sql, dentro de la carpeta “scripts/single-scripts”.

# Esquemas relacionales - *Reverse Engineer*

Se adjunta a este documento el archivo *diagram-mammoth.mwb* (dentro de la carpeta “diagrams”) con el diagrama EER obtenido aplicando ingeniería inversa en el cliente SGDB **MySQL Workbench**.

A continuación, se puede observar una visualización general.



# Inserción de datos

Se realizará la carga de registros en cada tabla mediante el procedimineto de importación de datos.

Cada tabla se corresponde con un archivo .csv del mismo nombre alojado dentro de la ruta “data/csv\_imports” que contiene la información y será el que se utilice para la inserción de los datos.

A fin de ilustrar el proceso de poblar cada una de las tablas de la base de datos **mammoth,** utilizaré el asistente de **MySQL Workbench.**

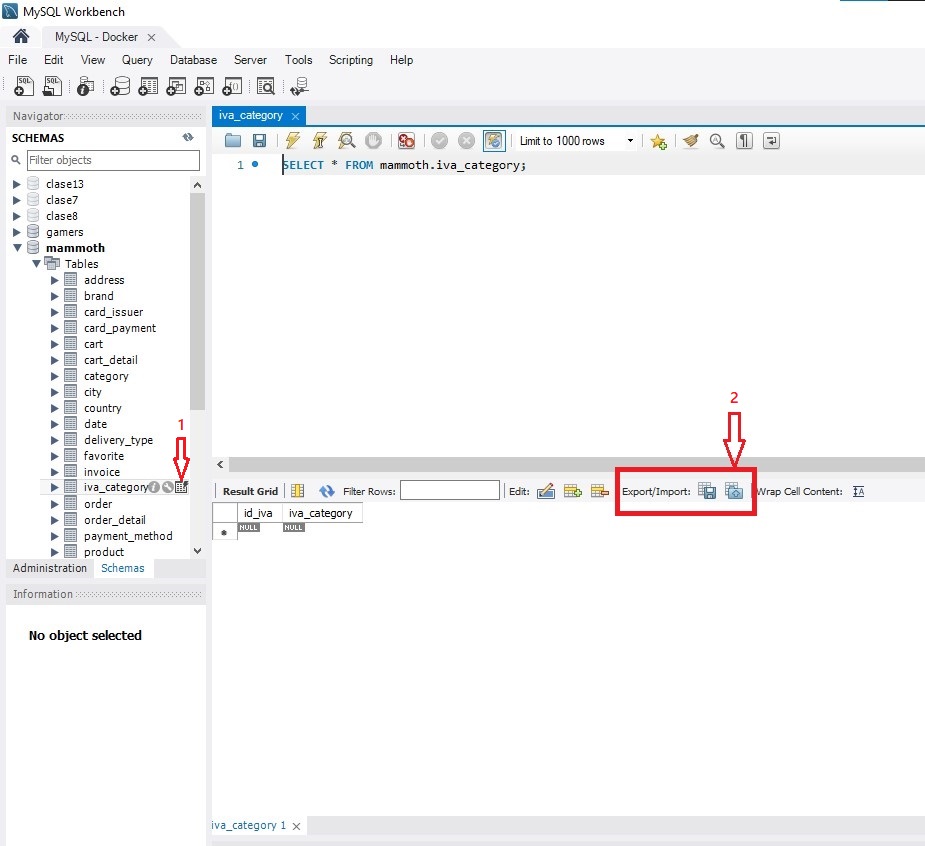
Es muy importante respetar un orden en este proceso ya que, debido a las relaciones e integridad referencial de nuestra BD, primero deben insertarse los datos en las tablas padres de nuestro árbol de relaciones, para luego proseguir por las tablas hijas, que hacen referencia a las primeras.

