2019

Manual SQL Server –Transact SQL Básico /Avanzado

"Proyecto realizado con financiamiento de la Secretaría de Educación Pública-Subsecretaría de Educación Superior-Dirección General de Educación Superior Universitaria", CONVENIO No.: 2018-19008-146.

Participantes:

Claudia Isabel Degollado Amaya

Claudia Tovar González

David Rey García Cedillo

Felipe Tijerina Martínez

José Manuel Chávez García

José Luis Olivares Cerda

Karen Janeth Buendía Cisneros

Martha Cecilia Rodríguez Martínez

Miriam Janeth Lizárraga Trewarta

Rosa Isela Hernández Salinas

Ricardo Viera Tamez

Sergio Ulises Galván Contreras

ÍNDICE DE CONTENIDO

ÍNDICE DE CONTENIDO 2
ÍNDICE DE FIGURAS 5
INTRODUCCIÓN 8
¿Qué es una base de datos? 1
Como crear una base de datos 1
archivos de la base de datos 1
Nombres de archivo lógico y físico 2
RESPALDO Y RESTAURACIÓN DE UNA BASE DE DATOS 3
EJERCICIO # 1 9
Planes de mantenimiento 10
Lenguaje de manipulacion de datos 12
EJERCICIO # 2 13
Sentencia select 14
joins 14
Union 16
group by y order by 17
EJERCICIO # 3 18
Insert 19
Update 20
Delete 21
EJERCICO # 4 22
merge 23

Tablas Temporales 26
variables tipo tablas 27
EJERCICIO # 5 28
Tipo de datos Tabla 29
Consultas ctes 30
ctes recursivas 31
EJERCICIO # 6 32
funciones del SQL server 33
EJERCICIO # 7 39
lenguaje de control de flujo 40
BEGIN… END 40
IF-ElSE 41
WHILE 41
TRY… CATCH 42
RAISERROR 43
THROW 44
CASE 45
EJERCICIO # 8 46
funciones definidas por el usuario 47
funcion escalar 47
Función con valor de tabla 48
Uso de Funciones 48
Modificar funciones 49
Eliminar una Función 49
Procedimientos Almacenados 49

EJERCICIO # 9 52
Servicios en la nube 53
¿Qué es Azure? 54
Modelos de gestión de Azure 56
EJERCICIO # 10 62
Reporting Services 63
EJERCICIO # 11 69
Integration Services 70
EJERCICIO # 12 72

ÍNDICE DE FIGURAS

Figura 1 Representación de una base de datos..... **¡Error! Marcador no definido.**
Figura 2 Script para crear una base de datos 2
Figura 3 Script para modificar una BD 3
Figura 4 Script para realizar un respaldo completo de BD 4
Figura 5 Back Up por asistente 4
Figura 6 Destino donde se guardara el Back Up 5
Figura 7 Representación de un respaldo diferencial 5
Figura 8 Script para realizar un respaldo diferencial 6
Figura 9 Selección tipo Back Up 6
Figura 10 Representación de los tipos de respaldo 7
Figura 11 Representación de restauración de back Up 8
Figura 12 Scripts de los tipos de restauración de back up 8
Figura 13 Seguimiento de uso del plan de mantenimiento 11
Figura 14 Automatización para la programación de respaldos 11
Figura 15 Sintaxis del comando select 14
Figura 16 Representación gráfica de los datos que considera el inner 15
Figura 17 Sintaxis de aplicabilidad del inner 15
Figura 18 Sintaxis de ejemplos del comando union 16
Figura 19 Sintaxis de uso de funciones agregadas 17
Figura 20 Sintaxis de uso para ordenar los datos 17
Figura 21 Sintaxis para agregar registros en una tabla 19
Figura 22 Ejemplo de diferentes usos de agregado de datos 19
Figura 23 Sintaxis para modificar datos en un registro 20
Figura 24 Ejemplo de uso con sintaxis en update 20
Figura 25 Ejemplo de aplicación en comando delete 21
Figura 26 Sintaxis de aplicabilidad en el comando marge 23
Figura 27 Ejemplo de uso del comando merge 24
Figura 28 Combinación de tablas con otra para modificar datos 25
Figura 29 Combinación de tablas para eliminar y modificar tablas 25

Figura 30 Sintaxis para crear tablas temporales y agregar datos en tabla 26
Figura 31 Ejemplo de uso para realizar variable tipo tabla y agregar datos 27
Figura 32 Sintaxis aplicada para realizar un tipo de dato tabla y un procedimiento almacenado 29
Figura 33 Ejemplo de consulta de CTE´s 30
Figura 34 Sintaxis para anidar dos o más consultas CTE´s 30
Figura 35 Sintaxis de CTE´s recursivas 31
Figura 36 Uso de las funciones agregadas 34
Figura 37 Sintaxis de funciones agregadas 34
Figura 38 Descripción de Funciones de validación de errores 35
Figura 39 Descripción de Funciones tipo fecha 35
Figura 40 Descripción de uso de las funciones de cadena parte 1 36
Figura 41 Descripción de uso de las funciones de cadena parte 2 37
Figura 42 Descripción de uso de validación de errores de cadena 38
Figura 43 Sintaxis de control de flujo por sentencias 42
Figura 44 Sintaxis de validación de errores por try-catch 43
Figura 45 Sintaxis validación por throw 44
Figura 46 Sintaxis de la sentencia case-when 45
Figura 47 Sintaxis para crear funciones con consultas 47
Figura 48 Ejemplo para crear funciones con valor de tabla en funciones agregadas 48
Figura 49 Ejemplo para realizar cambios en funciones 49
Figura 50 Ejemplo de creación de procedimiento almacenado 50
Figura 51 Ejemplo para modificar un procedimiento almacenado con función case 51
Figura 52 Representación gráfica de la nube en plataforma azure 53
Figura 53 Servicios de la nube en plataforma azure 54
Figura 54 Arquitectura de la nube azure 55
Figura 55 Servicios ofrecidos en las diferentes plataformas 56
Figura 56 Ejemplificación de servidores en la nube azure 56
Figura 57 Página de inicio para usuarios de azure 57

Figura 58 Recursos creados en plataforma azure 58
Figura 59 Página principal de usuarios en plataforma azure 58
Figura 60 Tabla comparativa de servicios en azure 59
Figura 61 Datos de contacto para dudas en la plataforma azure 59
Figura 62 Pantalla para crear base de datos por asistente en plataforma azure .. 60
Figura 63 Creación de base de datos en la nube en plataforma azure 60
Figura 64 Creación de tablas por asistente en la nube en plataforma azure 61
Figura 65 Creación de informe de servicios por asistente 64
Figura 66 Especificación del origen de uso de los datos 64
Figura 67 Pasos para crear la conexión del repositorio de datos 65
Figura 68 Agregar nuevo repositorio de datos 65
Figura 69 Creación de Reportes 66
Figura 70 Definición de destino de la extracción de datos en el reporte 66
Figura 71 Creación de una tabla con objetos 67
Figura 72 Vista previa del contenido de la tabla 67
Figura 73 Visualización previa de la creación del reporte 68
Figura 74 Representación gráfica de la integración de servicios 70

INTRODUCCIÓN

En este manual aprenderemos como respaldar, restaurar y saber cómo está formada físicamente una base de datos, también conoceremos sobre los planes de mantenimiento, sobre el lenguaje de manipulación de datos (DML) para ello se requieren, conocimientos básicos del lenguaje de definición de datos (DDL) para poder comprender algunos temas. Se darán a conocer los diferentes tipos de tablas, así como las funciones que se realizar con ellas, podremos aprender cómo aplicar las estructuras de control dentro de las bases de datos, y reconocer los diferentes tipos de manejadores de errores. Por consiguiente, otro de los temas aplicados son los servicios en la nube de la base de datos en la plataforma de Azure Microsoft.

¿QUÉ ES UNA BASE DE DATOS?

Una base de datos es una colección de información organizada de forma que un programa de ordenador pueda seleccionar rápidamente los fragmentos de datos que necesite, una base de datos es un sistema de archivos electrónico.

Las bases de datos tradicionales se organizan por campos, registros y archivos, un campo es una pieza única de información; un registro es un sistema completo de campos y un archivo es una colección de registros. Por ejemplo, una guía de teléfono es análoga a un archivo, contiene una lista de registros cada uno de los cuales consiste en tres campos: nombre, dirección y número de teléfono.

COMO CREAR UNA BASE DE DATOS

ARCHIVOS DE LA BASE DE DATOS

Existen tres tipos de archivos para agruparlos que son:

- **Principal o Main Data File** con extensión (.mdf) es el archivo de datos principal donde se encuentra la información inicial de la base de datos, en este archivo se almacenan los datos y los objetos del usuario, aunque también se puede realizar en archivos secundarios, toda base de datos cuenta con un archivo principal.
- **Secundarios o Secondary Data File** con extensión (.ndf) son los archivos de datos secundarios donde la base de datos puede contener uno o varios, se pueden usar opcional, están definidos por el usuario y almacenan los datos de los usuarios, se pueden utilizar para repartir la información en varios discos agregando cada archivo en una unidad de disco diferente, además si una base de datos supera el límite de almacenamiento para un archivo de Windows se podrán utilizar archivos de datos secundarios para el crecimiento de la misma.
- **De registro o Log Data File** con extensión (.log) son archivos del registro de transacciones, contiene la información de se utiliza para la recuperación

de la base de datos cada BD debe contar con al menos un archivo de registro.

NOMBRES DE ARCHIVO LÓGICO Y FÍSICO

Existen dos tipos de nombres de archivo que son:

logical_file_name es el nombre que le da para hacer referencia al archivo físico en todas las instrucciones Transact-SQL, el nombre del archivo lógico debe cumplir con las reglas de identificador de SQL Server, su nombre debe ser único.

Os_file_name: es el nombre que se le da para hacer referencia al archivo físico que incluye la ruta para ingresar al directorio, el nombre del archivo debe seguir las reglas para nombres de archivos del sistema operativo.

Para crear una base de datos es necesario realizar el script que se muestra en la siguiente figura:

```
CREATE DATABASE TickedDB
ON
    ( NAME = TickedDB,
     FILENAME = 'C:\DB\TickedDB.mdf',
     SIZE = 10,
     MAXSIZE = 50,
     FILEGROWTH = 5
    )
LOG ON
     ( NAME = TickedDB_log,
       FILENAME = 'C:\Log\TickedDB_log.ldf',
       SIZE = 5MB,
       MAXSIZE = 25MB,
       FILEGROWTH = 5MB
      );
```

Figura 1 Script para crear una base de datos

Se puede observar en el siguiente script la creación de una base de datos con un tipo de archivo lógico y físico.

NAME: hace referencia al nombre lógico de la base de datos

FILENAME: ruta donde se encuentra físicamente el archivo

SIZE: tamaño que soporta la base de datos

MAXSIZE: tamaño máximo de la base de datos

FILEGROWTH: crecimiento automático de los archivos de la base de datos

***NOTA**: recuerden se ingresan dos veces debido a que la primera hace referencia al archivo primario y la otra es del archivo de registro.*

Para modificar un BD es necesario realizar el Script que se muestra en la siguiente figura:

```
ALTER DATABASE TickedDB
      ADD FILEGROUP GroupFile_1;
GO

ALTER DATABASE TickedDB
ADD FILE
     ( NAME = Data_1,
      FILENAME = 'C:\DB\Data_1.ndf',
      SIZE = 5MB,
      MAXSIZE = 100MB,
      FILEGROWTH = 5MB
     ),
     ( NAME = Data_2,
      FILENAME = 'C:\DB\Data_2.ndf',
      SIZE = 5MB,
      MAXSIZE = 100MB,
      FILEGROWTH = 5MB
     )
TO FILEGROUP GroupFile_1;
```

Figura 2 Script para modificar una BD

RESPALDO Y RESTAURACIÓN DE UNA BASE DE DATOS

En SQL Server existen tres tipos de respaldos y restauración:

- **Respaldo Completo:** es una copia de toda la base de datos en un determinado momento, la restauración de este respaldo solo recupera la información del momento en que se realizó el respaldo, generalmente se usan para base de datos pocos críticas y pequeñas.

  El script que se utiliza para realizar un respaldo completo de BD es el que se muestra en la siguiente figura:

```
BACKUP DATABASE TickedDB
   TO DISK = 'C:\RespaldosDB\TickedDB.bak'
   WITH FORMAT;
```

Figura 3 Script para realizar un respaldo completo de BD

También se puede realizar por asistente colocándote en el explorador de objetos en la carpeta de base de datos (Databases) -> colocarte en la base de datos que requieres copiar -> botón derecho-> tareas (tasks) -> copia de seguridad (Back Up) como se muestra en la siguiente figura

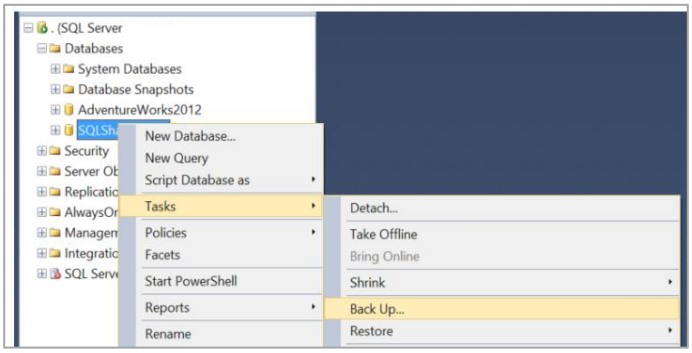


Figura 4 Back Up por asistente

Una vez seleccionando en copia de seguridad (Back Up) aparecerá la siguiente pantalla donde se deberá especificar que es una copia de seguridad completa (full) en backup type, después haciendo clic en el botón “Add”, debajo de “Destination” y especificando el nombre del archivo con la extensión (.bak) y el destino donde se va a guardar el archivo como se muestra en la siguiente figura:

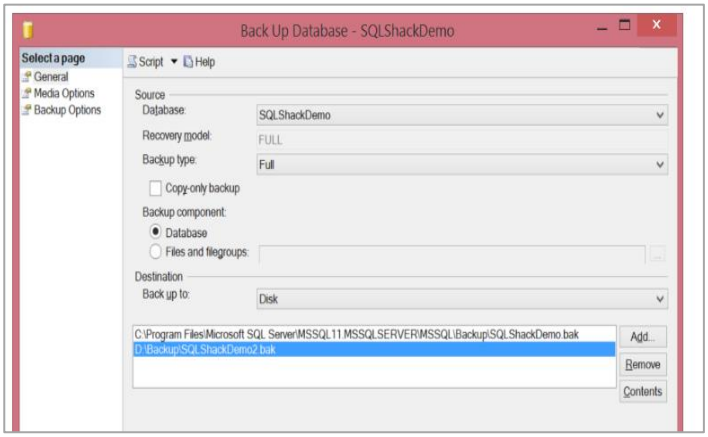


Figura 5 Destino donde se guardara el Back Up

- **Respaldo Diferencial:** solo realizan una copia de los datos nuevos y modificados a partir del último respaldo completo, ideal para grandes bases de datos o con poca densidad de transacciones, requiere menos espacio que un respaldo completo, para restaurar una base de datos con un mecanismo diferencial se debe de contar el respaldo completo. La información restaurada será la que existía en el momento que se realizó el respaldo diferencial.

