Proyecto II - GreenTec

Emmanuelle Aguilar Sánchez 2016009338
Alexis Gavriel Gómez 2016085662
Erick Obregón Fonseca 2016123157
Andrés Ramírez Quirós 2016142049

27 de octubre de 2018

1. Infraestructura

Para crear el cluster, se deben crear 3 máquinas virtuales, en nuestro caso se utilizó el programa VmWare Workstation 15. Un aspecto importante a considerar a la hora de realizar la creación de las máquinas virtuales es seleccionar la conexión de tipo 'Bridge'. Estp para puentear la red entre la máquina virtual y la máquina física. Dichas máquinas virtuales deben tener instalado el sistema operativo Windows Server preferibelemente una versión posterior a la 2008.

Una vez configuradas las máquinas virtuales, se debe proceder a desactivar el firewall de cada una de las máquinas para evitar problemas de conectividad. Luego se debe instalar los componentes de .NetFramework 3.5 en cada una de las máquinas y para los nodos del clúster los componentes de 'Fail Over Cluster'. Para mayor cómodiad, se recomienda cambiar el nombre de las máquinas para poder identificarlas mejor y para posteriores pasos en el proceso de instalación.

Cuando todo lo anterior se haya instalado, se debe proceder a crear el controlador del dominio el cual, tendrá los discos compartidos y proveera el dominio de Active Directory de Windows. Para ello, se debe instalar los componentes de Active Directory Domian Services en la máquina virtual que servirá de controlador del dominio, una vez instalado, se debe configurar el dominio al que se conectarán los nodos posteriormente, se debe configurar el DNS.

Luego de configurar el controlador de dominio, se debe proceder a unir los nodos a dicho dominio. Para ello se debe ir a la configuración de red de los nodos y cambiar la IP de estos por alguna que posteriormente se utilizará para configurar los discos compartidos. También se debe cambiar el DNS de los nodos por la dirección IP del controlador del dominio.

Para la configuración de los discos compartidos, se debe configurar el ISCSI en el controlador del dominio. Se deben crear los discos con su respectivo espacio asignado, nombre y su respectiva letra además de, los targets, que serán las máquinas las cuales usarán dichos discos compartidos. Una vez configurado, se debe seleccionar, en los nodos, el iniciador de ISCSI e indicar la dirección IP del controlador del dominio. Cuando se haya configurado esto, se debe ir al administrador de discos de Windows y poner la letra y nombre a los que se especificó en el controlador del dominio. Es importante que ambos nodos tengan el mismo nombre en los discos compartidos que el controlador de dominio para evitar problemas a la hora de configurar el clúster con Fail Over.

Siguiendo con la configuración, se deben seleccionar las herramientas de Fail Over en el Administrador del Servidor en uno de los nodos, para configurar el clúster. Primero se debe validar si se cumplen todas las condiciones necesarias para la creación del clúster.

Cuando el clúster esté configurado, se procede a instalar SQL Server, 2017 en nuestro caso, en modo clúster. Se debe configurar el nombre de la red virtual de SQL, la IP que tendrá y los servicios de SQL que se instalarán. Es importante verificar muy bien que los servicios y características de SQL se han instalado, ya que una vez hecho este proceso, instalar nuevas características o servicios de SQL requiere de un largo y tedioso proceso. Cuando se haya configurado SQL en uno de los nodos, se procede a instalar SQL Server en los nodos restantes con la excepción de que, se debe seleccionar la opción de agregar nodo a un clúster existente, cuando se está instalando SQL Server.

Una vez todo listo, ya se podrá acceder y utilizar el servidor y sus bases de datos. Es importante verificar que el rol de SQL Server instalado en el clúster se esté ejecutando, ya que este es el que maneja todo lo relacionado con las consultas y servicios que realiza el clúster.

2. Bases de datos relacional

La base de datos relacional se realizó con ayuda del diagramador de Sql Server Management Studio. Se siguieron las especificaciones del documento para determinar las entidades, tablas, y relaciones necesarias para producir un diagrama de base de datos que logre cumplir con las solicitudes del documento.