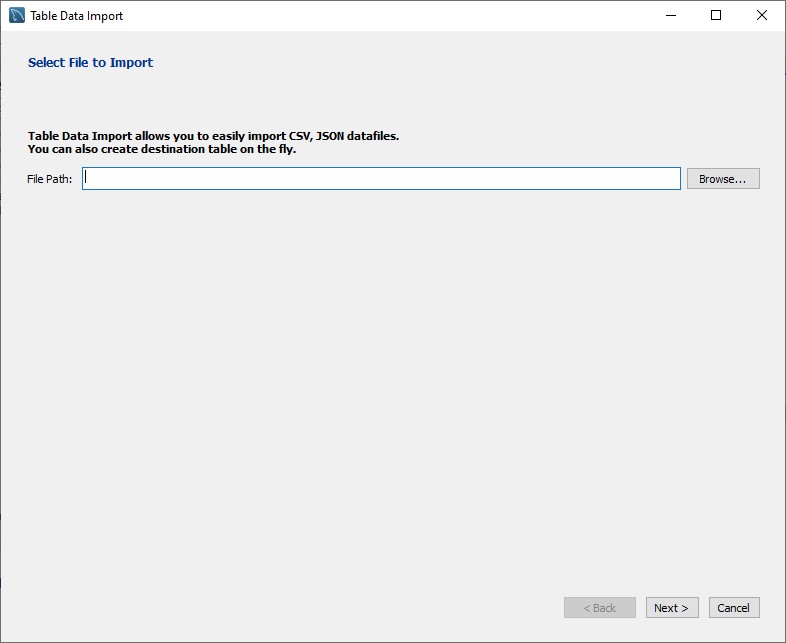
Un posible orden a seguir sería:

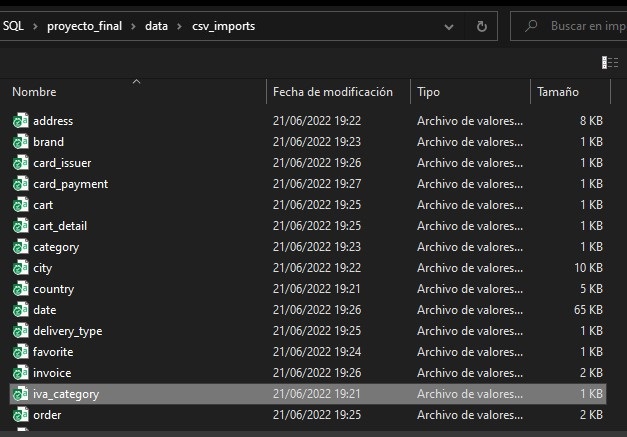
|  |  |
| --- | --- |
| 1. iva\_category 2. country 3. province 4. city 5. address 6. user 7. category 8. brand 9. provider 10. product 11. stock 12. favorite | 1. topic 2. subscription 3. cart 4. cart\_detail 5. delivery\_type 6. order 7. order\_detail 8. payment\_method 9. date 10. invoice 11. card\_issuer 12. card\_payment |

A continuación, se muestran los pasos a seguir para llevar a cabo la primera importación (tabla iva\_category). Para el resto de las tablas, se repite el mismo procedimiento.

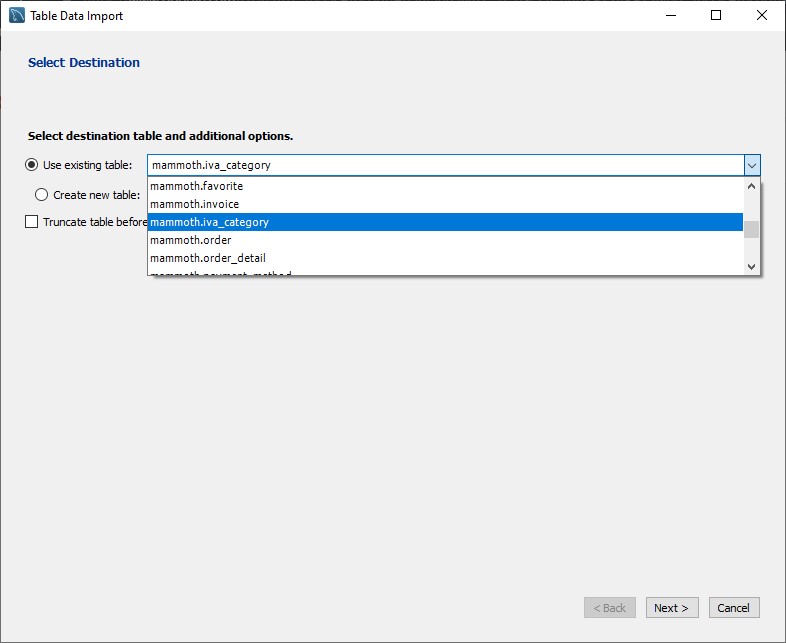
* Primero seleccionamos el contenido de la tabla (aún vacío) desde el ícono respectivo que aparece al lado del nombre de dicha tabla (1).
* En los resultados, sobre la barra de herramientas anexa, nos aparece una zona **Export/Import** y hacemos click sobre el ícono de importar (2).