El script que se utiliza para realizar un respaldo diferencial de BD es como se muestra en la siguiente figura:



Figura 6 Representación de un respaldo diferencial

```
BACKUP DATABASE TickedDB
  TO  DISK = N'C:\RespaldosDB\TickedDB.bak'
            WITH  DIFFERENTIAL,
            NOFORMAT, NOINIT,
            NAME = N'TickedDB Backup',
            SKIP, NOREWIND,
            NOUNLOAD,  STATS = 10
```

Figura 7 Script para realizar un respaldo diferencial

BACKUP DATABASE: nombre de la copia de seguridad de la base de datos.

TO DISK: ubicación donde se guardará el archivo de la copia del BD.

WITH DIFFERENTIAL: agregamos que es de tipo diferencial.

NOFORMAT: el encabezado del medio no debe escribirse para esta operación.

NOINIT: No sobrescribe los archivos de la copia de seguridad.

NAME: nombre del archivo que se va a generar como .bak.

SKIP: verifica la fecha y hora de vencimiento de los archivos de la copia de seguridad antes de sobrescribirlos.

NOREWIND: indica que SQL Server mantendrá la cinta abierta después de la operación de copia de seguridad

NOUNLOAD: son configuraciones de sesión que persisten durante la vida de la sesión o hasta que se reinicia especificando la alternativa.

STATS: informa el porcentaje completado a partir del umbral para informar el siguiente intervalo.

También se puede realizar por asistente al igual que el completo, lo diferente es seleccionar en tipo de copia de seguridad hay que seleccionar tipo diferencial, como se muestra en la figura a continuación.

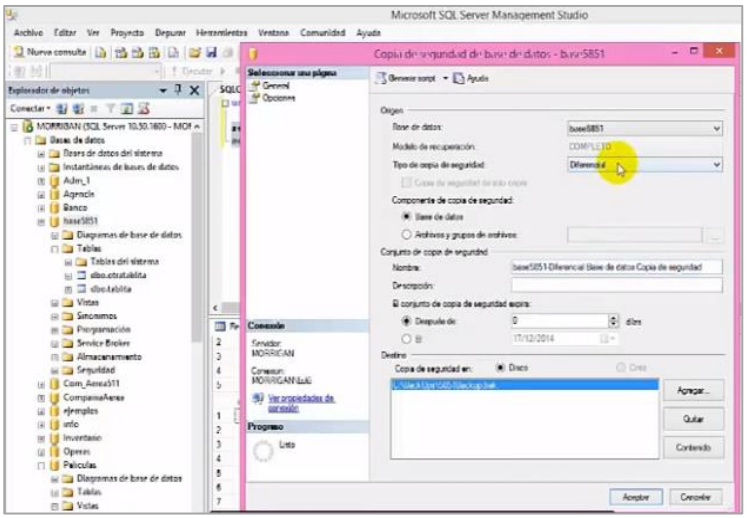


Figura 9 Representación de los tipos de respaldo

- **Respaldo Diferencial + Log de transacciones**
  - Este mecanismo es ideal para base de datos con alta densidad de transacciones y no se permiten perdidas de datos.
  - Si el log de transacciones no está dañado se puede recuperar hasta el último momento de la base de datos.
  - Se requieren de más ficheros y el espacio es un poco mayor con respecto al mecanismo diferencial.

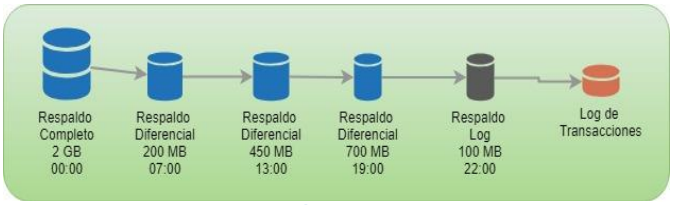


Figura 9 Representación de los tipos de respaldo

Este mecanismo consta de:

1. Un Respaldo completo.
2. Respaldos diferenciales.
3. Respaldos de log de transacciones (opcional).
4. Log de transacciones.

Para restaurar usando este mecanismo:

1. Restaurar el respaldo completo con la opción NORECOVERY
2. Restaurar el último respaldo diferencial con la opción NORECOVERY
3. Restaurar cada uno de los respaldos de log de transacciones en el mismo orden en que fueron realizados utilizando la opción NORECOVERY
4. Restaurar la base de datos usando la opción RECOVERY

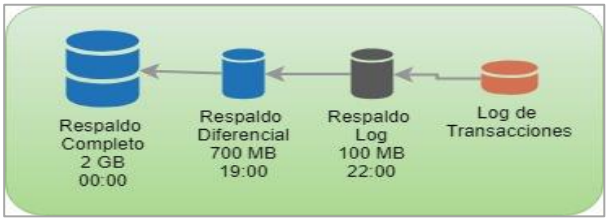


Figura 10 Representación de restauración de back Up

Los scripts que se utilizan para cada respaldo son:

1. Respaldo Completo:

```
RESTORE DATABASE TickedDB
 FROM  DISK = N'C:\DB\TickedDB.bak'
 WITH  FILE = 1,
 NORECOVERY
```

2. Respaldo Diferencial:

```
RESTORE DATABASE TickedDB
 FROM  DISK = N'C:\DB\TickedDB.bak'
 WITH  FILE = 4,
 NORECOVERY
```

3. Respaldo de Log de transacciones:

```
RESTORE LOG TickedDB
 FROM DISK = N'C:\DB\TickedDB_log.bak'
 WITH FILE = 1,
 NORECOVERY
```

4. Log de Transacciones:

```
RESTORE DATABASE TickedDB
   WITH RECOVERY
```

Figura 11 Scripts de los tipos de restauración de back up

EJERCICIO # 1

- Crear una base de datos e insertar datos.
- Realiza un respaldo completo.
- Restaura la base de datos.
- Agrega “DataFile” y “FileGroup” a la base de datos.
- Realiza un respaldo diferencial.
- Crea un objeto en la base de datos.
- Realiza un respaldo de log.
- Crea otro objeto.
- Realiza las restauraciones.

Rúbrica para evaluar:

Actividad	10-8	8-5	5-0
Crear una base de Datos e insertar datos	Hacer código de la base de datos * Tablas * Realizar llave primaria y foránea * Insertar datos mínimo cinco registros	Hacer código de la base de datos * Tablas Que este incompleto alguno de los siguientes puntos: * Realizar llave primaria y foránea * Insertar datos mínimo cinco registros	Hacer código de la base de datos * Tablas No cumple con ninguno de los siguientes puntos: * Realizar llave primaria y foránea * Insertar datos mínimo cinco registros
Realiza un respaldo completo.	Respaldo con código y asistente.	Respaldo con código o asistente.	No realizo la actividad.
Restaura la base de datos	Restauración con código y asistente.	Restauración de datos con código o asistente.	No realizo la actividad.
Agrega DataFile y FileGroup a la base de datos.	Agrega DataFile y FileGroup a la base de datos por asistente y genero código.	Agrego DataFile y FileGroup a la base de datos por asistente .	No realizo la actividad.
Realiza un respaldo diferencial.	Respaldo con código y asistente.	Respaldo con código o asistente.	No realizo la actividad.
Crea un objeto en la base de datos.	Agrego objeto a la base de datos por asistente y genero código.	Agrego objeto a la base de datos por asistente o código.	No realizo la actividad.
Realiza un respaldo de log.	Respaldo código y asistente.	Respaldo con código o asistente.	No realizo la actividad.
Crea otro objeto.	Agrego objeto a la base de datos por asistente y genero código.	Agrego objeto a la base de datos por asistente o código.	No realizo la actividad.
Realiza las restauraciones.	Realizo restauración con código y asistente.	Restauración con código o asistente.	No supo realizar ninguna de las dos maneras.

PLANES DE MANTENIMIENTO

Los planes de mantenimiento crean un flujo de trabajo de las tareas necesarias para asegurarse de que la base de datos está optimizada.

Los planes de mantenimiento crean un paquete de Integration Services, que ejecuta un trabajo del Agente SQL Server. Los planes de mantenimiento se pueden ejecutar manual o automáticamente a intervalos programados.

Los planes de mantenimiento nos permiten:

- Creación de flujos de trabajo con diferentes tareas de mantenimiento típicas. También puede crear sus propios scripts.
- Jerarquías conceptuales. Cada plan le permite crear o editar flujos de trabajo de tareas

Herramientas en el plan de mantenimiento:

- Tarea Copia de seguridad de la base de datos
- Tarea Comprobar la integridad de la base de datos
- Tarea Ejecutar trabajo del Agente SQL Server
- Tarea Ejecutar instrucción T-SQL
- Tarea Limpieza de historial
- Tarea Limpieza de mantenimiento
- Tarea Notificar al operador
- Tarea Volver a generar índice
- Tarea Reorganizar índice
- Tarea Reducir base de datos

- Tarea Actualizar estadísticas

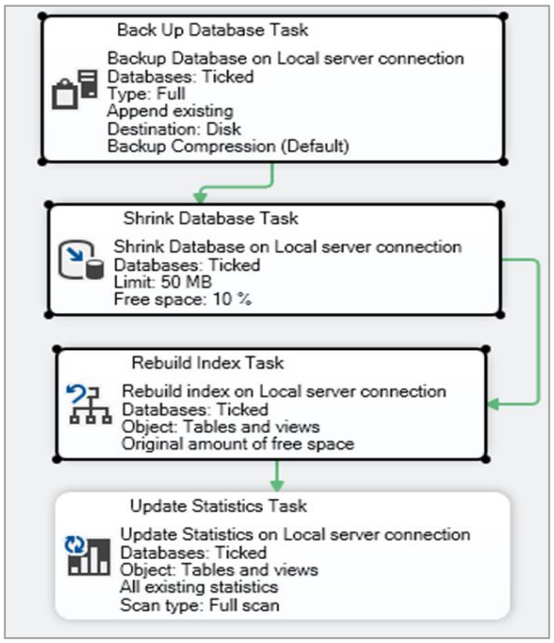


Figura 12 Seguimiento de uso del plan de mantenimiento

El sub-plan permite programar la ejecución del plan de mantenimiento.

El sub-plan contiene:

- Tipo de calendario
- Frecuencia
- Duración

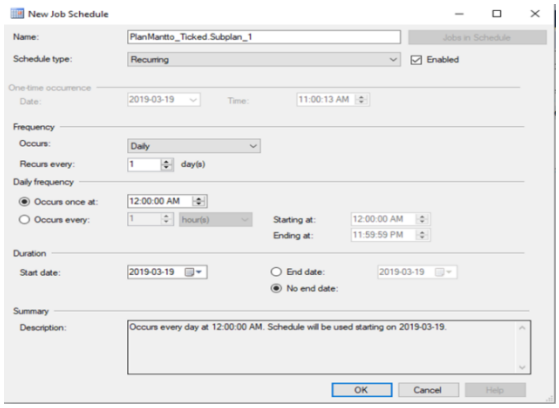


Figura 13 Automatización para la programación de respaldos

LENGUAJE DE MANIPULACION DE DATOS

El lenguaje de manipulación de datos (DML) permite realizar consultas, modificación y eliminación de datos dentro de una base de datos.

Las sentencias son las siguientes:

- SELECT. Permite consultar.
- INSERT. Inserta nuevos registros.
- UPDATE. Permite modificar.
- DELETE. Permite eliminar.
- MERGE. Permite realizar varias acciones dentro de una misma sentencia.

También existen condicionantes que permiten filtrar y manipular datos:

- WHERE
- Operadores lógicos
- JOIN
- UNION
- ORDER BY
- GROUP BY

EJERCICIO # 2

Crea un plan de mantenimiento para una base de datos:

- Respalda la base de datos.
- Libera espacio.
- Regenera índices.
- Actualiza estadísticas.

Rúbrica para evaluar:

Actividad	10	9	8	no acredita (cero)
Respalda la base de datos.	Respaldo utilizando código y asistente.	Respaldo archivos con código o asistente.	Respaldo solo con código o asistente.	No realizo ninguna actividad.
Libera espacio.	Libero espacio siguiendo todos los pasos del archivo .	no aplica.	Libero espacio de manera incompleta.	No realizo ninguna actividad.
Regenera índices.	Regenero índices siguiendo las instrucciones del archivo.	no aplica.	Regenero índices insatisfactorios.	No realizo ninguna actividad.

SENTENCIA SELECT

```
SELECT * | COL1,COL2,COLn…
FROM esquema.tabla
WHERE valor1[Condicion] valor2
```

Figura 14 Sintaxis del comando select

Dónde:

- El “*” devuelve todas las columnas de la “table” que se está consultando, también se puede especificar las columnas que devolverá la consulta.
- Valor1 y valor2 pueden ser columnas de la “table” o algún otro dato.
- Condición puede ser un operador lógico o alguna palabra reservada que indique una condición como BETWEEN.

JOINS

La sentencia JOIN permite combinar dos conjuntos de datos con una determinada condición.

Existen estos tipos de JOIN:

- INNER JOIN
- LEFT OUTER JOIN
- RIGHT OUTER JOIN
- FULL OUTER JOIN
- CROSS JOIN

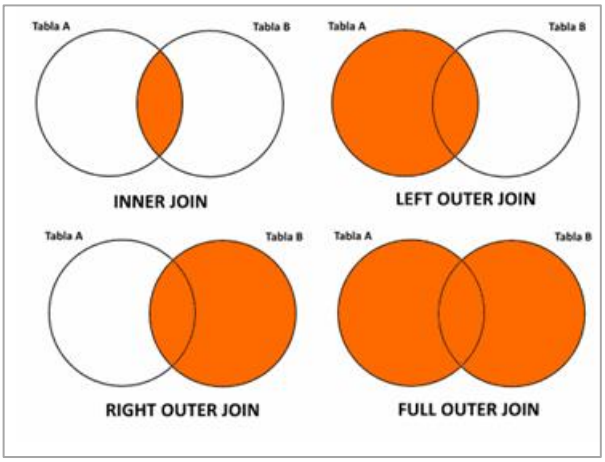


Figura 15 Representación gráfica de los datos que considera el inner

Ejemplo:

```
SELECT *
FROM esquema.tabla_1 alias1
INNER JOIN esquema.tabla_2 alias2 ON alias1.col = alias2.col

SELECT *
FROM esquema.tabla_1 alias1
LEFT OUTER JOIN esquema.tabla_2 alias2 ON alias1.col = alias2.col

SELECT *
FROM esquema.tabla_1 alias1
RIGHT OUTER JOIN esquema.tabla_2 alias2 ON alias1.col = alias2.col

SELECT *
FROM esquema.tabla_1 alias1
FULL OUTER JOIN esquema.tabla_2 alias2 ON alias1.col = alias2.col

SELECT *
FROM esquema.tabla_1 alias1
CROSS JOIN esquema.tabla_2 alias2
```

Figura 16 Sintaxis de aplicabilidad del inner

UNION

La sentencia UNION permite unir dos consultas en una sola.

