Universidad CENFOTEC

Maestría en Tecnología de Bases de Datos

Documento final de Proyecto de Investigación Aplicada 2

Propuesta de un Proceso de Monitoreo en Ambientes de Bases de Datos SQL Server

Solano Alpízar Andrés

Fecha: Abril, 2021

Declaratoria de Derecho de Autor

Declaro que el presente proyecto de investigación fue realizado en su totalidad por el autor Andrés Solano Alpízar, con base en indagación en Internet, literatura referente al tema y los conocimientos adquiridos de experiencias previas de proyectos similares al área de administración de bases de datos e inteligencia de negocios en SQL Server.

En caso de que se haya realizado referencias a definiciones específicas de diversos autores, se ha procedido a indicarlas, a fin de no violentar los derechos de autor.

Se autoriza la reproducción total o parcial de este trabajo, únicamente con fines exclusivos de tipo académico y científico, por cualquier medio o procedimiento, incluye la cita bibliográfica del documento, al respetar los derechos del presente autor.

Dedicatoria

Dedico este trabajo a mi madre, padre, hermana, abuela y tío quienes siempre me brindaron su cariño, apoyo y ayuda durante todo este proceso académico.

TRIBUNAL EXAMINADOR

Este proyecto fue aprobado por el Tribunal Examinador de la carrera: **Maestría en Tecnología de Bases de Datos,** requisito para optar por el título de grado de **Maestría,** para el estudiante: **Andrés Solano Alpízar.**

JOSE ALBERTO CABEZAS JAIKEL (FIRMA)
Digitally signed by JOSE ALBERTO CABEZAS JAIKEL (FIRMA)
Date: 2021.04.17 10:08:53 -06'00'

MBD. José Cabezas Jaikel
Tutor

MARISOL NUÑEZ VASQUEZ (FIRMA)
Firmado digitalmente por MARISOL NUÑEZ VASQUEZ (FIRMA)
Fecha: 2021.04.18 21:36:08 -06'00'

M. Sc. Marisol Núñez Vásquez
Lector 1

IGNACIO TREJOS ZELAYA (FIRMA)
Firmado digitalmente por IGNACIO TREJOS ZELAYA (FIRMA)
Fecha: 2021.04.19 08:57:59 -06'00'

M. Sc. Ignacio Trejos Zelaya
Lector 2

San José, Costa Rica, 16 de abril de 2021

Firmada digitalmente, de conformidad con la Ley de Certificados, Firmas Digitales y Documentos Electrónicos N° 8454, destacando el artículo 9°-

Tabla de Contenidos

Lista de Figuras

Lista de Tablas

Resumen Ejecutivo

Actualmente, existen muchas herramientas de monitoreo de bases de datos las cuales ayudan a los administradores de bases de datos en su trabajo diario para mantener los ambientes optimizados y que funcionen correctamente para satisfacer las necesidades de los diferentes negocios que conforman una empresa.

Por lo general, estas herramientas tienen un costo bastante alto, lo cual hace que varias empresas no puedan costear este tipo de productos o en otras ocasiones no quieran invertir en este tipo de software para ayudar a los administradores de bases de datos en su trabajo.

El monitoreo de bases de datos no es un lujo, sino necesidad, aI presente, los datos se han convertido en unos de los activos más importantes para las diferentes empresas alrededor del mundo, por lo cual poder realizar soporte proactivo puede mejorar la disponibilidad y la salud de las bases de datos de la empresa en el día a día.

La investigación fue parte fundamental del proceso para conocer qué métodos y medidas se pueden utilizar para el desarrollo de un proceso con la capacidad de monitorear los sistemas de bases de datos SQL Server y poder asegurar soporte proactivo, que utilice los servicios con los que ya cuentan las licencias de SQL Server como lo son SQL Server Integration Services (SSIS) y SQL Server Reporting Services (SSRS).

Palabras Claves: investigación, proponer, ahorrar, monitorear, costo, bases de datos, SQL Server

Capítulo 1. Introducción

1.1. Generalidades

Se propuso un sistema de monitoreo de bases de datos, el cual busca solventar un problema y con esto crear estándares y protocolos que puedan utilizarse como guía para los diferentes ambientes de bases de datos en la organización, sin embargo, para efectos de la investigación solo se mostrarán datos de prueba ya que actualmente no se cuenta con ambientes transaccionales. De manera que, se crearán ambientes de bases de datos SQL Server en los cuales se realizan las pruebas necesarias para el monitoreo de bases de datos, revisión de métricas y la extracción de datos.

Se debe tomar en cuenta que todas las bases de datos y servidores de bases de datos pueden tener un comportamiento diferente ya que esto puede variar en muchas características, como lo son el hardware, transacciones, uso, tamaño de las bases de datos, aplicación de mejores prácticas, entre otros.

Además, cabe destacar que se utilizó una licencia de SQL Server 2019 edición Enterprise, está licencia se obtuvo por medio de una subscripción de Visual Studio. Esta edición incluye todas la características y funcionalidades principales de SQL Server, por lo cual es funcional para la revisión de la extracción de datos y monitoreo de procesos como lo son los índices, bloqueos, "*deadlocks"*, estadísticas, seguridad, disponibilidad de respaldos, entre otros.

1.2. Antecedentes del Problema

Por experiencia en diferentes empresas se han examinado varias opciones para solventar el monitoreo de bases de datos en Microsoft SQL Server con herramientas como RedGate, Quest, SentryOne, Solarwinds, Idera, entre otros. Sin embargo, su alto costo ha implicado en la decisión de no implementar dichas herramientas, lo cual sugiere crear un estándar y proceso para satisfacer la necesidad, que no pueden costear las empresas.

1.3. Definición y Descripción del Problema

En el mercado existen varios sistemas de monitoreo de bases de datos, los cuales tienen un alto costo, por lo cual la empresa tomó la decisión de no invertir en algunas de estas herramientas.

Actualmente, muchas empresas cuentan con uno o varios administradores de bases de datos, los cuales deben de monitorear de forma manual e individual el estado de los servidores y los sistemas gestores de bases de datos de Microsoft SQL Server, sin embargo, a pesar de contar con profesionales en el área de bases de datos, esto se vuelve soporte reactivo ya que solamente pueden tomar acciones cuando el negocio informa sobre algún problema en sus sistemas, pues es imposible que, de forma manual, se sepa, constantemente, el estado actual de los servidores, lo cual puede causar una caída del sistema de bases de datos que inhabilite los programas utilizados por la empresa y producir pérdidas económicas.

Muchas veces solo se cuentan con alertas que son configuradas en los "*jobs"* de SQL Server que notifican a los administradores y a los usuarios en caso de fallo. Sin embargo, esto solamente se relaciona con el fallo del "*job",* lo cual sería específicamente en cierta tarea implementada por el "*job"*, en este caso, no existen monitoreos o reportes en caso de espacios de disco, consultas con alto costo en rendimiento, índices, inventario de base de datos y muchas más que son mencionadas más adelante en la investigación.

1.4. Justificación

Este trabajo evita que las empresas deban invertir en comprar un sistema de monitoreo, ya que será un sistema hecho en casa y basado en SQL Server y Windows. Además, mejora la eficiencia y eficacia en el servicio del equipo de administradores de bases de datos en su trabajo diario. Además, el monitoreo de bases de datos provee la posibilidad de realizar soporte proactivo, en donde se pueden tomar acciones antes de que los sistemas fallen en su totalidad.

Es importante destacar que se desarrolló una serie de reportes, los cuales buscan mantener actualizados a los administradores de bases de datos y cualquier interesado sobre el estado de estas y los servidores.

No es solo un sistema, si no, es todo un proceso que puede ser implementado para la administración de bases de datos, de esta forma crea estándares y protocolos que se pueden seguir como buenas prácticas para futuros ambientes y así estandarizar todos los procesos relacionados con las bases de datos.

Esta propuesta beneficia a las empresas ya que no implica gastos innecesarios en herramientas de terceros, además, al ser un proceso creado en casa se puede adaptar a la necesidad del negocio y la empresa, por otro lado, la empresa tampoco debe gastar en nuevas licencias ya que la misma y tiene licencias implementadas de Microsoft SQL Server y Microsoft Windows Server.

1.5. Viabilidad

La viabilidad del proyecto es realizable, ya que se desarrolló y probó en un ambiente controlado, tomando en cuenta que es una investigación totalmente accesible y realizable la cual puede beneficiar de manera positiva a los diferentes negocios.

1.5.1. Punto de Vista Técnico

El proyecto involucró conocimiento en Microsoft SQL Server y sus servicios adicionales como los son “Integration Services” y “Reporting Services”, además de programación en T-SQL y PowerShell, ya que la meta final fue diseñar un almacén de datos que funcione como inventario, el cual es capaz de almacenar la información de todas las instancias de SQL Server soportados por los equipos de administradores de bases de datos, también el proceso de extracción, transformación y carga (ETL por sus siglas en inglés). Para este desarrollo se contó con más de cinco años de experiencia como DBA en SQL Server en el desarrollo y soporte con las herramientas mencionadas anteriormente, por lo cual se cuentan con las habilidades necesarias para desarrollar el proceso.

1.5.2. Punto de Vista Operativo

El proyecto afectó el tiempo del investigador para el desarrollo y pruebas del proyecto.

1.5.3. Punto de Vista Económico

Desde el punto económico, es posible, las licencias para los ambientes de pruebas y desarrollo se obtuvieron desde una subscripción de Visual Studio. Tanto como SQL Server y Windows.

Tomando en cuenta que se utilizó una de las versiones más costosas como lo es SQL Server Edición Empresarial, desde un punto de vista económico esto sería el costo de las licencias de SQL Server, tomando en cuenta que para efectos del proyecto se debe de utilizar una edición Estándar o Empresarial, el costo se muestra en dólares en la figura 1.

Precios de SQL Server 2019

Ediciones	Precio de Open No Level (USD)	Modelo de licencia	Disponibilidad de canal
Enterprise	$13, 748[1]	Lote de 2 núcleos	Licencias por volumen, hosting
Standard: por núcleo	$3,586[1]	Lote de 2 núcleos	Licencias por volumen, hosting
Standard: servidor	$899[1]	Servidor[2]	Licencias por volumen, hosting
Standard: CAL	$209	CAL	Licencias por volumen, hosting
Developer	Gratis	Por usuario	Descarga gratuita
Web	Consulta los precios con tu partner de hosting	No aplicable	Solo hosting
Express	Gratis	No aplicable	Descarga gratuita

Figura 1: Precios SQL Server 2019. Recuperado de https://www.microsoft.com/es-es/sql-server/sql-server-2019-pricing

Para el desarrollo del proyecto se necesitó de un profesional con experiencia en administración de bases de datos SQL Server y desarrollo en inteligencia de negocio con habilidades en SSIS y SSRS, el salario de una persona es de alrededor de $3 800 mensuales, lo cual implica un costo de $126 y un costo de $15 dólares por hora.

El desarrollo tiene una duración de alrededor de 16 semanas, en los cuales se trabaja de forma diaria 2 horas por día, lo cual tiene un costo de $30 por día. El costo final sería de $10 800 por 112 horas laboradas.

También se da la necesidad de un servidor, ya sea en Linux o Windows, para este caso se utilizó una versión de Windows Server 2016 Edición Standard que tiene un costo de $972.

Al final el costo total se muestra en la tabla 1.

Descripción	Costo Por Hora	Horas Laboradas	Costo Individual
DBA/ Desarrollador de BI	$30,00	112	$3 360,00
Windows Server 2016 Standard Edition (CAL de Windows Server)	N/A	N/A	$972,00
SQL Server 2019 Enterprise Edition (Licencia por Servidor)	N/A	N/A	$899,00
		Total	$5 231,00

Tabla 1: Tablas de costos. Fuente: Elaboración propia.

1.6. Objetivos

Se seleccionó la taxonomía de Bloom de 1956, ya que es la taxonomía con mayor aceptación a nivel mundial para el desarrollo de objetivos de un trabajo de investigación.

1.6.1. Objetivo General

Proponer un proceso de monitoreo en ambientes de bases de datos SQL Server para las versiones entre 2012 a 2019.

1.6.2. Objetivos Específicos

- Definir las necesidades de los administradores de bases de datos para monitorear instancias y base de datos en Microsoft SQL Server.
- Comprender los datos que son necesarios para extraer desde cada instancia de Microsoft SQL Server.
- Elaborar el diseño del almacén de datos en donde se agregará la información.
- Desarrollar el proceso de extracción, transformación y carga de datos (ETL).
- Examinar el funcionamiento del proceso de extracción, transformación y carga de datos.

1.7. Alcances y Limitaciones

A continuación, se muestra la lista de alcances y limitaciones relacionadas con esta investigación.

1.7.1. Alcances

- Documento del diseño y desarrollo del proceso.
- Creación de bases de datos (Inventario) y procesos de ETL por parte de los administradores de bases de datos.
- Reportes finales.

1.7.2. Limitaciones

- Todo documento o archivo en relación con la investigación será entregado con datos de prueba.
- El proceso solo puede ser utilizado para sistemas SQL Server y en sistemas operativos Windows, ya que las consultas están diseñadas para solo los sistemas mencionados anteriormente.
- El proceso no puede ser implementado en ambientes que cuenten con las ediciones de SQL Server Express y Developer, esto se da porque las ediciones Express no incluyen todas las características necesarias como SSIS o SSRS para el buen funcionamiento del proyecto. En el caso de SQL Server Developer es una edición gratuita, sin embargo, su licencia está hecha para ser utilizada únicamente en ambientes de desarrollo. A pesar de esto, las ediciones mencionadas anteriormente sí pueden ser monitoreadas.
- El proceso puede ser implementado en versiones inferiores a SQL Server 2012 o superiores a SQL Server 2019, no obstante, el proyecto fue desarrollado y probado para las versiones entre 2012 a 2019, por lo cual, el funcionamiento en diferentes versiones a las especificadas puede que no funcione correctamente.

1.8. Estado de la Cuestión

1.8.1. Planificación de la Revisión

1.8.1.1. Formulación de la pregunta

Determinar los puntos clave más importantes que actualmente se utilizan en las buenas prácticas para rendimiento, mantenimiento y monitoreo de bases de datos SQL Server.

1.8.1.2. Amplitud y calidad de la pregunta

- *Definición del problema:* En los últimos años los datos se han convertido en uno de los activos más importantes para las compañías, por lo que es de vital importancia mantener las bases de datos seguras y disponibles, de manera que un ambiente sin monitoreo de estas puede complicar su mantenimiento, así como su monitoreo para los administradores de bases de datos, complicando sus tareas y trabajo diario.
- *Pregunta de investigación:* ¿Cuáles son las mejores prácticas y puntos por tomar en consideración para crear un proceso de monitoreo de bases de datos en ambientes SQL Server?
- *Palabras claves y Sinónimos*: A continuación, se presenta una tabla resumen con las palabras clave y conceptos relacionados los cuales se emplearán en esta revisión, este empleo de palabras se realizará en inglés y español.

Palabra Clave	Sinónimo	Traducción al inglés
Bases de Datos		Databases
Administración		Administration
SQL Server	MSSQL	
Monitoreo		Monitoring
Gestión		Management
Soporte		Support
Mantenimiento		Maintenance
Rendimiento		Performance

Tabla 2. Palabras clave. Fuente: Elaboración Propia

- *Intervención:* En el contexto de la revisión sistemática planificada, se observaron todas las posibles propuestas y enfoques existentes relacionados con el monitoreo, mantenimiento y rendimiento de bases de datos, de esta forma se extraen las más importantes y se procede a un posterior análisis de estas.
- *Control:* En la presente revisión sistemática se han observado algunos trabajos para poder definir las palabras clave y la idea general de la investigación, sin embargo, ninguno que se clasifique como un conjunto de resultados derivado de los criterios definidos y que cumplen con el objetivo buscado.
- *Resultado:* Los resultados esperados de esta revisión son detectar y conocer propuestas, recomendaciones o buenas prácticas relacionadas con el monitoreo, mantenimiento y rendimiento de los sistemas de bases de datos, para posteriormente analizarlos y encontrar aquellas que mayor probabilidad de acierto presenten.
- *Medida de Salida:* Para medir los resultados obtenidos se realizó una agrupación de las propuestas encontradas con base en el área en el que se centran.
- *Población:* La población se compone de las publicaciones presentes en los repositorios de las fuentes de datos seleccionados los cuales están relacionadas con el objetivo de esta revisión.
- *Aplicación:* Los beneficiarios de la revisión sistemática fueron las personas que están relacionadas directamente con el área de administración de bases de datos, en especial con el sistema gestor de bases de datos Microsoft SQL Server, así como todas aquellas personas interesadas en los trabajos relevantes al mantenimiento, monitoreo y rendimiento de bases de datos.
- *Diseño Experimental:* No se aplicó un metaanálisis de la revisión sistemática.

1.8.2. Selección de Fuentes

1.8.2.1. Definición del criterio de selección de fuentes

El criterio para la selección de las fuentes de búsqueda está basado en la opinión del investigador de este trabajo, el cual, basándose en su experiencia

profesional como administrador de bases de datos en ambientes SQL Server y desarrollador de inteligencia del negocio, sugirió una lista de fuentes sobre las cuales realizar la revisión. Otros requisitos exigidos a las fuentes para su selección son accesibilidad vía web y la inclusión de motores de búsqueda que permitan la realización de consultas avanzadas.

1.8.2.2. Lenguajes de los estudios

Para poder satisfacer el proyecto y obtener mayor plenitud de resultados a nivel literario se utilizaron palabras tanto en el idioma inglés como en el idioma español.

1.8.2.3. Identificación de fuentes

Se identificaron varias fuentes utilizando las palabras clave mencionadas anteriormente.

- *Métodos de búsqueda en las fuentes:* Las fuentes seleccionadas son provenientes de repositorios confiables y también el investigador se va a asegurar que estos cumplan con los parámetros de calidad académica necesarios.
- *Listado de fuentes:* Se seleccionarán los siguientes repositorios sobre la cual se ejecutará la revisión sistemática:
    - ACM Digital Library.
    - SpringerLink.
    - Google Scholar.
    - IEEE Xplore Digital Library.
    - IEEE Computer Society.
    - Research Gate.

Cadenas de búsqueda: Combinaciones de los operadores lógicos “Y” y “O” sobre las palabras clave, así como conceptos sinónimos citados anteriormente en la sección “Palabras clave y sinónimos”. De esta manera, se estableció la siguiente cadena de búsqueda por utilizar en la presente revisión:

SQL Server OR ETL AND (monitoring OR administration OR management OR support OR maintenance OR performance)

1.8.2.4. Selección de fuentes después de la evaluación

Las fuentes seleccionadas cumplieron y satisficieron los criterios de selección, pero se realizó una fase de refinado con el objetivo de agregar artículos que pueden ser provechosos los cuales no encuentran en las fuentes seleccionadas.

1.8.2.5. Comprobación de las fuentes

Todas las fuentes se aprobaron al cumplir con el método de selección de fuentes estipulado.

1.8.3. Selección de Estudios

1.8.3.1. Procedimiento para la selección de estudios

Se utilizó un procedimiento basado en la definición y manejo de las palabras clave de búsqueda comentadas anteriormente, a continuación, se muestra un diagrama de flujo con las etapas por seguir:

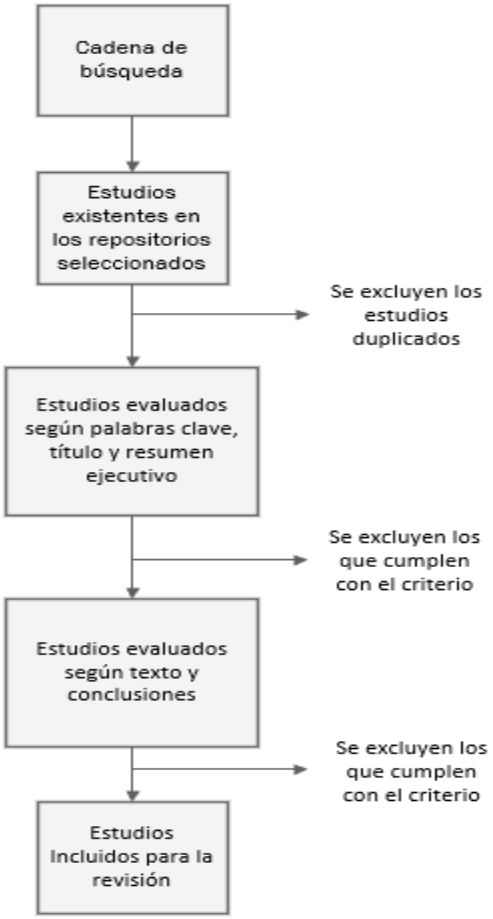


Figura 2. Procedimiento de selección de estudios. Fuente: Elaboración propia.

1.8.3.2. Definición de criterios de inclusión y exclusión de estudios

En el criterio de inclusión, por cada artículo se examinó el título, las palabras clave y el resumen ejecutivo de cada documento, de esta forma se validó que el artículo esté relacionado con la funcionalidad necesaria para el monitoreo y buen manejo de bases de datos.

En el criterio de exclusión, se revisa la información del contenido de los artículos y sus conclusiones, con el fin de conocer en detalle cada documento y saber si este está relacionado con los objetivos y comprobar que sea relevante para la revisión.

1.8.3.3. Definición de tipos de estudio

Los tipos de estudios primarios que fueron utilizados son los adquiridos en las fuentes definidas previamente en la sección 1.9.2.3, lo cuales cumplen con el

criterio definido de selección de fuentes que hayan pasado por el procedimiento definido en la sección 1.9.3.1.

1.8.4. Ejecución de la Selección

1.8.4.1. Definición del criterio de inclusión y exclusión de información

La información que se obtuvo del estudio debe incluir conceptos, técnicas, metodologías, procedimientos que detallen procesos, funciones o aspectos importantes relacionados con el monitoreo, rendimiento y mantenimiento de bases de datos.

Además, se sustrajo la información de los estudios seleccionados anteriormente que tienen relación con la investigación y el foco de estudio se toma en cuenta lo siguiente:

- El título relacionado con palabras claves.
- Resumen relacionado con el tema.
- El contenido con los objetivos.

1.8.4.2. Formulario para la extracción de información

Se usó un formulario para documentar la extracción de la información sobre cada documento encontrado el cual incluye lo que se muestra a continuación en la figura 2.

Identificación	Título
	Autores
	Año
	Enlace
Contenido	Elementos Tratados
Aspectos por destacar	
Extracción de resultados objetivos	Metología del estudio
	Resultados del estudio
	Problemas del estudio
Extracción de resultados subjetivos	Información a través de los autores
	Impresiones generales

Tabla 3. Formulario de Extracción de Información. Fuente: Elaboración propia.

1.8.4.3. Selección de estudios iniciales

A continuación, se muestra el resultado de la búsqueda que se realizó en la lista de repositorios mencionadas en el punto 1.9.2.3.

1.8.5. Extracción de Información

Esta sección tuvo como objetivo registrar el proceso de selección de los estudios primarios, informando los estudios obtenidos y el resultado de su evaluación. Todo se puede apreciar desde la Tabla 4 hasta la Tabla 14.

Identificación	
Título	Conceptual modeling for ETL processes
Autores	Panos Vassiliadis, Alkis Simitsis, Spiros Skiadopoulos
Año	2002
Link	https://dl.acm.org/doi/10.1145/583890.583893
Contenido	
Elementos tratados:	● Utilización de los ETL.
Aspectos por destacar	
El estudio se orientó en el uso y en el modelado al usar los procesos de extracción, transformación y carga de datos.	
Extracción de resultados objetivos	
Metodología del estudio	Los autores proveyeron conceptos relacionados al uso del ETL y sus componentes.
Resultados del estudio	Proveyó técnicas en ¿cómo se pueden trabajar con un modelo de ETL?
Problemas del estudio	No se encuentra ninguno.
Extracción de resultados subjetivos	
Información a través de los autores	No fue solicitado.

Impresiones generales	La investigación ayudó en la creación del ETL que es necesario para adquirir los datos relacionados con el monitor de bases de datos.

Tabla 4. Primera fuente literaria. Fuente: Elaboración Propia

Identificación	
Título	Training Kit - Exam 70-462: Administering Microsoft SQL Server 2012 Databases
Autores	Orin Thomas, Peter Ward, Bob Taylor
Año	2012
Link	https://dl.acm.org/doi/book/10.5555/2408067
Contenido	
Elementos tratados:	● Manejo de seguridad a nivel de bases de datos SQL Server. ● Auditorías y solución de problemas. ● Instalación y configuración de ambientes SQL Server.
Aspectos por destacar	
Extracción de resultados objetivos	
Metodología del estudio	Microsoft proveyó este libro como un guía para utilizar Microsoft SQL Server.
Resultados del estudio	Guía para instalar, mantener y soportar una instancia de SQL Server y las posibles variaciones que esta puede tener.
Problemas del estudio	No se encuentra ninguno.
Extracción de resultados subjetivos	
Información a través de los autores	No fue solicitado.
Impresiones generales	Este libro otorgó información útil que puede ser tomada en cuenta a la hora de escoger cuáles puntos pueden ser tomados en cuenta para el monitoreo de bases de datos.

Tabla 5. Segunda fuente literaria. Fuente: Elaboración Propia

Identificación	
Título	Pro SQL Server 2012 Administration
Autores	Ken Simmons, Sylvester Carstarphen
Año	2012
Link	https://link.springer.com/book/10.1007/978-1-4302-3916-1#toc
Contenido	
Elementos tratados:	● Utilización de SQL Server 2012. ● Rendimiento de las bases datos. ● Rendimientos para índices. ● Automatización de tareas de mantenimiento. ● Monitoreo de los servidores.
Aspectos por destacar	
Se destaca por la orientación a la administración de bases de datos Microsoft SQL Server, en donde se encuentran técnicas para la automatización de tareas y monitoreo de servidores SQL Server. Además, cubre una gran sección para el monitoreo de bases de datos.	
Extracción de resultados objetivos	
Metodología del estudio	Los autores se enfocaron en puntos más avanzados relacionados a las tareas que debe cumplir un administrador de bases de datos.
Resultados del estudio	Otorgaron una lista de las mejores prácticas, bastante amplia, en las cuales se tomaron en cuenta desde el hardware por utilizar para una instancia de SQL Server hasta conceptos básicos y avanzados de los objetos y arquitectura que utiliza SQL Server.
Problemas del estudio	No se encuentra ninguno.
Extracción de resultados subjetivos	
Información a través de los autores	No fue solicitado.

Impresiones generales	El libro es muy completo desde el punto de vista de la administración de bases de datos SQL Server, pero este en especial tiene un punto muy importante que involucra solamente el tema relacionado al monitoreo de bases de datos.

Tabla 6. Tercera fuente literaria. Fuente: Elaboración Propia

Identificación	
Título	Healthy SQL
Autores	Robert Pearl
Año	2015
Link	https://link.springer.com/book/10.1007/978-1-4302-6772-0#toc
Contenido	
Elementos tratados:	● Tipos de espera y consultas. ● Monitoreo y reportería.
Aspectos por destacar	
Este libro se centró completamente en cómo mantener un servidor de SQL Server saludable y optimizado para así mejorar el rendimiento, hay dos puntos importantes por destacar el cual habla acerca de cómo monitorear y crear reportes a los servidores y los tipos de espera que tiene SQL Server lo cual se relaciona con los tipos de bloqueos que pueden existir en un servidor.	
Extracción de resultados objetivos	
Metodología del estudio	El autor hizo énfasis en las formas efectivas para mantener un ambiente de bases de datos SQL Server saludable y Optimizado, ya que este enfocó la lectura en tiempos de espera, recuperación de desastres y alta disponibilidad, evitando abordar temas de configuración e instalación.
Resultados del estudio	Un listado de técnicas y métodos para el mantenimiento de ambientes SQL Server.
Problemas del estudio	No se encuentra ninguno.
Extracción de resultados subjetivos	

Información a través de los autores	No fue solicitado.
Impresiones generales	Fue muy útil cuando hablamos de monitoreo, ya que menciona y detalla los tiempos de espera que utiliza SQL Server, además de una serie de consultas que pueden ser útiles para el consumo de datos.

Tabla 7. Cuarta fuente literaria. Fuente: Elaboración Propia

Identificación	
Título	Professional Microsoft SQL Server 2012 Administration
Autores	Adam Jorgensen, Steven Wort, Ross LoForte, Brian Knight
Año	2012
Link	https://books.google.co.cr/books?hl=es&lr=&id=xTOGOjBFma4C&oi=fnd&pg=PR37&dq=sql+server+administration&ots=daPD1mtlyB&sig=jMabUYoiOTI7thKzvxr1G0x28BI&redir_esc=y#v=onepage&q=sql%20server%20administration&f=false
Contenido	
Elementos tratados:	● Monitoreo de instancias de SQL Server. ● Arquitectura de SQL Server. ● Mejores prácticas.
Aspectos por destacar	
Este libro se centró en la administración de bases de datos SQL Server y Azure SQL, además, involucra y desarrolla capítulos en los que se incluyen SSRS y SSIS.	
Extracción de resultados objetivos	
Metodología del estudio	Los autores explicaron de forma muy completa SQL Server y sus servicios, además de mencionar en gran detalle varias técnicas de alta disponibilidad y recuperación de desastres.
Resultados del estudio	Proveyeron una guía muy detallada de cómo funcionan partes esenciales de SQL Server.
Problemas del estudio	No se encuentra ninguno.

Extracción de resultados subjetivos	
Información a través de los autores	No fue solicitado.
Impresiones generales	El libro hizo mucho énfasis en secciones importantes sobre el monitoreo de bases de datos, además, es importante destacar que incluye información relacionada a los servicios de SSIS y SSRS, además la manipulación de archivos “logs”.

Tabla 8. Quinta fuente literaria. Fuente: Elaboración Propia

Identificación	
Título	Analyzing Integral Components of SQL Server Databases
Autores	Muthukumar Murugesan, K.Karthikeyan, K.Sivakumar
Año	2015
Link	https://www.researchgate.net/profile/Muthukumar_Murugesan/publication/285429493_Analyzing_integral_components_of_SQL_server_databases/links/5c3919ec299bf12be3c141de/Analyzing-integral-components-of-SQL-server-databases.pdf
Contenido	
Elementos tratados:	● Archivos de datos y logs de SQL Server. ● Componentes internos del sistema gestor de bases de datos SQL Server.
Aspectos por destacar	
Destacó la explicación y definición de las diferentes partes críticas que conforman SQL Server.	
Extracción de resultados objetivos	
Metodología del estudio	Los autores realizaron un análisis importante acerca de los componentes clave que forman parte de los ambientes de SQL Server
Resultados del estudio	Definición e identificación sobre los componentes más importantes que forman parte de SQL Server.

Problemas del estudio	No se encuentra ninguno.
Extracción de resultados subjetivos	
Información a través de los autores	No fue solicitado.
Impresiones generales	Es importante resaltar que la estructura física y lógica de SQL Server es importante a la hora de tomar en cuenta en el monitoreo, ya que en este puede haber puntos clave los cuales deben de ser monitoreados.

Tabla 9. Sexta fuente literaria. Fuente: Elaboración Propia

Identificación	
Título	SQLCM: a continuous monitoring framework for relational database engines
Autores	Surajit Chaudhuri, A.C. Konig, Vivek Narasayya
Año	2004
Link	https://ieeexplore.ieee.org/abstract/document/1320020
Contenido	
Elementos tratados:	● Monitoreo de bases de datos SQL Server. ● Metodología de monitoreo.
Aspectos por destacar	
Se destacó el enfoque hacia SQL Server	
Extracción de resultados objetivos	
Metodología del estudio	Los autores otorgaron ejemplos donde la metodología de SQLCM puede ser implementada, además de una breve explicación de sus aspectos y componentes.
Resultados del estudio	Una serie de procesos que pueden ser implementados para el monitoreo de bases de datos de forma eficiente.
Problemas del estudio	No se encuentra ninguno.
Extracción de resultados subjetivos	

Información a través de los autores	No fue solicitado.
Impresiones generales	Una metodología que puede ser utilizada en los puntos necesarios para el desarrollo de la aplicación.

Tabla 10. Séptima fuente literaria. Fuente: Elaboración Propia

Identificación	
Título	The main causes of failures SQL Server
Autores	Vasyl Romanchuk, Taras Andrukhiv, Ihor Kahalo
Año	2012
Link	https://ieeexplore.ieee.org/document/6192631
Contenido	
Elementos tratados:	● Causa de los fallos en SQL Server.
Aspectos por destacar	
El artículo comentó sobre las posibles causas que puede tener SQL Server en diferentes componentes.	
Extracción de resultados objetivos	
Metodología del estudio	Los autores explicaron cuáles son los problemas más comunes que se pueden encontrar dentro del sistema gestor de bases de datos SQL Server.
Resultados del estudio	Una lista de las posibles causas que provocan los fallos en SQL Server, además, de una serie de sugerencias para prevenir estos problemas.
Problemas del estudio	No se encuentra ninguno.
Extracción de resultados subjetivos	
Información a través de los autores	No fue solicitado.

Impresiones generales	El artículo mencionó partes importantes que deben de ser tomadas en cuenta para realizar el monitoreo en SQL Server.

Tabla 11. Octava fuente literaria. Fuente: Elaboración Propia

Identificación	
Título	Autonomic Computing in SQL Server
Autores	Abdul Mateen, Basit Raza, Tauqeer Hussain
Año	2008
Link	https://www.computer.org/csdl/proceedings-article/icis/2008/3131a113/12OmNxeM49Y
Contenido	
Elementos tratados:	● Automatización en SQL Server.
Aspectos por destacar	
La explicación y definición de los componentes automatizados de SQL Server, así como la serie de herramientas para el monitoreo de bases de datos SQL Server.	
Extracción de resultados objetivos	
Metodología del estudio	Explicaron una serie de componentes que tiene SQL Server, además mencionan la intervención que debe hacer el humano para la gestión de SQL Server y proveen información relacionada a las herramientas que tiene SQL Server para el monitoreo.
Resultados del estudio	Como resultado se obtuvo un listado de componentes que pueden ser automatizados.
Problemas del estudio	No se encuentra ninguno.
Extracción de resultados subjetivos	
Información a través de los autores	No fue solicitado.

