



UNIVERSIDAD NACIONAL DEL NORDESTE

Carrera: Licenciatura en Sistemas de la Información

Título: Proyecto de Estudio e Investigación

Tema: Backup y Restore de una base de datos

Profesores: Cuzziol Juan José – Sambrana Iván

Integrantes:

González Rodrigo Sebastián	43346372
Pinat Juan Matias	39190409
Olivos Nicolas	46073510
Gauna Natalia	42514355
Jaramillo Matias	40440862

Índice

1. Introducción	3
1.1 Tema	3
1.2 Definición del planteamiento o problema... ..	3
1.3 Objetivo del Trabajo Practico... ..	5
1.3.1 Objetivos Generales.....	5
1.3.2 Objetivos Específicos... ..	5
2. Marco conceptual o Referencial	5
3. Metodología Seguida	6
3.1. Descripción de cómo se realizó el TP	6
3.2 Herramientas (Instrumentos y Procedimientos)	6
4. Desarrollo del tema/Presentación de los resultados	7
5. Integración de temas.....	20
5.1 Procedimientos y funciones almacenadas.....	20
5.2 Triggers.....	33
5.3 Réplicas de base de datos.....	49
6. Conclusión	92
7. Bibliografía.....	92

Introducción

En el presente trabajo de investigación hablaremos acerca del grado de importancia que tienen las copias de seguridad (Backup – Backup en línea) y las restauraciones (Restore) en una Base de datos, a su vez, veremos en qué momento se utilizan o para que se utilizan, además de cómo es su implementación y su desarrollo.

Temas

Copia de Seguridad (Backup): Una copia de seguridad es una réplica de los datos almacenados en una base de datos en un momento específico. Estas copias se crean con el propósito de preservar la integridad y la disponibilidad de los datos, por si se produjera alguna pérdida, daño o eliminación accidental de los datos. Existen distintos tipos de copias de seguridad, tales como: Completas (Las cuales respaldan toda la base de datos), Diferenciales (respaldan solo los cambios desde la última copia completada), Incrementales (respaldan los cambios desde la última copia completada o incremental), o continuas (realizadas constantemente).

Restauración (Restore): La restauración es un proceso en donde se recuperan los datos de una copia de seguridad y se los vuelve a cargar en la base de datos, permitiendo así volver a obtener los datos hasta el momento de realizar la copia. Para realizar una restauración, se debe tener una copia válida de seguridad y acceso a herramientas de administración de la base de datos que permitirán llevar a cabo dicho proceso.

Backup en Línea: Una base de datos en línea, también conocida como base de datos en la nube o base de datos alojada en línea, es un sistema de gestión de datos que se encuentra en una ubicación remota de un proveedor de servicios y se accede a través de una conexión de red. Estas bases de datos permiten a los usuarios almacenar, administrar y acceder a información de manera colaborativa y desde diversas ubicaciones geográficas sin necesidad de disponer de dispositivos físico locales donde se almacenan los datos.

Definición y Planteamiento del Problema

¿Cuándo o en que situaciones se debe utilizar una copia de seguridad y una restauración de los datos?

Las copias de seguridad (Backup) como ya se mencionó previamente son esenciales para garantizar la integridad y la recuperación de los datos ante situaciones de desastres o de control, tales como:

- Prevención de pérdida de datos: Se realizan copias de seguridad de forma regular (diaria, semanal, mensual, etc.) para proteger los datos en caso de pérdida debido a errores humanos, fallos de hardware, o corrupción de datos.
- Actualizaciones o Cambios: Antes de realizar cambios significativos en la estructura de la base de datos o en los mismos datos, se debe realizar una copia de seguridad, de esta manera ante cualquier tipo de complicación o error, se puede volver a dejar la base como estaba previamente.

- Migración de Datos: Al migrar una base de datos de un sistema a otro, es fundamental tener una copia de seguridad de los datos originales, para garantizar una transición sin problemas y minimizar el riesgo de pérdida de datos.
- Respuesta ante Desastres Naturales: En situaciones de desastre naturales, tales como, inundaciones, incendios o terremotos, las copias de seguridad fuera del sitio (en ubicaciones geográficas diferentes) son vitales para la recuperación de los datos.

Podríamos decir que la restauración de los datos es la forma en la que nos podemos asegurar de volver a obtener la estructura de la base de datos o los mismos datos en sí, algunas de las situaciones comunes en las que se debe realizar una restauración de los datos son las siguientes:

- Perdida de datos por Errores Humanos: Cuando el usuario elimina accidentalmente datos importantes o sobrescribe información esencial, una restauración puede ayudar a recuperar esos datos.
- Corrupción de datos: Si se detecta que la base de datos se ha corrompido, lo que puede ocurrir debido a fallos de hardware, errores de software o problemas de almacenamiento, se puede realizar una restauración desde una copia de seguridad válida para restaurar la integridad de los datos.
- Ataques Cibernéticos: En caso de cualquier tipo de ataque cibernético hacia los datos, se puede optar por una restauración para de esta manera eliminar el impacto del ataque y recuperar los datos originales.
- Pruebas y Desarrollo: En entornos de desarrollo y pruebas, es común realizar restauraciones de datos a partir de copias de seguridad para crear un entorno de prueba con datos actualizados o específicos para realizar pruebas sin afectar los datos en producción.

Por último, tendríamos al Backup en línea, el cual tiene las mismas utilidades que el Backup “Normal” o “Tradicionales” pero la diferencia radica en donde se almacena los datos y como se acceden a estos:

1. Backup (Copia de Seguridad)

- **Almacenamiento:** Se crean copiando los datos de un dispositivo (Como una computadora o un servidor) y los guardan en un medio de almacenamiento externo o en dispositivos locales, como: discos duros externos, unidades USB, cintas magnéticas, etc.
- **Acceso:** Generalmente para restaurar los datos es necesario acceder físicamente al medio de almacenamiento donde se encuentra la copia de seguridad. Esto significa que se debe tener el dispositivo de respaldo al alcance o estar conectado a el de alguna manera.
- **Ventajas:** Las copias locales son utiles cuando se necesita recuperar datos de manera rápida y de esta manera no estar dependiendo al uso de la conexión a internet

2. Backup (Copia de seguridad en Línea o En la nube)

- Almacenamiento: Este tipo de Backup implica almacenar los datos en servidores remotos a los que se accede a través de una conexión de red. Los datos se cifran y se almacenan en centros de datos seguros gestionados por

proveedores de servicios en la nube. La nube es un término genérico que se utiliza para describir una infraestructura física, una red de ordenadores y almacenamiento seguros que están ubicados en un centro de datos que es propiedad de un proveedor de servicios en la nube que también lo gestiona.

- Acceso: Para restaurar datos desde este tipo de Backup, solo se necesita disponer una conexión de internet y las respectivas credenciales de acceso al servicio de la nube.
- Ventajas: las copias de seguridad online son seguras, prácticas y permiten el acceso desde “cualquier lugar”. Ofrecen una mayor seguridad contra pérdidas debidas a desastres locales, como incendios o robos, ya que los datos se almacenan en forma remota. Otra ventaja en el uso de estos servicios en la nube es que ni las compañías ni los particulares tienen que invertir en hardware o que ocuparse de tareas técnicas tales como configurar el hardware, cargar el software o proteger los datos.

Objetivo del Trabajo Practico

En el presente trabajo de estudio se llevó a cabo con el fin de investigar e informarnos el uso y el fin que tienen las copias de seguridad y las restauraciones en una base de datos, cuando y como se deben utilizar y la implementación de los mismos.

Objetivos Generales

Como objetivos generales buscamos alcanzar herramientas necesarias para lograr obtener la información necesaria sobre importancia y el uso de estas prácticas que son fundamentales dentro del contexto de las bases de datos.

Objetivos Específicos

Como objetivos específicos tuvimos la necesidad de obtener los conocimientos necesarios y suficientes para implementar estas prácticas en una base de datos, para así poder resguardar los datos en una copia de seguridad y restaurar los mismos en caso de que sea necesario hacerlo.

Marco Conceptual o Referencial

A continuación, definiremos algunos conceptos/fundamentos de manera más técnica y sólida, para que de esta manera nos ayuden a comprender mejor el funcionamiento que tendrán estas prácticas.

1. Backup (Copia de seguridad):

- **Definición:** El Backup es el proceso de crear una copia de los datos de una base de datos en un punto específico en el tiempo. Esto se hace para proteger los datos contra pérdidas debidas a fallos del sistema, errores humanos, corrupción de datos, o incluso desastres naturales.
- **Tipos de Backup:**
 - *Full Backup:* Realiza una copia completa de toda la base de datos.
 - *Differential Backup:* Guarda solo los cambios realizados desde el último Backup completo.
 - *Transaction Log Backup:* Almacena las transacciones registradas desde el último Backup del registro de transacciones.
- **Importancia:** Los backups son esenciales para la recuperación de datos en situaciones de pérdida, y son parte fundamental de la estrategia de recuperación de desastres.

2. Restore (Restauración):

- **Definición:** La restauración es el proceso de recuperar datos desde un Backup y volver a ponerlos en la base de datos. Permite regresar la base de datos a un estado anterior y es crucial para recuperar datos perdidos o dañados.
- **Tipos de restauración:**
 - *Restauración de base de datos completa:* Restaura una base de datos completa a partir de un Backup completo.
 - *Restauración de diferencial:* Restaura una base de datos completa y luego aplica un Backup diferencial.
 - *Restauración de registro de transacciones:* Permite restaurar la base de datos a un punto en el tiempo específico usando backups del registro de transacciones.
- **Importancia:** La restauración es la última línea de defensa contra la pérdida de datos y es crucial para garantizar la integridad de una base de datos.

3. Backup en un archivo de recurso compartido de red en SQL Server:

- **Definición:** Esto permite crear un Backup en un dispositivo remoto a través del acceso al recurso compartido de red. Disponiendo de los permisos necesarios para realizar las copias de seguridad, restauración, escribiendo y leyendo en el recurso compartido de red.
Para realizar una copia de seguridad en una unidad de red cuando SQL Server está en ejecución se debe asignar la unidad compartida como una unidad de red en la sesión iniciada.

Metodología Seguida

Como metodología lo que hicimos primeramente es investigar e informarnos en internet los pasos a seguir para poder realizar correctamente un Backup y un Restore en una base de datos y luego pasamos directamente al motor de SQL Server para intentar realizar dichas prácticas.

Descripción de cómo se realizó el T.P

Cada uno de los integrantes del grupo trato de informarse e investigar cómo llevar a cabo correctamente un Backup y un Restore en una base de datos, luego nos reunimos vía meet para comentar que fue lo que cada uno investigo y de esta manera llegamos a un mutuo acuerdo entre todos los integrantes para realizar el trabajo.

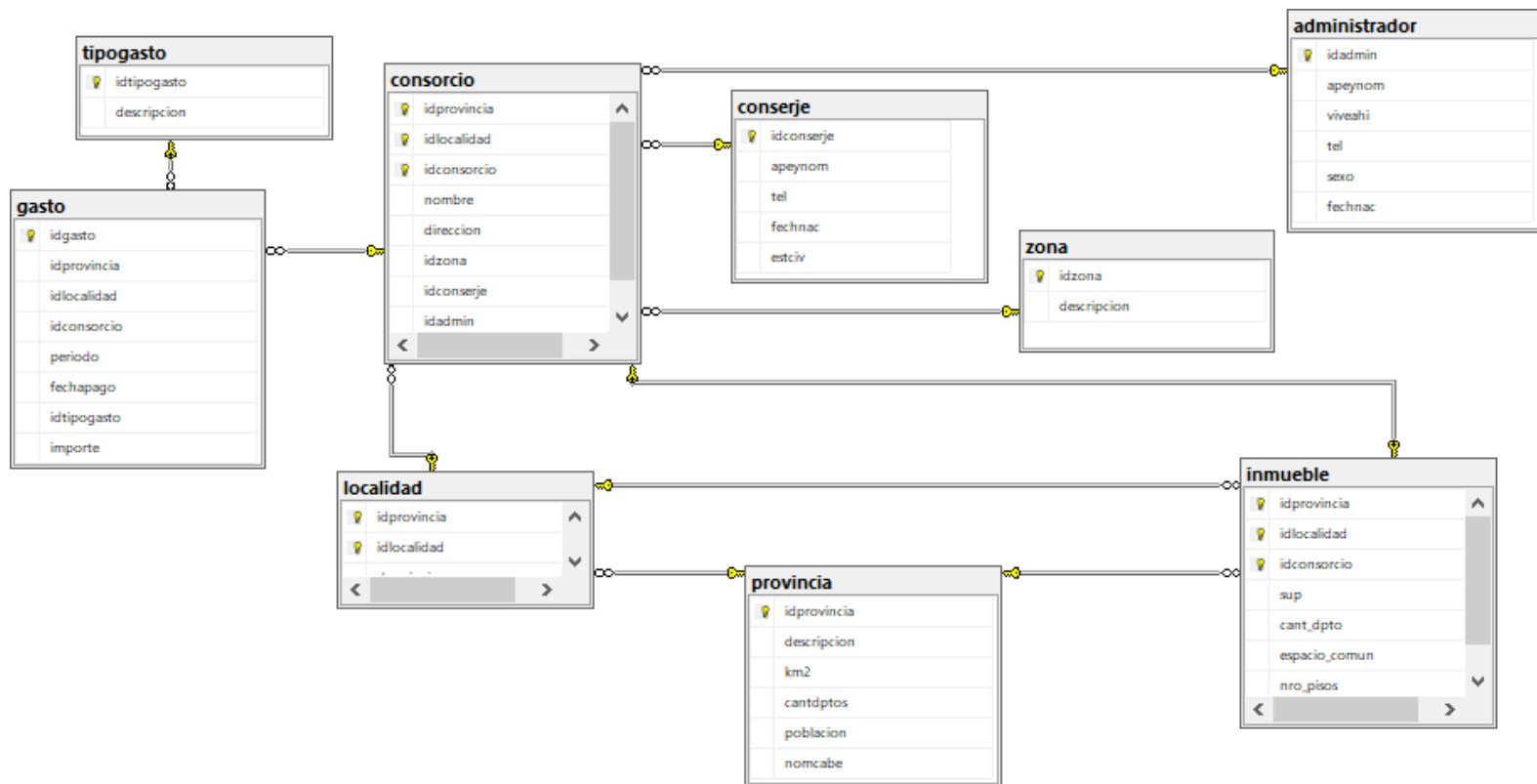
Herramientas (Instrumentos y Procedimientos)

Mayormente el método que utilizamos para recabar información fue internet ya sea desde artículos en donde se habla o mencionaba como realizar dichos procedimientos hasta videos en donde podíamos ver directamente el código que se utilizaba acompañado de una explicación en donde se comentaba que acción realizaba cada línea de código.

Desarrollo del Tema / Presentación de Resultados

A continuación presentaremos el desarrollo y los resultados que obtuvimos por parte de nuestra investigación realizada pero antes para ponernos en contexto recordemos lo que se habló al comienzo del presente trabajo de investigación: “Una copia de seguridad es una réplica de los datos almacenados en una base de datos en un momento específico - La restauración es un proceso en donde se recuperan los datos de una copia de seguridad y se los vuelve a cargar en la base de datos” esto se puede ver de una manera más ilustrativa en la siguiente imagen.

Una vez mencionado esto, pasaremos hablar primeramente de como realizamos el Backup, para ello dentro del moto SQL Server tomamos la base de datos llamada “base_consortio”

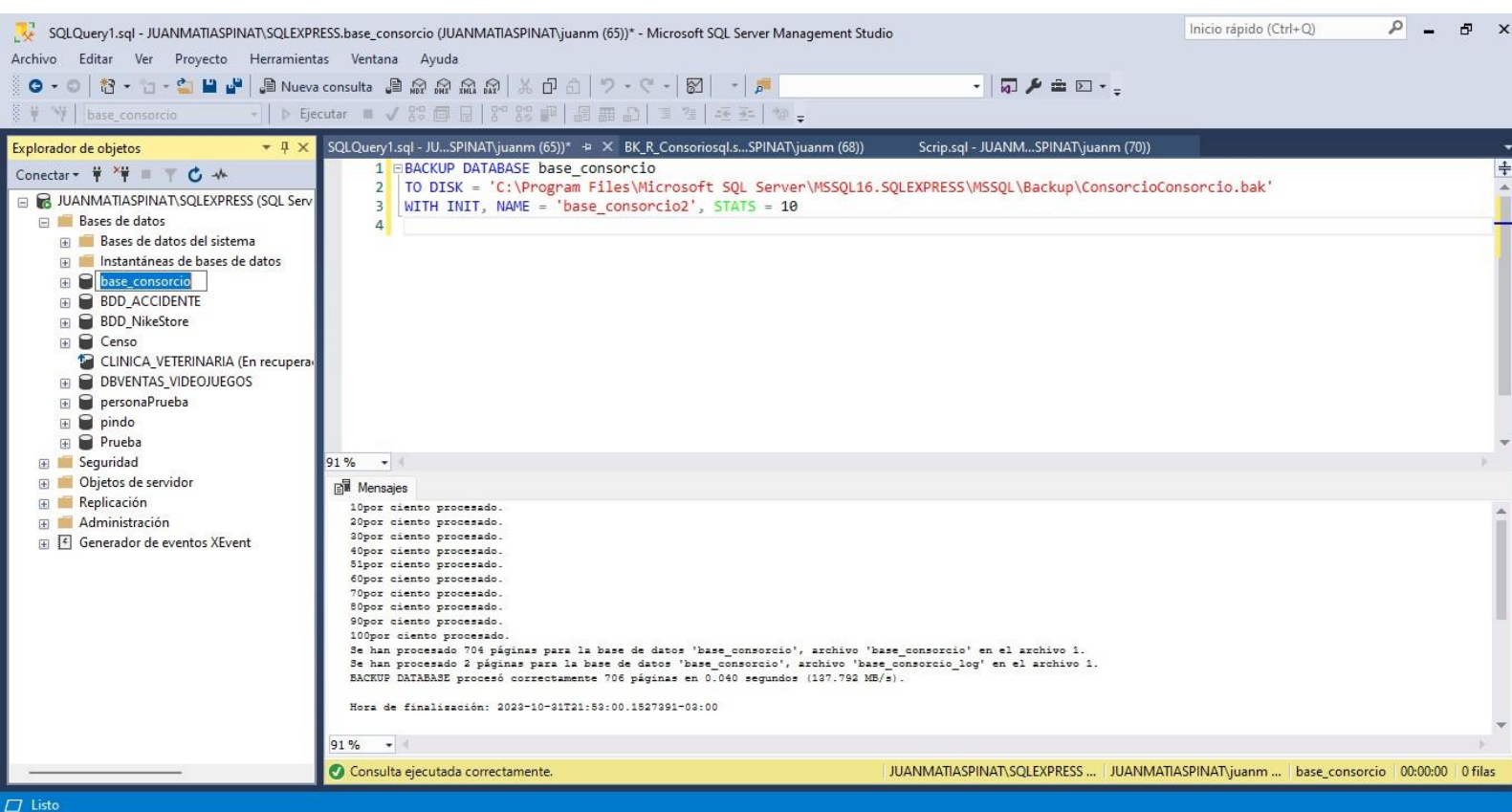



```
BACKUP DATABASE base_consortio  
TO DISK = 'C:\Program Files\Microsoft  
SQLServer\MSSQL15.MSSQLSERVER\MSSQL\Backup\Consortio\Consortio.bak'  
WITH INIT, NAME = 'base_consortio2', STATS = 10
```

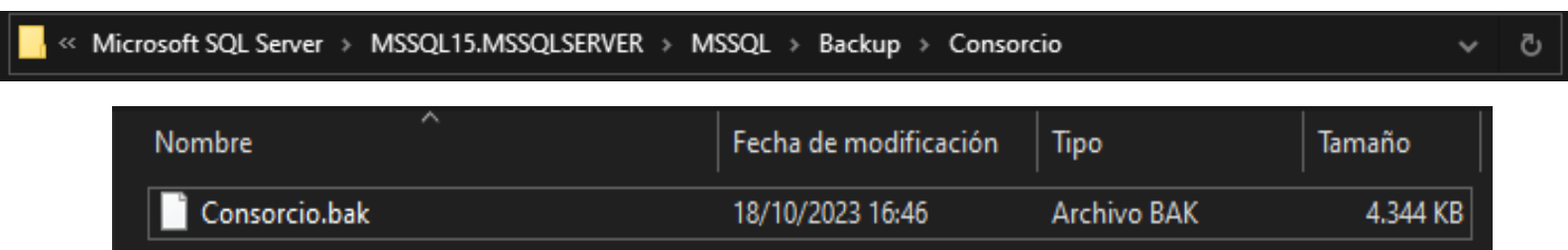
En donde nos posicionamos en ella y ejecutamos la siguiente sentencia:

Esta sentencia lo que nos está queriendo decir es que se realizara un Backup de la base de datos “base_consortio2” en la ruta seleccionada, en donde se creará un archivo, el cual contenga la base de datos completa y si hay algún archivo con el mismo nombre, lo sobrescribirá creando un nuevo archivo.

En la siguiente imagen podremos observar cómo sería la ejecución de la sentencia dentro del SQL Server.



Y también podremos ver cómo sería el archivo del Backup creado en la ruta especificada.



Al realizar el Backup de esta manera nos surgió una duda, la cual sería ¿Si queremos tener varios Backup con la fecha y la hora de creación en el nombre, como lo tendríamos que hacer? Entonces para solucionar este inconveniente investigamos un poco y más y llegamos a la siguiente sentencia.

```

/* Aquí se declara una variable llamada @Fecha con un tipo de datos VARCHAR de longitud 200. Esta variable se
usará para almacenar la fecha y hora actual en un formato específico.*/
DECLARE @Fecha VARCHAR(200)

/*Aquí se asigna un valor a la variable @Fecha la cual sera el resultado de la siguiente operacion, la cual convierte a VARCHAR la fecha
actual*/
SET @Fecha = REPLACE(CONVERT(VARCHAR,GETDATE()),100), ':', '.')

/*Se declara una segunda variable llamada @DireccionCarpeta con un tipo de datos VARCHAR de longitud 400. Esta variable se usará para
almacenar la ruta del archivo de respaldo de la base de datos.*/
Declare @DireccionCarpeta Varchar(400)



/*En esta línea,se asigna el valor que tomara la variable @DireccionCarpeta la cual sera la ruta del archivo de respaldo para la base de datos*/
Set @DireccionCarpeta = 'C:\Program Files\Microsoft SQL Server\MSSQL15.MSSQLSERVER\MSSQL\Backup\Consortio\Consortio ' + @Fecha + '.bak'

/*Esta es la sentencia donde se ejecuta el BACKUP de la base de datos llamada base_consortio en el archivo especificado
por la variable @DireccionCarpeta*/
BACKUP DATABASE base_consortio
TO DISK = @DireccionCarpeta

/*WITH INIT: Indica que se está realizando una copia de seguridad inicial. Si ya existen copias de seguridad en el archivo de respaldo,
esta opción las sobrescribe.
NAME = 'base_consortio': Aquí se asigna un nombre a la copia de seguridad.
STATS = 10: Muestra información de progreso en la operación de copia de seguridad cada vez que se completen 10 porcentajes de la operación.*/
WITH INIT, NAME = 'base_consortio', STATS = 10

```

Con esto logramos que al realizar un Backup nos guarde con la fecha y la hora actual del sistema.

<div> <div>Microsoft SQL Server</div> <div>MSSQL15.MSSQLSERVER</div> <div>MSSQL</div> <div>Backup</div> <div>Consortio</div> </div>			
Nombre	Fecha de modificación	Tipo	Tamaño
 Consortio Oct 18 2023 6.38PM.bak	18/10/2023 18:38	Archivo BAK	4.792 KB
 Consortio.bak	18/10/2023 18:35	Archivo BAK	4.792 KB

Por último, veremos cómo realizar el Restore de la Copia de seguridad (Backup) realizada, dicha sentencia se puede ver a continuación.

```

/*En esta línea, se declara una variable llamada @NombreDataBase y se le asigna el valor 'base_consortio'. Esta variable se utilizará para especificar
el nombre de la base de datos que se va a restaurar.*/
DECLARE @NombreDataBase VARCHAR(200) = 'base_consortio';

/*Se declara otra variable llamada @Ubicacion que se utilizará para almacenar la ubicación (ruta del archivo) de la última copia de seguridad realizada
para la base de datos base_consortio. Inicialmente, esta variable está vacía.*/
DECLARE @Ubicacion NVARCHAR(128);

/*Esta consulta se utiliza para recuperar la ubicación del archivo de la ***última copia de seguridad de la base de datos base_consortio.*** */

/*Se selecciona la columna physical_device_name de la tabla backupmediafamily, que contiene la ubicación del archivo de copia de seguridad.
Se filtra la consulta para que solo incluya registros donde el nombre de la base de datos (b.database_name) coincide con el valor almacenado en
@NombreDataBase (en este caso, 'base_consortio') y además que solo incluyan los archivos con la extensión '.bak'.
Los resultados se ordenan por la fecha de inicio de la copia de seguridad (b.backup_start_date) en orden descendente (del más reciente al más antiguo).
La cláusula TOP 1 se utiliza para seleccionar solo el primer registro (el más reciente) que cumple con las condiciones, y su valor se asigna a la
variable @Ubicacion.*/
SELECT top 1 @Ubicacion = m.physical_device_name
FROM msdb.dbo.backupset AS b
JOIN msdb.dbo.backupmediafamily AS m ON b.media_set_id = m.media_set_id
WHERE b.database_name = @NombreDataBase
/*AND RIGHT(m.physical_device_name, 4) = '.bak': Esta línea agrega otra condición al filtro. Utiliza la función RIGHT() para extraer los últimos cuatro
caracteres de la columna physical_device_name en la tabla backupmediafamily. Luego, compara esos cuatro caracteres con '.bak'. Esto se hace para asegurarse
de que la ubicación física del archivo de copia de seguridad termine con '.bak', lo que indica que es un archivo de copia de seguridad con la extensión
'.bak'.*/
AND RIGHT(m.physical_device_name, 4) = '.bak' --
ORDER BY b.backup_start_date DESC;

/*En esta línea, se ejecuta la sentencia de restauración de la base de datos base_consortio. La restauración se realiza desde el archivo de
copia de seguridad cuya ubicación se determinó en la consulta anterior y se almacena en la variable @Ubicacion. */
RESTORE DATABASE base_consortio

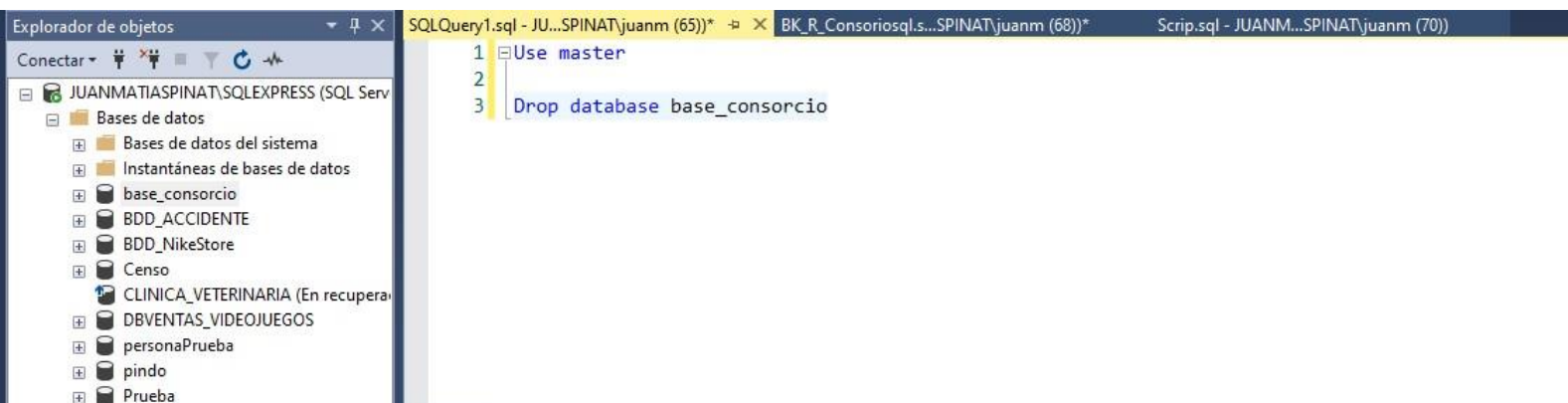
/* Especifica que la restauración se realiza desde el archivo de copia de seguridad que se encuentra en la ubicación almacenada en @Ubicacion.*/
FROM DISK = @Ubicacion

/*Esta opción permite reemplazar la base de datos existente con la restaurada.*/
/* Esta opción coloca la base de datos en estado de recuperación, lo que significa que la base de datos estará disponible para su uso después
de esta operación.*/
WITH REPLACE, RECOVERY;

```

De esta manera haríamos el Restore de nuestro Backup en caso de que haya alguna complicación u error.