Restricciones:

- El resultado de las dos consultas debe de contener el mismo número de columnas.
- Las columnas en ambas consultas deben de coincidir en el tipo de dato.
- Los nombres de las columnas del resultado de UNION serán los de la primera consulta individual.
- Puede utilizar un GROUP BY en cada consulta individual pero no al resultado final.
- La cláusula ORDER BY puede ser utilizada para afectar el resultado final, pero no se puede usar en cada consulta individual.

Ejemplo:

```
SELECT Nombre, ApellidoPaterno, ApellinoMaterno
FROM ventas.clientes ct
UNION
SELECT NombreCompleto, Apellido_Pat, Apellido_Mat
FROM seguridad.Usuarios

SELECT Nombre, 'Ventas' AS Transaccion , COUNT(*) Cantidad
  FROM venta.Ventas
 GROUP BY Nombre
UNION
SELECT Nombre, 'Compras' AS Transaccion , COUNT(*) Cantidad
 FROM compra.Compras
 GROUP BY Nombre
ORDER BY Nombre
```

Figura 17 Sintaxis de ejemplos del comando union

GROUP BY Y ORDER BY

La cláusula “GROUP BY” permite agrupar registros iguales bajo ciertos criterios en uno solo.

```
SELECT col1,col,2,coln,<función de agregado>
FROM esquema.tabla
GROUP BY col1,col2,coln
```

Figura 18 Sintaxis de uso de funciones agregadas

La cláusula “ORDER BY” permite ordenar de manera descendente o ascendente los registros bajo ciertos criterios.

```
SELECT col1,col2,coln
FROM esquema.tabla
ORDER BY col1,col2
```

Figura 19 Sintaxis de uso para ordenar los datos

EJERCICIO # 3

Usando una base de datos:

- Realizar un “Query” donde se obtenga el nombre del empleado y dirección, ordenar por apellidos.
- Realizar un “Query” donde se obtenga el número de ventas de cada persona de ventas, después obtener ventas por días.
- Realizar un “Query” para unir a los clientes y proveedores.

Rúbrica para evaluar:

Actividad	10-8	8-5	5-0
Realizar Query donde se obtenga el nombre del empleado y dirección, ordenar por apellidos.	Realizo query incluyo el nombre del empleado y dirección, ordenar por apellidos.	Realizo código donde se incluyó el nombre del empleado o dirección y ordeno por apellidos.	No realizo ninguna actividad.
Realizar Query donde se obtenga el número de ventas de cada persona de ventas, después obtener ventas por días.	Realizo Query se incluyó el número de ventas de cada persona de ventas, después obtener ventas por días.	no aplica	No realizo ninguna actividad.
Realizar Query para unir a los clientes y proveedores	Realizar Query para unir a los clientes y proveedores	no aplica	No realizo ninguna actividad.

INSERT

La sentencia INSERT permite añadir registros a una tabla dentro de una base de datos.

```
INSERT INTO NombreTabla [(Campo1, …, CampoN)] VALUES (Valor1, …, ValorN)
```

Figura 20 Sintaxis para agregar registros en una tabla

Dónde:

- **Nombre tabla**: la tabla en la que se van a insertar las filas.
- **(Campo1, …, CampoN)**: representa el campo o campos en los que vamos a introducir valores.
- **(Valor1, …, ValorN)**: representan los valores que se van a almacenar en cada campo.

Se puede realizar una inserción masiva de registros utilizando la sentencia INSERT de la siguiente forma:

Ejemplo:

```
INSERT INTO NombreTabla [(Campo1, …, CampoN)]
    SELECT ...

Ejemplos:
INSERT INTO Region VALUES(35, 'Madrid')
INSERT INTO Region (RegionID, RegionDescription) VALUES(35, 'Madrid')
INSERT INTO Region (RegionID, RegionDescription) VALUES(35, 'Madrid'),
(36,'Barcelona')

INSERT INTO Customers
   SELECT * FROM NewCustomers WHERE Country = 'Spain'
INSERT INTO Customers (Country, CustomerName)
   SELECT Country, CustomerName FROM NewCustomers WHERE Country = 'Spain'
```

Figura 21 Ejemplo de diferentes usos de agregado de datos

UPDATE

La sentencia UPDATE permite modificar los registros de una tabla en la base de datos.

```
UPDATE esquema.tabla SET col_1 = valor1, col_2 = valor2
```

Figura 22 Sintaxis para modificar datos en un registro

UPDATE se puede combinar con:

- WHERE
- FROM
- JOIN

Ejemplo:

```
UPDATE esquema.tabla SET col1 = valor, col2 = valor
WHERE col_Id = valor

UPDATE alias1
        SET alias1.col1 = alias2.col1,
                alias1.col2 = alias2.col2
FROM esquema.tabla1 alias1
JOIN esquema.tabla2 AS alias2 ON alias2.id = alias1.id
WHERE alias1.col3 <> valor
```

Figura 23 Ejemplo de uso con sintaxis en update

DELETE

La sentencia DELETE permite eliminar registros de una tabla en la base de datos.

Ejemplo:

```
DELETE FROM esquema.Tabla

Ejemplos:
DELETE Employees WHERE EmployeeID = 9

DELETE Customers WHERE LastName LIKE '%Desconocido%'

DELETE Products WHERE CategoryID =
(SELECT CategoryID FROM Categories
     WHERE CategoryName = 'Beverages')

DELETE Products WHERE CategoryID =
(SELECT CategoryID FROM Categories WHERE CategoryName =
'Beverages')
AND UnitPrice > 50
```

Figura 24 Ejemplo de aplicación en comando delete

EJERCICO # 4

- Crea una tabla con el nombre personas con los campos BussinesIdentityID, Nombre, Apellido, Tipo.
- Inserta empleados y clientes.
- Actualiza los empleados con el tipo empleado.
- Actualiza clientes con el tipo cliente.
- Elimina los clientes.

Rubrica para evaluar:

Actividad	10	9	8	no acredita (cero)
Crear una tabla con el nombre personas con los campos BussinesIdentityID, nombre, apellido, tipo.	Creo una tabla con cinco registros del nombre personas con los campos BussinesIdentityID, nombre, apellido, tipo.	Crea una tabla con cinco registros con ejecución incorrecta del nombre personas con los campos BussinesIdentityID, nombre, apellido, tipo.	Crea una tabla sin registros	No realizo ninguna actividad.
Insertar empleados y clientes	Inserta la tabla de empleados y clientes con cinco registros cada una.	Inserta la tabla de empleados o clientes con 5 registros	Inserta la tabla de empleados y clientes sin registros	No realizo ninguna actividad.
Actualizar los empleados con el tipo empleado	Actualiza los empleados de acuerdo a la validación de tipo empleado y genera llaves primarias y foráneas necesarias	Actualiza los empleados de acuerdo a la validación de tipo Empleado sin genera llaves primarias y foráneas	realiza query con ejecución incorrecta al actualiza los empleados de acuerdo a la validación de tipo Empleado	No realizo ninguna actividad.
Eliminar a los clientes.	Elimina los clientes con validación y todos (dos script)	Elimina los clientes con validación o todos	realiza script con ejecución insatisfactoria en Eliminar los clientes con validación y todos	No realizo ninguna actividad.

MERGE

La sentencia MERGE nos permite insertar, actualizar o borrar filas de acuerdo a los resultados que se realizan de la combinación de una tabla, con otro origen de datos, esta última puede ser un CTE, una variable tipo tabla, entre algunas otras:

```
MERGE INTO <tablaobjetivo> AS obj
USING <tablafuente> AS fuente
 ON <clausula coincidencia>
WHEN MATCHED [ AND < clausula >]
 THEN < codigo >
WHEN NOT MATCHED [BY TARGET] [AND < clausula >]
 THEN INSERT...
WHEN NOT MATCHED BY SOURCE [AND < clausula >]
 THEN <codigo>
[OUTPUT ...]
```

Figura 25 Sintaxis de aplicabilidad en el comando marge

Dónde:

- MERGE INTO <tablaObjetivo>: Define la tabla a la cual le realizaremos las operaciones INSERT, UPDATE, o DELETE.
- USING <tablaFuente>: Define la tabla de la cual provienen los datos, aunque también se puede utilizar un CTE o tabla derivada entre algunas otras opciones. Lo más común es utilizar una tabla.
- ON <cláusula de coincidencia>: Define la cláusula utilizada para encontrar las coincidencias entre ambas tablas, fuente y destino, muy parecido al ON de un JOIN.
- WHEN MATCHED [AND <clausula>] THEN <código>: Se utiliza cuando existen coincidencias a través de la cláusula ON, por lo tanto, la acción INSERT no está permitida; es posible utilizar dos cláusulas WHEN MATCHED, una para utilizar la acción UPDATE y

otra para la acción DELETE, la única condicionante es que deben tener filtros si se utilizan ambas.

- WHEN NOT MATCHED [BY TARGET] [AND <clausula>] THEN INSERT... Se utiliza cuando una fila existe en la fuente, pero no en el destino, por lo tanto, la única operación permitida es un INSERT.

- WHEN NOT MATCHED BY SOURCE [AND < clausula >] THEN <código> Es el caso contrario a la cláusula anterior, cuando la fila existe en la tabla destino, pero no en la fuente, no se puede aplicar una operación INSERT, pero si UPDATE y DELETE, también se puede declarar dos cláusulas de este tipo al igual que la cláusula WHEN MATCHED, con la misma condicionante que deben tener filtros.

- OUTPUT Se pueden obtener los datos insertados, eliminados y actualizados por medio de las palabras reservadas insert y delete

Ejemplo:

Tabla origen (@origen)

clave	tamanio	cantidad	nombre
1	CHICO	15	MOUSE
2	CHICO	8	MEMORIA USB
3	GRANDE	5	MONITOR
4	MEDIANO	35	TECLADO

Tabla destino (@destino)

clave	tamanio	cantidad	nombre
2	MEDIANO	16	DISCO DURO
3	MEDIANO	15	MONITOR
4	GRANDE	12	TECLADO Y MOUSE
5	CHICO	50	CARGADOR
6	CHICO	3	MEMORIA RAM

```
MERGE INTO @destino AS obj
USING @origen AS fuente
    ON obj.clave = fuente.clave
WHEN MATCHED AND obj.cantidad > fuente.cantidad
    THEN UPDATE  SET obj.tamanio = fuente.tamanio,
                     obj.cantidad = fuente.cantidad,
                     obj.nombre = fuente.nombre
WHEN MATCHED AND obj.cantidad < fuente.cantidad
    THEN DELETE
WHEN NOT MATCHED BY TARGET
    THEN INSERT VALUES (fuente.clave,fuente.tamanio,fuente.cantidad,fuente.nombre)
WHEN NOT MATCHED BY SOURCE AND obj.cantidad <= 25
    THEN DELETE
WHEN NOT MATCHED BY SOURCE AND obj.cantidad > 25
    THEN UPDATE SET obj.nombre = 'MODIFICADO'
OUTPUT $action AS theAction, inserted.* , deleted.*;
```

Figura 26 Ejemplo de uso del comando merge

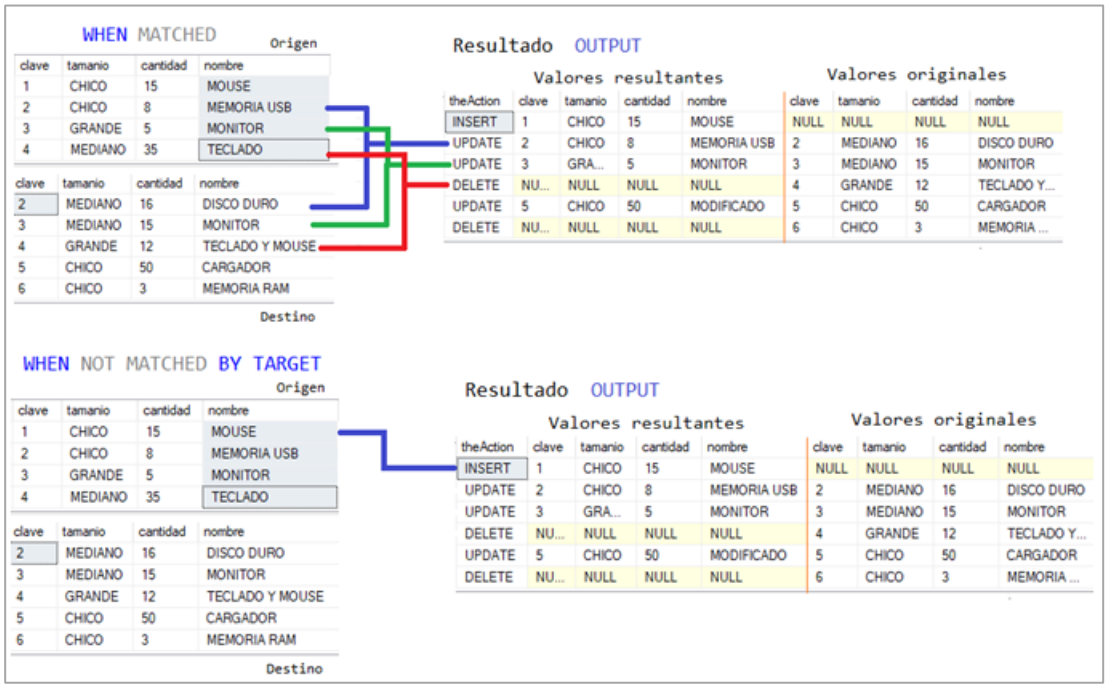


Figura 27 Combinación de tablas con otra para modificar datos

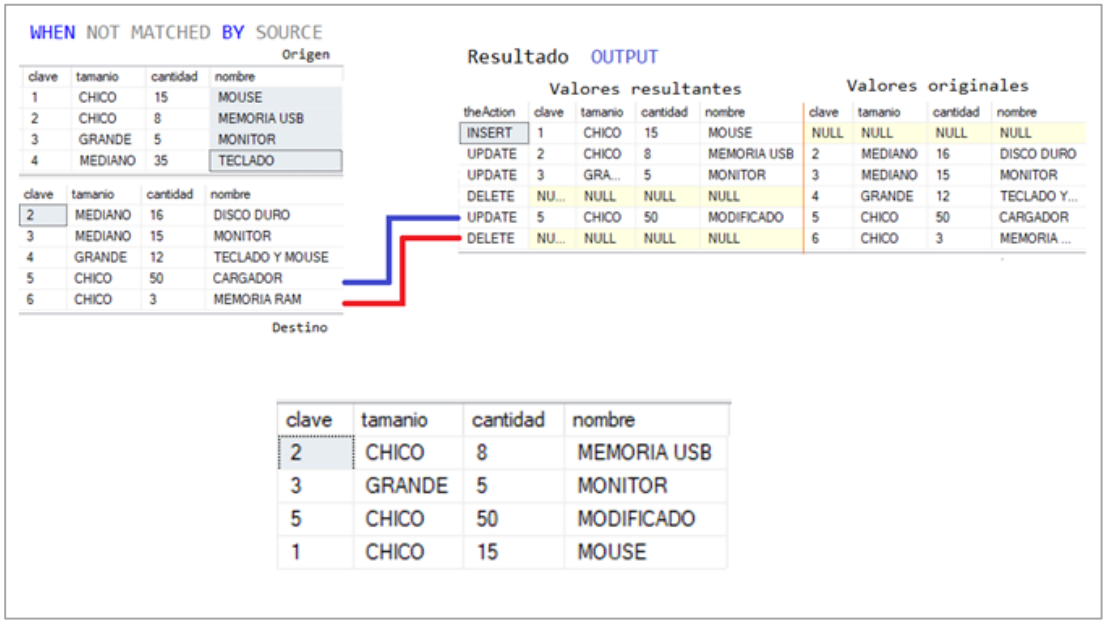


Figura 28 Combinación de tablas para eliminar y modificar tablas

TABLAS TEMPORALES

Las tablas temporales son objetos que se almacenan en la base de datos tempdb, se usan para manipular grandes cantidades de datos.

