Practica3: Servidores Vinculados y Particionamiento

Galicia Cobaxin Daniel
Pérez López Leonardo

June 2025

1. Script de cadabase de datos por región

```
    create database datoscovid_norte
    use datoscovid_norte
    SELECT * INTO datoscovid_norte FROM covidHistorico.dbo.
        ↪ datoscovid
    WHERE ENTIDAD_RES IN ('02','03','08','05','19','24','28','10','
        ↪ 25');
    create database datoscovid_centro
    use datoscovid_centro
    SELECT * INTO datoscovid_centro FROM covidHistorico.dbo.
        ↪ datoscovid
    WHERE ENTIDAD_RES IN ('01','11','22','24','32','14','15','09','
        ↪ 13','17','29','21');
    create database datoscovid_sur
    use datoscovid_sur
    SELECT * INTO datoscovid_sur FROM covidHistorico.dbo.datoscovid
    WHERE ENTIDAD_RES IN ('12','20','07','30','27','04','31','23','
        ↪ 16','06','18');
```

En MySQL se creó la base de datos covidHistorico y posteriormente se creo la tabla covidHistorico_sur, la cual fue llenada bajo un archivo .csv que contenía toda la información correspondiente a la región sur de México.

2. Linked_Servers y y consutas de prueba

```
    EXEC sp_addlinkedserver
       @server = 'MYSQL_SUR',                    -- nombre que le das
           ↪  al Linked Server
       @srvproduct = 'MySQL',                    -- nombre del
           ↪ proveedor (libre)
       @provider = 'MSDASQL',                    -- controlador ODBC
       @datasrc = 'mysql_covidHistorico_sur';              -- nombre
           ↪  del DSN creado (debe coincidir EXACTO)

    -- Configura las credenciales de acceso
```

```
8     EXEC sp_addlinkedsrvlogin
9        @rmtsrvname = 'MYSQL_SUR',              -- mismo nombre del
             ↪ linked server
10       @useself = 'false',
11       @locallogin = NULL,                     -- permite cualquier
             ↪ usuario local
12       @rmtuser = 'Alumno',             -- usuario de MySQL
13       @rmtpassword = 'Estudiante1';
14
15    SELECT *
16    FROM OPENQUERY(MYSQL_SUR, 'SELECT * FROM covidhistorico.
          ↪ covidhistorico_sur LIMIT 10');
17
18
19    EXEC sp_addlinkedserver
20       @server = 'SQL_NORTE',
21       @srvproduct = '',
22       @provider = 'SQLNCLI11',          -- o 'MSOLEDBSQL' si usas
             ↪ la versi n  m s reciente
23       @datasrc = '192.168.229.8';       -- IP o nombre del
             ↪ servidor remoto
24
25    EXEC sp_addlinkedsrvlogin
26       @rmtsrvname = 'SQL_NORTE',
27       @useself = 'false',
28       @locallogin = NULL,
29       @rmtuser = 'sa',
30       @rmtpassword = 'estudiante1';
31
32
33    SELECT *
34    FROM OPENQUERY(SQL_NORTE, 'SELECT TOP 10 * FROM
          ↪ datoscovid_norte.dbo.datoscovid_norte');
35
36
37    EXEC sp_addlinkedserver
38        @server = 'SQL_CENTRO',
39        @srvproduct = '',
40        @provider = 'SQLNCLI11',           -- o 'MSOLEDBSQL' si lo
              ↪ prefieres
41        @datasrc = 'localhost';            -- tambi n puede ser
              ↪ '.' o el nombre de tu instancia
42
43    EXEC sp_addlinkedsrvlogin
44        @rmtsrvname = 'SQL_CENTRO',
45        @useself = 'true';                 -- usa las mismas
              ↪ credenciales del usuario conectado
46
47    SELECT *
48    FROM OPENQUERY(SQL_CENTRO, 'SELECT TOP 10 * FROM
          ↪ datoscovid_centro.dbo.datoscovid_centro');
```

