****

**Curso SQL: Proyecto Final**

*Datos para todos*

****

**Profesor:** Juan Ignacio Garicoche

**Tutor:** German Rodriguez

**Alumno:** Alexander Billiet

**Comisión:** 31265

**Fecha límite de entrega:** 01/08/2022

INDICE

[Consigna: 7](#_Toc109244311)

[Introducción a la temática, Objetivo, Situación problemática y Modelo de Negocio. 8](#_Toc109244312)

[Diagrama Entidad – Relación 10](#_Toc109244313)

[Relaciones Entre Entidades 11](#_Toc109244314)

[Relación CLIENTE – PEDIDO 11](#_Toc109244315)

[Relación CLIENTE – TARJETA 11](#_Toc109244316)

[Relación PEDIDO – PRODUCTO 11](#_Toc109244317)

[Relación PEDIDO – DESCUENTO 12](#_Toc109244318)

[Relación PEDIDO – MEDIO\_DE\_PAGO 12](#_Toc109244319)

[Relación PRODUCTO – PROVEEDOR 13](#_Toc109244320)

[Listado de Tablas 14](#_Toc109244321)

[Tabla Cliente 14](#_Toc109244322)

[Tabla Tarjeta 14](#_Toc109244323)

[Tabla METODO\_DE\_PAGO 15](#_Toc109244324)

[Tabla Pedido 16](#_Toc109244325)

[Tabla Descuento 16](#_Toc109244326)

[Tabla puente Descuento – Pedido 17](#_Toc109244327)

[Tabla Producto 17](#_Toc109244328)

[Tabla puente Pedido – Producto 18](#_Toc109244329)

[Tabla Proveedor 19](#_Toc109244330)

[Inserción de Datos 20](#_Toc109244331)

[Consideraciones 20](#_Toc109244332)

[Importación de registros con “Table Data Import Wizard” 21](#_Toc109244333)

[Vistas 30](#_Toc109244334)

[Vista # 1 – Top 5 Libros más vendidos 30](#_Toc109244335)

[Vista # 2 – Top 5 Autores más vendidos 31](#_Toc109244336)

[Vista # 3 – Facturación y Ganancia por Autor 32](#_Toc109244337)

[Vista # 4 – Descuento Menor Aplicado 33](#_Toc109244338)

[Vista # 5 – ocurrencias\_descuentos 34](#_Toc109244339)

[Funciones 36](#_Toc109244340)

[Función #1 – GANANCIA\_PERIODO 36](#_Toc109244341)

[Función #2 – CONTAR\_TARJETAS 38](#_Toc109244342)

[Stored Procedures 39](#_Toc109244343)

[Stored Procedure #1 – ordenamiento\_producto 39](#_Toc109244344)

[Stored Procedure #2 – nuevo\_proveedor 41](#_Toc109244345)

[Stored Procedure # 3 – nuevo\_pedido 42](#_Toc109244346)

[Triggers 46](#_Toc109244347)

[Aclaración importante: 46](#_Toc109244348)

[Trigger # 1 - log\_insercion\_proveedor: 46](#_Toc109244349)

[Trigger # 2 – chequeo\_vacios\_proveedor: 47](#_Toc109244350)

[Trigger # 3 – trigger\_insercion\_producto: 48](#_Toc109244351)

[Trigger # 4 – chequeo\_producto\_prohibido: 50](#_Toc109244352)

[Implementación de sentencias 52](#_Toc109244353)

[Creación de los usuarios: 52](#_Toc109244354)

[GRANT de permisos a los usuarios: 52](#_Toc109244355)

[Check del funcionamiento de los permisos en la BBDD: 53](#_Toc109244356)

[Sentencias del sub-lenguaje TCL 57](#_Toc109244357)

[DELETE registros tabla TARJETA con TCL 57](#_Toc109244358)

[INSERT registros tabla DESCUENTO con TCL 58](#_Toc109244359)

[Backup de la base de datos 62](#_Toc109244360)

[Backup 62](#_Toc109244361)

[Tecnologías y Bibliografía 64](#_Toc109244362)

[Tecnologías: 64](#_Toc109244363)

[Bibliografía: 64](#_Toc109244364)

# Consigna:

Graphical user interface, text, application, email

Description automatically generated

# Introducción a la temática, Objetivo, Situación problemática y Modelo de Negocio.

La temática elegida para desarrollar la base de datos es simular una compañía de **e-commerce** dedicada a la compra de libros a editoriales / empresas y su posterior reventa, a través de su página web, a consumidores finales u a otras empresas revendedoras.

El listado de productos que esta compañía se dedica a comercializar (los libros) se sacó de la base de datos del top 50 de libros más vendidos por Amazon desde el 2009 al 2019: “Amazon Top 50 Bestselling Books 2009 – 2019”[[1]](#footnote-1)

Esta base de datos cuenta aproximadamente con 500 registros donde hay información sobre el libro, autor, rating de usuarios, cantidad de reviews, precio, año, género, etc.



A partir de esta información, se revisaron ejemplos de base de datos de modelos de negocios de comercio electrónico como para tener una idea de cuáles deberían ser las entidades más importantes a la hora de diagramar una base de datos propia de este estilo[[2]](#footnote-2).

En base a estos ejemplos, se llegó a la conclusión de que, para poder capturar correctamente el flujo de negocios de una firma dedicada al comercio electrónico, era necesario formular entidades del estilo de: clientes o usuarios, pedidos, proveedores, métodos de pago, descuentos, etc.

Algunas otras entidades, como inventarios o métodos de envío, fueron concienzudamente dejadas de lado para evitar sobre complejizar el proceso de aprendizaje del curso con cuestiones que no hacían directamente al contenido de MySQL.