Por ejemplo: En el caso hipotético que borráramos accidentalmente la base de datos



The screenshot shows the SQL Server Enterprise Manager interface on the left, displaying the 'Bases de datos' (Databases) folder under 'JUANMATIASPINAT\SQLEXPRESS (SQL Serv)'. The 'base_consortio' database is highlighted. On the right, the 'SQLQuery1.sql' window shows the following T-SQL commands:

```
1 Use master
2
3 Drop database base_consortio
```

The 'Mensajes' (Messages) pane at the bottom indicates that the commands were executed successfully:

Los comandos se han completado correctamente.
Hora de finalización: 2023-10-31T22:04:12.2519950-03:00

Mediante la ejecución de la sentencia Restore podremos recuperar la base a partir del último Backup realizado.

The screenshot shows the SQL Server Enterprise Manager interface on the left, displaying the 'base_consortio' database under 'Bases de datos'. On the right, the 'SQLQuery1.sql' window shows the following T-SQL commands for restoring the database:

```
1 DECLARE @NombreDataBase VARCHAR(200) = 'base_consortio';
2
3
4 DECLARE @Ubicacion NVARCHAR(128);
5
6 SELECT TOP 1 @Ubicacion = m.physical_device_name
7 FROM msdb.dbo.backupset AS b
8 JOIN msdb.dbo.backupmediafamily AS m ON b.media_set_id = m.media_set_id
9 WHERE b.database_name = @NombreDataBase
10 AND RIGHT(m.physical_device_name, 4) = '.bak'
11 ORDER BY b.backup_start_date ASC;
12
13
14
15 RESTORE DATABASE base_consortio
16 FROM DISK = @Ubicacion
17 WITH REPLACE, RECOVERY;
```

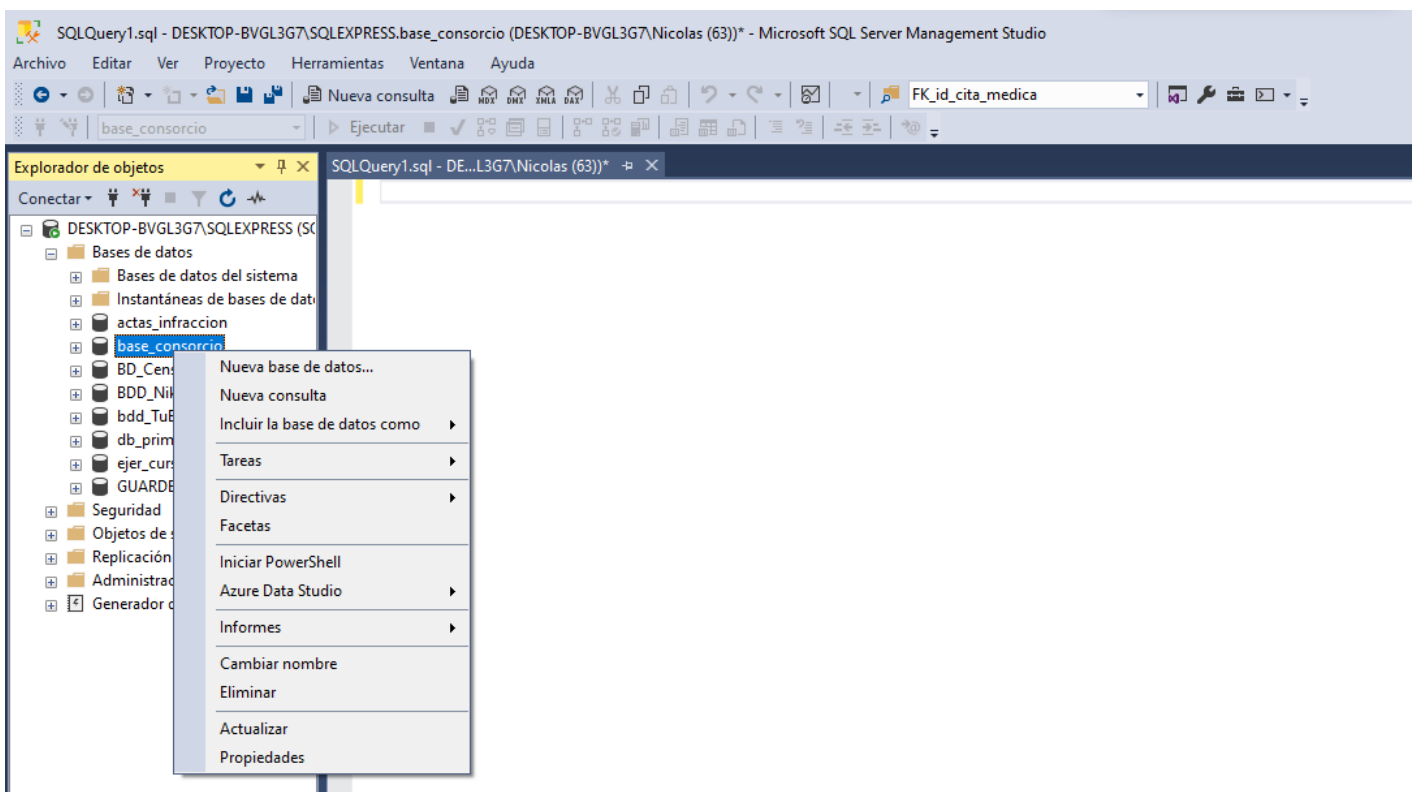
The 'Mensajes' (Messages) pane at the bottom indicates that the restore operation was successful:

Se han procesado 696 páginas para la base de datos 'base_consortio', archivo 'base_consortio' en el archivo 1.
Se han procesado 2 páginas para la base de datos 'base_consortio', archivo 'base_consortio_log' en el archivo 1.
RESTORE DATABASE procesó correctamente 698 páginas en 0.082 segundos (66.453 MB/s).
Hora de finalización: 2023-10-31T22:25:44.7059962-03:00

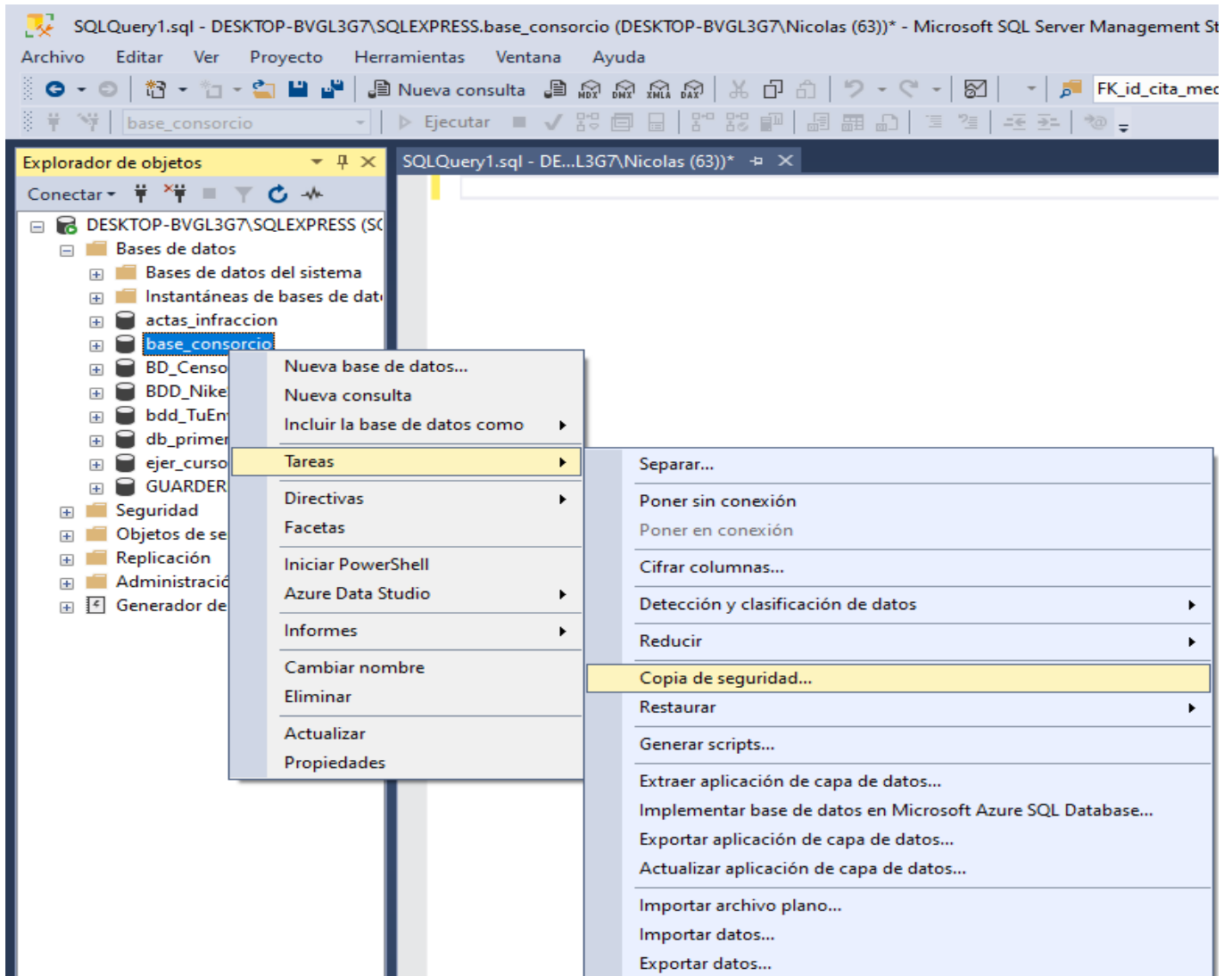
Por último, vamos a explicar los pasos para realizar un Backup en línea:

Para realizar un Backup en línea desde Microsoft SQL server, debemos tener una cuenta en Microsoft Azure, una vez creada ésta, debemos registrar un contenedor donde vamos a guardar nuestro respaldo de la base de datos.

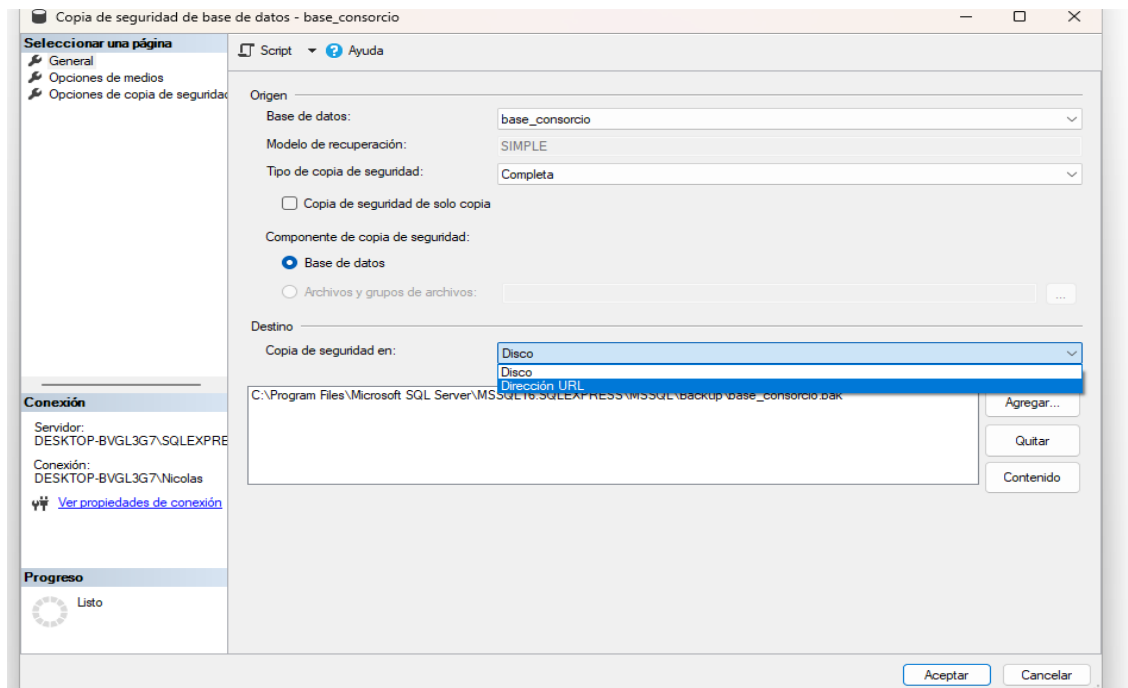
Luego nos vamos al Management Studio, seleccionamos en el explorador de objetos la base de datos a la cual vamos a realizarle el Backup (En nuestro caso la base de datos es “base_consortio”) con clic derecho:



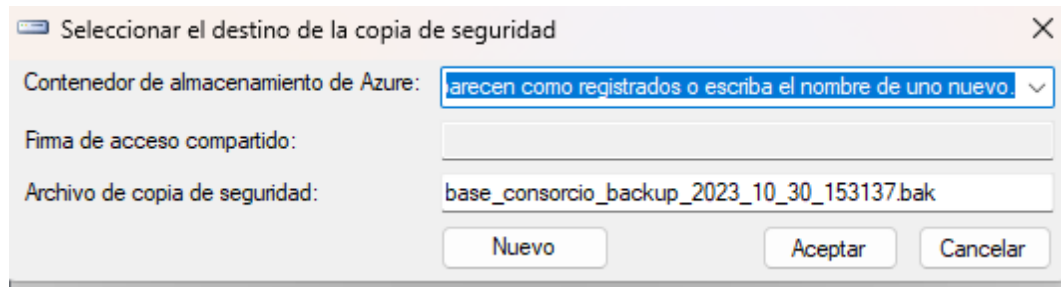
Luego, seleccionamos la opción “Tareas”, después seleccionamos “Copia de seguridad”.



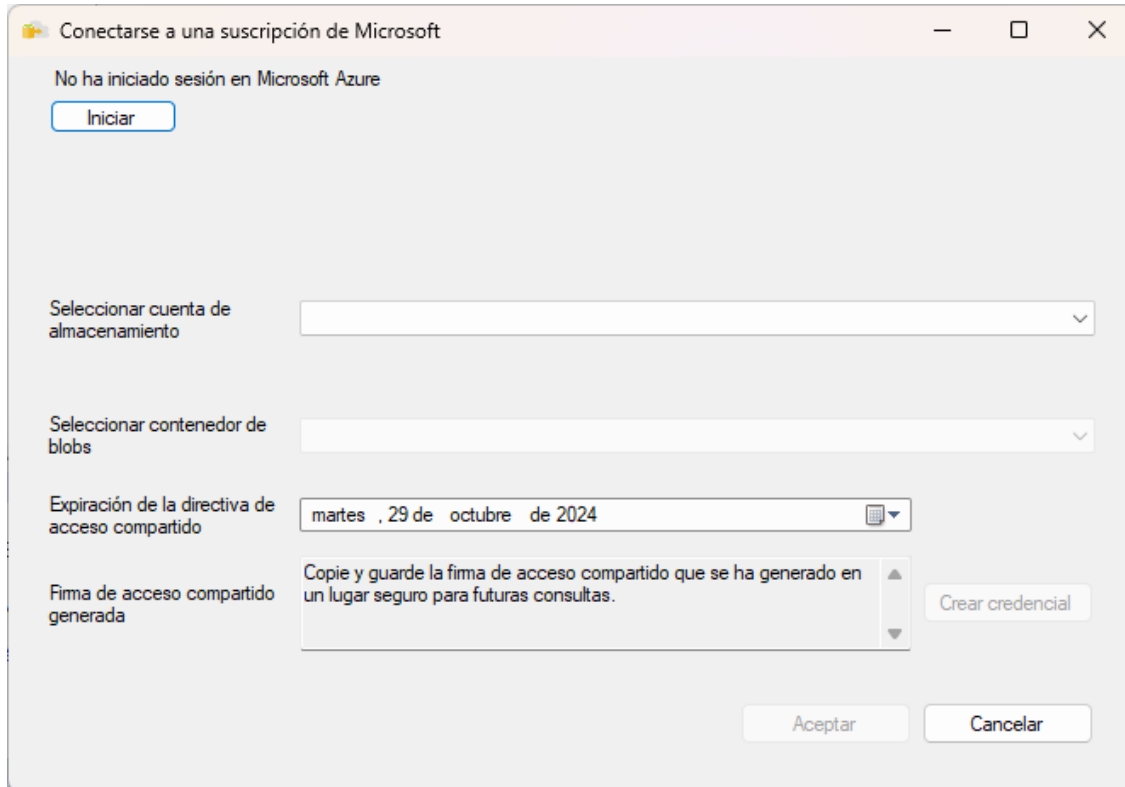
Luego nos mostrará el asistente de copias de seguridad:



Aquí, en “Destino” colocamos “Dirección URL”, luego hacemos clic en “Agregar” y nos va a mostrar lo siguiente:



Aquí damos clic en la opción “Nuevo” y nos va a mostrar lo siguiente:



Conectarse a una suscripción de Microsoft

No ha iniciado sesión en Microsoft Azure

Iniciar

Seleccionar cuenta de almacenamiento

Seleccionar contenedor de blobs

Expiración de la directiva de acceso compartido: martes, 29 de octubre de 2024

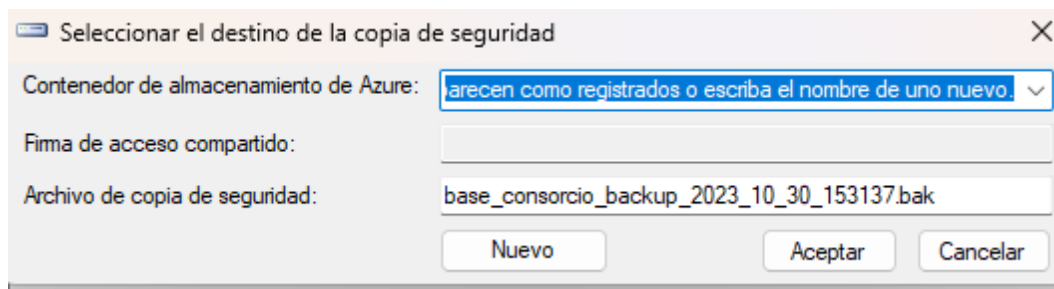
Firma de acceso compartido generada: Copie y guarde la firma de acceso compartido que se ha generado en un lugar seguro para futuras consultas.

Crear credencial

Aceptar Cancelar

Aquí debemos iniciar sesión en Microsoft Azure, seleccionar una cuenta de almacenamiento y también seleccionar un contenedor de blobs.

Una vez realizado esto, debemos volver a seleccionar el destino de la copia de seguridad, colocamos el contenedor de almacenamiento de Azure y le damos en “Aceptar”.



Seleccionar el destino de la copia de seguridad

Contenedor de almacenamiento de Azure: aparecen como registrados o escriba el nombre de uno nuevo.

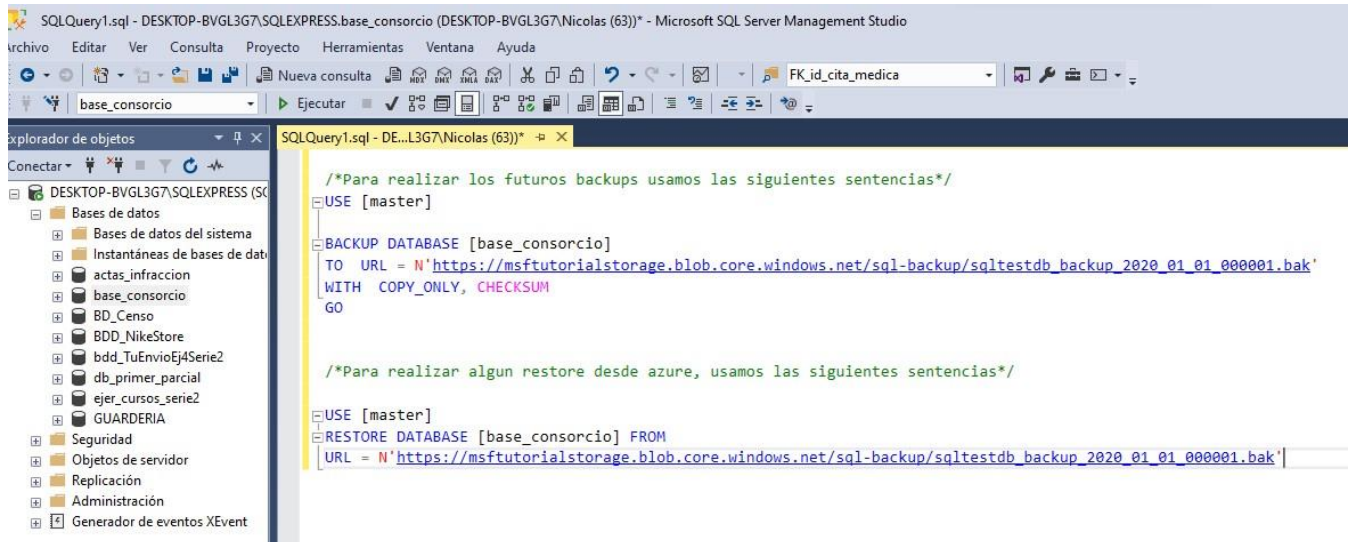
Firma de acceso compartido:

Archivo de copia de seguridad: base_consortio_backup_2023_10_30_153137.bak

Nuevo Aceptar Cancelar

Una vez realizado esto, la base de datos seleccionada (base_consortio) ya tendrá su Backup en línea en Azure.

Una vez sincronizada la cuenta de Microsoft Azure con SQL server, podremos realizar los siguientes Backups y Restores en línea usando Transact-SQL:



The screenshot shows the Microsoft SQL Server Management Studio interface. The title bar indicates the connection is to 'DESKTOP-BVGL3G7\SQLEXPRESS.base_consortio (DESKTOP-BVGL3G7\Nicolas (63))'. The 'Explorador de objetos' (Object Explorer) on the left shows the 'base_consortio' database selected under the 'Bases de datos' folder. The main query window displays the following Transact-SQL script:

```
/*Para realizar los futuros backups usamos las siguientes sentencias*/
USE [master]

BACKUP DATABASE [base_consortio]
TO URL = N'https://msftutorialstorage.blob.core.windows.net/sql-backup/sqltestdb_backup_2020_01_01_000001.bak'
WITH COPY_ONLY, CHECKSUM
GO

/*Para realizar algun restore desde azure, usamos las siguientes sentencias*/

USE [master]
RESTORE DATABASE [base_consortio] FROM
URL = N'https://msftutorialstorage.blob.core.windows.net/sql-backup/sqltestdb_backup_2020_01_01_000001.bak'
```

Integración de temas

Procedimientos y funciones almacenadas

Introducción

En el dinámico mundo de la gestión de bases de datos, la eficacia y la eficiencia son elementos esenciales que impulsan el rendimiento de cualquier proyecto. El uso de las herramientas adecuadas puede marcar una diferencia significativa en la optimización de las operaciones de manipulación de datos.

Este estudio se adentra en el universo de las bases de datos y se enfoca en dos herramientas fundamentales: los procedimientos almacenados y las funciones definidas por el usuario en SQL Server. Estos elementos desempeñan un papel crucial en la simplificación y eficiencia de las operaciones relacionadas con bases de datos.

Este estudio proporcionará a profesionales de bases de datos, desarrolladores y administradores, el conocimiento básico sobre cómo utilizar procedimientos almacenados y funciones definidas por el usuario en sus proyectos. Con ello, se espera mejorar la eficiencia y la productividad en la gestión de datos.

Marco conceptual

Un procedimiento almacenado de SQL Server es un grupo de una o varias instrucciones Transact-SQL o una referencia a un método de Common Runtime Language (CLR) de Microsoft .NET Framework.

Los procedimientos se asemejan a las construcciones de otros lenguajes de programación, porque pueden:

Aceptar parámetros de entrada y devolver varios valores en forma de parámetros de salida al programa que realiza la llamada.

Contener instrucciones de programación que realicen operaciones en la base de datos. Entre otras, pueden contener llamadas a otros procedimientos.

- Devolver un valor de estado a un programa que realiza una llamada para indicar si la operación se ha realizado correctamente o se han producido errores, y el motivo de estos.

Una función definida por el usuario es una rutina de Transact-SQL o Common Language Runtime (CLR) que acepta parámetros, realiza una acción, como un cálculo complejo, y devuelve el resultado de esa acción como valor.