3. Consultas distribuidas

3.1. Consulta 3

```
-- Consulta 3
SELECT
    CAST(SUM(Total_Diabetes) * 100.0 / SUM(Total) AS DECIMAL
        ↪ (5,2)) AS Porcentaje_Diabetes,
    CAST(SUM(Total_Obesidad) * 100.0 / SUM(Total) AS DECIMAL
        ↪ (5,2)) AS Porcentaje_Obesidad,
    CAST(SUM(Total_Hipertension) * 100.0 / SUM(Total) AS
        ↪ DECIMAL(5,2)) AS Porcentaje_Hipertension
FROM (
    SELECT
        COUNT(*) AS Total,
        SUM(CASE WHEN DIABETES = 1 THEN 1 ELSE 0 END) AS
            ↪ Total_Diabetes,
        SUM(CASE WHEN OBESIDAD = 1 THEN 1 ELSE 0 END) AS
            ↪ Total_Obesidad,
        SUM(CASE WHEN HIPERTENSION = 1 THEN 1 ELSE 0 END) AS
            ↪ Total_Hipertension
    FROM OPENQUERY(SQL_NORTE, '
        SELECT DIABETES, OBESIDAD, HIPERTENSION
        FROM datoscovid_norte.dbo.datoscovid_norte
        WHERE CLASIFICACION_FINAL IN (1,2,3)
    ')

    UNION ALL

    SELECT
        COUNT(*),
        SUM(CASE WHEN DIABETES = 1 THEN 1 ELSE 0 END),
        SUM(CASE WHEN OBESIDAD = 1 THEN 1 ELSE 0 END),
        SUM(CASE WHEN HIPERTENSION = 1 THEN 1 ELSE 0 END)
    FROM OPENQUERY(SQL_CENTRO, '
        SELECT DIABETES, OBESIDAD, HIPERTENSION
        FROM datoscovid_centro.dbo.datoscovid_centro
        WHERE CLASIFICACION_FINAL IN (1,2,3)
    ')

    UNION ALL

    SELECT
        COUNT(*),
        SUM(CASE WHEN DIABETES = 1 THEN 1 ELSE 0 END),
        SUM(CASE WHEN OBESIDAD = 1 THEN 1 ELSE 0 END),
        SUM(CASE WHEN HIPERTENSION = 1 THEN 1 ELSE 0 END)
    FROM OPENQUERY(MYSQL_SUR, '
        SELECT DIABETES, OBESIDAD, HIPERTENSION
        FROM covidhistorico.covidHistorico_sur
        WHERE CLASIFICACION_FINAL IN (1,2,3)
    ')
) AS SubTotales;
```

3.2. Consulta 4

```
-- Consulta 4
SELECT c.entidad AS Estado, COUNT(*) AS
    ↪ Total_Casos_Recuperados_Neumonia
FROM (
    -- Nodo Norte (SQL Server)
    SELECT
        CAST(ENTIDAD_RES AS varchar(2)) AS ENTIDAD_RES,
        CAST(CLASIFICACION_FINAL AS int) AS CLASIFICACION_FINAL
            ↪ ,
        CAST(NEUMONIA AS int) AS NEUMONIA
    FROM OPENQUERY(SQL_Norte, '
        SELECT
            ENTIDAD_RES,
            CLASIFICACION_FINAL,
            NEUMONIA
        FROM datoscovid_norte.dbo.datoscovid_norte
        WHERE CLASIFICACION_FINAL = 3
          AND NEUMONIA = 1
    ')

    UNION ALL

    -- Nodo Centro (SQL Server)
    SELECT
        CAST(ENTIDAD_RES AS varchar(2)) AS ENTIDAD_RES,
        CAST(CLASIFICACION_FINAL AS int) AS CLASIFICACION_FINAL
            ↪ ,
        CAST(NEUMONIA AS int) AS NEUMONIA
    FROM OPENQUERY(SQL_Centro, '
        SELECT
            ENTIDAD_RES,
            CLASIFICACION_FINAL,
            NEUMONIA
        FROM datoscovid_centro.dbo.datoscovid_centro
        WHERE CLASIFICACION_FINAL = 3
          AND NEUMONIA = 1
    ')

    UNION ALL

    -- Nodo Sur (MySQL)
    SELECT
        CAST(ENTIDAD_RES AS varchar(2)) AS ENTIDAD_RES,
        CAST(CLASIFICACION_FINAL AS int) AS CLASIFICACION_FINAL
            ↪ ,
        CAST(NEUMONIA AS int) AS NEUMONIA
    FROM OPENQUERY(MYSQL_SUR, '
        SELECT
            ENTIDAD_RES,
            CLASIFICACION_FINAL,
            NEUMONIA
        FROM covidhistorico.covidHistorico_sur
        WHERE CLASIFICACION_FINAL = 3
          AND NEUMONIA = 1
    ')
```

```
    ) d
    JOIN cat_entidades c ON CAST(d.ENTIDAD_RES AS varchar(2)) =
        ↪ CAST(c.clave AS varchar(2))
    WHERE d.CLASIFICACION_FINAL = 3
      AND d.NEUMONIA = 1
    GROUP BY c.entidad
    ORDER BY Total_Casos_Recuperados_Neumonia DESC;
```