Se consideró la opción de crear índices de tipo Non-Clustered para optimizar las consultas de la cantidad de parques por comunidad y la cantidad de área que hay en un parque por cada comunidad.

En lo que concierne a la generación de los datos, éstos fueron generados en un editor de texto. Gracias a la ayuda de librerías en el editor de texto se generaron datos pseudoaleatorios.

Para la parte de cargar los datos en el servidor utilizando BulkCopy se crearon consultas que cargan los datos almacenados en archivos planos. Debido a una diferencia en la cantidad de columnas de los datos creados y las columnas físicamente en la base de datos se requirió el uso de archivos de formato. Los archivos de formato (.fmt) indican cómo se debe cargar los datos de un archivo plano en una tabla. Permiten indicar los caracteres separadores de filas y columnas, así como indicar el formato de los caracteres de string y cuál columna del archivo plano se transferirá a una determinada columna en la tabla del servidor.

3. ETL a Base de datos de grafos

Para el ETL (Extract Transform and Load), se creó un un programa básico en Visual Studio 2017. El programa consiste en tres etapas. Las primeras dos etapas extraen de la base de datos activa dos tablas a partir de consultas, para reducir el tamaño de los datos extraídos. Se extraen las tablas de Cadena Alimenticia y varias columnas de la tabla Especie. Estas tablas y datos se guardan en sus respectivos archivos CSV. La tercera etapa del programa crea una base de datos de grafos en Neo4j. Esta base de datos contiene los datos de los archivos extraídos. Cada nodo del grafo representa una especie, y cada relación entre nodos representa la cadena alimenticia entre dos especies. Esto se realiza gracias a un programa en Java que se encarga de crear la base de datos de grafos a partir de las consultas tomadas de un archivo ".cypher". Estos últimos archivos contienen las consultas a realizar en la base de datos de Neo4j. Como son por ejemplo: cargar archivos, colocar datos en nodos, clasificar nodos, realizar relaciones entre nodos, entre otros.

4. Replication Services

Para los replication services, es importante haber instalado este servicio en el instalador de SQL Server. Para crear una replicación, se debe dar click derecho en la carpeta Replication de SSMS y crear una nueva publicación. La publicación la debe realizar el servidor que va a replicar la base, existen 4 tipos de replicación, para el caso de una replicación bidireccional se debe seleccionar Merge Replication. Una vez realizado los pasos pedidos por el asistente, se debe crear la subscripción, para esto es importante tener conexión con el servidor que tiene la base que consumirá la replicación. Para ello se debe crear una subscripción en la carpeta Replication de SSMS y seguir los pasos del asistente. Un aspecto importante es que se puede ver el estado de la sincronización seleccionando la publicación, luego la subscripción y dando click derecho en View Synchronization Status.

5. Scripts

5.1. Vistas

Las vistas son tablas temporales que se crean en el momento que llega la consulta, por lo que siempre está actualizadas con la última información disponible en la base, y se destruyen automáticamente cuando se dejan de utilizar. Las vistas se pueden componer de distintas columnas de diferentes tablas, según se vea necesario.

Por lo general, las vistas son utilizadas para presentar la información de una manera más concisa pues se ocultan detalles de implementación como los id's y otros datos, que dependiendo del tipo de usuario son datos que no interesa. Además, ofrece una facilidad para manejar la privacidad y la seguridad de la base de datos, pues se puede limitar accesos a usuarios con pocos permisos para que sólo tengan acceso a información poco comprometedora para la empresa o usuarios con más permisos de la base.

5.2. Cursor

Los cursores son tablas temporales que se crean en ocasiones muy particulares debido a la alta ineficiencia de estas y a la posibilidad de usar en la mayoría de las ocasiones medidas alternativas para evitarlos. El grado de ineficiencia de la operación es comparable con el de un while o un for, en cual para bases de datos más allá que una de prueba, causa una gran perdida de eficiencia en la base de datos.