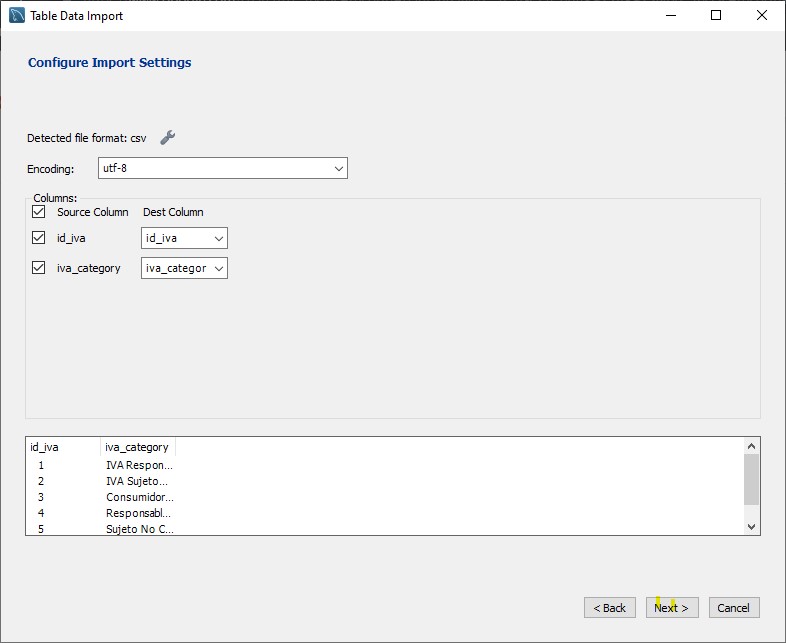


* Este ícono abre la ventana de importación de datos. En esta ventana tenemos la posibilidad de buscar y seleccionar el archivo desde nuestra computadora con el botón Browse, o escribir la ruta del mismo.