Para la realización de estas tablas, en algunos casos, se utilizó la página Mockaroo para crear, por ejemplo, números de DNI, contraseñas, direcciones de email, nombres de proveedores, etc. En otros casos se utilizó directamente MS Excel, tanto para crear información aleatoria como para generar relaciones coherentes entre los registros de las diferentes tablas (ejemplo: un CLIENTE que realiza un PEDIDO y decide utilizar el MÉTODO DE PAGO tarjeta, debe de tener una tarjeta registrada a su nombre en la tabla TARJETA). Otros tantos registros fueron simplemente inventados de manera discrecional pero siempre guardando una relación lógica con el resto de la información (ejemplo: los DESCUENTOs por cantidad se aplican sólo a aquellos PEDIDOs que superan los 10 items).

Habiendo definido las tablas más importantes que dan forma al modelo de negocios, el flujo de información que se intenta capturar con la base de datos es relativamente intuitivo y se advierte incluso en el Diagrama de Entidad Relación del apartado siguiente:

Diagram

Description automatically generated

CLIENTEs de nuestra compañía se registran en nuestra página web, desde donde se captura información pertinente como: si son empresas o consumidores finales, qué tipo de identificación única poseen (CUIT o DNI), contraseña para loggearse y operar en nuestra interfaz y un mail como medio de contacto.

Esos CLIENTEs realizan PEDIDOs que a su vez contienen PRODUCTOs que, lógicamente, son comprados por nuestra firma a un listado de PROVEEDORes.

Sobre esta capa fundamental se agregan otras dimensiones como, por ejemplo: tarjetas de crédito y débito que nuestros CLIENTEs hayan dado de alta en nuestra página, MÉTODOs de PAGO con los que nuestros CLIENTEs pagarán los PEDIDOs y diversos DESCUENTOs que nuestra compañía ofrece (a veces en función a variables internas como, por ejemplo, descuentos por cantidad u otra veces descuentos sobre fechas especiales como el día del padre o madre, día del libro, etc.).

Con esto se terminó de conformar una base de datos que captura correctamente un modelo simplificado de negocios.

# Diagrama Entidad – Relación



# Relaciones Entre Entidades

## Relación CLIENTE – PEDIDO

Cada **CLIENTE** puede realizar muchos **PEDIDOS** (ejemplo: Cliente 1 realiza Pedido 1, Pedido 2, Pedido 3).

Cada **PEDIDO** es realizado por unsolo **CLIENTE** (ejemplo: el Pedido 1 es realizado únicamente por Cliente 1).

A screenshot of a computer

Description automatically generated with low confidence

## Relación CLIENTE – TARJETA

Cada **CLIENTE** posee una o más **TARJETAS.** Cada **TARJETA** es poseída por un solo **CLIENTE.**

A screenshot of a phone

Description automatically generated with low confidence

## Relación PEDIDO – PRODUCTO

Cada **PEDIDO** contiene uno o muchos **PRODUCTOS** (en determinadas cantidades).

Cada **PRODUCTO** puede estar contenido en muchos **PEDIDOS** diferentes. (Es decir: El libro 1 puede estar en el pedido 1, en el pedido 2, etc.).

Al ser relación muchos a muchos, esta relación genera una tabla puente **PEDIDO\_PRODUCTO** donde cada registro es una cantidad específica de un producto dentro de un pedido específico.

Shape

Description automatically generated

## Relación PEDIDO – DESCUENTO

A cada **PEDIDO** se le aplica uno o muchos **DESCUENTOS** (un pedido puede tener más de 1 descuento: un descuento por cantidad, un descuento por el medio de pago, etc.)

Cada **DESCUENTO** puede ser aplicado a muchos **PEDIDOS** (ejemplo: el descuento por cantidad puede aplicar al pedido 1, al pedido 2, etc.).

Al ser relación muchos a muchos, esta relación genera una tabla puente **PEDIDO\_DESCUENTO** donde cada registro es un descuento aplicado a un pedido. Esto se debe a que un pedido puede tener muchos descuentos y, además, el mismo descuento puede aparecer en muchos pedidos.

A screenshot of a computer

Description automatically generated with low confidence

## Relación PEDIDO – MEDIO\_DE\_PAGO

Cada **PEDIDO** se pagacon un único **MEDIO\_DE\_PAGO,** pero un **MEDIO\_DE\_PAGO** se puede utilizar en muchos pedidos (ejemplo: Pedido 1, Pedido 2 y Pedido 3 son todos pagados con efectivo pero Pedido 4 no puede ser pagado mitad en efectivo y mitad con tarjeta de débito).

Diagram

Description automatically generated

## Relación PRODUCTO – PROVEEDOR

Cada **PRODUCTO** es vendido / proveído únicamente por un **PROVEEDOR**. (Libro 1 es solo vendido por proveedor 1. No puede existir otro proveedor diferente del 1 que nos venda también el libro 1).

Cada **PROVEEDOR** vende muchos **PRODUCTO** (es decir: Proveedor 1 puede vender Libro 1, Libro 2, Libro 3, etc.)

A screenshot of a computer

Description automatically generated with low confidence

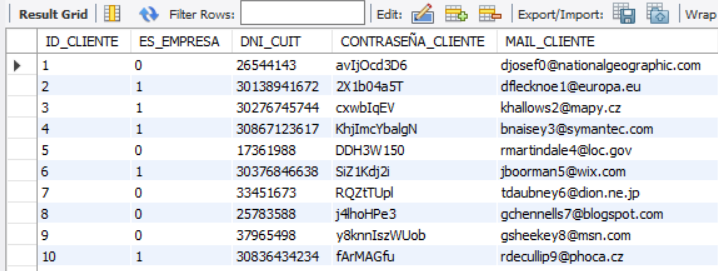
# Listado de Tablas

## Tabla Cliente

Son los clientes / usuarios que se registran en la página web de nuestra compañía para comprar libros. Pueden ser consumidores finales o empresas revendedoras. Es requisito excluyente estar registrado en nuestra página para poder realizar un pedido.