Impresiones generales	El texto explicó una seria de componentes que pueden se provechosos para la creación de una propuesta para el monitoreo de bases de datos SQL Server.

Tabla 12. Novena fuente literaria. Fuente: Elaboración Propia

1.8.6. Análisis de resultados

1.8.6.1. Presentación de resultados

Se encontraron varias opciones, en algunos casos se toca el tema ya sea de monitoreo o administración de bases de datos en SQL Server o sobre SSIS y SSRS, sin embargo, muchos de estos no han tenido el enfoque deseado, no obstante, se han encontrado una variedad de artículos y libros en las fuentes indicadas anteriormente los cuales se relacionan correctamente con la cadena de búsqueda y puede ayudar al desarrollo de la investigación. Los artículos se resumieron de la siguiente forma según la fuente donde fueron encontrados:

Fuente	Estudios	Excluidos	Relevantes
ACM Digital Library	6	4	2
SpringerLink	4	2	2
Google Scholar	2	0	2
IEEE Xplore Digital Library	2	0	2
IEEE Computer Society	1	0	1

Tabla 13. Resultados de artículos según fuente. Fuente: Elaboración Propia

Además, se resumen de la siguiente forma según el tema que cubre el artículo:

Monitoreo	Administración	Rendimiento	ETL	Encontrados	Seleccionados
3	3	2	1	15	9

Tabla 14. Resultados de artículos según tema. Fuente: Elaboración Propia

1.8.6.2. Análisis de sensibilidad

No fue aplicado.

1.8.6.3. Gráficos

No fueron aplicados.

1.8.6.4. Comentarios finales

Número de estudios: 14 estudios encontrados, 9 seleccionados.

Sesgo de búsqueda, selección y extracción: Todos los estudios que fueron publicados antes del año 2000 han sido descartados.

Sesgo de publicación: No fue definido.

Variación entre revisores: No hay variación.

Aplicación de resultados: Los estudios y libros seleccionados abastecieron información fundamental para el uso y desarrollo de la investigación ya que estos se complementan muy bien.

Recomendaciones: Ninguna.

Capítulo 2. Marco Conceptual

2.1. Elaboración del Marco Conceptual

El marco conceptual se creó utilizando la herramienta de nube de palabras, como lo muestra la Figura 2, la nube de palabras fue creada utilizando las más relevantes de los estudios seleccionados en el estado de la cuestión, para esto se utilizó las palabras en inglés, también es importante destacar que algunos conceptos utilizan sus siglas.

Figura 3. Nube de Palabras. Fuente: Elaboración propia.

Una vez generada la nube de palabras se utilizó un mapa conceptual jerárquico comenzando desde los conceptos más específicos hasta los más genéricos y básicos, estos establecieron el orden en el cual el marco conceptual se redacta y cómo se unen los conceptos. La figura 3 muestra el mapa conceptual jerárquico vertical con los conceptos más relevantes.

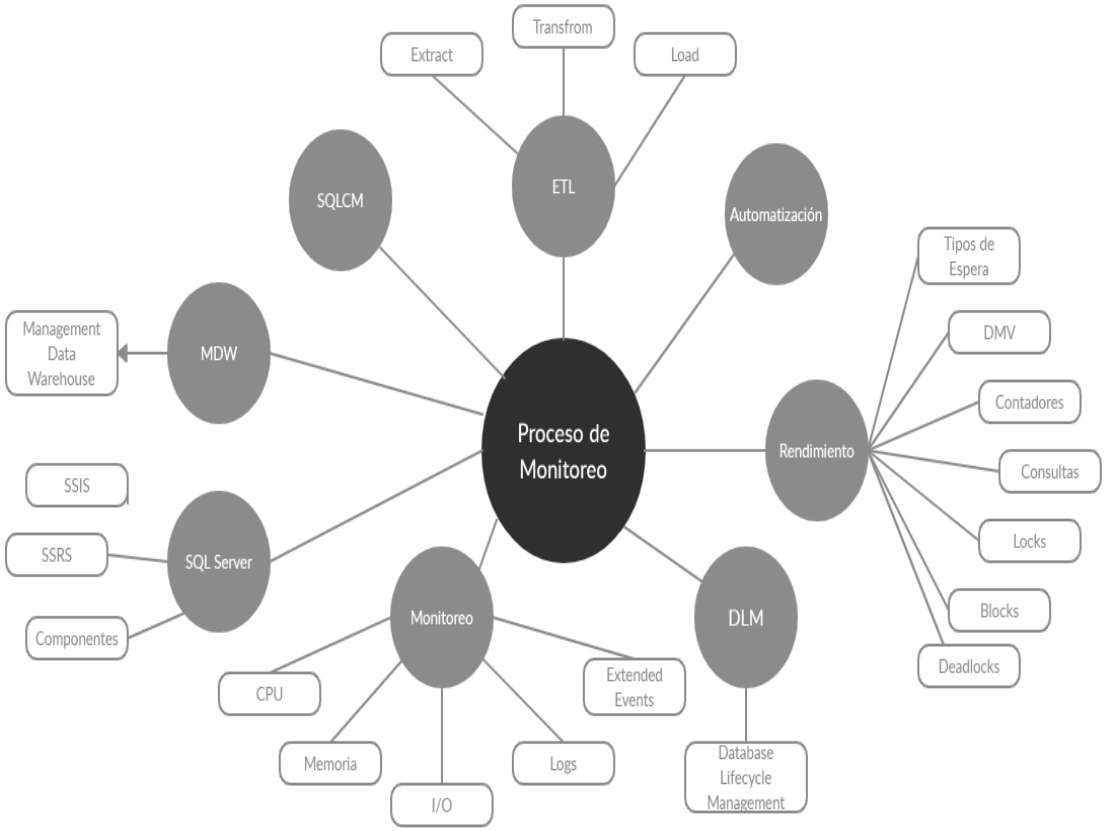


Figura 4. Mapa Conceptual Jerárquico. Fuente: Elaboración propia.

2.2. SQL Server

Según el criterio de Vasyl Romanchuk, Taras Andrukhiv e Ihor Kahalo (2012) SQL Server es un producto de Microsoft el cual es utilizado para almacenar y recuperar datos según sea solicitado de las diferentes aplicaciones de software la cuales se relacionan con bases de datos relacionales.

2.2.1. Arquitectura de Archivos de SQL Server

SQL Server tiene componentes integrales que son vitales para su funcionamiento los cuales se dividen en físicos y lógicos, los cuales son mencionados y definidos por Muthukumar Murugesan, K. Karthikeyan y K. Sivakumar (2015, P.2) de la siguiente forma:

- "**Archivo de datos principal**; Es el punto de partida de la base de datos y tiene la referencia a los otros archivos de la base de datos. Los datos de usuario y los objetos se pueden almacenar en este archivo o en

archivos de datos secundarios. Cada base de datos debe tener un solo archivo de datos principal. Este archivo utiliza la extensión .mdf."

- "**Archivo de datos secundarios**; componen todos los archivos de datos, excepto el archivo de datos principal. Esto es opcional y algunas bases de datos pueden no tener archivos de datos secundarios, mientras que otras tienen varios archivos de datos secundarios. Este archivo utiliza la extensión .ndf."
- "**Archivos de registro**; cada base de datos debe tener al menos un archivo de registro. Los archivos de registros contienen la información del registro y que se utiliza para recuperar la base de datos. Este archivo utiliza la extensión .ldf."

Además, la arquitectura de SQL Server divide los archivos con dos nombres, un nombre para los archivos lógicos y uno para los archivos físicos, los cuales Microsoft Dosc (2020), los define de la siguiente forma:

- **Nombre Archivos Físicos**: El "logical_file_name" es el nombre que se utiliza para relacionar el archivo físico con las diferentes instrucciones TraTnsact-SQL (también conocidas como Transact-SQL).
- **Nombre Archivos Lógicos**: El "os_file_name" es el nombre físico, el cual se puede encontrar en la ruta del directorio del sistema operativo.

También se debe destacar la existencia de las cuatro bases de datos de sistema que son creadas por cada instancia SQL Server que se encuentra en un servidor conocidas como máster, msd, tempdb y model, las cuales se definen a continuación según la información de la página oficial de Microsoft

- **máster:** Contiene toda la información a nivel de sistema dentro de SQL Server; Microsoft (2019).
- **model:** Es utilizada como una plantilla para crear nuevas bases de datos dentro de SQL Server; Microsoft (2020).
- **msdb:** Contiene la información relacionada al SQL Agent, básicamente todo lo relacionado a la configuración del correo de

bases de datos, Jobs y la calendarización de los Jobs; Microsoft (2016).

- **tempdb:** Es un recurso global que está disponible para todos los usuarios conectados a la instancia de SQL Server, esta base de datos almacena todo lo relacionado con objetos temporales; Microsoft (2019).

2.2.1. Database Engine Service

El servicio de SQL Server Database Engine es uno de los componentes principales para almacenar, procesar y proteger los datos, el servicio de Database Engine proporciona acceso contralado y procesamiento rápido de transacciones para cumplir con los requisitos de las aplicaciones que utilizan constantemente los datos; Microsoft (Microsoft, 2019).

2.2.2. SQL Agent Service

Microsoft (2019) define el servicio de SQL Agent como “el servicio encargardo de ejecutar tareas calendarizadas, más conocidas como jobs”. Este servicio es necesario para poder configurar todas las tareas que deben de ser ejecutadas de forma diaria, por hora, por minuto o por segundo, lo cual permite el uso constante de ciertas tareas previamente programadas.

2.2.3. SQL Server Integration Services (SSIS)

SQL Server Integration Services es uno de los componentes más importantes para el uso de inteligencia de negocio que son incluidas en algunas de las ediciones de SQL Server. “Su principal responsabilidad es el movimiento y la limpieza de datos. Sin esta limpieza y movimiento, todos los demás componentes no existirían o, como mínimo, reportarían datos incorrectos.” (Jorgensen, Wort, LoForte y Knight, 2012, P. 696)

2.2.4. SQL Server Reporting Services (SSRS)

SQL Server Reporting Services es otro de los servicios que se incluyen dentro del catálogo de servicios que ofrece Microsoft SQL Server, Microsoft **lo** describe como, “un conjunto de herramientas y servicios locales que crean, implementan y administran informes móviles y paginados.”

2.2.5. PowerShell

Microsoft (2020) **(página)** define PowerShell como un marco de referencia para la configuración y automatización de tareas multiplataforma, la cual consta de Shell de línea de comando y un leguaje para la secuencia de comandos.

2.2.6. Windows Server Failover Cluster (WSFC)

SQL Server puede ser configurado de muchas maneras, una de estas configuraciones es con un clúster, el cual Microsoft (2019) define como un grupo de servidores independientes (nodos) que trabajan juntos para así habilitar y aumentar la disponibilidad y la escalabilidad. Esto funciona si un nodo falla, inmediatamente, otro nodo, que forma parte del clúster, pueda iniciar y seguir brindando el servicio.

2.2.7. Always On Availability Groups (Always On AG)

SQL Server Always On es una característica introducida a partir de las versiones de SQL Server 2012, el cual habilita la alta disponibilidad y recuperación de desastres, actualmente existen dos tipos de configuración para Always On, cabe destacar que Always On necesita una configuración previa de Windows Server Failover Cluster para que este pueda ser configurado con éxito.

“AlwaysOn es una solución nueva de alta disponibilidad y recuperación ante desastres que permite maximizar la disponibilidad para una o varias bases de datos de usuarios como grupo.” (Jorgensen, Wort, LoForte y Knight, 2012, P. 514)

2.2.8. Max Degree Of Parallelism (MAXDOP)

Microsoft (2020) define MAXDOP como el número de procesadores utilizador para ejecutar una sola transacción, para cada ejecución de forma paralela.

2.2.9. Cost Threshold for Parallelism

Microsoft (2017) define el límite del costo de paralelismo en el que SQL Server crea y ejecuta planes paralelos para consultas

2.3. Monitoreo de Bases de Datos

El monitoreo de bases de datos como explica Jorgensen, Wort, LoForte y Knight (2012, P. 318) “le permite pasar de tratar reactivamente los eventos a diagnosticar problemas de manera proactiva y solucionarlos antes de que sus

usuarios se den cuenta de que hay un problema.” A través de los años se han creado diferentes prácticas para mejorar la forma en la cual se monitorean las bases de datos y los servidores para así anticipar los problemas críticos que puedan ocurrir en sistemas productivos de base de datos causando daños en los sistemas que puedan tener consecuencias económicas dentro de las diferentes organizaciones que utilizan sistemas de bases de datos.

Actualmente, no existe un marco de referencia o estándar que mencione como debería de ser el monitoreo de bases de datos, sin embargo, existen una serie de objetivos claves que pueden ser aplicados para obtener un monitoreo eficiente y eficaz para las diferentes bases de datos y sus servidores, los cuales son mencionados por Jorgensen, Wort, LoForte y Knight y se listan, a continuación:

Establecer una línea de base.

Identificar nuevas tendencias antes de que se conviertan en problemas.

Monitorear el crecimiento de la base de datos.

Identificar tareas de mantenimiento diarias, semanales y mensuales.

Identificar los cambios de rendimiento a lo largo del tiempo.

Auditar la actividad del usuario.

Diagnosticar un problema de rendimiento específico. (2012, P. 318)

Definir un sistema de monitoreo puede dar muchas ventajas Davis (2013), menciona una lista de los 8 pasos para monitorear SQL Server de forma eficiente, los cuales se listan, a continuación:

1) Conocer la lista de servidores SQL Server.
2) Documentación básica de los servidores y las bases de datos.
3) Backups y SQL Agent Jobs.
4) Tasas de crecimiento y espacio (planificación de la capacidad).
5) Monitoreo de CPU, I/O y Memoria.
6) Monitorear errores y problemas específicos.

7) Mejorar la estrategia de monitoreo.
8) Medir los efectos del monitoreo.

2.3.1. Extended Events

Los “extended events” (eventos extendidos) son unas de las herramientas que se utilizan actualmente para el monitoreo de bases de datos Pearl (2015, P.149) los define como “un sistema de supervisión del rendimiento y el estado que se puede configurar para capturar información sobre toda la actividad del servidor. Presentaré eventos extendidos”

2.3.2. Deadlocks y Blocks

En muchas ocasiones cuando se habla de problemas o simplemente de lentitudes o procesos que no avanzan, esto se puede deber a un “lock” o un “block”. Los cuales Fritchey (2018, P. 18) explica a continuación:

- **Blocks:** Ocurre cuando varias transacciones intentan, concurrentemente, acceder a un recurso de una manera no compatible.
- **Deadlocks:** Un “deadlock” se crea cuando dos recursos intentan escalar o expandir recursos bloqueados y entran en conflicto entre sí.

2.3.3. Índices

Microsoft (2019), define los índices como una estructura en disco asociada con una tabla o vista que acelera la recuperación de los datos de filas de la tabla o vista. SQL Server tiene dos tipos de índices conocidos como agrupados (Clustered) y no agrupados (NonClustered), los cuales Microsft describe de la siguiente manera:

- **Clustered Indexes:** Clasifican y almacenan las filas de datos en la tabla o vista en función de sus valores clave, estos siempre son agregados en como llave primaria, además solo se puede crear un “clustered index”.
- **NonClustered Indexes:** Contiene los valores clave del índice no agrupado y cada entrada de valor clave tiene un puntero a la fila de datos que contiene el valor clave. Este tipo de índice puede ser creado múltiples veces a diferencia del “clustered index”.

2.3.4. Dynamic Management Views (DMV)

Los DMV's son una herramienta muy útil al momento de revisar problemas de Rendimiento, Jorgensen, Wort, LoForte y Knight (2012, P. 179) los definen como "vistas que retornan información relacionada del estado del servidor que se puede utilizar para supervisar el estado de una instancia del servidor, diagnosticar problemas y ajustar el rendimiento".

3. SQLCM

SQL Continuous Monitoring (también conocido por sus siglas en inglés como SQLCM) es de los pocos "framework" (marco de referencia) que existe para el monitoreo de bases de datos SQL Server, como lo describen Chaudhuri, König, Narasayya (2012, P.1), SQLCM facilita el desarrollo de una gran cantidad de tareas importantes basadas en el monitoreo, además aborda las limitaciones de los mecanismos existentes de SQL Server, además SQLCM se implementa completamente dentro del servidor de base de datos, y la información monitoreada se puede agrupar y agregar automáticamente con las columnas de agrupación y agregación, funciones especificadas por el administrador de la base de datos.

3.2. Repositorio MDW (Management Data Warehouse)

No muchas veces se oye mencionar el Management Data Warehouse de Microsoft, el cual se define de la siguiente forma," ayuda a configurar un repositorio para almacenar sus datos de rendimiento, lo ayuda a decidir qué desea recopilar y con qué frecuencia desea recopilar los datos, y luego le brinda algunos informes que ayudan a analizar sus datos" (Simmons, K., & Carstarphen, S., 2012, P. 380).

Esta es una opción que se puede utilizar para la extracción de información para luego realizar un debido análisis de los datos.

3.3. Extracción, Transformación y Carga de Datos

Un proceso de ETL es parte fundamental de este proyecto, ya que un ETL es la parte responsable para la extracción de datos de varias fuentes, su limpieza, personalización e inserción en un almacén de datos como mencionan Vassiliadis, Simitsis y Skiadopoulos (2002, P.1). Para la investigación y el desarrollo del proceso de monitoreo de bases de datos un proceso de ETL es necesario para la extracción

y manipulación de los datos que son relacionados con el desempeño y rendimiento de bases de datos.

3.4. Ciclo de Vida de las Bases de Datos

También se utilizará la documentación relacionada al ciclo de vida de bases de datos (DLM, por sus siglas en inglés), en donde Microsoft Docs (2013) define DLM como:

> …un enfoque basado en políticas para administrar bases de datos y activos de datos. DLM no es un producto sino un enfoque integral para administrar el esquema, los datos y los metadatos de la base de datos para una aplicación de base de datos. Un enfoque reflexivo y proactivo para DLM permite a una organización administrar los recursos de datos de acuerdo con los niveles apropiados de rendimiento, protección, disponibilidad y costo.

Este punto es clave para el desarrollo de las bases de datos que fueron creadas para almacenar los datos de las diferentes fuentes que son monitoreadas.

Figura 5. Ciclo de Vida de Base de Datos. Recuperado de https://www.idera.com/resourcecentral/infographics/idera-database-lifecycle-management

Capítulo 3. Marco Metodológico

3.1. Tipo de Investigación

Debido a que este proyecto busca solventar una necesidad y resolver un problema que afronta actualmente el equipo de administradores de bases de datos, el tipo de investigación que se utilizó es la aplicada, de esta forma poder proponer un proyecto con los conocimientos existentes y la experiencia de los administradores de bases de datos.

Esta investigación utilizó conocimientos ya existentes para solucionar un problema específico. El mismo se centró en la resolución de problemas en un ambiente determinado, el cual busca la aplicación y/o utilización de conocimientos con el propósito de implementarlos de forma práctica para satisfacer la falta de un sistema de monitoreo para base de datos de SQL Server.

3.2. Alcance Investigativo

El alcance investigativo escogido fue el descriptivo ya que se busca recolectar información relacionada con las buenas prácticas acerca del monitoreo de bases de datos en SQL Server, además de información relacionada a la extracción de datos, los cuales fueron necesarios para obtener alertas y reportes que forman parte del monitoreo.

Pretenden medir o recoger información de manera independiente o conjunta sobre los conceptos o las variables a las que se refieren, esto es, su objetivo no es indicar cómo se relacionan estas (Hernández, Fernández y Baptista, 2014, P.92).

3.3. Enfoque

El enfoque optado para la implementación de un proceso de monitoreo en ambientes de bases de datos SQL Server es cualitativo.

Las investigaciones cualitativas no se planean con detalle y pueden adaptarse al cambio según sea necesario en el desarrollo de la investigación, al intentar crear un nuevo proceso el cual no ha sido implementado anteriormente en diferentes equipos de administración de bases de datos es probable que los cambios sean constantes y sin previo aviso según sean necesarios.

La investigación cualitativa se enfoca en comprender los fenómenos, explorándolos desde la perspectiva de los participantes en un ambiente natural y en relación con su contexto (Hernández, Fernández y Baptista, 2014, P.358).

3.4. Diseño

Por la naturaleza del proyecto el tipo de diseño que se utilizó fue la Investigación-Acción.

La finalidad de la investigación-acción es comprender y resolver problemáticas específicas de una colectividad vinculadas a un ambiente. Asimismo, se centra en aportar información que guíe la toma de decisiones para proyectos, procesos y reformas estructurales. (Hernández, Fernández y Baptista, 2014, P.496).

Al final se pretende tener una propuesta de un sistema de monitoreo de bases de datos SQL Server funcional para la utilización diaria de los administradores de bases de datos.

3.5. Población y Muestreo

Al ser una investigación con un enfoque cualitativo la población y el muestreo fueron intencionados o por conveniencia.

3.6. Instrumentos de Recolección de Datos

Al ser una investigación con un enfoque cualitativo los instrumentos para la recolección de datos no son estandarizados ni predeterminados. Los instrumentos utilizados fueron el de grupo focal, ya que permite reuniones con los involucrados del proceso como los administradores de bases de datos y gerentes, en donde se realiza el levantamiento de requerimientos y con esto se vieron las necesidades de la unidad de negocio. Cabe destacar que, en este aspecto, por términos de confidencialidad, no se pueden mencionar las empresas donde esto fue aplicado.

3.7. Técnicas de Análisis de la Información

Como técnicas de análisis de la información se utilizaron los diagramas de flujo de datos, mapas conceptuales y los diagramas UML. Estas técnicas de análisis garantizaron el diseño de las bases de datos y el flujo del proceso de monitoreo de bases de datos.

3.8. Estrategia de Desarrollo de la Propuesta

Las estrategias de desarrollo de la propuesta que se usaron para llevar a cabo el proceso de monitoreo de bases de datos fue el ciclo de vida de datos y el ciclo de vida de bases de datos, estos dos para la creación y el desarrollo de las bases de datos donde fueron almacenados los datos, además, se utilizó SCRUM para poder llevar la creación del proyecto de una mejor forma.

Capítulo 4. Análisis de Diagnóstico

4.1. Problemática

En la actualidad podríamos decir que muchas pequeñas, medianas y grandes empresas al menos cuentan con un sistema de bases de datos en donde almacenan, en general, la mayoría de la información relacionada al negocio u organización. Como se sabe, los datos son una fuente muy poderosa para impulsar el desarrollo y crecimiento de una empresa, es común ver sistemas conectados y consumiendo información para alimentar a las aplicaciones y los usuarios finales de los datos necesarios, o un equipo de analistas que estudian los datos pare descifrar el mercado y buscar puntos de aprovechamiento para incrementar las ventas y así hacer crecer las ganancias, sin duda alguna, los datos son uno de los pilares más importantes en la informática.

Mencionado lo anterior, nace la siguiente pregunta: ¿es importante monitorear los sistemas gestores de bases datos? La respuesta es clara, sí, la importancia de monitorear la base de datos consiste en que puede ahorrar a la organización tiempo y dinero en caso de una falla, el monitoreo de base de datos promueve el soporte proactivo y preventivo, al conocer el estado de los servidores y los sistemas de bases de datos, se pueden tomar decisiones antes de que suceda una eminente falla, la cual pueda causar daños al negocio en su operación y así solucionar el problema con anterioridad.

4.2. Escenarios

En discusiones con diferentes equipos de trabajo en diferentes empresas y negocios relacionado con los ambientes de bases de datos SQL Server se han encontrado varios escenarios los cuales se explican a continuación:

Por motivos de confidencialidad no se menciona el nombre de las empresas, ni los servicios que ofrecen.

4.2.1. Ambientes sin Monitoreo

Según las investigaciones realizadas en diferentes empresas las que no tienen un sistema de monitoreo son las más vulnerables, ya que en estos casos se

trabajan y se solucionan los problemas que fallan en el momento, es decir, se provee soporte reactivo, se espera que sucedan los eventos, en vez de prevenirlos.

Este tipo de ambientes no tienen puntos positivos o ventajas, ya que el negocio está vulnerable a una falla y así deteniendo la productividad de la empresa.

Sin duda alguna, este tipo de ambientes debe ser solventado de alguna forma para así mantener los sistemas de bases de datos disponibles.

4.2.2. Ambientes Monitoreados con Herramientas de Terceros

Por lo general, en las empresas más grandes podemos encontrar muchos sistemas de bases de datos siendo monitoreados por una o más herramientas de terceros, sin embargo, algunas empresas medianas y pequeñas optan por esta opción.

La cantidad de herramientas que se encontraron pueden variar en la forma en que funcionan las organizaciones dentro de la empresa, en este caso se hallaron las siguientes:

- Empresas que se dividen en varias organizaciones, donde cada organización se encarga de su propio soporte, de estas formas cada organización escoge la herramienta por utilizar según sus criterios y presupuesto.
- Empresas que se dividen en varias organizaciones, donde esta cuenta con un equipo de soporte único para todas las diferentes bases de datos de las organizaciones utiliza únicamente un producto para todos los sistemas de bases de datos.

Es importante reconocer que estas herramientas son complejas, sin embargo, proveen muchísimas opciones en cuestión del monitoreo de bases de datos. No obstante, las licencias de estas herramientas son costosas y el precio puede incrementar en caso de que se necesite más de una licencia. La mayoría de las herramientas de terceros al ser de un proveedor no puede ser ajustada a la necesidad del negocio o equipo de bases de datos y en otras ocasiones esto implica un costo extra para el negocio. Además, los equipos de bases de datos no son

expertos en estas herramientas, por lo cual necesitan capacitación y en caso de una falla tienen que trabajar con el equipo de soporte del proveedor.

Sin embargo, estas aplicaciones son completamente desarrolladas por expertos en la materia y si un grupo u organización tiene la capacidad para pagar las licencias y el soporte pueden asegurar que los sistemas se van a monitorear en forma correcta.

Como referencia se utilizó el costo de los autores RedGate, Quest Foglight e Idera, los cuales disponen los precios de sus productos en los sitios web

RedGate SQLMonitor

Este producto tiene un costo de $1 645, el cual incluye un año de soporte más actualizaciones, sin embargo, este producto bajo este costo solo tiene un alcance de 4 instancias de SQL Server.

Idera ofrece diferentes aplicaciones y utilidades, sin embargo, el SQL Diagnostic for SQl Server es la herramienta especial que es utilizada para monitorear bases de datos de SQL Server, la cual tiene un costo de $1 996 por instancia de SQL Server.

Quest Foglight

Una herramienta muy completa desarrollada por Quest la cual tiene un costo de $2 398.99 por instancia de SQL Server.

4.2.3. Ambientes Monitoreados con un Proceso “Hecho en Casa”

Igual que en los escenarios anteriores se pueden ver en varios tipos de empresa, desde pequeñas empresas que tienen pequeños reportes y alertas hasta grandes empresas con procesos más complejos de monitoreo de alertas.

Las organizaciones que escogen este método ahorran gran cantidad de dinero evitando invertir en productos de terceros. Además, de crear una aplicación ajustándola y creándola con base en la necesidad del negocio.

Según lo investigado, la empresa que utiliza este método, dicho proceso de monitoreo fue desarrollado por el equipo de administradores de bases de datos, este sistema de monitoreo utiliza diferentes métodos para alertar y mantener a los administradores de bases de datos atentos a lo que sucede en los diferentes

sistemas, cabe destacar que este es un proceso hecho en casa que funciona para más de 2 000 instancias de SQL Server.

Una de las ventajas que tiene este tipo de ambientes es que los sistemas constantemente se actualizan a la necesidad del equipo y de esta forma se crea una cultura de soporte igual en todos los miembros del equipo de administradores de bases de datos, lo que influye en la creación de políticas y estándares para los diferentes procesos.

Cabe destacar que la necesidad del monitoreo y del negocio van a imponer la complejidad de la herramienta de monitoreo, ya que esto puede ir desde una simple alerta del cambio de un rol en un usuario a un sistema que se encarga automáticamente de aplicar los parches en los servidores de bases de datos. También estos ambientes pueden involucrar distintos equipos para integrar varios procesos o simplemente para crear una interfaz y, por último, el nivel técnico requerido es muy alto para poder soportar una herramienta de este tipo.

Sin embargo, el tamaño del proceso de monitoreo siempre va a depender del negocio.

4.3. Posible Solución

Al saber que un ambiente de monitoreo de bases de datos no es una opción, como se menciona en la sección 4.2.2 los sistemas de monitoreo de bases de datos de terceros son muy costosos, muchas empresas no pueden darse el lujo de hacer este tipo de inversión y como se menciona en la sección 4.2.3 el sistema de monitoreo de bases de datos puede ser complejo, puede involucrar a varios equipos y requiere un alto conocimiento para su mantenimiento; se propone un proceso de monitoreo de bases de datos que involucre todos los componentes que incluye una licencia de SQL Server, para así sacar provecho de todas las características que este incluye y así poder crear un proceso con una complejidad aceptable y que pueda abarcar la mayoría de los puntos más importantes que se monitorean en una base de datos, de esta forma conocer el estado de los sistemas de SQL Server.

Capítulo 5. Propuesta de Solución

5.1. Ambiente de Desarrollo

Para efectos de la investigación y el desarrollo se creó un ambiente de pruebas utilizando varias máquinas virtuales en VMWare el cual pueda cumplir las características más similares de un ambiente productivo real, cabe destacar que todos los SQL Server fueron parchados a su última versión. Al final, para crear el ambiente de desarrollo con instancias SQL Server se utilizaron las siguientes versiones:

- Microsoft SQL Server 2019 (RTM-CU8) (KB4577194) - 15.0.4073.23 (X64)
- Microsoft SQL Server 2017 (RTM-CU22) (KB4577467) - 14.0.3356.20 (X64)
- Microsoft SQL Server 2016 (SP2-CU15) (KB4577775) - 13.0.5850.14 (X64)
- Microsoft SQL Server 2014 (SP3-CU-GDR) (KB4535288) - 12.0.6372.1 (X64)
- Microsoft SQL Server 2012 (SP4-GDR) (KB4532098) - 11.0.7493.4 (X64)

5.1.1. Active Directory Domain Controller

Este servidor con el nombre “ADDC” se creó con el objetivo de agregar todos los servidores a un mismo dominio, de esta forma simulando un ambiente real con las cuentas de Windows. El dominio que se generó mediante Active Directory se nombró “WINSQL.local”.

Además, utilizando Active Directory se crearon las siguientes cuentas que fueron asignadas como administradores dentro de las instancias o las encargadas de ejecutar los servicios dentro de los servidores:

- **WINSQL\sqlserver_admin:** Cuenta de servicio, inicia el servicio de Database Engine.

- **WINSQL\sqlagent_admin:** Cuenta de servicio, inicia el servicio de SQLAgent.
- **WINSQL\sqlssis_admin:** Cuenta de servicio, inicia el SSIS.
- **WINSQL\sqlssrs_admin:** Cuenta de servicio, inicia el SSRS.
- **WINSQL\sqlssas_admin:** Cuenta de servicio, inicia el SSAS.
- **WINSQL\sqlmonitor:** Cuenta encargada de ejecutar el proceso de monitoreo, extrajo los datos necesarios de los diferentes servidores y se encargó de la transformación y carga de los datos para la base de datos SQLMonitor. También fue la cuenta encargada de ejecutar los diferentes SQLAgent “Jobs”.
- **WINSQL\asolanoa:** Cuenta para el administrador de bases de datos.

Además, todos los servidores fueron agregados al dominio WINSQL, como se muestra en la siguiente imagen:

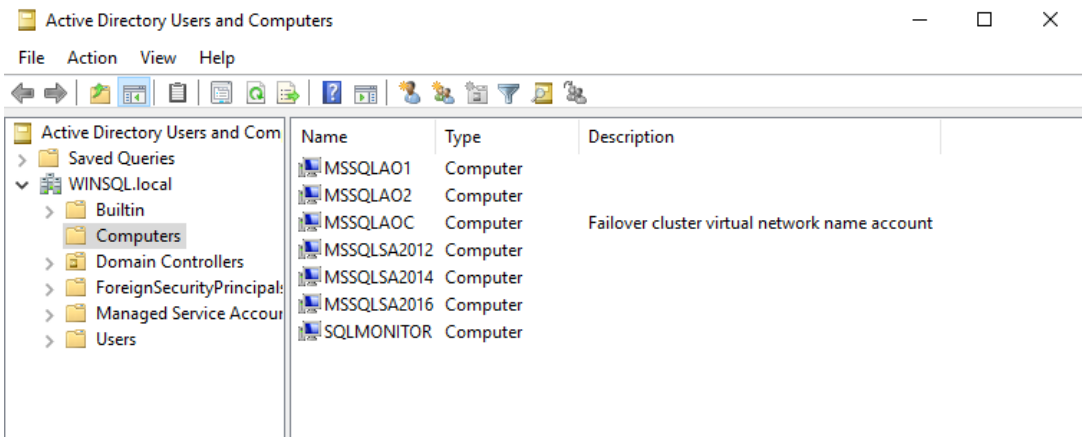


Figura 6: Servidores en Active Directory. Fuente: Tomada del servidor de pruebas ADDC

5.1.2. SQL Server Monitoring Server

Este es el servidor principal en donde se almacenaron los datos de los diferentes servidores dentro de los ambientes de pruebas, en este servidor se instaló la última versión disponible de SQL Server, SQL Server 2019 Enterprise Edition y aquí fueron ejecutados los reportes de SSRS y los paquetes de SSIS. El servidor tuvo el nombre “SQLMonitor”.

5.1.3. SQL Server Stand Alone

Para este caso se crearon varios servidores sencillos los cuales fueron nombrado inicialmente como “MSSQLSA”, seguidamente por la versión de SQL Server que fue instalada. A continuación, se mencionan los nombres de los servidores:

- MSSQLSA2012
- MSSQLSA2014
- MSSQLSA2016

Cada servidor tiene instalada una versión de SQL Server diferente para así probar el proceso de monitoreo en las diferentes versiones disponibles hasta el momento de SQL Server, en este caso son instancias nombradas y se instalaron las siguientes versiones:

- **MSSQLSA2012\SQL2012:** SQL Server 2012 Enterprise Edition.
- **MSSQLSA2014\SQL2014:** SQL Server 2014 Enterprise Edition.
- **MSSQLSA2016\SQL2016:** SQL Server 2016 Enterprise Edition.