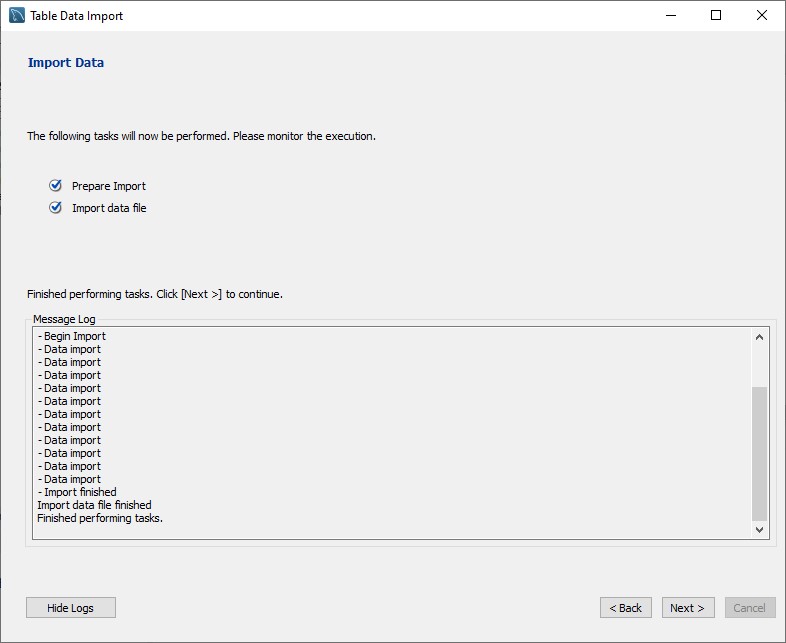


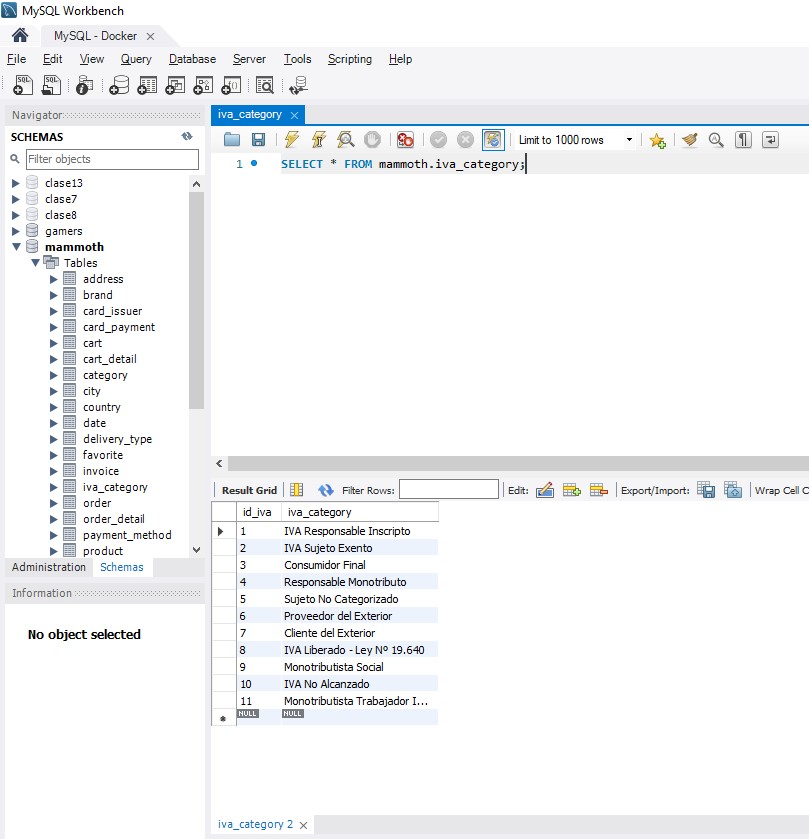
* Una vez seleccionado el archivo se nos abre el siguiente menú en donde escogemos la primera de las opciones y seleccionamos la tabla de destino correspondiente en donde se van a insertar los datos (iva\_category en este caso)



* El Asistente detectará el formato del archivo y hará un match estimado del contenido, el cual podremos ajustar:
  + UTF-8 es el formato de mayor compatibilidad de caracteres.
  + Verificamos que coincidan las columnas Source / Dest.
  + Validamos columnas / datos, en la vista previa inferior.
* Finalmente, validamos todo el proceso realizado por el asistente de importación, observando los logs y verificando que no se produzcan errores.

Si todo se realizó correctamente, continuamos y veremos una ventana de que se ha finalizado la importación de registros, con el total de datos que se agregaron en la tabla elegida.



* Refrescamos la consulta de los datos de la tabla para validar la correcta importación de la información.

## Script de inserción de datos

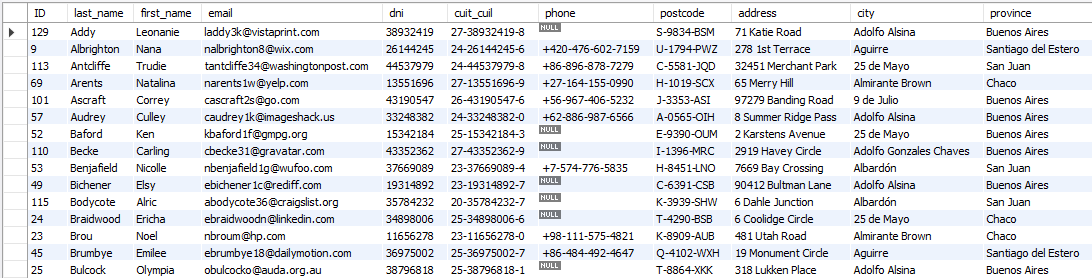
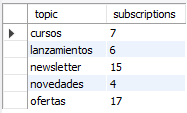
***A modo de simplificar la población de la BD, se adjunta un script de inserción general de los registros.*** El archivo se llama mammoth-INSERT.sql y se encuentra dentro de la carpeta “scripts/single-scripts”. Igualmente, como se mencionó más arriba, dichas sentencias de inserción se encuentran en el script global mammoth-DB.sql, al igual que el resto de las sentencias que dan origen a los distintos objetos de nuestra BD.