Así se ve la tabla en MySQL Workbench:



## Tabla Tarjeta

Las diferentes tarjetas de crédito y débito que han registrado nuestros clientes al darse de alta en la página web de nuestra compañía. Administrar esta información puede servir para campañas de venta, lanzar promociones, entender el comportamiento de nuestros clientes, etc.



Así se ve la tabla en MySQL Workbench:

Graphical user interface, table

Description automatically generated

## Tabla METODO\_DE\_PAGO

Los diferentes métodos de pago que acepta nuestra compañía.



Así se ve la tabla en MySQL Workbench:

Graphical user interface, application, table

Description automatically generated with medium confidence

## Tabla Pedido

Todos los pedidos que van generando los clientes a través de la página web. Cada registro de pedido tiene, además de la identificación, una fecha y hora, el cliente que realiza el pedido y el método de pago elegido para concretar la transacción.



Así se ve la tabla en MySQL Workbench:

A screenshot of a computer

Description automatically generated

## Tabla Descuento

Listado de los diferentes descuentos que la compañía ofrece a sus clientes con sus respectivos nombres y descripciones.



Así se ve la tabla en MySQL Workbench:

Graphical user interface, text, application, email

Description automatically generated

## Tabla puente Descuento – Pedido

Dado que la relación entre DESCUENTO y PEDIDO es muchos a muchos, se genera una tabla puente donde cada línea es un descuento específico aplicado a un producto. Un pedido puede tener más de un descuento aplicado (ejemplo: un descuento por cantidad y además un descuento por haberse realizado el pedido en el Día Del Libro).



Así se ve la tabla en MySQL Workbench:

Table

Description automatically generated

## Tabla Producto

Es el listado de los productos que comercializa nuestra compañía. En este caso, los libros. Tanto el nombre del libro, como el autor, y el precio de compra, se sacó directamente de la base de datos de Kaggle (mencionado más arriba). Para calcular el precio de venta, se le sumó un margen de ganancia del 50%. Además, se asignó cada uno de estos producto a un proveedor.



Así se ve la tabla en MySQL Workbench:

Graphical user interface

Description automatically generated

## Tabla puente Pedido – Producto

Dado que la relación entre PEDIDO y PRODUCTO es muchos a muchos, se genera una tabla puente donde cada línea es un producto (en determinada cantidad) contenido dentro de un pedido. Para entender mejor esta tabla, se puede mirar el screen de abajo donde se ve que el pedido 4 tiene 7 items del libro 32 y además 6 items del libro 98.



Así se ve la tabla en MySQL Workbench:

Graphical user interface, table

Description automatically generated

## Tabla Proveedor

Listado de proveedores que le venden los productos (libros) a nuestra compañía.



Graphical user interface, text, application

Description automatically generated

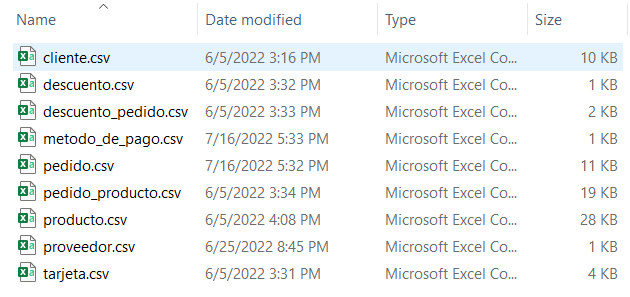
# Inserción de Datos

## Consideraciones

En este documento se adjuntan imágenes de cómo se insertaron los registros en la base de datos utilizando el “**Table Data Import Wizard**” de MySQL Workbench para una de las tablas.

Además, en la carpeta que se comparte, se adjuntan pertinentemente los archivos planos de CSV desde donde se importaron los registros.

Para la realización de este trabajo se importaron todos los archivos planos que se ven a continuación. Todos se logran importar correctamente.



Vale la aclaración: Dada la relación entre claves primarias y foráneas, el orden de importación debería ser el siguiente:

Text

Description automatically generated with low confidence

## Importación de registros con “Table Data Import Wizard”

Puede resultar obvio, pero, antes de comenzar con la importación de registros es necesario correr el script para crear la base de datos y las tablas correspondientes:

Graphical user interface, text, application

Description automatically generated

Se observa que la creación de tablas fue satisfactoria:

Graphical user interface, text, application

Description automatically generated with medium confidence

Se comienza a importar los registros utilizando el “**Table Data Import Wizard**” de MySQL Workbench. En el siguiente ejemplo, se demuestra cómo se importan los registros de la tabla “CLIENTE”.

Click derecho sobre la tabla CLIENTE dentro del esquema bbdd\_e\_commerce y se selecciona “Table Data import Wizard”.

Graphical user interface, application

Description automatically generated

Se busca el archivo plano en formato CSV utf-8 guardado localmente en mi computadora y se le da a “next”:

Graphical user interface, text, application, email

Description automatically generated

Se deja seleccionado el destino que ya viene por default. Es decir, usar la tabla CLIENTE en la base de datos bbdd\_e\_commerce y se le da a “next”:

Graphical user interface, text, application, email

Description automatically generated

Se comprueba que las columnas de la tabla del script sean las mismas que las columnas del archivo CSV y se le da “next”:

Graphical user interface

Description automatically generated with medium confidence

Antes de proceder a la importación, le damos a “show logs” como para ver el detalle y asegurarnos la integridad de la importación. Luego se le da a “next”:

Graphical user interface, text, application, email

Description automatically generated

Se leen lo mensajes de Log para asegurarnos que la importación haya ido bien. Se ve que todos los registros se han importado sin inconvenientes. Se le da a “next”:

Graphical user interface, text, application

Description automatically generated

Se comprueba que la cantidad de registros agregados a la base de datos coincida con la cantidad de registros en el archivo plano. En este caso no se registran inconvenientes. Se le da a “Finish”:

Graphical user interface, text, application

Description automatically generated

Se hace una última comprobación en el script para chequear que efectivamente los datos se hayan cargado bien:

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application

Description automatically generated

# Vistas

## Vista # 1 – Top 5 Libros más vendidos

Esta vista ofrece los 5 artículos (libros) más vendidos, así como sus respectivos autores y cantidad de ítems vendidos.