5.1.4. SQL Server Always On Availability Groups

Está configuración se instaura al crear dos servidores con el nombre de “MSSQLAO1” y “MSSQLAO2”, este utiliza una configuración de Always On Availabity Groups, de manera que estos servidores se comunican entre sí. Es importante destacar que para estos se instaló una versión de SQL Server 2017 Enterprise Edition; en donde MSSQLAO1 es la réplica primaria y MSSQLAO2 es la réplica secundaria de este ambiente de alta disponibilidad.

5.2. Estandarización de SQL Server

Como parte del proyecto fue importante que este creara una cultura de estandarización de las instancias SQL Servers. Basándose en las mejores prácticas los servidores de prueba se crearon utilizando los estándares que se describen a continuación. Hay que tomar en cuenta que muchos de estos pueden ser implementados como estándar, sin embargo, se debe realizar un análisis previo del

uso de la aplicación y el comportamiento de esta misma, ya que este estándar puede variar.

5.2.1. Estandarización de Discos

Como se buscaba crear estándares y protocolos, los servidores del ambiente de pruebas fueron creados según las buenas prácticas en ambientes de SQL Server, por lo cual se crearon los siguientes discos para almacenar las bases de datos:

Volume	Layout	Type	File System	Status	Capacity	Free Space	% Free
	Simple	Basic		Healthy (EFI System Partition)	200 MB	200 MB	100 %
SO (C:)	Simple	Basic	NTFS	Healthy (Boot, Page File, Crash Dump, Primary Partition)	49.83 GB	29.84 GB	60 %
SQLData (D:)	Simple	Basic	NTFS	Healthy (Primary Partition)	49.80 GB	49.62 GB	100 %
SQLLog (L:)	Simple	Basic	NTFS	Healthy (Primary Partition)	20.02 GB	19.92 GB	100 %
TempDB (T:)	Simple	Basic	NTFS	Healthy (Primary Partition)	20.02 GB	19.93 GB	100 %

Figura 7. Distribución de Discos. Fuente: Tomada del servidor de pruebas SQLMonitor.

- **SO (C:):** Disco que almacena el sistema operativo, además de los diferentes softwares necesarios.
- **SQLData (D:):** Disco exclusivo para los archivos de datos de SQL Server mdf y ldf, tanto para las bases de datos de usuario como las de sistema.
- **SQLLog (L:):** Disco exclusivo para los archivos de log de transacciones de las bases de datos de usuario.
- **TempDB (T:):** Disco exclusivos para la base de datos de sistema TempDB como los mdf, ndf y ldf.

Además, como buena práctica todos los discos fueron configurados con un tamaño de 64 KB para unidad de asignación.

5.2.2. MAXDOP

La configuración de paralelismo en SQL Server más conocido como MAXDOP (Max Degree Of Parallelism) es una de las configuraciones más importantes en un servidor de SQL Server ya que este puede cambiar el rendimiento drásticamente si se utiliza la configuración correcta. Por lo cual, se creó la siguiente tabla basada en los estándares de Microsoft:

Configuración del Servidor	# de Procesadores	Recomendación
Servidor con un solo nodo	Menos de 4	Mantener MAXDOP en 0, configuración predeterminada.
	Menos de 8	Mantener MAXDOP en ese número de procesadores lógicos o por debajo de este.
	Más de 8	Mantener MAXDOP en 8.
Servidor con varios nodos	Menos de 16	Mantener MAXDOP en ese número de procesadores lógicos por nodo NUMA o por debajo de este.
	Más de 16	Mantener MAXDOP a la mitad del número de procesadores lógicos por nodo de NUMA con un valor máximo de 16.

Tabla 15. Configuración MAXDOP. Fuente: Elaboración Propia

5.2.3. Cost Threshold of Parellelism

Está configuración viene de forma predeterminada con el valor cinco, pero como buena práctica este valor debe estar en el rango de 25 y 50, por lo cual se ha tomado la decisión, para la estandarización, de utilizar el valor 25, sin embargo, este valor puede cambiar según sea necesario por la demanda de la aplicación.

5.2.4. Configuración de Memoria

SQL Server de forma predeterminada no configura el uso máximo de la memoria, esto puede causar un conflicto en el uso de memoria entre SQL Server y el sistema operativo. En este caso se debe ser cuidadoso, como estándar se sugiere dejar libre 6 GB para el sistema operativo y el resto para el uso de SQL Server.

En otras configuraciones se deben tomar en cuenta si existen otras aplicaciones instaladas dentro del servidor, reportería, entre otros.

Para casos extremos donde la configuración es mayor a 124 GB este se debe analizar y tomar la mejor opción para el procesamiento de SQL Server y el sistema operativo.

5.2.5. Management Data Warehouse

La herramienta de Managerment Data Warehouse que se incluye dentro de las licencias de SQL Server inicialmente se tomó en cuenta para ser parte del

estándar relacionado al proceso de monitoreo, sin embargo, después de una revisión y un análisis de la herramienta, esta fue excluida dentro del estándar ya que este tipo de reportes deben ser utilizados dentro de las diferentes instancias de SQL Server, es decir, los reportes generados por este proceso se deben ejecutar a nivel local de un servidor, por lo cual, se vuelve más complejo el proceso de extracción de datos.

5.2.6. Extended Events

Como estándar se crearon y se habilitaron varias configuraciones de Extended Events, estas configuraciones pueden ser muy útiles por servidor ya que funcionan constantemente como el SQL Profiler, muestran datos en vivo y generan información importante acerca de las diferentes opciones que existen en SQL Server. A diferencia del SQL Profiler, los Extended Events no tienen impacto en el rendimiento del servidor de SQL Server.

Por lo anterior, se implementaron los siguientes Extended Events basados en las plantillas ofrecidas por Microsoft:

- **Count Query Locks:** Conteo de las consultas que han tenido un bloqueo.
- **TSQL_Duration:** Captura la duración de las transacciones en SQL Server.
- **TSQL_Locks:** Captura las transacciones con deadlocks.
- **Tuning:** Captura información acerca de las ejecuciones de las transacciones.

5.3. Consultas para la Extracción de Datos

Parte fundamental de la investigación y desarrollo del proyecto fue clave crear una serie de consultas las cuales fueron capaces de extraer información necesaria para la generación y manipulación de los datos relacionados con el estado de los servidores de SQL Server, estos datos alimentan los reportes que se deben crear para así poder obtener un proceso completo para el monitoreo de bases de datos.

Los scripts que se describen a continuación son fundamentales para alimentar la base de datos SQLMonitor en donde se almacenaron los resultados de

las consultas necesarias que proveerán la información ya transformada y mejorada para su uso en reportes y consultas.

5.3.1. Datos a Nivel de Instancia

Las siguientes consultas fueron vitales para la extracción de datos que alimentaron una de las tablas principales para la base de datos de inventario, la cual contiene toda la información relacionada a las propiedades y configuraciones de las diferentes instancias de SQL Server en los diferentes servidores.

5.3.1.1. Consulta – Propiedades de Instancia

Las propiedades de las instancias se extraen para conocer diversa información relacionada con la configuración que tienen los servidores de SQL Server.

```
SELECT @@SERVERNAME AS 'InstanceName'
          ,SERVERPROPERTY('MachineName') AS 'HostName'
       ,SERVERPROPERTY('Collation') AS 'Collation'
          ,value_data AS 'TcpPort'
          ,REVERSE(SUBSTRING(REVERSE(@@VERSION),
CHARINDEX(')46X(',REVERSE(@@VERSION)), LEN(@@VERSION))) AS 'VERSION'
          ,SERVERPROPERTY('Edition') AS 'Edition'
          ,SERVERPROPERTY('InstanceDefaultDataPath') AS 'DefaultDataFilePath'
          ,SERVERPROPERTY('InstanceDefaultLogPath') AS 'DefaultLogPath'
          ,CASE
                       WHEN SERVERPROPERTY('IsHadrEnabled') = 0 THEN 'Disabled.'
                       WHEN SERVERPROPERTY('IsHadrEnabled') = 1 THEN 'Enabled.'
                       ELSE NULL
           END 'IsAlwaysOnAGEnable'
          ,CASE
                       WHEN SERVERPROPERTY('IsIntegratedSecurityOnly') = 1 THEN
'Windows Authentication Mode'
                       WHEN SERVERPROPERTY('IsIntegratedSecurityOnly') = 0 THEN 'SQL
Server and Windows Authentication Mode'
                       ELSE NULL
           END 'AuthenticationMode'
FROM sys.dm_server_registry
WHERE registry_key LIKE '%IPALL%'
```

5.3.1.2. Consulta – Configuración Avanzados de Instancia

SQL Server cuenta con una configuración avanzada las cuales habilitan ciertas funciones para realizar una serie de tareas avanzadas, además este tipo de configuración avanzada muestra datos como el paralelismo y la configuración de

memoria para SQL Server, entre otros. En este caso se filtra específicamente por los valores que son de interés para el proceso de monitoreo.

```
/*Script - Advance Configurations Data*/
SELECT  name AS 'ConfigurationName'
       ,value AS 'Value'
       ,description AS 'ConfigurationDescription'
FROM sys.configurations
WHERE configuration_id IN (518, 1520, 1538, 1539, 1540, 1543, 1544, 16390);
```

Asimismo, se agregó una pequeña consulta extra, la cual indica la cantidad de núcleos del CPU utilizados por SQL Server.

```
/*Script - CPU Count*/
SELECT COUNT(*) AS 'CPUCount'
      ,MAX(online_scheduler_count) AS 'LogicaCPUCount'
FROM sys.dm_os_nodes
WHERE node_id <> 64;
```

5.3.2. Datos a Nivel de Host

La siguiente consulta que fue creada en Powershell, extra de la información necesaria para almacenar los datos relacionados con el host en donde se encuentra el servidor de SQL Server.

```
<# ETL - Host Data #>

#Extract CPU information
$CPUInfo = Get-WmiObject Win32_Processor -ComputerName $HostName


#Extract Operating System information
$OSInfo = Get-WmiObject Win32_OperatingSystem -ComputerName $HostName

#Extract memory information
$TotalMemoryGB = Get-WmiObject CIM_PhysicalMemory -ComputerName $HostName |
Measure-Object -Property capacity -Sum | % { [Math]::Round(($_.sum / 1GB), 2)
```

5.3.3. Datos a Nivel de Base de Datos

Tener información sobre las bases de datos tanto de sistema como usuario es importante para el estado de estas, por lo cual se crearon las siguientes consultas que extraen la información más relevante de las bases de datos.

5.3.3.1. Consulta – Configuración de las Bases de Datos

La siguiente consulta extrae la información relacionada con la configuración de la base de datos lo cual determina el estado de la misma.

```
/*Script – Databases Information*/
SELECT @@SERVERNAME 'ServerName'
          ,d.database_id AS 'DatabaseId'
          ,d.[name] AS 'DatabaseName'
          ,d.[compatibility_level] As 'CompatibilityLevel'
          ,d.collation_name AS 'Collation'
          ,d.user_access_desc AS 'UserAccess'
          ,d.log_reuse_wait_desc AS 'LogReuseWait'
          ,d.is_read_only AS 'IsReadOnly'
          ,d.is_auto_close_on AS 'IsAutoCloseOn'
          ,d.is_auto_shrink_on AS 'IsAutoShrink'
          ,d.is_in_standby As 'IsStandBy'
          ,d.state_desc AS 'State'
          ,d.recovery_model_desc AS 'RecoveryModel'
          ,d.create_date AS 'DatabaseCreatedDate'
FROM sys.databases d
WHERE name NOT IN ('master', 'tempdb', 'model', 'msdb')
AND name NOT LIKE 'ReportServerTempDB%'
```

5.3.3.2. Consulta – Información de Archivos de Datos

Un conocimiento fundamental que se debe tomar en cuenta son los archivos de bases de datos más conocidos como archivos de datos y “logs”, por lo cual el siguiente script extrae esta información.

```
/*Script - Data Files Information*/
SELECT database_id AS 'DatabaseId'
          ,DB_NAME (database_id) AS 'DatabaseName'
          ,type_desc AS 'TypeDescription'
          ,name AS 'LogicaFilelName'
          ,REVERSE(SUBSTRING(REVERSE(physical_name), 0,
CHARINDEX('\',REVERSE(physical_name)))) AS 'PhysicalFileName'
          ,SUM((size*8)/1024) AS 'FileSize(MB)'
          ,state_desc AS 'FileState'
          ,REVERSE(SUBSTRING(REVERSE(physical_name),
CHARINDEX('\',REVERSE(physical_name)),LEN(physical_name))) AS 'FilePath'
FROM sys.master_files
GROUP BY database_id, name, type_desc, size, physical_name, state_desc
ORDER BY DB_NAME (database_id);
```

5.3.4. Acceso de Usuario con Altos Privilegios

Como buena práctica se debe limitar el uso de accesos de tipos administrativo como los sysadmin, serveradmin, securityadmin y procesadmin, los cuales se definen de la siguiente forma según Microsoft (2017):

- **Sysadmin:** El privilegio más alto; pueden realizar cualquier actividad en el servidor.
- **Serveradmin:** Tienen el privilegio de cambiar las opciones de configuración de todo el servidor y apagar el servicio de SQL Server.
- **Securityadmin:** Pueden administrar los inicios de sesión y sus propiedades
- **Processadmin:** Pueden terminar los procesos que se ejecutan en una instancia de SQL Server.

5.3.4.1. Consulta – Verificación de Altos Privilegios

El objetivo de la siguiente consulta es extraer la información relacionada con los usuarios que cuenten con un permiso de sysadmin, serveradmin, securityadmin y/o procesadmin.

```
/*Scrip - Privilege User Access*/
SELECT s.name AS 'LoginName'
          ,sp.type_desc AS 'LoginType'
          ,CASE
               WHEN s.sysadmin = 1 THEN 'Yes'
               ELSE 'No'
           END 'IsSysadmin'
          ,CASE
               WHEN s.serveradmin = 1 THEN 'Yes'
               ELSE 'No'
           END 'IsServeradmin'
          ,CASE
               WHEN s.securityadmin = 1 THEN 'Yes'
               ELSE 'No'
           END 'IsSecurityadmin'
          ,CASE
               WHEN s.processadmin = 1 THEN 'Yes'
               ELSE 'No'
           END 'IsProcessadmin'
          ,CASE
               WHEN sp.is_Disabled = 1 THEN 'Yes'
               ELSE 'No'
           END 'IsDisabled'
          ,s.createdate AS 'AccountCreateDate'
          ,s.updatedate AS 'AccountUpdateDate'
FROM syslogins s
INNER JOIN sys.server_principals sp ON sp.sid = s.sid
WHERE sp.name NOT LIKE '##%'
AND sp.name NOT LIKE 'NT %'
ORDER BY s.name;
```

5.3.5. Respaldos de Bases de Datos

Es importante conocer el historial de respaldos de las bases de datos en los ambientes de SQL Server tanto como para las bases de datos de usuario, como las de sistema, por lo cual la siguiente consulta extrae la información relacionada con los respaldos.

5.3.5.1. Consulta – Respaldos de Bases de Datos

Para esta consulta solo se extrae la información de los respaldos creados los últimos 30 días por servidor. Además, se extrae solamente la información relacionada con los respaldos de la siguiente manera:

```
/*Script - Backup History*/
SELECT s.database_name AS 'DatabaseName'
          ,s.user_name AS 'LoginName'
          ,CASE s.[type]
               WHEN 'D' THEN 'Database'
               WHEN 'I' THEN 'Differential'
               WHEN 'L' THEN 'Log'
               WHEN 'F' THEN 'File or filegroup'
               WHEN 'G' THEN 'Differential file'
               WHEN 'P' THEN 'Partial'
               WHEN 'Q' THEN 'Differential partial'
               ELSE NULL
           END 'BackupType'
          ,s.recovery_model AS 'RecoveryModel'
          ,s.[compatibility_level] AS 'CompatibilityLevel'
          ,s.collation_name AS 'Collation'
          ,CASE s.is_copy_only
               WHEN 1 THEN 'Yes'
               ELSE 'No'
           END 'IsCopyOnly'
          ,CASE mf.device_type
               WHEN 2 THEN 'Disk'
               WHEN 5 THEN 'Tape'
               WHEN 7 THEN 'Virtual Device'
               WHEN 9 THEN 'Azure Storage'
               WHEN 105 THEN 'A permanent backup device'
               ELSE NULL
           END 'DeviceType'
           ,mf.physical_device_name AS 'PhysicalDeviceName'
          ,CONVERT(DECIMAL(10,2),((s.backup_size/1024)/1024)) AS 'BackupSize(MB)'
          ,DATEDIFF(MI, s.backup_start_date, s.backup_finish_date) AS 'TimeTaken(Min)'
          ,s.backup_start_date AS 'BackupStartDate'
          ,s.backup_finish_date AS 'BackupEndDate'
FROM msdb.dbo.backupset s
INNER JOIN msdb.dbo.backupmediafamily mf ON mf.media_set_id = s.media_set_id
WHERE s.backup_start_date >= GETDATE() - 30
ORDER BY s.backup_start_date DESC;
```

5.3.6. Restauraciones de Bases de Datos

Conocer la información de las restauraciones de las bases de datos que se han realizado no es tan importante como conocer el reporte de bases de datos respaldadas, sin embargo, está información puede ser útil para conocer el historial de restauraciones.

5.3.6.1. Consulta – Restauraciones de Bases de Datos.

Al igual que las restauraciones esta consulta solamente devuelve los datos de los últimos 30 días de las restauraciones realizadas:

```
/*Script - Restore History*/
SELECT DISTINCT rh.destination_database_name AS 'DatabaseName'
      ,rh.user_name AS 'LoginName'
         ,CASE rh.restore_type
              WHEN 'D' THEN 'Database'
              WHEN 'F' THEN 'File'
              WHEN 'G' THEN 'Filegroup'
              WHEN 'I' THEN 'Differential'
              WHEN 'L' THEN 'Log'
              WHEN 'V' THEN 'VerifyOnly'
           ELSE NULL
          END AS 'RestoreType'
         ,CASE rh.replace
              WHEN 1 THEN 'Yes'
              ELSE 'No'
          END 'Replace'
         ,CASE rh.recovery
              WHEN 1 THEN 'Yes'
              ELSE 'No'
          END 'Recovery'
          ,bmf.physical_device_name as BackupFileLocation
         ,(SELECT rf.destination_phys_name FROM msdb..restorefile rf WHERE rf.restore_history_id =
rh.restore_history_id AND rf.file_number = 1) AS 'DestinationDataFile'
         ,(SELECT rf.destination_phys_name FROM msdb..restorefile rf WHERE rf.restore_history_id =
rh.restore_history_id AND rf.file_number = 2) AS 'DestinationLogFile'
         ,rh.restore_date AS 'RestoreDate'
FROM msdb.dbo.restorehistory rh
INNER JOIN msdb..backupset bs ON rh.backup_set_id = bs.backup_set_id
INNER JOIN msdb..backupmediafamily bmf ON bs.media_set_id = bmf.media_set_id
ORDER BY rh.restore_date DESC, rh.destination_database_name;
```

5.3.7. Estado de Always On

Muchas empresas necesitan de una configuración de alta disponibilidad para evitar afectaciones en su servicio en caso de algún fallo o problemas en sus servidores de alta disponibilidad, por eso, es importante conocer la salud de la configuración de SQL Server Always On.

5.3.7.1. Consulta – Estado de Always On a Nivel de Instancia

Always On tiene varios puntos de vista en donde conocer el estado del clúster a nivel de "server" (servidor), son de importancia para conocer los datos relacionados con el estado de los diferentes nodos:

```
/*Script - Always On Instance Status*/
SELECT ars.replica_id AS 'ReplicaId'
          ,ag.group_id AS 'GroupId'
          ,ag.name AS 'AvailabilityGroupName'
          ,ars.role_desc AS 'Role'
          ,ar.replica_server_name AS 'InstanceName'
          ,ar.availability_mode_desc AS 'AvailabilityMode'
          ,ar.failover_mode_desc AS 'FailoverMode'
          ,ars.connected_state_desc AS 'ConnectedState'
          ,ars.recovery_health_desc AS 'RecoveryHealth'
          ,ars.synchronization_health_desc AS 'SynchronizationHealth'
          ,ars.last_connect_error_number AS 'LastConnectErrorNumber'
          ,ars.last_connect_error_description AS 'LastConnectErrorDescription'
          ,ars.last_connect_error_timestamp AS 'LastConnectErrorTimeStamp'
FROM sys.dm_hadr_availability_replica_states ars
INNER JOIN sys.availability_groups ag ON ag.group_id = ars.group_id
INNER JOIN sys.availability_replicas ar ON ar.replica_id = ars.replica_id;
```

5.3.7.2. Consulta – Estado de Always On a Nivel de Bases de Datos

Como complemento del punto 5.2.6.1, es necesarios también conocer el estado de la sincronización entre los nodos que conforman la configuración SQL Server Always On, por lo mismo, se creó la siguiente consulta para extraer esta información:

```
/*Script - Always On Database Data*/
SELECT drs.replica_id AS 'ReplicaId'
          ,ag.group_id AS 'GroupId'
          ,drs.group_database_id AS 'GroupDatabaseId'
          ,ag.name AS 'AvailabilityGroupName'
          ,DB_NAME(drs.database_id) AS 'DatabaseName'
          ,CASE
                      WHEN drs.is_primary_replica = 1 THEN 'Yes'
                      ELSE 'No'
           END AS 'IsPrimaryReplica'
           ,CASE
                      WHEN dcs.is_failover_ready = 1 THEN 'Yes'
                      ELSE 'No'
           END 'IsFailoverReady'
          ,CASE
                      WHEN dcs.is_database_joined = 1 THEN 'Yes'
                      ELSE 'No'
           END 'IsDatabaseJoined'
          ,drs.synchronization_state_desc AS 'SynchronizationState'
          ,drs.synchronization_health_desc AS 'SynchronizationStateHealth'
          ,drs.database_state_desc AS 'DatabaseState'
          ,CASE
                      WHEN drs.is_suspended = 1 THEN 'Yes'
                      ELSE 'No'
           END AS 'IsSuspended'
          ,drs.suspend_reason_desc AS 'SuspenedReason'
          ,drs.last_hardened_lsn AS 'LastHardenedLSN'
          ,drs.last_hardened_time AS 'LastHardenedTime'
          ,drs.last_commit_lsn AS 'LastCommitLSN'
          ,drs.last_commit_time AS 'LastCommitTime'
          ,DATEDIFF(s,last_hardened_time,GETDATE()) as 'SecondsBehindPrimary'
FROM sys.dm_hadr_database_replica_states drs
INNER JOIN sys.availability_groups ag ON ag.group_id = drs.group_id
INNER JOIN sys.dm_hadr_database_replica_cluster_states dcs ON dcs.replica_id =
drs.replica_id
AND dcs.group_database_id = drs.group_database_id
ORDER BY drs.group_database_id, drs.is_primary_replica DESC;
```

5.3.8. Estado de los “Jobs”

Varias empresas configuran tareas de calendarización constantemente para que sean ejecutadas durante el día, por lo cual en ocasiones es importante conocer el estado de los “Jobs” que ejecuta SQL Server constantemente. Por eso la extracción de estos datos nos puede ayudar a identificar el estado de las ejecuciones de los “Jobs”.

5.3.8.1. Consulta – Estado de los Jobs

La siguiente consulta extrae la información relacionada a los SQL Server “Jobs”, en este caso se extrajeron diferentes datos como lo son el último mensaje de error y el estado de la última ejecución del “Jobs”.

```
/*Script - Jobs Status*/
SELECT j.job_id AS 'JobId'
          ,j.[name] AS 'JobName'
          ,SUSER_SNAME(j.owner_sid) AS 'JobOwner'
          ,CASE
                      WHEN j.[enabled] = 1 THEN 'Yes'
                      ELSE 'No'
           END 'IsEnabled'
          ,jh.step_name AS StepName
          ,jh.message AS [Message]
          ,CASE jh.run_status
               WHEN 0
                      THEN 'Failed'
               WHEN 1
                      THEN 'Succeeded'
               WHEN 2
                      THEN 'Retry'
               WHEN 3
                      THEN 'Canceled'
               WHEN 4
                      THEN 'In Progress'
               ELSE NULL
           END RunStatus
          ,js.last_run_duration AS 'LastRunDuration'
          ,js.last_run_date AS 'LastRunDate'
          ,js.last_run_time AS 'LastRunTime'
          ,j.date_created AS 'JobDateCreated'
          ,j.date_modified AS 'JobDateModified'
FROM msdb..sysjobs j
INNER JOIN msdb..sysjobhistory jh ON jh.job_id = j.job_id
INNER JOIN msdb..sysjobservers js ON js.job_id = j.job_id
INNER JOIN msdb..syscategories c ON c.category_id = j.category_id
WHERE jh.run_date = (SELECT MAX(jh2.run_date)
                                  FROM msdb..sysjobhistory jh2
                                     WHERE jh2.job_id = jh.job_id )
AND jh.run_time = (SELECT MAX(jh2.run_time)
                                FROM msdb..sysjobhistory jh2
                                WHERE jh2.job_id = jh.job_id
                                AND jh2.run_date = jh.run_date)
AND jh.instance_id = (SELECT MAX(jh2.instance_id)
                                   FROM msdb..sysjobhistory jh2
                                   WHERE jh2.job_id = jh.job_id
                                      AND jh2.step_id <> 0)
AND jh.step_id <> 0
ORDER BY j.name
```

5.3.9. Bloqueos y Deadlocks

En muchos ambientes productivos se pueden encontrar con bloqueos en las transacciones que se realicen, por eso es importante conocer al momento los bloqueos existentes en las transacciones y los causantes de estos.

5.3.9.1. Consulta – Bloqueos y Deadlocks en las Transacciones

La consulta que se creó con el objetivo de crear una lista de las transacciones bloqueadas por otras, además, muestra la transacción responsable de iniciar el bloqueo y el comando que ejecuta cada transacción.

```
/*Script - Check Blocking and Deadlocks Transaction*/
SELECT  db_name(dtl.resource_database_id) AS 'Database'
       ,dtl.resource_type AS 'ResourceType'
       ,CASE
            WHEN dtl.resource_type IN ('DATABASE', 'FILE', 'METADATA') THEN
dtl.resource_type
            WHEN dtl.resource_type = 'OBJECT' THEN
OBJECT_NAME(dtl.resource_associated_entity_id)
            WHEN dtl.resource_type IN ('KEY', 'PAGE', 'RID') THEN (SELECT
OBJECT_NAME(object_id)
                                                                   FROM
sys.partitions
                                                                   WHERE
sys.partitions.hobt_id = dtl.resource_associated_entity_id)
             ELSE 'Unidentified'
        END AS 'ParentObject'
       ,dtl.request_mode AS 'LockType'
       ,dtl.request_status AS 'RequestStatus'
       ,dowt.wait_duration_ms AS 'WaitDuration(ms)'
       ,dowt.wait_type AS 'WaitType'
       ,dowt.session_id AS 'BlockedSessionID'
       ,des_blocked.login_name AS 'BlockedLogin'
       ,SUBSTRING(dest_blocked.text, (der.statement_start_offset / 2) + 1, ( CASE WHEN
der.statement_end_offset = -1 THEN DATALENGTH(dest_blocked.text) ELSE
der.statement_end_offset
                    END - der.statement_start_offset) / 2) AS 'BlockedCommand'
       ,dowt.blocking_session_id AS 'BlockingSessionID'
       ,des_blocking.login_name AS 'BlockingLogin'
       ,dest_blocking.text AS 'BlockingCommand'
       ,dowt.resource_description AS 'BlockingResourceDetail'
FROM sys.dm_tran_locks dtl
INNER JOIN sys.dm_os_waiting_tasks dowt ON dtl.lock_owner_address =
dowt.resource_address
INNER JOIN sys.dm_exec_requests der ON dowt.session_id = der.session_id
INNER JOIN sys.dm_exec_sessions des_blocked ON dowt.session_id =
des_blocked.session_id
INNER JOIN sys.dm_exec_sessions des_blocking ON dowt.blocking_session_id =
des_blocking.session_id
INNER JOIN sys.dm_exec_connections dec ON dowt.blocking_session_id =
dec.most_recent_session_id
CROSS APPLY sys.dm_exec_sql_text(dec.most_recent_sql_handle) AS dest_blocking
CROSS APPLY sys.dm_exec_sql_text(der.sql_handle) AS dest_blocked
ORDER BY dowt.blocking_session_id
```

5.3.10. Indices

Los índices en las diferentes tablas de una base de datos pueden mejorar drásticamente el rendimiento de una consulta simplemente agregando uno, por lo

cual es indispensable realizar un reporte que identifique y genere el comando que debe ser ejecutado para crear el índice.

5.3.10.1. Consulta - Índices Faltantes

Esta consulta realiza una extracción y creación de los posibles índices faltantes en las diferentes tablas de las bases de datos de un servidor SQL Server, cabe destacar, que el resultado sobre los índices faltantes debe ser probado por los diferentes equipos de desarrollo, los cuales deben verificar si el índice puede ayudar en el rendimiento de las consultas. A continuación, se muestra la consulta:

```
SELECT db.name AS DatabaseName
         ,OBJECT_NAME(id.object_id, db.database_id) AS 'ObjectName'
         ,id.statement AS 'FullyQualifiedObjectName'
         ,gs.user_seeks * gs.avg_total_user_cost * (gs.avg_user_impact * 0.01) AS
'IndexAdvantage'
         ,id.equality_columns AS 'EqualityColumns'
         ,id.inequality_columns AS 'InEqualityColumns'
         ,id.included_columns AS 'IncludedColumns'
         ,gs.unique_compiles AS 'UniqueCompiles'
         ,gs.user_seeks AS 'UserSeeks'
         ,gs.user_scans AS 'UserScans'
         ,gs.last_user_seek AS 'LastUserSeekTime'
         ,gs.last_user_scan AS 'LastUserScanTime'
         ,gs.avg_total_user_cost AS 'AvgTotalUserCost'
         ,gs.avg_user_impact AS 'AvgUserImpact'
         ,'CREATE INDEX IX_' + OBJECT_NAME(id.object_id, db.database_id) + '_' +
REPLACE(REPLACE(REPLACE(ISNULL(id.equality_columns, ''), ', ', '_'), '', ''), '', '')
+
              CASE
             WHEN id.equality_columns IS NOT NULL AND id.inequality_columns IS NOT
NULL THEN '_'
              ELSE ''
           END + REPLACE(REPLACE(REPLACE(ISNULL(id.inequality_columns, ''), ', ',
'_'), '', ''), '', '') + '_' + LEFT(CAST(NEWID() AS NVARCHAR(64)), 5) + '' + ' ON ' +
id.statement + ' (' + ISNULL(id.equality_columns, '') +
              CASE
             WHEN id.equality_columns IS NOT NULL AND id.inequality_columns IS NOT
NULL THEN ','
              ELSE ''
           END + ISNULL(id.inequality_columns, '') + ')' + ISNULL(' INCLUDE (' +
id.included_columns + ')', '') AS 'ProposedIndex'
         ,CAST(CURRENT_TIMESTAMP AS smalldatetime) AS 'CollectionDate'
FROM sys.dm_db_missing_index_group_stats gs WITH (NOLOCK)
INNER JOIN sys.dm_db_missing_index_groups ig WITH (NOLOCK) ON gs.group_handle =
ig.index_group_handle
INNER JOIN sys.dm_db_missing_index_details id WITH (NOLOCK) ON ig.index_handle =
id.index_handle
INNER JOIN sys.databases db WITH (NOLOCK) ON db.database_id = id.database_id
WHERE  db.database_id = DB_ID()
ORDER BY ObjectName, IndexAdvantage DESC;
```

5.3.11. Estado de los Servicios SQL Server

SQL Server cuenta con varios servicios que ejecutan diferentes funciones, por lo cual es de suma importancia conocer el estado de los servicios, los servicios a revisar son los siguientes:

- SQL Server Database Engine.
- SQL Server Integrations Services.
- SQL Server Analysis Services.
- SQL Server Reporting Services.

5.3.11.1. Consulta – Estado de los Servicios SQL Server

En este caso se utilizó el lenguaje de programación PowerShell para extraer está información, por ahora este es el simple comando que se ejecuta para obtener el resultado con el estado de los servicios de SQL Server. Más adelante se complementa con una función especial que se utilizó para ejecutar el ETL de esta información.

```
<# Script - Check SQL Services Status #>
#DatabaseEngine
Get-WmiObject -Class WIN32_service -ComputerName $HostName | Where-Object
{$_.name -like 'MSSQLServer' -OR $_.name -like 'MSSQL$*' }

#SQLAgent
Get-WmiObject -Class WIN32_service -ComputerName $HostName | Where-Object
{$_.name -like 'SQLSERVERAGENT' -OR $_.name -like 'SQLAgent$*' }

#SSAS
Get-WmiObject -Class WIN32_service -ComputerName $HostName | Where-Object
{$_.name -like 'MSSQLServerOLAPService' -OR $_.name -like 'MSOLAP$*' }

#SSRS
Get-WmiObject -Class WIN32_service -ComputerName $HostName | Where-Object
{$_.name -like 'SQLServerReportingServices' -OR $_.name -like 'ReportServer*'
}

#SSIS
Get-WmiObject -Class WIN32_service -ComputerName $HostName | Where-Object
{$_.name -like "*MsDtsServer*"}
```

5.3.12. Espacio en los Discos

Los archivos de SQL Server constantemente están aumentando su tamaño, en especial el archivo del log de transacciones, por lo cual conocer el espacio disponible de los mismos puede evitar problemas.