Tipos de tablas temporales:

- **Tabla Temporal Local:** Se utiliza con un # (hash) para indicar que la tabla temporal será utilizada localmente (Solo será accedida a través de la conexión con cual fue creada). Esta tabla es destruida automáticamente en cuanto se cierre la conexión de SQL Server al menos que no se haya eliminado la tabla manualmente mediante el Query.
- **Tabla Temporal Global:** Se utiliza con doble ## (hash) para indicar que la tabla temporal será utilizada globalmente. Esta tabla puede utilizarla cualquier conexión de SQL Server y estas conexiones pueden modificar, borrar o eliminar la tabla temporal y sus registros. Al igual que las tablas temporales locales, esta se destruye cuando la última sesión utilizada por esta, cierre sesión.

  Ejemplo:

```
CREATE TABLE #TablaTemporal
(
  Id int identity primary key,
  Orden int,
  Campo varchar(50)
)


INSERT INTO #TablaTemporal VALUES (1,'Primer campo')
INSERT INTO #TablaTemporal VALUES (2,'Segundo campo')

SELECT * FROM #TablaTemporal

DROP TABLE #TablaTemporal
```

Figura 29 Sintaxis para crear tablas temporales y agregar datos en tabla

Recomendaciones para usar Tablas Temporales:

- Evitar usar las tablas temporales.
- Si la tabla temporal es muy grande, utilizar índices.
- Eliminar la tabla temporal cuando ya no se necesite.
- Evitar crear tablas temporales con SELECT INTO dentro de procedimientos almacenados y funciones.
- No usar tablas temporales en Trigger ni en Transacciones.

VARIABLES TIPO TABLAS

Las variables tipo tabla son tablas que se crean en memoria y solo existen durante la ejecución del código sql.

Se utiliza con arroba (@) para indicar que se utiliza como variable, al crear la tabla temporal como variable, se debe construir la estructura de la tabla en el Query.

Ejemplo:

```
DECLARE @Usuarios TABLE
(
  Clave INT IDENTITY PRIMARY KEY,
  Nombre VARCHAR(50),
  Apellidos VARCHAR(50),
  INDEX IX_Nombre NONCLUSTERED(Nombre)
 )

INSERT INTO @Usuarios VALUES ('Samanta',
'Rodriguez')

SELECT * FROM @Usuarios
```

Figura 30 Ejemplo de uso para realizar variable tipo tabla y agregar datos

EJERCICIO # 5

- Crear una tabla temporal con los campos BussinesEntityID, nombre, apellido, tipo, ventas_compras, fechaMax.
- Insertar empleados.
- Crear una variable tabla con los campos BussinesEntityID, nombre, Apellido, tipo, ventas_compras, fechaMax.
- Insertar vendedores con el número de venta y la última fecha de ventas.
- Insertar clientes con el número de compras y la última fecha de compras.
- Realizar un MERGE para combinar ambas tablas donde el destino será la tabla temporal.

Rúbrica para evaluar:

Actividad	10-9	8-5	4-0
Crear una tabla temporal con los campos BussinesEntityID, Nombre, Apellido, Tipo, Ventas_Compras, FechaMax	Creo una tabla temporal con los campos BussinesEntityID, nombre, apellido, tipo, ventas_compras, fechaMax	Creo una tabla temporal con los campos BussinesEntityID, nombre, apellido, tipo, Ventas_Compras, FechaMax con ejecución insatisfactoria	No realizo la actividad.
Insertar Empleados	crear tabla empleados e insertar cinco registros.	creo tabla empleados e Inserto registros con ejecución media.	No realizo la actividad.
Crear una Variable tabla con los campos BussinesEntityID, Nombre, Apellido, Tipo, Ventas_Compras, FechaMax	Creo una tabla temporal con los campos BussinesEntityID, nombre, apellido, tipo, ventas_compras, fechaMax	Creo una variable tabla con los campos BussinesEntityID o nombre, apellido, tipo, ventas_compras, fechaMax	No realizo la actividad.
Insertar Vendedores con el número de venta y la última fecha de ventas	Creo tabla vendedores con el número de venta y la última fecha de ventas , insertan cinco registros	Creo tabla vendedores con el número de venta y la última fecha de ventas, inserto menos de cinco registros.	No realizo la actividad.
Insertar Clientes con el número de compras y la última fecha de compras	crear tabla clientes e insertar cinco registros con el número de compras y la última fecha de compras	Creo tabla de clientes e inserto menos de cinco registros con el número de compras y la última fecha de compras con ejecución.	No realizo la actividad.
Realizar un MERGE para combinar ambas tablas donde el destino será la tabla temporal.	Realizar un MERGE para combinar ambas tablas donde el destino será la tabla temporal.	Realizo un MERGE y una tabla donde el destino será la tabla temporal	No realizo la actividad.

TIPO DE DATOS TABLA

Un tipo de datos tabla es la definición de la estructura de una tabla creada por el usuario, lo cual permite crear variable tipo tablas y pasar estas variables como parámetros a procedimientos almacenados y funciones.

- El tipo de datos tabla no se puede asignar a una columna.
- El tipo de datos tabla no se puede modificar.
- Se crea con la instrucción CREATE TYPE.
- Se elimina con la instrucción DROP TYPE.
- Cuando se utiliza como parámetro en un procedimiento o función se declara con la propiedad READONLY.

Ejemplo:

```
CREATE TYPE compra.TLineasPedido AS TABLE
                  (
                   CodArticulo INT NOT NULL
                   ,Cantidad INT NOT NULL
                   ,DescriArticulo VARCHAR(50) NOT NULL
                   ,PrecioArticulo NUMERIC(9,2) NOT NULL
                   PRIMARY KEY (CodArticulo)
                  )

DECLARE @lineasPedido compra.TLineasPedido

CREATE PROCEDURE compra.agregarLineasPedido
        @CodPedido INT, @LineasPedido compra.TLineasPedido READONLY
AS
BEGIN
    INSERT INTO compra.LineasPedido
    SELECT @CodPedido, CodArticulo, Cantidad, DescriArticulo, PrecioArticulo
    FROM @LineasPedido
END
```

Figura 31 Sintaxis aplicada para realizar un tipo de dato tabla y un procedimiento almacenado

CONSULTAS CTES

Common Table Expressions o (CTEs). Se trata de una manera de definir un conjunto de datos temporal (como una tabla temporal) que sólo pervive mientras se ejecute nuestra consulta y no se almacena en ningún sitio. Sin embargo, nos permite consultarla y trabajar con ella como si fuese una tabla real de la base de datos. Estas CTE pueden hacer referencia a sí mismas y se puede usar varias veces en la misma consulta.

```
WITH
NombreCTE [(campos devueltos)]
AS
(Subconsulta)
SELECT * FROM NombreCTE...
```

Ejemplo:

```
WITH EmpTitle (JobTitle,numtibles)
AS
   (
    SELECT a.JobTitle,
           COUNT(*) numtibles
    FROM HumanResources.Employee a
    GROUP BY a.JobTitle
   )

SELECT B.BusinessEntityID,
       B.JobTitle, C.numtibles
FROM EmpTitle as C
INNER JOIN HumanResources.Employee B
           ON C.JobTitle=B.JobTitle;
```

```
SELECT B.BusinessEntityID,
     B.JobTitle, EmpTitle.numtibles
FROM (
   SELECT a.JobTitle,
        COUNT(*) numtibles
   FROM HumanResources.Employee as a
   GROUP BY a.JobTitle
      ) as EmpTitle
INNER JOIN HumanResources.Employee B
    ON EmpTitle.JobTitle=B.JobTitle;
```

Figura 32 Ejemplo de consulta de CTE´s

Otra posibilidad que nos da las consultas CTEs es anidar dos o más consultas CTEs y permite que la siguiente consulta haga referencia a las anteriores.

```
WITH NombreCTE1 [(campos devueltos)]
    AS
      (Subconsulta),
  NombreCTE2 [(campos devueltos)]
    AS
      (Subconsulta),
  NombreCTE3 [(campos devueltos)]
    AS
      (Subconsulta)
```

Figura 33 Sintaxis para anidar dos o más consultas CTE´s

CTES RECURSIVAS

Una expresión de tabla común (CTE) ofrece la gran ventaja de poder hacer referencia a sí misma, creando así una CTE recursiva. Una CTE recursiva es aquella en la que una CTE inicial se ejecuta varias veces para devolver subconjuntos de datos hasta que se obtenga el conjunto de resultados completo.

Todas las definiciones de consulta de miembro no recursivo deben colocarse antes de la primera definición de miembro recursivo y debe utilizarse un operador UNION ALL para combinar el último miembro no recursivo con el primer miembro recursivo.

```
WITH cte_name ( column_name [,...n] )
  AS
   (
    CTE_query_definition -- Query no recursivo.
    UNION ALL
    CTE_query_definition -- Query recursivo
   )
SELECT * FROM cte_name
```

Figura 34 Sintaxis de CTE´s recursivas

EJERCICIO # 6

- Crear un tipo de dato con los datos BussinesEntityID, agente, no transacciones, tipo transacción.
- Crea un CTE para agrupar el número de ventas y el nombre y apellido del agente y tipo de transacción “Ventas”.
- En el CTE agrupa a los empleados con nombre y apellido por el número de compras realizadas y tipo de transacción “compras”
- Inserta los datos en una variable del tipo de datos creado anteriormente.

Rúbrica para evaluar:

Actividad	10	9	8	no acredita (cero)
Crear un tipo de dato con los datos BussinesEntityID, Agente, NoTransacciones, TipoTransaccion	Creo un tipo de dato con los datos BussinesEntityID, agente, no transacciones, tipo transacción.	no aplica	Creo un tipo de dato con los datos BussinesEntityID, agente, NoTransacciones, TipoTransaccion no incluyo una de las especificaciones.	No realizo la actividad.
Crea un CTE para agrupar el número de ventas y el nombre y apellido del agente y tipo de transacción “Ventas”	Creo un CTE para agrupar el número de ventas y el nombre y apellido del agente y tipo de transacción “ventas”.	no aplica	Creo un CTE para agrupar el número de ventas y el nombre y apellido del agente y tipo de transacción “Ventas”. No incluyo alguna de las especificaciones.	No realizo la actividad.
En el CTE Agrupa a los empleados con nombre y apellido por el número de compras realizadas y tipo de transacción “Compras”	En el CTE agrupo a los empleados con nombre y apellido por el número de compras realizadas y tipo de transacción “compras”.	no aplica	En el CTE Agrupa a los empleados con nombre y apellido por el número de compras realizadas y tipo de transacción “Compras”. No incluyo un aspecto.	No realizo la actividad.
Inserta los datos en una variable del tipo de datos creado anteriormente	Inserto los datos en una variable del tipo de datos creado anteriormente.	no aplica	Inserta los datos en una variable del tipo de datos creado anteriormente la ejecución no se dio por falta de datos.	No realizo la actividad.

FUNCIONES DEL SQL SERVER

SQL Server proporciona numerosas funciones integradas que ayudan a la manipulación y transformación de los datos.

- Funciones de agregado.
- Funciones de configuración.
- Funciones de conversión.
- Funciones del cursor.
- Tipos de datos y funciones de fecha y hora.
- Funciones JSON.
- Funciones lógicas.
- Funciones matemáticas.
- Funciones de metadatos.
- Funciones de seguridad.
- Funciones de cadena.
- Funciones del sistema.
- Funciones estadísticas.
- Funciones de texto e imagen.