Los cursores pueden ser locales, los cuales se crean en un query, como un Stored Procedure, se utilizan y se destruyen al finalizar. Los globales, son creados de la misma forma pero no existe una necesidad de cerrarlos cuando se termina la función, lo cual da la posibilidad de que sean usados por otros procedimientos que lo puedan llegar a ocupar.

5.3. Optimización Execution Plan

Para el query en el cual se requiere el uso de múltiples funciones, subqueries y otras especificaciones adicionales, se diseñó una consulta que mostrara el tamaño del lado del parque (asumiendo una forma cuadrada para este) y la cantidad de diferentes especies que viven en ella, a excepción de aquellos parques que posean un alojamiento de tipo 'Resort and Spa'; además de una clasificación entre la primera mitad de los hoteles, según el índice correspondiente, y la segunda mitad.

Para lograrlo, se utilizó una combinación de case, subqueries, inner joins, agrupar el resultado por parques y ordenar según el tamaño del lado de cada uno. El uso de wildcard para evaluar la categoría del alojamiento pues no se sabe su id. Una vez accedidas a las tablas necesarias para obtener los datos de consulta, se utilizan diversas funciones para obtener el resultado buscado (sum, sqrt, count, convert...).

Para optimizar el query, se cambiaron algunos subqueries anidados por joins, además de utilizar un índice para la columna de extensión debido a la gran cantidad de consultas que se le hacen a este dato. Esto es un cambio entre recursos de procesamiento pues se utiliza más memoria para guardar los índices, pero es menor la carga para el procesador.

5.4. Optimización de la consulta del área parque por comunidad

Para realizar la optimización de la consulta para obtener el area total de parques por comunidad por medio del noncluster index se realizaron dos pruebas. La primer prueba era ver la duración de la consulta sin el index creado, lo cual generaba un tiempo de duración mayor. En la segunda consulta, se procedió a crear el noncluster index en la tabla AreaParque en la columna Extensión. Luego de terminar la consulta se borraba el index antes creado y se veía el tiempo de ejecución.

5.5. Optimización de la consulta de número de parques por comunidad

Para realizar la optimización de la consulta para obtener el número de parques por comunidad por medio del noncluster index se realizaron dos pruebas. La primer prueba era ver la duración cuanto duraba la consulta sin el index creado. En la segunda consulta, se procedió a crear el noncluster index en la tabla Parque en la columna idParque. Luego de terminar la consulta se borraba el index antes creado y se veía el tiempo de ejecución. Dicha prueba resultaba en un tiempo de ejecución menor.

6. Mantenimiento

Para realizar el plan de mantenimiento, se debe dar click en Managment de SSMS, click derecho en Maintenance Plan y luego click en Maintenance Plan Wizard. Se deben seguir los pasos del asistente para elegir el horario del plan y el tipo de back up. Existen dos tipos, full backup y differential backup. El full hace un backup completo de la base de datos y el differential hace un backup de los últimos datos cambiados desde que se hizo el full backup

7. Seguridad

Para crear un usuario nuevo, se debe dar click derecho en Security/Logs en el asistente, se puede seleccionar la base de datos que verá el usuario, los roles que tendrá y las acciones que podrá hacer en la base de datos. Esto último se puede hacer de dos formas, mediante el User Mapping del asistente de creación de usuarios, en él se especifíca cuales acciones podrá realizar, por ejemplo, datareader permite sólo leer y sino se selecciona datawriter no podrá escribir. La otra forma es creando roles, en estos se puede especificar que tipo de datos podrá ver el usuario por ejemplo, que tablas y stored procedures podrá ver el usuario.

8. Concurrencia

8.1. ISOLATION LEVEL

Las transacciones especifican un nivel de aislamiento que define el grado en que se debe aislar una transacción de las modificaciones de recursos o datos realizadas por otras transacciones. Los niveles de aislamiento se describen en función de los efectos secundarios de la simultaneidad que se permiten, como las lecturas de datos sucios o las lecturas fantasmas.