3.3. Consulta 5

```
    -- Consulta 5

    SELECT Estado, SUM(Total) AS Total_Casos_Recuperados_Neumonia
    FROM (
        SELECT CAST(c.entidad AS VARCHAR(100)) AS Estado, COUNT(*)
            ↪ AS Total
        FROM OPENQUERY(SQL_Norte, '
            SELECT d.ENTIDAD_RES
            FROM datoscovid_norte.dbo.datoscovid_norte d
            WHERE d.CLASIFICACION_FINAL = 3 AND d.NEUMONIA = 1
        ') d
        JOIN cat_entidades c
            ON CAST(d.ENTIDAD_RES AS VARCHAR(10)) = CAST(c.clave AS
                ↪ VARCHAR(10))
        GROUP BY CAST(c.entidad AS VARCHAR(100))

        UNION ALL

        SELECT CAST(c.entidad AS VARCHAR(100)), COUNT(*)
        FROM OPENQUERY(SQL_Centro, '
            SELECT d.ENTIDAD_RES
            FROM datoscovid_centro.dbo.datoscovid_centro d
            WHERE d.CLASIFICACION_FINAL = 3 AND d.NEUMONIA = 1
        ') d
        JOIN cat_entidades c
            ON CAST(d.ENTIDAD_RES AS VARCHAR(10)) = CAST(c.clave AS
                ↪ VARCHAR(10))
        GROUP BY CAST(c.entidad AS VARCHAR(100))

        UNION ALL

        SELECT CAST(c.entidad AS VARCHAR(100)), COUNT(*)
        FROM OPENQUERY(MYSQL_SUR, '
            SELECT d.ENTIDAD_RES
            FROM covidhistorico.covidHistorico_sur d
            WHERE d.CLASIFICACION_FINAL = 3 AND d.NEUMONIA = 1
        ') d
        JOIN cat_entidades c
            ON CAST(d.ENTIDAD_RES AS VARCHAR(10)) = CAST(c.clave AS
                ↪ VARCHAR(10))
        GROUP BY CAST(c.entidad AS VARCHAR(100))
    ) AS Resultados
    GROUP BY Estado
    ORDER BY Total_Casos_Recuperados_Neumonia DESC;
```

3.4. Consulta 7

```
-- Consulta 7
WITH CasosMensuales AS (
    SELECT
        YEAR(CAST(d.FECHA_INGRESO AS date)) AS A o ,
        MONTH(CAST(d.FECHA_INGRESO AS date)) AS Mes,
        c.entidad AS Estado,
        SUM(CASE WHEN d.CLASIFICACION_FINAL IN (1, 2, 3, 6)
            ↪ THEN 1 ELSE 0 END) AS Total_Casos
    FROM (
        -- Datos del Norte (SQL Server)
        SELECT
            FECHA_INGRESO,
            CAST(ENTIDAD_RES AS varchar(2)) AS ENTIDAD_RES,
            CAST(CLASIFICACION_FINAL AS int) AS
                ↪ CLASIFICACION_FINAL
        FROM OPENQUERY(SQL_Norte, 'SELECT
            TRY_CONVERT(date, FECHA_INGRESO) AS FECHA_INGRESO,
            ENTIDAD_RES,
            CLASIFICACION_FINAL
        FROM datoscovid_norte.dbo.datoscovid_norte
        WHERE TRY_CONVERT(date, FECHA_INGRESO) IS NOT NULL')

        UNION ALL

        -- Datos del Centro (SQL Server)
        SELECT
            FECHA_INGRESO,
            CAST(ENTIDAD_RES AS varchar(2)) AS ENTIDAD_RES,
            CAST(CLASIFICACION_FINAL AS int) AS
                ↪ CLASIFICACION_FINAL
        FROM OPENQUERY(SQL_Centro, 'SELECT
            TRY_CONVERT(date, FECHA_INGRESO) AS FECHA_INGRESO,
            ENTIDAD_RES,
            CLASIFICACION_FINAL
        FROM datoscovid_centro.dbo.datoscovid_centro
        WHERE TRY_CONVERT(date, FECHA_INGRESO) IS NOT NULL')

        UNION ALL

        -- Datos del Sur (MySQL)
        SELECT
            CAST(FECHA_INGRESO AS date) AS FECHA_INGRESO,
            CAST(ENTIDAD_RES AS varchar(2)) AS ENTIDAD_RES,
            CAST(CLASIFICACION_FINAL AS int) AS
                ↪ CLASIFICACION_FINAL
        FROM OPENQUERY(MYSQL_SUR, 'SELECT
            STR_TO_DATE(FECHA_INGRESO, ''%Y-%m-%d'') AS
                ↪ FECHA_INGRESO,
            ENTIDAD_RES,
            CLASIFICACION_FINAL
        FROM covidhistorico.covidHistorico_sur
        WHERE STR_TO_DATE(FECHA_INGRESO, ''%Y-%m-%d'') IS NOT
            ↪ NULL')
    ) d
```

```
49          JOIN cat_entidades c ON CAST(d.ENTIDAD_RES AS varchar(2)) =
                ↪  CAST(c.clave AS varchar(2))
50          WHERE d.CLASIFICACION_FINAL IN (1, 2, 3, 6)
51            AND YEAR(CAST(d.FECHA_INGRESO AS date)) IN (2020, 2021)
52          GROUP BY YEAR(CAST(d.FECHA_INGRESO AS date)), MONTH(CAST(d.
                ↪ FECHA_INGRESO AS date)), c.entidad
53      ),
54      RankingMensual AS (
55          SELECT
56              A o ,
57              Estado,
58              Mes,
59              Total_Casos,
60              RANK() OVER (PARTITION BY A o , Estado ORDER BY
                    ↪ Total_Casos DESC) AS Ranking
61          FROM CasosMensuales
62      )
63      SELECT
64          A o ,
65          Estado,
66          Mes,
67          Total_Casos
68      FROM RankingMensual
69      WHERE Ranking = 1
70      ORDER BY A o , Estado;
```

