

LABORATORIO #4

ADMINISTRACIÓN DE BASE DE DATOS

GESTIÓN DE CAMBIOS

1. OBJETIVOS

- Realizar la administración de cambios a través de procesos documentados en una base de datos.
- Establecer procesos que permitan realizar acciones de forma segura y con el menor impacto posible.
- Evitar la pérdida de integridad de los datos al realizar cambios que afecten a las bases de datos y a las tablas.
- Utilizar herramientas que puedan automatizar y garantizar los procedimientos de gestión de cambios.
- Realizar control de versiones.

2. MARCO TEORICO

Para implementar con éxito la gestión efectiva de un cambio, es esencial comprender un conjunto de requisitos básicos para garantizar el éxito.

Los factores que deben incorporarse a la gestión de cambios son:

- Pro-actividad
- Inteligencia
- Análisis (planificación e impacto)
- Automatización
- Estandarización
- Confiabilidad
- Previsibilidad y entrega rápida y eficiente.

La perspectiva del DBA de la gestión de cambios

El DBA es el custodio de los cambios en la base de datos, eso quiere decir que, se encarga de realizar los cambios en la base de datos y de garantizar que cada cambio se realice con éxito y sin impacto en el resto de la base de datos.

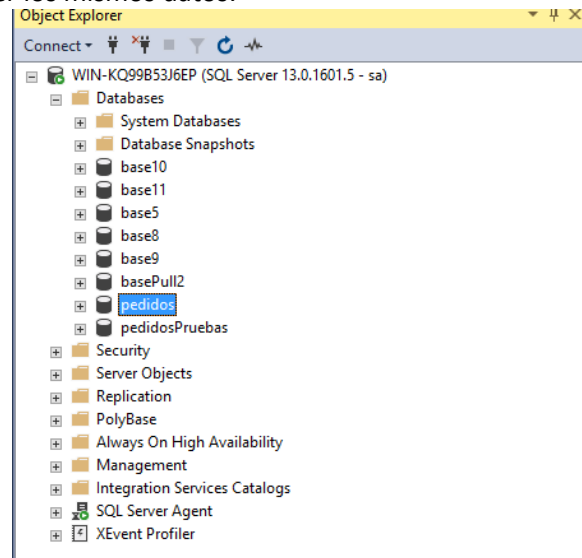
Para realizar cambios en la base de datos de manera efectiva, el DBA debe considerar cada uno de los elementos antes mencionados, pro-actividad, análisis de impacto, etc.

Existen varios tipos de cambios:

- Cambios en el software DBMS
- Cambios en la configuración de hardware
- Cambios en el diseño lógico y físico.
- Cambios en las aplicaciones
- Cambios en las estructuras físicas de la base de datos

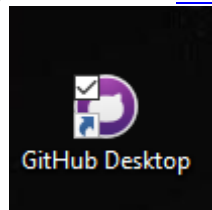
3. PRACTICA

1.- Se debe definir dos ambientes (de producción y pruebas) para esta base de datos. Las bases de datos deben ser iguales y contener los mismos datos.