5.3.12.1. Consulta – Espacio en los Discos del Servidor

La siguiente consulta de PowerShell extrae los datos relacionados con el espacio de los discos donde se ubica SQL Server, según el estándar con el que se crearon las máquinas virtuales:

```
Get-WmiObject -Class Win32_logicaldisk -Filter "DeviceId != 'C:'" |
Select-Object -Property DeviceID, VolumeName,
@{L='FreeSpaceGB';E={"{0:N2}" -f ($_.FreeSpace /1GB)}},
@{L="Capacity";E={"{0:N2}" -f ($_.Size/1GB)}} | Format-Table
```

5.3.13. Consultas Exigentes para el CPU, I/O y la Memoria

Constantemente las consultas que son ejecutadas en un servidor de SQL Server pueden tener diferentes comportamientos al tomar en cuenta que puede mostrar lentitud al momento de la ejecución, sin embargo, no se encuentran bloqueos, estos pueden estar relacionado con el uso del CPU, I/O y la memoria del servidor. Por lo cual se crearon las siguientes consultas:

5.3.13.1. Consulta – Consultas Intensas para el CPU

La siguiente consulta genera un reporte de las últimas 20 consultas que tienden a exigir mayor rendimiento al CPU:

```
/*Script - CPU Intensive Queries*/
SELECT TOP (20)
          RANK() Over (ORDER BY deqs.total_worker_time DESC) As 'Rank'
         ,CONVERT(decimal(38,2), CONVERT(float, total_worker_time) / 1000) AS
'TotalCPUTime(ms)'
         ,execution_count AS 'ExecutionCount'
         ,CONVERT(decimal(38,2), (CONVERT(float, total_worker_time) /
execution_count) / 1000) AS 'AverageCPUTime(ms)'
         ,SUBSTRING(execText.text,
                           CASE WHEN deqs.statement_start_offset = 0  OR
deqs.statement_start_offset IS NULL THEN 1
                                 ELSE deqs.statement_start_offset/2 + 1 END,
                           CASE WHEN deqs.statement_end_offset = 0 OR
deqs.statement_end_offset = -1 OR deqs.statement_end_offset IS NULL THEN
LEN(execText.text)
                                 ELSE deqs.statement_end_offset/2 END -
                           CASE WHEN deqs.statement_start_offset = 0 OR
deqs.statement_start_offset IS NULL THEN 1
                                 ELSE deqs.statement_start_offset/2  END + 1) AS
'QueryText',
        execText.text AS 'ObjectText'
FROM    sys.dm_exec_query_stats deqs
        CROSS APPLY sys.dm_exec_sql_text(deqs.plan_handle) AS execText
ORDER BY deqs.total_worker_time DESC ;
```

5.3.13.2. Consulta – Consultas Intensas para el I/O

Al igual que en el punto 5.2.13.1, se genera un reporte que muestra la información de las 20 consultas que tienden a exigir mayor rendimiento al I/O.

```
/*Script - CPU Intensive Queries*/
SELECT TOP 20
         total_logical_reads AS 'LogicalReads'
        ,total_logical_writes AS 'LogicalWrites'
        ,execution_count AS 'ExecutionCount'
        ,total_logical_reads+total_logical_writes AS 'AggIO'
        ,(total_logical_reads+total_logical_writes)/(execution_count+0.0) AS 'AvgIO'
        ,st.TEXT AS 'QueryText'
        ,DB_NAME(st.dbid) AS 'DatabaseName'
        ,st.objectid AS 'ObjectId'
        ,creation_time AS  'CreationTime'
        ,last_execution_time AS 'LastExecutionTime'
FROM sys.dm_exec_query_stats qs
CROSS APPLY sys.dm_exec_sql_text(sql_handle) st
WHERE total_logical_reads+total_logical_writes > 0
AND sql_handle IS NOT NULL
ORDER BY [AggIO] DESC
```

5.3.13.3. Consulta – Consultas Intensas para la Memoria

Al igual que los puntos anteriores, es necesario conocer las consultas que consumen más memoria:

```
/*Script - I/O Intensive Queries*/
SELECT mg.granted_memory_kb AS 'GrantedMemory(kb)'
         ,mg.session_id AS 'SessionId'
         ,t.text AS 'QueyText'
         ,qp.query_plan AS 'QueryPlan'
FROM sys.dm_exec_query_memory_grants AS mg
CROSS APPLY sys.dm_exec_sql_text(mg.sql_handle) AS t
CROSS APPLY sys.dm_exec_query_plan(mg.plan_handle) AS qp
ORDER BY 1 DESC OPTION (MAXDOP 1)
```

5.3.14. Altos Consumidores de la TempDB

La base de datos de TempDB en muchos casos tiene un uso constante, lo cual implica un crecimiento en sus archivos de datos que puede provocar problemas en las consultas.

5.3.14.1. Consulta – Altos Consumidores de la TempDB

El siguiente script muestra las sesiones activas y el comando que está consumiendo gran parte de la base de datos de TempDB, además, muestra información como por ejemplo sobre el uso del CPU, lectura y escrituras al disco.

```
/*Script - TempDB Consumers*/
SELECT es.host_name AS 'HostName'
          ,es.login_name AS 'LoginName'
          ,es.program_name AS 'ProgamName'
          ,st.dbid AS 'QueryExecContextDBID'
          ,DB_NAME(st.dbid) AS 'QueryExecContextDBNAME'
          ,st.objectid AS 'ModuleObjectId'
          ,SUBSTRING(st.text, er.statement_start_offset/2 + 1,(CASE WHEN
er.statement_end_offset = -1 THEN LEN(CONVERT(nvarchar(max),st.text)) * 2

                                    ELSE er.statement_end_offset

                               END - er.statement_start_offset)/2) AS 'QueryText'
          ,tsu.session_id AS 'SessionId'
          ,tsu.request_id AS 'RequestId'
          ,tsu.exec_context_id AS 'ExecContextId'
          ,(tsu.user_objects_alloc_page_count - tsu.user_objects_dealloc_page_count)
AS 'OutstandingUserObjectPageCounts'
          ,(tsu.internal_objects_alloc_page_count -
tsu.internal_objects_dealloc_page_count) AS 'OutStandingInternalObjectsPageCounts'
          ,er.start_time AS 'StartTime'
          ,er.command AS 'Command'
          ,er.open_transaction_count AS 'OpenTransactionCount'
          ,er.percent_complete AS 'PercentComplete'
          ,er.estimated_completion_time AS 'EstimatedCompletionTime'
          ,er.cpu_time AS 'CPUTime'
          ,er.total_elapsed_time AS 'TotalElapsedTime'
          ,er.reads AS 'Reads'
          ,er.writes AS 'Writes'
          ,er.logical_reads AS 'LogicalReads'
          ,er.granted_query_memory AS 'GrantedQueryMemory'
FROM sys.dm_db_task_space_usage tsu inner join sys.dm_exec_requests er
 ON ( tsu.session_id = er.session_id and tsu.request_id = er.request_id)
inner join sys.dm_exec_sessions es ON ( tsu.session_id = es.session_id )
CROSS APPLY sys.dm_exec_sql_text(er.sql_handle) st
WHERE (tsu.internal_objects_alloc_page_count+tsu.user_objects_alloc_page_count) > 0
AND es.session_id!=@@spid
ORDER BY (tsu.user_objects_alloc_page_count -
tsu.user_objects_dealloc_page_count)+(tsu.internal_objects_alloc_page_count -
tsu.internal_objects_dealloc_page_count)  DESC
```

5.4. Proceso de Monitoreo

El proceso de monitoreo de bases de datos se creó en un repositorio en SQL Server 2019 llamado SQLMonitor dentro de la instancia de monitoreo en el servidor que lleva su mismo nombre. Para cumplir con la extracción, transformación y carga de los datos utilizaron dos esquemas, en donde se creó uno nuevo llamado “staging” y el otro fue llamado monitor. El esquema “staging” fue necesario para recibir los datos de los diferentes ambientes donde es necesario trabajar con datos históricos o datos que requieren una transformación especial y el esquema monitor identifica las tablas y objetos claves para la ejecución del proceso de monitoreo. En estos

casos especiales una vez que fueron tratados los datos, estos se almacenaron dentro de la base de datos SQLMonitor, los cuales incluyen la información de las diferentes instancias y servidores de las cuales se extrae la información utilizada para el monitor de bases de datos. Además, la base de datos SQLMonitor contiene la lista de servidores que son monitoreados por este proceso. Para la creación de esta base de datos se utilizaron las mejores prácticas de implementación para SQL Server.

Cabe destacar que el proceso de monitoreo necesita una tabla de configuración, la cual contuvo los datos necesarios para realizar las diferentes conexiones entre las instancias y los hosts, esto se debe a que este proceso ejecutó la extracción de datos utilizando dos métodos y la conexión de datos se realizó de forma dinámica en los diferentes métodos.

El primero fue por medio de la utilización de los paquetes de SQL Server Integrations Services, en este caso en particular extrajo la información relacionada con las instancias y las bases de datos de SQL Server.

El segundo método fue utilizando “scripts” de PowerShell los cuales solamente extrajeron datos relacionado con los servidores Windows y estos mismos “scripts” realizan un proceso de inserción de datos.

Ambos métodos fueron calendarizados mediante SQL Server Jobs, los cuales serán ejecutados en diferentes horarios dependiendo de la necesidad de los datos.

Por último, se generó un “dashboard” el cual muestra los reportes necesarios para visualizar los datos que pueden facilitar el trabajo de un administrador de bases de datos.

Es importante señalar que se debe crear una cuenta de servicio con la capacidad y roles necesarios para extraer e insertar la información de las diferentes bases de datos, instancias y servidores.

5.5. Inventario de Instancias y Bases de Datos

Como se comentó el punto 5.3, se creó un repositorio llamado SQLMonitor, esta base de datos recibió los datos de los diferentes servidores y también almacenó los datos después de que fueron tratados en el esquema “staging”, además la base de datos SQLMonitor fue la fuente de los diferentes reportes.

5.5.1. Esquemas

Como se comentó en puntos anteriores, se crearon dos esquemas llamados staging y monitor para identificar los objetos según la necesidad. A continuación, se define el uso de cada esquema:

- **“Staging”:** Es utilizado para los objetos que reciban datos y deban de procesar datos históricos y/o la transformación y limpieza de datos necesarios antes de que estos sean agregados a las tablas que pertenecen al esquema monitor.
- **Monitor:** Este esquema está diseñado para los objetos que forman parte del proceso principal de monitoreo y que funcionan como fuente para los diferentes reportes y alertas.

5.5.1.1. Esquema – “staging” y monitor

Los esquemas en este se crearon de forma simple, solamente se agregó dbo como dueño de los esquemas y se utilizaron los siguientes comandos para su creación:

```
/*Monitor schema*/
CREATE SCHEMA [monitor] AUTHORIZATION [dbo]
GO
/*Staging schema*/
CREATE SCHEMA [staging] AUTHORIZATION [dbo]
GO
```

5.5.2. Tabla de Configuración

Se desarrollaron varios objetos claves para que fueran utilizados al completar ciertas tareas necesarias para el funcionamiento del monitoreo de SQL Server, en esta sección se describieron algunas de los objetos más importantes y destacados de este proyecto.

5.5.2.1. Tabla de Configuración – tMonitorInstance y tMonitorHost

Como se mencionó en la sección 5.3, parte clave del proyecto fue crear una tabla que fue utilizada como fuente para crear las diferentes conexiones dinámicas que son necesarias para acceder a las instancias de SQL Server y Windows Server de donde se deben de extraer datos o simplemente correr reportes en vivo desde SQL Server Reporting Services.

Es importante mencionar que como parte del proceso los datos dentro de esta tabla deben ser ingresados manualmente, una vez agregados los datos el proceso funcionará según la calendarización de los SQL Server Jobs. En este caso se crearon dos tablas para cumplir esta tarea con el nombre de tMonitorInstance y tMonitorHost, una para agregar la información básica de la instancia de SQL Server y la segunda tabla para agregar los datos relacionados al host de Windows Server. Cabe destacar que estas tablas se crearon dentro del esquema monitor.

La siguiente tabla fue creada para poder cumplir este rol:

```
/*Configuration Table for Host*/
CREATE TABLE [monitor].[tMonitorHost](
	[HostId] [int] PRIMARY KEY IDENTITY(1,1) NOT NULL,
	[HostName] [varchar](255) NOT NULL,
	[Environment] [varchar](15) NOT NULL,
	[RecordCreatedDate] [datetime] DEFAULT (GETDATE()) NOT NULL,
	[RecordUpdatedDate] [datetime] DEFAULT (GETDATE())  NULL
)

/*Configuration Table for Instance*/
CREATE TABLE [monitor].[tMonitorInstance](
	[InstanceId] [int] PRIMARY KEY IDENTITY(1,1) NOT NULL,
	[InstanceName] [varchar](255) NOT NULL,
	[TcpPort] [char](4) NOT NULL,
	[Environment] [varchar](15) NOT NULL,
	[HostId] INT NOT NULL,
	[IsEnable] [bit] NOT NULL,
	[RecordCreatedDate] [datetime] DEFAULT (GETDATE()) NOT NULL,
	[RecordUpdatedDate] [datetime] DEFAULT (GETDATE()) NULL,
	CONSTRAINT [FK_tMonitorInstance_tMonitorHost] FOREIGN KEY([HostId]) REFERENCES
[monitor].[tMonitorHost] ([HostId])
)
```

5.5.2.2. Procedimiento Almacenado – spInsertMonitor

Para facilitar la inserción de datos se creó un procedimiento almacenado que inserta los datos necesarios para satisfacer las columnas de la tabla de configuración:

```
/*Inserts instance, port and hostname information into the configuration tables*/
CREATE PROCEDURE [monitor].[spInsertMonitor]
	@HostName VARCHAR(255),
	@InstanceName VARCHAR(255),
	@TcpPort CHAR(4),
	@Environment VARCHAR(15),
	@IsEnable BIT
AS
BEGIN
	SET NOCOUNT ON;

	DECLARE @HostId INT

	/*Check if Host is already in the table*/
	IF (SELECT HostName FROM [monitor].[tMonitorHost] WHERE HostName = @HostName)
= @HostName
	BEGIN
		SELECT @HostId = HostId FROM [monitor].[tMonitorHost] WHERE HostName =
@HostName;
		PRINT 'Host already added, host ' + @HostName +  ' ID is ' +
CONVERT(CHAR(4), @HostId) + '.';
	END
	ELSE
	BEGIN
		INSERT INTO [monitor].[tMonitorHost] (HostName, Environment)
		 VALUES (@HostName, @Environment);
		 SELECT @HostId = HostId FROM [monitor].[tMonitorHost] WHERE HostName =
@HostName;
	END

	/*Check if Instance is already in the table*/
	IF (SELECT InstanceName FROM [monitor].[tMonitorInstance] WHERE InstanceName =
@InstanceName) = @InstanceName
		 PRINT @InstanceName + ' instance already added.';
	ELSE INSERT INTO [monitor].[tMonitorInstance] (InstanceName,TcpPort, HostId,
Environment, IsEnable)
		 VALUES (@InstanceName, @TcpPort, @HostId, @Environment, @IsEnable);

END
```

A continuación, se muestra un ejemplo de la manera en que se debe ejecutar el procedimiento almacenado:

```
/*Insert values into configuration table*/
EXEC [monitor].[spInsertMonitor]
     @InstanceName = 'SQLMonitor' --SQL Server Instance Name
    ,@HostName     = 'SQLMonitor' --Windows Server Hostname
    ,@TcpPort      = '1433' --SQL Server TCP Port
    ,@Environment  = 'Test' --Environment type like test, pre-prod and prod
    ,@IsEnable     = 1  -- Is instance active?
```

5.5.2.3. Cuenta WINSQL\sqlmonitor

Para la ejecución de los diferentes procesos y funciones se utilizó la cuenta de Active Directory WINSQL\sqlmonitor, esta cuenta estuvo a cargo de ejecutar los diferentes ETL's tanto en PowerShell, como en SSIS, esta es una cuenta de servicio especial que tuvo el acceso necesario a las diferentes bases de datos y servidores del ambiente de pruebas. Una cuenta como esta es fundamental para poder extraer y cargar los datos. En este caso se agregó con acceso de administrador tanto a nivel de SQL Server como de Windows para facilitar la extracción de datos.

Cabe destacar, que también fue la cuenta que se utilizó para la creación de la cuenta proxy a nivel de SQL Server para la ejecución de los "Jobs" la cual tuvo el nombre de SQLMonitorProxy.

5.5.3. Diseños de Bases de Datos

A continuación, se muestra el diseño que se creó para la base de datos de SQLMonitor (Figura 9), estos diseños se desarrollaron con base en las consultas creadas en el punto 5.2.

5.5.3.1. Diseño SQLMonitor (Esquema monitor)

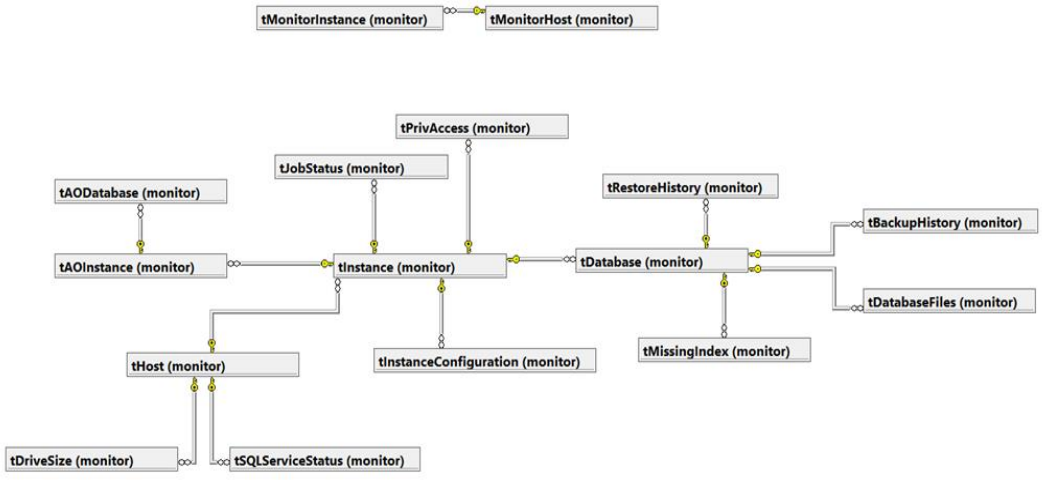


Figura 8: Diseño SQLMonitor. Fuente: Tomada del servidor de pruebas SQLMonitor

5.5.3.2. Conjunto de tablas (Esquema "Staging")

Parte importante de este proyecto son las tablas que almacenan datos temporalmente para su procesamiento que forman parte del esquema "staging". Es

importante mencionar que estas tablas no tienen ninguna relación como las tablas del esquema monitor. Por lo cual, al no existir un diseño de base de datos para el esquema “staging”, estas solamente se listaron a continuación:

- staging.tBackupHistory
- staging.tDatabase
- staging.tDatabaseFiles
- staging.tDriveSize
- staging.tHost
- staging.tInstance
- staging.tInstanceConfiguration
- staging.tJobStatus
- staging.tMissingIndex
- staging.tRestoreHistory
- staging.tSQLServiceStatus

Estas tablas son mencionadas y descritas con más detalles en el diccionario de datos mencionado en el siguiente punto 5.4.1.

5.5.4. Diccionario de Datos

En esta sección para facilitar la lectura al lector, simplemente se agregó las definiciones de cada una de las tablas que formaron partes de la base de datos SQLMonitor. Además, se creó el Anexo 1, el cual contiene el diccionario completo, con toda la información de las tablas. En este caso se utilizó la plantilla creada en la tabla 16**.**

Nombre		**Esquema**	
Definición			
Nombre Columna	**Tipo de Dato**	**Longitud**	**Descripción**

Tabla 16. Plantilla Diccionario de Datos.". Fuente: Elaboración Propia

A continuación, se muestran las definiciones de las tablas de la base de datos SQL Monitor, para esto se generó la tabla 17**.**

Nombre Tabla	Definición
[monitor].[tAODatabase]	Tabla que se conecta al sistema de reportería, suministrando los datos relacionados con los estados de bases de datos en configuraciones Always On.
[monitor].[tAOInstance]	Tabla que se conecta al sistema de reportería, suministrando los datos relacionados con los estados de instancias en configuraciones Always On.
[monitor].[tBackupHistory]	Tabla que se conecta al sistema de reportería, suministrando el historial de respaldos de los últimos 30 días.
[monitor].[tDatabase]	Tabla que se conecta al sistema de reportería, suministrando los datos relacionados a las bases de datos de las diferentes instancias.
[monitor].[tDatabaseFiles]	Tabla que se conecta al sistema de reportería, suministrando los datos relacionados con los archivos (mdf, ndf y ldf) de las diferentes bases de datos.
[monitor].[tDriveSize]	Tabla que se conecta al sistema de reportería, suministrando los datos sobre el tamaño de los discos de los hosts.
[monitor].[tHost]	Tabla que se conecta al sistema de reportería, suministrando los datos sobre los hosts en donde SQL Server está instalado.

[monitor].[tInstance]	Tabla que se conecta al sistema de reportería, suministrando los datos sobre las diferentes instancias de SQL Server.
[monitor].[tInstanceConfiguration]	Tabla que se conecta al sistema de reportería, suministrando las configuraciones avanzadas establecidas en cada instancia de SQL Server.
[monitor].[tJobStatus]	Tabla que se conecta al sistema de reportería, suministrando el estado de los “Jobs” ejecutados en las instancias.
[monitor].[tMissingIndex]	Tabla que se conecta al sistema de reportería, suministrando las sugerencias de los índices faltantes.
[monitor].[tMonitorHost]	Tabla que suministra la lista de los hosts de donde se debe extraer datos.
[monitor].[tMonitorInstance]	Tabla que suministra la lista de las instancias de donde se debe extraer datos.
[monitor].[tPrivAccess]	Tabla que se conecta al sistema de reportería, suministrando los datos sobre los altos privilegios dentro de las instancias.
[monitor].[tRestoreHistory]	Tabla que se conecta al sistema de reportería, suministrando el historial de restauraciones de los últimos 30 días.
[monitor].[tSQLServiceStatus]	Tabla que se conecta al sistema de reportería, suministrando el estado de los servicios a nivel de host relacionados a SQL Server.
[staging].[tAODatabase]	Tabla utilizada temporalmente para la manipulación y transformación de los datos antes de cargar los datos a la tabla [monitor].[tAODatabase].
[staging].[tAOInstance]	Tabla utilizada temporalmente para la manipulación y transformación de los datos antes de cargar los datos a la tabla [monitor].[tAOInstance].

[staging].[tBackupHistory]	Tabla utilizada temporalmente para la manipulación y transformación de los datos antes de cargar los datos a la tabla [monitor].[tBackupHistory] .
[staging].[tDatabase]	Tabla utilizada temporalmente para la manipulación y transformación de los datos antes de cargar los datos a la tabla [monitor].[tDatabase].
[staging].[tDatabaseFiles]	Tabla utilizada temporalmente para la manipulación y transformación de los datos antes de cargar los datos a la tabla [monitor].[tDatabaseFiles].
[staging].[tDriveSize]	Tabla utilizada temporalmente para la manipulación y transformación de los datos antes de cargar los datos a la tabla [monitor].[tDriveSize].
[staging].[tHost]	Tabla utilizada temporalmente para la manipulación y transformación de los datos antes de cargar los datos a la tabla [monitor].[tHost].
[staging].[tInstance]	Tabla utilizada temporalmente para la manipulación y transformación de los datos antes de cargar los datos a la tabla [monitor].[tInstance].
[staging].[tInstanceConfiguration]	Tabla utilizada temporalmente para la manipulación y transformación de los datos antes de cargar los datos a la tabla [monitor].[tInstanceConfiguration].
[staging].[tJobStatus]	Tabla utilizada temporalmente para la manipulación y transformación de los datos antes de cargar los datos a la tabla [monitor].[tJobStatus].
[staging].[tMissingIndex]	Tabla utilizada temporalmente para la manipulación y transformación de los datos antes de cargar los datos a la tabla [monitor].[tMissingIndex].
[staging].[tRestoreHistory]	Tabla utilizada temporalmente para la manipulación y transformación de los datos antes de cargar los datos a la tabla [monitor].[tRestoreHistory].

[staging].[tSQLServiceStatus]	Tabla utilizada temporalmente para la manipulación y transformación de los datos antes de cargar los datos a la tabla [monitor].[tSQLServiceStatus].

Tabla 17. Definición de Tablas. Fuente: Elaboración Propia

5.4. Procesos de ETL

El proceso de extracción, transformación y carga de datos se creó por medio de varios paquetes SSIS y comandos PowerShell los cuales cumplen las características necesarias para cargar las tablas de las bases de datos de SQLMonitor, en varios casos se utilizan tablas “staging” para mantener temporalmente los datos nuevos o utilizarlas como fuente de transformación antes de insertar los datos en las tablas que fueron creadas para el proceso de reportería. Además, muchos de estos procesos culminan con un procedimiento de carga o con una operación “merge” que fueron creadas en un procedimiento almacenado para una ejecución más sencilla.

5.4.1. Procedimientos Almacenados

A continuación, se agregó una lista de los procedimientos almacenados que cumplieron una función importante para la carga y transformación de datos para las tablas pertenecientes al esquema monitor. Es importante destacar que los procedimientos almacenados fueron divididos en dos tipos, unos llamados “spLoad” y otros “spMerge”, en donde “spLoad” realizó una limpieza de la tabla monitor asociada al proceso y seguidamente una inserción de los datos. El proceso “spMerge” como lo dice su nombre ejecuta una operación “merge” entre las tablas del esquema “staging” y las tablas del esquema monitor, esto porque las tablas de reportería no pueden ser truncadas y en algunos casos para la necesidad de comparación y creación de los datos entre los diferentes esquemas.

5.5.1.1. Operaciones Load

Los procedimientos almacenados que cargan datos y transforman datos utilizan el mismo proceso.

Primero se trunca la tabla relacionada con el esquema monitor, seguidamente se cargan los provenientes de la tabla “straging” la cual ejecuta un

“inner join” con la tabla monitor.tDatabsae o monitor.tInstance, esto para cargar el dato DatabaseId o el dato InstanceId, esta acción depende de la relación que tengan las tablas.

- **monitor.spLoadtBackupHistory:** Carga la tabla monitor.tBackupHistory, con los datos relacionados a los respaldos realizados los últimos 30 días.

```
CREATE PROCEDURE [monitor].[spLoadtBackupHistory]
AS
BEGIN
	TRUNCATE TABLE [monitor].[tBackupHistory];

	INSERT INTO [monitor].tBackupHistory (LoginName, BackupType, RecoveryModel,
CompatibilityLevel, Collation, IsCopyOnly, DeviceType, PhysicalDeviceName,
BackupSizeMB, TimeTakenMin, BackupStartDate, BackupEndDate, DatabaseId,
RecordCreatedDate)
	SELECT b.LoginName
		,b.BackupType
		,b.RecoveryModel
		,b.CompatibilityLevel
		,b.Collation
		,b.IsCopyOnly
		,b.DeviceType
		,b.PhysicalDeviceName
		,b.BackupSizeMB
		,b.TimeTakenMin
		,b.BackupStartDate
		,b.BackupEndDate
		,d.DatabaseId
		,GETDATE()
	FROM [staging].[tBackupHistory] b
	INNER JOIN [monitor].[tDatabase] d ON d.DatabaseName = b.DatabaseName AND
d.InstanceId = b.InstanceId
END
```

- **monitor.spLoadtDatabaseFiles:** Carga la tabla monitor.tDatabaseFiles con los datos relacionados con los archivos .mdf, .ndf y .ldf de las bases de datos, transformando la ruta en donde se encuentra el archivo al nombre de archivo físico y el directorio donde se encuentra el archivo.

```
CREATE PROCEDURE [monitor].[spLoadtDatabaseFiles]
AS
BEGIN
	TRUNCATE TABLE [monitor].[tDatabaseFiles];

	INSERT INTO [monitor].[tDatabaseFiles] (TypeDescription, LogicaFilelName,
PhysicalFileName, FilePath, FileState, FileSizeMB, DatabaseId, RecordCreatedDate,
RecordUpdatedDate)
	SELECT df.TypeDescription
		,df.LogicaFilelName
		,REVERSE(SUBSTRING(REVERSE(df.PhysicalFileName), 0,
CHARINDEX('\',REVERSE(df.PhysicalFileName)))) AS 'PhysicalFileName'
		,REVERSE(SUBSTRING(REVERSE(df.PhysicalFileName),
CHARINDEX('\',REVERSE(df.PhysicalFileName)),LEN(PhysicalFileName))) AS 'FilePath'
		,df.FileState
		,SUM((df.Size*8)/1024) AS 'FileSizeMB'
		,d.DatabaseId
		,GETDATE()
		,NULL
	FROM [staging].[tDatabaseFiles] df
	INNER JOIN [monitor].[tDatabase] d ON d.InstanceDatabaseId =
df.InstanceDatabaseId AND d.InstanceId = df.InstanceId
	GROUP BY df.TypeDescription, df.Size, LogicaFilelName, df.PhysicalFileName,
df.FileState, d.DatabaseId
	ORDER BY d.DatabaseId
END
```

- **monitor.spLoadtDriveSize:** Carga la tabla monitor.tDriveSize, en donde se cargan todos los datos relacionados al espacio que tiene cada disco dentro de un servidor.

```
CREATE PROCEDURE [monitor].[spLoadtDriveSize]
AS
BEGIN
	TRUNCATE TABLE [monitor].[tDriveSize];

	INSERT INTO [monitor].[tDriveSize]
	SELECT DriveLetter
		,DriveName
		,FreeSpaceGB
		,CONVERT(DECIMAL(5,1),(FreeSpaceGB * 100)/TotalSizeGB) AS
'FreeSpacePercentage'
		,CONVERT(DECIMAL(5,1),TotalSizeGB - FreeSpaceGB) AS 'UsedSpaceGB'
		,CONVERT(DECIMAL(5,1),((TotalSizeGB - FreeSpaceGB) *
100)/TotalSizeGB) AS 'UsedSpacePercentage'
		,TotalSizeGB
		,HostId
		,RecordCreatedDate
		,NULL
	FROM [staging].[tDriveSize];
END
```

- **monitor.spLoadtMissingIndex:** Carga la tabla monitor.tMissingIndex, con los datos sobre las recomendaciones de índices que podrían ser implementados.

```
CREATE PROCEDURE [monitor].[spLoadtMissingIndex]
AS
BEGIN
	TRUNCATE TABLE [monitor].[tMissingIndex];

	INSERT INTO [monitor].[tMissingIndex] (ObjectName, FullyQualifiedObjectName,
IndexAdvantage, EqualityColumns, InEqualityColumns, IncludedColumns, UniqueCompiles,
UserSeeks, UserScans, LastUserSeekTime, LastUserScanTime, AvgTotalUserCost,
AvgUserImpact, ProposedIndex, DatabaseId, RecordCreatedDate)
	SELECT mi.ObjectName
		,mi.FullyQualifiedObjectName
		,mi.IndexAdvantage
		,mi.EqualityColumns
		,mi.InEqualityColumns
		,mi.IncludedColumns
		,mi.UniqueCompiles
		,mi.UserSeeks
		,mi.UserScans
		,mi.LastUserSeekTime
		,mi.LastUserScanTime
		,mi.AvgTotalUserCost
		,mi.AvgUserImpact
		,mi.ProposedIndex
		,d.DatabaseId
		,GETDATE()
	FROM [staging].[tMissingIndex] mi
	INNER JOIN [monitor].[tDatabase] d ON d.DatabaseName = mi.DatabaseName AND
d.InstanceId = mi.InstanceId
END
```

- **monitor.spLoadtJobStatus:** Carga la tabla monitor.tJobStatus, en donde se carga la última ejecución de los “jobs” almacenados en una instancia.

```
CREATE PROCEDURE [monitor].[spLoadtJobStatus]
AS
BEGIN
	TRUNCATE TABLE [monitor].[tJobStatus];

	INSERT INTO [monitor].[tJobStatus] (JobId, JobName, JobOwner, IsEnabled,
StepName, Message, RunStatus, LastRunDurationSeconds, LastRunDuration,
LastRunDate, JobDateCreated, JobDateModified, InstanceId, RecordCreatedDate)
	SELECT JobId
		,JobName
		,JobOwner
		,IsEnabled
		,StepName
		,Message
		,RunStatus
		,CASE
			WHEN LEN(LastRunDuration) < 3
				THEN LastRunDuration
```

```
                    WHEN LEN(LastRunDuration) = 3
                           THEN (CAST(LEFT(LastRunDuration,1) AS INT) * 60) +
CAST(RIGHT(LastRunDuration,2) AS INT)
                    WHEN LEN(LastRunDuration) = 4
                           THEN (CAST(LEFT(LastRunDuration,2) AS INT) * 60) +
CAST(RIGHT(LastRunDuration,2) AS INT)
                    WHEN LEN(LastRunDuration) > 4
                           THEN (CAST(LEFT(LastRunDuration, LEN(LastRunDuration)
- 4) AS INT) * 3600) + (CAST(LEFT(RIGHT(LastRunDuration,4),2) AS INT) * 60) +
CAST(RIGHT(LastRunDuration,2) AS INT)
                    ELSE NULL
                END AS LastRunDurationSeconds
               ,CASE
                WHEN LEN(LastRunDuration) > 4
                    THEN CASE
                                    WHEN LEN(LastRunDuration)  = 5
                                           THEN '0' +
SUBSTRING(CAST(LastRunDuration AS VARCHAR(30)),0,LEN(LastRunDuration)-3) + ':' +
LEFT(RIGHT(LastRunDuration,4),2) + ':' + RIGHT(LastRunDuration,2)
                                    ELSE SUBSTRING(CAST(LastRunDuration AS
VARCHAR(30)),0,LEN(LastRunDuration)-3) + ':' +  LEFT(RIGHT(LastRunDuration,4),2)
+ ':' + RIGHT(LastRunDuration,2)
                                    END
                WHEN LEN(LastRunDuration) = 4
                    THEN CASE
                                    WHEN (RIGHT(LastRunDuration,2) = 00)
                                           THEN '00:' + CAST((LastRunDuration /
100) AS VARCHAR(2)) + ':' + CAST((LastRunDuration % 100) AS VARCHAR(2)) + '0'
                                    WHEN (RIGHT(LastRunDuration,2) IN
(01,02,03,04,05,06,07,08,09))
                                           THEN '00:' + CAST((LastRunDuration /
100) AS VARCHAR(2)) + ':0' + CAST((LastRunDuration % 100) AS VARCHAR(2))
                                    ELSE '00:' + CAST((LastRunDuration / 100) AS
VARCHAR(2)) + ':' + CAST((LastRunDuration % 100) AS VARCHAR(2))
                                    END
                WHEN LEN(LastRunDuration) = 3
                    THEN CASE
                                    WHEN (RIGHT(LastRunDuration,2) = 00)
                                           THEN '00:0' + CAST((LastRunDuration /
100) AS VARCHAR(2)) + ':' + CAST((LastRunDuration % 100) AS VARCHAR(2)) + '0'
                                    WHEN (RIGHT(LastRunDuration,2) IN
(01,02,03,04,05,06,07,08,09))
                                           THEN '00:0' + CAST((LastRunDuration /
100) AS VARCHAR(2)) + ':0' + CAST((LastRunDuration % 100) AS VARCHAR(2))
                                    ELSE '00:0' + CAST((LastRunDuration / 100) AS
VARCHAR(2)) + ':' + CAST((LastRunDuration % 100) AS VARCHAR(2))
                                    END
                WHEN LEN(LastRunDuration) = 2
                    THEN '00:00:' + CAST(LastRunDuration AS VARCHAR(2))
                ELSE'00:00:0' + CAST(LastRunDuration AS VARCHAR(2))
                END AS LastRunDuration
               ,CAST(LEFT(CAST(LastRunDate AS VARCHAR(30)),4) +'-'+
LEFT(RIGHT(CAST(LastRunDate AS VARCHAR(30)),4),2)  +'-'+ RIGHT(CAST(LastRunDate
AS VARCHAR(30)),2) + ' ' +
                STUFF(STUFF(RIGHT(REPLICATE('0', 6) +  CAST(LastRunTime as
varchar(6)), 6), 3, 0, ':'), 6, 0, ':') AS DATETIME) AS LastRunDate
```

```
,JobDateCreated
                ,JobDateModified
                ,InstanceId
                ,GETDATE()
        FROM [staging].[tJobStatus]
END
```

monitor.spLoadtRestoreHistory: Carga la tabla monitor.tRestoreHistory, carga los datos relacionados a las restauraciones realizadas en los últimos 30 días.

```
CREATE PROCEDURE [monitor].[spLoadtRestoreHistory]
AS
BEGIN
        TRUNCATE TABLE [monitor].[tRestoreHistory];

        INSERT INTO [monitor].[tRestoreHistory] (LoginName, RestoreType, Replace,
Recovery, BackupFileLocation, DestinationDataFile, DestinationLogFile, RestoreDate,
DatabaseId,RecordCreatedDate)
        SELECT rh.LoginName
                ,rh.RestoreType
                ,rh.Replace
                ,rh.Recovery
                ,rh.BackupFileLocation
                ,rh.DestinationDataFile
                ,rh.DestinationLogFile
                ,rh.RestoreDate
                ,d.DatabaseId
                ,GETDATE()
        FROM [staging].[tRestoreHistory] rh
        INNER JOIN [monitor].[tDatabase] d ON d.DatabaseName = rh.DatabaseName AND
d.InstanceId = rh.InstanceId
END
```

5.5.1.2. Operaciones "Merge"

Los procedimientos que utilizan la función "merge", descarta la eliminación completa de los datos en las tablas con el esquema monitor y ejecuta tres funciones diferentes que dependen del cumplimiento de las condiciones entre la comparación de las tablas "staging" con las tablas monitor. Las condiciones son las siguientes:

- **Update:** Si "staging" y monitor tienen una columna con algún dato diferente, esta actualiza la diferencia.
- **Insert:** Si el dato existe en "staging" y no en monitor, esta inserta el nuevo dato.