El valor devuelto puede ser un valor escalar (único) o una tabla. Utilice esta instrucción para crear una rutina reutilizable que se pueda utilizar de estas formas:

- En instrucciones Transact-SQL como SELECT
- En las aplicaciones que llaman a la función
- En la definición de otra función definida por el usuario
- Para parametrizar una vista o mejorar la funcionalidad de una vista indizada
- Para definir una columna en una tabla
- Para definir una restricción CHECK en una columna
- Para reemplazar un procedimiento almacenado
- Usar una función insertada como predicado de filtro de la directiva de seguridad

METODOLOGÍA

CREACIÓN DE LA BASE DE DATOS DE EJEMPLO

En una etapa inicial, se generó una base de datos de ejemplo utilizando los scripts proporcionados por el equipo docente. Esta base de datos sirvió como punto de partida para nuestros estudios y experimentos.

DESARROLLO DE NUEVOS SCRIPTS

Estos scripts representaron la aplicación de los conocimientos obtenidos durante la investigación y constituyeron una parte esencial de nuestro trabajo.

PRUEBAS Y EVALUACION DE SCRIPTS

Los nuevos scripts desarrollados fueron sometidos a pruebas para evaluar su rendimiento y eficacia.

Durante este proceso, se identificaron las ventajas de utilizar los objetos de la investigación en el contexto de la base de datos.

HERRAMIENTAS

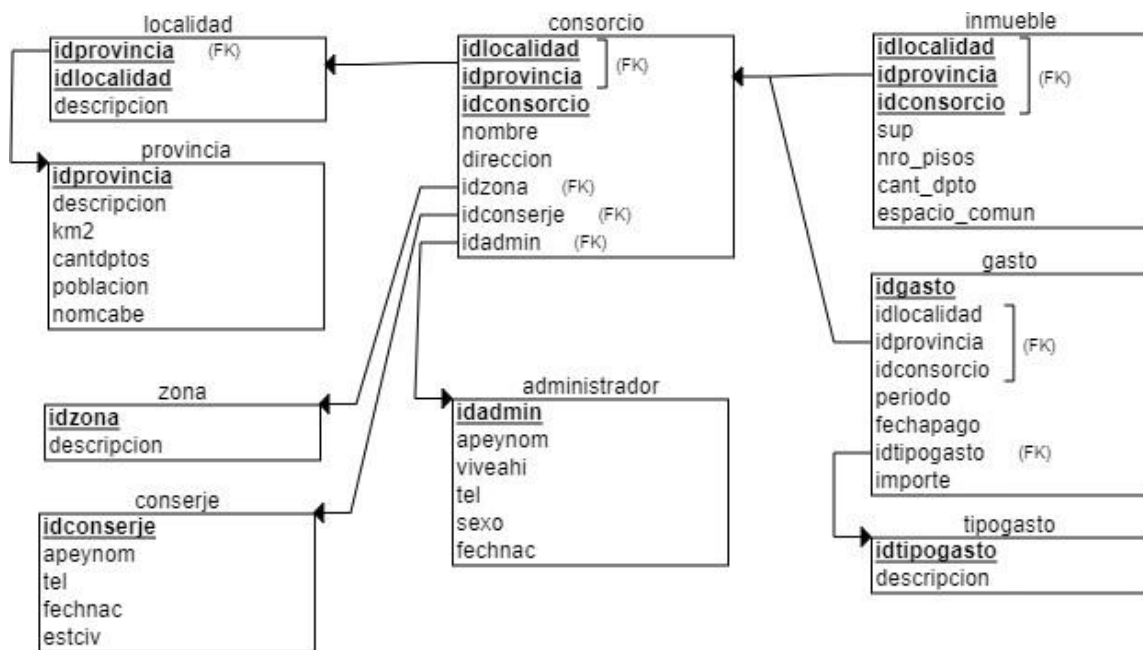
En lo que respecta a las herramientas utilizadas en este estudio, se optó por emplear el motor de base de datos SQL Server 16 y el SQL Server Management Studio 18. Estas herramientas proporcionaron el entorno adecuado para llevar a cabo las pruebas y experimentos de manera eficiente.

Además, cabe destacar que los conocimientos necesarios para llevar a cabo esta investigación se obtuvieron a partir de la documentación oficial de SQL Server proporcionada por Microsoft. Esta documentación sirvió como fuente de referencia fundamental, garantizando la correcta aplicación de los conceptos y procedimientos relacionados con SQL Server.

DESARROLLO

En este experimento iniciaremos con una base de datos que mantiene la información relacionada a consorcios, administradores y conserjes de edificios.

MODELO LÓGICO



INSERTANDO DATOS

En la siguiente imagen se observa la forma más simple de insertar datos en la tabla administrador utilizando la sentencia INSERT:

```
INSERT INTO administrador (apeynom, viveahi, tel, sexo, fechnac)
VALUES ('MARIELA CORREA', 'N', '3794123456', 'F', '1990-10-03')
```

La ejecución de la misma inserta los datos de forma exitosa. ¿Pero qué sucede cuando se viola alguna restricción de la tabla?

```
INSERT INTO administrador (apeynom, viveahi, tel, sexo, fechnac)
VALUES ('MARIELA CORREA', 'G', '3794123456', 'F', '1990-10-03')
```

Se obtiene el siguiente error:

```
Messages
Msg 547, Level 16, State 0, Line 3
The INSERT statement conflicted with the CHECK constraint "CK_habitante_viveahi". The conflict occurred in database "base_consortio", table "dbo.administrador", column "viveahi".
The statement has been terminated.

Completion time: 2023-10-28T19:12:46.3705245-03:00
```

En el script anterior, se está violando la restricción del campo vive ahí, eso hace que la inserción falle, y eso ocasiona el campo idadmin se incremente de todas formas, ya que posee la propiedad de ser auto incremental. Esto puede prevenirse con el manejo de errores. Observe el siguiente script:

```
BEGIN TRY
    BEGIN TRAN
    INSERT INTO administrador (apeynom, viveahi, tel, sexo, fechnac)
    VALUES ('MARIELA CORREA', 'G', '3794123456', 'F', '1990-10-03')
    COMMIT TRAN
END TRY
BEGIN CATCH
    DECLARE @mensaje varchar(200)
    SET @mensaje = CASE
        WHEN ERROR_MESSAGE() LIKE '%CK_habitante_viveahi%' THEN 'Valor incorrecto para el campo viveahi. Los valores posibles son S y N.'
        ELSE ERROR_MESSAGE()
    END
    ROLLBACK TRAN;
    THROW 51000, @mensaje, 1;
END CATCH
```

Aquí se ve cómo puede prevenirse el incremento del campo auto incremental con el uso de transacciones y de bloques TRY-CATCH. También se añadió la personalización del mensaje de error con el uso de la instrucción THROW. Se obtuvo el siguiente resultado:


```
(0 rows affected)
Msg 51000, Level 16, State 1, Line 16
Valor incorrecto para el campo viveahi. Los valores posibles son S y N.

Completion time: 2023-10-28T19:27:08.8894239-03:00
```

Toda esta lógica utilizada tendría que repetirse cada vez que el usuario necesite insertar un registro, bien podría estar dentro de un procedimiento almacenado. Esto se vería de la siguiente forma:

```
CREATE PROCEDURE [InsertarAdministrador] (
    @apeynom varchar(50) = null,
    @viveahi varchar(1) = null,
    @tel varchar(20) = null,
    @sexo varchar(1) = null,
    @fechnac datetime = null
)
AS
BEGIN
    BEGIN TRY
        BEGIN TRAN
            INSERT INTO administrador (apeynom, viveahi, tel, sexo, fechnac)
            VALUES (@apeynom, @viveahi, @tel, @sexo, @fechnac)
            COMMIT TRAN
        END TRY
        BEGIN CATCH
            DECLARE @mensaje varchar(200)
            SET @mensaje = CASE
                WHEN ERROR_MESSAGE() LIKE '%CK_habitante_viveahi%' THEN 'Valor incorrecto para el campo viveahi. Los valores posibles son S y N.'
                ELSE ERROR_MESSAGE()
            END
            ROLLBACK TRAN;
            THROW 51000, @mensaje, 1;
        END CATCH
    END
```

O bien podría utilizar el tipo de parámetros OUT para indicar si la inserción fue exitosa. Observe:

```
CREATE PROCEDURE [InsertarAdministrador] (
    @apeynom varchar(50) = null,
    @viveahi varchar(1) = null,
    @tel varchar(20) = null,
    @sexo varchar(1) = null,
    @fechnac datetime = null,
    @exito bit OUT,
    @error varchar(200) OUT
)
AS
BEGIN
    SET @exito = 1
    BEGIN TRY
        BEGIN TRAN
            INSERT INTO administrador (apeynom, viveahi, tel, sexo, fechnac)
            VALUES (@apeynom, @viveahi, @tel, @sexo, @fechnac)
            COMMIT TRAN
        END TRY
        BEGIN CATCH
            SET @error = CASE
                WHEN ERROR_MESSAGE() LIKE '%CK_habitante_viveahi%' THEN 'Valor incorrecto para el campo viveahi. Los valores posibles son S y N.'
                ELSE ERROR_MESSAGE()
            END
            ROLLBACK TRAN;
            SET @exito = 0
        END CATCH
    END
```

La siguiente imagen demuestra el comportamiento ante una inserción fallida:

```
114 DECLARE @insercionExitosa bit
115 DECLARE @mensajeError varchar(200)
116 EXEC InsertarAdministrador 'MARIELA CORREA', 'G', '3794123456', 'F', '1990-10-03', @insercionExitosa OUT, @mensajeError OUT
117 SELECT @insercionExitosa
118 SELECT @mensajeError
119
```

Y este script que sigue una inserción exitosa:

```
113 DECLARE @insercionExitosa bit
114 DECLARE @mensajeError varchar(200)
115 EXEC InsertarAdministrador 'MARIELA CORREA', 'S', '3794123456', 'F', '1990-10-03', @insercionExitosa OUT, @mensajeError OUT
116 SELECT @insercionExitosa
117 SELECT @mensajeError
118
119
```

MODIFICANDO DATOS

Veamos qué sucede cuando lo que se desea es modificar datos de una tabla. Observe el siguiente script:

```
UPDATE administrador
SET viveahi='S'
WHERE idadmin=174
```

De forma análoga a lo realizado en la inserción de los datos, se puede aplicar lógica para el manejo de errores aplicado el uso de subrutinas.

```

CREATE PROCEDURE [CambiarResidenciaAdministrador] (
    @idadmin int = null,
    @viveahi varchar(1) = null,
    @exito bit OUT,
    @error varchar(200) OUT
)
AS
BEGIN
    SET @exito = 1
    BEGIN TRY
        BEGIN TRAN
        UPDATE administrador
        SET viveahi=@viveahi
        WHERE idadmin=@idadmin
        COMMIT TRAN
    END TRY
    BEGIN CATCH
        SET @error = CASE
            WHEN ERROR_MESSAGE() LIKE '%CK_habitante_viveahi%' THEN 'Valor incorrecto para el campo viveahi. Los valores posibles son S y N.'
            ELSE ERROR_MESSAGE()
        END
        ROLLBACK TRAN;
        SET @exito = 0
    END CATCH
END

```

Observe el resultado al tratar de actualizar información que viola las restricciones de la tabla:

```

121 DECLARE @insercionExitosa bit
122 DECLARE @mensajeError varchar(200)
123 EXEC CambiarResidenciaAdministrador 174, 'F' , @insercionExitosa OUT, @mensajeError OUT
124 SELECT @insercionExitosa
125 SELECT @mensajeError

```

(No column name)
0

(No column name)
Valor incorrecto para el campo viveahi. Los valores posibles son S y N.

ELIMINANDO DATOS

Cuando lo que se quiere es eliminar un registro sucede lo mismo, observe la siguiente sentencia:

```

DELETE FROM administrador
WHERE idadmin=174

```

Cuando se quiere eliminar, para verificar que se elimina el registro deseado, se puede agregar una validación de la existencia del mismo:

```

CREATE PROCEDURE [EliminarAdministrador] (
    @idadmin int = null,
    @exito bit OUT,
    @error varchar(200) OUT
)
AS
BEGIN
    SET @exito = 1
    BEGIN TRY
        IF NOT EXISTS (SELECT * FROM administrador WHERE idadmin=@idadmin)
        BEGIN
            SET @exito = 0
            SET @error = 'Administrador inexistente'
            RETURN
        END
        BEGIN TRAN
        DELETE FROM administrador
        WHERE idadmin=@idadmin
        COMMIT TRAN
    END TRY
    BEGIN CATCH
        ROLLBACK TRAN;
        SET @error = ERROR_MESSAGE()
        SET @exito = 0
    END CATCH
END

```

En el caso que no exista el registro, se puede informar mediante los parametros OUT:

```

126
127 DECLARE @insercionExitosa bit
128 DECLARE @mensajeError varchar(200)
129 EXEC [EliminarAdministrador] 174, @insercionExitosa OUT, @mensajeError OUT
130 SELECT @insercionExitosa
131 SELECT @mensajeError

```

Results	Messages
(No column name)	
0	
(No column name)	
	Administrador inexistente

CONSULTADO DATOS

Cuando se consultan datos sucede lo siguiente:

```
3 SELECT
4     [Nombre y Apellido]    = apeynom,
5     [Telefono]             = tel,
6     [Sexo]                 = IIF(sexo='F', 'Femenino', 'Masculino'),
7     [Fecha de Nacimiento] = FORMAT(CONVERT(date, fechnac), 'dd-MM-yyyy'),
8     [Edad]                 = DATEDIFF(YEAR, fechnac, GETDATE())
9 FROM administrador
```

Nombre y Apellido	Telefono	Sexo	Fecha de Nacimiento	Edad
Perez Juan Manuel	3794112233	Masculino	18-02-1985	38
BASUALDO DELMIRA	3624231689	Femenino	09-10-1980	43
SEGOVIA ALEJANDRO H.	3624232689	Masculino	02-06-1974	49
ROMERO ELEUTERIO	3624233689	Masculino	19-08-1972	51
NAHMIAS DE K. NIDIA	3624234689	Femenino	28-11-1971	52
CORREA DE M. MARIA G.	3624235689	Femenino	16-01-1990	33
NAHMIAS JOSE	3624236689	Masculino	02-09-1974	49
NAHMIAS DE R. REBECA J.	3624237689	Femenino	07-03-1989	34
LOVATO CERENTTINI ISABEL	3624238689	Femenino	15-10-1973	50
GOMEZ MATIAS GABRIEL	3624239689	Masculino	20-03-1974	49
CORREA HUGO E.	3624231689	Masculino	11-08-1993	30
MACHUCA CEFERINA	3624232689	Femenino	16-09-1991	32
CARDOZO MAXIMA	3624233689	Femenino	07-11-1988	35
RODRIGUEZ MARTIN J.	3624234689	Masculino	09-12-1985	38
SOTELO GERTRUDIS	3624235689	Femenino	12-11-1990	33
AYALA FLORENTINA	3624236689	Femenino	27-04-1974	49

Esto bien se puede usar dentro de un procedimiento almacenado para poder ver los datos de los administradores de una forma deseada.

```
CREATE PROCEDURE [VerAdministradores]
AS
BEGIN
SELECT
    [Nombre y Apellido]      = apeynom,
    [Telefono]               = tel,
    [Sexo]                   = IIF(sexo='F', 'Femenino', 'Masculino'),
    [Fecha de Nacimiento]    = FORMAT(CONVERT(date, fechnac), 'dd-MM-yyyy'),
    [Edad]                   = DATEDIFF(YEAR, fechnac, GETDATE())
FROM administrador
END
```

Pero nos encontramos ante un problema si queremos saber la edad de algun administrador por fuera de este procedimiento. Por ello utilizaremos funciones definidas por el usuario. Esto nos da la habilidad de si hay algun tipo de error en el cálculo, solo haya que modificarse en un solo lugar, la definición de dicha función.

```
CREATE FUNCTION [CalcularEdad] (  
    @FechaNacimiento date  
)  
RETURNS int  
AS  
BEGIN  
    RETURN DATEDIFF(YEAR, @FechaNacimiento, GETDATE())  
END
```

Quedando así el procedimiento para ver los datos de los administradores:

```
CREATE PROCEDURE [VerAdministradores]  
AS  
BEGIN  
SELECT  
    [Nombre y Apellido]      = apeynom,  
    [Telefono]              = tel,  
    [Sexo]                  = IIF(sexo='F', 'Femenino', 'Masculino'),  
    [Fecha de Nacimiento]   = FORMAT(CONVERT(date, fechnac), 'dd-MM-yyyy'),  
    [Edad]                  = dbo.CalcularEdad(fechnac)  
FROM administrador  
END
```

La desventaja de utilizar procedimientos para mostrar datos, es que no se pueden aplicar filtros a los datos, como por ejemplo, solo ver los datos de los administradores mayores a 35 años. Pero esto se ve resuelto con otro tipo de funciones llamadas funciones de tabla.

```

CREATE FUNCTION [FuncionVerAdministradores] (
    @edadMinima int
)
RETURNS TABLE
AS
RETURN (
SELECT
    [Nombre y Apellido]      = apeynom,
    [Telefono]               = tel,
    [Sexo]                   = IIF(sexo='F', 'Femenino', 'Masculino'),
    [Fecha de Nacimiento]    = FORMAT(CONVERT(date, fechnac), 'dd-MM-yyyy'),
    [Edad]                   = dbo.CalcularEdad(fechnac)
FROM administrador
WHERE dbo.CalcularEdad(fechnac) >= @edadMinima
)

```

Observe:

140
141 SELECT * FROM FuncionVerAdministradores(35)
142

%

Results Messages

	Nombre y Apellido	Telefono	Sexo	Fecha de Nacimiento	Edad
1	Perez Juan Manuel	3794112233	Masculino	18-02-1985	38
2	BASUALDO DELMIRA	3624231689	Femenino	09-10-1980	43
3	SEGOVIA ALEJANDRO H.	3624232689	Masculino	02-06-1974	49
4	ROMERO ELEUTERIO	3624233689	Masculino	19-08-1972	51
5	NAHMIA DE K. NIDIA	3624234689	Femenino	28-11-1971	52
6	NAHMIA JOSE	3624236689	Masculino	02-09-1974	49
7	LOVATO CERENTINI ISABEL	3624238689	Femenino	15-10-1973	50
8	GOMEZ MATIAS GABRIEL	3624239689	Masculino	20-03-1974	49
9	CARDOZO MAXIMA	3624233689	Femenino	07-11-1988	35
10	RODRIGUEZ MARTIN J.	3624234689	Masculino	09-12-1985	38
11	AYALA FLORENTINA	3624236689	Femenino	27-04-1974	49
12	RATTI JUAN E.	3614238689	Masculino	07-08-1967	56
13	ROCH GRACIELA	3614238689	Femenino	24-05-1969	54
14	VILLALBA #AMANDU, ALEJAND	3614231689	Masculino	27-04-1981	42
15	CACERES ROSA	3614232689	Femenino	08-10-1980	43
16	PALACIOS DE F. OLGA	3624233689	Femenino	17-10-1966	57
17	CONTRERAS JUAN	3624231689	Femenino	11-02-1985	38

Vea que la función puede ser utilizada como si fuese una tabla nueva, y así poder aplicar filtros desde la llamada:

140
141
142
143

SELECT * FROM FuncionVerAdministradores(35)
WHERE Sexo='Masculino'

9 %

Results

Messages

	Nombre y Apellido	Telefono	Sexo	Fecha de Nacimiento	Edad
1	Perez Juan Manuel	3794112233	Masculino	18-02-1985	38
2	SEGOVIA ALEJANDRO H.	3624232689	Masculino	02-06-1974	49
3	ROMERO ELEUTERIO	3624233689	Masculino	19-08-1972	51
4	NAHMIAS JOSE	3624236689	Masculino	02-09-1974	49
5	GOMEZ MATIAS GABRIEL	3624239689	Masculino	20-03-1974	49
6	RODRIGUEZ MARTIN J.	3624234689	Masculino	09-12-1985	38
7	RATTI JUAN E.	3614238689	Masculino	07-08-1967	56
8	VILLALBA AMANDU, ALEJAND	3614231689	Masculino	27-04-1981	42
9	SAUCEDO LORENZO	3614235689	Masculino	19-04-1967	56

Triggers

Introducción

El propósito fundamental de esta investigación es proporcionar una comprensión en profundidad de los triggers en bases de datos y su relevancia en la gestión de datos. Los triggers son objetos esenciales en cualquier Sistema de Gestión de Bases de Datos (SGBD) y consisten en un conjunto de reglas predefinidas que se asocian a tablas específicas. Estas reglas se desencadenan automáticamente en la base de datos en respuesta a eventos particulares, como inserciones, actualizaciones o eliminaciones de registros. En otras palabras, los triggers automatizan acciones específicas en las tablas de la base de datos, lo que ahorra tiempo y mejora la eficiencia en la gestión de datos.

Los triggers, o disparadores, son scripts en lenguaje SQL que se utilizan ampliamente en sistemas de bases de datos como MySQL, PostgreSQL y, en este caso, SQL Server. Su función principal es optimizar la gestión de bases de datos al permitir que muchas operaciones se realicen de manera automática, eliminando la necesidad de intervención humana. Además de su funcionalidad automatizada, los triggers también desempeñan un papel fundamental en la mejora de la seguridad y la integridad de los datos. Esto se logra a través de la programación de restricciones y verificaciones que minimizan errores y mantienen la sincronización de la información.

Por lo tanto, esta investigación se propone demostrar y enseñar el propósito y el uso de los triggers en un motor de base de datos. Exploraremos su importancia en la operatividad de una base de datos, abordando temas como los tipos de triggers, su creación, modificación y eliminación. Nuestro enfoque se centrará en el motor de base de datos SQL Server, lo que

permitirá a los lectores obtener una comprensión sólida de esta tecnología fundamental en la gestión de datos.

Marco Conceptual o Referencial

Concepto de Triggers (Disparadores)

Un trigger, también conocido como disparador en español, es un conjunto de sentencias SQL que se ejecutan de forma automática cuando se produce un evento que modifica una tabla. Estos eventos se relacionan con la manipulación de datos almacenados, es decir, cuando se ejecutan sentencias INSERT, UPDATE o DELETE. Lo que hace que los triggers sean interesantes es su capacidad para programarse de manera que se ejecuten antes o después de estas operaciones.

Para ilustrar este concepto, consideremos un ejemplo: Supongamos que tenemos una tabla de inventario en una base de datos y deseamos realizar un seguimiento de los productos agotados. Podríamos crear un trigger que se ejecute automáticamente después de una operación de eliminación (DELETE) en la tabla de ventas. Este trigger podría verificar si un producto está agotado y, en caso afirmativo, actualizar la tabla de inventario para reflejar ese cambio. Esto se hace automáticamente cada vez que se elimina un registro de ventas, asegurando que la información de inventario esté siempre actualizada.

Estructura de un Trigger

La estructura y el funcionamiento de un trigger pueden resumirse en tres pasos:

1. Se produce una llamada de activación al código que se ejecutará.
2. Se aplican las restricciones necesarias para realizar la acción, como verificar una condición o nulidad.
3. Una vez verificadas las restricciones, se ejecuta la acción basada en las instrucciones recibidas en el primer punto.
- 4.

Para comprender mejor la sintaxis de un trigger, aquí hay un ejemplo simple genérico:

```
CREATE TRIGGER nombre_disparador BEFORE / AFTER  
INSERT / UPDATE / DELETE  
ON nombre_tabla FOR EACH ROW BEGIN  
-- Código del trigger END;
```

Explicación de cada parámetro:

CREATE TRIGGER: nombre_disparador: Se utiliza para crear un nuevo trigger o cambiar el nombre de uno existente.

BEFORE / AFTER: Define cuándo se ejecutará el trigger, ya sea antes o después de un evento específico.

INSERT / UPDATE / DELETE: Describe la acción que se desea llevar a cabo en la tabla.

ON nombre_tabla: Especifica la tabla a la que se asocia el trigger.

FOR EACH ROW: Esta declaración se refiere a los triggers de filas, lo que significa que se ejecutarán cada vez que se modifique una fila.

BEGIN-END: Aquí se especifica el código que se ejecutará como parte del trigger.

Y aquí hay un ejemplo que hemos creado para la aplicación de los scripts:

```
CREATE TRIGGER  
trg_auditConsortio_update ON  
dbo.consortio  
AFTER  
UPDATE AS  
BEGIN  
    -- Registrar los valores antes de la modificación en una tabla  
    auxiliar INSERT INTO auditoriaConsortio  
    SELECT *, GETDATE(), SUSER_NAME(), 'Update'  
    FROM deleted  
END  
;
```

Tipos de Triggers

Los triggers SQL son funciones almacenadas que se ejecutan inmediatamente cuando ocurren eventos específicos. Se asemejan a la programación basada en eventos y pueden ser de varios tipos:

- Disparadores DML (Lenguaje de Manipulación de Datos): Estos se activan en respuesta a comandos DML como INSERT, UPDATE y DELETE. También se conocen como "Disparadores a nivel de tabla".
- Disparadores DDL (Lenguaje de Definición de Datos): Estos permiten ejecutar código en respuesta a cambios en la estructura de la base de datos, como la creación o eliminación de tablas. Son conocidos como "Disparadores a nivel de base de datos".
- Disparadores de inicio de sesión: Se invocan cuando se produce un evento de inicio de sesión, como el inicio de sesión, cierre de sesión o apagado del sistema. Se utilizan para auditar el acceso al sistema y gestionar la identidad de los usuarios.
- Disparadores CLR (Common Language Runtime): Estos son triggers únicos que aprovechan la tecnología .NET para ejecutar código. Son útiles cuando se necesitan cálculos complejos o interacción con entidades no relacionadas con SQL.

Ventajas de Utilizar Triggers

El uso de triggers proporciona varias ventajas importantes:

- Validación de datos: Los triggers permiten validar valores que no se pueden verificar mediante restricciones, lo que garantiza la integridad de los datos.
- Ejecución de reglas de negocios: Pueden aplicarse reglas empresariales complejas de

manera automatizada.

- Automatización de acciones complejas: Los triggers permiten realizar acciones sofisticadas en respuesta a eventos específicos.
- Registro de cambios: Ayudan a rastrear y mantener un registro de los cambios realizados en una tabla, a menudo utilizando una tabla de registro.