Está compuesta por las tablas PEDIDO\_PRODUCTO y PRODUCTO.

La intuición de esta vista es poder tener una consulta para entender cuáles son los libros más populares en ventas y tomar decisiones de negocios, campañas de publicidad, etc.

Graphical user interface, text, application

Description automatically generated

Esto retorna la vista cuando se la llama con los datos cargados:

Graphical user interface, text, application

Description automatically generated

## Vista # 2 – Top 5 Autores más vendidos

Esta vista ofrece los 5 autores de libros más vendidos. Es interesante porque un autor puede tener más de un libro a la venta. Por lo tanto, esta vista trae información diferente de la anterior.

Está compuesta por las tablas PEDIDO\_PRODUCTO y PRODUCTO.

La intuición de esta vista es poder tener una consulta para entender cuáles son los autores más populares en ventas y tomar decisiones de negocios, campañas de publicidad, etc.

Graphical user interface, text, application

Description automatically generated

Esto retorna la vista cuando se la llama con los datos cargados:

Graphical user interface, text, table

Description automatically generated with medium confidence

## Vista # 3 – Facturación y Ganancia por Autor

Esta vista trae la facturación bruta por cada autor, el costo de la mercadería vendida y el beneficio bruto y su respectivo %.

Además, se decide mostrar solo los aquellos autores cuya nombre contenga la letra “J” para utilizar el operador LIKE.

Está compuesta por las tablas PEDIDO\_PRODUCTO y PRODUCTO.

La intuición de esta vista es poder tener una consulta para entender cuales son los autores más populares en ventas y más rentables y tomar decisiones de negocio en consecuencia.

Graphical user interface, text, application

Description automatically generated

Esto retorna la vista cuando se la llama con los datos cargados[[3]](#footnote-3):

Table

Description automatically generated

## Vista # 4 – Descuento Menor Aplicado

Esta vista trae un join entre dos tablas para saber cuáles son los pedidos recibieron el menor de los descuentos posibles que actualmente ofrece la compañía.

La gracia de esta vista es poder filtrar la información con un WHEREsin conocer el ID ni el nombre del menor de los descuentos de la compañía a través de una subconsulta.

Está compuesta por las tablas DESCUENTO\_PEDIDO, PEDIDO y hace la subconsulta hacia la tabla DESCUENTO.

Graphical user interface, text, application, email

Description automatically generated

Esto retorna la vista cuando se la llama con los datos cargados:

Graphical user interface, table

Description automatically generated

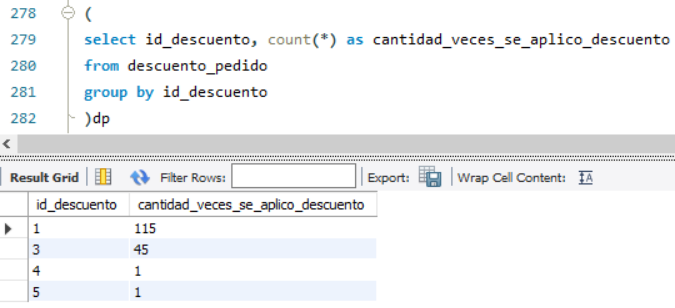
## Vista # 5 – ocurrencias\_descuentos

Esta vista cuenta las ocurrencias de los descuentos (tabla descuento) en los pedidos (tabla descuento\_pedido).

Para lograr esto, se armó un LEFT JOIN desde la tabla DESCUENTO vs la tabla DESCUENTO\_PEDIDO antes agrupada por cada ID\_DESCUENTO y una función que cuenta las ocurrencias de cada ID\_DESCUENTO.

Esto para que la vista pueda arrojar también aquellos descuentos que no fueron aplicados a ningún pedido.

Como para que se entienda, la ejecución del paréntesis “dp” devuelve lo siguiente:



Por lo que el LEFT JOIN finalmente se ejecuta entre la tabla DESCUENTO y la agrupación bajo el alias “dp”.

Está compuesta por las tablas DESCUENTO y DESCUENTO\_PEDIDO.

La intuición de esta VISTA es poder comprender qué tanto se están aplicando los DESCUENTOS que ofrece nuestra compañía y en base a esto evaluar su eficacia.

Graphical user interface, text, application, email

Description automatically generated

Esto retorna la vista cuando se la llama con los datos cargados:

Graphical user interface, application

Description automatically generated

# Funciones

## Función #1 – GANANCIA\_PERIODO

La idea de esta función es que reciba dos parámetros de entrada (fecha inicial y fecha final) y devuelva la ganancia por ventas de ese período.

Lo que hace es calcular las ventas y a eso substraerle los costos de esa mercadería vendida para mostrar, justamente, la ganancia del período que se pasa como parámetro.

Es importante mencionar que las fechas que se pasen como parámetro deben ser “válidas”.

Graphical user interface, text, application, email

Description automatically generated

Esta función requiere juntar tres tablas con 2 joins dado que:

* Precio de venta y compra de cada producto se encuentra en la tabla PRODUCTO.
* Cantidad de cada producto vendido se encuentra en la tabla PEDIDO\_PRODUCTO.
* Fecha de cada pedido se encuentra en la tabla PEDIDO.

Los joins entre las tablas toman esta forma:



O bien en el script se ve de esta manera:

Graphical user interface, text, application

Description automatically generated

Como se muestra arriba, se calculan las ganancias por venta (prod.precio\_venta\_libro \* pp.cantidad) y a eso se le substrae el costo de la mercadería vendida (prod.costo\_compra\_libro \* pp.cantidad).