8.1.1. READ COMMITTED

Especifica que las declaraciones no pueden leer datos que han sido modificados y que aún no han sido commiteados por otras transacciones. Esto evita lecturas sucias. Los datos pueden ser cambiados por otras transacciones entre declaraciones individuales dentro de la transacción actual, dando como resultado lecturas no repetibles o datos fantasmas.

Para solucionar este problema se realizaron dos transacciones por separado. La primera transacción realiza un inserción en una tabla de la base de datos, luego espera unos segundos y solicita todos los datos en dicha tabla. Para la segunda transacción se solicitan todos los datos de la tabla. En este caso, como la primer transacción está escribiendo en la base de datos la segunda no puede realizar el query hasta que la primer transacción de termine.

8.1.2. READ UNCOMMITTED

Especifica que las consultas o declaraciones pueden leer filas que hayan sido modificadas por otras transacciones pero que aún no se han hecho commit.

Para la primer transacción, se inserta un valor dentro de una de las tablas de la base de datos y se deja esperando un cierto tiempo, luego se realiza el commit de la transacción y se piden todos los valores dentro de dicha tabla. Para la segunda transacción, se leen todos los datos de la tabla antes modificada. Como el nivel de aislamiento seleccionado fue READ UNCOMMITTED, dicha transacción puede ver los datos dentro de la tabla sin ningún tipo de lock.

8.1.3. REPEATABLE READ

Especifica que las declaraciones no pueden leer datos que se han modificado y que aún no han sido commiteados por otras transacciones, así como que ninguna otra transacción puede modificar datos los cuales hayan sido leídos por la transacción actual hasta que se complete la transacción vigente.

En la primer transacción, se inserta un valor dentro una de las tablas de la base de datos y se deja esperando un cierto tiempo y después se realiza el commit. Para la segunda transacción, se realizó una lectura de datos de la tabla anteriormente modificada, dejando la consulta en espera hasta que la anterior transacción termine.

8.1.4. SERIALIZABLE

Especifica lo siguiente:

- Las consultas o transacciones no pueden leer datos que se hayan modificado pero que aún no hayan sido commiteados por otras transacciones.
- Ninguna otra transacción puede modificar los datos que han sido leídos por la transacción actual hasta que la transacción actual se complete.
- Otras transacciones no pueden insertar nuevas filas con valores clave que estarían dentro del rango de claves leídas por cualquier declaración en la transacción actual hasta que la transacción actual se complete.

Para la primer consulta, se seleccionan todos los valores de una tabla de la base de datos donde su id sea mayor que 3. Para probar el nivel de aislamiento Serializable se realizaron dos transacciones, Una de ellas modifica la edad de una persona en la tabla Persona donde su id sea 2, de esta manera la consulta es ejecutada y no se bloquea. La otra consulta, modifica la edad de una persona en la tabla Persona donde su id sea 4, de esta manera la consulta es ejecutada y es bloqueda por ser uno de los valores que están siendo tomados por la primer consulta. Cuando dicha transacción termina, se ejecuta el query en espera.

8.2. Rollback

Restaura una transacción explícita o implícita al principio de la transacción, o a un punto de salvaguarda dentro de la transacción. Puede utilizar ROLLBACK TRANSACTION para borrar todas las modificaciones de datos realizadas desde el inicio de la transacción o en un punto de salvaguarda. También libera recursos en poder de la transacción.

Para realizar el rollback se realizó una consulta con todos los datos de la tabla y se puso la consulta en tiempo de espero lo suficiente para generar el timeout. Para evitar el error, se usó la sentencia Try-Catch. Cuando el error es detectado se genera un rollback.

8.3. Cursor

Para probar que un cursor bloque la lectura del otro mientras actualiza los datos, se crea un cursor de tipo SCROLL_LOCKS, el cual toma todos los datos de la tabla y los va modificando uno por uno mientras espera dos segundos por actualización (esto con el fin de que se note más el bloque de la lectura). Después de eso, se pone a ejecutar el segundo cursor, el cual va leyendo toda la información de la base de datos, pero como actualmente se está ejecutando la primer transacción, esta consulta queda en espera hasta que la primera termine y así puede seguir mostrando los datos.