Desventajas de Utilizar Triggers

Sin embargo, el uso de triggers conlleva ciertas desventajas:

- ❑ Pérdida de visibilidad: Al ejecutarse automáticamente, puede ser difícil rastrear y depurar sentencias SQL.
- ❑ Sobrecarga del servidor: Un uso excesivo o incorrecto de triggers puede causar respuestas lentas del servidor y aumentar la carga de trabajo.
- ❑ Complejidad adicional: Los triggers pueden agregar complejidad al diseño de la base de datos. A medida que aumenta el número de triggers, el mantenimiento y la comprensión del sistema pueden volverse más difíciles.
- ❑ Riesgo de bucles infinitos: Si no se configuran correctamente, los triggers pueden generar bucles infinitos, donde uno dispara otro en un ciclo sin fin. Esto puede bloquear la base de datos y causar un rendimiento deficiente.
- ❑ Desencadenadores ocultos: Los triggers pueden no ser obvios para los desarrolladores y administradores que no estén familiarizados con la base de datos. Esto puede llevar a sorpresas inesperadas y dificultades para diagnosticar problemas.
- ❑ Dificultad en la migración y la transferencia: Los triggers pueden complicar la migración de datos entre bases de datos o sistemas, ya que es necesario asegurarse de que los triggers se activen correctamente en el nuevo entorno.
- ❑ Entre otros.

Metodología

En este capítulo se presenta el plan seguido o las acciones llevadas a cabo para realizar el trabajo, las dificultades encontradas y cualquier otra información que proporcione la idea de cómo se realizó el trabajo.

Para llevar a cabo este Trabajo Práctico, se han seguido los siguientes pasos:

En reunión virtual mencionada se había definido la implementación de triggers de auditorías para las tablas (Consortio, Gasto y Administrador) que responderán a cada operación de actualización (UPDATE) o eliminación (DELETE). Estos triggers registrarán en tablas auxiliares los valores de los registros antes de ser modificados o eliminados, junto con la fecha y hora de la operación, así como el usuario de la base de datos que realizó la acción. Esta acción tiene como objetivo mantener un registro detallado de las modificaciones realizadas en las tablas y proporcionar trazabilidad.

Además, se ha definido un trigger específico que impedirá la ejecución de una operación DELETE en la tabla (Administrador). Cuando se intente eliminar un registro en esta tabla, se emitirá un mensaje de error y no se permitirá la operación. Esto se ha implementado para garantizar la integridad y seguridad de los datos en la tabla Administrador.

Las herramientas y procedimientos utilizados en este trabajo son los siguientes:

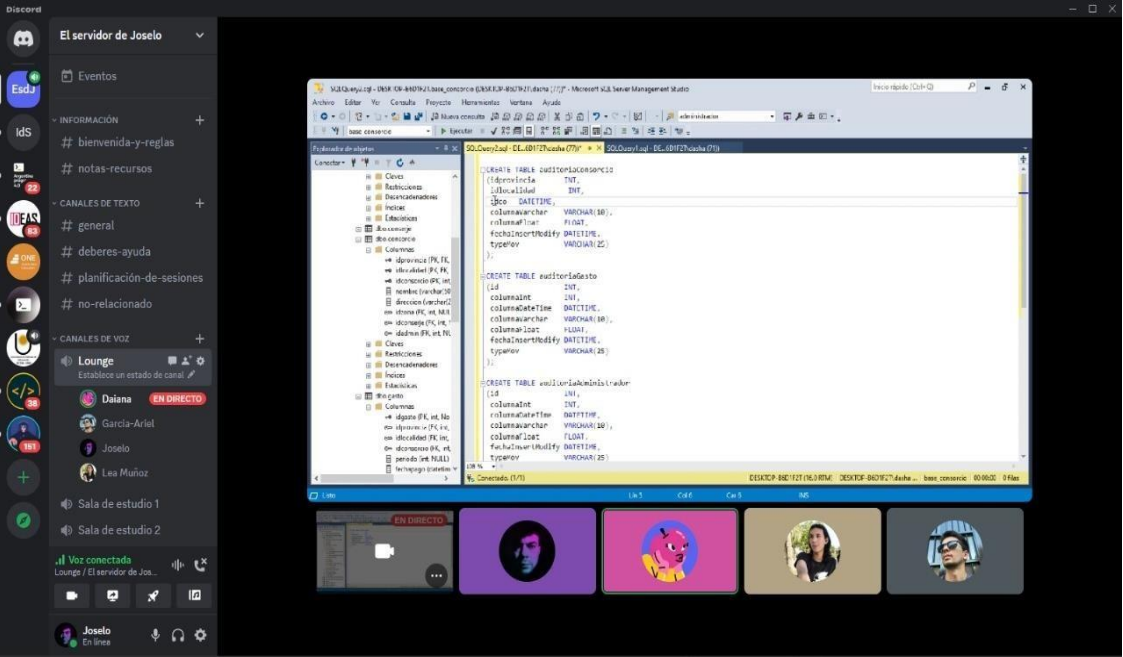
Herramienta Principal:

Para llevar a cabo la implementación de los triggers y ejemplificar su funcionamiento, se ha empleado el Gestor de Base de Datos SQL Server Management Studio.

Origen de los Datos: Los datos y ejemplos utilizados se extraen de una base de datos ya creada y cargada, denominada "base_consortio". Estos datos proporcionan un contexto realista para ejemplificar los triggers y su aplicación en situaciones prácticas.

Esta metodología se ha seguido para proporcionar ejemplos concretos y prácticos de cómo funcionan los triggers en un entorno de base de datos. La implementación de triggers de auditorías y la restricción de DELETE en la tabla Administrador son ejemplos concretos que ilustran el uso y la utilidad de los triggers en la gestión de bases de datos y que se harán ver en el desarrollo del mismo.

Además, para llevar a cabo el desarrollo del presente trabajo de investigación se han realizado reuniones grupales para la elaboración del Script de los Triggers solicitados y las tablas auxiliares necesarias, para ellos se dispuso un día y horario determinado para concertar una reunión virtual por videollamada a través de la plataforma Discord.



Desarrollo del Tema/ Resultados

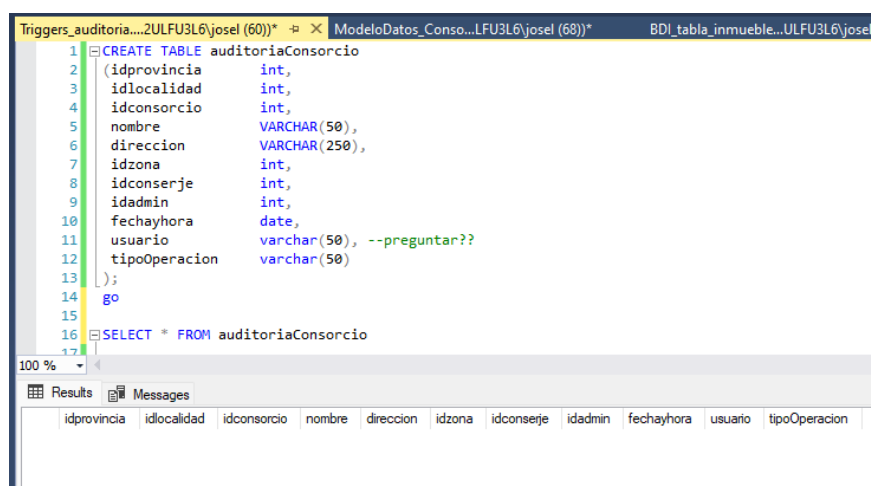
La seguridad y la integridad de los datos son de vital importancia en la actualidad. La creciente cantidad de información almacenada en sistemas de bases de datos ha generado la necesidad de implementar mecanismos efectivos de auditoría. La auditoría de bases de datos es una práctica que permite rastrear y registrar cambios en los datos, identificar actividades sospechosas y garantizar la transparencia.

Los triggers de auditoría de bases de datos son elementos que permiten la monitorización y el registro automatizado de ciertos eventos en una base de datos, tales como update, delete o insert permitiendo su trazabilidad

A continuación, se intentará explicar de la manera más sencilla y clara posible la implementación de triggers sobre las tablas consorcio, gasto y administrador, tablas que corresponde a la base de datos “base_consorcio”.

Para empezar con el desarrollo del tema, luego de haber buscado información e interiorizarnos, comenzamos con la elaboración del código necesario para la implementación de los TRIGGERS solicitados, cuyo Script es un producto del aporte en simultáneo de todos los integrantes del grupo.

Para comenzar se creó una tabla auditoriaConsorcio:



```
1 CREATE TABLE auditoriaConsorcio
2 (idprovincia int,
3 idlocalidad int,
4 idconsorcio int,
5 nombre VARCHAR(50),
6 direccion VARCHAR(250),
7 idzona int,
8 idconserje int,
9 idadmin int,
10 fechayhora date,
11 usuario varchar(50), --preguntar??
12 tipoOperacion varchar(50)
13 );
14 go
15
16 SELECT * FROM auditoriaConsorcio
```

idprovincia	idlocalidad	idconsorcio	nombre	direccion	idzona	idconserje	idadmin	fechayhora	usuario	tipoOperacion
-------------	-------------	-------------	--------	-----------	--------	------------	---------	------------	---------	---------------

Figura 1 - Código de creación de la tabla auditoriaConsorcio

Esta tabla guardará los datos de los registros que han de sufrir modificaciones luego de que ello ocurra.

Una vez creada la tabla mencionada en la base de datos creamos el TRIGGER trg_auditConsorcio_update que actuará como un disparador cada vez que se intente realizar una actualización en algún registro de la tabla consorcio en la base de datos base_consorcio cuyo objetivo principal es el de registrar en la tabla auditoriaConsorcio los valores antiguos de la fila que está siendo modificada.

```

CREATE TRIGGER trg_auditConsortio_update
ON dbo.consortio
AFTER UPDATE
AS
BEGIN
    -- Registrar los valores antes de la modificación en una tabla auxiliar
    INSERT INTO auditoriaConsortio
    SELECT *, GETDATE(), ORIGINAL_LOGIN(), 'Update'
    FROM deleted;
END;
go

```

Figura 2 - Código de creación del TRIGGER auditoriaConsortio – Operación Update

El TRIGGER se ejecutará después de que se haya realizado la actualización en la tabla realizando una inserción en la tabla auditoriaConsortio de los datos antiguos, la fecha de la actualización el nombre de usuario que realizó la operación. A continuación, en primer lugar se realizará una consulta (select) para obtener los datos de los registros de la tabla consortio.

143
144 SELECT * FROM consortio
145
146

100 %

	idprovincia	idlocalidad	idconsorcio	nombre	direccion	idzona	idconserje	idadmin
1	1	1	1	EDIFICIO-111	PARAGUAY Nº 630	5	100	1
2	1	2	2	EDIFICIO-122	Bº 250 VIV SEC 2 MZ 4 CSA Nº 2	5	99	2
3	2	48	1	EDIFICIO-2481	SAN LUIS Nº 1035, 4º piso, Dpto c	4	98	3
4	2	55	2	EDIFICIO-2552	REMEDIOS DE ESCALADA Nº 5353	3	97	4
5	3	16	1	EDIFICIO-3161	Bº VENEZUELA, GR.4, MZ 21,C.2	2	96	5
6	3	20	2	EDIFICIO-3202	LAVALLE Nº 1386	3	95	6
7	3	25	3	EDIFICIO-3253	JOSE G OMEZ Nº 770	4	94	7
8	3	27	4	EDIFICIO-3274	MARIANO MORENO Nº 1626, 2º piso, Dpto D	4	93	8
9	3	29	5	EDIFICIO-3295	MZ1 PC6 C26 Bº SANTA BARBARA	5	92	9
10	4	21	1	EDIFICIO-4211	COMANDANTE FONTANA Nº 174, -2º piso, Dpto -	6	91	10
11	4	36	2	EDIFICIO-4362	SAN MARTIN Nº 2195, PAº piso, Dpto 1	4	90	11
12	4	37	3	EDIFICIO-4373	PADRE BORGATTI Y MORENO	1	89	12
13	5	3	1	EDIFICIO-531	JULIO A. ROCA Nº 1281	6	88	13
14	5	4	2	EDIFICIO-542	Bº L SECA, 119VIV. STOR A, CASA 3	2	87	14
15	5	12	3	EDIFICIO-5123	BELGRANO Nº 2481	4	86	15

Query executed successfully. LAPTOP-ZULFU3L6\SQLEXPRESS ... LAPTOP-ZULFU3L6\SQLEXPRESS ...

Figura 3 - Consulta de la tabla consortio

A continuación, ejecutamos un update para comprobar el funcionamiento del TRIGGER, y podemos comprobar que el registro que ha sido modificado y se han guardado en la tabla auditoriaConsortio los valores antiguos del mencionado registro.

149 update consortio set direccion = 'Paraguay 387 - Planta Baja' where idadmin = 1
150 SELECT * FROM consortio
151

100 %

	idprovincia	idlocalidad	idconsorcio	nombre	direccion	idzona	idconserje	idadmin
1	1	1	1	EDIFICIO-111	Paraguay 387 - Planta Baja	5	100	1

Figura 4 - Update en tabla consortio


```

43
44 CREATE TABLE auditoriaGasto
45 (
46     idgasto INT,
47     idprovincia INT,
48     idlocalidad int,
49     idconsorcio int,
50     periodo int,
51     fechapago DATE,
52     idtipogasto int,
53     importe decimal(8,2),
54     fechayhora date,
55     usuario varchar(50),
56     tipoOperacion varchar(50)
57 );
58 Go
59 CREATE TRIGGER trg_auditGasto_update
60 ON dbo.gasto
61 AFTER UPDATE
62 AS
63 BEGIN
64     -- Registrar los valores antes de la modificación en una tabla auxiliar
65     INSERT INTO auditoriaGasto
66     SELECT *, GETDATE(), SUSER_NAME(), 'Update'
67     FROM deleted;
68 END;
69 go
70

```

Figura 8 - Tabla auditoriaGasto y TRIGGER auditGasto – Operación Update

El TRIGGER desarrollado realiza la acción de registrar en la tabla auxiliar auditoriaGasto, los datos antiguos después de realizarse el Update en la tabla gasto, así como también la fecha en que se realizó la modificación, el nombre de usuario de quien lo realizó y el tipo de operación.

Realizamos una prueba del código desarrollado, para ello primero efectuamos una consulta sobre los registros guardados en la tabla gasto, seguidamente hicimos un update sobre el

154
155
156

```
SELECT * FROM GASTO
```

100 %

	idgasto	idprovincia	idlocalidad	idconsorcio	periodo	fechapago	idtipogasto	importe
1	11	1	2	2	3	2013-03-28 00:00:00.000	2	9723.16
2	12	1	2	2	2	2013-02-19 00:00:00.000	4	3696.11
3	13	1	2	2	9	2013-09-14 00:00:00.000	3	1238.12
4	14	1	2	2	9	2013-09-14 00:00:00.000	3	3846.92
5	15	1	2	2	5	2013-05-19 00:00:00.000	3	55294.59
6	16	1	2	2	4	2013-04-01 00:00:00.000	5	828.13
7	17	1	2	2	5	2013-05-02 00:00:00.000	2	9346.01

importe del registro con idgasto = 11

Figura 9 - SELECT de la tabla gasto

Confirmada la actualización del registro en la tabla gasto ejecutamos una consulta sobre la tabla auditoriaGasto, cuyos resultados fueron los siguientes:

157
158
159
160

```
update gasto set importe = '14879' where idgasto = 11
SELECT * FROM auditoriaGasto
```

100 %

	idgasto	idprovincia	idlocalidad	idconsorcio	periodo	fechapago	idtipogasto	importe	fechayhora	usuario	tipoOperacion
81	11	1	2	2	3	2013-03-28	2	9723.16	2023-10-30	LAPTOP-2ULFU3L6\josel	Update

Figura 10 - Prueba de modificación en tabla gasto y registro en auditoriaConsortorio.

Seguidamente se crea el trigger para la operación de DELETE en la tabla de gasto con nombre tgr_auditGasto_delete.

```
CREATE TRIGGER trg_auditGasto_delete
ON dbo.gasto
AFTER DELETE
AS
BEGIN
    -- Registrar los valores antes de la eliminación en una tabla auxiliar
    INSERT INTO auditoriaGasto
    SELECT *, GETDATE(), SUSER_NAME(), 'Delete'
    FROM deleted;
END
GO
```

Figura 11 - Código de creación del trigger de la tabla gasto para la operación DELETE

Como se puede ver en el código lo que realiza el disparador es insertar dentro de la tabla auxiliar auditoriaGasto todos los datos del registro que se elimina y además se guarda la fecha y hora, el nombre del usuario de la base de dato que se encuentre actualmente, y la leyenda 'Delete' para saber de qué tipo de operación se trata.

Para probar el disparador primero se crea un registro nuevo de gasto que se eliminará luego.

```
INSERT INTO gasto VALUES (1,1,1,2,GETDATE(), 1, 156215.20);
SELECT * FROM gasto;
```

Figura 12 - Código de inserción de un gasto nuevo.

7...	7998	24	17	6	8	2017-08-18 00:00:00.000	5	107.52
7...	7999	24	17	6	1	2017-01-01 00:00:00.000	2	1424.34
8...	8000	24	17	6	7	2017-07-02 00:00:00.000	5	869.17
8...	8002	1	1	1	2	2023-10-28 21:13:22.373	1	156215.20

Figura 13 - Resultado luego de la inserción que se visualiza en el id 8002.

Luego, se elimina el registro que se había ingresado anteriormente asignado con el idgasto nro 8002.

```
DELETE FROM gasto WHERE idgasto = 8002;
```

Figura 14 - Código de eliminación del gasto 8002

Results		Messages									
	idgasto	idprovincia	idlocalidad	idconsorcio	periodo	fechapago	idtipogasto	importe			
7...	7996	24	17	6	4	2017-04-08 00:00:00.000	5	195.62			
7...	7997	24	17	6	9	2017-09-02 00:00:00.000	5	871.30			
7...	7998	24	17	6	8	2017-08-18 00:00:00.000	5	107.52			
7...	7999	24	17	6	1	2017-01-01 00:00:00.000	2	1424.34			
8...	8000	24	17	6	7	2017-07-02 00:00:00.000	5	869.17			
	idgasto	idprovincia	idlocalidad	idconsorcio	periodo	fechapago	idtipogasto	importe	fechayhora	usuario	tipoOperacion
1	8002	1	1	1	2	2023-10-30	1	156215.20	2023-10-30	LeaMunoz	Delete

Figura 15 - -Resultado luego de la eliminación visualizando las tablas gasto y auditoriaGasto

Y como se puede comprobar, el disparador realiza la función requerida, guardando los datos del registro eliminado en la tabla auditoriaGasto(egistro de tabla inferior).

A continuación, se procede a la creación de disparadores para la tabla administrador, en el cual se debe resguardar los datos que se han modificado y evitar la eliminación de algún registro. Por lo tanto, en primer lugar, se crea una tabla auxiliar a la que llamaremos auditoriaAdministrador que contendrá las mismas columnas que la tabla administradora, además una columna para la fecha y hora, otra para el nombre de usuario de la base de datos y otro para el tipo de operación.

```
CREATE TABLE auditoriaAdministrador
(
    idadmin INT,
    apeynom VARCHAR(50),
    viveahi VARCHAR(1),
    tel VARCHAR(20),
    sexo VARCHAR(1),
    fechanac DATETIME,
    fechayhora DATE,
    usuario VARCHAR(50),
    tipoOperacion VARCHAR(50)
);
GO
```

Figura 16 - Código de creación de una tabla auxiliar auditoriaAdministrador para resguardar los datos actualizados de la tabla administrador.

Luego, se procede a la creación del disparador para la operación de UPDATE, que llevará el nombre de tgr_auditAdmin_update.

```
CREATE TRIGGER tgr_auditAdmin_update
ON dbo.administrador
AFTER UPDATE
AS
BEGIN
    -- Registrar los valores antes de la modificación en una tabla auxiliar
    INSERT INTO auditoriaAdministrador
    SELECT *, GETDATE(), SUSER_NAME(), 'Update'
    FROM deleted;
END;
GO
```

Figura 17 - Código de creación del trigger de la tabla administrador para la operación UPDATE

El código indica que, al igual que los disparadores anteriores, al momento de actualizar algún dato de la tabla el disparador insertará dentro de la tabla auxiliar auditoriaAdministrador todos los datos del registro que se actualiza y además se guarda la fecha y hora, el nombre del usuario de la base de datos y la leyenda 'Update'.

Nuevamente, para probar que el disparador funciona correctamente, primero se crea un nuevo administrador.

```
INSERT INTO administrador VALUES ('Gomez María', 'N', '3794568542', 'F', '19900615');
SELECT * FROM administrador;
```

Figura 18 - Código de inserción de un registro en la tabla administrador.

172	172	SAUCEDO FAUSTINA	S	3678235689	F	1987-07-23 00:00:00.000
173	173	GONZALEZ LIDIA ESTER	N	3671235689	F	1984-11-20 00:00:00.000
174	174	ARAUJO GUILLERMO JOSE	S	3672235689	M	1985-10-13 00:00:00.000
175	175	Gomez María	N	3794568542	F	1990-06-15 00:00:00.000
176	176	Gomez María	N	3794568542	F	1990-06-15 00:00:00.000

Figura 19 - Resultado luego de insertar 2 veces el registro anterior (fila 175 y 176).

Como se puede notar, se ingresó dos veces el mismo registro por lo que se procede a modificar el último.

```

UPDATE administrador
SET apeynom = 'Fernandez Laura', tel = '3794611462', fechnac = '19840418'
WHERE idadmin = 176;

SELECT * FROM administrador;
SELECT * FROM auditoriaAdministrador;

```

Figura 20 - Código de actualización de los datos del registro 176 de la tabla administrador, cambiando nombre, teléfono y fecha de nacimiento

	idadmin	apeynom	viveahi	tel	sexo	fechnac			
169	169	TALAVERA CONSTANCIA	N	3655235689	F	1986-01-02 00:00:00.000			
170	170	VARGAS MIGUEL	S	3676235689	M	1986-03-15 00:00:00.000			
171	171	MACIEL PAULINA	N	3677235689	F	1970-03-18 00:00:00.000			
172	172	SAUCEDO FAUSTINA	S	3678235689	F	1987-07-23 00:00:00.000			
173	173	GONZALEZ LIDIA ESTER	N	3671235689	F	1984-11-20 00:00:00.000			
174	174	ARAUJO GUILLERMO JOSE	S	3672235689	M	1985-10-13 00:00:00.000			
175	175	Gomez María	N	3794568542	F	1990-06-15 00:00:00.000			
176	176	Fernandez Laura	N	3794611462	F	1984-04-18 00:00:00.000			

	idadmin	apeynom	viveahi	tel	sexo	fechanac	fechayhora	usuario	tipoOperacion
1	176	Gomez María	N	3794568542	F	1990-06-15 00:00:00.000	2023-10-28	LeaMunoz	Update

Figura 21 - Resultado de las tablas administrador y auditoriaAdministrador luego de la actualización.

El disparador realiza la función requerida correctamente, guardando los datos del registro modificado en la tabla auditoriaAdministrador (Figura 21-2do registro de tabla inferior).

En cuanto al disparador para la operación DELETE, éste difiere de los demás ya que no debe permitir la eliminación de un registro de administrador. Por lo tanto, la creación del trigger con el nombre tgr_auditoriaAdmin_delete se realiza de la siguiente manera:

```

CREATE TRIGGER trg_auditAdmin_delete
ON dbo.administrador
AFTER DELETE
AS
BEGIN
    -- Emitir un mensaje y prevenir la operación de eliminación
    RAISERROR('La eliminación de registros en la tabla Administrador no está permitida.', 16, 1);
    ROLLBACK;
END
GO

```

Figura 22 - Código de creación del trigger de la tabla administrador para la operación DELETE

Cuando se intente eliminar un registro lo que realizará el disparador será, en primer lugar, emitir un mensaje de error que rece “La eliminación de registros de la tabla Administrador no está permitida”, luego ejecutará un rollback que revertirá la eliminación que se intentó realizar.

De igual manera se procede a realizar la prueba del disparador intentando eliminar el registro ingresado anteriormente.


```
14 | DELETE FROM administrador WHERE idadmin = 176;
15 |
```

Messages

Msg 50000, Level 16, State 1, Procedure trg_auditAdmin_delete, Line 8 [Batch Start Line 13]
La eliminación de registros en la tabla Administrador no está permitida.
Msg 3609, Level 16, State 1, Line 14
The transaction ended in the trigger. The batch has been aborted.

Figura 23 - Código de eliminación del registro 176 de la tabla administrador y mensaje de error lanzado por el trigger.

```
14 | DELETE FROM administrador WHERE idadmin = 176;
15 |
16 | SELECT * FROM administrador;
```

124 %

	idadmin	apeynom	viveahi	tel	sexo	fechnac
165	165	ARMOA DEMETRIA	N	3651235689	F	1992-08-15 00:00:00.000
166	166	LUGO DE R. RAMONA	S	3652235689	F	1975-10-18 00:00:00.000
167	167	LOPEZ ELOYSA	N	3653235689	F	1989-06-08 00:00:00.000
168	168	LOPEZ CONRADO	S	3654235689	M	1983-10-26 00:00:00.000
169	169	TALAVERA CONSTANCIA	N	3655235689	F	1986-01-02 00:00:00.000
170	170	VARGAS MIGUEL	S	3676235689	M	1986-03-15 00:00:00.000
171	171	MACIEL PAULINA	N	3677235689	F	1970-03-18 00:00:00.000
172	172	SAUCEDO FAUSTINA	S	3678235689	F	1987-07-23 00:00:00.000
173	173	GONZALEZ LIDIA ESTER	N	3671235689	F	1984-11-20 00:00:00.000
174	174	ARAUJO GUILLERMO JOSE	S	3672235689	M	1985-10-13 00:00:00.000
175	175	Gomez Maria	N	3794568542	F	1990-06-15 00:00:00.000
176	176	Fernandez Laura	N	3794611462	F	1984-04-18 00:00:00.000

Figura 24 - Consulta de la tabla administrador en la que se visualiza el registro que se intentó eliminar anteriormente.

Luego, se puede comprobar que el disparador funciona correctamente ya que al ejecutar la eliminación del registro lanza el error que se había programado y al visualizar la tabla de administradores se puede ver que sigue ahí (Figura 24).

Réplicas de base de datos

Introducción

El presente trabajo práctico tiene como objetivo proporcionar una comprensión sólida acerca de un tema crítico en el mundo de la gestión de datos: las “**Réplicas de Base de**

Datos”. Esta introducción tiene como finalidad presentar los fundamentos esenciales del trabajo, estableciendo de manera clara el por qué y para qué se lleva a cabo.

En un mundo cada vez más interconectado y orientado a los datos, la gestión eficaz de la información se ha vuelto fundamental para el funcionamiento de empresas, organizaciones y sistemas. Tener en cuenta que también es una estrategia clave para la administración de información en un entorno tecnológico en constante evolución.