Se filtran las fechas ingresadas como parámetro con un WHERE y BETWEEN y el SELECT se agrupa por las fechas.

Para llamar a la función se le pasa como parámetro la fecha inicial y la final como se muestra abajo:

Graphical user interface

Description automatically generated with medium confidence

Lo cual devuelve:

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

## Función #2 – CONTAR\_TARJETAS

La función contar\_tarjetas recibe un parámetro de entrada “param\_tipo\_tarjeta” que puede ser “credito” o “debito” y devolverá la cantidad de tarjetas que se encuentren registradas en nuestra base de datos.

Lo que hace la función es contar la cantidad de filas de la tabla TARJETA con un count(\*) filtrando con un WHERE por la columna “debito\_credito” según se haya ingresado como parámetro “debito” o “crédito”.

Graphical user interface, text, application, email

Description automatically generated

Como se ve abajo, hay 56 tarjetas de crédito registradas y 44 de débito:

Graphical user interface, text

Description automatically generated

Graphical user interface, application

Description automatically generated

Graphical user interface, application

Description automatically generated

# Stored Procedures

## Stored Procedure #1 – ordenamiento\_producto

Ordena la tabla “producto” por la columna que se quiera utilizar para ordenar. Tiene dos parámetros de entrada:

* “campo\_a\_ordenar” es la columna por la cual se quiere ordenar la tabla “producto”
* “orden” es un booleano que acepta 0 para ordenar de forma ascendente y 1 de forma descendente

Graphical user interface, text, application, email

Description automatically generated

Se llama al stored procedure sobre la columna “autor\_libro”:

Text

Description automatically generated with medium confidence

Y esto es lo que devuelve respectivamente:

Tabla “producto” ordenada de forma ascendente por la columna “autor\_libro”

Graphical user interface

Description automatically generated with medium confidence

Tabla “producto” ordenada de forma descendente por la columna “autor\_libro”:

Graphical user interface, website

Description automatically generated

## Stored Procedure #2 – nuevo\_proveedor

Stored Procedure que inserta un nuevo proveedor en la tabla PROVEEDOR. Admite 2 parámetros de entrada:

* "param\_nombre\_proveedor" que será el nombre del proveedor a agregar
* "param\_tipo\_proveedor" que será el tipo de proveedor (SRL, SA, SAS, etc).

Se le agrega un IF para asegurarse que el nombre del proveedor no exista ya en la tabla PROVEEDOR. En caso de que ya exista, el SP no agrega el registro y avisa mediante un mensaje.

Text

Description automatically generated

Se llama al SP de la siguiente manera:

Text

Description automatically generated with medium confidence

Y obtenemos que se han agregado los primeros 3 registros mientras que el último no se agrega y se levanta un mensaje avisando que ya existe un proveedor con ese nombre en la tabla:

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

## Stored Procedure # 3 – nuevo\_pedido

Stored Procedure que realiza un INSERT de un nuevo pedido. Al ingresar un nuevo pedido se deben impactar al menos dos tablas (PEDIDO y PEDIDO\_PRODUCTO) y quizás una tercera (DESCUENTO\_PEDIDO) en el caso de que le aplique un descuento al PEDIDO. Ejemplo: Si el pedido tiene más de 10 unidades, entonces se le aplica el descuento # 1 que es el descuento por cantidad.

Admite 5 parámetros de entrada:

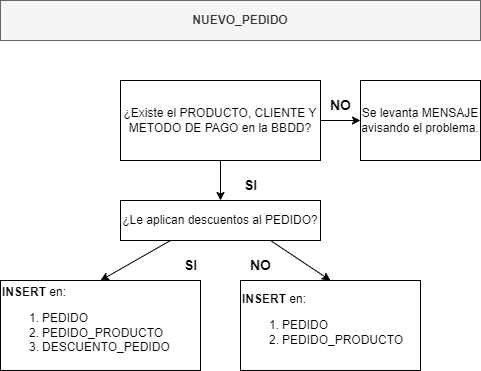
* "**id\_libro\_param**" PRODUCTO que se está comprando en el pedido.
* "**cant\_prod\_param**" Cantidad del PRODUCTO que se está comprando.
* "**fecha\_pedido\_param**" Fecha en que se realiza el pedido.
* "**id\_cliente\_param**" ID del CLIENTE que realiza el pedido.
* "**metodo\_de\_pago\_param**" MÉTODO\_DE\_PAGO con el que se realiza el pedido.

En caso de que el ID\_PRODUCTO, ID\_CLIENTE o METODO\_DE\_PAGO no exista en la base de datos el SP arrojará un mensaje indicando el problema.

Además, en caso de que la fecha del PEDIDO sea una fecha especial, se aplicará automáticamente el descuento correspondiente en la base de datos. Ejemplo: Si la fecha es el 23 de abril, se aplicará en la tabla DESCUENTO\_PEDIDO el descuento por compra en el "Día del Libro".

En caso de que el PEDIDO supere los 10 ITEMS, directamente se aplicará el descuento # 1 (Descuento por cantidad).

A grandes rasgos, este es el flujo que sigue la información en el Stored Procedure:



En esta primer parte del SP, primero se hace el check con un IF de que los parámetros ingresados (ID\_LIBRO\_PARAM, ID\_CLIENTE\_PARAM y METODO\_DE\_PAGO\_PARAM) existan previamente en la base de datos.

En caso de que no existan, se levanta un mensaje de advertencia y se suspende la inserción de registros. En caso de que sí existan, se prosigue con el script:

A picture containing diagram

Description automatically generated

Luego, se chequea si la FECHA\_PEDIDO\_PARAM coincide con la fecha de alguno de los descuentos y se guarda esa información en la variable @ID\_DESCUENTO\_FECHA. Si no aplica ningún descuento por fecha entonces @ID\_DESCUENTO\_FECHA se hace igual a False:

Text

Description automatically generated with medium confidence

El último de los chequeos es para entender si aplica el descuento por cantidad: si CANT\_PROD\_PARAM resulta mayor a 10 ítems, entonces aplica el descuento por cantidad y se guarda en @ID\_DESCUENTO\_CANTIDAD el “1” (ID del descuento por cantidad). Si, por el contrario, CANT\_PROD\_PARAM resulta menor a 10 ítems, entonces no aplica el descuento por cantidad y @ID\_DESCUENTO\_CANTIDAD se guarda como False.