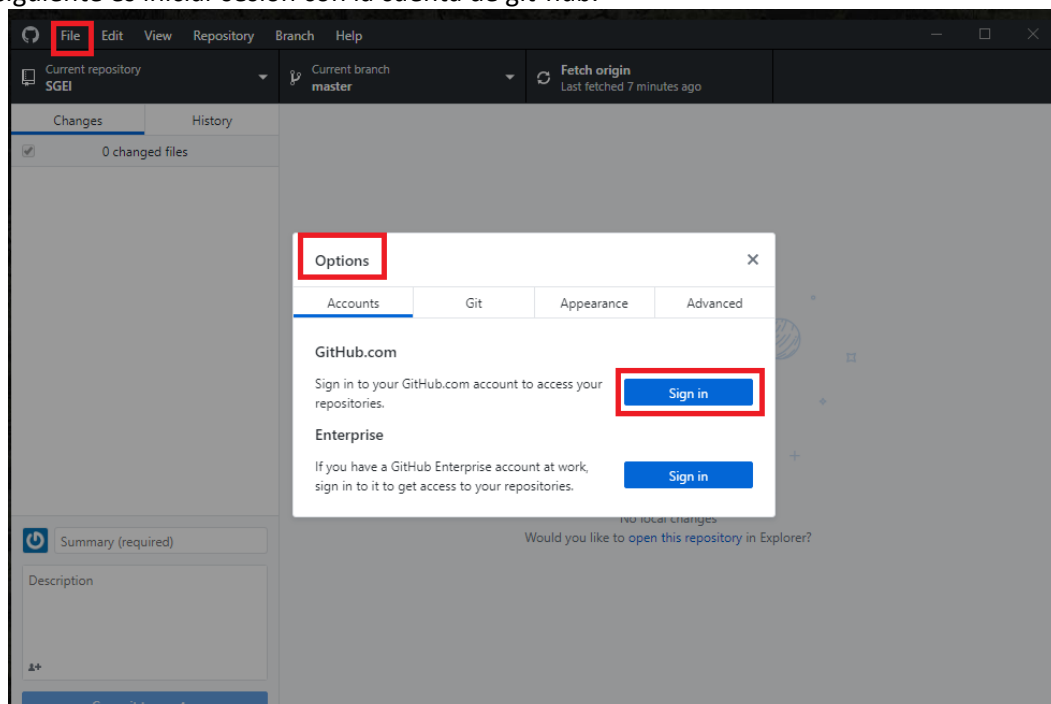


2.- Definir los procedimientos para realizar control de versiones utilizando una herramienta git

- Lo primero es instalar la aplicación de control de versión git preferida, en este caso será GitHub que puede ser descargado desde el siguiente enlace: <https://desktop.github.com/>



- Lo siguiente es iniciar sesión con la cuenta de git-hub.



Sign in

Username or email address

Ernesteins

Password

.....

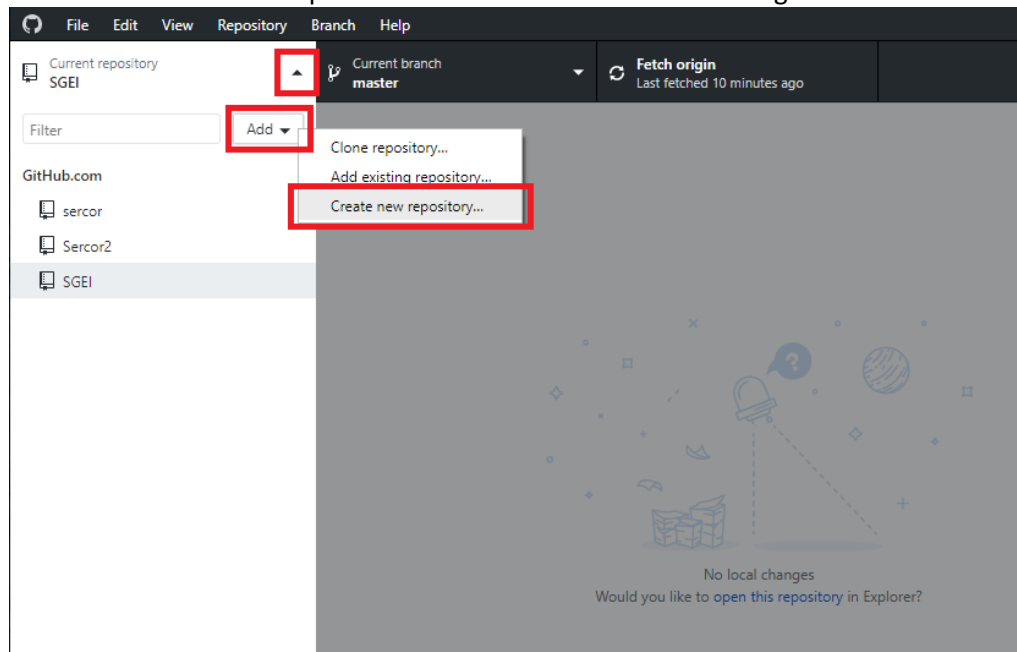
[Forgot password?](#)