- **Delete:** Si el dato no existe en “staging” y sí en monitor, este elimina el dato, esta función no aplica para las tablas tHost, tDatabase y tInstances ya que son tablas principales y se debió de evitar la eliminación de los datos.

A continuación, se listaron los procedimientos almacenados que realizan “merge” para cargar los datos en las tablas monitor.

- **monitor.spMergetSQLServiceStatus:** Carga la tabla monitor.tSQLServiceStatus por medio de la función “merge” sobre el estado de los servicios utilizados por SQL Server.

```
CREATE PROCEDURE [monitor].[spMergetSQLServiceStatus]
AS
BEGIN

MERGE [monitor].[tSQLServiceStatus] AS TARGET
USING [staging].[tSQLServiceStatus] AS SOURCE
ON TARGET.ServiceName = SOURCE.ServiceName
AND TARGET.HostId = SOURCE.HostId
WHEN MATCHED AND (TARGET.StartMode<> SOURCE.StartMode OR TARGET.StartMode IS NULL)
                        OR (TARGET.Status<> SOURCE.Status OR TARGET.Status IS NULL)

THEN UPDATE SET TARGET.StartMode = SOURCE.StartMode
                         ,TARGET.Status = SOURCE.Status
                         ,TARGET.RecordUpdatedDate = GETDATE()


WHEN NOT MATCHED BY TARGET THEN
INSERT (ServiceName
       ,StartMode
          ,Status
          ,HostId
          ,RecordCreatedDate)
VALUES (SOURCE.ServiceName
          ,SOURCE.StartMode
          ,SOURCE.Status
          ,SOURCE.HostId
          ,SOURCE.RecordCreatedDate)

WHEN NOT MATCHED BY SOURCE THEN
DELETE;
END
```

- **monitor.spMergetDatabase:** Carga la tabla monitor.tDatabase. en donde se cargan los datos nuevos y se actualizan los viejos que están relacionados con la información de las bases de datos.

```
CREATE PROCEDURE [monitor].[spMergetDatabase]
AS
BEGIN

MERGE [monitor].[tDatabase] AS TARGET
USING [staging].[tDatabase] AS SOURCE
ON TARGET.InstanceDatabaseId = SOURCE.InstanceDatabaseId
AND TARGET.InstanceId = SOURCE.InstanceId
WHEN MATCHED AND (TARGET.DatabaseName<> SOURCE.DatabaseName OR
TARGET.DatabaseName IS NULL)
                        OR (TARGET.CompatibilityLevel<> SOURCE.CompatibilityLevel
OR TARGET.CompatibilityLevel IS NULL)
                        OR (TARGET.Collation<> SOURCE.Collation OR
TARGET.Collation IS NULL)
                        OR (TARGET.UserAccess<> SOURCE.UserAccess OR
TARGET.UserAccess IS NULL)
                        OR (TARGET.RecoveryModel<> SOURCE.RecoveryModel OR
TARGET.RecoveryModel IS NULL)
                        OR (TARGET.State<> SOURCE.State OR TARGET.State IS NULL)
                        OR (TARGET.LogReuseWait<> SOURCE.LogReuseWait OR
TARGET.LogReuseWait IS NULL)
                        OR (TARGET.IsReadOnly<> SOURCE.IsReadOnly OR
TARGET.IsReadOnly IS NULL)
                        OR (TARGET.IsAutoCloseOn<> SOURCE.IsAutoCloseOn OR
TARGET.IsAutoCloseOn IS NULL)
                        OR (TARGET.IsAutoShrink<> SOURCE.IsAutoShrink OR
TARGET.IsAutoShrink IS NULL)
                        OR (TARGET.IsStandBy<> SOURCE.IsStandBy OR
TARGET.IsStandBy IS NULL)
                        OR (TARGET.DatabaseCreatedDate<>
SOURCE.DatabaseCreatedDate OR TARGET.DatabaseCreatedDate IS NULL)

THEN UPDATE SET TARGET.DatabaseName = SOURCE.DatabaseName
                         ,TARGET.CompatibilityLevel = SOURCE.CompatibilityLevel
                         ,TARGET.Collation = SOURCE.Collation
                         ,TARGET.UserAccess = SOURCE.UserAccess
                         ,TARGET.RecoveryModel = SOURCE.RecoveryModel
                         ,TARGET.State = SOURCE.State
                         ,TARGET.LogReuseWait = SOURCE.LogReuseWait
                         ,TARGET.IsReadOnly = SOURCE.IsReadOnly
                         ,TARGET.IsAutoCloseOn = SOURCE.IsAutoCloseOn
                         ,TARGET.IsAutoShrink = SOURCE.IsAutoShrink
                         ,TARGET.IsStandBy = SOURCE.IsStandBy
                         ,TARGET.DatabaseCreatedDate = SOURCE.DatabaseCreatedDate
                         ,TARGET.RecordUpdatedDate = GETDATE()

WHEN NOT MATCHED BY TARGET THEN
INSERT (InstanceDatabaseId
       ,DatabaseName
       ,CompatibilityLevel
       ,Collation
```

```
        ,UserAccess
        ,RecoveryModel
        ,State
        ,LogReuseWait
        ,IsReadOnly
        ,IsAutoCloseOn
        ,IsAutoShrink
        ,IsStandBy
        ,DatabaseCreatedDate
        ,InstanceId
        ,RecordCreatedDate)
VALUES (SOURCE.InstanceDatabaseId
           ,SOURCE.DatabaseName
           ,SOURCE.CompatibilityLevel
           ,SOURCE.Collation
           ,SOURCE.UserAccess
           ,SOURCE.RecoveryModel
           ,SOURCE.State
           ,SOURCE.LogReuseWait
           ,SOURCE.IsReadOnly
           ,SOURCE.IsAutoCloseOn
           ,SOURCE.IsAutoShrink
           ,SOURCE.IsStandBy
           ,SOURCE.DatabaseCreatedDate
           ,SOURCE.InstanceId
           ,SOURCE.RecordCreatedDate)

WHEN NOT MATCHED BY SOURCE THEN
DELETE;
END
```

- **monitor.spMergetHost:** Carga la tabla monitor.tHost, en donde se cargan todos los datos relacionados con el servidor donde está instalado SQL Server.

```
CREATE PROCEDURE [monitor].[spMergetHost]
AS
BEGIN

MERGE [monitor].[tHost] AS TARGET
USING [staging].[tHost] AS SOURCE
ON TARGET.HostId = SOURCE.HostId
WHEN MATCHED AND (TARGET.HostName<> SOURCE.HostName OR TARGET.HostName IS NULL)
                        OR (TARGET.WindowsEdition<> SOURCE.WindowsEdition OR
TARGET.WindowsEdition IS NULL)
                        OR (TARGET.WindowsVersion<> SOURCE.WindowsVersion OR
TARGET.WindowsVersion IS NULL)
                        OR (TARGET.PhysicalCores<> SOURCE.PhysicalCores OR
TARGET.PhysicalCores IS NULL)
                        OR (TARGET.LogicalCores<> SOURCE.LogicalCores OR
TARGET.LogicalCores IS NULL)
                        OR (TARGET.TotalMemoryGB<> SOURCE.TotalMemoryGB OR
TARGET.TotalMemoryGB IS NULL)

THEN UPDATE SET TARGET.HostName = SOURCE.HostName
                         ,TARGET.WindowsEdition = SOURCE.WindowsEdition
                         ,TARGET.WindowsVersion = SOURCE.WindowsVersion
                         ,TARGET.PhysicalCores = SOURCE.PhysicalCores
```

```
                        ,TARGET.LogicalCores = SOURCE.LogicalCores
                        ,TARGET.TotalMemoryGB = SOURCE.TotalMemoryGB
                        ,TARGET.RecordUpdatedDate = GETDATE()

WHEN NOT MATCHED BY TARGET THEN
INSERT (HostId
           ,HostName
           ,Environment
       ,WindowsEdition
       ,WindowsVersion
       ,PhysicalCores
       ,LogicalCores
       ,TotalMemoryGB
           ,RecordCreatedDate)
VALUES (SOURCE.HostId
           ,SOURCE.HostName
           ,SOURCE.Environment
       ,SOURCE.WindowsEditiona
       ,SOURCE.WindowsVersion
       ,SOURCE.PhysicalCores
       ,SOURCE.LogicalCores
       ,SOURCE.TotalMemoryGB
           ,SOURCE.RecordCreatedDate)
WHEN NOT MATCHED BY SOURCE THEN
DELETE;
END
```

- **monitor.spMergetInstance:** Carga la tabla monitor.tInstance, con los datos relacionados a las instancias de SQL Server.

```
CREATE PROCEDURE [monitor].[spMergetInstance]
AS
BEGIN

MERGE [monitor].[tInstance] AS TARGET
USING [staging].[tInstance] AS SOURCE
ON TARGET.InstanceId = SOURCE.InstanceId
AND TARGET.HostId = SOURCE.HostId
WHEN MATCHED AND (TARGET.InstanceName<> SOURCE.InstanceName OR TARGET.InstanceName IS
NULL)
             OR (TARGET.Collation<> SOURCE.Collation OR TARGET.Collation IS NULL)
             OR (TARGET.TcpPort<> SOURCE.TcpPort OR TARGET.TcpPort IS NULL)
             OR (TARGET.Version<> SOURCE.Version OR TARGET.Version IS NULL)
             OR (TARGET.Edition<> SOURCE.Edition OR TARGET.Edition IS NULL)
             OR (TARGET.Environment<> SOURCE.Environment OR TARGET.Environment IS
NULL)
             OR (TARGET.DefaultDataFilePath<> SOURCE.DefaultDataFilePath OR
TARGET.DefaultDataFilePath IS NULL)
            OR (TARGET.DefaultLogPath<> SOURCE.DefaultLogPath OR
TARGET.DefaultLogPath IS NULL)
             OR (TARGET.IsAlwaysOnAGEnable<> SOURCE.IsAlwaysOnAGEnable OR
TARGET.IsAlwaysOnAGEnable IS NULL)
             OR (TARGET.AuthenticationMode<> SOURCE.AuthenticationMode OR
TARGET.AuthenticationMode IS NULL)

THEN UPDATE SET TARGET.InstanceName = SOURCE.InstanceName
                        ,TARGET.Collation = SOURCE.Collation
                        ,TARGET.TcpPort = SOURCE.TcpPort
                        ,TARGET.Version = SOURCE.Version
```

```
                        ,TARGET.Edition = SOURCE.Edition
                        ,TARGET.Environment = SOURCE.Environment
                        ,TARGET.DefaultDataFilePath = SOURCE.DefaultDataFilePath
                        ,TARGET.DefaultLogPath = SOURCE.DefaultLogPath
                        ,TARGET.IsAlwaysOnAGEnable = SOURCE.IsAlwaysOnAGEnable
                        ,TARGET.AuthenticationMode = SOURCE.AuthenticationMode
                        ,TARGET.RecordUpdatedDate = GETDATE()

WHEN NOT MATCHED BY TARGET THEN
INSERT (InstanceId
          ,InstanceName
       ,Collation
       ,TcpPort
       ,Version
       ,Edition
       ,Environment
       ,DefaultDataFilePath
       ,DefaultLogPath
       ,IsAlwaysOnAGEnable
       ,AuthenticationMode
          ,HostId
       ,RecordCreatedDate)
VALUES (SOURCE.InstanceId
          ,SOURCE.InstanceName
       ,SOURCE.Collation
       ,SOURCE.TcpPort
       ,SOURCE.Version
       ,SOURCE.Edition
       ,SOURCE.Environment
       ,SOURCE.DefaultDataFilePath
       ,SOURCE.DefaultLogPath
       ,SOURCE.IsAlwaysOnAGEnable
       ,SOURCE.AuthenticationMode
          ,SOURCE.HostId
       ,SOURCE.RecordCreatedDate)

WHEN NOT MATCHED BY SOURCE THEN
DELETE;
END
```

- **monitor.spMergetInstanceConfiguration:** Carga la tabla monitor.tInstanceConfiguration, con las diferentes configuraciones avanzadas de las instancias SQL Server.

```
CREATE PROCEDURE [monitor].[spMergetInstanceConfiguration]
AS
BEGIN

MERGE [monitor].[tInstanceConfiguration] AS TARGET
USING [staging].[tInstanceConfiguration] AS SOURCE
ON TARGET.ConfigurationId = SOURCE.ConfigurationId
AND TARGET.InstanceId = SOURCE.InstanceId
WHEN MATCHED AND (TARGET.ConfigurationName<> SOURCE.ConfigurationName OR
TARGET.ConfigurationName IS NULL)
```

```
                    OR (TARGET.ConfigurationDescription<>
            SOURCE.ConfigurationDescription OR TARGET.ConfigurationDescription IS
            NULL)
                    OR (TARGET.Value<> SOURCE.Value OR TARGET.Value IS NULL)

THEN UPDATE SET TARGET.ConfigurationName = SOURCE.ConfigurationName
                     ,TARGET.ConfigurationDescription =
SOURCE.ConfigurationDescription
                     ,TARGET.Value = SOURCE.Value
                     ,TARGET.RecordUpdatedDate = GETDATE()

WHEN NOT MATCHED BY TARGET THEN

INSERT (ConfigurationId
          ,ConfigurationName
          ,ConfigurationDescription
          ,Value
          ,InstanceId
          ,RecordCreatedDate)
VALUES (ConfigurationId
          ,SOURCE.ConfigurationName
          ,SOURCE.ConfigurationDescription
          ,SOURCE.Value
          ,SOURCE.InstanceId
          ,SOURCE.RecordCreatedDate)

WHEN NOT MATCHED BY SOURCE THEN
DELETE;
END
```

- **monitor.spMergetAOInstance:** Carga la tabla monitor.tAOInstance, donde cargó la información relacionada al estado de las instancias incluidas en ambiente de “Always On”.

```
CREATE PROCEDURE [monitor].[spMergetAOInstance]
AS
BEGIN

MERGE [monitor].[tAOInstance] AS TARGET
USING [staging].[tAOInstance] AS SOURCE
ON TARGET.ReplicaId = SOURCE.ReplicaId
AND TARGET.GroupId = SOURCE.GroupId
WHEN MATCHED AND (TARGET.AvailabilityGroupName<> SOURCE.AvailabilityGroupName OR
TARGET.AvailabilityGroupName IS NULL)
                    OR (TARGET.Role<> SOURCE.Role OR TARGET.Role IS NULL)
                    OR (TARGET.AvailabilityMode<> SOURCE.AvailabilityMode OR
TARGET.AvailabilityMode IS NULL)
                    OR (TARGET.FailoverMode<> SOURCE.FailoverMode OR
TARGET.FailoverMode IS NULL)
                    OR (TARGET.ConnectedState<> SOURCE.ConnectedState OR
TARGET.ConnectedState IS NULL)
                    OR (TARGET.RecoveryHealth<> SOURCE.RecoveryHealth OR
TARGET.RecoveryHealth IS NULL)
                    OR (TARGET.SynchronizationHealth<>
SOURCE.SynchronizationHealth OR TARGET.SynchronizationHealth IS NULL)
                    OR (TARGET.LastConnectErrorNumber<>
SOURCE.LastConnectErrorNumber OR TARGET.LastConnectErrorNumber IS NULL)
```

```
                        OR (TARGET.LastConnectErrorDescription<>
SOURCE.LastConnectErrorDescription OR TARGET.LastConnectErrorDescription IS NULL)
                        OR (TARGET.LastConnectErrorTimeStamp<>
SOURCE.LastConnectErrorTimeStamp OR TARGET.LastConnectErrorTimeStamp IS NULL)

THEN UPDATE SET TARGET.AvailabilityGroupName = SOURCE.AvailabilityGroupName
                        ,TARGET.Role = SOURCE.Role
                        ,TARGET.AvailabilityMode = SOURCE.AvailabilityMode
                        ,TARGET.FailoverMode = SOURCE.FailoverMode
                        ,TARGET.ConnectedState = SOURCE.ConnectedState
                        ,TARGET.RecoveryHealth = SOURCE.RecoveryHealth
                        ,TARGET.SynchronizationHealth = SOURCE.SynchronizationHealth
                        ,TARGET.LastConnectErrorNumber =
SOURCE.LastConnectErrorNumber
                        ,TARGET.LastConnectErrorDescription =
SOURCE.LastConnectErrorDescription
                        ,TARGET.LastConnectErrorTimeStamp =
SOURCE.LastConnectErrorTimeStamp
                        ,TARGET.RecordUpdatedDate = GETDATE()
WHEN NOT MATCHED BY TARGET THEN
INSERT (ReplicaId
           ,GroupId
       ,AvailabilityGroupName
       ,Role
       ,AvailabilityMode
       ,FailoverMode
       ,ConnectedState
       ,RecoveryHealth
       ,SynchronizationHealth
       ,LastConnectErrorNumber
       ,LastConnectErrorDescription
       ,LastConnectErrorTimeStamp
       ,InstanceId
       ,RecordCreatedDate)
VALUES (SOURCE.ReplicaId
           ,SOURCE.GroupId
           ,SOURCE.AvailabilityGroupName
           ,SOURCE.Role
           ,SOURCE.AvailabilityMode
           ,SOURCE.FailoverMode
           ,SOURCE.ConnectedState
           ,SOURCE.RecoveryHealth
           ,SOURCE.SynchronizationHealth
           ,SOURCE.LastConnectErrorNumber
           ,SOURCE.LastConnectErrorDescription
           ,SOURCE.LastConnectErrorTimeStamp
           ,SOURCE.InstanceId
           ,GETDATE());
WHEN NOT MATCHED BY SOURCE THEN
DELETE
END
```

- **Monitor.spMergetAODatabase:** Carga la tabla monitor.tAODatabase, con los datos relacionados con el estado de las bases de datos que se encuentran dentro de un grupo de disponibilidad.

```
CREATE PROCEDURE [monitor].[spMergetAODatabase]
AS
BEGIN

MERGE [monitor].[tAODatabase] AS TARGET
USING [staging].[tAODatabase] AS SOURCE
ON TARGET.ReplicaId = SOURCE.ReplicaId
AND TARGET.GroupId = SOURCE.GroupId
AND TARGET.GroupDatabaseId = SOURCE.GroupDatabaseId
WHEN MATCHED AND (TARGET.AvailabilityGroupName<> SOURCE.AvailabilityGroupName OR
TARGET.AvailabilityGroupName IS NULL)
                        OR (TARGET.DatabaseName<> SOURCE.DatabaseName OR
TARGET.DatabaseName IS NULL)
                        OR (TARGET.IsPrimaryReplica<> SOURCE.IsPrimaryReplica OR
TARGET.IsPrimaryReplica IS NULL)
                        OR (TARGET.IsFailoverReady<> SOURCE.IsFailoverReady OR
TARGET.IsFailoverReady IS NULL)
                        OR (TARGET.IsDatabaseJoined<> SOURCE.IsDatabaseJoined OR
TARGET.IsDatabaseJoined IS NULL)
                        OR (TARGET.SynchronizationState<>
SOURCE.SynchronizationState OR TARGET.SynchronizationState IS NULL)
                        OR (TARGET.SynchronizationStateHealth<>
SOURCE.SynchronizationStateHealth OR TARGET.SynchronizationStateHealth IS NULL)
                        OR (TARGET.DatabaseState<> SOURCE.DatabaseState OR
TARGET.DatabaseState IS NULL)
                        OR (TARGET.IsSuspended<> SOURCE.IsSuspended OR
TARGET.IsSuspended IS NULL)
                        OR (TARGET.SuspenedReason<> SOURCE.SuspenedReason OR
TARGET.SuspenedReason IS NULL)
                        OR (TARGET.LastHardenedLSN<> SOURCE.LastHardenedLSN OR
TARGET.LastHardenedLSN IS NULL)
                        OR (TARGET.LastHardenedTime<> SOURCE.LastHardenedTime OR
TARGET.LastHardenedTime IS NULL)
                        OR (TARGET.LastCommitLSN<> SOURCE.LastCommitLSN OR
TARGET.LastCommitLSN IS NULL)
                        OR (TARGET.LastCommitTime<> SOURCE.LastCommitTime OR
TARGET.LastCommitTime IS NULL)
                        OR (TARGET.SecondsBehindPrimary<>
SOURCE.SecondsBehindPrimary OR TARGET.SecondsBehindPrimary IS NULL)

THEN UPDATE SET TARGET.AvailabilityGroupName = SOURCE.AvailabilityGroupName
                         ,TARGET.DatabaseName = SOURCE.DatabaseName
                         ,TARGET.IsPrimaryReplica = SOURCE.IsPrimaryReplica
                         ,TARGET.IsFailoverReady = SOURCE.IsFailoverReady
                         ,TARGET.IsDatabaseJoined = SOURCE.IsDatabaseJoined
                         ,TARGET.SynchronizationState =
SOURCE.SynchronizationState
                         ,TARGET.SynchronizationStateHealth =
SOURCE.SynchronizationStateHealth
                         ,TARGET.DatabaseState = SOURCE.DatabaseState
                         ,TARGET.IsSuspended = SOURCE.IsSuspended
                         ,TARGET.SuspenedReason = SOURCE.SuspenedReason
                         ,TARGET.LastHardenedLSN = SOURCE.LastHardenedLSN
                         ,TARGET.LastHardenedTime = SOURCE.LastHardenedTime
                         ,TARGET.LastCommitLSN = SOURCE.LastCommitLSN
                         ,TARGET.LastCommitTime = SOURCE.LastCommitTime
```

```
                        ,TARGET.SecondsBehindPrimary =
SOURCE.SecondsBehindPrimary
                        ,TARGET.RecordUpdatedDate = GETDATE()

WHEN NOT MATCHED BY TARGET THEN
INSERT (ReplicaId
           ,GroupId
           ,GroupDatabaseId
           ,AvailabilityGroupName
           ,DatabaseName
           ,IsPrimaryReplica
           ,IsFailoverReady
           ,IsDatabaseJoined
           ,SynchronizationState
           ,SynchronizationStateHealth
           ,DatabaseState
           ,IsSuspended
           ,SuspenedReason
           ,LastHardenedLSN
           ,LastHardenedTime
           ,LastCommitLSN
           ,LastCommitTime
           ,SecondsBehindPrimary
           ,RecordCreatedDate)
VALUES (SOURCE.ReplicaId
           ,SOURCE.GroupId
           ,SOURCE.GroupDatabaseId
           ,SOURCE.AvailabilityGroupName
           ,SOURCE.DatabaseName
           ,SOURCE.IsPrimaryReplica
           ,SOURCE.IsFailoverReady
           ,SOURCE.IsDatabaseJoined
           ,SOURCE.SynchronizationState
           ,SOURCE.SynchronizationStateHealth
           ,SOURCE.DatabaseState
           ,SOURCE.IsSuspended
           ,SOURCE.SuspenedReason
           ,SOURCE.LastHardenedLSN
           ,SOURCE.LastHardenedTime
           ,SOURCE.LastCommitLSN
           ,SOURCE.LastCommitTime
           ,SOURCE.SecondsBehindPrimary
           ,GETDATE());
END
```

5.5.2. Vistas

Para facilitar las consultas y la transformación de los datos se creó una serie de vistas para proporcionar la consulta de los reportes y las notificaciones. Por lo cual se crearon dos tipos de vistas, unas específicamente utilizadas para los reportes y otras para las notificaciones, las cuales se dividen de la siguiente manera:

5.5.2.1. Vistas para Reportes

Esta lista de vistas fueron las encargadas de alimentar los diferentes reportes informativos para los administradores de bases de datos o los encargados de los diferentes negocios de una empresa:

- **monitor.vReportAODatabase:** Vista que extrajo los datos finales relacionados con el estado de las bases de datos incluidas en una configuración de “Always On”.

```
CREATE VIEW [monitor].[vReportAODatabase] AS
SELECT d.ReplicaId
         ,d.GroupId
         ,d.GroupDatabaseId
         ,i.InstanceId
         ,i.InstanceName
         ,d.AvailabilityGroupName
         ,d.DatabaseName
         ,d.IsPrimaryReplica
         ,d.IsFailoverReady
         ,d.IsDatabaseJoined
         ,d.SynchronizationState
         ,d.SynchronizationStateHealth
         ,d.DatabaseState
         ,d.IsSuspended
         ,d.SuspenedReason
         ,d.LastHardenedLSN
         ,d.LastHardenedTime
         ,d.LastCommitLSN
         ,d.LastCommitTime
         ,d.SecondsBehindPrimary
         ,d.RecordCreatedDate
         ,d.RecordUpdatedDate
FROM [monitor].[tAODatabase] d
INNER JOIN [monitor].[tAOInstance] a ON a.ReplicaId = d.ReplicaId
INNER JOIN [monitor].[tInstance] i ON i.InstanceId = a.InstanceId
GO
```

- **monitor.vReportAOInstance:** Vista que extrajo los datos finales relacionados con el estado de las instancias incluidas en una configuración de “Always On”.

```
CREATE VIEW [monitor].[vReportAOInstance] As
SELECT a.ReplicaId
         ,a.GroupId
         ,a.InstanceId
         ,i.InstanceName
         ,a.AvailabilityGroupName
         ,a.Role
         ,a.AvailabilityMode
         ,a.FailoverMode
         ,a.ConnectedState
         ,a.RecoveryHealth
         ,a.SynchronizationHealth
         ,a.LastConnectErrorNumber
         ,a.LastConnectErrorDescription
         ,a.LastConnectErrorTimeStamp
         ,a.RecordCreatedDate
         ,a.RecordUpdatedDate
FROM [monitor].[tAOInstance] a
INNER JOIN [monitor].[tInstance] i ON i.InstanceId = a.InstanceId
GO
```

- **monitor.vReportBackupHistoty:** Vista que extrajo los datos finales relacionados con los respaldos de una instancia de SQL Server en los últimos 30 días.

```
CREATE   VIEW [monitor].[vReportBackupHistoty] AS
SELECT b.BackupHistoryId
         ,i.InstanceId
         ,d.DatabaseId
         ,i.InstanceName
         ,d.DatabaseName
         ,b.LoginName
         ,b.BackupType
         ,b.RecoveryModel
         ,b.CompatibilityLevel
         ,b.Collation
         ,b.IsCopyOnly
         ,b.DeviceType
         ,b.PhysicalDeviceName
         ,b.BackupSizeMB
         ,b.TimeTakenMin
         ,b.BackupStartDate
         ,b.BackupEndDate
         ,b.RecordCreatedDate
         ,b.RecordUpdatedDate
FROM [monitor].[tBackupHistory] b
INNER JOIN [monitor].[tDatabase] d ON d.DatabaseId = b.DatabaseId
INNER JOIN [monitor].[tInstance] i ON i.InstanceId = d.InstanceId
WHERE b.BackupEndDate > 30
GO
```

- **monitor.vReportDatabaseFiles:** Vista que extrajo los datos finales relacionados con los archivos pertenecientes a una base de datos SQL Server.

```
CREATE   VIEW [monitor].[vReportDatabaseFiles] AS
SELECT f.DatabaseFileId
         ,d.DatabaseId
         ,d.DatabaseName
         ,f.TypeDescription
         ,f.LogicaFilelName
         ,f.PhysicalFileName
         ,f.FilePath
         ,f.FileState
         ,f.FileSizeMB
         ,f.RecordCreatedDate
         ,f.RecordUpdatedDate
FROM [monitor].[tDatabaseFiles] f
INNER JOIN [monitor].[tDatabase] d ON d.DatabaseID = f.DatabaseId
GO
```

- **monitor.vReportDatabases:** Vista que extrajo los datos finales relacionados con las bases de datos almacenadas dentro de una instancia SQL Server.

```
CREATE VIEW [monitor].[vReportDatabases] AS
SELECT d.DatabaseId
         ,i.InstanceId
         ,i.InstanceName
         ,d.InstanceDatabaseId
         ,d.DatabaseName
         ,d.CompatibilityLevel
         ,d.Collation
         ,d.UserAccess
         ,d.RecoveryModel
         ,d.State
         ,d.LogReuseWait
         ,d.IsReadOnly
         ,d.IsAutoCloseOn
         ,d.IsAutoShrink
         ,d.IsStandBy
         ,d.DatabaseCreatedDate
         ,d.RecordCreatedDate
         ,d.RecordUpdatedDate
FROM [monitor].[tDatabase] d
INNER JOIN [monitor].[tInstance] i  ON i.InstanceId = d.InstanceId
GO
```

- **monitor.vReportDriveSize:** Vista que extrajo los datos finales relacionados con el espacio disponible y utilizado dentro de los diferentes servidores.

```
CREATE   VIEW [monitor].[vReportDriveSize] AS
SELECT d.DriveSizeId
         ,h.HostId
         ,h.HostName
         ,d.DriveLetter
         ,d.DriveName
         ,d.FreeSpaceGB
         ,d.FreeSpacePercentage
         ,d.UsedSpaceGB
         ,d.UsedSpacePercentage
         ,d.TotalSizeGB
         ,d.RecordCreatedDate
         ,d.RecordUpdatedDate
FROM [monitor].[tDriveSize] d
INNER JOIN [monitor].[tHost] h ON h.HostId = d.HostId
GO
```

- **monitor.vReportHost:** Vista que extrajo los datos finales relacionados al host donde fue instalado SQL Server.

```
CREATE   VIEW [monitor].[vReportHost] AS
SELECT h.HostId
         ,i.InstanceId
         ,i.InstanceName
         ,h.HostName
         ,h.Environment
         ,h.WindowsEdition
         ,h.WindowsVersion
         ,h.PhysicalCores
         ,h.LogicalCores
         ,h.TotalMemoryGB
         ,h.RecordCreatedDate
         ,h.RecordUpdatedDate
FROM [monitor].[tHost] h
INNER JOIN [monitor].[tInstance] i ON i.HostId = h.HostId
GO
```

- **monitor.vReportInstance:** Vista que extrajo los datos finales relacionados con la instancia de SQL Server.

```
CREATE   VIEW [monitor].[vReportInstance] AS
SELECT i.InstanceId
         ,i.InstanceName
         ,i.InstanceName + ', '+ i.TcpPort AS ConnectionString
         ,i.HostId
         ,h.HostName
         ,i.Collation
         ,i.TcpPort
         ,i.Version
         ,i.Edition
         ,i.Environment
         ,i.DefaultDataFilePath
         ,i.DefaultLogPath
         ,i.IsAlwaysOnAGEnable
         ,i.AuthenticationMode
         ,i.RecordCreatedDate
         ,i.RecordUpdatedDate
FROM [monitor].[tInstance] i
INNER JOIN [monitor].[tHost] h ON h.HostId = i.HostId
GO
```