A picture containing text

Description automatically generated

Luego se procede al INSERT del PEDIDO. Se abre una transacción y el primer INSERT se da en la tabla PEDIDO y el segundo en la tabla PEDIDO\_PRODUCTO:

Text

Description automatically generated

Finalmente, en caso de que haya descuentos, se insertan estos registros en la tabla DESCUENTO\_PEDIDO, de la siguiente forma:

Graphical user interface, text, application

Description automatically generated

Probamos el SP para comprobar que funcione correctamente.

Vemos que, previo a usar el SP, la tabla PEDIDO solo llega hasta el ID\_PEDIDO = 500:

Graphical user interface, text, application

Description automatically generated

Procedemos a llamar al SP pasándole como parámetros:

* id\_libro\_param = 1
* cant\_prod\_param = 11
* fecha\_pedido\_param = '2022-10-16'
* id\_cliente\_param = 1
* metodo\_de\_pago = 1



Chequeamos las tablas para ver si el procedimiento corrió correctamente.

Vemos que aparece el registro en la tabla PEDIDO con ID\_PEDIDO = 511 con los parámetros que le pasamos más arriba:

Graphical user interface, table

Description automatically generated

Lo mismo ocurre en la tabla PEDIDO\_PRODUCTO donde están los 11 ítems del ID\_LIBRO = 1 reflejados:

Graphical user interface, application

Description automatically generated

Finalmente, se observa que, dado que en este pedido aplica el descuento por cantidad (ID\_DTO = 1) y el descuento por el día de la madre (ID\_DTO = 5), se registran ambos descuentos en la tabla DESCUENTO\_PEDIDO:

Graphical user interface, application

Description automatically generated

# Triggers

## Aclaración importante:

Hay dos triggers que sirven para impedir el INSERT de registros que no cumplen con determinada condición. Para esto se levanta una **signal sqlstate '45000'** como se vió en clase. La ejecución de esta señal detiene el script de SQL por lo que en el script la prueba de estos triggers está comentada para que no cause problemas.

## Trigger # 1 - log\_insercion\_proveedor:

Es un trigger que luego de que se inserte un registro nuevo en la tabla “proveedor” loggea información en la tabla “log\_insercion\_proveedor”. La información que se almacena en esta última tabla es el “id\_proveedor”, “nombre\_proveedor”, “tipo\_proveedor”, el “usuario” que agregó el registro, la fecha y la hora.

Como se ve abajo, primero se crea la tabla “log\_insercion\_proveedor” y luego el trigger “log\_insercion\_proveedor”.

Este TRIGGER ES “AFTER” un evento de “INSERT” en la tabla “proveedor” y básicamente captura los datos ya mencionados y los guarda como un nuevo registro en la tabla “log\_insercion\_proveedor”.

Graphical user interface, text

Description automatically generated with medium confidence

Para probar si el trigger “log\_insercion\_proveedor” funciona correctamente se insertan 3 proveedores a la tabla “proveedor”:

Graphical user interface, text, application

Description automatically generated

Después se hace un select \* from “log\_insercion\_proveedor” para chequear que el trigger haya funcionado bien y agregado los registros a la tabla correctamente:



Graphical user interface, application

Description automatically generated

## Trigger # 2 – chequeo\_vacios\_proveedor:

Este trigger lo que hace es validar ANTES del INSERT de un registro a la tabla “proveedor” que el registro que se está intentando insertar cumpla con determinada condición.

En este caso, si el nombre del proveedor que se intenta INSERTAR está vacío (new.nombre\_proveedor = ‘’) entonces se levanta una señal que interrumpe el INSERT y manda un mensaje al usuario explicando el porqué de la interrupción del INSERT.

Graphical user interface, text, application, email

Description automatically generated

Cuando se intenta insertar un nombre de proveedor vacío como se ve abajo, Workbench devuelve le siguiente mensaje:

A picture containing table

Description automatically generated



## Trigger # 3 – trigger\_insercion\_producto:

Este trigger básicamente almacena datos de los INSERTs que se hayan realizado a la tabla PRODUCTO en la tabla “tabla\_log\_insercion\_producto” con algunas particularidades.

Además de almacenar información sobre el “usuario” que realiza el INSERT, “la fecha”, “la hora”, el “id\_libro” y el “nombre\_libro” también almacena la cantidad de libros en el catálogo luego del INSERT y el nuevo costo promedio.

Primero se crea la tabla tabla\_log\_insercion\_producto donde se almacenarán los datos anteriormente nombrados:

Graphical user interface, text, application, chat or text message

Description automatically generated

Y luego se crea el TRIGGER:

Graphical user interface, text, application, email

Description automatically generated

Como se ve, se crean dos variables @cantidad\_catalogo\_producto y @promedio\_costo\_libros que luego se insertarán como registros en la tabla “tabla\_log\_insercion\_producto”.

La primera @cantidad\_catalogo\_producto lo que hace es un count(\*) de la tabla PRODUCTO. La segunda @promedio\_costo\_libros calcula el promedio de la columna costo\_compra\_libro de la tabla PRODUCTO.

Dado que el TRIGGER se ejecuta DESPUÉS del INSERT, la @cantidad\_catalogo\_producto y @promedio\_costo\_libros refieren a los valores LUEGO de que se haya realizado el INSERT que dispara el TRIGGER.