Función	Descripción
AVG	Devuelve el promedio de los valores de un grupo
COUNT	Devuelve el numero de elementos de un grupo en INT
COUNT_BIG	Devuelve el numero de elementos de un grupo en BIGINT
MAX	Devuelve el valor máximo
MIN	Devuelve el valor mínimo
SUM	Devuelve la suma de todos los valores o solo de los valores DISTINCT de la expresión

Figura 35 Uso de las funciones agregadas

Estas funciones se pueden combinar con la cláusula OVER, la cláusula OVER permite dividir el resultado de la consulta en grupos con la palabra PARTITION BY y también permite ordenar el resultado con ORDER BY.

```
AVG ( [ ALL | DISTINCT ] expression ) [ OVER ( [ partition_by_clause ]
order_by_clause ) ]

COUNT ( { [ [ ALL | DISTINCT ] expression ] | * } )
COUNT ( [ ALL ] { expression | * } ) OVER ( [ <partition_by_clause> ] )

COUNT_BIG ( { [ [ ALL | DISTINCT ] expression ] | * } )
COUNT_BIG ( [ ALL ] { expression | * } ) OVER ( [ <partition_by_clause> ] )

MAX( [ ALL | DISTINCT ] expression )
MAX ([ ALL ] expression) OVER ( [ <partition_by_clause> ] [ <order_by_clause> ] )

MIN ( [ ALL | DISTINCT ] expression )
MIN ( [ ALL ] expression ) OVER ( [ <partition_by_clause> ] [ <order_by_clause> ] )

SUM ( [ ALL | DISTINCT ] expression )
SUM ([ ALL ] expression) OVER ( [ partition_by_clause ] order_by_clause)
```

Figura 36 Sintaxis de funciones agregadas

Función	Descripción
CAST	Convierte un tipo de dato a otro
CONVERT	Convierte un tipo de dato a otro y permite darle un estilo
TRY_CAST	Convierte un tipo de dato a otro, si esto falla devuelve NULL
TRY_CONVERT	Convierte un tipo de dato a otro y permite darle un estilo, si esto falla devuelve NULL

Figura 37 Descripción de Funciones de validación de errores

CAST (expression AS data_type [(length)])

CONVERT (data_type [(length)], expression [, style])

Función	Descripción
DATEADD	Agrega un valor a una fecha, ya sea día, mes o año
DATEDIFF	Devuelve el recuento (como un valor entero con firma) de los límites datepart que se han cruzado entre los valores *startdate* y *enddate* especificados
DATENAME	Esta función devuelve una cadena de caracteres que representa el parámetro *datepart* especificado del argumento *date* especificado.
DATEPART	Esta función devuelve un entero que representa el parámetro *datepart* especificado del parámetro *date* especificado
GETDATE	Devuelve la fecha del sistema
DAY	Devuelve un entero que representa el día del mes
MONTH	Devuelve un entero que representa el mes
YEAR	Devuelve un entero que representa el año

Figura 38 Descripción de Funciones tipo fecha

DATEADD (datepart, number, date)

DATEDIFF (datepart, startdate, enddate)

DATENAME (datepart, date)

DATEPART (datepart, date)

GETDATE ( )

DAY (date)

MONTH (date)

YEAR (date)

Función	Descripción
CHARINDEX	Busca una expresión de caracteres dentro de una segunda expresión de caracteres, y devuelve la posición inicial de la primera expresión si se encuentra.
CONCAT	Devuelve una cadena resultante de la concatenación, o la combinación, de dos o más valores de cadena de una manera integral
LEFT	Devuelve la parte izquierda de una cadena de caracteres con el número de caracteres especificado
LEN	Devuelve el número de caracteres de la expresión de cadena especificada, excluidos los espacios en blanco finales
LOWER	Devuelve una expresión de caracteres después de convertir en minúsculas los datos de caracteres en mayúsculas
LTRIM	Devuelve una expresión de caracteres tras quitar todos los espacios iniciales en blanco
REPLACE	Reemplaza todas las instancias de un valor de cadena especificado por otro valor de cadena

Figura 39 Descripción de uso de las funciones de cadena parte 1

CHARINDEX (expressionToFind , expressionToSearch [, start_location])

CONCAT (string_value1, string_value2 [, string_valueN])

LEFT (character_expression , integer_expression)

LEN (string_expression)

LOWER (character_expression)

LTRIM (character_expression)

REPLACE (string_expression , string_pattern , string_replacement)

REPLICATE	Repite un valor de cadena un número especificado de veces
REVERSE	Devuelve el orden inverso de un valor de cadena
RIGHT	Devuelve la parte derecha de una cadena de caracteres con el número de caracteres especificado
RTRIM	Devuelve una cadena de caracteres después de truncar todos los espacios finales
STRING_SPLIT	Divide una cadena en filas de subcadenas, según un carácter separador especificado
SUBSTRING	Devuelve parte de una expresión de caracteres
UPPER	Devuelve una expresión de caracteres con datos de caracteres en minúsculas convertidos a mayúsculas
ROWNUMBER	

Figura 40 Descripción de uso de las funciones de cadena parte 2

REPLICATE (string_expression ,integer_expression)

REVERSE (string_expression)

RIGHT (character_expression , integer_expression)

RTRIM (character_expression)

STRING_SPLIT (string , separator)

SUBSTRING (expression ,start , length)

UPPER (character_expression)

Función	Descripción
ROWNUMBER	Enumera los resultados de un conjunto de resultados. Concretamente, devuelve el número secuencial de una fila dentro de una partición de un conjunto de resultados, empezando por 1 para la primera fila de cada partición
ISNULL	Sustituye el valor NULL por el valor especificado
ISNUMERIC	Determina si una expresión es un tipo numérico válido
NEWID	Crea un valor único del tipo **uniqueidentifier**
@@IDENTITY	Devuelve el último valor de identidad insertado
@@ROWCOUNT	Devuelve el número de filas afectadas por la última instrucción

Figura 41 Descripción de uso de validación de errores de cadena

ROW_NUMBER ( ) OVER ([PARTITION BY value_expression , ... [n]] order_by_clause)

ISNULL (check_expression , replacement_value)

ISNUMERIC (expression)

NEWID ( )

@@IDENTITY

@@ROWCOUNT

EJERCICIO # 7

- Crea una variable tipo tabla o tabla temporal con los campos nombre, apellido, fecha, monto, tipotransaccion, día.
- Inserta todos los Clientes con el nombre y apellido en mayúsculas, la fecha máxima de compra, y el promedio del monto, tipo transacción “ventas + la orden de venta colocando un prefijo OV-”.
- Inserta todos los empleados con el nombre y apellido en minúsculas, fecha mínima de compra y la suma de todos los montos, tipo transacción “compras + orden de compra colocando un prefijo OC-”.
- Actualiza el campo día con el nombre del día de la fecha.

Rúbrica para evaluar:

Actividad	10-8	8-5	5-0
Crear una variable tipo tabla o tabla temporal con los campos nombre, apellido, fecha, monto, tipo transacción, día.	Creo una variable tipo tabla o tabla temporal con los campos nombre, apellido, fecha, monto, tipo transacción, día.	Creo una variable tipo tabla o tabla temporal omitió algunos puntos.	No realizo la actividad.
Insertar todos los clientes con el nombre y apellido en mayúsculas, la fecha máxima de compra, y el promedio del monto, tipo transacción “ventas + la orden de venta colocando un prefijo OV-”.	Inserto todos los clientes con el nombre y apellido en mayúsculas, la fecha máxima de compra, y el promedio del monto, tipo transacción “ventas + la orden de venta colocando un prefijo OV-”.	Inserto todos los Clientes con el nombre y apellido en mayúsculas, la fecha máxima de compra, omitió puntos.	no realizo la actividad.
Insertar todos los empleados con el nombre y apellido en minúsculas, fecha mínima de compra y la suma de todos los montos, tipo transacción “compras + orden de compra colocando un prefijo OC-”.	Inserto todos los empleados con el nombre y apellido en minúsculas, fecha mínima de compra y la suma de todos los montos, tipo transacción “Compras + orden de compra colocando un prefijo OC-”	Inserto todos los empleados con el nombre y apellido en minúsculas, fecha mínima de compra omitió puntos de la actividad.	no realizo la actividad.
Actualizar el campo día con el nombre del día de la fecha.	Actualizo el campo día con el nombre del día de la fecha	Actualizo el campo día con información no correspondiente.	No realizo la actividad.

LENGUAJE DE CONTROL DE FLUJO

El lenguaje de control de flujo permite controlar bloques de sentencias dándole un sentido programado.

Elementos del Control de flujo:

- IF...ELSE.
- BEGIN...END.
- WHILE.
- BREAK.
- CONTINUE.
- RETURN.
- TRY...CATCH.
- CASE.
- RAISERROR.

BEGIN… END

Begin- End: agrupa un conjunto de instrucciones que forman parte de una instrucción de control de flujo.

```
BEGIN
    { sql_statement | statement_block }
END
```

IF-ELSE

La sentencia IF permite evaluar una o más condiciones, si esta condición resulta verdadera se ejecutara un bloque de sentencias en un BEGIN… END, si resulta falso, opcionalmente con la sentencia ELSE puede ejecutar otro bloque de sentencias contenidas en un BEGIN… END.

IF Boolean_expression

{ sql_statement | statement_block }

[ELSE

{ sql_statement | statement_block }]

WHILE

Establece una condición para la ejecución repetida de una instrucción o bloque de instrucciones SQL, las instrucciones se ejecutan repetidamente siempre que la condición especificada sea verdadera. Se puede controlar la ejecución de instrucciones en el bucle WHILE con las palabras clave BREAK y CONTINUE

WHILE Boolean_expression { sql_statement | statement_block | BREAK | CONTINUE }

- **BREAK**: Produce la salida del bucle WHILE más interno, se ejecutan las instrucciones que aparecen después de la palabra clave END, que marca el final del bucle.

- **CONTINUE**: Hace que se reinicie el bucle WHILE y omite las instrucciones que haya después de la palabra clave CONTINUE.
- **EXISTS**: Se usa como condición, el WHILE será verdadero mientras exista valor en la condición.

Ejemplo:

```
DECLARE @contador INT = 0

WHILE (@contador <5)
BEGIN
PRINT 'Iniciando vuelta ‘
      + convert(varchar,@contador);
SET @contador = @contador + 1
IF @contador = 2
CONTINUE
ELSE
 BEGIN
   PRINT 'Sin saltar'
   IF @contador = 4
     BREAK
     END

END
```

```
DECLARE @persona TABLE
        ( Id int,
          Nombre Varchar(20),
          Apellido varchar(20))

DECLARE @id int = 0,
        @cadena varchar (100)

INSERT INTO @persona
SELECT TOP 10 BusinessEntityID,
        FirstName, LastName
FROM Person.Person

WHILE EXISTS (SELECT * FROM @persona)
BEGIN
   SELECT TOP 1 @id = Id,
        @cadena = Nombre + ' ' + Apellido
   FROM @persona
   PRINT @cadena
   DELETE FROM @persona WHERE Id = @id
END
```

Figura 42 Sintaxis de control de flujo por sentencias

TRY… CATCH

Es un mecanismo de control de errores, se puede incluir un grupo de instrucciones Transact-SQL en un bloque TRY, si se produce un error en el bloque TRY, el control se transfiere a otro grupo de instrucciones que está incluido en un bloque CATCH.

BEGIN TRY

{ sql_statement | statement_block }

END TRY

BEGIN CATCH

[{ sql_statement | statement_block }]

END CATCH

Ejemplo:

```
BEGIN TRY
    -- División entre cero.
    SELECT 1/0;
END TRY
BEGIN CATCH
    SELECT
        ERROR_NUMBER() AS ErrorNumber
        ,ERROR_SEVERITY() AS ErrorSeverity
        ,ERROR_STATE() AS ErrorState
        ,ERROR_PROCEDURE() AS ErrorProcedure
        ,ERROR_LINE() AS ErrorLine
        ,ERROR_MESSAGE() AS ErrorMessage;
END CATCH;
GO
```

Figura 43 Sintaxis de validación de errores por try-catch

RAISERROR

Genera un mensaje de error e inicia el procesamiento de errores de la sesión. RAISERROR puede hacer referencia a un mensaje definido por el usuario almacenado en la vista de catálogo sys.messages o puede generar un mensaje dinámicamente, el mensaje se devuelve como un mensaje de error de servidor a la aplicación que realiza la llamada o a un bloque CATCH asociado de una construcción TRY…CATCH.

RAISERROR ({ msg_id | msg_str | @local_variable } { ,severity ,state }

THROW

Genera una excepción y transfiere la ejecución a un bloque CATCH de una construcción TRY…CATCH en SQL Server 2017.

THROW [{ error_number | @local_variable }, { message | @local_variable }, { state | @local_variable }]

Ejemplo:

```
THROW 51000, 'The record does not exist.', 1;

---------------------------------------------------------

USE tempdb;
GO
CREATE TABLE dbo.TestRethrow
(     ID INT PRIMARY KEY
);
BEGIN TRY
    INSERT dbo.TestRethrow(ID) VALUES(1);
--  Force error 2627, Violation of PRIMARY KEY constraint to be
raised.
    INSERT dbo.TestRethrow(ID) VALUES(1);
END TRY
BEGIN CATCH

    PRINT 'In catch block.';
    THROW;
END CATCH;
```

Figura 44 Sintaxis validación por throw

CASE

Evalúa una lista de condiciones y devuelve una de las varias expresiones de resultado posibles.

La expresión CASE tiene dos formatos:

- La expresión CASE sencilla compara una expresión con un conjunto de expresiones sencillas para determinar el resultado.
- La expresión CASE buscada evalúa un conjunto de expresiones booleanas para determinar el resultado.

Ambos formatos admiten un argumento ELSE opcional.

Ejemplo:

```
--- CASE Simple:
CASE input_expression
   WHEN when_expression THEN result_expression [...n]
     [ ELSE else_result_expression ]
END

--- CASE Buscada:
CASE
   WHEN Boolean expression THEN result_expression
      [...n ]
     [ ELSE else_result_expression ]
END
```

Figura 45 Sintaxis de la sentencia case-when

EJERCICIO # 8

- Crea una variable tipo tabla con las columnas nombre, apellido, dirección, rol donde rol tendrá los valores [Cliente,AgenteVentas,AgenteCompras]
- Crea una consulta para obtener los registros para la variable tipo tabla usando solo JOIN y usa CASE para darle valor a la columna de Rol.
- Crea otra variable tipo tabla con las mismas columnas que la anterior e inserta 5 veces los registros de la primera variable a la nueva usando WHILE pero en la tercera vuelta no insertes el tipo

Rúbrica para evaluar:

Actividad	10-8	8-5	5-0
Crear una variable tipo tabla con las columnas nombre, apellido, dirección, Rol donde Rol tendrá los valores [Cliente,AgenteVentas,AgenteCompras]	Creo una variable tipo tabla con las columnas nombre, apellido,-dirección, Rol donde Rol tendrá los valores [Cliente,AgenteVentas,AgenteCompras]	Creo una variable tipo tabla con las columnas nombre, apellido, dirección, Rol donde Rol, sin embargo no realizo todos los procedimientos.	No realizo la actividad.
Crear una consulta para obtener los registros para la variable tipo tabla usando solo JOIN y usa CASE para darle valor a la columna de Rol.	Creo una consulta para obtener los registros para la variable tipo tabla usando solo JOIN y usa CASE para darle valor a la columna de Rol.	Creo una consulta para obtener los registros para la variable tipo tabla usando solo JOIN o CASE, sin embargo no realizo todos los procedimientos.	No realizo la actividad.
Crear otra variable tipo tabla con las mismas columnas que la anterior e inserta 5 veces los registros de la primera variable a la nueva usando WHILE pero en la tercera vuelta no insertes el tipo	Creo otra variable tipo tabla con las mismas columnas que la anterior e inserta 5 veces los registros de la primera variable a la nueva usando WHILE pero en la tercera vuelta no insertes el tipo	Creo otra variable tipo tabla con las mismas columnas que la anterior e inserta cinco veces los registros de la primera variable, sin embargo no realizo todos los procedimientos.	No realizo la actividad.

FUNCIONES DEFINIDAS POR EL USUARIO

Las funciones definidas por el usuario de SQL Server son rutinas que aceptan parámetros, realizan una acción, como un cálculo complejo, y devuelven el resultado de esa acción como un valor. El valor devuelto puede ser un valor escalar único o un conjunto de resultados (tabla).

FUNCIÓN ESCALAR

- Las funciones escalares definidas por el usuario devuelven un único valor de datos del tipo definido en la cláusula RETURNS.
- El tipo devuelto puede ser de cualquier tipo de datos excepto text, ntext, image, cursory timestamp.