- **monitor.vReportInstanceConfiguration:** Vista que extrajo los datos finales relacionados con la configuración avanzada perteneciente a una instancia de SQL Server.

```
CREATE VIEW [monitor].[vReportInstanceConfiguration] AS
SELECT c.InstanceConfigurationId
         ,i.InstanceName
         ,c.ConfigurationId
         ,c.ConfigurationName
         ,c.ConfigurationDescription
         ,c.Value
         ,c.RecordCreatedDate
FROM [monitor].[tInstanceConfiguration] c
INNER JOIN [monitor].[tInstance] i ON i.InstanceId = c.InstanceId
GO
```

- **monitor.vReportInstanceServices:** Vista que extrajo los datos finales relacionados con la lista de servicios pertenecientes a cada instancia de SQL Server.

```
CREATE VIEW [monitor].[vReportInstanceServices] AS
SELECT s.SQLServiceId
         ,h.HostId
         ,i.InstanceId
         ,h.HostName
         ,i.InstanceName
         ,s.ServiceName
         ,s.StartMode
         ,s.Status
         ,s.RecordCreatedDate
         ,s.RecordUpdatedDate
FROM [monitor].[tSQLServiceStatus] s
INNER JOIN [monitor].[tHost] h ON h.HostId = s.HostId
INNER JOIN [monitor].[tInstance] i ON i.HostId = s.HostId
WHERE s.ServiceName LIKE '%MsDtsServer%'
OR i.InstanceName LIKE '%\%' AND s.ServiceName LIKE '%$%'
OR i.InstanceName NOT LIKE '%\%' AND s.ServiceName NOT LIKE '%$%'
GO
```

- **monitor.vReportJobStatus:** Vista que extrajo los datos finales relacionados al último estado de ejecución de los “jobs” pertenecientes a una instancia de SQL Server.

```
CREATE   VIEW [monitor].[vReportJobStatus] AS
SELECT j.JobStatusId
         ,j.InstanceId
         ,i.InstanceName
         ,j.JobId
         ,j.JobName
         ,j.JobOwner
         ,j.IsEnabled
         ,j.StepName
         ,j.Message
         ,j.RunStatus
         ,j.LastRunDurationSeconds
         ,j.LastRunDuration
         ,j.LastRunDate
         ,j.JobDateCreated
         ,j.JobDateModified
         ,j.RecordCreatedDate
         ,j.RecordUpdatedDate
FROM [monitor].[tJobStatus] j
INNER JOIN [monitor].[tInstance] i ON i.InstanceId = j.InstanceId
GO
```

- **monitor.vReportMissingIndex:** Vista que extrajo los datos finales relacionados con los posibles índices faltantes dentro de una base de datos.

```
CREATE     VIEW [monitor].[vReportMissingIndex] AS
SELECT m.MissingIndexId
         ,d.DatabaseId
         ,i.InstanceId
         ,i.InstanceName
         ,d.DatabaseName
         ,m.ObjectName
         ,m.FullyQualifiedObjectName
         ,m.IndexAdvantage
         ,m.EqualityColumns
         ,m.InEqualityColumns
         ,m.IncludedColumns
         ,m.UniqueCompiles
         ,m.UserSeeks
         ,m.UserScans
         ,m.LastUserSeekTime
         ,m.LastUserScanTime
         ,m.AvgTotalUserCost
         ,m.AvgUserImpact
         ,m.ProposedIndex
         ,m.RecordCreatedDate
         ,m.RecordUpdatedDate
FROM [monitor].[tMissingIndex] m
INNER JOIN [monitor].[tDatabase] d ON d.DatabaseId = m.DatabaseId
INNER JOIN [monitor].[tInstance] i ON i.InstanceId = d.InstanceId
GO
```

- **monitor.vReportPrivAccess:** Vista que extrajo los datos finales relacionados con los usuarios que tienen altos privilegios dentro de la instancia de SQL Server.

```
CREATE VIEW [monitor].[vReportPrivAccess] AS
SELECT i.InstanceId
         ,i.InstanceName
         ,p.LoginName
         ,p.LoginType
         ,p.IsSysadmin
         ,p.IsServeradmin
         ,p.IsSecurityadmin
         ,p.IsProcessadmin
         ,p.IsDisabled
         ,p.AccountCreatedDate
         ,p.AccountUpdatedDate
         ,p.RecordCreatedDate
FROM [monitor].[tPrivAccess] p
INNER JOIN [monitor].[tInstance] i ON i.InstanceId = p.InstanceId
GO
```

- **monitor.vReportRestoreHistory:** Vista que extrajo los datos finales relacionados con las restauraciones de bases de datos realizadas los últimos 30 días.

```
CREATE     VIEW [monitor].[vReportRestoreHistory] AS
SELECT r.RestoreHistoryId
         ,i.InstanceId
         ,d.DatabaseId
         ,i.InstanceName
         ,d.DatabaseName
         ,r.LoginName
         ,r.RestoreType
         ,r.Replace
         ,r.Recovery
         ,r.BackupFileLocation
         ,r.DestinationDataFile
         ,r.DestinationLogFile
         ,r.RestoreDate
         ,r.RecordCreatedDate
         ,r.RecordUpdatedDate
FROM [monitor].[tRestoreHistory] r
INNER JOIN [monitor].[tDatabase] d ON d.DatabaseID = r.DatabaseId
INNER JOIN [monitor].[tInstance] i ON i.InstanceId = d.InstanceId
WHERE RestoreDate > GETDATE() - 30
GO
```

- **monitor.vReportSQLServiceStatus:** Vista que extrajo los datos finales relacionados con el estado de los servicios ejecutados por SQL Server

```
CREATE VIEW [monitor].[vReportSQLServiceStatus] AS
SELECT s.SQLServiceId
         ,h.HostName
         ,s.ServiceName
         ,s.StartMode
         ,s.Status
         ,s.RecordCreatedDate
         ,s.RecordUpdatedDate
FROM [monitor].[tSQLServiceStatus] s
INNER JOIN [monitor].[tHost] h ON h.HostId = s.HostId
GO
```

5.5.2.2. Vistas para Notificaciones

Estas son las vistas de donde se extrajeron los datos, las cuales alimentaron las notificaciones del proceso de alerta. Estas vistas filtran y convierten datos según sea necesario.

- **monitor.vNotifyAODatabase:** Vista que seleccionaba los datos para crear la estructura de correo para los datos relacionados con el estado de las bases de datos que están incluidas dentro de una configuración “Always On”.

```
CREATE VIEW [monitor].[vNotifyAODatabase]
AS
SELECT m.ReplicaId
         ,m.GroupId
         ,m.GroupDatabaseId
         ,i.InstanceName
         ,m.AvailabilityGroupName
         ,m.DatabaseName
         ,m.IsPrimaryReplica
         ,m.IsFailoverReady
         ,m.IsDatabaseJoined
         ,s.SynchronizationState
         ,s.SynchronizationStateHealth
         ,s.DatabaseState
         ,COALESCE(m.IsSuspended, 'No Data') AS IsSuspended
         ,COALESCE(m.SuspenedReason, 'No Data') AS SuspenedReason
         ,COALESCE(m.LastHardenedLSN, 0) AS LastHardenedLSN
         ,COALESCE(m.LastHardenedTime, '1900-01-01 00:00:00') AS LastHardenedTime
         ,COALESCE(m.LastCommitLSN, 0) AS LastCommitLSN
         ,COALESCE(m.LastCommitTime, '1900-01-01 00:00:00') AS LastCommitTime
         ,COALESCE(m.SecondsBehindPrimary, 0)  AS SecondsBehindPrimary
         ,DENSE_RANK() OVER(ORDER BY m.GroupDatabaseId) AS R
FROM [monitor].[tAODatabase] m
INNER JOIN [staging].[tAODatabase] s ON s.ReplicaId = m.ReplicaId AND s.GroupId =
m.GroupId AND s.GroupDatabaseId = m.GroupDatabaseId
INNER JOIN [monitor].[tAOInstance] a ON a.ReplicaId = m.ReplicaId
INNER JOIN [monitor].[tInstance] i ON i.InstanceId = a.InstanceId
WHERE (s.SynchronizationState != 'SYNCHRONIZED' AND s.SynchronizationState !=
'SYNCHRONIZING' OR s.SynchronizationStateHealth != 'HEALTHY' OR s.DatabaseState
!= 'ONLINE')
AND (s.SynchronizationState != m.SynchronizationState OR
s.SynchronizationStateHealth != m.SynchronizationStateHealth OR s.DatabaseState
!= m.DatabaseState);
GO
```

- **monitor.vNotifyAOInstance:** Vista que seleccionaba los datos para crear la estructura de correo para los datos relacionados con el estado de las instancias que están incluidas dentro de una configuración “Always On”.

```
CREATE VIEW [monitor].[vNotifyAOInstance] AS
SELECT m.ReplicaId
         ,m.GroupId
         ,m.AvailabilityGroupName
         ,m.Role
         ,m.AvailabilityMode
         ,m.FailoverMode
         ,s.ConnectedState
         ,s.RecoveryHealth
         ,s.SynchronizationHealth
         ,COALESCE(m.LastConnectErrorNumber,0) AS LastConnectErrorNumber
         ,COALESCE(m.LastConnectErrorDescription,'No Data') AS
LastConnectErrorDescription
         ,COALESCE(m.LastConnectErrorTimeStamp,'1900-01-01 00:00:00') AS
LastConnectErrorTimeStamp
         ,i.InstanceName
         ,DENSE_RANK() OVER(ORDER BY m.GroupId) AS R
FROM [monitor].[tAOInstance] m
INNER JOIN [staging].[tAOInstance] s ON s.ReplicaId = m.ReplicaId
INNER JOIN [monitor].[tInstance] i ON i.InstanceId = m.InstanceId
WHERE (s.ConnectedState != 'CONNECTED' OR s.RecoveryHealth != 'ONLINE' OR
s.SynchronizationHealth != 'HEALTHY')
AND (s.ConnectedState != m.ConnectedState OR s.RecoveryHealth != m.RecoveryHealth
OR s.SynchronizationHealth != m.SynchronizationHealth);
GO
```

- **monitor.vNotifyDriveSize:** Vista que seleccionaba la información para crear la estructura de correo para los datos relacionados con el espacio en disco, solo se muestran los datos cuando el uso del disco fuera mayor a 90%.

```
CREATE VIEW [monitor].[vNotifyDriveSize] AS
SELECT d.DriveSizeId
         ,h.HostName
         ,d.DriveLetter
         ,d.DriveName
         ,d.FreeSpaceGB
         ,d.FreeSpacePercentage
         ,d.UsedSpaceGB
         ,d.TotalSizeGB
         ,ROW_NUMBER() OVER(ORDER BY d.HostId) AS RN
FROM [monitor].[tDriveSize] d
INNER JOIN [monitor].[tHost] h ON h.HostId = d.HostId
WHERE UsedSpacePercentage > 90;
GO
```

- **monitor.vNotifyInstanceConfiguration:** Vista que seleccionaba los datos para crear la estructura de correo para los datos relacionados con las configuraciones avanzadas dentro de una instancia de SQL Server.

```
CREATE VIEW [monitor].[vNotifyInstanceConfiguration] AS
SELECT *
FROM
(SELECT i.InstanceName
          ,c.ConfigurationName
       ,c.Value

FROM [monitor].[tInstanceConfiguration] c
INNER JOIN [monitor].[tInstance] i ON i.InstanceId = c.InstanceId ) AS t
PIVOT (MAX(Value) FOR ConfigurationName IN ([show advanced options]
              ,[remote query timeout (s)]
              ,[cost threshold for parallelism]
              ,[max degree of parallelism]
              min memory per query (KB)]
              ,[min server memory (MB)]
              ,[max server memory (MB)]
              ,[xp_cmdshell])
) AS PIV

GO
```

- **monitor.vNotifySQLServiceStatus** Vista que seleccionaba los datos para crear la estructura de correo para los datos relacionados con el estado de los servicios de SQL Server, siempre y cuando un estado fuera diferente a “Running”.

```
CREATE VIEW [monitor].[vNotifySQLServiceStatus]
AS

SELECT m.SQLServiceId
         ,m.ServiceName
         ,m.StartMode
         ,s.Status
         ,h.HostName
         ,ROW_NUMBER() OVER(ORDER BY m.SQLServiceId) AS RN
FROM [monitor].[tSQLServiceStatus] m
INNER JOIN [staging].[tSQLServiceStatus] s ON s.SQLServiceId = m.SQLServiceId
INNER JOIN [monitor].[tHost] h ON h.HostId = m.HostId
AND s.Status != 'Running'
AND s.Status != m.Status;
GO
```

5.5.3. ETL – PowerShell

Powershell fue necesario para extraer la información relacionada con el host en donde SQL Server está instalado, PowerShell ofreció mayor flexibilidad que SQL Server para completar dicha tarea, de manera que se crearon los siguientes procesos para la extracción, transformación y carga de datos.

Los procesos que se describen a continuación fueron estandarizados de manera que todos funcionen de una forma similar.

El proceso consta de los siguientes pasos para cargar los datos en las tablas que pertenecen al esquema “staging”.

1) Truncar las tablas “staging”.
2) Extraer lista de servidores donde el script de PowerShell debe ser ejecutado.
3) Los comandos que extraen la información directamente desde el servidor, esta cambia según el ETL.
4) Cargar variables las cuales fueron asociados en la sección de inserción.
5) Inserción de datos en las tablas “staging” de la base de datos SQLMonitor.
6) Ejecución de procedimiento almacenado para cargar las tablas utilizadas en los reportes finales pertenecientes al esquema monitor; estos procedimientos almacenados son ejecutados en una de las etapas de los “jobs”.

5.5.3.1. Conexión Dinámica

La conexión dinámica es el proceso en el cual se utilizaron las tablas creadas en el punto 5.5.2.1. Ya que los datos que se extraen son relacionados con la información del servidor, solo se utilizó la tabla monitor.tMonitorHost. Este proceso por medio del comando Invoke-Sqlcmd de PowerShell permite realizar una conexión con las instancias de SQL Server. Este comando realiza una extracción de datos para crear una lista de los hosts en donde se tuvo que realizar una conexión para consumir los datos necesarios.

Es importante destacar que esta lista es recorrida en la función ForEach y el valor del host es almacenado en la variable $HostName. Cada “script” que extrae información utilizó la propiedad -ComputerName que permite realizar una conexión

con otro servidor, donde finalmente -ComputerName utiliza el valor almacenado en la variable $HostName.

5.5.3.2. Load_tHost

Este proceso extrae los datos del host en donde se utilizaron funciones de PowerShell que fueron capaces de otorgar los datos relacionados al CPU, sistema operativo y memoria. El proceso utiliza una función ForEach la cual recorre por cada host extrayendo los datos de los diferentes servidores donde SQL Server está instalado y que sean cargados de la tabla staging.tHost.

```
<# ETL - Host Data #>

#Truncate staging table
$tSQLHost = "TRUNCATE TABLE [staging].[tHost]"
Invoke-Sqlcmd -Query $tSQLHost -ServerInstance "SQLMonitor" -Database
"SQLMonitor"

#Extract host list
$sSQLHost = "SELECT HostId
                   ,HostName
                   ,Environment
             FROM [monitor].[tMonitorHost]"
$sSQLHost = Invoke-Sqlcmd -Query $sSQLHost -ServerInstance "SQLMonitor" -
Database "SQLMonitor"

#Execute the process for each host
ForEach ($Server in $sSQLHost){
    $HostId      = $Server.HostId
    $HostName    = $Server.HostName
    $Environment = $Server.Environment
    #Extract CPU information
    $CPUInfo        = Get-WmiObject Win32_Processor       -ComputerName
$HostName #Get CPU Information
    #Extract Operating System information
    $OSInfo         = Get-WmiObject Win32_OperatingSystem -ComputerName
$HostName #Get OS Information
    #Extract memory information
    $TotalMemoryGB  = Get-WmiObject CIM_PhysicalMemory    -ComputerName
$HostName | Measure-Object -Property capacity -Sum | % { [Math]::Round(($_.sum
/ 1GB), 2) }

    $WindowsEdition = $OSInfo.Caption
    $WindowsVersion = $OSInfo.Version

    #Execute the process for each CPU
    Foreach ($CPU in $CPUInfo){
        $PhysicalCores  = $CPU.NumberOfCores
       $LogicalCores   = $CPU.NumberOfLogicalProcessors
    }

    #Insert data into SQLMonitor database
    $iSQLHostInfo = "INSERT INTO [staging].[tHost]
                     VALUES ('$HostId',
                             '$HostName',
                             '$Environment',
                             '$WindowsEdition',
                             '$WindowsVersion',
                             '$PhysicalCores',
                             '$WindowsVersion',
                             '$PhysicalCores',
                             '$LogicalCores',
```

```
'$TotalMemoryGB',
                                GETDATE())"
    Invoke-Sqlcmd -Query $iSQLHostInfo -ServerInstance "SQLMonitor" -Database
"SQLMonitor"
}
```

Una vez extraída la información, esta es cargada en la tabla tHost del esquema “staging”, la cual será procesada seguidamente por una función merge, el cual carga o actualiza la información extraída por medio de la ejecución del procedimiento almacenado monitor.spMergetHost.

Este procedimiento evalúa los datos nuevos que deben ser insertados o actualizados en la tabla monitor.tHost, además, también elimina los datos que no fueron encontrados en la tabla “staging”.

5.5.3.3. Load_tSQLServiceStatus

Para un sistema de monitoreo es de suma importancia saber cuál es el estatus de los diferentes servicios relacionados con SQL Server que son ejecutados dentro de un servidor, por lo cual este “script” de PowerShell es uno de los más importantes, él extrae y carga la información relacionada con el estado de los servicios de SQL Server en la tabla staging.tSQLServicesStatus.

```
<# ETL - SQL Service Status Data #>
#Truncate staging table
$tSQLHost = "TRUNCATE TABLE [staging].[tSQLServiceStatus]"
Invoke-Sqlcmd -Query $tSQLHost -ServerInstance "SQLMonitor" -Database
"SQLMonitor"

#Extract host list
$sSQLHost = "SELECT HostId
                  ,HostName
                  ,Environment
              FROM [monitor].[tMonitorHost]"

$sSQLHost = Invoke-Sqlcmd -Query $sSQLHost -ServerInstance "SQLMonitor" -
Database "SQLMonitor"

#Execute the process for each host
ForEach ($Server in $sSQLHost){
    $HostId      = $Server.HostId
    $HostName    = $Server.HostName

    #Extract SQL Server services inforamtion
    $Engine = Get-WmiObject -Class WIN32_service -ComputerName $HostName |
Where-Object {$_.name -like 'MSSQLServer' -OR $_.name -like 'MSSQL$*' }
    $Agent = Get-WmiObject -Class WIN32_service -ComputerName $HostName |
Where-Object {$_.name -like 'SQLSERVERAGENT' -OR $_.name -like 'SQLAgent$*' }
    $SSAS = Get-WmiObject -Class WIN32_service -ComputerName $HostName |
Where-Object {$_.name -like 'MSSQLServerOLAPService' -OR $_.name -like
'MSOLAP$*' }
```

```
$SSRS = Get-WmiObject -Class WIN32_service -ComputerName $HostName | Where-
Object {$_.name -like 'SQLServerReportingServices' -OR $_.name -like
'ReportServer*' }
    $SSIS = Get-WmiObject -Class WIN32_service -ComputerName $HostName |
Where-Object {$_.name -like "*MsDtsServer*"}

    #Create array variable
    $SQLServiceInfo = @()
    #Add services information into the array
    $SQLServiceInfo += $Engine
    $SQLServiceInfo += $Engine
    $SQLServiceInfo += $Agent
    $SQLServiceInfo += $SSAS
    $SQLServiceInfo += $SSIS
    $SQLServiceInfo += $SSRS

    #Execute the process for each service
    Foreach ($SQLService in $SQLServiceInfo){
        $ServiceName   = $SQLService.Name
        $DisplayName   = $SQLService.StartMode
        $Status = $SQLService.State

        #Insert data into SQLMonitor database
        $iSQLSQLServerInfo = "INSERT INTO [staging].[tSQLServiceStatus]
                              VALUES
('$ServiceName','$DisplayName','$Status','$HostId',GETDATE(),GETDATE())"
        Invoke-Sqlcmd -Query $iSQLSQLServerInfo -ServerInstance "SQLMonitor" -
Database "SQLMonitor"
    }

}
```

Al igual que el proceso anterior este finaliza con la ejecución del procedimiento almacenado monitor.spMergetSQLServiceStatu, el cual realiza una operación “merge” como el proceso mencionado en el punto 5.5.3.2.

5.5.3.4. Load_tDriveSize

Conocer el espacio del disco y su capacidad es muy importante a nivel de SQL Server, un disco lleno puede causar problemas de rendimiento en la base de datos, por eso, mediante PowerShell se creó el siguiente proceso para conocer la capacidad y estado de los discos. Igual que los procesos anteriores, se realiza una limpieza de la tabla y se cargan los datos.

```
#ETL - Host Drives Data
$tSQLHost = "TRUNCATE TABLE [monitor].[tDriveSize]"

Invoke-Sqlcmd -Query $tSQLHost -ServerInstance "SQLMonitor" -Database
"SQLMonitor"

$sSQLHost = "SELECT HostId
                    ,HostName
                    ,Environment
              FROM [monitor].[tMonitorHost]"

$sSQLHost = Invoke-Sqlcmd -Query $sSQLHost -ServerInstance "SQLMonitor" -
Database "SQLMonitor"
```

```
ForEach ($Server in $sSQLHost){

    $HostId      = $Server.HostId
    $HostName    = $Server.HostName

    $DriveInfo = Get-WmiObject -Class Win32_logicaldisk -ComputerName
$HostName  -Filter "DeviceId != 'Z:'" |
                 Select-Object -Property DeviceID, VolumeName,
                 @{L='FreeSpaceGB';E={"{0:N2}" -f ($_.FreeSpace /1GB)}},
                 @{L="TotalSizeGB";E={"{0:N2}" -f ($_.Size/1GB)}}

    ForEach ($Drive in $DriveInfo){

        $DriveLetter = $Drive.DeviceID
        $DriveName = $Drive.VolumeName
        $FreeSpaceGB = $Drive.FreeSpaceGB
        $TotalSizeGB = $Drive.TotalSizeGB

        $iSQLDriveInfo = "INSERT INTO [monitor].[tDriveSize]
                     VALUES ('$DriveLetter',
                             '$DriveName',
                             '$FreeSpaceGB',
                             '$TotalSizeGB',
                             '$HostId',
                             GETDATE(),
                             GETDATE())"

        $sSQLHost = Invoke-Sqlcmd -Query $iSQLDriveInfo -ServerInstance
"SQLMonitor" -Database "SQLMonitor"
    }
}
```

5.5.4. ETL - Paquetes SSIS

La mayoría de los procesos de extracción de datos fueron creados con paquetes de SSIS. Al igual que con los scripts de PowerShell mencionados anteriormente se creó un proceso estandarizado, así que todos los paquetes ejecutan los mismos pasos para la carga de datos, a continuación, se definieron algunos de los elementos que fueron utilizados para completar esta tarea. Como ejemplo se utilizan imágenes del SSIS llamado Load_Instance.

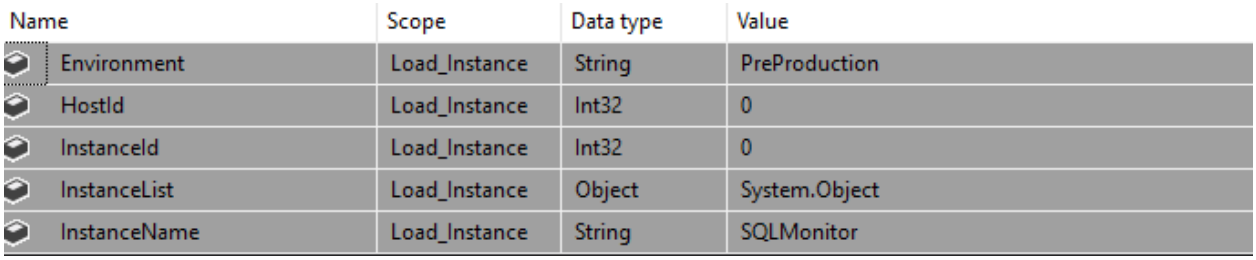
1) Truncar las tablas “staging”.
2) Extraer lista de instancias.
3) Recorrer la lista de instancias creando una conexión dinámica.
4) Extraer datos de la instancia fuente y cargar los datos en la base de datos SQLMonitor.
5) Ejecución del procedimiento almacenado “load” o “merge”.

5.5.4.1. Variables

Para este proceso se crearon cinco variables, las cuales son utilizadas para manejar conexión dinámica e insertar ciertos datos en las tablas que son necesarios

para generar relaciones. Estas variables son cargadas en el paso llamado “Extract Instance List”. Las variables se definen de la siguiente forma:

- **Environment:** Contiene el valor del tipo de ambiente.
- **HostId:** Contiene el identificador único para cada Host.
- **InstanceId:** Contiene el identificador único para cada Host.
- **InstanceList:** Contiene la lista de instancias que fueron recorridas para la extracción de datos.
- **InstanceName:** Contiene el nombre de la instancia.



Name	Scope	Data type	Value
Environment	Load_Instance	String	PreProduction
HostId	Load_Instance	Int32	0
InstanceId	Load_Instance	Int32	0
InstanceList	Load_Instance	Object	System.Object
InstanceName	Load_Instance	String	SQLMonitor

Figura 9. Variables Paquetes SSIS. Fuente: Tomada del servidor de pruebas SQLMonitor.

5.5.4.2. Conexión Dinámica

Para las conexiones de los paquetes de SSIS se crearon dos conexiones las cuales se definen de la siguiente manera:

- DynamicConnection: Realiza una conexión dinámica basada en los resultados de la variable InstanceList en donde toma el valor de la variable InstanceName que lleva el nombre de la instancia más el puerto, esto es agregado en la sección de expresiones de dicha conexión.
- SQLMonitor: Realiza la conexión al servidor y la base de datos SQLMonitor.

5.5.4.3. Funcionalidad – Paquetes SSIS

Como se mencionó anteriormente los paquetes de SSIS fueron estandarizados para que ejecutaran un proceso similar. Por lo cual, de esta forma se definen las siguientes funciones que son utilizadas en este proceso, como ejemplo se utilizó el paquete Load_InstanceConfiguration, ya que este en especial ejecuta un proceso alerta.

La figura 10 hace referencia al diseño del paquete SSIS y la figura 11 hace referencia a la instrucción Data Flow que se encarga de la extracción y carga de datos.

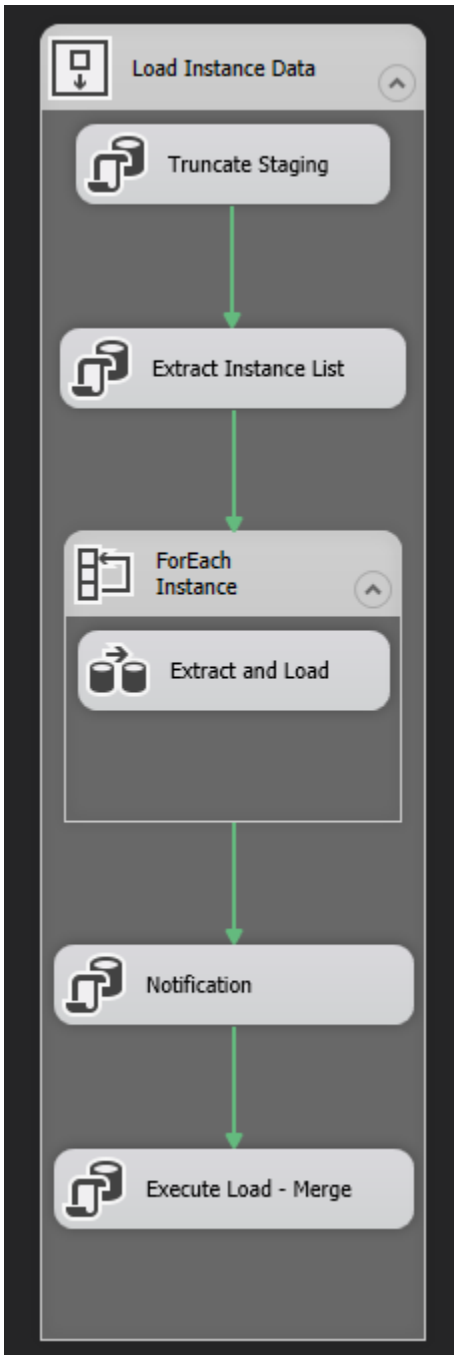


Figura 10: Paquete SSIS. Fuente: Tomada del servidor de pruebas SQLMonitor

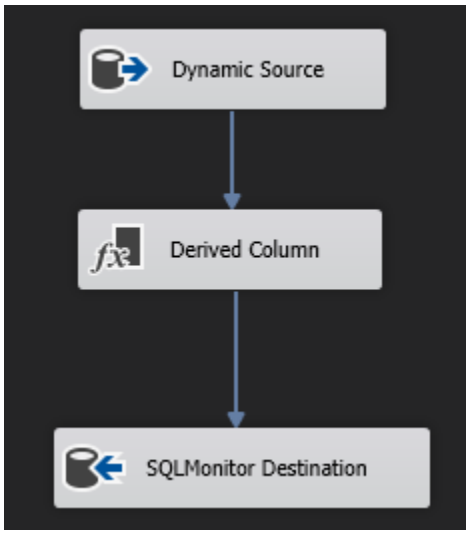


Figura 11: Paquete SSIS – Data Flow Task. Fuente: Tomada del servidor de pruebas SQLMonitor

1) **Sequence Container - Load Instance Data:** Es un simple contenedor en donde se agregaron las diferentes tareas que ejecuta el paquete de SSIS.
2) **Execute SQL Task - Truncate Staging:** Trunca los datos de las tablas pertenecientes al esquema “staging”.
3) **Execute SQL Task - Extract Instance List:** Extrae la lista de instancias en donde debe ser ejecutado el proceso de ETL, esta tarea realizó una consulta a las tablas monitor.tMonitorInstance las cuales crearon las diferentes conexiones dinámicas.

```
SELECT InstanceId
         ,InstanceName + ',' + TcpPort
         ,Environment
         ,HostId
FROM [monitor].[tMonitorInstance]
WHERE IsEnable = 1
```

4) **Foreach Loop Container - ForEach Instance:** Recorre los datos contenidos en la variable InstanceList, para así capturar los diferentes valores y crear las conexiones necesarios para la extracción y carga de los datos.

4.1) Data Flow Task - Extract and Load: Tarea encargada de extraer y cargar los datos.

4.1.1) OLE DB Source – Dynamic Source: Esta tarea realizó la extracción de datos de las diferentes fuentes en donde se ejecutan las consultas descritas en la sección 5.3.

4.1.2) Derived Columns: Aquí se acomodan las variables Environment e InstanceId, ya que en muchos casos estos valores deben ser insertados en las tablas.

4.1.3) OLE DB Destination – SQLMonitor Destination: Esta tarea ejecutó el proceso de carga en la base de datos SQLMonitor para las tablas que forman parte del esquema “staging”.

5) Notification: Esta tarea está presente en los procesos en los cuales se debe notificar a los administradores de bases de datos si hubo cambio, esto se realiza por medio de un procedimiento almacenado que compara los valores de las tablas “staging” con las tablas monitor, en caso de haber alguna diferencia este envía una notificación por medio de SQL Server.

6) Execute Load – Merge: El último paso es cargar las tablas del esquema monitor, las cuales son utilizadas para el proceso de monitoreo. Esto se realiza utilizando uno de los dos procedimientos almacenados mencionados en la sección 5.5.1.

5.5.4.4. Lista de Paquetes SSIS

Anteriormente se explicó y describió el proceso que ejecutan todos los paquetes de SSIS, cada uno de estos tiene relación con un proceso que es necesario para extraer los datos desde las diferentes fuentes. A continuación, la tabla 18 muestra la definición y uso de cada uno de los paquetes de Integration Services:

Paquete SSIS	Descripción
Load_AODatabase.dtsx	Extrae y carga los datos a nivel de base de datos relacionados con el estado de “Always On”.
Load_AOInstance.dtsx	Extrae y carga los datos a nivel de instancia relacionados con el estado de “Always On”.

Load_BackupHistory.dtsx	Extrae y carga el historial de respaldos en la instancia, el proceso extrae los últimos 30 días de respaldos.
Load_Database.dtsx	Extrae y carga los datos relacionados con las bases de datos que forman parte de cada instancia.
Load_DatabaseFiles.dtsx	Extrae y carga los datos relacionados con los archivos (mdf, ndf y ldf) que utilizan las diferentes bases de datos.
Load_Instance.dtsx	Extrae y carga los datos de las Instancias de SQL Server.
Load_InstanceConfiguration.dtsx	Extrae y carga los datos relacionados con las configuraciones avanzadas de las instancias SQL Server.
Load_JobStatus.dtsx	Extrae y carga el estado de los “jobs” que ejecutan las instancias de SQL Server.
Load_MissingIndex.dtsx	Extrae y carga las sugerencias sobre los índices faltantes en las bases de datos de usuario.
Load_PrivAccess.dtsx	Extrae y carga la lista de usuarios con altos privilegios en las diferentes instancias de SQL Server.
Load_RestoreHistory.dtsx	Extrae y carga el historial de restauración en la instancia, el proceso extrae los últimos 30 días de restauraciones.

Tabla 18. Definición de paquetes de SSIS. Fuente: Elaboración Propia

5.5.4.5. Integration Services Catalogs

Para finalizar el proceso fue importante utilizar la opción de “deployment” desde Visual Studio para cargar los paquetes de SSIS al catálogo de Integration Services dentro de la instancia SQLMonitor y así el manejo de errores es más sencillo y se puede manejar un control de cambios según sea necesario.