El problema fundamental que abordaremos en este informe es la creciente necesidad de garantizar la integridad, disponibilidad y escalabilidad de las bases de datos en un entorno donde los datos se han convertido en el recurso más valioso. A medida que las organizaciones se enfrentan a la gestión de volúmenes de datos en constante crecimiento, se hace evidente la necesidad de afrontar este desafío de manera efectiva. En este contexto, las réplicas de bases de datos se presentan como una solución potencial para mejorar el rendimiento, la tolerancia a fallos y la disponibilidad de los sistemas de información. Sin embargo, su implementación y gestión eficiente plantean desafíos significativos que requieren una atención especializada.

Objetivos

Objetivos Generales

Nuestro objetivo general es explorar a fondo el concepto de réplicas de base de datos, investigar sus ventajas y desafíos, y proporcionar una guía integral sobre su implementación efectiva. Aspiramos a brindar una comprensión sólida y clara de cómo las réplicas de bases de datos pueden contribuir a la gestión de datos de manera eficiente y confiable en un entorno laboral y tecnológico en constante evolución.

Objetivos Específicos

- Identificar y analizar las ventajas estratégicas y operativas de utilizar réplicas de base de datos en entornos laborales.
- Explorar y evaluar las tecnologías y enfoques más comunes para la implementación de réplicas de bases de datos, incluyendo replicación sincrónica y asincrónica, y sus aplicaciones en diferentes contextos.
- Ofrecer pautas detalladas para la configuración, mantenimiento y monitoreo exitosos de réplicas de bases de datos, destacando mejores prácticas y posibles desafíos a superar.
- Evaluar casos de estudio y escenarios reales donde las réplicas de bases de datos han demostrado ser beneficiosas, ilustrando su impacto positivo en la eficiencia y la continuidad de los sistemas de información.
- Proporcionar recomendaciones prácticas para abordar desafíos y mitigar riesgos asociados con la implementación de réplicas de bases de datos, incluyendo cuestiones de seguridad, rendimiento y mantenimiento.

MARCO CONCEPTUAL O REFERENCIAL

En este capítulo, se presentará el marco conceptual de nuestro informe, es decir las bases para comprender la replicación de bases de datos en SQL Server. Se abordarán conceptos fundamentales, se destacó la importancia de establecer un marco conceptual sólido en este contexto.

Conceptos Fundamentales de Replicación de Bases de Datos

¿Qué es la Replicación?

La replicación de bases de datos se describe como el proceso de transferir datos desde una base de datos de origen a una o varias bases de datos de destino. Este proceso asegura que los datos se mantengan consistentes en todas las ubicaciones, permitiendo que los usuarios accedan a información actualizada y sincronizada, ya sea en tiempo real o como parte de una operación programada por lotes. En última instancia, la replicación de bases de datos crea una red distribuida de almacenamiento de datos que garantiza una mayor disponibilidad y acceso eficiente a datos cruciales en diversas ubicaciones.

¿Cómo Funciona la Replicación De Base de Datos?

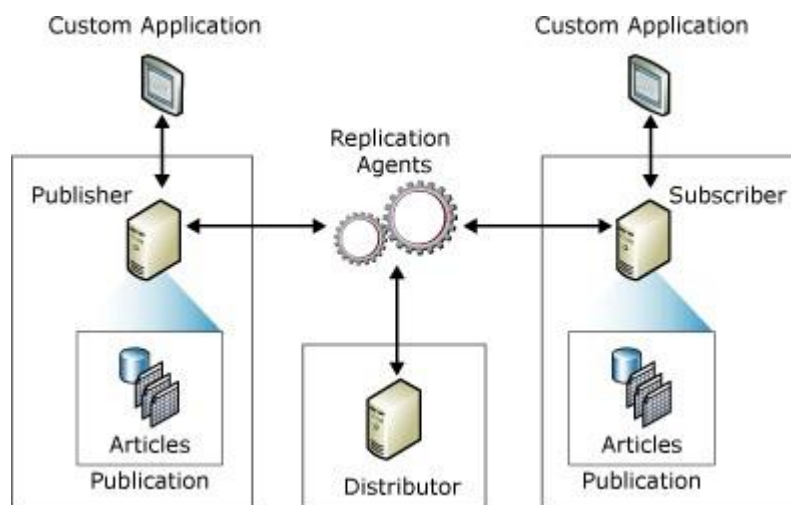
La replicación de bases de datos puede ser un proceso único o continuo que implica a todas las fuentes de datos de una organización. Se utiliza un sistema de gestión de bases de datos distribuidas (DBMS) para distribuir los datos a todas las fuentes pertinentes.

Cuando se realizan cambios, adiciones o eliminaciones en la base de datos de origen, se garantiza la sincronización automática con otras bases de datos de destino, siempre que esos cambios sean necesarios. Este proceso se rige por el paradigma convencional de software Editor-Suscriptor, donde intervienen uno o varios "editores" y "suscriptores".

El "publicador" se refiere a la base de datos de origen donde se originan los cambios, mientras que el "suscriptor" representa un sistema en el cual se replican los cambios. Las

modificaciones efectuadas en un sistema "publicador" se replican en las bases de datos "suscriptoras". Los usuarios también tienen la capacidad de efectuar cambios en las bases de datos de los suscriptores, que luego se propagan a la base de datos del editor. Este enfoque distribuye las actualizaciones a todos los demás miembros de la red si el sistema admite una replicación bidireccional.

Además, la mayoría de los suscriptores están conectados de manera permanente al editor, lo que permite que las actualizaciones se efectúen automáticamente sin necesidad de intervención manual. Estas actualizaciones pueden ocurrir en lotes a intervalos regulares o pueden aplicarse en tiempo real, según las necesidades y configuraciones específicas.



Tipos De Replicación De Datos

Existen diversos tipos de Replicación los cuales procederemos a nombrar, pero en el que nosotros nos centraremos será en la Replicación Transaccional.

Replicación de Tabla Completa

Crea una copia completa de la base de datos de origen en el almacenamiento de destino, incluyendo filas nuevas, modificadas y existentes.

Requiere una alta capacidad de procesamiento y ancho de banda de red, lo que puede resultar en sobrecarga de la red y retrasos en la replicación, especialmente con grandes volúmenes de datos.

Replicación de Instantáneas

Se utiliza una "instantánea" de la base de datos de origen en un momento específico para replicar datos en la base de datos de destino.

No tiene en cuenta los cambios en los datos (nuevos, actualizados o eliminados) y es preferible cuando los cambios de datos son escasos.

Es más rápida que la replicación de tabla completa pero no rastrea datos eliminados.

Replicación por Fusión

Permite la modificación de objetos y datos en ambas bases de datos (editor y suscriptor), lo que puede dar lugar a conflictos de datos relacionados con versiones.

Los agentes de fusión resuelven conflictos siguiendo un proceso predefinido.

Replicación Incremental Basada en Claves

Este método verifica las claves o índices en busca de cambios (borrados, nuevos o actualizados) en una base de datos y replica solo las claves necesarias para reflejar los cambios desde la última actualización.

Es rápido, pero no admite eliminaciones permanentes.

Replicación Incremental Basada en Registros

Réplica datos según el archivo de registro binario de la base de datos de origen, que contiene información sobre cambios, como actualizaciones, inserciones y eliminaciones. Es eficaz, especialmente para bases de datos estáticas, y es compatible con muchos proveedores de bases de datos.

Replicación transaccional

Es un método de replicación de bases de datos, en el que nos centraremos, y que el mismo se enfoca en mantener la consistencia de los datos entre una base de datos de origen y una o varias bases de datos de destino. Su funcionamiento se basa en el seguimiento de transacciones individuales realizadas en la base de datos de origen. Cuando ocurre una novedad en los datos de origen, ya sea una inserción, actualización o eliminación, la replicación transaccional captura y registra esta transacción.

La característica distintiva de la replicación transaccional es que se asegura de que todas las transacciones realizadas en la base de datos de origen se repliquen en tiempo real o de manera programada en las bases de datos de destino. Esto significa que los datos en las bases de destino reflejan de manera precisa y casi inmediata los cambios que ocurren en la base de datos de origen.

Es eficaz en escenarios donde la integridad y la consistencia de los datos son críticas, como en aplicaciones financieras, sistemas de reservas o aplicaciones médicas. Sin embargo, la replicación transaccional se utiliza principalmente para operaciones de lectura, lo que significa que está diseñada para garantizar que los datos sean accesibles y precisos para los usuarios finales. No está destinada a operaciones de creación, borrado o actualización de datos en las bases de destino.

¿Por qué es Importante la Replicación?

- Fiabilidad y disponibilidad de los datos.
- Recuperación en caso de desastre.
- Rendimiento del servidor.
- Mejor rendimiento de la red.
- Mejora del rendimiento del sistema de pruebas.

¿Qué inconvenientes se pueden tener con la Replicación?

- Costes más elevados.
- Limitaciones de tiempo.
- Ancho de banda.
- Datos incoherentes.

Técnicas de Replicación

Antes de comenzar a hablar de las técnicas como tal, es importante resaltar la diferencia que hay entre los Tipos de Replicaciones, respecto a las Técnicas, si bien la diferencia es sutil puede, prestar a confusión.

Tipos de Replicación

Son las Categorías generales o enfoques de cómo se realiza la replicación de bases de datos en función de sus características y objetivos. Estos tipos suelen definirse en función de la forma en que se copian y sincronizan los datos, así como de las necesidades específicas de un sistema. Cada tipo de replicación tiene sus propias ventajas y desventajas, y se elige en función de los requisitos y las prioridades del sistema.

Técnicas de Replicación

Son los métodos y estrategias específicas utilizados para implementar un tipo de replicación en particular. Estas técnicas abordan los detalles de cómo se capturan, transmiten y aplican los cambios de datos entre la base de datos de origen y las bases de datos de destino. Las técnicas pueden variar según el tipo de replicación elegido.

Entre las técnicas a destacar podemos mencionar la **Replicación Completa de Bases de Datos**. Es una técnica que implica la duplicación de toda la base de datos en distintos hosts, lo que proporciona un alto grado de redundancia y disponibilidad de datos. Esta técnica es valiosa para empresas globales que necesitan que los usuarios de diferentes regiones accedan a los mismos datos a velocidades óptimas. En caso de una falla en el servidor local, los usuarios pueden recurrir a copias de seguridad en otros servidores globales.

Sin embargo, es importante tener en cuenta que esta técnica puede ser lenta para realizar actualizaciones, ya que implica la replicación de grandes volúmenes de datos. Además, mantener la consistencia en la ubicación de cada archivo puede ser un desafío, especialmente cuando los datos experimentan cambios constantes.

Replicación Parcial de Bases de Datos

La replicación parcial de bases de datos consiste en dividir los datos de una base de datos en segmentos lógicos y almacenarlos en ubicaciones diferentes según su relevancia y uso. Este enfoque es particularmente beneficioso para profesionales como expertos en seguros, asesores financieros y vendedores, quienes pueden llevar consigo subconjuntos de datos relevantes en dispositivos o portátiles y sincronizarlos periódicamente con un servidor central.

Además, para analistas y empresas, esta técnica puede ser más rentable, ya que permite mantener los datos geográficamente cercanos a los consumidores, al tiempo que se conserva una copia completa de los datos en la sede central para análisis de alto nivel.

METODOLOGÍA

Organización del Equipo y Proceso de Trabajo

La organización del equipo desempeñó un papel crucial en el desarrollo exitoso del proyecto, que tenía como objetivo principal abordar el tema de la "Réplica de Base de Datos". A continuación, se describe el proceso de organización y colaboración empleado por el equipo:

- 1. Organización del Grupo de Trabajo:** En una primera reunión, se llevó a cabo una discusión para determinar de qué manera íbamos a trabajar, con qué herramientas, etc. El tema que nos tocó desarrollar fue "Réplica de Base de Datos". Se determinó que el proyecto se centraría en la implementación de una base de datos con el tipo de replicación transaccional.
- 2. Búsqueda de Información:** Una vez definido el tema, se asignó a cada miembro del equipo la tarea de buscar información relevante sobre la réplica de bases de datos. A través de comunicación vía WhatsApp, Discord, nos compartimos enlaces y datos que fue recolectando cada uno.
- 3. Reuniones de Sincronización:** Se organizaron reuniones periódicas para reunir toda la información recopilada por cada miembro del equipo. Durante estas reuniones, se compartió la información mencionada.
- 4. Planificación en Trello:** Para una gestión eficiente de las tareas, se utilizó la plataforma Trello. Se crearon tableros en Trello para dividir y asignar las responsabilidades a cada integrante del equipo. Esto permitió un seguimiento claro de las actividades en curso.
- 5. Colaboración en Google Drive:** Se estableció un espacio de colaboración en Google Drive donde se almacenaron y compartieron las actualizaciones y revisiones.

del informe en desarrollo. Esto aseguró que todos los miembros del equipo pudieran acceder a los documentos y realizar contribuciones de manera efectiva.

6. **Reuniones de Avance:** Se llevaron a cabo llamadas regulares dos veces por semana para revisar el progreso individual de cada miembro y discutir los avances del proyecto en su conjunto.
7. **Seguimiento de Tutoriales:** Se programó una llamada grupal para poder seguir tutoriales proporcionados por Microsoft SQL Server para adquirir el conocimiento necesario sobre la réplica de bases de datos.
8. **Unificación del Informe:** Finalmente, se procedió a unificar todas las partes del informe siguiendo las directrices establecidas durante las reuniones de avance y la revisión conjunta de la información recopilada.

Este proceso de organización y trabajo en equipo garantizó la eficiencia en la realización del proyecto, permitiendo una colaboración efectiva y una presentación coherente de los resultados.

Herramientas, Instrumentos y Procedimientos

En esta sección, se detalla la elección de herramientas y procedimientos utilizados para la recopilación y el tratamiento de la información requerida para el proyecto de estudio

1. **Recolección de Datos:** Los datos necesarios para el proyecto se obtuvieron principalmente a través de fuentes en línea. Este enfoque incluyó la revisión de documentación técnica, tutoriales, y recursos disponibles en la web.
2. **Herramientas Informáticas y Plataformas:** Para la organización y la colaboración en el proyecto, se emplearon herramientas informáticas como Trello, Google Drive y GitHub. Estas plataformas fueron esenciales para mantener un flujo de trabajo eficiente y garantizar la accesibilidad a los recursos compartidos.

3. **Comunicación Interpersonal:** Discord y WhatsApp se utilizaron para la comunicación y coordinación entre los miembros del equipo. La programación de reuniones regulares permitió compartir información de manera efectiva y garantizar la cohesión del grupo de trabajo.
4. **Gestión de la Base de Datos:** Para la implementación de la réplica de base de datos, se optó por utilizar SQL Server Management Developer Edition. Esta elección se basó en las capacidades y características que esta herramienta proporcionó, que resultaron fundamentales para el desarrollo exitoso del proyecto.

SQL Server Management Developer Edition permitió:

- **Diseño de la Base de Datos:** Facilitó el diseño y modelado de la base de datos, permitiendo definir la estructura y las relaciones necesarias.
- **Administración de la Réplica:** Ofreció las herramientas necesarias para configurar y gestionar la réplica de datos entre los servidores, lo que fue esencial para el objetivo del proyecto.
- **Análisis de Datos:** Permitió la realización de análisis y consultas sobre los datos replicados, lo que contribuyó a la evaluación de la efectividad de la réplica y a la obtención de conclusiones relevantes.

Réplica de la Base de Datos

La réplica de la base de datos fue un componente fundamental en la metodología aplicada para el proyecto de estudio. Para llevar a cabo este proceso, se siguió un conjunto de tutoriales proporcionados por Microsoft SQL Server, los cuales ofrecieron una guía detallada y precisa sobre la preparación del servidor y la replicación de datos.

DESARROLLO DEL TEMA / PRESENTACIÓN DE RESULTADOS

Requisitos previos

Tener instalado SQL Server, SQL Server Management Studio (SSMS) y una base de datos en este caso **base_consortio**.

En el servidor de publicación (**maestro**), instalar SQL Server Developer Edition u otra versión que permita la publicación (**EXPRESS no lo permite**).

En el servidor de suscripción (**esclavo**), instalar cualquier edición de SQL Server Compact.

SQL Server Compact no puede ser un suscriptor de una replicación transaccional.

Creación de cuentas de Windows para replicación

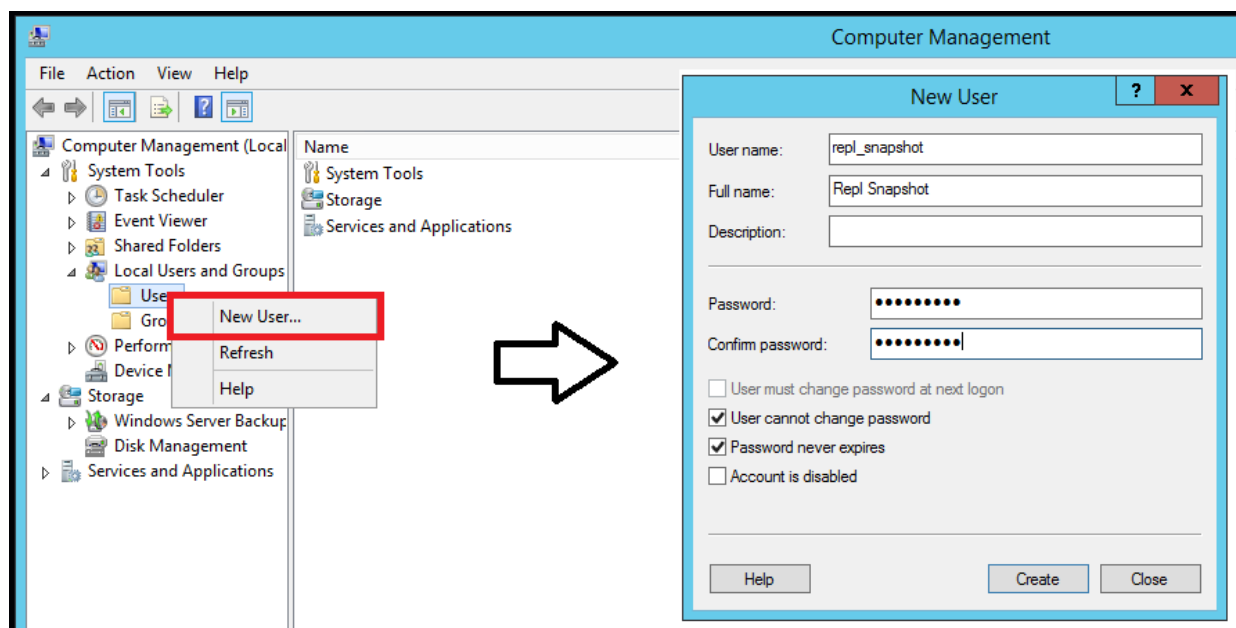
Antes de hacer la replicación una buena práctica y fundamental para la seguridad configurar bien los permisos que tendrán cada rol, cabe aclarar que, si el publicador y el suscriptor comparten la misma instancia, no se requieren los pasos que se utilizan para crear las cuentas en el suscriptor, pero en este caso están en dos instancias distintas por lo que para ello se crearán cuatro cuentas en windows:

- Agente de instantáneas (**Publisher**) que llamaremos **repl_snapshot**
- Agente de registro del LOG (**Published**) que llamaremos **repl_logreader**
- Agente de distribución (**Publicador y suscriptor**) que llamaremos

repl_distribution

- Agente de mezcla (**Publicador y suscriptor**) que llamaremos **repl_merge**
- Creación de cuentas locales de Windows para agentes de replicación en el publicador**

Se Debe ir “Administración de equipos”, luego a “Usuarios y grupos locales”, luego crea un nuevo usuario, coloque los datos (**contraseña a preferir solo debe recordarla al momento de usarla**) por cada usuario.

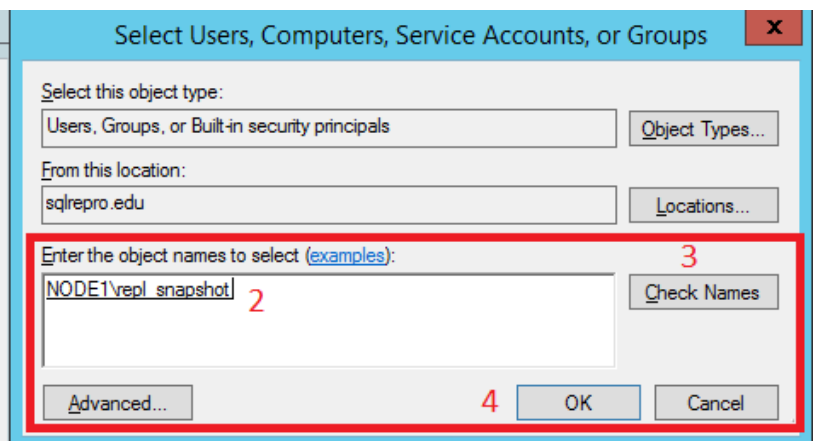
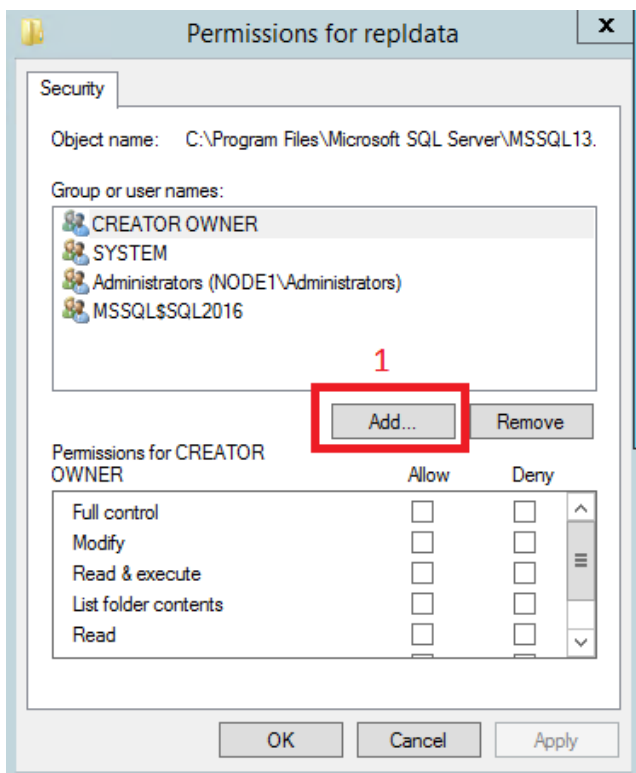
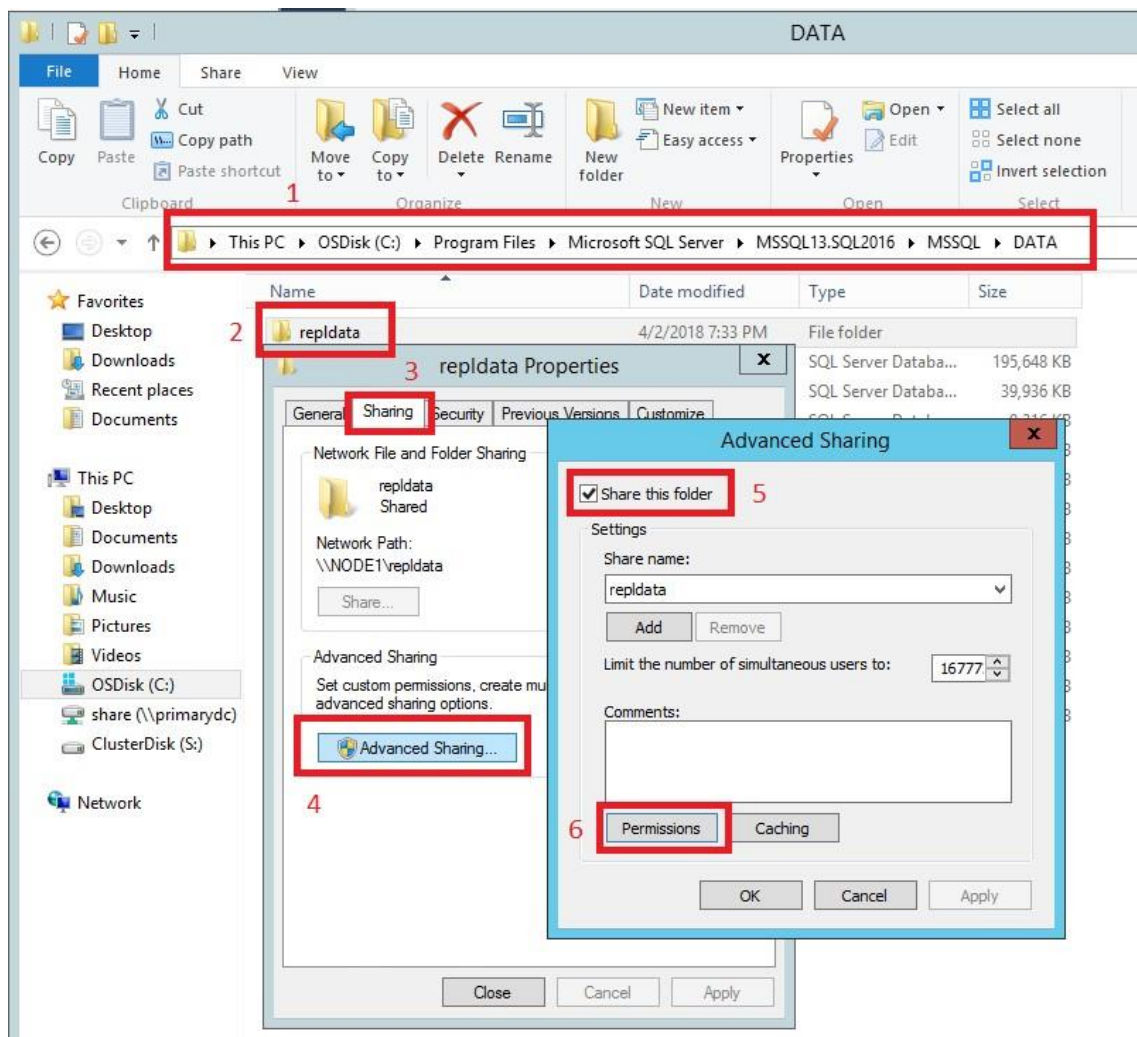


Preparación de la carpeta de instantáneas

En esta sección se va a configurar la carpeta de instantáneas que se utiliza para crear y almacenar la instantánea de publicación.

Crear un recurso compartido para la carpeta de instantáneas y asignar permisos

1. En el Explorador de archivos, vaya a la carpeta de datos SQL Server. La ubicación predeterminada es C:\Archivos de programa\Microsoft SQL Server\MSSQL.X\MSSQL\Data.
2. Cree una carpeta con el nombre **repldata**.
3. Haga clic con el botón derecho en esta carpeta y seleccione Propiedades.
 - En la pestaña Compartir del cuadro de diálogo Propiedades de repldata, seleccione Uso compartido avanzado.
 - En el cuadro de diálogo Uso compartido avanzado, seleccione Compartir esta carpeta y, después, seleccione Permisos.