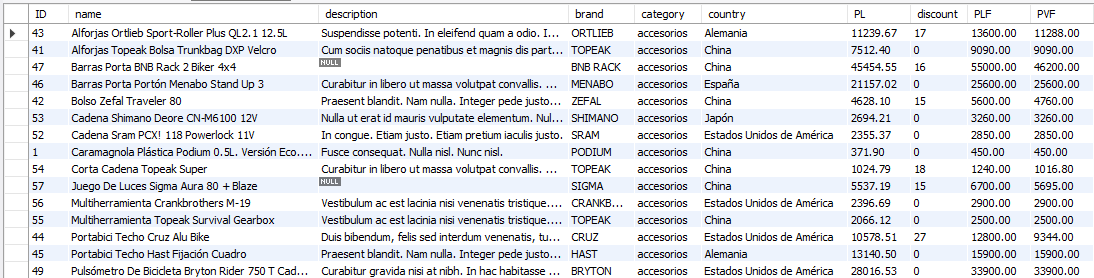
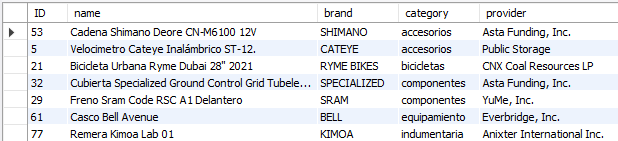
# Vistas

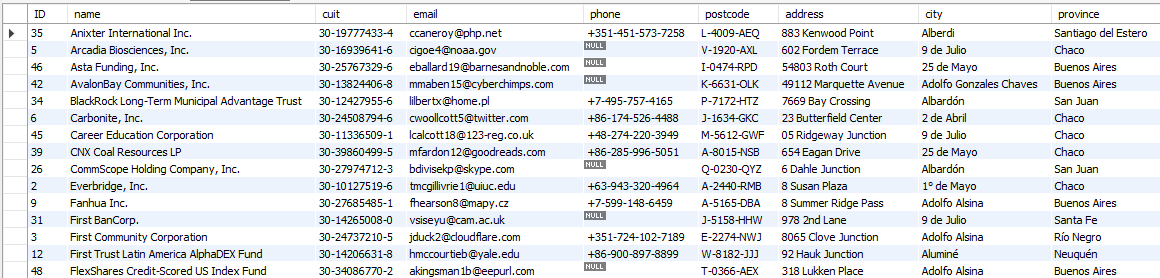
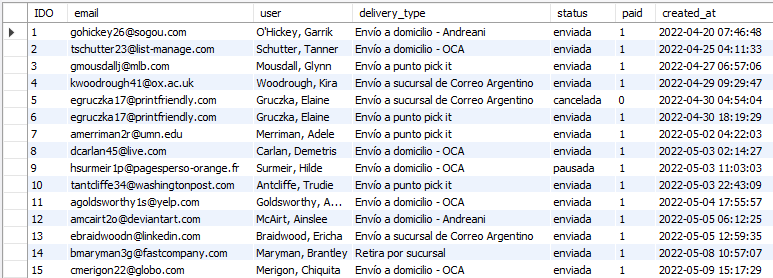
Se crearon las siguientes vistas útiles para la consulta segura de los datos de nuestra base de datos. Aparte de mejorar el rendimiento al evitar tener que ejecutar queries complejas para obtener la información.

|  |  |
| --- | --- |
| * user\_list | *Listado de usuarios* |
| * favorite\_rank | *Ranking de productos favoritos* |
| * subscription\_by\_topic | *Cantidad de suscripciones por tema* |
| * product\_list | *Listado de productos* |
| * product\_without\_stock | *Listado de productos sin stock* |
| * provider\_list | *Listado de proveedores* |
| * order\_list | *Listado de pedidos* |
| * order\_to\_prepare | *Listado de pedidos para preparar* |
| * sales\_by\_day\_of\_week | *Ventas por día de la semana* |
| * sales\_by\_product\_category | *Ventas por categoría de productos* |

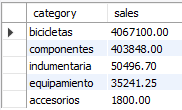
## Descripción











## Script de creación de vistas

El archivo que contiene la creación de vistas se llama mammoth-VIEWS.sql y se encuentra dentro de la carpeta “scripts/single-scripts”.