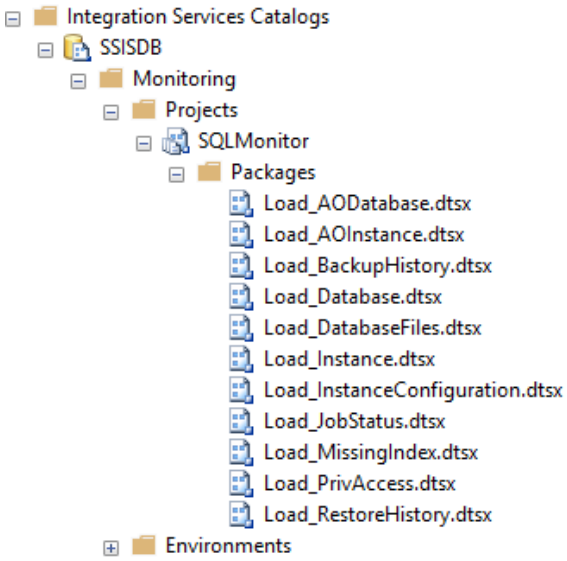


Figura 12: Integration Service Catalog. Fuente: Tomada del servidor de pruebas SQLMonitor

5.6. SQL Server Jobs

Los “jobs” de SQL Server fueron configurados y calendarizados para ejecutar los diferentes procesos de ETL que se crearon en PowerShell y SSIS. Para definir los “Jobs” se dividen en dos tipos, los que ejecutan los “scripts” de PowerShell y los que ejecutan los paquetes de SSIS, sin embargo, todos llevan un nombre estandarizado:

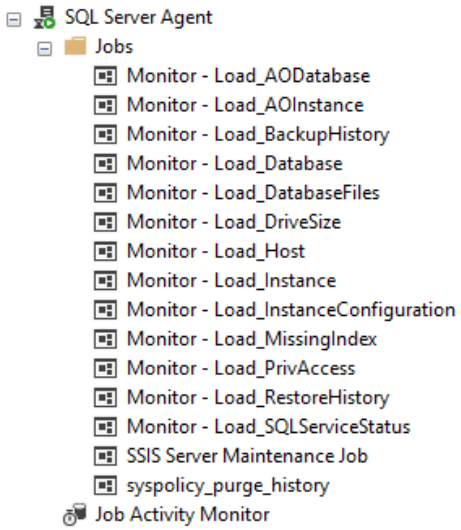


Figura 13: Jobs de Monitoreo. Fuente: Tomada del servidor de pruebas SQLMonitor

5.6.1. Lista de “Jobs”

Se crearon los siguientes “jobs” para ejecutar cada proceso:

Nombre Job	Nombre Step	Categoría Step	Dueño del Job	Nombre Proxy
Monitor - Load_AODatabase	Process ETL	SSIS	WINSQL\sqlmonitor	SQLMonitorProxy
Monitor - Load_AOInstance	Process ETL	SSIS	WINSQL\sqlmonitor	SQLMonitorProxy
Monitor - Load_BackupHistory	Process ETL	SSIS	WINSQL\sqlmonitor	SQLMonitorProxy
Monitor - Load_Database	Process ETL	SSIS	WINSQL\sqlmonitor	SQLMonitorProxy
Monitor - Load_DatabaseFiles	Process ETL	SSIS	WINSQL\sqlmonitor	SQLMonitorProxy
Monitor - Load_Instance	Process ETL	SSIS	WINSQL\sqlmonitor	SQLMonitorProxy
Monitor - Load_InstanceConfiguration	Process ETL	SSIS	WINSQL\sqlmonitor	SQLMonitorProxy
Monitor - Load_JobStatus	Process ETL	SSIS	WINSQL\sqlmonitor	SQLMonitorProxy
Monitor - Load_MissingIndex	Process ETL	SSIS	WINSQL\sqlmonitor	SQLMonitorProxy
Monitor - Load_PrivAccess	Process ETL	SSIS	WINSQL\sqlmonitor	SQLMonitorProxy
Monitor - Load_RestoreHistory	Process ETL	SSIS	WINSQL\sqlmonitor	SQLMonitorProxy
Monitor - Load_SQLServiceStatus	Process ETL	PowerShell	WINSQL\sqlmonitor	SQLMonitorProxy
	Notification	TSQL	WINSQL\sqlmonitor	N/A
	Merge Staging and Monitor	TSQL	WINSQL\sqlmonitor	N/A
Monitor - Load_DriveSize	Process ETL	PowerShell	WINSQL\sqlmonitor	SQLMonitorProxy
	Merge Staging and Monitor	TSQL	WINSQL\sqlmonitor	N/A
	Notification	TSQL	WINSQL\sqlmonitor	N/A
Monitor - Load_Host	Process ETL	PowerShell	WINSQL\sqlmonitor	SQLMonitorProxy
	Merge Staging and Monitor	TSQL	WINSQL\sqlmonitor	N/A

Tabla 19. Lista de “Job” con Detalle. Fuente: Elaboración Propia

Además, en la tabla 20 se muestra el calendario en el cual cada proceso fue ejecutado durante el tiempo de pruebas:

Nombre Job	Nombre Programación	Ocurrencia	Frecuencia

Monitor - Load_BackupHistory	Daily - 4 hour	Every 1 day(s)	Occurs every 4 Hour(s)
Monitor - Load_DatabaseFiles	Daily - 1 AM	Every 1 day(s)	Occurs once at 01:00:00
Monitor - Load_RestoreHistory	Daily - 4 Hours	Every 1 day(s)	Occurs every 10 Minute(s)
Monitor - Load_Database	Daily - 1 hour	Every 1 day(s)	Occurs every 1 Hour(s)
Monitor - Load_DriveSize	Daily - 15 min	Every 1 day(s)	Occurs every 15 Minute(s)
Monitor - Load_AOInstance	Daily - 10 min	Every 1 day(s)	Occurs every 10 Minute(s)
Monitor - Load_InstanceConfiguration	Daily - 1 AM	Every 1 day(s)	Occurs once at 01:00:00
Monitor - Load_Instance	Monthly	Day 1 of every 1 month(s)	Occurs once at 03:00:00
Monitor - Load_JobStatus	Daily - 1 AM	Every 1 day(s)	Occurs once at 01:00:00
Monitor - Load_MissingIndex	Daily - 12 Hours	Every 1 day(s)	Occurs every 12 Hour(s)
Monitor - Load_Host	Monthly	Day 1 of every 1 month(s)	Occurs once at 00:00:00
Monitor - Load_PrivAccess	Daily - 1 AM	Every 1 day(s)	Occurs once at 01:00:00
Monitor - Load_AODatabase	Daily - 10 min	Every 1 day(s)	Occurs every 10 Minute(s)
Monitor - Load_SQLServiceStatus	Daily - 5 min	Every 1 day(s)	Occurs every 5 Minute(s)

Tabla 20. Calendario de Ejecución. Fuente: Elaboración Propia

5.6.2. PowerShell Jobs

Para los “jobs” que ejecutaron procesos desde los “scripts” de PowerShell se utilizan entre dos a tres pasos para completar la tarea, el “job” “Monitoreo – Load_Host” es el único de los tres que no cuenta con un proceso de notificación, ya que los cambios en un host son poco constantes, lo cual hace que el “job” se ejecute en un rango de tiempo más amplio que los demás. Los pasos se definen de la siguiente forma:

- **Process ETL:** Ejecutó la instrucción PowerShell que acciona el proceso de ETL.
- **Merge Staging and Monitor:** Ejecutó el procedimiento almacenado que accionaba el proceso de “Load” o “Merge”.
- **Notification:** Ejecuta el procedimiento almacenado que acciona la notificación según las diferencias entre las tablas “staging” y monitor.

Los “Jobs” que ejecutaron los procesos de PowerShell son los siguientes:

- Monitor - Load_SQLServiceStatus
- Monitor - Load_DriveSize
- Monitor - Load_Host

5.6.3. SSIS Jobs

En este caso es más sencillo, ya que los siguientes “Jobs” solo contaron con un paso el cual accionaba el paquete de SSIS. Esto debido a que el paquete de SSIS acciona todas las instrucciones necesarias para extraer, cargar, aplicar el “Merge” o “Load” y enviar notificaciones por medio de un procedimiento almacenado. La lista de “Jobs” que ejecutan paquetes de SSIS es la siguiente:

- Monitor - Load_AODatabase
- Monitor - Load_AOInstance
- Monitor - Load_BackupHistory
- Monitor - Load_Database
- Monitor - Load_DatabaseFiles
- Monitor - Load_Instance
- Monitor - Load_InstanceConfiguration
- Monitor - Load_MissingIndex
- Monitor - Load_PrivAccess
- Monitor - Load_RestoreHistory

5.7. Alertas de Monitoreo

Lo esencial del proyecto fue la creación de alertas, las cuales en algunos escenarios específicos alertan al usuario final en caso de que se exista un cambio

en una tabla o simplemente envía notificaciones al correo electrónico mostrando ciertos resultados que pueden ser de interés para los DBA o los diferentes miembros de las unidades de negocio.

Cabe destacar que para esta sección es necesario un servidor SMTP, para efectos del proyecto se utilizó el servidor de SMTP de Outlook, con el cual se creó una cuenta llamada sqlmonitor.winsql@outlook.com, que será la encargada de enviar los correos.

Para el desarrollo de las notificaciones se utilizó el procedimiento almacenado del sistema SQL Server llamada sp_send_dbmail, el cual permitió crear notificaciones HTML basado en los resultados de las tablas de la base de datos SQLMonitor.

Es importante mencionar que el proceso de notificación es creado por medio de un procedimiento almacenado, el cual fue ejecutado en las diferentes secciones de lo “jobs” de SQL Server o en los paquetes de SSIS.

De esta forma se crearon dos tipos de alerta, una que solo se envía cuando se encuentra una diferencia entre los datos viejos y los datos nuevos, para este caso solo aplica para los datos que se encuentran en las tablas tDriveSize, tSQLServiceStatus, tAOInstance y tAODatabase. El otro tipo de alerta se envía de forma diaria y es para las configuraciones avanzadas de las diferentes instancias de SQL Server, esta notificación solo aplica para la tabla tInstanceConfiguration.

5.7.1. Procedimientos Almacenados de Alerta

Para la ejecución de las alertas se creó una serie de procedimientos almacenados los cuales validan la información y envían los reportes. A continuación, se muestra la lista de los procedimientos almacenados:

5.7.1.1. Alerta Diaria

- **monitor.spNotifyInstanceConfiguration:** Reporta de forma diaria las listas de configuraciones avanzadas en las diferentes instancias que forma parte del proceso de SQLMonitor.

```
CREATE PROCEDURE [monitor].[spNotifyInstanceConfiguration]
	@recipients NVARCHAR(MAX)
AS
BEGIN

	DECLARE @query NVARCHAR(MAX)
	DECLARE @subject NVARCHAR(MAX) = 'Instance Advance Configuration Daily
Report'

	SET @query =  N'<H1>Please see below the status of the advance
configuration of the instances included in the SQLMonitor:</H1>' +
N'<table border="1">' +
N'<tr><th>Instance Name</th><th>ShowAdvancedOptions</th>' +
N'<th>RemoteQueryTimeout(s)</th><th>CostThresholdforParallelism</th><th>MaxDegree
OfParallelism</th>' +
N'<th>MinMemoryPerQuery(KB)</th><th>MinServerMemory(MB)</th><th>MaxServerMemory(M
B)</th>' +
N'<th>XpCmdShell</th><th>Report Date</th></tr>' +
CAST ( ( SELECT td = [InstanceName], '',
                            td = [show advanced options], '',
                            td = [remote query timeout (s)], '',
                            td = [cost threshold for parallelism], '',
                            td = [max degree of parallelism], '',
                            td = [min memory per query (KB)], '',
                            td = [min server memory (MB)], '',
                            td = [max server memory (MB)], '',
                            td = [xp_cmdshell], '',
                      td = CONVERT(SMALLDATETIME, GETDATE(),120)
FROM [monitor].[vNotifyInstanceConfiguration]
FOR XML PATH('tr'), TYPE
        ) AS NVARCHAR(MAX) ) +
N'</table>';

	EXEC msdb.dbo.sp_send_dbmail @recipients = @recipients,
	@subject = @subject,
	@body = @query,
	@body_format = 'HTML' ;

END
```

5.7.1.2. Alertas según diferencias

- **monitor.spNotifyAODatabase:** Envía una notificación dependiendo del estado de la replicación entre las bases de datos que forman parte de “Always On”.

```
CREATE PROCEDURE [monitor].[spNotifyAODatabase]
@recipients NVARCHAR(MAX)
AS
BEGIN

DECLARE @query NVARCHAR(MAX)
DECLARE @subject NVARCHAR(MAX);
DECLARE @val INT = 1
DECLARE @count INT = (SELECT MAX(R) FROM [monitor].[vNotifyAODatabase])

WHILE @val <= @count
BEGIN

SELECT @subject = 'Issues in the AG named ' + AvailabilityGroupName +' on the database
name ' + DatabaseName + ', on primary node ' + InstanceName
FROM [monitor].[vNotifyAODatabase]
WHERE R = @val

SET @query =  N'<H1>The following databases are having replication issues:</H1>' +
  N'<table border="1">' +
  N'<tr><th>Instance Name</th><th>Availability Group Name</th>' +
  N'<th>Database Name</th><th>Is Primary Replica</th><th>Is Failover Ready</th>' +
  N'<th>Is Database Joined</th><th>Synchronization State</th><th>Synchronization State
Health</th>' +
  N'<th>Database State</th><th>Is Suspended</th><th>Suspened  Reason</th>' +
  N'<th>Last Hardened LSN</th><th>Last Hardened Time</th><th>Last Commit LSN</th>' +
  N'<th>Last Commit Time</th><th>Seconds Behind Primary</th>' +
  N'<th>Report Date</th></tr>' +
  CAST ( ( SELECT td = InstanceName, '',
  td = AvailabilityGroupName, '',
  td = DatabaseName, '',
  td = IsPrimaryReplica, '',
  td = IsFailoverReady, '',
  td = IsDatabaseJoined, '',
  td = SynchronizationState, '',
  td = SynchronizationStateHealth, '',
  td = DatabaseState, '',
  td = IsSuspended, '',
  td = SuspenedReason, '',
  td = LastHardenedLSN, '',
  td = LastHardenedTime, '',
  td = LastCommitLSN, '',
  td = LastCommitTime, '',
  td = SecondsBehindPrimary, '',
  td = CONVERT(SMALLDATETIME, GETDATE(),120)
          FROM [monitor].[vNotifyAODatabase]
          WHERE R = @val  FOR XML PATH('tr'), TYPE
   ) AS NVARCHAR(MAX) ) +  N'</table>';
EXEC msdb.dbo.sp_send_dbmail @recipients = @recipients,
@subject = @subject,
@body = @query,
@body_format = 'HTML' ;
SET @val = @val + 1
END
END
```

- **monitor.spNotifyAOInstance:** Envía una notificación instancia del estado de la replicación entre las instancias dentro de una configuración “Always On”.

```
CREATE PROCEDURE [monitor].[spNotifyAOInstance]
       @recipients NVARCHAR(MAX)
AS
BEGIN
       DECLARE @query NVARCHAR(MAX)
       DECLARE @subject NVARCHAR(MAX);
       DECLARE @val INT = 1
       DECLARE @count INT = (SELECT MAX(R) FROM [monitor].[vNotifyAOInstance])
       WHILE @val <= @count
BEGIN
SELECT @subject = 'Issues in the AG named ' + AvailabilityGroupName +', on
primary node ' + InstanceName
FROM [monitor].[vNotifyAOInstance]
WHERE R = @val
AND Role = 'Primary'

SET @query =  N'<H1>The following instances are having replication issues:</H1>'
+
 N'<table border="1">' +
 N'<tr><th>Instance Name</th><th>Availability Group Name</th>' +
 N'<th>Role</th><th>Availability Mode</th><th>Failover Mode</th>' +
 N'<th>Connected State</th><th>Recovery Health</th><th>Synchronization
Health</th>' +
 N'<th>Last Connect Error Number</th><th>Last Connect Error
Description</th><th>Last Connect Error Time Stamp</th>' +
 N'<th>Report Date</th></tr>' +
 CAST ( ( SELECT td = InstanceName, '',
td = AvailabilityGroupName, '',
td = Role, '',
td = AvailabilityMode, '',
td = FailoverMode, '',
td = ConnectedState, '',
td = RecoveryHealth, '',
td = SynchronizationHealth, '',
td = LastConnectErrorNumber, '',
td = LastConnectErrorDescription, '',
td = LastConnectErrorTimeStamp, '',
td = CONVERT(SMALLDATETIME, GETDATE(),120)
  FROM [monitor].[vNotifyAOInstance]
  WHERE R = @val
ORDER BY Role
  FOR XML PATH('tr'), TYPE
        ) AS NVARCHAR(MAX) ) +
 N'</table>';

EXEC msdb.dbo.sp_send_dbmail @recipients = @recipients,
@subject = @subject,
@body = @query,
@body_format = 'HTML' ;
SET @val = @val + 1
       END
END
```

- **monitor.spNotifyDriveSize:** Envía una notificación si el espacio utilizado de los discos de un servidor supera el 90%.

```
CREATE PROCEDURE [monitor].[spNotifyDriveSize]
	@recipients NVARCHAR(MAX)
AS
BEGIN

	DECLARE @query NVARCHAR(MAX)
	DECLARE @subject NVARCHAR(MAX);
	DECLARE @val INT = 1
	DECLARE @count INT = (SELECT MAX(RN) FROM [monitor].[vNotifyDriveSize])

	WHILE @val <= @count
	BEGIN

SELECT @subject = 'Drive ' + DriveLetter + ' exceeded the 90% of usage in the
host ' + HostName
FROM [monitor].[vNotifyDriveSize]
WHERE RN = @val

SET @query =  N'<H1>The following drive is exceeding the 90% of usage:</H1>' +
N'<table border="1">' +
N'<tr><th>Host Name</th><th>Drive Letter</th>' +
N'<th>Drive Name</th><th>Free Space GB</th>' +
N'<th>Free Space Percentage</th><th>UsedSpaceGB</th><th>Used Space
Percentage</th>' +
N'<th>Total Size GB</th><th>Report Date</th></tr>' +
CAST ( ( SELECT td = HostName, '',
td = DriveLetter, '',
td = DriveName, '',
td = FreeSpaceGB, '',
td = FreeSpacePercentage, '',
td = UsedSpaceGB, '',
td = UsedSpacePercentage, '',
td = TotalSizeGB, '',
td = CONVERT(SMALLDATETIME, GETDATE(),120)
FROM [monitor].[vNotifyDriveSize]
WHERE RN = @val
FOR XML PATH('tr'), TYPE
   ) AS NVARCHAR(MAX) ) +
N'</table>';

EXEC msdb.dbo.sp_send_dbmail @recipients = @recipients,
@subject = @subject,
@body = @query,
@body_format = 'HTML' ;

SET @val = @val + 1
	END

END
```

- **monitor.spNotifySQLServiceStatus:** Envía una notificación si el estado de algunos de los servicios de SQL Server pasa de “RUNNING” a “STOPPED”.

```
CREATE PROCEDURE [monitor].[spNotifySQLServiceStatus]
       @recipients NVARCHAR(MAX)
AS
BEGIN

       DECLARE @query NVARCHAR(MAX)
       DECLARE @subject NVARCHAR(MAX);
       DECLARE @val INT = 1
       DECLARE @count INT = (SELECT MAX(RN) FROM
[monitor].[vNotifySQLServiceStatus])

       WHILE @val <= @count
       BEGIN

SELECT @subject = 'Service ' + ServiceName +' is stopped in the host ' + HostName
FROM [monitor].[vNotifySQLServiceStatus]
WHERE RN = @val

SET @query =  N'<H1>The following services are stopped:</H1>' +
 N'<table border="1">' +
 N'<tr><th>SQL Service Id</th><th>Service Name</th>' +
 N'<th>Host Name</th><th>Start Mode</th><th>Status</th>' +
 N'<th>Report Date</th></tr>' +
 CAST ( ( SELECT td = SQLServiceId, '',
td = ServiceName, '',
td = HostName, '',
td = StartMode, '',
td = Status, '',
td = CONVERT(SMALLDATETIME, GETDATE(),120)
         FROM [monitor].[vNotifySQLServiceStatus]
         WHERE RN = @val
         FOR XML PATH('tr'), TYPE
   ) AS NVARCHAR(MAX) ) +
 N'</table>';

EXEC msdb.dbo.sp_send_dbmail @recipients = @recipients,
@subject = @subject,
@body = @query,
@body_format = 'HTML' ;

SET @val = @val + 1
       END

END
```

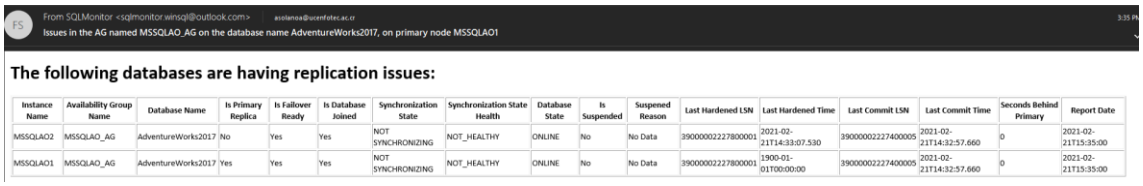
5.7.2. Alertas Enviadas

Como se comentó anteriormente las alertas son enviadas al correo de los involucrados o interesados para obtener información de alta importancia sobre el

mantenimiento y control de bases de datos. A continuación, se muestran los resultados de las diferentes alertas:

5.7.2.1. Base de Datos con "Always On"

Esta alerta solo se envía cuando uno de los atributos que muestran el estado de "Always On" cambia, eso quiere decir que si la replicación en algún momento no es saludable esta alerta es enviada.


From SQLMonitor <sqlmonitor.winsql@outlook.com> asolanoa@ucenfotec.ac.cr 3:35 PM
Issues in the AG named MSSQLAO_AG on the database name AdventureWorks2017, on primary node MSSQLAO1

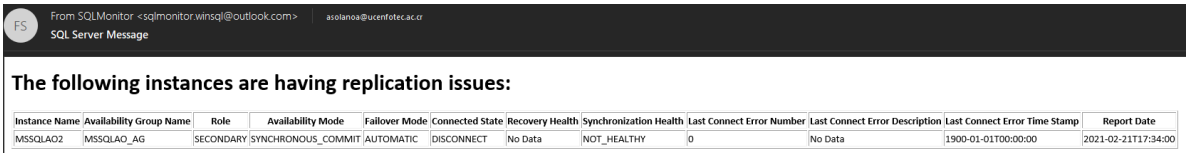
The following databases are having replication issues:

Instance Name	Availability Group Name	Database Name	Is Primary Replica	Is Failover Ready	Is Database Joined	Synchronization State	Synchronization State Health	Database State	Is Suspended	Suspened Reason	Last Hardened LSN	Last Hardened Time	Last Commit LSN	Last Commit Time	Seconds Behind Primary	Report Date
MSSQLAO2	MSSQLAO_AG	AdventureWorks2017	No	Yes	Yes	NOT SYNCHRONIZING	NOT_HEALTHY	ONLINE	No	No Data	39000002227800001	2021-02-21T14:33:07.530	39000002227400005	2021-02-21T14:32:57.660	0	2021-02-21T15:35:00
MSSQLAO1	MSSQLAO_AG	AdventureWorks2017	Yes	Yes	Yes	NOT SYNCHRONIZING	NOT_HEALTHY	ONLINE	No	No Data	39000002227800001	1900-01-01T00:00:00	39000002227400005	2021-02-21T14:32:57.660	0	2021-02-21T15:35:00

Figura 14: Alerta Base de Datos con Always On. Fuente: Tomada del servidor de pruebas SQLMonitor

5.7.2.2. Instancias con "Always On"

Al igual que la alerta en el punto 5.7.2.1 se basa en estado de salud de los nodos que conforman un grupo de replicación de "Always On".


From SQLMonitor <sqlmonitor.winsql@outlook.com> asolanoa@ucenfotec.ac.cr
SQL Server Message

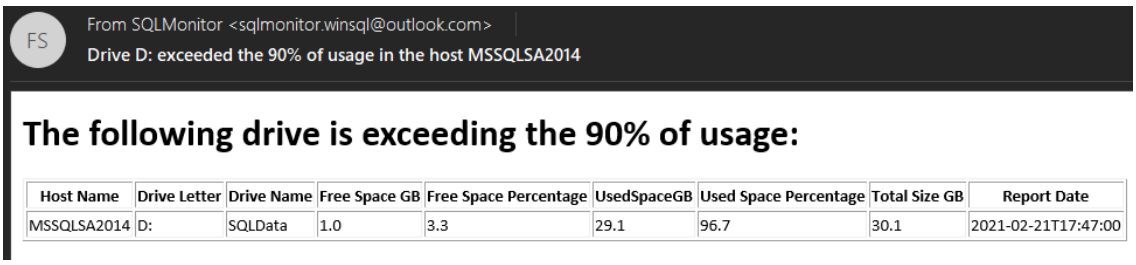
The following instances are having replication issues:

Instance Name	Availability Group Name	Role	Availability Mode	Failover Mode	Connected State	Recovery Health	Synchronization Health	Last Connect Error Number	Last Connect Error Description	Last Connect Error Time Stamp	Report Date
MSSQLAO2	MSSQLAO_AG	SECONDARY	SYNCHRONOUS_COMMIT	AUTOMATIC	DISCONNECT	No Data	NOT_HEALTHY	0	No Data	1900-01-01T00:00:00	2021-02-21T17:34:00

Figura 15: Alerta Instancia con "Always On". Fuente: Tomada del servidor de pruebas SQLMonitor

5.7.2.3. Espacio en Disco

En este caso se usa un estándar en donde se alerta cuando el espacio del disco supera un 90% de uso del total de la capacidad del disco.


From SQLMonitor <sqlmonitor.winsql@outlook.com>
Drive D: exceeded the 90% of usage in the host MSSQLSA2014

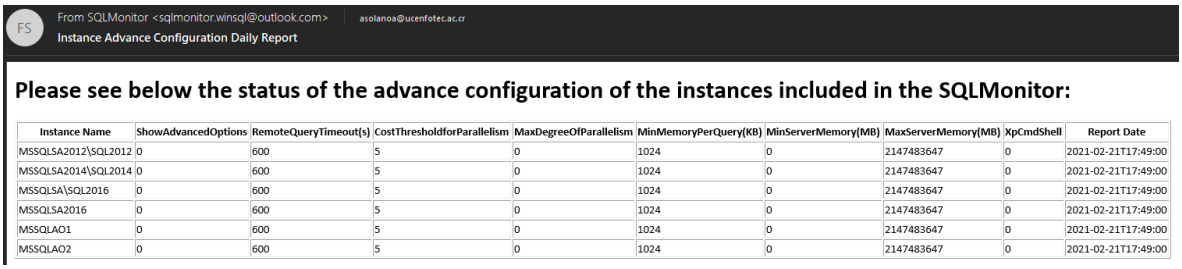
The following drive is exceeding the 90% of usage:

Host Name	Drive Letter	Drive Name	Free Space GB	Free Space Percentage	UsedSpaceGB	Used Space Percentage	Total Size GB	Report Date
MSSQLSA2014	D:	SQLData	1.0	3.3	29.1	96.7	30.1	2021-02-21T17:47:00

Figura 16: Alerta Espacio en Disco. Fuente: Tomada del servidor de pruebas SQLMonitor

5.7.2.4. Configuración Avanzada de Instancias

Esta alerta es una notificación informativa, la cual envía un listado de las configuraciones avanzadas más importantes de SQL Server. Al ser informativa se envía de forma diaria.


From SQLMonitor <sqlmonitor.winsql@outlook.com> asolanoa@ucenfotec.ac.cr

Instance Advance Configuration Daily Report

Please see below the status of the advance configuration of the instances included in the SQLMonitor:

Instance Name	ShowAdvancedOptions	RemoteQueryTimeout(s)	CostThresholdforParallelism	MaxDegreeOfParallelism	MinMemoryPerQuery(KB)	MinServerMemory(MB)	MaxServerMemory(MB)	XpCmdShell	Report Date
MSSQLSA2012\SQL2012	0	600	5	0	1024	0	2147483647	0	2021-02-21T17:49:00
MSSQLSA2014\SQL2014	0	600	5	0	1024	0	2147483647	0	2021-02-21T17:49:00
MSSQLSA\SQL2016	0	600	5	0	1024	0	2147483647	0	2021-02-21T17:49:00
MSSQLSA2016	0	600	5	0	1024	0	2147483647	0	2021-02-21T17:49:00
MSSQLAO1	0	600	5	0	1024	0	2147483647	0	2021-02-21T17:49:00
MSSQLAO2	0	600	5	0	1024	0	2147483647	0	2021-02-21T17:49:00

Figura 17: Alerta Espacio en Disco. Fuente: Tomada del servidor de pruebas SQLMonitor

5.7.2.5. Estado de Servicios SQL Server

Una de las alertas más importantes es cuando un servidor o servicio de SQL Server está detenido. Por lo cual esta alerta se envía cuando el estatus cambia a un estado que sea diferente a “Running”.


From SQLMonitor <sqlmonitor.winsql@outlook.com> asolanoa@ucenfotec.ac.cr

Service MSSQL$SQL2012 is stopped in the host MSSQLSA2012

The following services are stopped:

SQL Service Id	Service Name	Host Name	Start Mode	Status	Report Date
19	MSSQL$SQL2012	MSSQLSA2012	Auto	Stopped	2021-02-21T17:53:00

Figura 18: Alerta de Estado de Servicios SQL Server. Fuente: Tomada del servidor de pruebas SQLMonitor

5.8. Reportes

Una de las partes más importantes de este proyecto es la creación de reportes para así poder visualizar de forma más clara y sencilla los datos relacionados con los diferentes ambientes de SQL Server.

Se utilizó el servicio de SQL Server Reporting Services para crear los reportes ya que la licencia para esto ya está incluida dentro del SQL Server, lo cual evita una inversión extra en otra herramienta de reportería.

Durante la realización de los reportes se intentó crear un sistema que pudiese entrelazarse entre sí, así poder crear un tipo de navegación dentro del sistema de reportería.

Es importante destacar que los valores en las tablas como nombre de instancia, host, base de datos, gráficos y los valores que fueron agregados en una sección llamada “Actions” pueden ser utilizados para navegar dentro del sistema de reportería para visualizar datos más precisos según lo seleccionado.

5.8.1. Fuentes de Datos

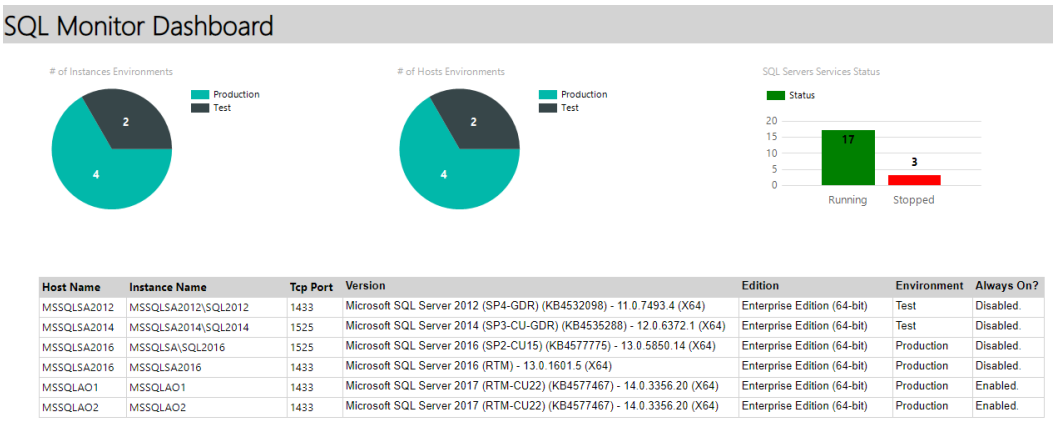
Se crearon dos fuentes de datos que se utilizaron para conectarse a las instancias, cada reporte cuenta con su propia fuente de datos que puede variar dependiendo de su objetivo. En algunos casos se realizó una conexión al servidor de monitoreo y en otros casos se utilizó una conexión dinámica directamente a un servidor para consultar datos de rendimiento en tiempo real.

5.8.2. Definición de Reportes

Como se comentó en la sección 5.5.2, todos los reportes extraen datos de las vistas que previamente fueron creadas para mostrar la información necesaria, por lo cual solo se realiza una simple consulta para extraer los datos. A continuación, se muestran los reportes creados:

5.8.2.1. SQLMonitorDashboard

Se creó un pequeño “dashboard” que muestra cuatro elementos los cuales se dividen en tres gráficas y una tabla. Las gráficas fueron meramente informativas las cuales muestran información relacionada a la cantidad de instancias y servidores agregados en la herramienta de monitoreo, el tercer gráfico muestra los servicios que actualmente están funcionando o los servicios que están detenidos dentro de un servidor. Por último, la tabla muestra una lista de relación entre los servidores y las instancias.



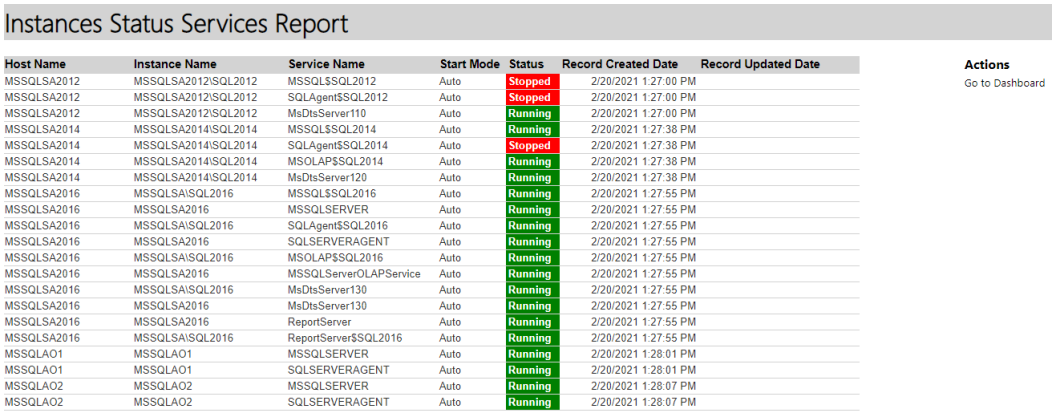
Host Name	Instance Name	Tcp Port	Version	Edition	Environment	Always On?
MSSQLSA2012	MSSQLSA2012\SQL2012	1433	Microsoft SQL Server 2012 (SP4-GDR) (KB4532098) - 11.0.7493.4 (X64)	Enterprise Edition (64-bit)	Test	Disabled.
MSSQLSA2014	MSSQLSA2014\SQL2014	1525	Microsoft SQL Server 2014 (SP3-CU-GDR) (KB4535288) - 12.0.6372.1 (X64)	Enterprise Edition (64-bit)	Test	Disabled.
MSSQLSA2016	MSSQLSA\SQL2016	1525	Microsoft SQL Server 2016 (SP2-CU15) (KB4577775) - 13.0.5850.14 (X64)	Enterprise Edition (64-bit)	Production	Disabled.
MSSQLSA2016	MSSQLSA2016	1433	Microsoft SQL Server 2016 (RTM) - 13.0.1601.5 (X64)	Enterprise Edition (64-bit)	Production	Disabled.
MSSQLAO1	MSSQLAO1	1433	Microsoft SQL Server 2017 (RTM-CU22) (KB4577467) - 14.0.3356.20 (X64)	Enterprise Edition (64-bit)	Production	Enabled.
MSSQLAO2	MSSQLAO2	1433	Microsoft SQL Server 2017 (RTM-CU22) (KB4577467) - 14.0.3356.20 (X64)	Enterprise Edition (64-bit)	Production	Enabled.