4. Conclusiones

En esta práctica se han aprendido muchas cosas, ademas de que se han reforzado algunas otras. Principalmente aprendimos a como particionar una base de datos de la mejor manera y lo que más me pareció interesante fue como conectar varios servidores (nodos) entre sí, sin importar que fueran MySQL o SQL Server. Para empezar esta prática, investigue un poco sobre Linked Servers en SQL Server, cómo funcionan y cuál es el proceso de crealos y establecer una conexión entre distintos nodos. Obvio esto fue visto previamente en clase pero suele pasar que no recuerdo todo las configuraciones que se hacen para lograr una conexión exitosa.

El primer problema al que nos enfrentamos como equipos fue que, solo éramos dos, y para realizar la práctica se necesitaban al menos 3 integrantes o tres equipos de cómputo (nosotros contabamos con dos equipos más uno de escritorio). Para resolver esto lo que hice fue, instalar dos instancias en mi computadora de escritorio una MYSQL y otra SQL Server, y una SQL en mi laptop personal.

El proceso de realizar la conexión entre nuestros nodos no fue muy complicado, ya que en mi red no tengo ningún tipo de restricción (a comparación de la escuela), entonces, ahí fue mucho más sencillo realizar la práctica para posteriormente llervarla a las computadoras de la escuela.

El proceso de migrar una base de datos SQL Server a MySQL es un poco más çomplejo", pues tuvimos que crear un archivo tipo .csv para migrar los datos, pero como son millones de filas tardó aproximadamente unas 3 horas, a dife-

rencia de SQL Server a SQL Server, donde sólo hay que hacer un backup full y restaurar en donde queramos.
Para restaurar las bases de datos en las computadoras de la escuela, fue un proceso más dificil ya que la red y dichas computadoras tienen muchas restricciones, para lo cual el profesor nos ayudó dandonos acceso desde un usuario administrador. Finalmente el proceso fue el mismo de siempre, crear un dsn del sistema para establecer la conexión con MYSQL usando el conector odbc, para SQL Server esto no fue necesario ya que unicamente con la IP de la otra computadora se puede establecer. Mucho ojo, ya que si no hay puertas de entrada y salida establecidas o un firewall mal configurado, no se va a poder realizar la conexión entre nodos.
Para la consultas, usamos la misma logica que aplicamos en la práctica 1, solo que de forma distribuida, primeramente, al probar nuestras consultas, nos dimos cuenta que estas demoraban demasiado, algunas hasta 5 minutos, por lo cual optamos por recurrir a una IA para que nos ayudará a optimizar alguas de estas consultas, claro simpre de manera en que nosotros puediésemos entender que es lo que está realizando y todas la funciones que ocupa.
En estas consultas distribuidas ocupamos OPENQUERY en lugar del nombramiento de cuatri partes, esto nosotros lo elegimos por facilidad, ya que en la sintáxis unicamente necesitamos en nombre del LS y la consulta, pero si investigamos más a fondo, encontramos que el nombramiento de 4 partes nos ayuda para consultas simples y directas, util para operaciones básicas como joins, filtros, etc., mientras que OPENQUERY es más eficiente ya que se ejecuta directo en el servidor remoto, filtrando antes de enviar los datos, además acepta sintaxis del motor remoto, util para consultas más complejas.
Por ultimo, hay que tener cuidado con los tipos de datos entre diferentes motores de bases de datos, ya que algunos no son compatibles entre sí y tendrás que hacer muchos CAST, para hacer la conversión de tipo de datos.