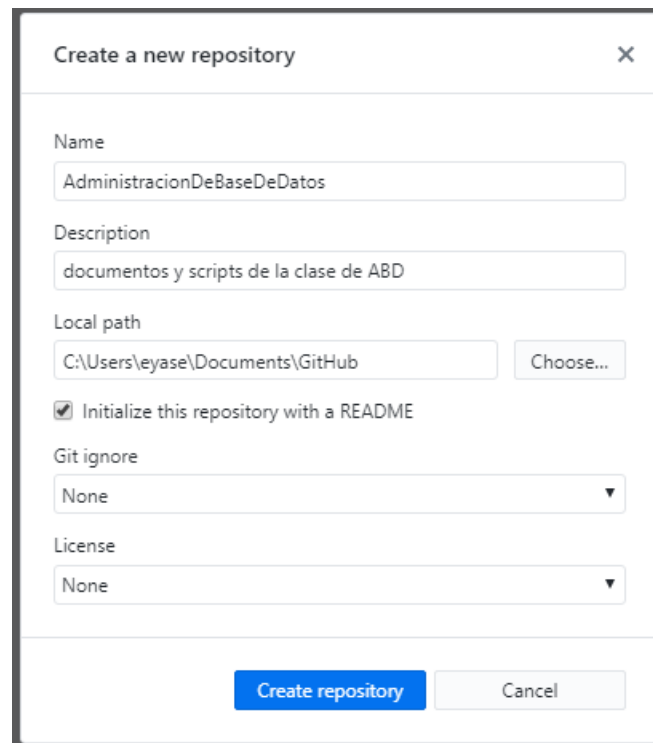
or

[Sign in using your browser](#)

Sign in Cancel

- Ahora se debe crear un repositorio donde se almacenará los códigos de la base de datos.





Create a new repository

Name
AdministracionDeBaseDeDatos

Description
documentos y scripts de la clase de ABD

Local path
C:\Users\eyase\Documents\GitHub Choose...

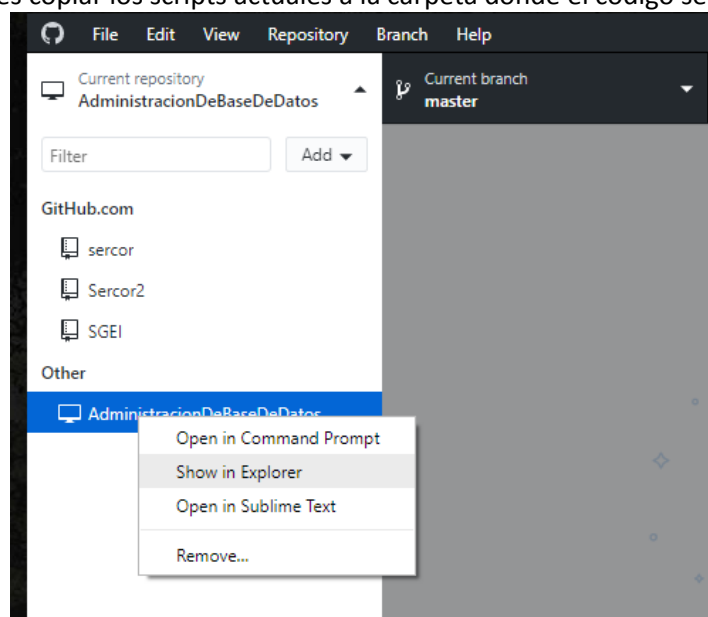
☒ Initialize this repository with a README