# Funciones

Se crearon las siguientes funciones almacenadas a fin de obtener ciertos resultados personalizados que serán útiles tanto a la hora de consultar como insertar datos en la base de datos.

|  |  |
| --- | --- |
| * get\_subtotal | *Obtener el subtotal de una orden* |
| * get\_order\_amount * invoice\_type | *Obtener el importe total de una orden*  *Devuelve el tipo de factura* |
| * next\_invoice\_n | *Devuelve el número de factura siguiente* |

## Descripción



## Script de creación de funciones

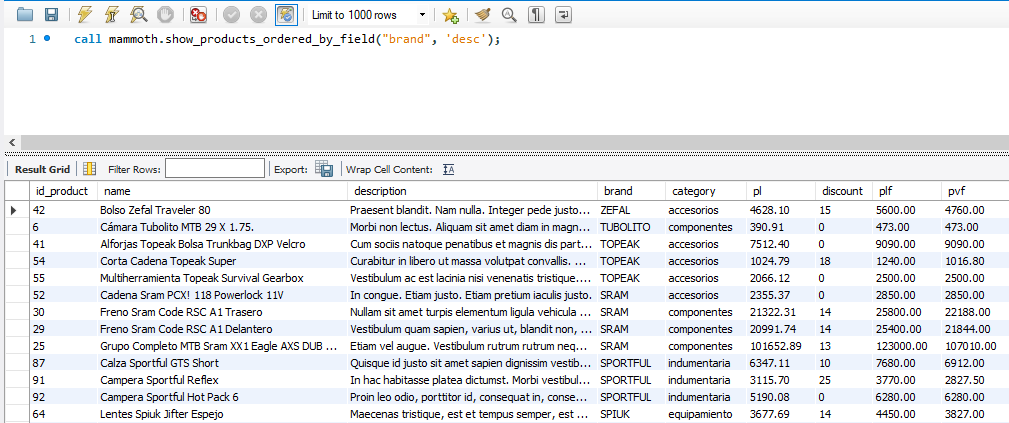
El archivo que contiene la creación de funciones se llama mammoth-FUNCTIONS.sql y se encuentra dentro de la carpeta “scripts/single-scripts”.

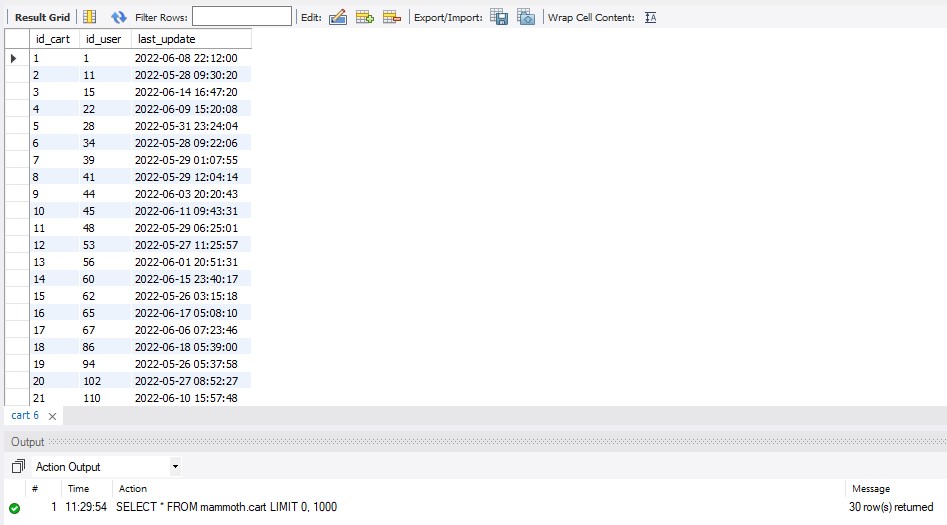
# Procedimientos almacenados

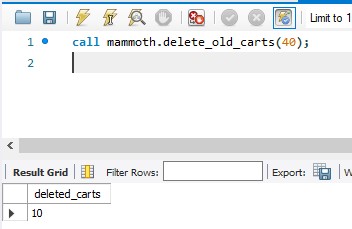
Se crearon los siguientes procedimientos almacenadas para obtener ciertas funcionalidades personalizadas que serán útiles en la operación de la base de datos.