Ejemplo:

```
CREATE FUNCTION dbo.ufnGetInventoryStock(@ProductID int)
RETURNS int
AS
-- Returns the stock level for the product.
BEGIN
    DECLARE @ret int;
    SELECT @ret = SUM(p.Quantity)
    FROM Production.ProductInventory p
    WHERE p.ProductID = @ProductID
        AND p.LocationID = '6';
     IF (@ret IS NULL)
        SET @ret = 0;
    RETURN @ret;
END;
```

Figura 46 Sintaxis para crear funciones con consultas

FUNCIÓN CON VALOR DE TABLA

- Las funciones con valores de tabla definidas por el usuario devuelven un tipo de datos table.
- La tabla es el conjunto de resultados de una sola instrucción SELECT.

Ejemplo:

```
CREATE FUNCTION Sales.ufn_SalesByStore (@storeid int)
RETURNS TABLE
AS
RETURN
(
    SELECT P.ProductID, P.Name, SUM(SD.LineTotal) AS 'Total'
    FROM Production.Product AS P
    JOIN Sales.SalesOrderDetail AS SD ON SD.ProductID = P.ProductID
    JOIN Sales.SalesOrderHeader AS SH ON SH.SalesOrderID = SD.SalesOrderID
    JOIN Sales.Customer AS C ON SH.CustomerID = C.CustomerID
    WHERE C.StoreID = @storeid
    GROUP BY P.ProductID, P.Name
);
```

Figura 47 Ejemplo para crear funciones con valor de tabla en funciones agregadas

USO DE FUNCIONES

- Las funciones se usan en una sentencia, en el contexto en donde se espera el tipo de resultado.

Ejemplo con el comando EXEC:

- EXEC @RESP = dbo.SUMA 4, 5

MODIFICAR FUNCIONES

Para realizar un cambio en una función, se usa la cláusula ALTER.

Ejemplo:

```
ALTER FUNCTION SUMA(@PAR1 DECIMAL(18,2), @PAR2 DECIMAL(18,2))
    RETURNS DECIMAL(18,2)   AS
    BEGIN
        RETURN @PAR1 + @PAR2 + 1
    END
```

Figura 48 Ejemplo para realizar cambios en funciones

ELIMINAR UNA FUNCIÓN

- Para eliminar una función se usa la cláusula DROP
- DROP FUNCTION SUMA

PROCEDIMIENTOS ALMACENADOS

- Un procedimiento almacenado de SQL Server es un grupo de una o varias instrucciones.
- Los procedimientos se asemejan a las construcciones de otros lenguajes de programación, porque pueden:
  - Aceptar parámetros de entrada
  - Contener llamadas a otros procedimientos.
  - Realizar operaciones en la base de datos.

- Devolver un valor de estado.

Ventajas de SP

- Tráfico de red reducido entre el cliente y el servidor.
- Mayor seguridad.
- Reutilización del código.
- Mantenimiento más sencillo.
- Rendimiento mejorado.

Crear un SP

Se muestra la estructura básica para crear un sp:

```
USE AdventureWorks2012;
GO
CREATE PROCEDURE HumanResources.uspGetEmployeesTest2
    @LastName nvarchar(50),
    @FirstName nvarchar(50)
AS

    SET NOCOUNT ON;
    SELECT FirstName, LastName, Department
    FROM HumanResources.vEmployeeDepartmentHistory
    WHERE FirstName = @FirstName AND LastName = @LastName
    AND EndDate IS NULL;
GO
```

Figura 49 Ejemplo de creación de procedimiento almacenado

Ejecutar un SP

- A continuación, se muestra una sentencia que invoca la ejecución de un sp existente en la base de datos.
- EXECUTE HumanResources.uspGetEmployeesTest2 N'Ackerman', N'Pilar';

Modificar un SP

Para realizar un cambio en un procedimiento almacenado, se utiliza la cláusula ALTER.

```
ALTER PROCEDURE Purchasing.uspVendorAllInfo
    @Product varchar(25)
AS
    SET NOCOUNT ON;
    SELECT LEFT(v.Name, 25) AS Vendor, LEFT(p.Name, 25) AS 'Prc
    'Rating' = CASE v.CreditRating
        WHEN 1 THEN 'Superior'
        WHEN 2 THEN 'Excellent'
        WHEN 3 THEN 'Above average'
        WHEN 4 THEN 'Average'
        WHEN 5 THEN 'Below average'
        ELSE 'No rating'
        END
    , Availability = CASE v.ActiveFlag
        WHEN 1 THEN 'Yes'
        ELSE 'No'
```

Figura 50 Ejemplo para modificar un procedimiento almacenado con función case

Eliminar un SP

- Para eliminar un procedimiento almacenado, existente en la base de datos, se usa la cláusula DROP
- La estructura es:
    - DROP PROCEDURE ESQUEMA.NOMBRE_SP
- Ejemplo:
    - DROP PROCEDURE dbo.uspGetEmployeesTest2

EJERCICIO # 9

- Crear una función que devuelva si la persona es agente de ventas, agente de compras o cliente, si la persona no es ninguno de los tres genera un error.
- Crear un procedimiento almacenado donde realices una división 10 / 10 pero cada vez el numerador aumente uno y el divisor disminuya en 1, usa un tray cath para que cuando llegue al error imprimas error de división.

Rúbrica para evaluar:

Actividad	10-8	8-5	5-0
Crear una función que devuelva si la persona es Agente de ventas, Agente de compras o Cliente, si la persona no es ninguno de los tres genera un error.	Creo una función que devuelva si la persona es agente de ventas, agente de compras o cliente, si la persona no es ninguno de los tres genera un error.	Creo una función que devuelva si la persona es agente de ventas, agente de compras o cliente, no realizo todos los procedimientos.	No realizo la actividad.
Crear un procedimiento almacenado donde realices una división 10 / 10 pero cada vez el numerador aumente uno y el divisor disminuya en 1, usa un tray cath para que cuando llegue al error imprimas Error de división	Creo un procedimiento almacenado donde realices una división 10 / 10, pero cada vez el numerador aumente uno y el divisor disminuya en uno, usa un tray cath para que cuando llegue al error imprimas Error de división	Creo un procedimiento almacenado donde realices una división 10 / 10 pero cada vez el numerador aumente uno y el divisor disminuya en 1, usa un tray cath, no realizo los procedimientos y genero un error.	No realizo la actividad.

SERVICIOS EN LA NUBE

Características de las soluciones de servicios en la Nube:

- Servicio bajo demanda.
- Amplio acceso a la red.
- Puesta en común de recursos.
- Elasticidad rápida.
- Servicio medido.

Ventajas de la nube:

- Acceso a una amplia gama de servicios administrados.
- Gastos de capital minimizados o eliminados.
- Reducción de los gastos operativos.
- Modelos de facturación basado en el uso.
- Mayor agilidad.



Figura 51 Representación gráfica de la nube en plataforma azure

¿QUÉ ES AZURE?

Microsoft Azure es la plataforma que ofrece servicios en la nube con la cual las organizaciones pueden crear, administrar e implementar aplicaciones en una red con sus herramientas y marcos favoritos. Entre los servicios que ofrece Azuse son: Plataforma como Servicio (PaaS) e Infraestructura como Servicio (IaaS).



Figura 52 Servicios de la nube en plataforma azure

WINDOWS AZURE

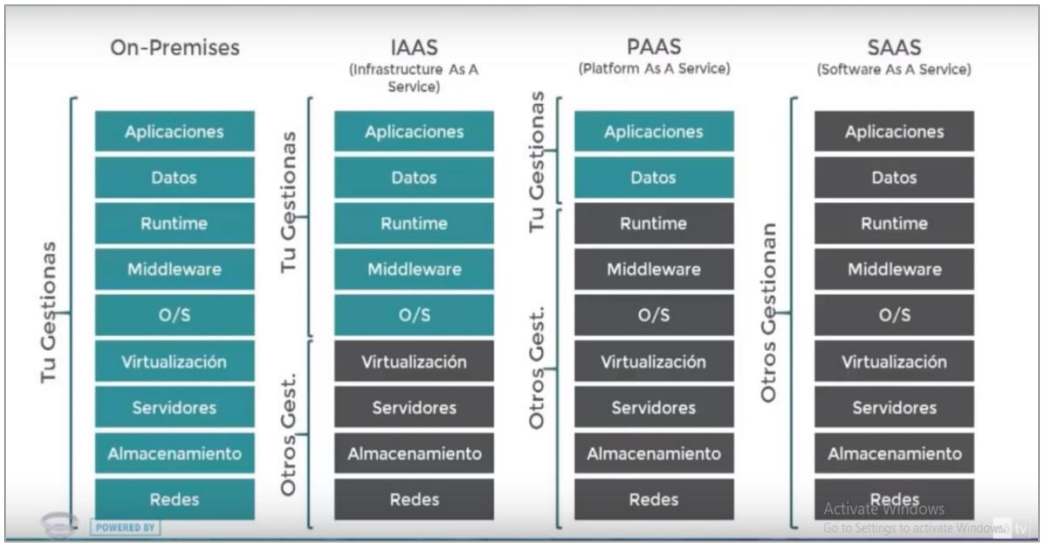


Figura 53 Arquitectura de la nube azure

- Servicios de Azure le permiten:
    - Implementar y operar aplicaciones basadas en la nube.
    - Cargas de trabajo de host en la nube.
    - Integrar los servicios en la nube con una infraestructura local.

La colocación de los centros de datos sigue el principio de redundancia.

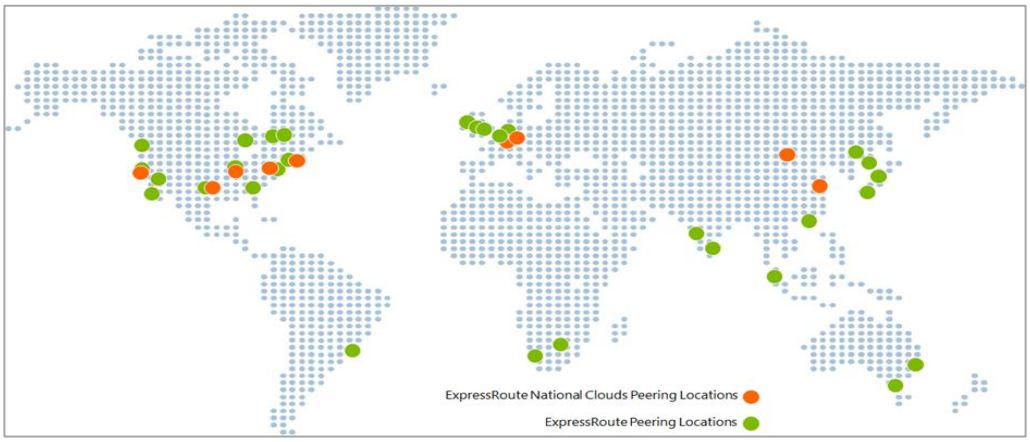


Figura 55 Ejemplificación de servidores en la nube azure

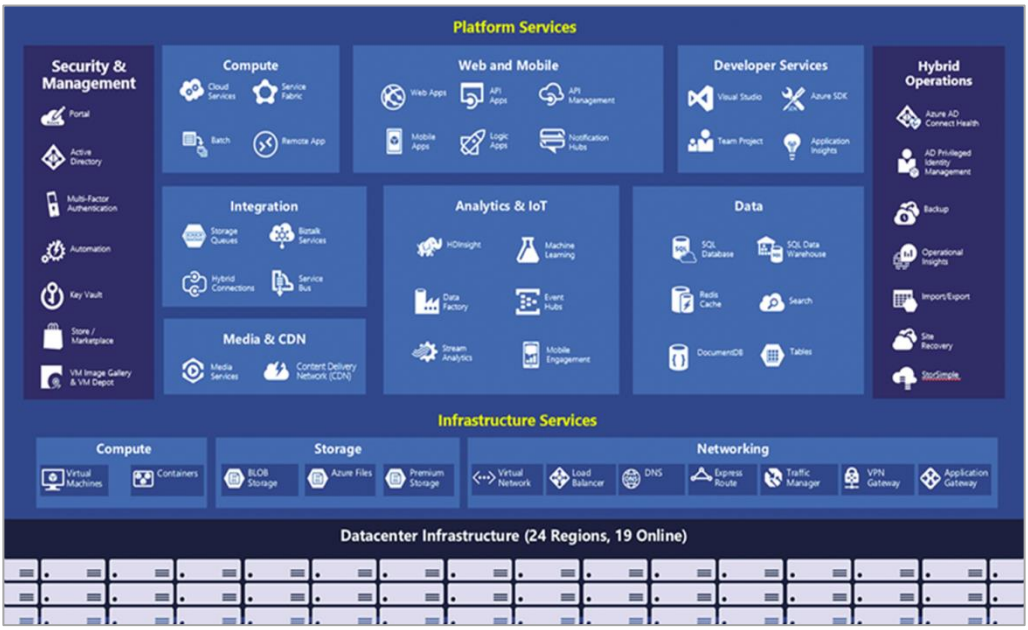


Figura 54 Servicios ofrecidos en las diferentes plataformas

MODELOS DE GESTIÓN DE AZURE

Azure Service Management (Clásico):

- Portal clásico de Azure.
- Proporciona un soporte RBAC limitado.

Azure Resource Manager:

- Se basa en el concepto de grupos de recursos.
- Posibilidad de etiquetar los grupos de recursos.
- Soporta implementaciones basadas en plantillas.
- Proporciona soporte RBAC.
- No está disponible a través del portal clásico de Azure.