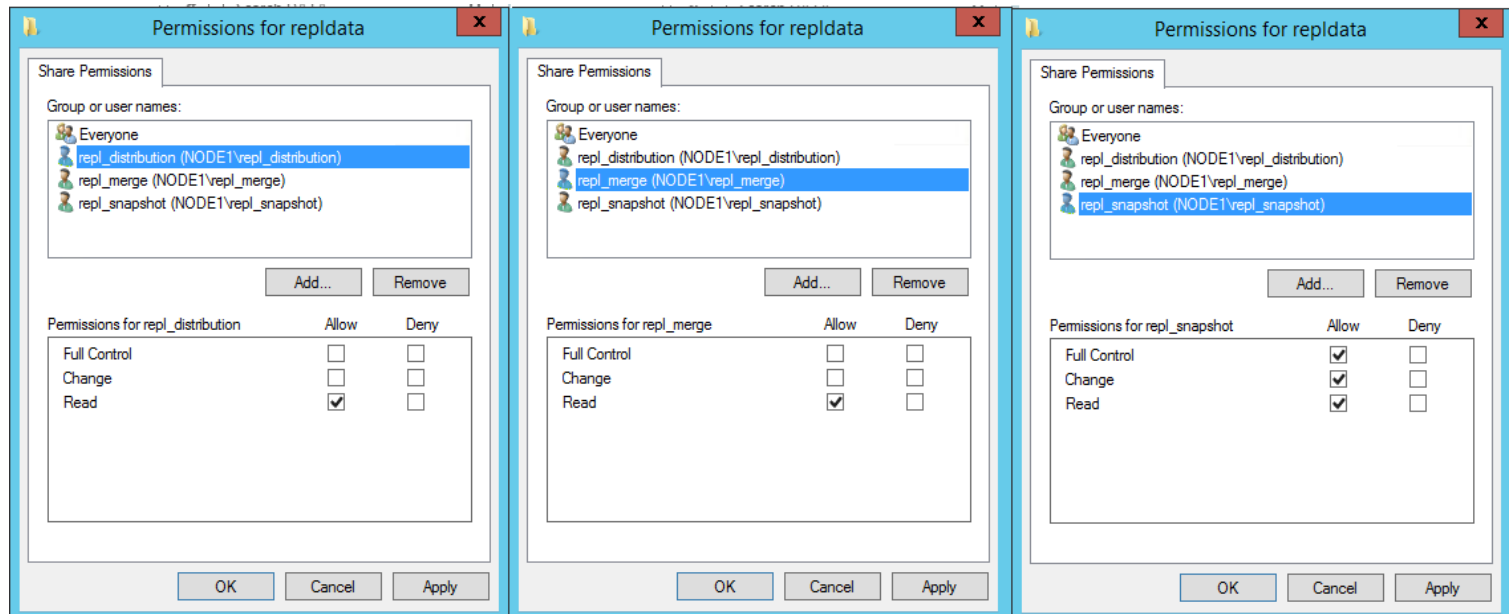


4. En el cuadro de diálogo Permisos de repldata, seleccione Agregar. En el cuadro de texto Seleccionar usuarios, equipos, cuentas de servicio o grupos, escriba el nombre de la cuenta del Agente de instantáneas creado anteriormente, como <Nombre_De_Equipo_Publicador>**repl_snapshot**. Seleccione Comprobar nombres, después, Aceptar. (Repita por cada usuario).

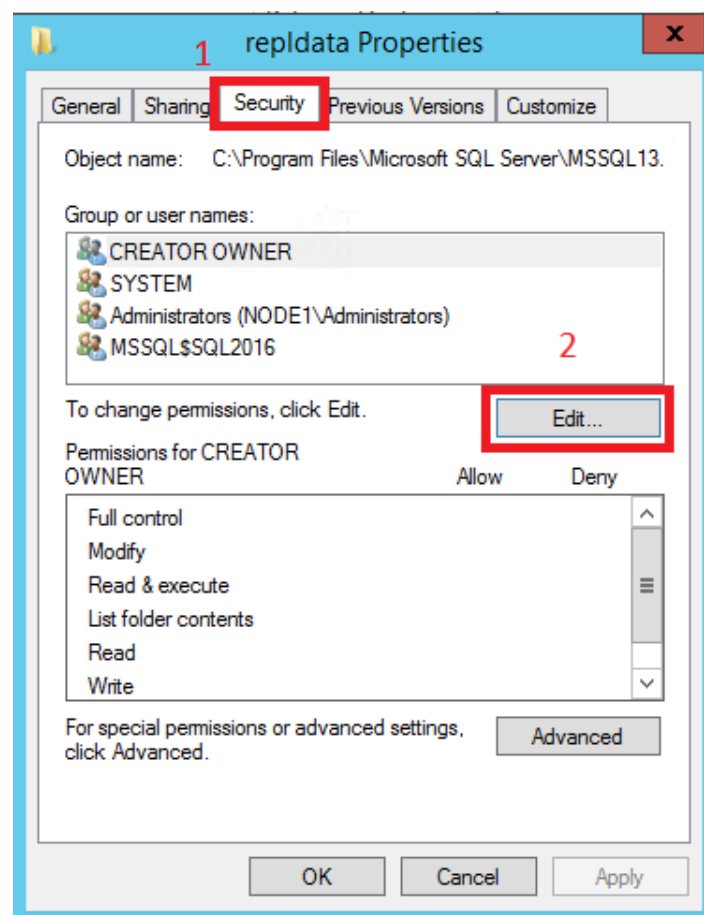
5. Asegúrese que los permisos queden de la siguiente forma:

repl_distribution: Lectura **repl_merge:**

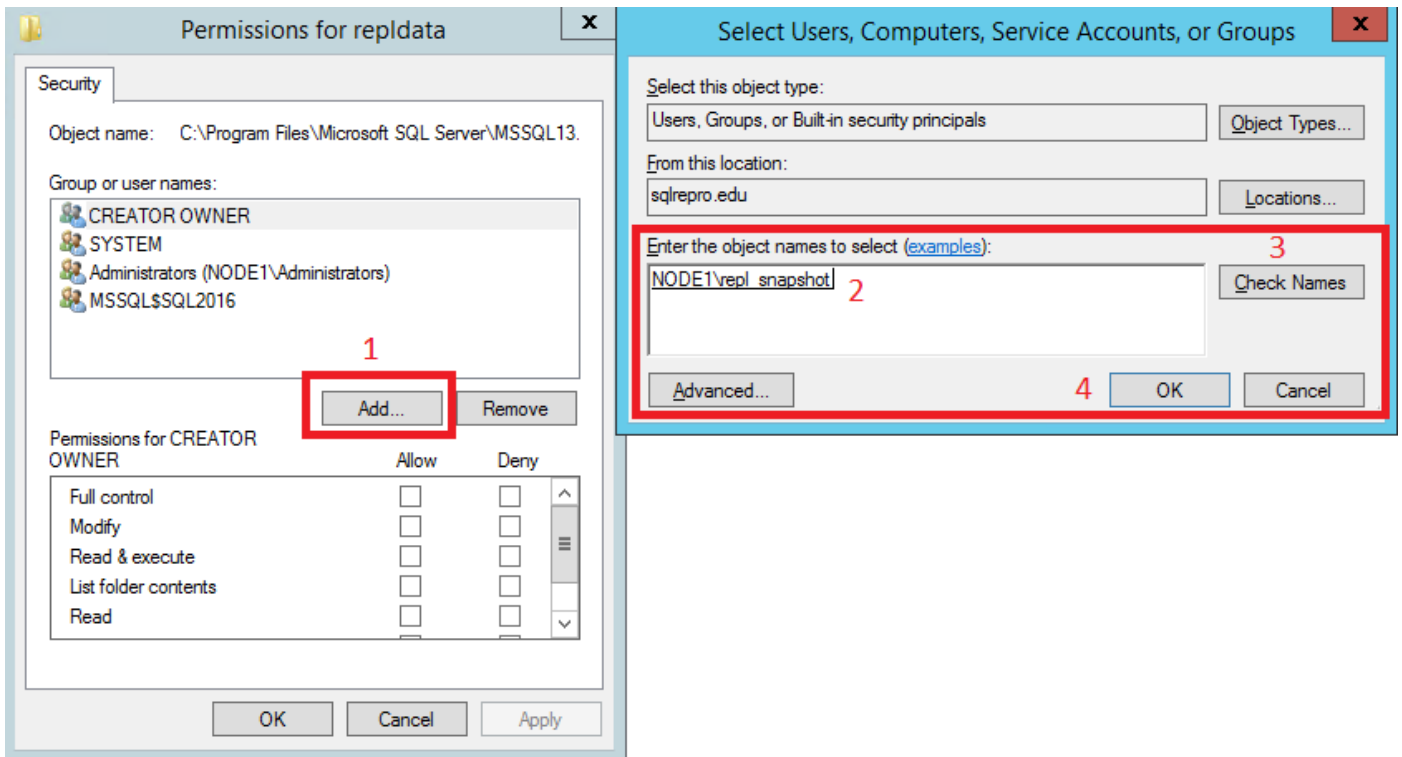
Lectura **repl_snapshot:** Control total



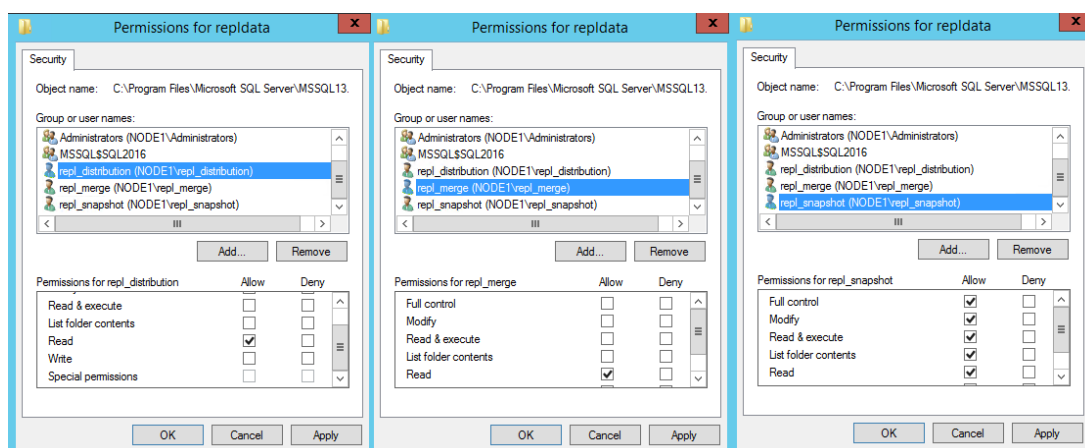
6. Vaya a pestaña seguridad en las propiedades de la carpeta **repl_data**, luego a edición:



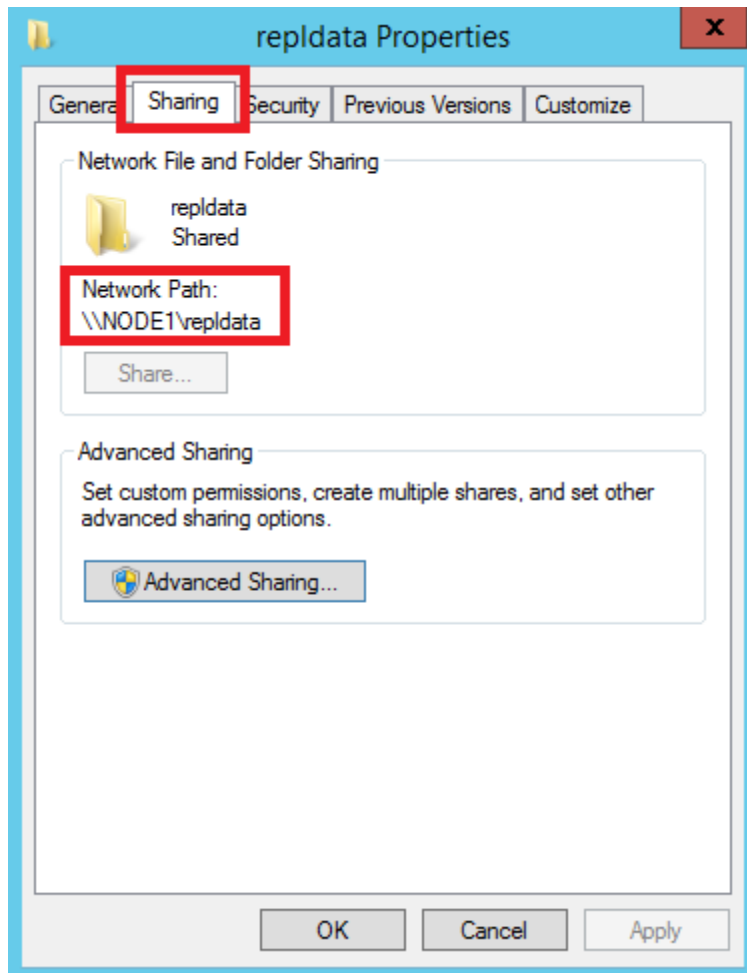
7. En el cuadro de diálogo Permisos de repldata, seleccione Agregar. En el cuadro de texto Seleccionar usuarios, equipos, cuentas de servicio o grupos, escriba el nombre de la cuenta del Agente de instantáneas creado anteriormente, como <Nombre_De_Equipo_Publicador>\repl_snapshot.



8. Repita el paso anterior para agregar permisos para el Agente de distribución, por ejemplo <Nombre_De_Equipo_Publicador>\repl_distribution y, para el Agente de mezcla, <Nombre_De_Equipo_Publicador>\repl_merge.
9. compruebe que se admiten los siguientes permisos:
- repl_distribution:** Lectura
- repl_merge:** Lectura
- repl_snapshot:** Control total



10. Vuelva a seleccionar la pestaña Compartir y anote la Ruta de acceso a la red para el recurso compartido. Necesitará esta ruta de acceso más adelante cuando configure la carpeta de instantáneas.



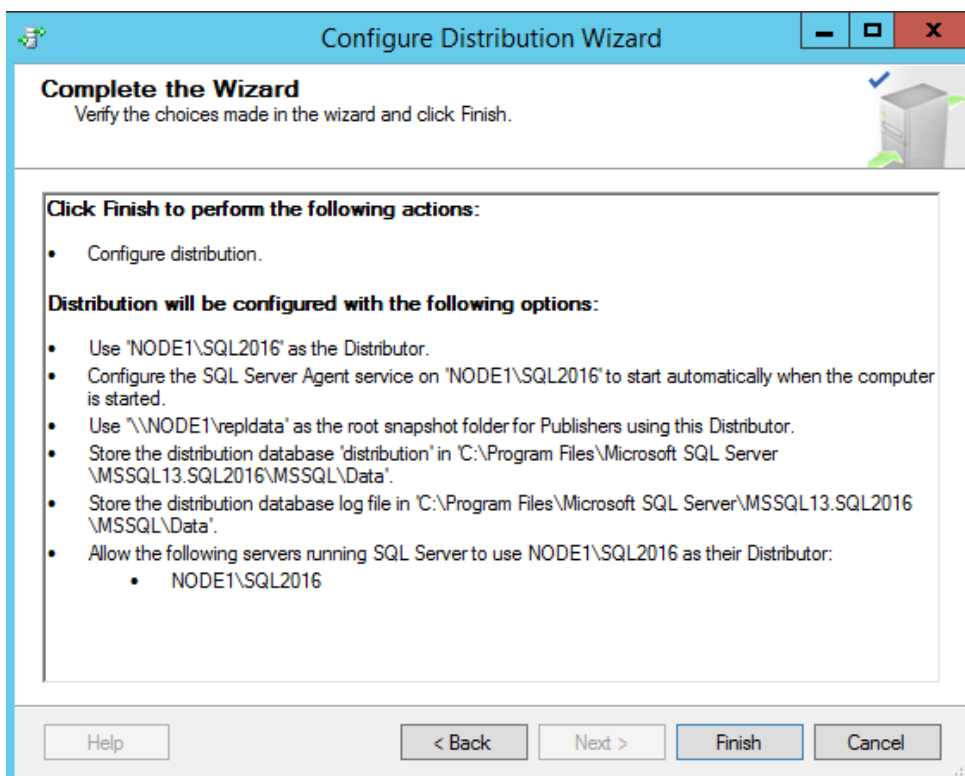
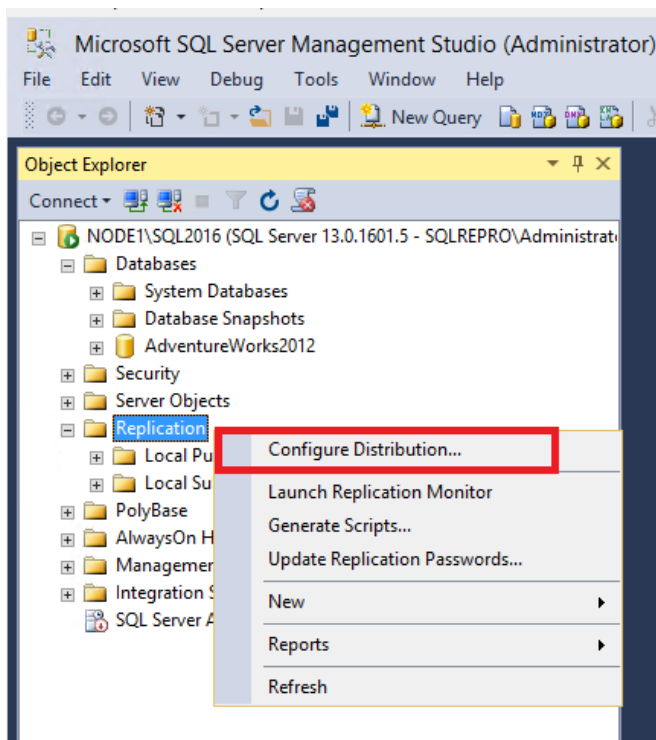
Configuración de distribución

En esta sección se va a configurar la distribución en el publicador y establecer los permisos requeridos en las bases de datos de publicación y distribución. Si ya ha configurado el distribuidor, debe deshabilitar la publicación y distribución antes de iniciar esta sección. No lo haga si debe mantener una topología de replicación existente, especialmente en producción.

Configuración de la distribución en el publicador

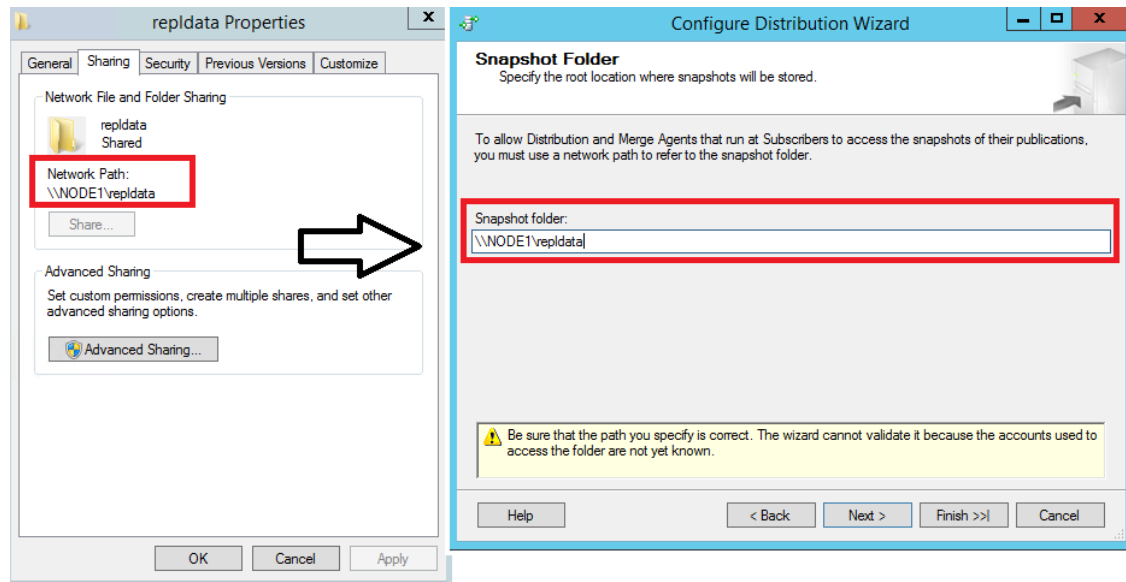
1. Conéctese al publicador en SQL Server Management Studio y, a continuación, expanda el nodo de servidor.

2. Haga clic con el botón derecho en la carpeta Replicación y luego seleccione
Configurar distribución:



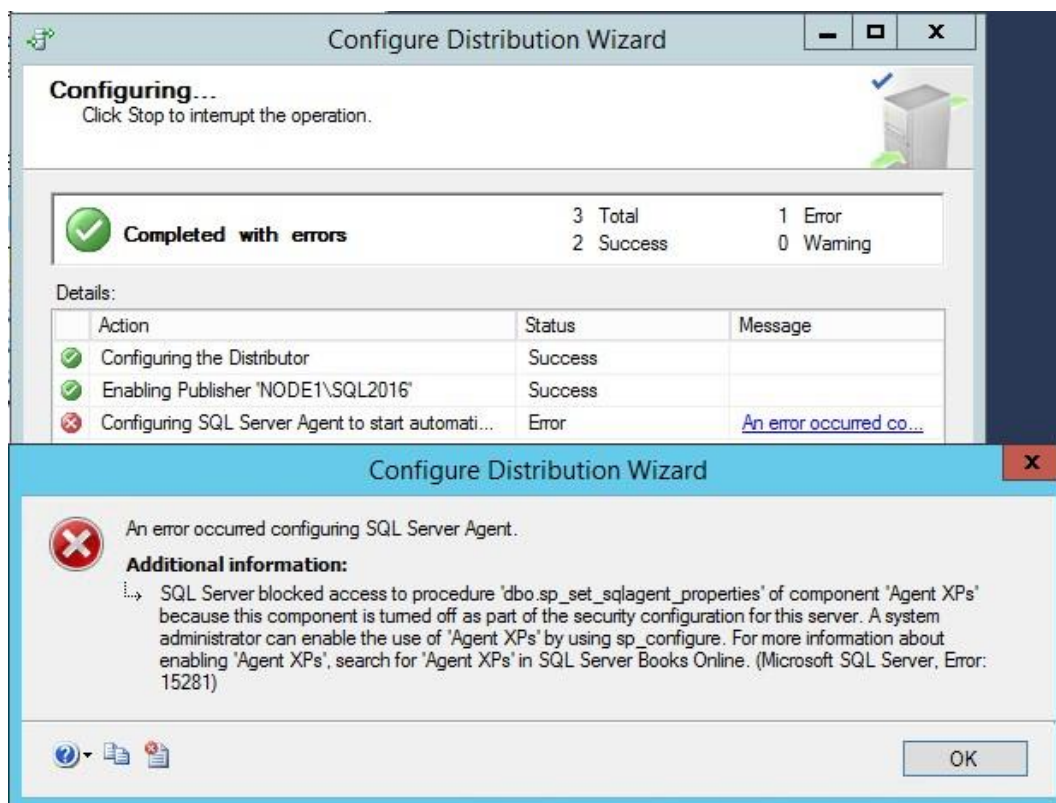
Nota: no conectarse al servidor como localhost, debe conectarse con el nombre real del servidor. Para más información vaya a la bibliografía.

3. En la página Distribuidor, seleccione <'NombreServidor'> actuará como su propio distribuidor; SQL Server creará una base de datos y un registro de distribución.

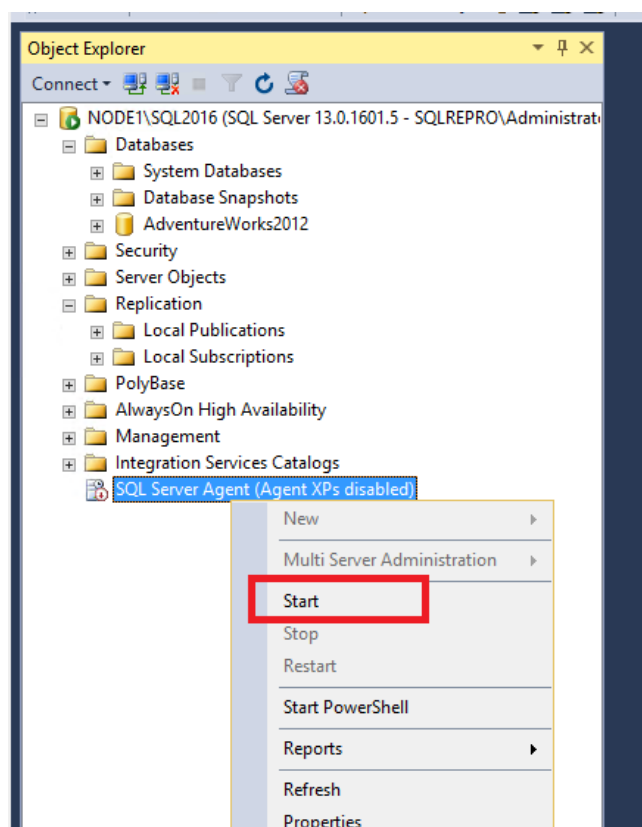


4. Si el Agente de SQL Server no se está ejecutando, en la página Inicio del Agente, seleccione Sí, configurar el servicio del Agente para que se inicie automáticamente. Seleccione Siguiente.
5. Escriba la ruta de acceso \\<Nombre_De_Equipo_Publicador>\repldata en el cuadro de texto Carpeta de instantáneas y, después, seleccione Siguiente. Esta ruta de acceso debe coincidir con lo que vimos anteriormente en Ruta de acceso a la red de la carpeta de propiedades de repldata después de configurar las propiedades del recurso compartido.
6. Acepte los valores predeterminados en las páginas restantes del asistente y finalmente seleccione finalizar.

El siguiente error puede aparecer al configurar el distribuidor. Es una indicación de que la cuenta que se utilizó para iniciar la cuenta del Agente SQL Server no es un administrador del sistema. Necesitará iniciar manualmente el Agente SQL Server, conceder dichos permisos a la cuenta existente o modificar la cuenta que utiliza el Agente SQL Server.