Veamos cómo se ejecuta:

Primero, se busca la cantidad de productos diferentes en la tabla PRODUCTO y el promedio de costo\_compra\_libro, para capturar estos valores ANTES de INSERTAR el nuevo registro:



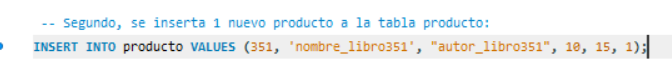
Lo cual arroja una cantidad de libros de 350 cuyo valor de compra promedio es $13.02:

Graphical user interface, website

Description automatically generatedGraphical user interface, text, application, chat or text message, website

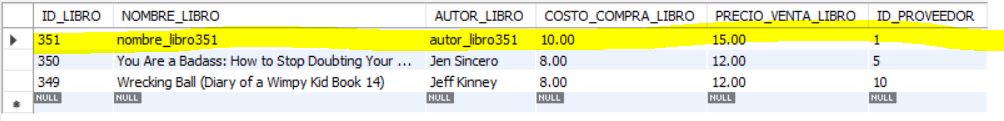
Description automatically generated

Luego se procede a insertar un nuevo registro en la tabla PRODUCTO:



Y se chequea que se haya insertado correctamente:





Ahora sí, chequeamos que el TRIGGER haya funcionado correctamente:

Text

Description automatically generated

Al llamar a la “tabla\_log\_insercion\_producto” obtenemos:



Lo que demuestra que se han almacenado todos los campos luego del INSERT. Se observa además cómo la cantidad de libros ha subido a 351 y el promedio ha cambiado a $13.01 ya que éstos dos cálculos están considerando el INSERT en cuestión.

Luego se elimina el registro ingresado a la tabla PRODUCTO solo para mantener la prolijidad de la base de datos:

Graphical user interface, text, application

Description automatically generated

## Trigger # 4 – chequeo\_producto\_prohibido:

Este último TRIGGER es muy similar al #2. El TRIGGER es llamado ante un evento de INSERT en la tabla PRODUCTO y al ser de tipo BEFORE lo que hace es prevenir el INSERT en caso de que el autor\_libro del registro sea igual a VOLDEMORT (por política de la compañía se prohíbe la incorporación de libros escritos por VOLDEMORT al catálogo).

De igual manera, el TRIGGER levanta un mensaje por consola para alertar al usuario de por qué no puede llevarse adelante el INSERT:

Graphical user interface, text, application, email, website

Description automatically generated

Cuando se lo prueba, se observa que al intentar insertar un registro a la tabla PRODUCTO con el campo autor\_libro = VOLDEMORT el INSERT se interrumpe:

Text

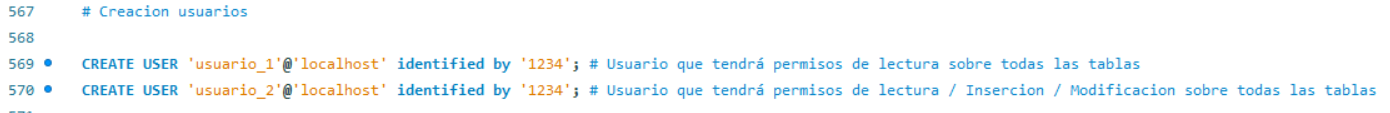
Description automatically generated with medium confidence



# Implementación de sentencias

## Creación de los usuarios:

Se crean los dos usuarios (**usuario\_1** y **usuario\_2**), ambos con el dominio “localhost” por ser dominio local y con una contraseña igual a 1234:



Se chequea que se hayan creado los usuarios correctamente:

Graphical user interface, text, application

Description automatically generated

Se ve que no tienen aún permisos otorgados:

Graphical user interface

Description automatically generated with medium confidence

Graphical user interface, text, application

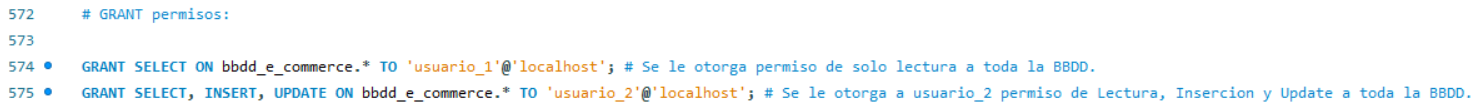
Description automatically generated

A picture containing graphical user interface

Description automatically generated

## GRANT de permisos a los usuarios:

Según la consigna, se le otorgan permisos de lectura al **usuario\_1** sobre toda la base de datos mientras que al **usuario\_2** se le otorga permisos de lectura, inserción y modificación:



Se observan los permisos otorgados:

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

## Check del funcionamiento de los permisos en la BBDD:

Primero se crean en MySQL Workbench dos nuevas conexiones para cada uno de los usuarios:

Graphical user interface, text, application, chat or text message

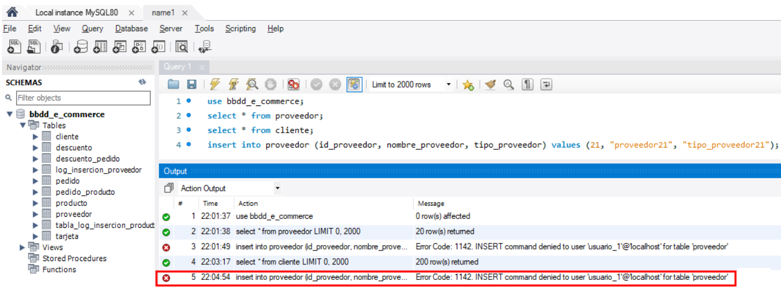
Description automatically generated

En el caso del **usuario\_1** vemos que puede acceder en modo lectura (es decir, llamar información a través de un SELECT) en todas las tablas de la BBDD:

Graphical user interface

Description automatically generated

Pero cuando se intenta insertar un registro obtenemos el siguiente error:



En el caso del **usuario\_2** vemos que puede acceder en modo lectura a todas las tablas sin inconvenientes:

Graphical user interface, text, application

Description automatically generated

Además, puede realizar un INSERT sin problemas:

Graphical user interface, text, application, email

Description automatically generated

Y un UPDATE:

Graphical user interface, text, application, email

Description automatically generated

Pero cuando se intenta eliminar el registro obtenemos la restricción del permiso:

Graphical user interface, text, application, email

Description automatically generated

# Sentencias del sub-lenguaje TCL

## DELETE registros tabla TARJETA con TCL

Se hará un DELETE de registros de la tabla TARJETA utilizando TCL. Primero, se llama a la tabla TARJETA para ver qué contiene y vemos que tiene 100 tarjetas registradas:

Graphical user interface, text, application

Description automatically generated

Para el DELETE con TCL se inicia la transacción con START TRANSACTION. En la siguiente línea se eliminan los últimos 3 registros (id = 100, 99, 98), que por estar dentro de la TRANSACTION no surten efecto hasta que se cierre la transacción con un COMMIT.

Finalmente se hacen efectivos los cambios con la sentencia COMMIT:

A picture containing chart

Description automatically generated

Se chequea la tabla TARJETA y se observa que los últimos 3 registros fueron eliminados correctamente:

Graphical user interface, text, application

Description automatically generated

Por último, se vuelven a insertar los 3 registros eliminados para dejar la BBDD como estaba antes del DELETE:

Graphical user interface, text

Description automatically generated

## INSERT registros tabla DESCUENTO con TCL

De acuerdo con la consigna se procede a hacer un INSERT de 8 registros a la tabla DESCUENTO con TCL e implementando SAVEPOINTs.

Primero, se chequea la tabla antes del INSERT y se observa que existen solo 5 registros en la tabla:

Graphical user interface, text, application, email

Description automatically generated

Se arma el INSERT con TCL de los 8 nuevo registros. Además, se agrega un SAVEPOINT (primeros\_cuatro\_registros) después de la inserción del 4to registro y otro SAVEPOINT (segundos\_cuatro\_registros) después de la inserción de los últimos 4 registros.

A picture containing calendar

Description automatically generated

A modo de demostrar el funcionamiento, aún dentro de la TRANSACTION, llamamos un SELECT y se ve que los 8 registros fueron insertados en la tabla, pero aún no han impactado en la base de datos dado que la TRANSACTION sigue abierta:

Graphical user interface, text, application, email

Description automatically generated

Al hacer ROLLBACK TO segundos\_cuatro\_registros, volvemos al mismo lugar dado que este ROLLBACK se agregó después del INSERT de los 8 registros:

Graphical user interface, text, application

Description automatically generated

Pero al hacer el ROLLBACK TO primeros\_cuatro\_registros y luego un SELECT vemos que se vuelve al momento de la TRANSACTION donde solo se habían agregado 4 registros:

Graphical user interface, text, application

Description automatically generated

Finalmente, se eliminan los SAVEPOINTs creados y se cierra la TRANSACTION con un ROLLBACK para evitar impactar la base de datos como se observa en el último SELECT:

Graphical user interface, text, application, email

Description automatically generated

# Backup de la base de datos

## Backup

De acuerdo con lo revisado en clase, para generar el backup de nuestra base de datos a través de MySQL Workbench, vamos a la solapa “Administration” y ahí a “Data Export”:

Graphical user interface, text, application

Description automatically generated

Lo cual nos lleva a la siguiente pantalla dónde se elige la base de datos que se quiere exportar así cómo sus tablas y vistas:

Graphical user interface, application

Description automatically generated

Se selección la base de datos del proyecto final (bbdd\_e\_commerce), todas sus tablas y en el selector de abajo a la derecha se elige “Dump Structure and Data”.

En los “objetos a exportar” tildamos Store Procedures, funciones y Triggers:

Graphical user interface, application

Description automatically generated

Por último, y, de acuerdo con la consigna, exportamos todo a un único archivo, elegimos la carpeta destino dónde se guardará el .sql y tildamos la opción de “Include Create Schema” antes de darle “Start Export”:

Graphical user interface, text, application, email

Description automatically generated

Se controla la exportación y se observa que todo salió bien:

A picture containing shape

Description automatically generated

Este archivo se incluye como parte de la entrega.

# Tecnologías y Bibliografía

## Tecnologías:

* MySQL & MySQL Workbench.
* MS Excel, Word, Power Point.
* <https://www.mockaroo.com/>

## Bibliografía:

Blog Fondo Monetario Internacional, 2022:

* <https://blogs.imf.org/2022/03/17/pandemics-e-commerce-surge-proves-less-persistent-more-varied/#:~:text=On%20average%2C%20the%20online%20share,to%2012.2%20percent%20in%202021>

Conferencia de las Naciones Unidas sobre Comercio y Desarrollo, 2021:

* <https://unctad.org/es/node/32413#:~:text=As%20lockdowns%20became%20the%20new,to%20about%2017%25%20in%202020>.

Ejemplos bases de datos comercio electrónico:

* <https://fabric.inc/blog/ecommerce-database-design-example/>
* <http://www.webassist.com/tutorials/Free-eCommerce-MySQL-Database>
* <https://github.com/runninguru/MySQL-eCommerce>

Investopedia:

* <https://www.investopedia.com/terms/e/ecommerce.asp>

Otros:

* <https://stackoverflow.com>
* <https://www.w3schools.com/>

1. <https://www.kaggle.com/datasets/sootersaalu/amazon-top-50-bestselling-books-2009-2019> [↑](#footnote-ref-1)
2. <https://fabric.inc/blog/ecommerce-database-design-example/>, <http://www.webassist.com/tutorials/Free-eCommerce-MySQL-Database>, <https://github.com/runninguru/MySQL-eCommerce>. [↑](#footnote-ref-2)
3. El 50% de beneficio bruto es lógico dado que a todos los productos se le aplicó el mismo margen de ganancia. [↑](#footnote-ref-3)