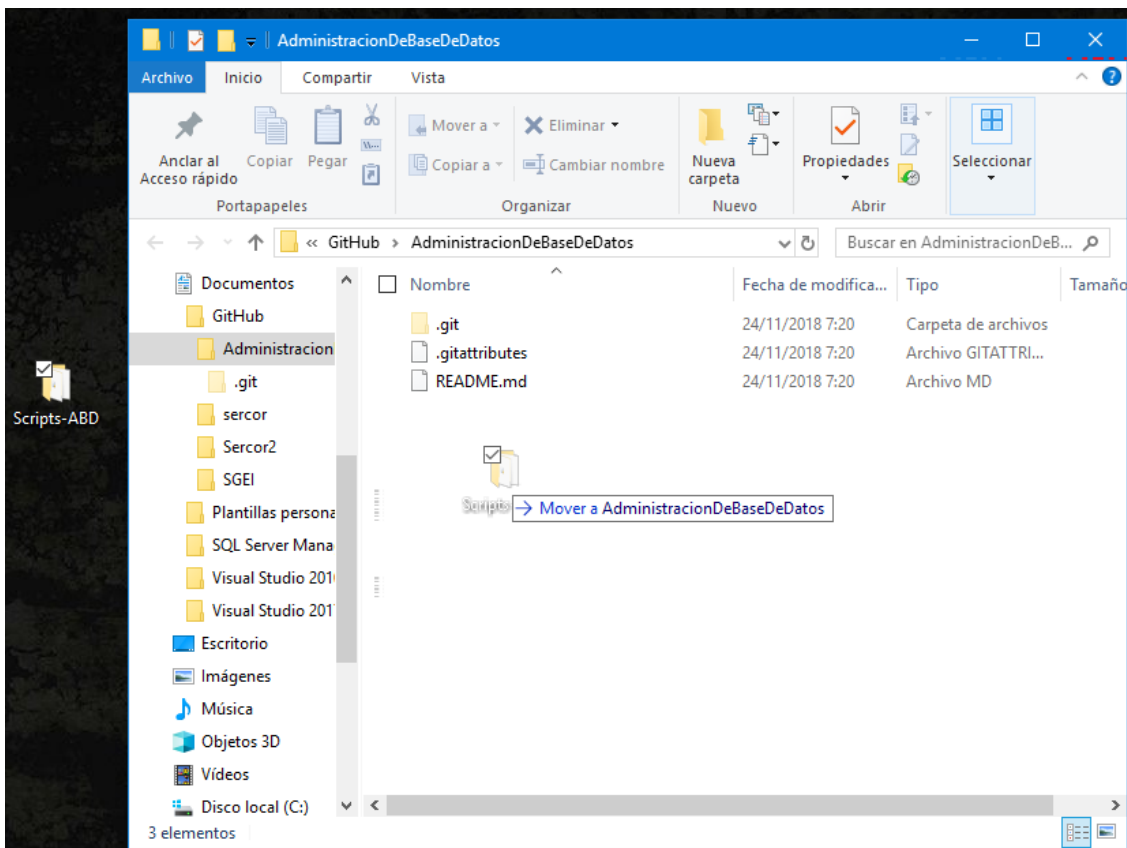
Git ignore
None

License
None

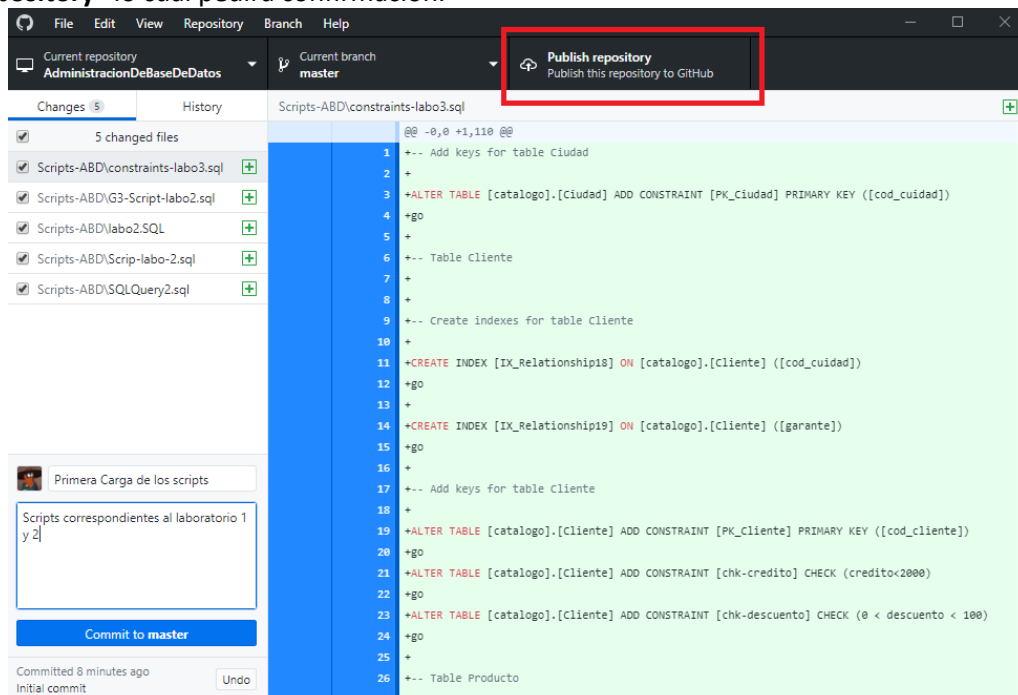
Create repository Cancel

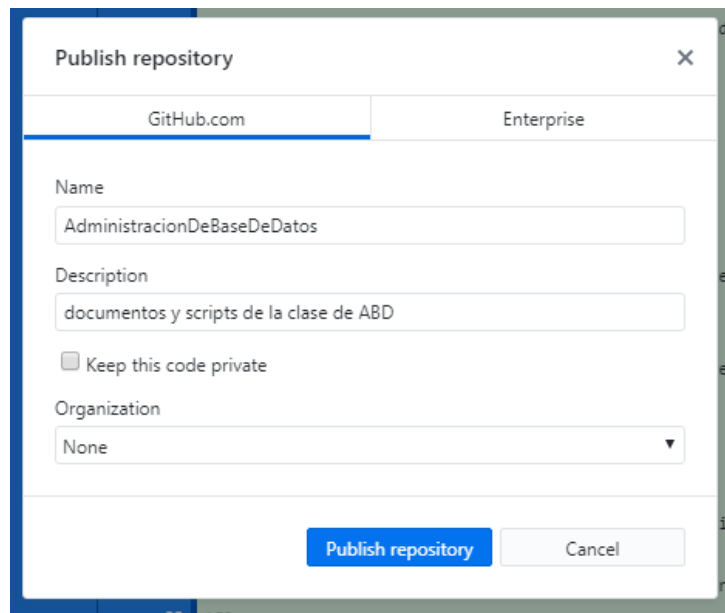
- Lo siguiente es copiar los scripts actuales a la carpeta donde el código se sincroniza.



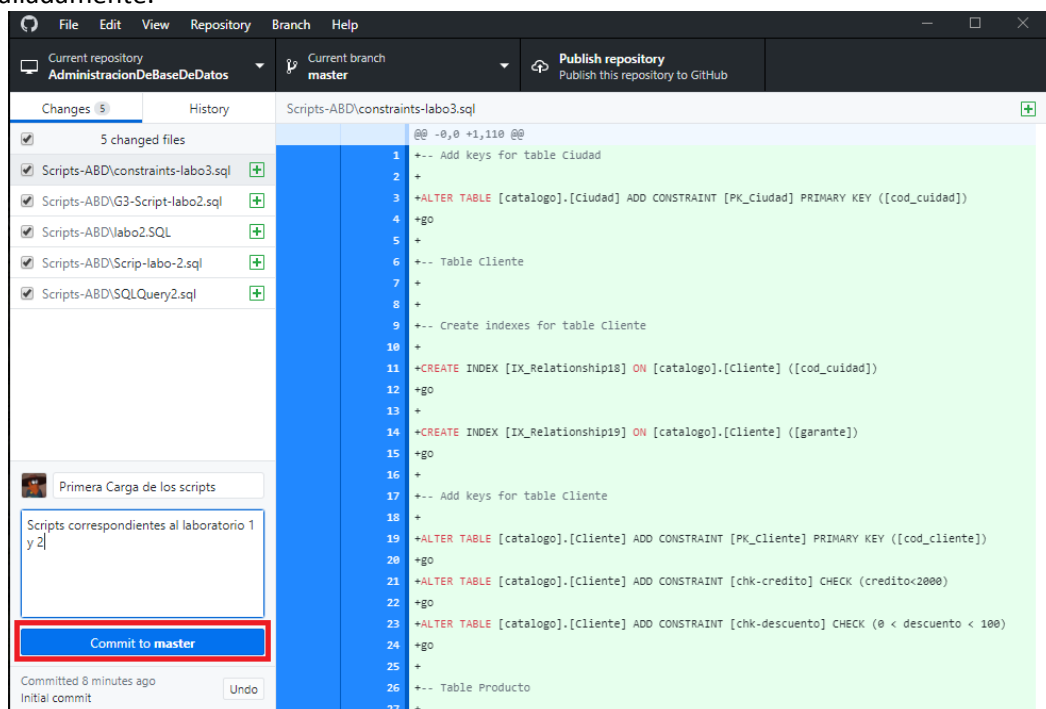


- Lo siguiente es publicar el repositorio en la nube haciendo clic en la opción: “**publish repository**” lo cual pedirá confirmación.

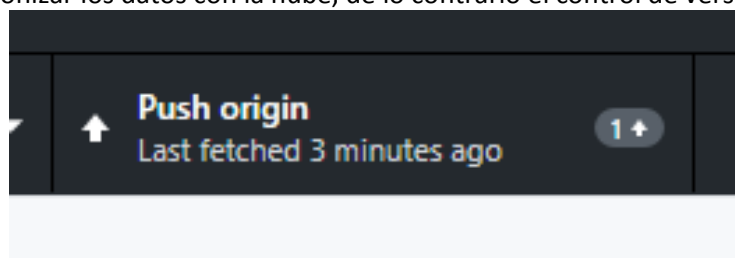




- Ahora se debe usar la aplicación para hacer “commit” de los códigos realizados siempre con un pequeño resumen de lo que lleva el cambio como mínimo, lo recomendado sería describirlo detalladamente.