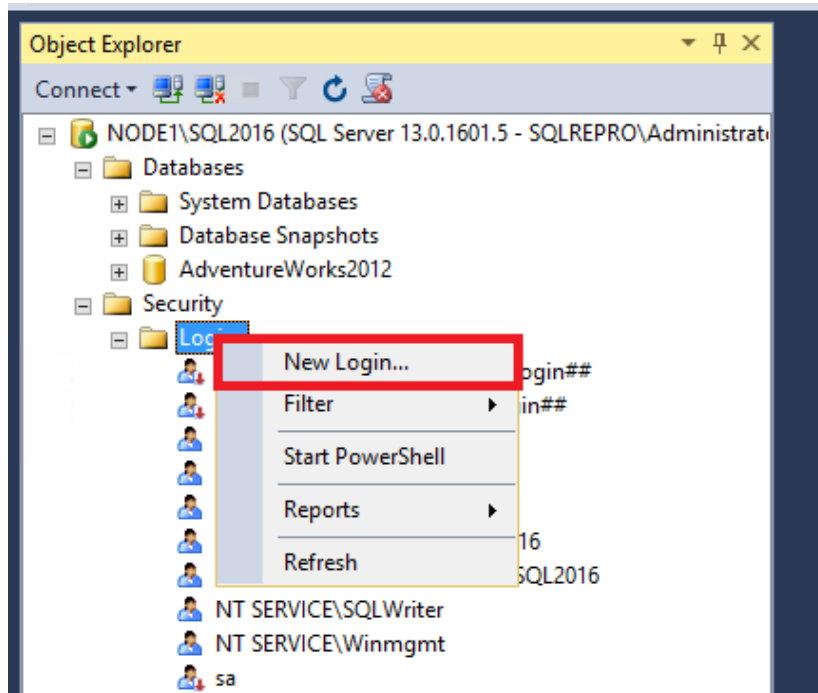
Si su instancia de SQL Server Management Studio se ejecuta con derechos administrativos, puede iniciar el Agente de SQL manualmente desde dentro de SSMS:



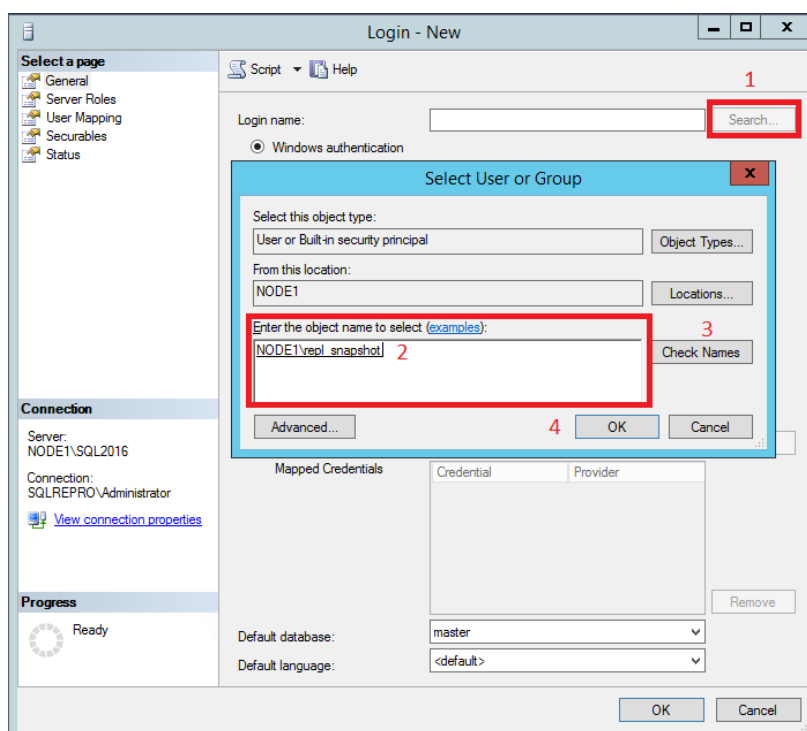
Nota: Si no se inicia el Agente SQL de forma visible, haga clic con el botón derecho en el Agente SQL Server en SSMS y seleccione Actualizar. Si está todavía en estado detenido, inícialo manualmente desde el Administrador de configuración de SQL Server.

Establecer los permisos para la base de datos

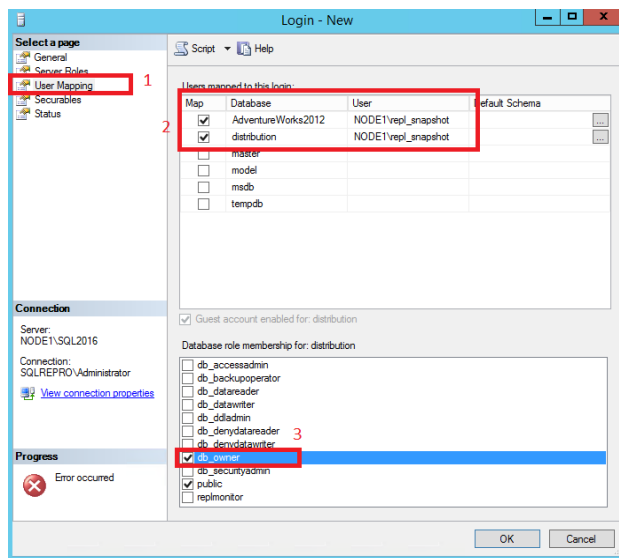
1. En SQL Server Management Studio, expanda Seguridad, haga clic con el botón derecho en Inicios de sesión y seleccione Nuevo inicio de sesión.



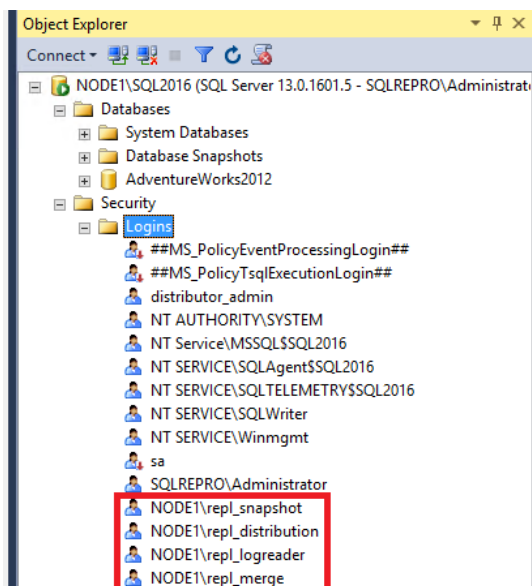
- 2.** En la página General, seleccione Buscar. Escriba `<>\repl_snapshot`.



3. En la página Asignación de usuarios, en la lista Usuarios asignados a este inicio de sesión, seleccione las bases de datos de distribución y de **base_consortio**. En la lista Pertenencia al rol de la base de datos, seleccione el rol **db_owner**.



4. Seleccione Aceptar para crear el inicio de sesión.
5. Repita los pasos del 1 al 4 para crear un inicio de sesión para el resto de cuentas locales (repl_distribution, repl_logreader y repl_merge). Estos inicios de sesión también se deben asignar a usuarios que son miembros del rol fijo de base de datos db_owner en las bases de datos distribution y base_consortio.



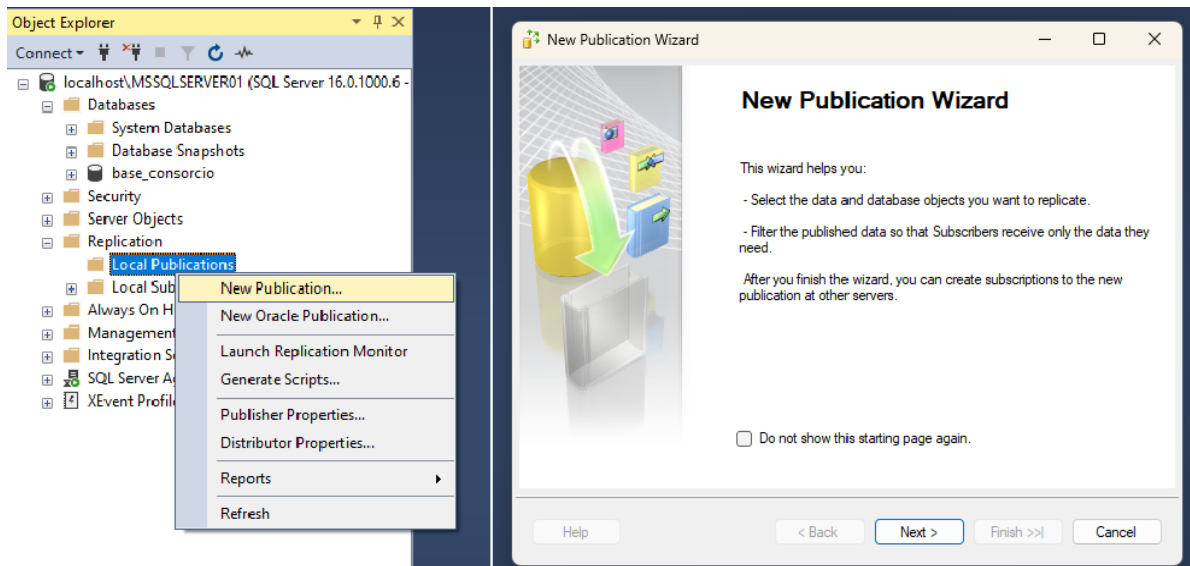
Configurar la replicación entre dos servidores conectados completamente (transaccional)

La replicación transaccional es un proceso crucial en bases de datos, donde los cambios hechos en un servidor de base de datos (llamado servidor principal o emisor) se copian automáticamente en otro servidor (llamado servidor réplica o receptor) para mantener los datos sincronizados y garantizar la disponibilidad y redundancia. Esta forma de replicación asegura que cada transacción realizada en el servidor principal se replique exactamente en el servidor réplica, manteniendo la consistencia de los datos.

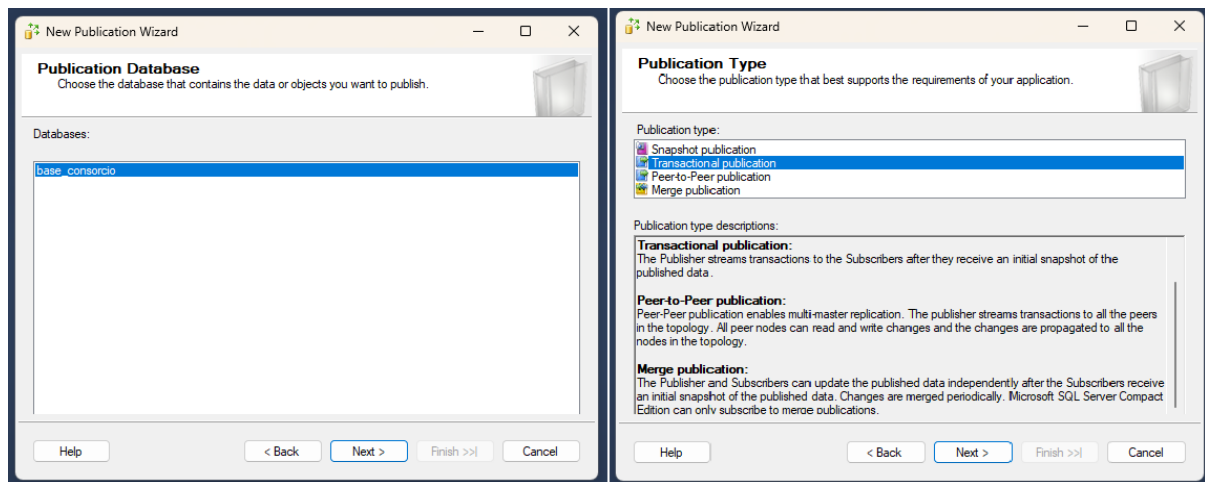
Para llevar a cabo el proceso de replicación, se empleará la herramienta SQL Server Management Studio (SSMS) junto con SQL Server como motor de la base de datos. Se utilizará una base de datos de ejemplo llamada "**base_consortio**".

Crear publicación y definir artículos:

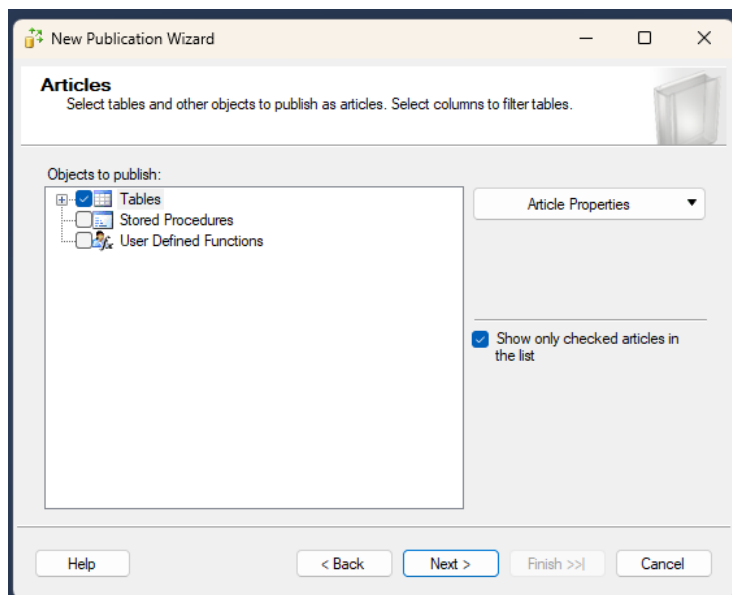
Paso 1: Abre SQL Server Management Studio (SSMS) y ve al directorio "**Replication**". Haz clic derecho en "**Local Publications**" y selecciona "**New Publication**". Esto abrirá el Asistente para la creación de la publicación.



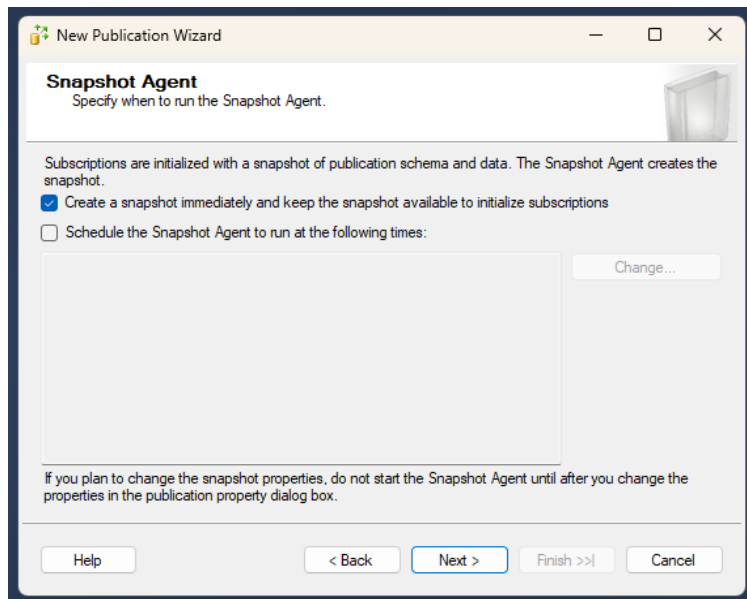
Paso 2: Selecciona la base de datos "**base_consortio**" y elige el tipo de publicación como "**Transactional Publication**". Haz clic en "**Next**".



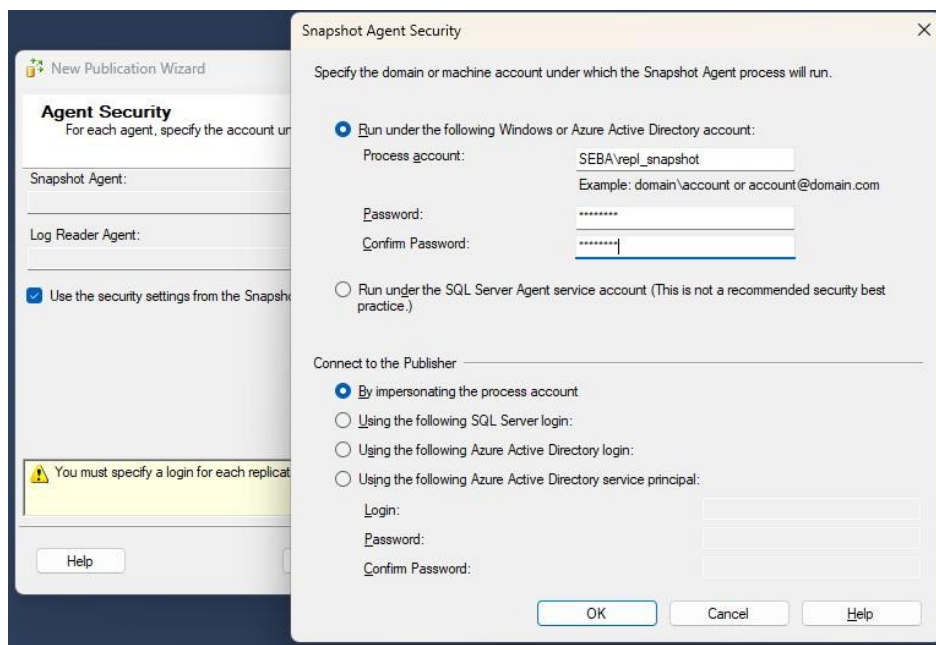
Paso 3: En la sección de **Articles** se encuentran las tablas y objetos que forman parte de la base de datos, para efectos prácticos seleccionaremos todas las tablas como artículos y daremos **Next**.



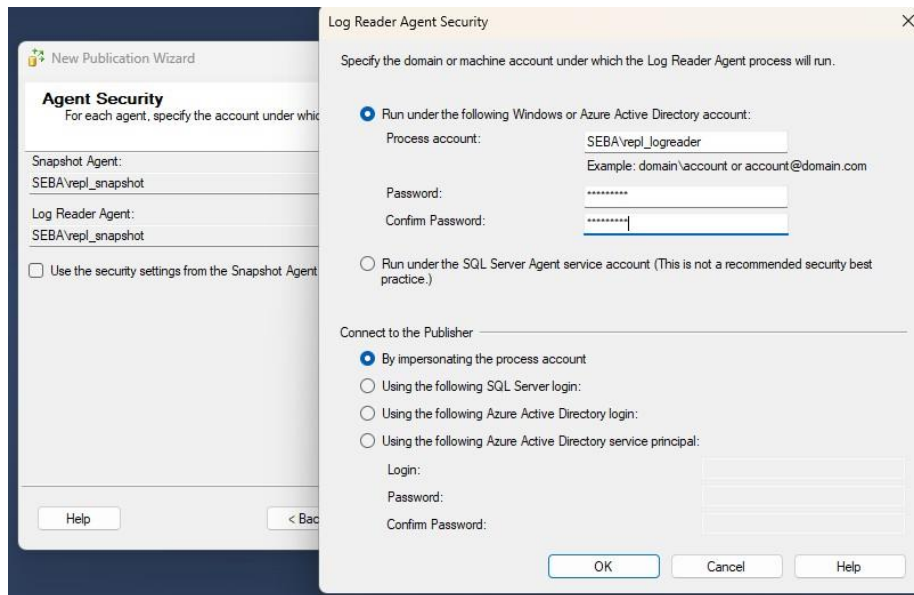
Paso 4: Ahora deberemos seleccionar la casilla que nos permitirá crear un Snapshot Agent y daremos **Next**.



Paso 5: Para crear el agente seleccionaremos el path del **repl_snapshot** ya creado con anterioridad y agregaremos una contraseña para luego guardar dando en **OK**



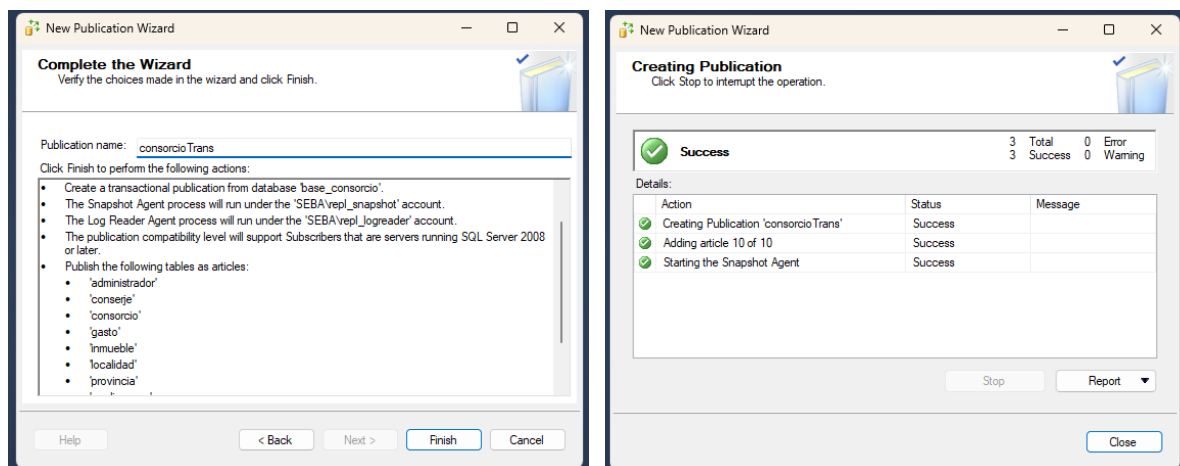
Paso 6: Configurar el nombre de la publicación y hacer clic en **"Finish"** para completar el asistente



Agente de distribución a la lista de acceso de la publicación (PAL)

En el directorio **Local Publications**, haga clic derecho en **consorcioTrans** y luego seleccione **Properties**. Aparece el cuadro de diálogo **Publication Properties**.

Paso 1: Seleccionar la página **Publication Access List** y seleccionar **Add**. **Paso 2:** En el cuadro de diálogo **Add Publication Access** daremos permisos **arepl_distribution** y seguidamente **OK**.

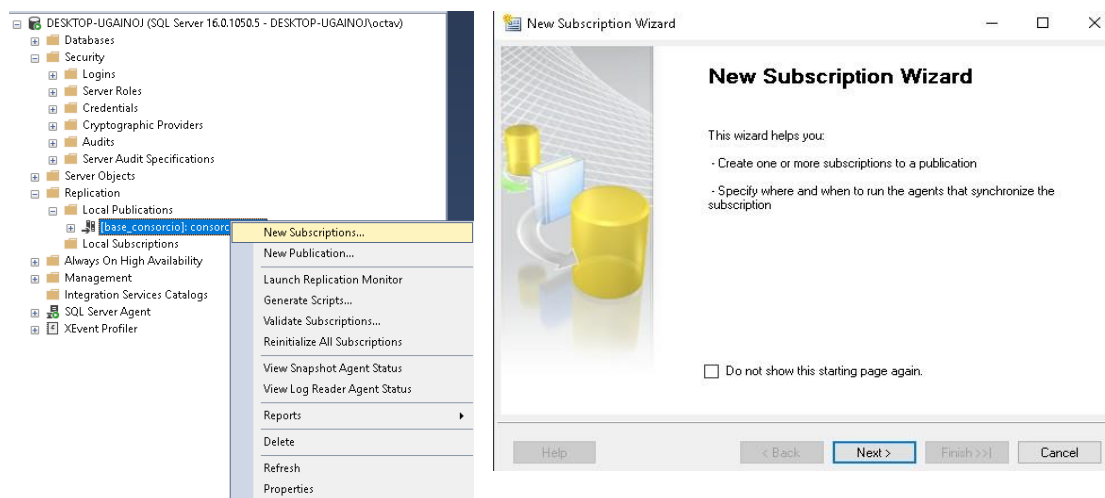


Creación de una suscripción a la publicación transaccional

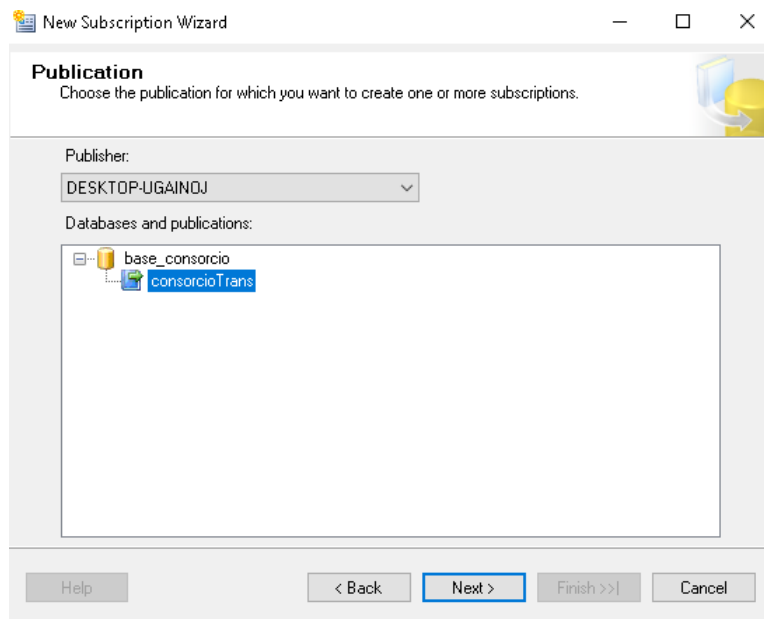
En esta sección agregamos un suscriptor que sería nuestra base de datos que va a recibir y aplicar los cambios realizados en el publicador(que sería nuestra base de datos principal que contiene los datos originales en este caso “**base_consorcio**”) que creamos y configuramos anteriormente en Crear publicaciones y definir artículos para publicar un conjunto de todas las tablas de nuestra **base_consorcio** mediante la replicación transaccional

Creación de la suscripción

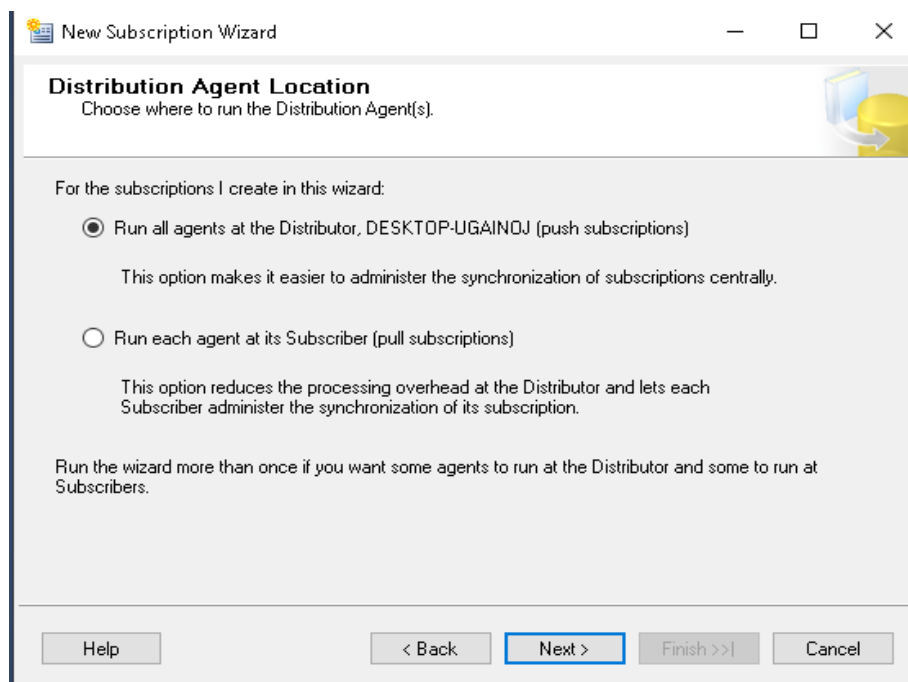
1. Debemos abrir el SQL Server Management Studio y conectarnos a nuestro servidor que aloja la base de datos publicadora “**base_consorcio**” y, a continuación, expanda la carpeta Replicación .
2. En la carpeta Publicaciones locales o Local Publications, haga clic con el botón derecho en la publicación “**[base_consorcio]:consorcioTrans**” que creamos anteriormente y, después, seleccione Nuevas suscripciones o new Subscription.