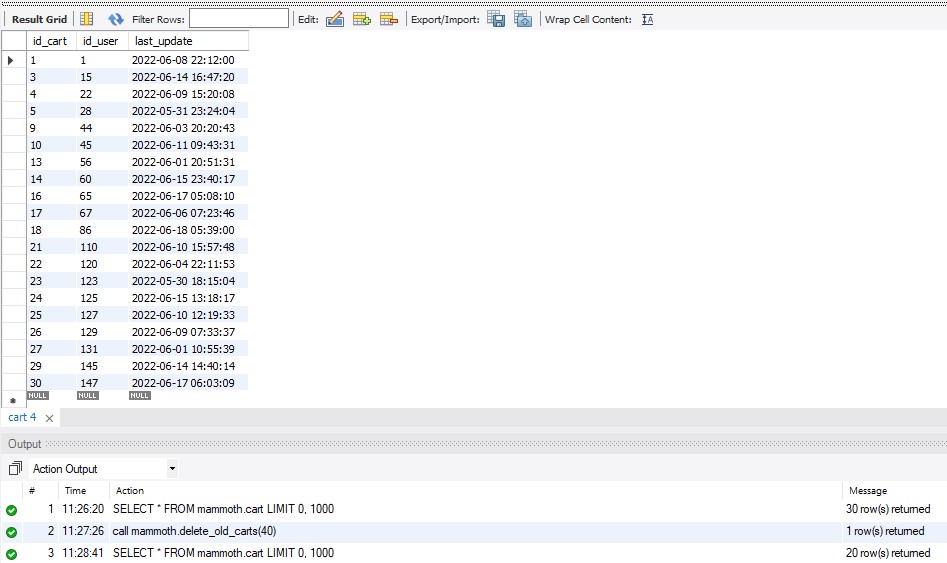
|  |  |
| --- | --- |
| * show\_products\_ordered\_by\_field | *Obtener el listado de productos ordenados por un campo específico* |
| * delete\_old\_carts * generate\_order\_from\_cart | *Borra los carritos antiguos*  *Genera una orden de pedido y detalle de la misma a partir del carrito que le da origen* |

## Descripción

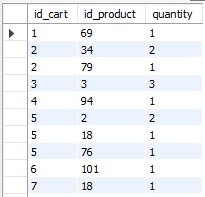
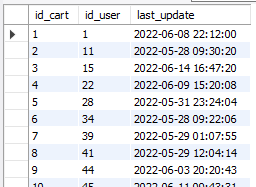


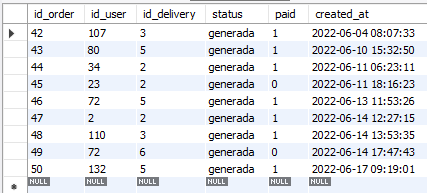
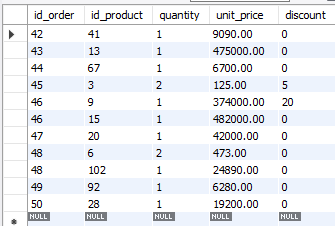
Tabla cart, antes de llamar al SP

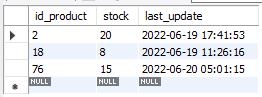
Llamando al SP

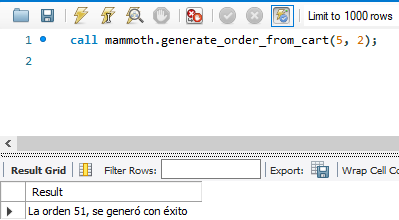
Tabla cart, después de llamar al SP

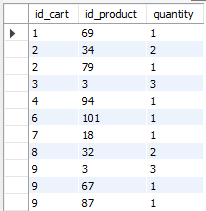
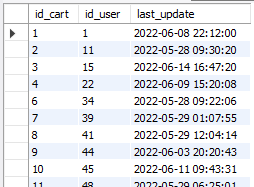