- Después, sincronizar los datos con la nube, de lo contrario el control de versiones será local.



- Ahora se debe repetir los últimos 2 pasos “commit” y “push” cada vez que se realicen cambios

al código que deban ser versionados.

3.- La compañía está creciendo debido al éxito obtenido y por tanto incrementarán sus clientes, por lo que se prevee a futuro unos 10000 clientes activos y para ofrecer más facilidades se planea incrementar su crédito en un 10%, por tanto el código de cliente actual no podrá ser usado por las limitaciones de espacio y se debe ampliar a un formato de la siguiente forma: "C00001". Adicionalmente, este aumento de clientes ocasionará un incremento en los pedidos por lo que el departamento de ventas solicita se incremente la capacidad de este campo a 10 posiciones y dado que la letra "R" utilizada en su codificación causa muchos inconvenientes se requiere que cambie a "PE", ejemplo: "PE00000001"

4.- Establecer la gestión de cambios para esta solicitud (Plan de implementación del cambio, Plan de contingencia).

- Plan de implementación del cambio
 - i. Realizar el código necesario para poder satisfacer los cambios.
 - ii. Implementar los cambios en el ambiente de pruebas
 - iii. Si los cambios son exitosos en el ambiente de prueba y no generan ningún inconveniente, implementar los cambios en el ambiente de producción.
- Plan de contingencia
 - i. Realizar un Backup de la base de datos antes de realizar los cambios.
 - ii. En caso de surgir un problema al implementar los cambios, usar el Backup para volver al estado previo a los cambios.

5.- Ejecutar el plan de implementación del cambio en el ambiente de pruebas. Si es exitoso, implementarlo en el ambiente de producción y documentarlo (Considerar la gestión de versiones del script)

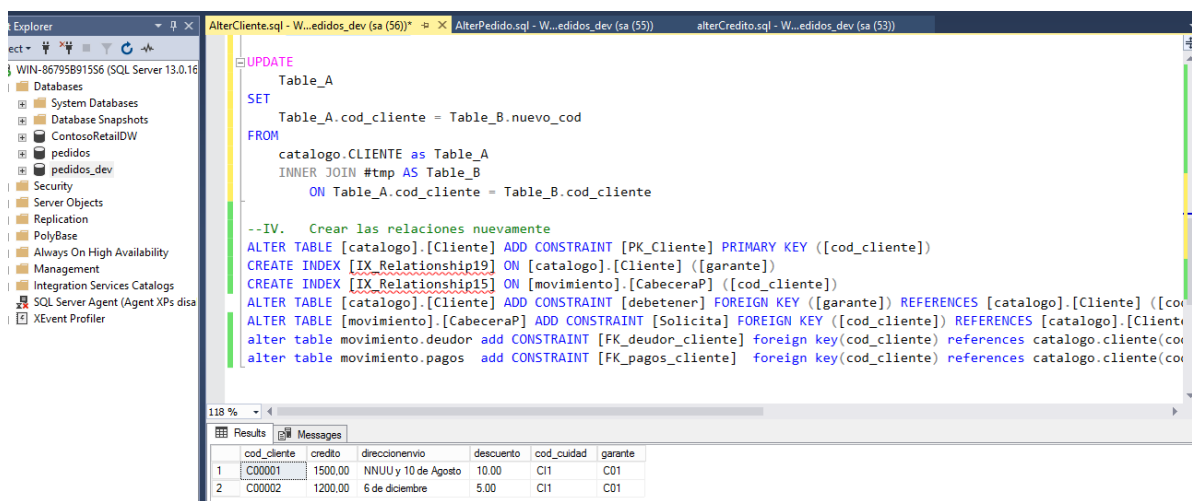
- Cambiar el código de Cliente
 - i. Realizar un respaldo de la tabla
 - ✓ Realizar las actividades de respaldo en el plan de contingencia.
 - ii. Eliminar las relaciones de la tabla Cliente
 - ✓ ForeignKey cabeceraP
 - ✓ ForeignKey Pagos
 - ✓ PrimaryKey Cliente
 - ✓ Relación Cliente (garante)
 - ✓ Relación CabeceraP
 - ✓ Índice garante
 - ✓ Índice CabeceraP
 - iii. Alterar la columna
 - ✓ En Cliente
 - a. Cod_cliente
 - b. Garante
 - ✓ En CabeceraP
 - a. Cod_Cliente
 - ✓ En Pagos
 - a. Cod_Cliente
 - ✓ En Deudor
 - a. Cod_Cliente
 - b. Garante

iv. Actualizar datos de codificación de los registros existentes

- ✓ Registros en Cliente
- ✓ Registros en CabeceraP
- ✓ Registros en Deudor
- ✓ Registros en Pagos