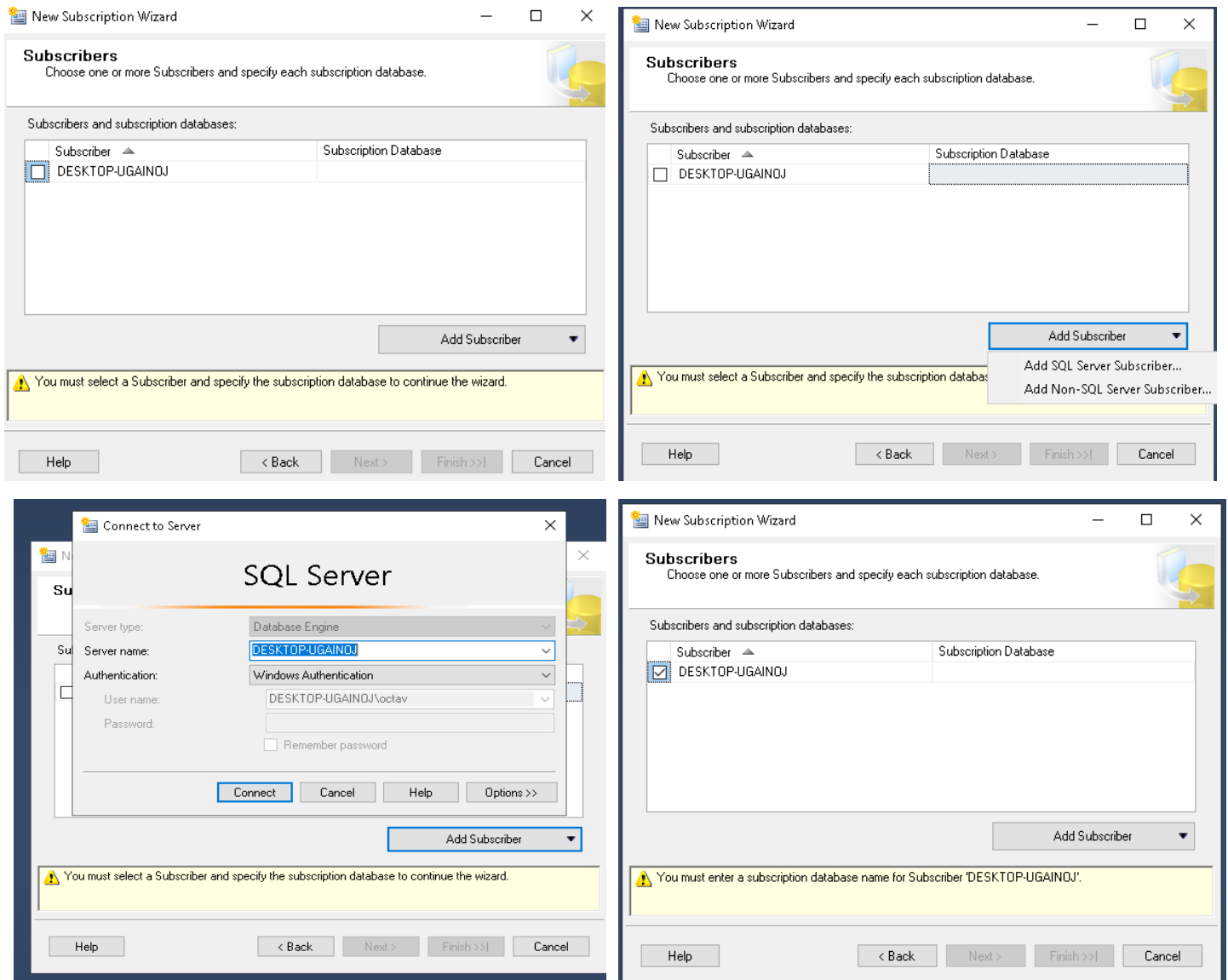
3. En la página Publicación, seleccione **consorcioTrans** (nuestra publicación) y, después, seleccione Siguiente:



4. En la página Ubicación del Agente de distribución, seleccione Ejecutar todos los agentes en el distribuidor y luego seleccione Siguiente. Esto hace que el agente de distribución se ejecute en el mismo servidor que el distribuidor, que es la base de datos replicada, esto se llama suscripción de inserción ya que el publicador o el distribuidor insertar los cambios en el suscriptor



5. En la Página de Suscriptores o New Subscription Wizard sino se muestra el nombre de la instancia del suscriptor ,debemos de Agregar el suscriptor de nuestra publicación que creamos anteriormente Seleccione Agregar suscriptor y, después, seleccione Agregar suscriptor de SQL Server en la lista desplegable.



6. Ahora Seleccionamos en Subscription Database nueva base de datos para crear una base de datos vacía en el suscriptor que almacenará los datos replicados, y escribimos dentro del cuadro de diálogo que se nos abrió el nombre de la base de datos de suscripción en este caso **ConsorcioReplica** y le damos en aceptar y en siguiente

New Subscription Wizard

Subscribers

Choose one or more Subscribers and specify each subscription database.

Subscribers and subscription databases:

Subscriber	Subscription Database
<input checked="" type="checkbox"/> DESKTOP-UGAINQJ	<div><New database...> <Refresh database list> base_consortio BD_CellMarket</div>

Add Subscriber

You must enter a subscription database name for Subscriber 'DESKTOP-UGAINQJ'.

Help

< Back

Next >

Finish >>

Cancel

New Database

Select a page

GeneralOptionsFilegroups

ScriptHelp

Database name:ConsortioReplica

Owner:<default>

☒ Use full-text indexing

Database files:

Logical Name	File Type	Filegroup	Initial Size (MB)	Autogrowth / Maxsize	Pa
ConsortioR...	ROWS...	PRIMARY	8	By 64 MB, Unlimited	C:
ConsortioR...	LOG	Not Applicable	8	By 64 MB, Unlimited	C:

Connection

Server:DESKTOP-UGAINQJ

Connection:DESKTOP-UGAINQJ\octav

[View connection properties](#)

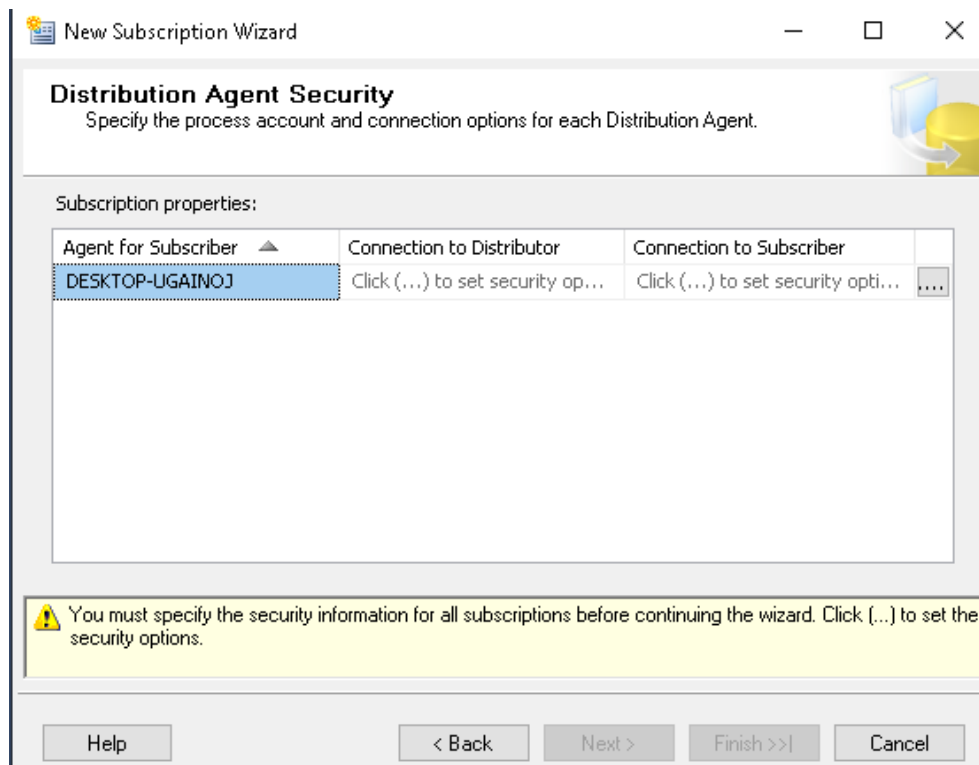
Progress

Ready

OKCancel

- El Agente de distribución es el componente de replicación que mueve los cambios desde el distribuidor al suscriptor.

7. Entonces Haga clic en el botón de puntos suspensivos (...) para abrir el cuadro de diálogo Configuración de seguridad del Agente de distribución



8. Escriba <Publisher_Machine_Name>\repl_distribution y seleccione Aceptar paracerrar el cuadro de diálogo y guardar la configuración.

Distribution Agent Security

Specify the domain or machine account under which the Distribution Agent process will run when synchronizing this subscription.

☒ Run under the following Windows or Azure Active Directory account:

Process account:
Example: domain\account or account@domain.com

Password:

Confirm Password:

☐ Run under the SQL Server Agent service account (This is not a recommended security best practice.)

Connect to the Distributor

☒ By impersonating the process account

☐ Using a SQL Server login

The connection to the server on which the agent runs must impersonate the process account. The process account must be a member of the Publication Access List.

Connect to the Subscriber

☒ By impersonating the process account

☐ Using the following SQL Server login:

☐ Using the following Azure Active Directory login:

☐ Using the following Azure Active Directory service principal:

Azure Active Directory service principal UUID:

Azure Active Directory service principal secret:

Confirm Azure Active Directory service principal secret:

The login used to connect to the Subscriber must be a database owner of the subscription database.

OK Cancel Help

9. Seleccione Finalizar para aceptar los valores predeterminados en las páginas restantes y finalizar el asistente.

New Subscription Wizard

Creating Subscription(s)...

Click Stop to interrupt the operation.

2 Remaining

Action	Status	Message
Creating subscription for 'DESKTOP-UGAINOJ'	In progress...	
Starting the Snapshot Agent		

Stop Report

Close

New Subscription Wizard

Creating Subscription(s)...

Click Stop to interrupt the operation.

Success

Action	Status	Message
Creating subscription for 'DESKTOP-UGAINOJ'	Success	
Starting the Snapshot Agent	Success	

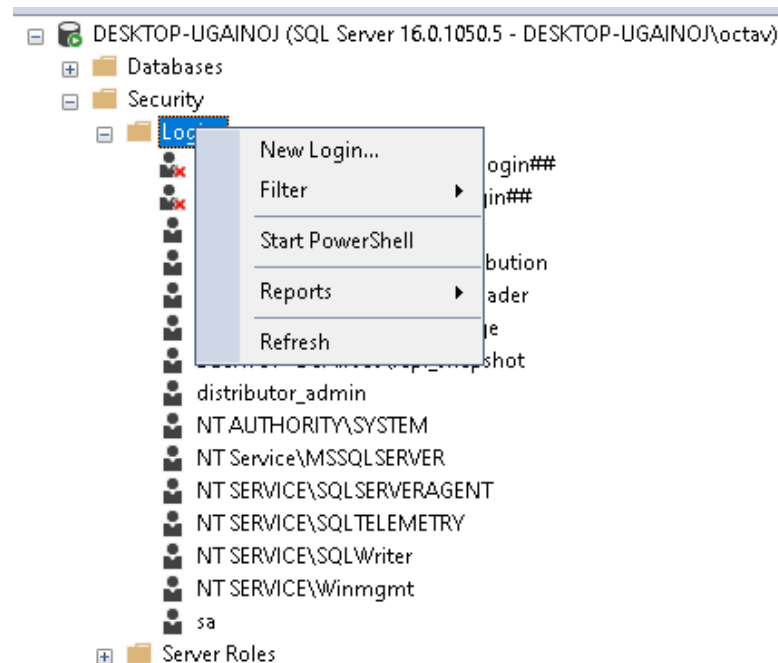
Stop Report

Close

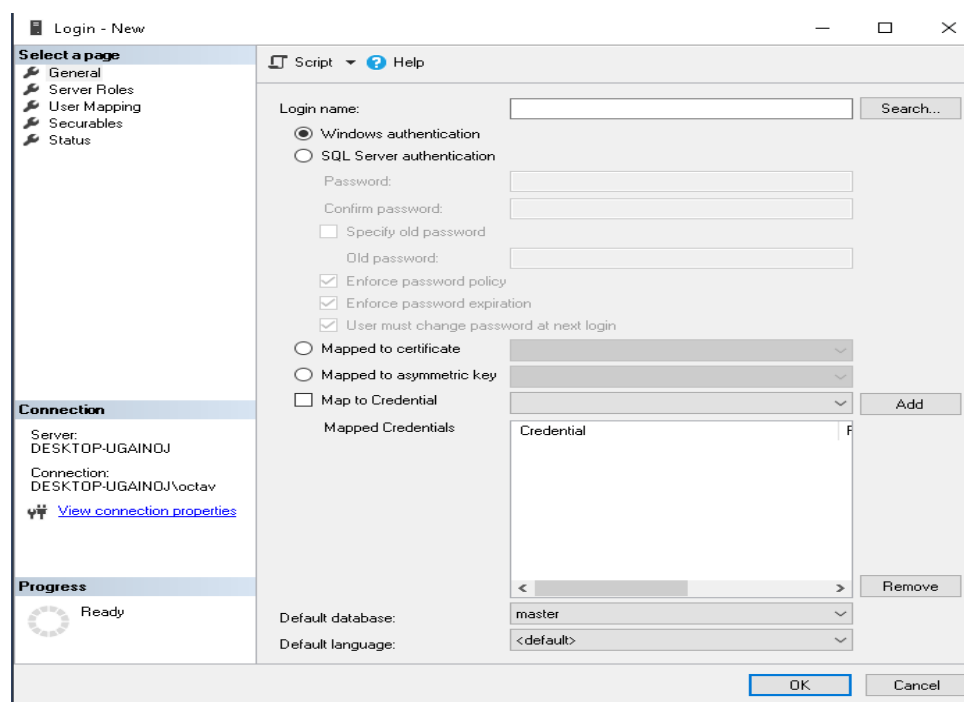
Establecer permisos de base de datos en el suscriptor

Debemos de conectarnos al suscriptor en SQL server que es el servidor que recibe los datos replicadores del publicador.

- Expanda la carpeta Seguridad, haga clic con el botón derecho en la carpeta Inicios de sesión(login) y seleccione Nuevo inicio de sesión(new login) para crear un nuevo inicio de sesión de SQL Server esto nos va a permitir autenticar la conexión y garantizar que solo los usuario autorizados puedan acceder a la base de datos
- En la página General, en Nombre de inicio de sesión, seleccione Buscar (search)yagregue el inicio de sesión de <Subscriber_Machine_Name>\repl_distribution.



- En la página Asignaciones de usuario, otorgue el inicio de sesión al miembro db_owner para la base de datos **ConsorcioReplica**, que es la base de datos que almacena los datos replicados



Login Properties - DESKTOP-UGAINOJ\repl_distribution

Select a page

General

Server Roles

User Mapping

Securables

Status

Connection

Server:
DESKTOP-UGAINOJ

Connection:
DESKTOP-UGAINOJ\octav

View connection properties

Progress

Ready

Script Help

Users mapped to this login:

Map	Database	User	Default Schema
<input checked="" type="checkbox"/>	base_consortio	DESKTOP-UGAINOJ\re...	
<input type="checkbox"/>	BD_CeliMarket		
<input checked="" type="checkbox"/>	ConsortioReplica	DESKTOP-UGAINOJ\re...	
<input checked="" type="checkbox"/>	distribution	DESKTOP-UGAINOJ\re...	dbo
<input type="checkbox"/>	master		
<input type="checkbox"/>	model		
<input type="checkbox"/>	msdb		
<input type="checkbox"/>	tempdb		

☐ Guest account enabled for: ConsortioReplica

Database role membership for: ConsortioReplica

☐ db_accessadmin

☐ db_backupoperator

☐ db_datareader

☐ db_datawriter

☐ db_ddladmin

☐ db_denydatareader

☐ db_denydatawriter

☒ db_owner

☐ db_securityadmin

☒ public

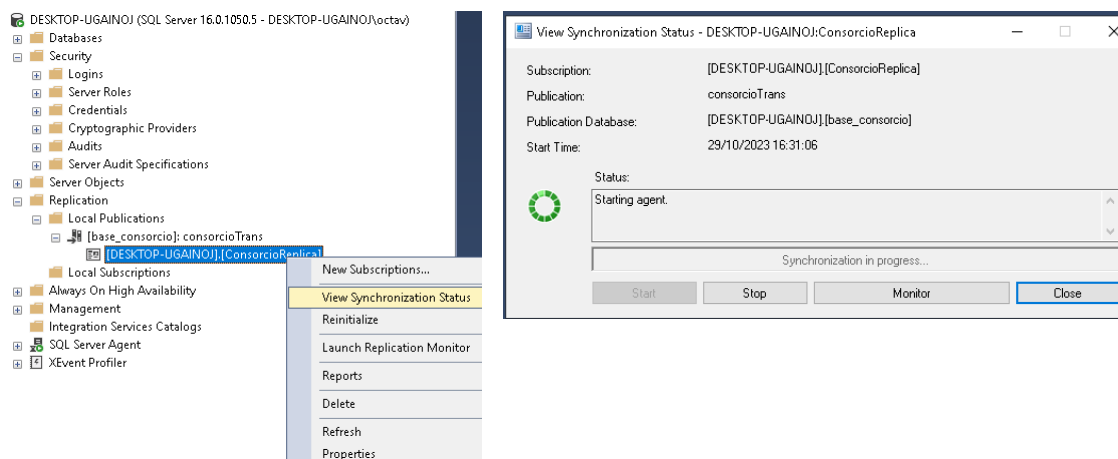
OK

Cancel

Ver el estado de sincronización de la suscripción

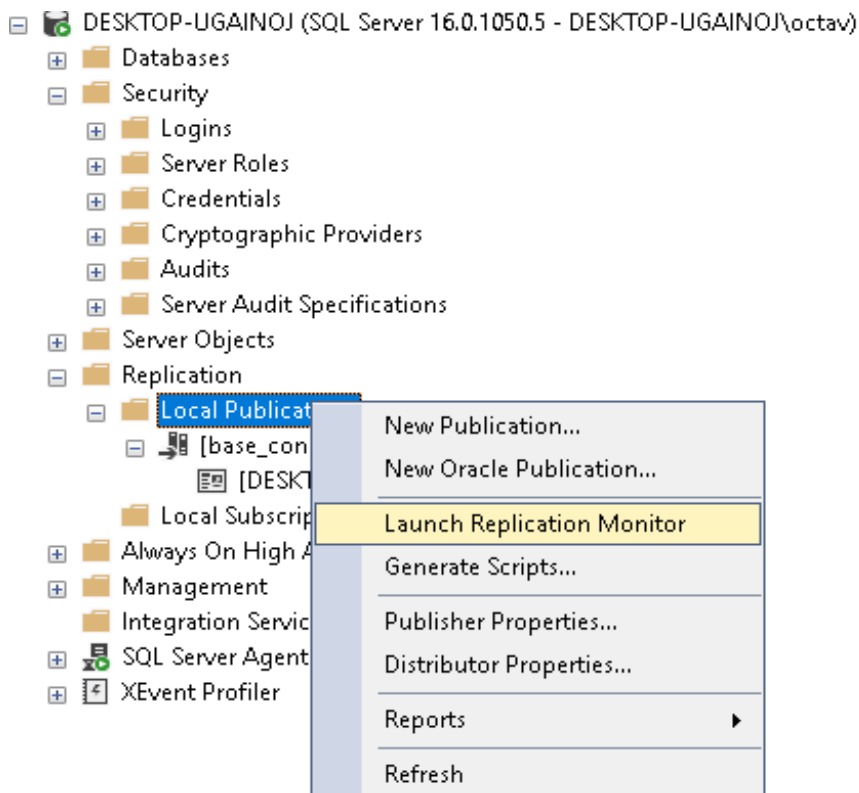
En esta parte vamos a ver el estado de sincronización de la suscripción que ha creado anteriormente. El estado de sincronización muestra información sobre el rendimiento y la actividad de la replicación, como la latencia, el número de cambios replicados y los posibles errores.

- Conéctese al publicador en SQL Server Management Studio. Expanda el nodo del servidor y luego la carpeta Replicación.
- Expanda la carpeta replicación que contiene las opciones de configuración y administración de la replicación, luego en la carpeta publicaciones locales expanda la publicación **consorcioTrans** que es la publicación transacción que creamos.



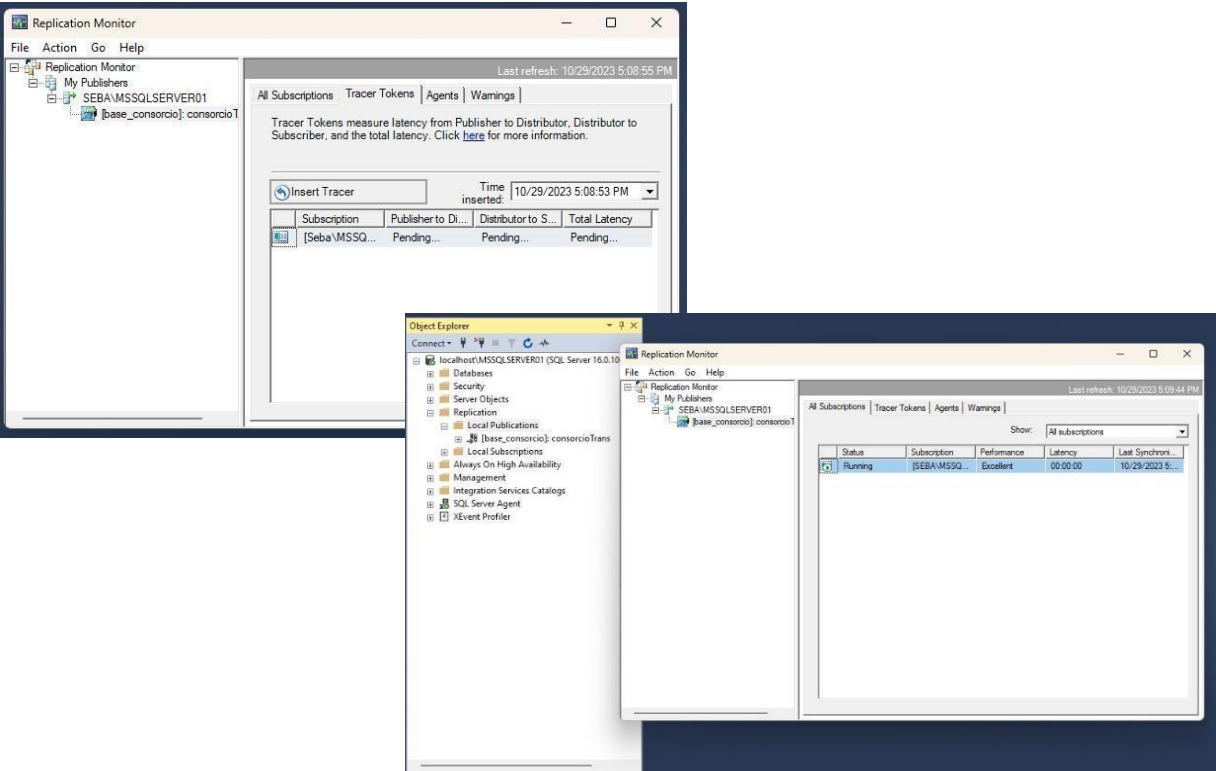
En esta sección se emplean testigos de seguimiento para comprobar que los cambios se replican en el suscriptor y para determinar la latencia. La latencia es el tiempo necesario para que un cambio realizado en el publicador aparezca en el suscriptor.

1. Conéctese al publicador en SQL Server Management Studio. Expanda el nodo del servidor, haga clic con el botón derecho en la carpeta Replicación y luego seleccione Iniciar Monitor de replicación:



2. Expanda un grupo de publicador en el panel izquierdo, expanda la instancia del publicador y, después, seleccione la publicación **consorcioTrans**
 - Seleccione la pestaña Testigos de seguimiento.
 - Seleccione Insertar seguimiento

- Vea el tiempo transcurrido para el testigo de seguimiento en las siguientes columnas: Publicador a distribuidor, Distribuidor a suscriptor y Latencia total.



Conclusión

Como conclusión podríamos decir que la restauración de los datos (Restore) y la copia de seguridad (Backup) están muy relacionada una con la otra dado que no nos sirve realizar una copia de seguridad si no vamos a restaurar dichos datos en algún momento determinado y para realizar dicha restauración debemos tener una copia de seguridad válida para ello. De esta manera siempre que se haga una copia de seguridad vamos a tener la seguridad de poder volver a dichos datos ante cualquier tipo de complicación o accidente, cabe mencionar que volveremos a obtener los datos de la última copia de seguridad realizada, por esto, es recomendable realizar copias de seguridad de manera constante para de esta manera una vez al momento de realizar una restauración poder obtener los datos más cercanos al momento del error o inconveniente que se produjo.

Bibliografía

[Microsoft – Documentación SQL:](#)

<https://learn.microsoft.com/es-es/sql/relational-databases/backup-restore/back-up-and-restore-of-sql-server-databases?view=sql-server-ver16>

<https://learn.microsoft.com/en-us/sql/relational-databases/backup-restore/backup-overview-sql-server?view=sql-server-ver16>

<https://learn.microsoft.com/en-us/sql/relational-databases/tutorial-sql-server-backup-and-restore-to-azure-blob-storage-service?view=sql-server-ver16&tabs=SSMS>

<https://learn.microsoft.com/es-es/sql/t-sql/statements/backup-transact-sql?view=sql-server-ver16>

[SQLShack:](#)

<https://www.sqlshack.com/es/respaldar-y-restaurar-una-base-de-datos-sql-server-usando-multiples-archivos/>

[SQL Server Ya – Ejemplos Prácticos:](#)

<https://www.tutorialesprogramacionya.com/sqlserverya/>