Tabla cart y cart\_detail antes de llamar al SP

Tabla order y order\_detail antes de llamar al SP

Tabla stock antes de llamar al SP

Llamando al SP

Tabla cart y cart\_detail después de llamar al SP

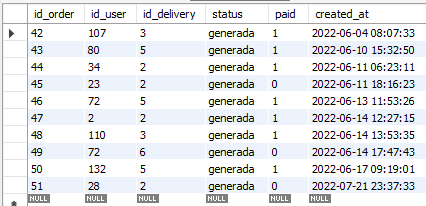
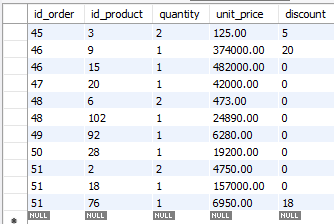
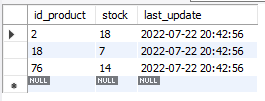
Tabla order y order\_detail después de llamar SP

Tabla stock después de llamar al SP

****

## Script de creación de procedimientos almacenados

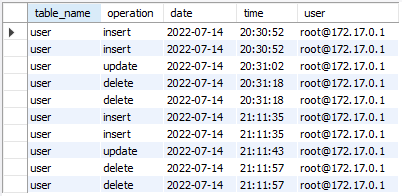
El archivo que contiene la creación de procedimientos almacenados (stored procedures) se llama mammoth-SP.sql y se encuentra dentro de la carpeta “scripts/single-scripts”.

# Triggers

Se crearon los siguientes triggers para guardar dentro de una tabla de logs (table\_manipulation\_log) la información de operaciones DML sobre ciertas tablas, como también los movimientos de precios de los productos dentro de otra tabla de auditoría (product\_price\_update).

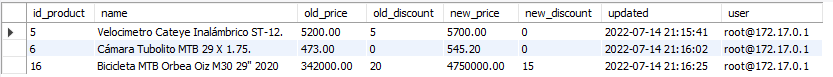
|  |  |
| --- | --- |
| * AF\_IN\_user\_table\_manipulation\_log |  |
| * AF\_UP\_user\_table\_manipulation\_log * AF\_DE\_user\_table\_manipulation\_log * BF\_UP\_product\_product\_price\_update |  |

Descripción

Ejemplo tabla table\_manipulation\_log



Ejemplo tabla product\_price\_update



## Script de creación de triggers

El archivo que contiene la creación de triggers se llama mammoth-TRIGGERS.sql y se encuentra dentro de la carpeta “scripts/single-scripts”.

# Usuarios

A fines prácticos, se crearon dos usuarios para la base de datos **mammoth.**

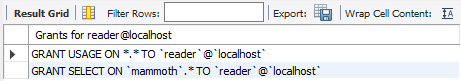
Uno de los usuarios: “reader”, tiene permisos de sólo lectura sobre todas las tablas y vistas. Su contraseña será “Reader2022”.

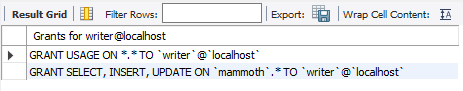
El otro usuario: “writer”, tiene permisos de lectura, inserción y modificación de datos sobre los mismos objetos anteriores. Su contraseña será “Writer2022”

Ninguno de ellos podrá eliminar registros de ninguna tabla.

A continuación, se muestran las consultas sobre los permisos que nos devuelve MySQL, luego de realizar los correspondientes pasos para la creación de los usuarios y otorgación de los respectivos permisos.

usuario reader



usuario writer

## Script de creación de usuarios y permisos

El archivo que contiene la creación de usuarios y otorgación de permisos se llama mammoth-USERS.sql y se encuentra dentro de la carpeta “scripts/single-scripts”.

# Transacciones

A modo de prueba del uso de transacciones, se adjuntan unas sentencias de ejemplos en el archivo mammoth-TRANSACTIONS.sql que se encuentra dentro de la carpeta “scripts/single-scripts”.

Por otro lado, como se adelantó en el apartado de procedimientos almacenados, se creo un SP con transacciones (generate\_order\_from\_cart), en donde se registra un manejador de errores que básicamente invoca un ROLLBACK ante la aparición de una excepción o warning, deshaciendo las operaciones dentro de la transacción.

En caso de que todo ocurra de manera correcta, la transacción finaliza con un COMMIT de todas las operaciones de inserción y eliminación de datos que se llevaron a cabo de manera atómica, manteniendo la integridad de la infomación.