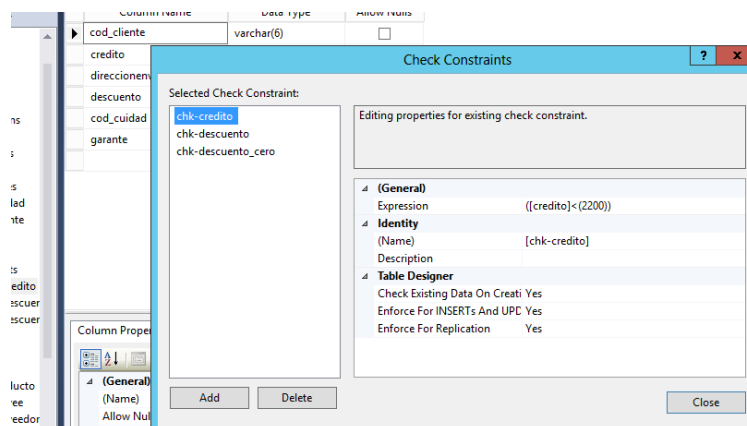
v. Crear las relaciones nuevamente

- ✓ PrimaryKey Cliente
- ✓ ForeignKey cabeceraP
- ✓ FereignKey Pagos
- ✓ Relación CabeceraP
- ✓ Relación Cliente (garante)
- ✓ Índice garante
- ✓ Índice CabeceraP



• Cambio en el descuento

- ✓ Eliminar Check de límite de descuento en Cliente
- ✓ Crear check de límite de descuento en Cliente con el nuevo cambio
- ✓ Revisión de inconformidades



• Cambiar el código de pedido

i. Eliminar las relaciones de la tabla Cliente

- ✓ PrimaryKey CabeceraP
- ✓ PrimaryKey DetalleP
- ✓ Relación DetalleP

ii. Alterar la columna

- ✓ En CabeceraP
 - a. Cod_pedido
- ✓ En DetalleP
 - a. Cod_pedido

iii. Actualizar los registros existentes

- ✓ Registros en CabeceraP
- ✓ Registros en DetalleP

iv. Crear las relaciones nuevamente

- ✓ PrimaryKey CabeceraP
- ✓ PrimaryKey DetalleP
- ✓ Relación DetalleP

```

ALTER TABLE [movimiento].[CabeceraP] ADD CONSTRAINT [PK_CabeceraP] PRIMARY KEY ([cod_pedido])
go
ALTER TABLE [movimiento].[DetalleP] ADD CONSTRAINT [PK_DetalleP] PRIMARY KEY ([cod_pedido],[numlinea])
go
ALTER TABLE [movimiento].[DetalleP] ADD CONSTRAINT [Tiene] FOREIGN KEY ([cod_pedido]) REFERENCES [movimi.
go
  
```

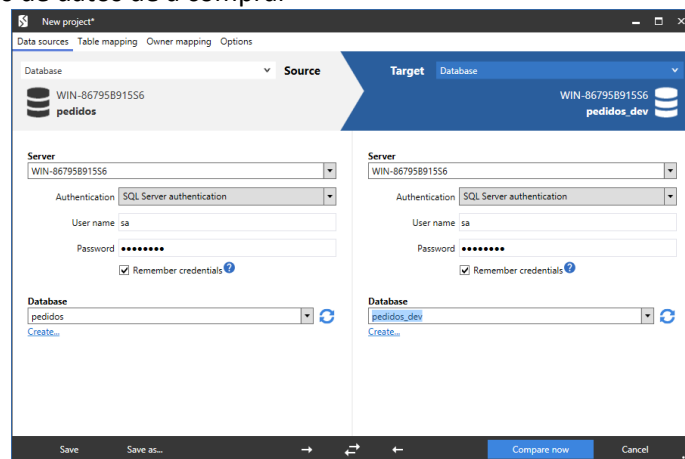
	fecha	montoTotal	cod_pedido	cod_cliente	TipoPed
1	2018-10-01	500.00	PE00000001	C01	contado
2	2018-10-11	165.00	PE00000002	C01	contado
3	2018-10-01	500.00	PE00000003	C01	credito
4	2018-10-07	550.00	PE00000008	C01	credito

	numlinea	precioUnit	cantidad	cod_producto	cod_pedido
1	1	10.50	5	P02	PE00000002
2	2	7.50	4	P01	PE00000002
3	6	7.50	4	P02	PE00000008