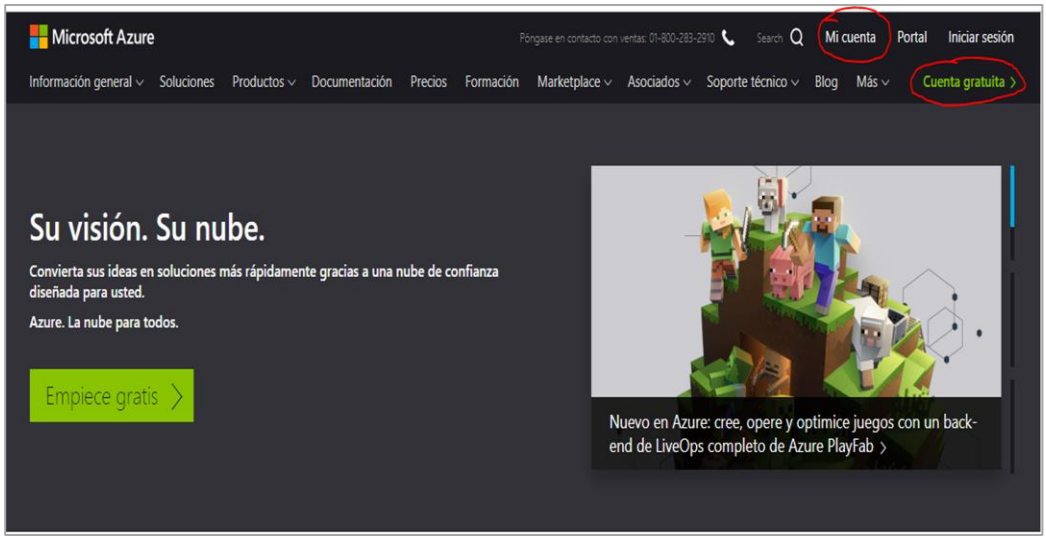


Figura 56 Página de inicio para usuarios de azure

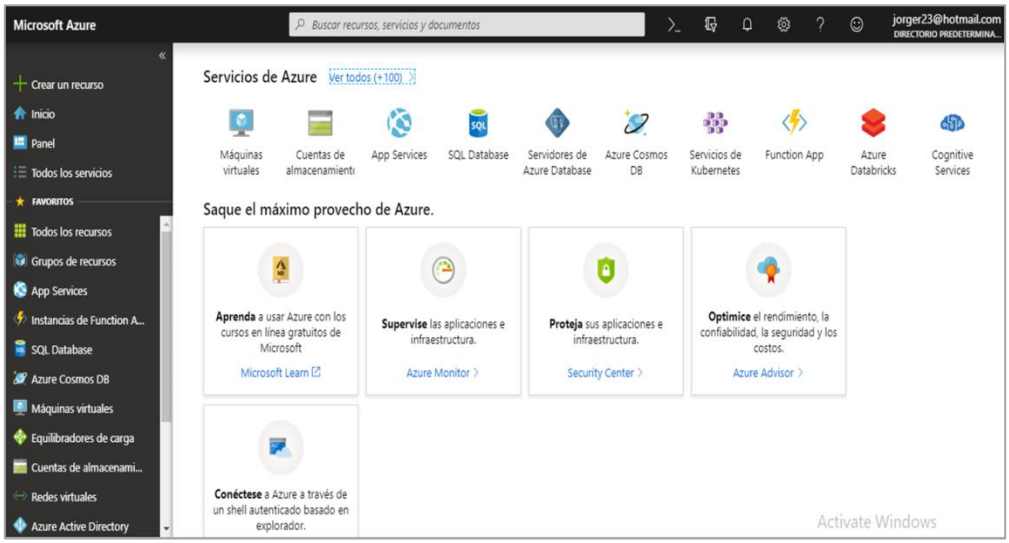


Figura 58 Página principal de usuarios en plataforma azure

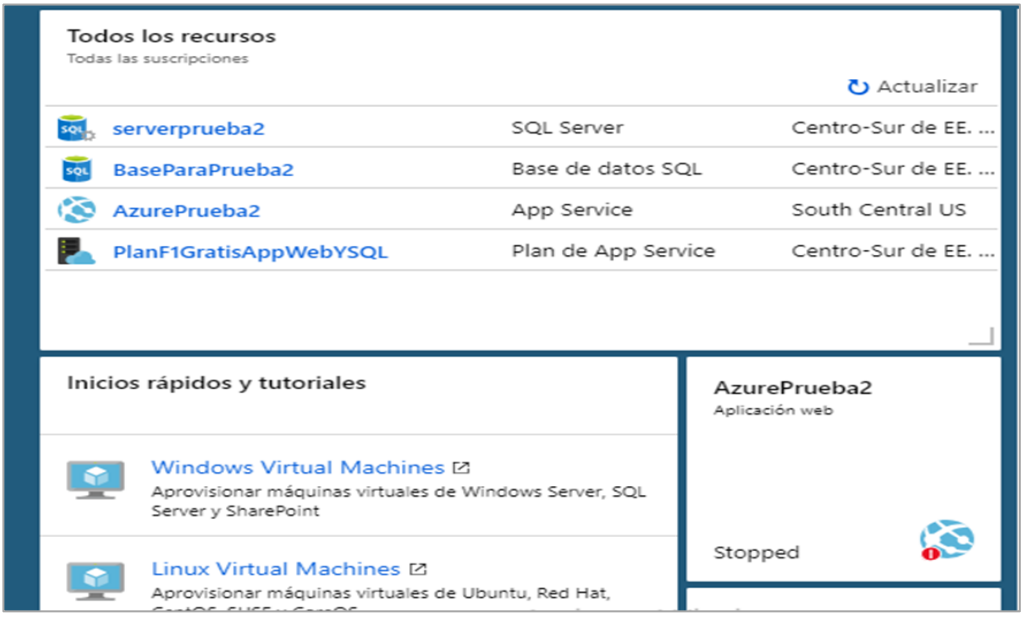


Figura 57 Recursos creados en plataforma azure



Figura 60 Datos de contacto para dudas en la plataforma azure

Paas	Iaas
Reducción de la administración	Mayor capacidad de administración
Costo minimizado	Incluye cargos por VM
Rápida implementación	Requiere configuración de maquina virtual
Características parciales de paridad locales de SQL Server	Características de paridad dentro de SQL Server Local
No hay soporte de red virtual	Soporte de red virtual
Alta disponibilidad y escalabilidad administradas	Soporte para alta disponibilidad y escalabilidad

Figura 59 Tabla comparativa de servicios en azure

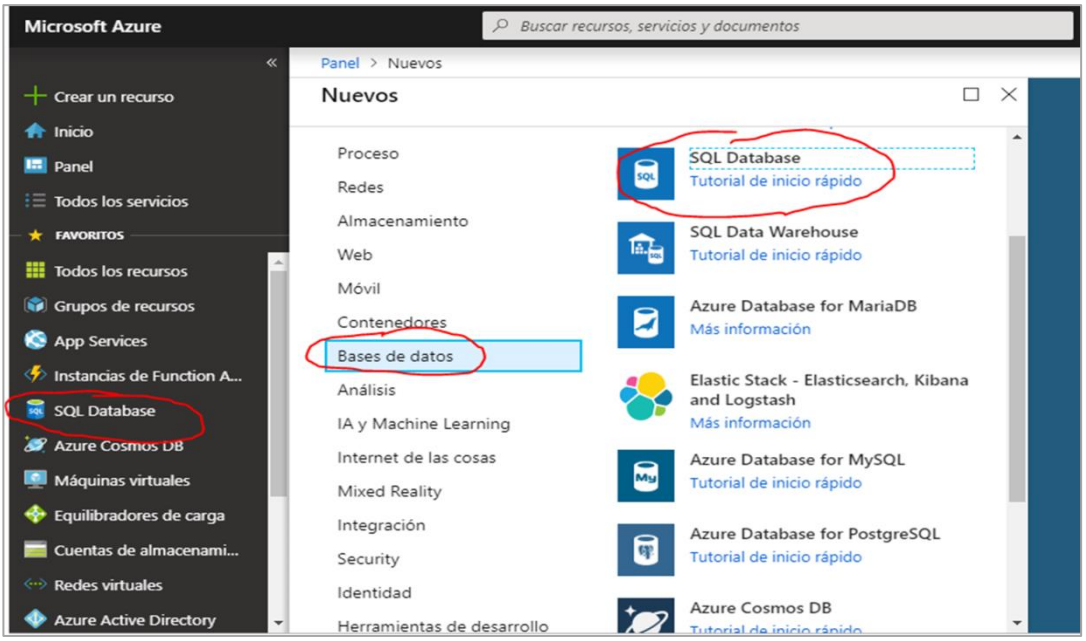


Figura 62 Creación de base de datos en la nube en plataforma azure

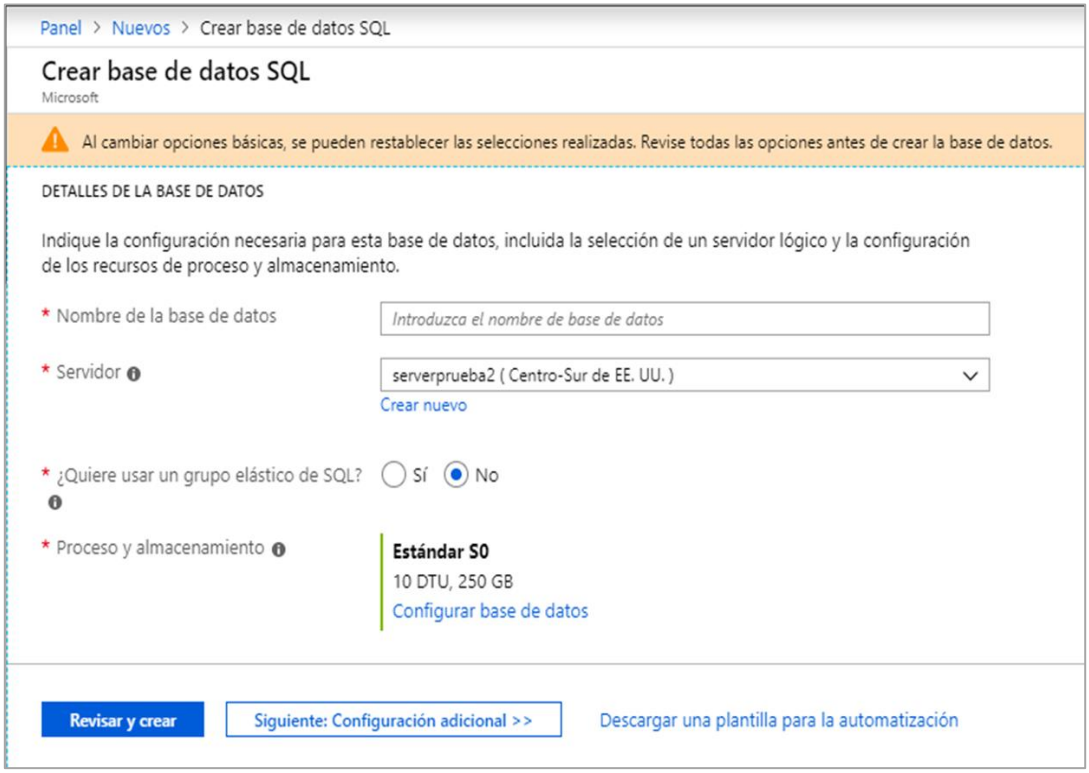


Figura 61 Pantalla para crear base de datos por asistente en plataforma azure

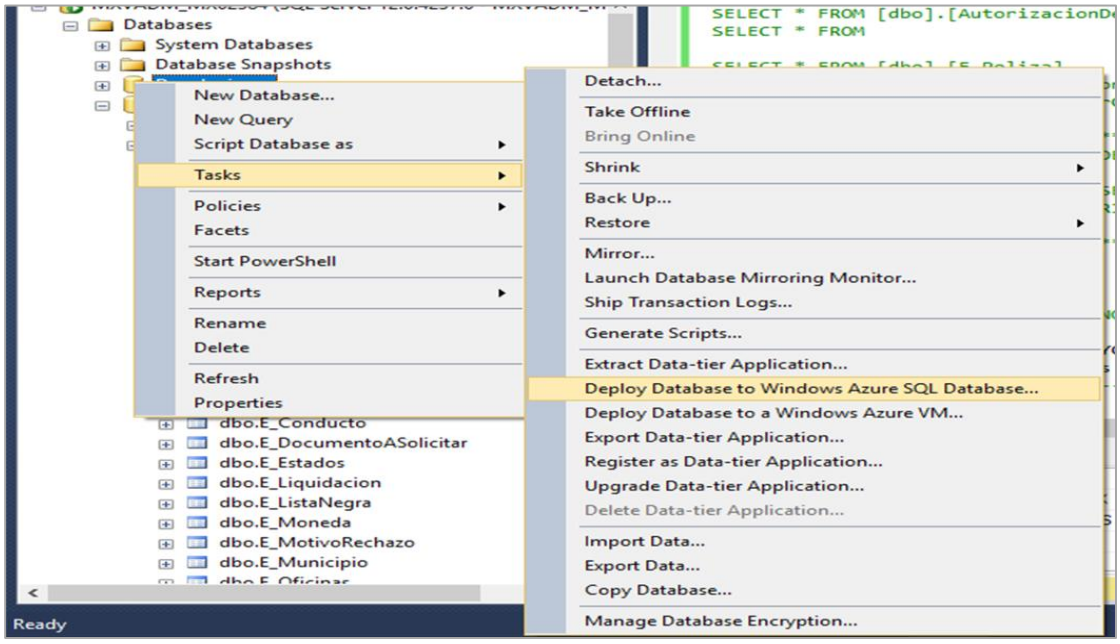


Figura 63 Creación de tablas por asistente en la nube en plataforma azure

EJERCICIO # 10

- Reación de una base de datos única mediante Azure Portal*.
- Configuración de una regla de firewall por IP de nivel de servidor mediante Azure Portal.
- Conéctese a la base de datos con SSMS.
- Crear tablas con SSMS.
- Carga masiva de datos con BCP.
- Consulta de datos con SSMS.

Rúbrica para evaluar:

Actividad	10-5	5-0
configuración de una base de datos única mediante Azure Portal*	configuración de una base de datos única mediante Azure Portal*	No realizo la actividad.
Configuración de una regla de firewall por IP de nivel de servidor mediante Azure Portal	Configuración de una regla de firewall por IP de nivel de servidor mediante Azure Portal	No realizo la actividad.
Conéctese a la base de datos con SSMS	Conéctese a la base de datos con SSMS	No realizo la actividad.
Crear tablas con SSMS	Crear 5 tablas con SSMS	No realizo la actividad.
Carga masiva de datos con BCP	Carga masiva de datos con BCP	No realizo la actividad.
Consulta de datos con SSMS	Consulta de datos con SSMS	No realizo la actividad.

REPORTING SERVICES

- Reporting Services es toda una plataforma que nos permitirá explotar nuestros repositorios de datos a gran escala
- La plataforma de Reporting Services se compone de 4 cuatro elementos principalmente.
    - Una sección de Obtención de Datos.
    - Una sección de Diseño de Informes.
    - Una sección de Entrega de Informes.
    - Suscripciones a los diferentes Informes.

Obtención de datos

- Reporting Services puede extraer información prácticamente de cualquier origen de datos, nativamente se comunica con SQL server pero podemos acceder a bases de datos de terceros como MySQL, Oracle, Informix, etc. para el caso de Reporting Services 2005 también podemos consultar Analisys Services, por lo que podremos explotar los cubos generados

Diseño de informes

- Visual Studio.
- Hasta 2008, se instalaba un plug in que permitía generar proyectos con la plantilla de Reporting.
- A partir de 2010, se usa un complemento llamado Data Tools.

Ejemplo

- Los pasos para crear un informe son:
    - Abrir Visual Studio Data Tools.

- Crear un proyecto de Informes.
- Conectarse a un origen de datos.
- Generar una consulta a los datos.
- Ajustar los aspectos visuales en el diseñador.