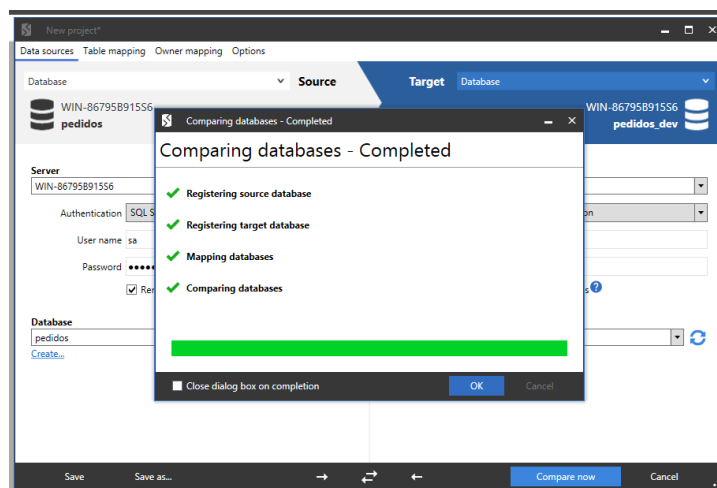
6.- Mediante una herramienta de comparación de base de datos (investigar en internet) (idera.com, redgate.com), determinar que la base de pruebas y de producción son iguales (después de implementar el cambio)

Usar el programa para comparación de bases de datos de Redgate.

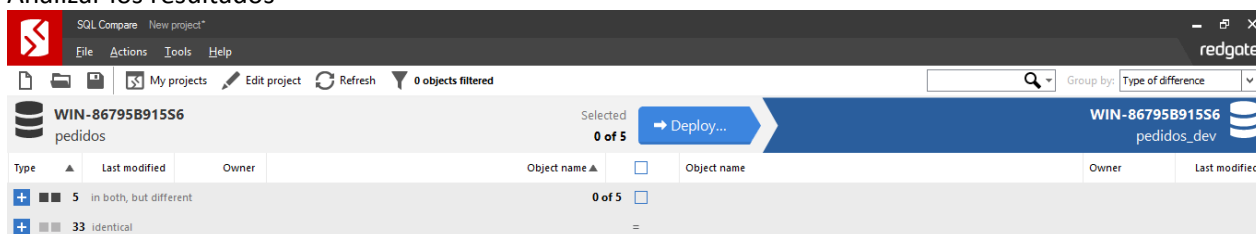
Seleccionar los las bases de datos de a comprar



Esperamos a que termine la comparación.



Analizar los resultados



7.- Simular que hubo un error en la implementación del cambio en el ambiente de producción, (Ejemplo, se puede perder información). Implementar el plan de contingencia para que se deshagan todos los cambios y la base de datos de producción retorne al estado anterior a este proceso.

- Se omitió la definición “NOT NULL” al modificar la columna cod_cliente de la tabla catalogo.cliente.
- A esta altura ya se ha modificado los campos y los registros.
- Para retornar al punto antes de que el ambiente de producción estaba sin el problema, es necesario aplicar las actividades de recuperación del plan de contingencia.

4. CONCLUSIONES

- Se analizó la dificultad para realizar un cambio. Ya que se debe realizar un análisis de todas las relaciones que tiene la tabla, para así identificar a cuales afectará.
- Se observó que mantener la integridad de datos es una tarea que requiere mucha atención al realizar un cambio.
- Debido a que al volver a crear las relaciones o modificar los campos, se puede olvidar de añadir la integridad. Convirtiéndose esto en un error humano.
- Se estableció un checklist diseñado específicamente para los cambios solicitados en la práctica
- Se utilizó una herramienta de comparación de Base de Datos.
- Durante la ejecución del plan de implementación se detectaron anomalías en las claves foráneas, habían 12 constraints que hacían lo mismo.

5. REFERENCIAS

[1] C. Mullins, *Database administration: the complete guide to DBA practices and procedures*, 2nd ed. Upper Saddle River NJ: Addison-Wesley, 2013.