Crear proyecto de Informe

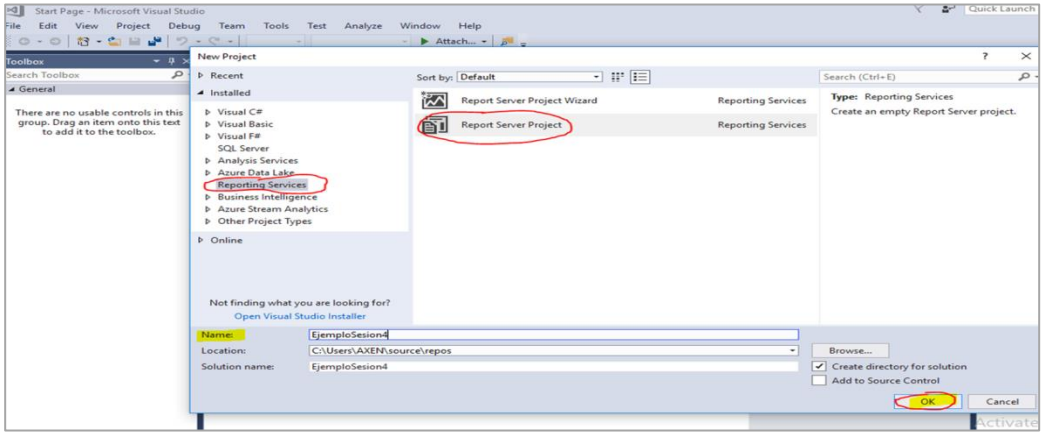


Figura 64 Creación de informe de servicios por asistente

Conectar origen de datos

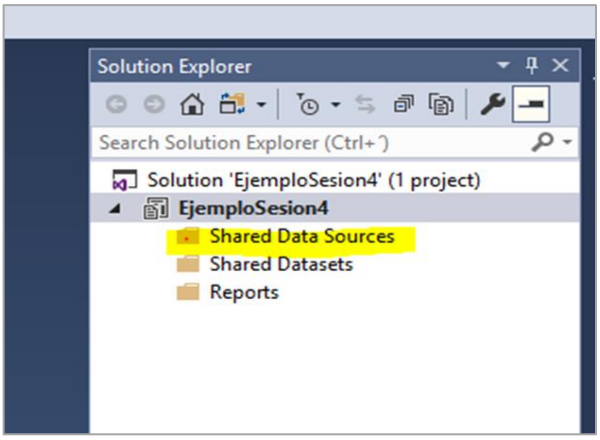


Figura 65 Especificación del origen de uso de los datos

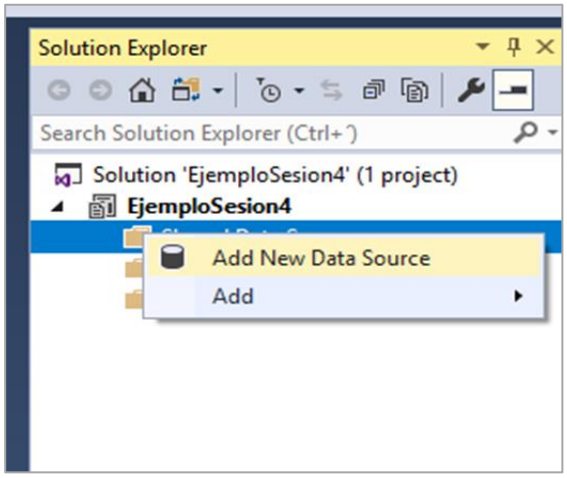


Figura 67 Agregar nuevo repositorio de datos

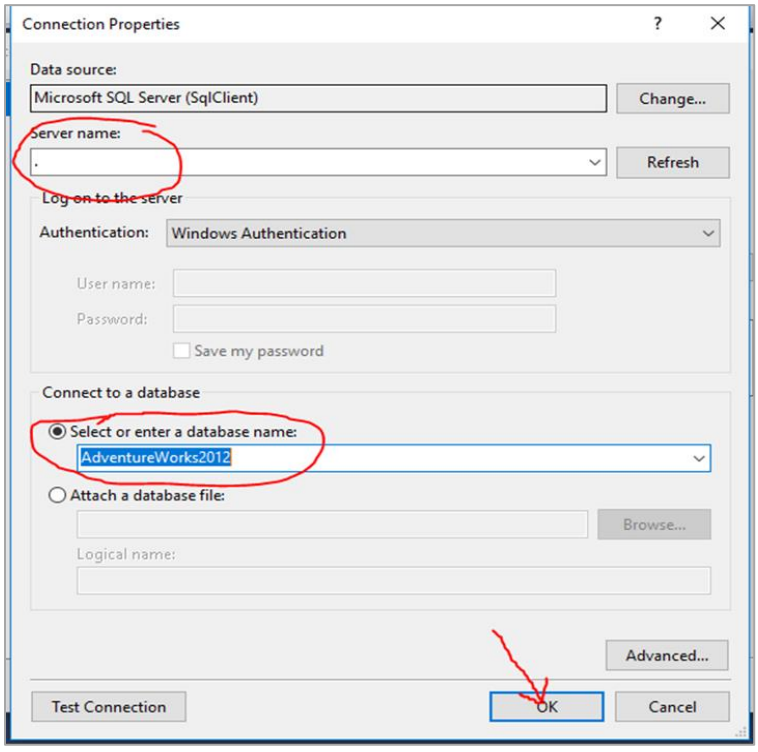


Figura 66 Pasos para crear la conexión del repositorio de datos

Agregar un Informe

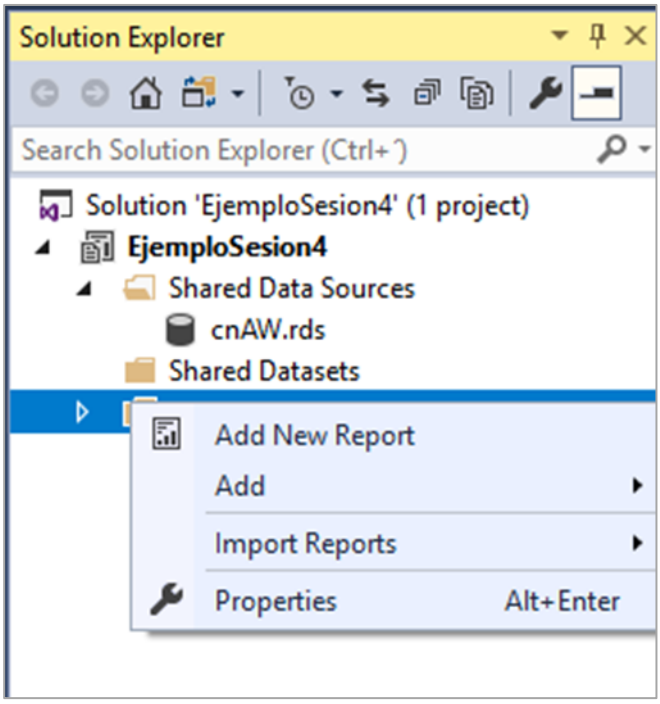


Figura 68 Creación de Reportes

Definir un conjunto de datos

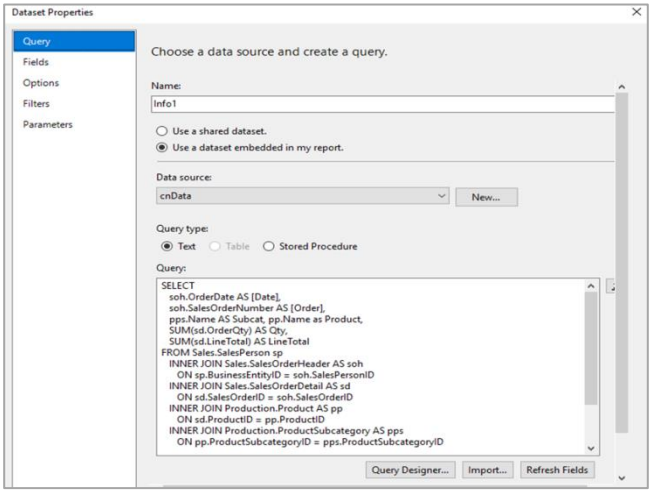


Figura 69 Definición de destino de la extracción de datos en el reporte

Diseñar el aspecto visual

- Añadimos una tabla de datos.
- Arrastramos los campos desde el conjunto.

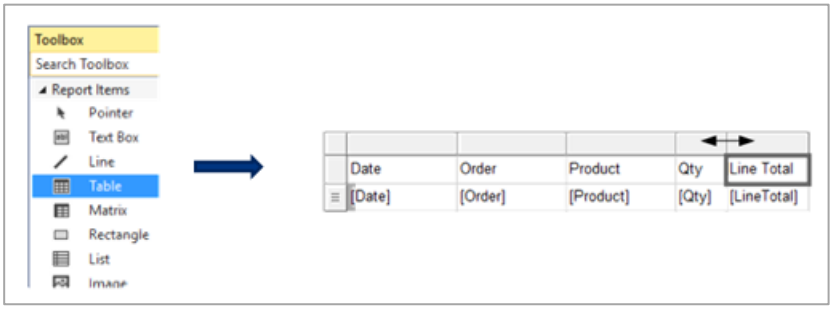


Figura 70 Creación de una tabla con objetos

Vista Previa

- En la parte superior del diseñador, está la pestaña de vista previa.

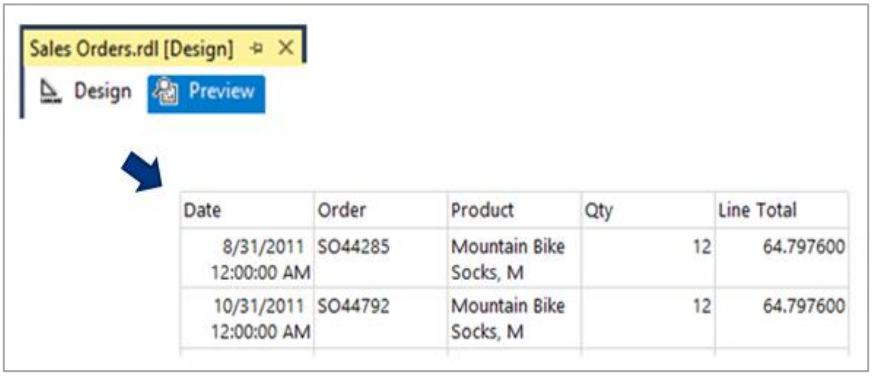


Figura 71 Vista previa del contenido de la tabla

Aplicar ajustes al informe

- Se puede aplicar formato a los campos.
- Se ajusta el tamaño de las columnas.



Figura 72 Visualización previa de la creación del reporte

EJERCICIO # 11

- Crear un proyecto de informe.
- Configurar una conexión de datos.
- Definir una consulta.
- Agregar una región de datos de table.
- Dar formato al informe.
- Agrupar y totalizar los campos.
- Obtener una vista previa del informe.

Rúbrica para evaluar:

Actividad	10	9	no acredita (cero)
Crear un proyecto de informe	Crear un proyecto de informe	Creo un informe no completo	No realizo la actividad.
Configurar una conexión de datos	Configurar una conexión de datos	Configuro una conexión de datos, no completo	no realizo la actividad
Definir una consulta	Definir una consulta	No definió una consulta completa	no realizo la actividad.
Agregar una región de datos de tabla	Agregar una región de datos de tabla	Agrego una región de datos no completa	no realizo la actividad.
Dar formato al informe	Dar formato al informe	No dio formato al informe completo.	no realizo la actividad.
Agrupar y totalizar los campos	Agrupar y totalizar los campos	Agrupo solo los campos	no realizo la actividad.

INTEGRATION SERVICES

Los servicios de integración de una base de datos SQL Server son una herramienta de integración, transformación y migración de datos empresariales desarrollada por Microsoft para la base de datos SQL Server. Puede ser usada para una variedad de tareas relacionadas con la integración, tales como análisis y depuración de datos, así como para extracción, transformación y carga de procesos de actualización de un data warehouse.

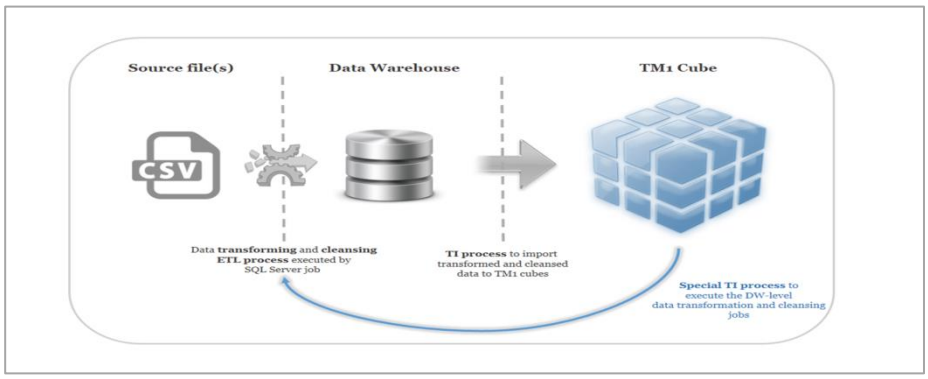


Figura 73 Representación gráfica de la integración de servicios

¿Qué es SSIS?

- SSIS es básicamente una poderosa herramienta para realizar tareas tipo ETL.
- Integration Serviceses más que un asistente para mover datos.
- Basado en Visual Studio.

¿Qué se puede hacer con SSIS?

- Cargar datos desde diferentes fuentes de datos.
    - Archivos planos.

    - Diferentes motores de bases de datos.
    - XML.
    - Excel.

- Limpiar y estandarizar datos.
- Aplicar lógica desea a los datos antes de cargarlos.
- Automatizar tareas administrativas de bases de datos.

¿Qué se necesita?

- SQL Server.
- SQL Server Data Tools.

Herramientas

- Control de flujo:
    - Componentes de gestión de flujos de procesos.
    - Manipulación de archivos.
    - Envío de correos.
- Flujo de datos:
    - Gestión de conexiones.
    - Transformaciones y limpieza de datos.

EJERCICIO # 12

- Crear un paquete ETL sencillo que extrae datos de un único archivo plano
- Transforma los datos con transformaciones de búsqueda
- Carga los resultados en un destino de tabla de hechos

Rubrica para evaluar:

Actividad	10	9	no acredita (cero)
Crear un paquete ETL sencillo que extrae datos de un único archivo plano	Crear un paquete ETL sencillo que extrae datos de un único archivo plano	Solo creo el paquete ETL.	No realizo la actividad.
Transforma los datos con transformaciones de búsqueda	Transforma los datos con transformaciones de búsqueda	Transforma los datos, pero no completo todo los pasos.	No realizo la actividad.
Carga los resultados en un destino de tabla de hechos	Carga los resultados en un destino de tabla de hechos	Cargo los resultados, pero no completo todos los pasos de la actividad.	No realizo la actividad.