Figura 19: Reporte SQLMonitor Dashboard. Fuente: Tomada del servidor de pruebas SQLMonitor

Es importante saber que estos datos no son detallados, pero el reporte es lo suficientemente dinámico para funcionar de una forma rápida y sencilla, dando un clic en el gráfico llamado “SQL Servers Services Status”, en el nombre de un host, en el nombre de una instancia o en una de las opciones de la columna nombrada “Always On”, donde sea igual a “Enabled” y este redirecciona el “dashboard” a un reporte con información más detallada según la opción escogida.

5.8.2.2. InstanceServicesReport

Este reporte mostró un listado de todos los servicios relacionados a SQL Server que fueron instalados en los servidores, el reporte marcó en verde donde los servicios tienen un estado favorecedor; se indicaba la correcta ejecución de un servicio. Se marcó en rojo cuando un servicio mostraba un estado diferente a “Running.”



Host Name	Instance Name	Service Name	Start Mode	Status	Record Created Date	Record Updated Date
MSSQLSA2012	MSSQLSA2012\SQL2012	MSSQL$SQL2012	Auto	Stopped	2/20/2021 1:27:00 PM	
MSSQLSA2012	MSSQLSA2012\SQL2012	SQLAgent$SQL2012	Auto	Stopped	2/20/2021 1:27:00 PM	
MSSQLSA2012	MSSQLSA2012\SQL2012	MsDtsServer110	Auto	Running	2/20/2021 1:27:00 PM	
MSSQLSA2014	MSSQLSA2014\SQL2014	MSSQL$SQL2014	Auto	Running	2/20/2021 1:27:38 PM	
MSSQLSA2014	MSSQLSA2014\SQL2014	SQLAgent$SQL2014	Auto	Stopped	2/20/2021 1:27:38 PM	
MSSQLSA2014	MSSQLSA2014\SQL2014	MSOLAP$SQL2014	Auto	Running	2/20/2021 1:27:38 PM	
MSSQLSA2014	MSSQLSA2014\SQL2014	MsDtsServer120	Auto	Running	2/20/2021 1:27:38 PM	
MSSQLSA2016	MSSQLSA\SQL2016	MSSQL$SQL2016	Auto	Running	2/20/2021 1:27:55 PM	
MSSQLSA2016	MSSQLSA2016	MSSQLSERVER	Auto	Running	2/20/2021 1:27:55 PM	
MSSQLSA2016	MSSQLSA\SQL2016	SQLAgent$SQL2016	Auto	Running	2/20/2021 1:27:55 PM	
MSSQLSA2016	MSSQLSA2016	SQLSERVERAGENT	Auto	Running	2/20/2021 1:27:55 PM	
MSSQLSA2016	MSSQLSA\SQL2016	MSOLAP$SQL2016	Auto	Running	2/20/2021 1:27:55 PM	
MSSQLSA2016	MSSQLSA2016	MSSQLServerOLAPService	Auto	Running	2/20/2021 1:27:55 PM	
MSSQLSA2016	MSSQLSA\SQL2016	MsDtsServer130	Auto	Running	2/20/2021 1:27:55 PM	
MSSQLSA2016	MSSQLSA2016	MsDtsServer130	Auto	Running	2/20/2021 1:27:55 PM	
MSSQLSA2016	MSSQLSA2016	ReportServer	Auto	Running	2/20/2021 1:27:55 PM	
MSSQLSA2016	MSSQLSA\SQL2016	ReportServer$SQL2016	Auto	Running	2/20/2021 1:27:55 PM	
MSSQLAO1	MSSQLAO1	MSSQLSERVER	Auto	Running	2/20/2021 1:28:01 PM	
MSSQLAO1	MSSQLAO1	SQLSERVERAGENT	Auto	Running	2/20/2021 1:28:01 PM	
MSSQLAO2	MSSQLAO2	MSSQLSERVER	Auto	Running	2/20/2021 1:28:07 PM	
MSSQLAO2	MSSQLAO2	SQLSERVERAGENT	Auto	Running	2/20/2021 1:28:07 PM	

Figura 20: Reporte Estado de Servicios. Fuente: Tomada del servidor de pruebas SQLMonitor

Al igual que el reporte anterior, si seleccionaba el nombre de un servidor o el nombre de una instancia redireccionaba el sitio a un reporte con más información.

5.8.2.3. InstanceReport

Este reporte se abre cuando se seleccionaba una instancia de la tabla que se muestra en el “dashboard” o en alguna otra sección que muestre el nombre de instancia, cuando se seleccionaba el nombre de una instancia como MSSQLSA2014\SQL2014 abre inmediatamente un reporte con más información relacionado con la instancia.

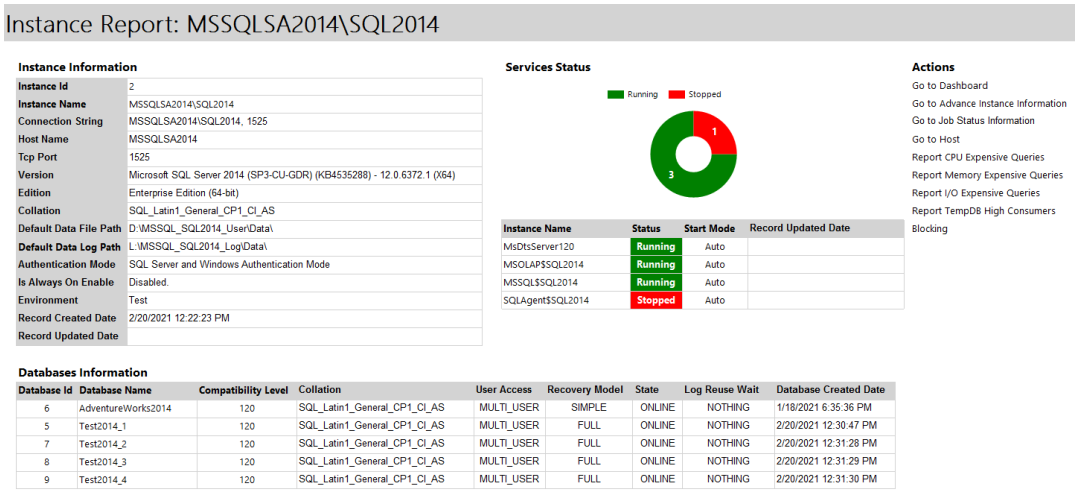


Figura 21: Reporte Instancia. Fuente: Tomada del servidor de pruebas SQLMonitor

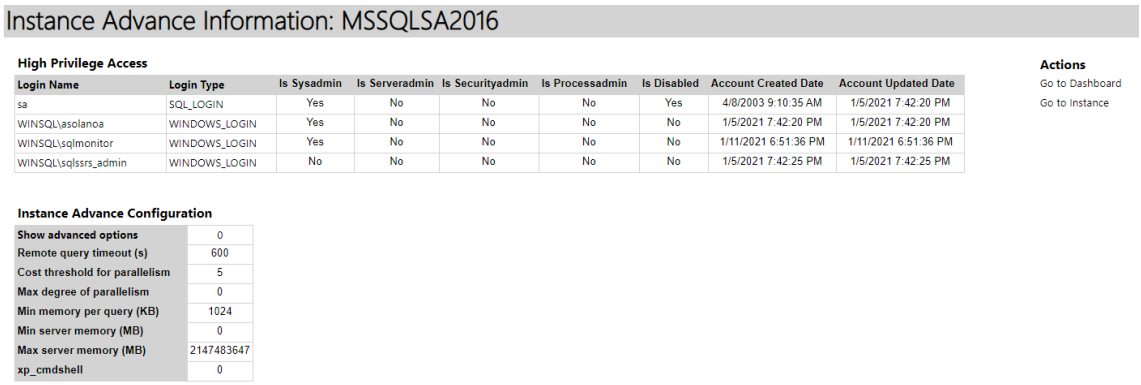
Este reporte mostró diferente información relacionada con la instancia, la primera columna titulada “Instance Information” mostró toda la información relacionada con la instancia que se seleccionó.

La segunda sección muestra la información relacionada con el estado de los servicios de SQL Server que están relacionados con esta instancia específica.

Además, se muestra una tabla con la información de las bases de datos que forman parte de esta instancia, al igual que los reportes anteriores si selecciona el nombre de una base de datos, el reporte mostraba más información de lo seleccionado.

5.8.2.4. InstanceAdvanceInformationReport

Este reporte se mostró en la sección de “Actions” la cual despliega los datos sobre las configuraciones avanzadas dentro de la instancia, también desplegó la información acerca de los usuarios que tiene privilegios de alto nivel dentro de una instancia de SQL Server.



Instance Advance Information: MSSQLSA2016

High Privilege Access

Login Name	Login Type	Is Sysadmin	Is Serveradmin	Is Securityadmin	Is Processadmin	Is Disabled	Account Created Date	Account Updated Date
sa	SQL_LOGIN	Yes	No	No	No	Yes	4/8/2003 9:10:35 AM	1/5/2021 7:42:20 PM
WINSQL\asolanoa	WINDOWS_LOGIN	Yes	No	No	No	No	1/5/2021 7:42:20 PM	1/5/2021 7:42:20 PM
WINSQL\sqlmonitor	WINDOWS_LOGIN	Yes	No	No	No	No	1/11/2021 6:51:36 PM	1/11/2021 6:51:36 PM
WINSQL\sqlssrs_admin	WINDOWS_LOGIN	No	No	No	No	No	1/5/2021 7:42:25 PM	1/5/2021 7:42:25 PM

Actions

Go to Dashboard

Go to Instance

Instance Advance Configuration

Show advanced options	0
Remote query timeout (s)	600
Cost threshold for parallelism	5
Max degree of parallelism	0
Min memory per query (KB)	1024
Min server memory (MB)	0
Max server memory (MB)	2147483647
xp_cmdshell	0

Figura 22: Reporte Información Avanzada Instancia. Fuente: Tomada del servidor de pruebas SQLMonitor

5.8.2.5. JobStatusReport

Este reporte permitía visualizar la información sobre el estado de los diferentes “jobs” que pueden ser ejecutados dentro de una instancia de SQL Server, de esta forma se conocía el estado de la última ejecución del “job”.

Job Status: MSSQLSA2016

Job Status Information

Job Name	Job Owner	Is Enabled	Step Name	Message	Run Status	Run Duration (s)	Run Duration	Run Date	Job Date Created	Job Date Modified
TestJob_1	WINSQL\sqlssrs_admin	Yes	Select	'EXECUTE AS LOGIN' failed for the requested login 'WINSQL\sqlssrs_admin'. The step failed.	Failed	0	00:00:00	2/22/2021 4:45:33 PM	2/22/2021 4:42:51 PM	2/22/2021 4:44:05 PM
TestJob_2	WINSQL\sqlssrs_admin	Yes	Select	'EXECUTE AS LOGIN' failed for the requested login 'WINSQL\sqlssrs_admin'. The step failed.	Failed	0	00:00:00	2/22/2021 4:45:35 PM	2/22/2021 4:44:27 PM	2/22/2021 4:44:27 PM
TestJob_3	WINSQL\sqlmonitor	Yes	Select	Executed as user: WINSQL\sqlagent_admin. The step succeeded.	Succeeded	0	00:00:00	2/22/2021 4:45:38 PM	2/22/2021 4:45:15 PM	2/22/2021 4:45:16 PM

Actions

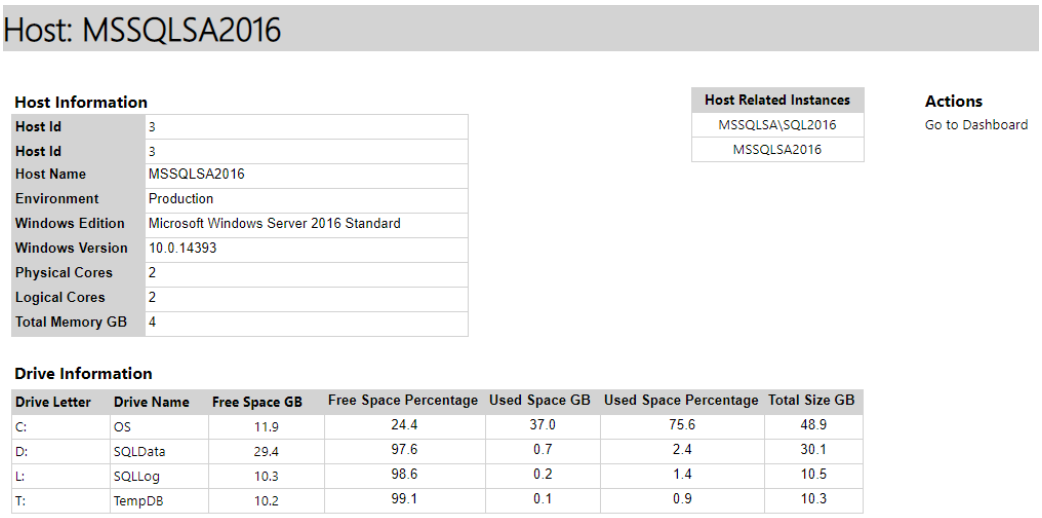
Go to Dashboard

Go to Instance

Figura 23: Reporte Estado de los “Jobs”. Fuente: Tomada del servidor de pruebas SQLMonitor

5.8.2.6. HostReport

El reporte de host mostraba la información sobre los hosts que tienen una instancia de SQL Server ejecutándose, en este caso este reporte cuenta con la información sobre el host, un listado de las instancias que están instaladas y un reporte sobre la capacidad de los discos.



Host: MSSQLSA2016

Host Information

Host Id	3
Host Id	3
Host Name	MSSQLSA2016
Environment	Production
Windows Edition	Microsoft Windows Server 2016 Standard
Windows Version	10.0.14393
Physical Cores	2
Logical Cores	2
Total Memory GB	4

Host Related Instances
MSSQLSA\SQL2016
MSSQLSA2016

Actions

Go to Dashboard

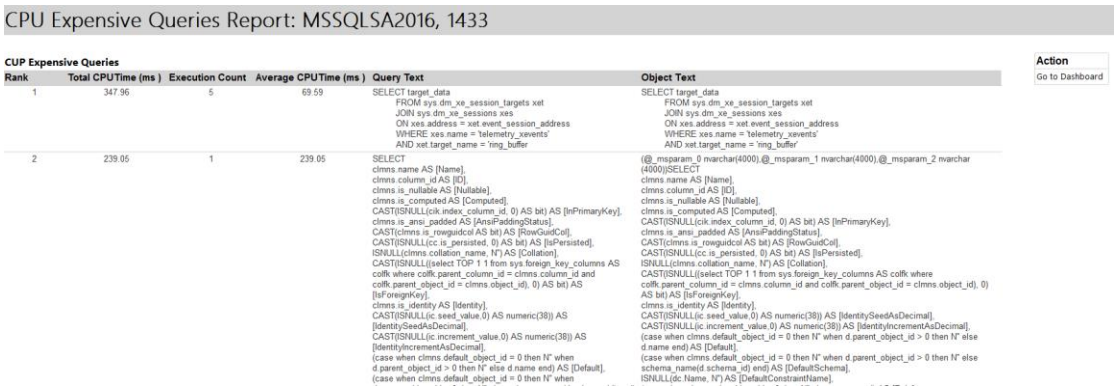
Drive Information

Drive Letter	Drive Name	Free Space GB	Free Space Percentage	Used Space GB	Used Space Percentage	Total Size GB
C:	OS	11.9	24.4	37.0	75.6	48.9
D:	SQLData	29.4	97.6	0.7	2.4	30.1
L:	SQLLog	10.3	98.6	0.2	1.4	10.5
T:	TempDB	10.2	99.1	0.1	0.9	10.3

Figura 24: Reporte Host. Fuente: Tomada del servidor de pruebas SQLMonitor

5.8.2.7. ExpensiveQueriesCPUReport

Este reporte utilizó la conexión dinámica, debido a la naturaleza de la consulta este tipo de datos puede cambiar dependiendo del rendimiento del servidor, por lo cual se conecta dinámicamente a la instancia escogida para mostrar los datos de rendimiento. Este reporte muestra las consultas más costosas para el CPU dentro de SQL Server.



CPU Expensive Queries Report: MSSQLSA2016, 1433

CUP Expensive Queries

Rank	Total CPUTime (ms)	Execution Count	Average CPUTime (ms)	Query Text	Object Text
1	347.96	5	69.59	SELECT target_data FROM sys.dm_xe_session_targets xet JOIN sys.dm_xe_sessions xes ON xes.address = xet.event_session_address WHERE xes.name = 'telemetry_xevents' AND xet.target_name = 'ring_buffer'	SELECT target_data FROM sys.dm_xe_session_targets xet JOIN sys.dm_xe_sessions xes ON xes.address = xet.event_session_address WHERE xes.name = 'telemetry_xevents' AND xet.target_name = 'ring_buffer'
2	239.05	1	239.05	SELECT clmns.name AS [Name], clmns.column_id AS [ID], clmns.is_nullable AS [Nullable], clmns.is_computed AS [Computed], CAST(ISNULL(cik.index_column_id, 0) AS bit) AS [InPrimaryKey], clmns.is_ansi_padded AS [AnsiPaddingStatus], CAST(clmns.is_rowguidcol AS bit) AS [RowGuidCol], CAST(ISNULL(cc.is_persisted, 0) AS bit) AS [IsPersisted], ISNULL(clmns.collation_name, N'') AS [Collation], CAST(ISNULL((select TOP 1 1 from sys.foreign_key_columns AS colfk where colfk.parent_column_id = clmns.column_id and colfk.parent_object_id = clmns.object_id), 0) AS bit) AS [IsForeignKey], clmns.is_identity AS [Identity], CAST(ISNULL(ic.seed_value,0) AS numeric(38)) AS [IdentitySeedAsDecimal], CAST(ISNULL(ic.increment_value,0) AS numeric(38)) AS [IdentityIncrementAsDecimal], (case when clmns.default_object_id = 0 then N'' when d.parent_object_id > 0 then N'' else d.name end) AS [Default], (case when clmns.default_object_id = 0 then N'' when	(@_msparam_0 nvarchar(4000),@_msparam_1 nvarchar(4000),@_msparam_2 nvarchar (4000))SELECT clmns.name AS [Name], clmns.column_id AS [ID], clmns.is_nullable AS [Nullable], clmns.is_computed AS [Computed], CAST(ISNULL(cik.index_column_id, 0) AS bit) AS [InPrimaryKey], clmns.is_ansi_padded AS [AnsiPaddingStatus], CAST(clmns.is_rowguidcol AS bit) AS [RowGuidCol], CAST(ISNULL(cc.is_persisted, 0) AS bit) AS [IsPersisted], ISNULL(clmns.collation_name, N'') AS [Collation], CAST(ISNULL((select TOP 1 1 from sys.foreign_key_columns AS colfk where colfk.parent_column_id = clmns.column_id and colfk.parent_object_id = clmns.object_id), 0) AS bit) AS [IsForeignKey], clmns.is_identity AS [Identity], CAST(ISNULL(ic.seed_value,0) AS numeric(38)) AS [IdentitySeedAsDecimal], CAST(ISNULL(ic.increment_value,0) AS numeric(38)) AS [IdentityIncrementAsDecimal], (case when clmns.default_object_id = 0 then N'' when d.parent_object_id > 0 then N'' else d.name end) AS [Default], (case when clmns.default_object_id = 0 then N'' when d.parent_object_id > 0 then N'' else schema_name(d.schema_id) end) AS [DefaultSchema], ISNULL(dc.Name, N'') AS [DefaultConstraintName],

Action

Go to Dashboard

Figura 25: Reporte Consulta Costosas CPU. Fuente: Tomada del servidor de pruebas SQLMonitor

5.8.2.8. ExpensiveQueriesIOReport

Este reporte al igual que el anterior en la sección 5.8.2.7 utiliza una conexión dinámica para seleccionar las consultas que pueden exigir mayor rendimiento a nivel de lectura y escritura de disco.

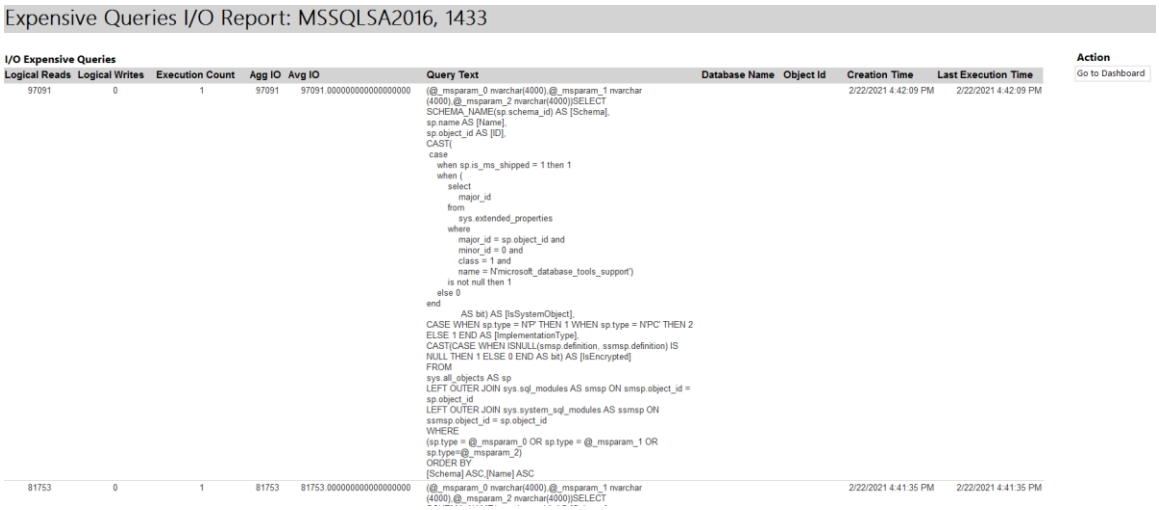


Figura 26: Reporte Consulta Costosas I/O. Fuente: Tomada del servidor de pruebas SQLMonitor

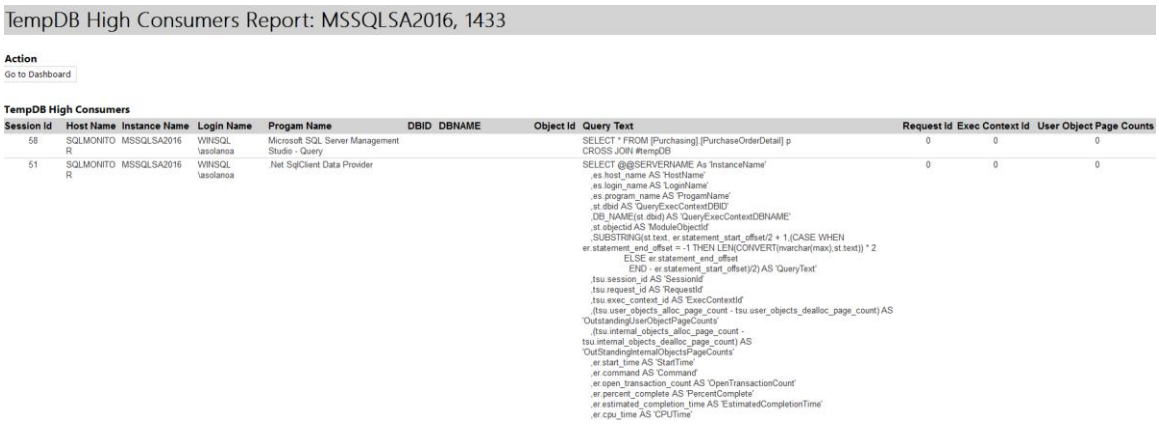
5.8.2.9. ExpensiveQueriesMemoryReport

Este reporte muestra los datos relacionados con las consultas costosas para la memoria del servidor, también utiliza una conexión dinámica para consultar los datos al momento.

Figura 27: Reporte Consulta Costosas Memoria. Fuente: Tomada del servidor de pruebas SQLMonitor

5.8.2.10. TempDBHighConsumersReport

Este reporte se conecta de forma dinámica y consulta los datos que pueden consumir mayores recursos de la base de datos TempDB. A continuación, se mostraron las imágenes que están divididas en dos secciones.



TempDB High Consumers Report: MSSQLSA2016, 1433

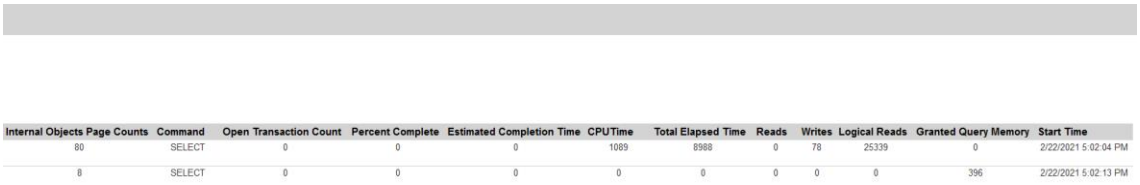
Action

Go to Dashboard

TempDB High Consumers

Session Id	Host Name	Instance Name	Login Name	Progam Name	DBID	DBNAME	Object Id	Query Text	Request Id	Exec Context Id	User Object Page Counts
58	SQLMONITOR	MSSQLSA2016	WINSQL\asolanoa	Microsoft SQL Server Management Studio - Query				SELECT * FROM [Purchasing].[PurchaseOrderDetail] p CROSS JOIN #tempDB	0	0	0
51	SQLMONITOR	MSSQLSA2016	WINSQL\asolanoa	.Net SqlClient Data Provider				SELECT @@SERVERNAME As 'InstanceName' ,es.host_name AS 'HostName' ,es.login_name AS 'LoginName' ,es.program_name AS 'ProgamName' ,st.dbid AS 'QueryExecContextDBID' ,DB_NAME(st.dbid) AS 'QueryExecContextDBNAME' ,st.objectid AS 'ModuleObjectId' ,SUBSTRING(st.text, er.statement_start_offset/2 + 1,(CASE WHEN er.statement_end_offset = -1 THEN LEN(CONVERT(nvarchar(max),st.text)) * 2 ELSE er.statement_end_offset END - er.statement_start_offset)/2) AS 'QueryText' ,tsu.session_id AS 'SessionId' ,tsu.request_id AS 'RequestId' ,tsu.exec_context_id AS 'ExecContextId' ,(tsu.user_objects_alloc_page_count - tsu.user_objects_dealloc_page_count) AS 'OutstandingUserObjectPageCounts' ,(tsu.internal_objects_alloc_page_count - tsu.internal_objects_dealloc_page_count) AS 'OutStandingInternalObjectsPageCounts' ,er.start_time AS 'StartTime' ,er.command AS 'Command' ,er.open_transaction_count AS 'OpenTransactionCount' ,er.percent_complete AS 'PercentComplete' ,er.estimated_completion_time AS 'EstimatedCompletionTime' ,er.cpu_time AS 'CPUTime'	0	0	0

Figura 28: Reporte Altos Consumidores de TempDB 1. Fuente: Tomada del servidor de pruebas SQLMonitor



Internal Objects Page Counts	Command	Open Transaction Count	Percent Complete	Estimated Completion Time	CPUTime	Total Elapsed Time	Reads	Writes	Logical Reads	Granted Query Memory	Start Time
80	SELECT	0	0	0	1089	8988	0	78	25339	0	2/22/2021 5:02:04 PM
8	SELECT	0	0	0	0	0	0	0	0	396	2/22/2021 5:02:13 PM

Figura 29: Reporte Altos Consumidores de TempDB 2. Fuente: Tomada del servidor de pruebas SQLMonitor

5.8.2.11. BlockingReport

Tal vez, este es uno de los reportes más importantes, el cual evitó realizar una conexión directa a la instancia de SQL Server y mostró los bloqueos existentes dentro de las instancias SQL Server, que ayuda a visualizar los posibles problemas de rendimiento. Cabe destacar que esta consulta muestra las transacciones que se están bloqueando.

Blocking Report: MSSQLSA2016, 1433

Action

Go to Dashboard

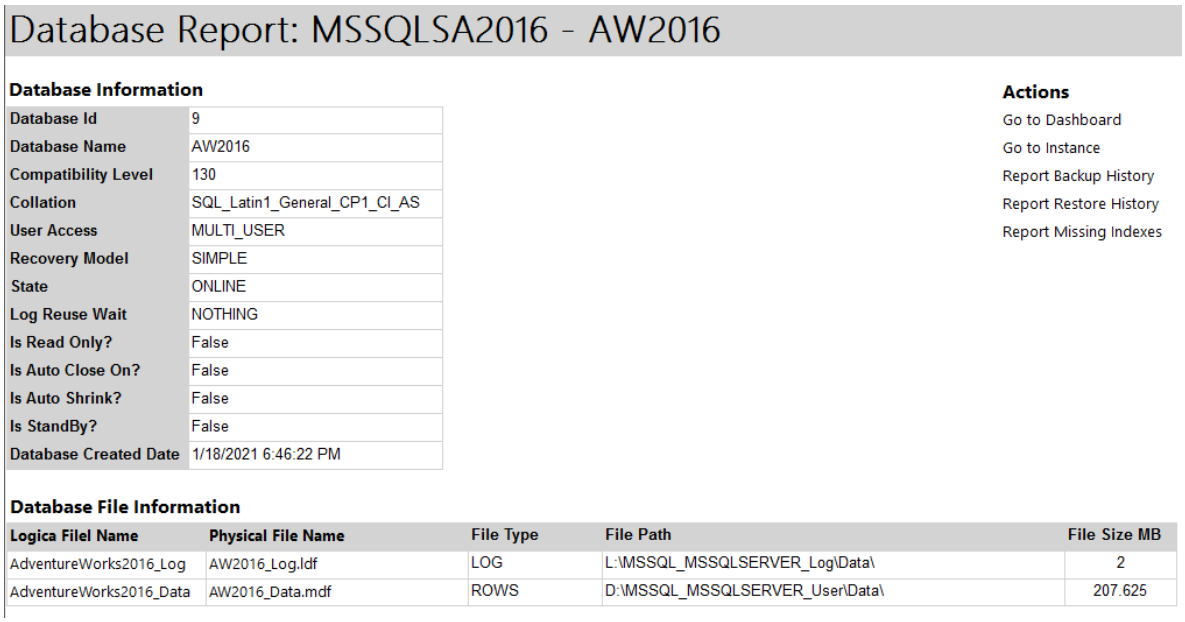
Blocking Report

Blocked SPID	Blocked Login	Blocked Script	Blocking SPID	Blocking Login	Blocking Script	Database	Parent Object	Lock Type	Request Status	Wait Duration ms	Wait Type	Blocking Resource Detail
60	WINSQL\asolanoa	SELECT * FROM [Purchasing].[PurchaseOrderDetail] WITH (TABLOCKX, HOLDLOCK	58	WINSQL\asolanoa	--SELECT * --FROM [Purchasing].[PurchaseOrderDetail] SELECT * FROM [Purchasing].[PurchaseOrderDetail] p CROSS JOIN #tempDB t --ON t.PurchaseOrderID = p.PurchaseOrderID	AW2016		X	WAIT	18371	LCK_M_X	objectlock lockPartition=0 objid=1170103209 subresource=FULL dbid=9 id=lock23657ceac80 mode=IS associatedObjectId=1170103209
51	WINSQL\asolanoa	UPDATE [Purchasing].[PurchaseOrderDetail] SET RecevedQty = 3 WHERE OrderQty =	60	WINSQL\asolanoa	BEGIN TRANSACTION SELECT * FROM [Purchasing].[PurchaseOrderDetail] WITH (TABLOCKX, HOLDLOCK) WAITFOR DELAY '00:05:00' -- 5 minutes commit	AW2016		IX	WAIT	17364	LCK_M_IX	objectlock lockPartition=0 objid=1170103209 subresource=FULL dbid=9 id=lock23657ceac80 mode=IS associatedObjectId=1170103209

Figura 30: Reporte de Bloqueos. Fuente: Tomada del servidor de pruebas SQLMonitor

5.8.2.12. DatabaseReport

Este reporte se divide en dos partes: la primera que muestra la información de la base de datos, la segunda se refiere a los archivos relacionados con la base de datos y es importante mencionar que en la sección “Actions” existe un listado de reportes extra para visualizar la información de los respaldos, restauraciones y de los índices perdidos.



Database Report: MSSQLSA2016 - AW2016

Database Information

Database Id	9
Database Name	AW2016
Compatibility Level	130
Collation	SQL_Latin1_General_CP1_CI_AS
User Access	MULTI_USER
Recovery Model	SIMPLE
State	ONLINE
Log Reuse Wait	NOTHING
Is Read Only?	False
Is Auto Close On?	False
Is Auto Shrink?	False
Is StandBy?	False
Database Created Date	1/18/2021 6:46:22 PM

Actions

Go to Dashboard
Go to Instance
Report Backup History
Report Restore History
Report Missing Indexes

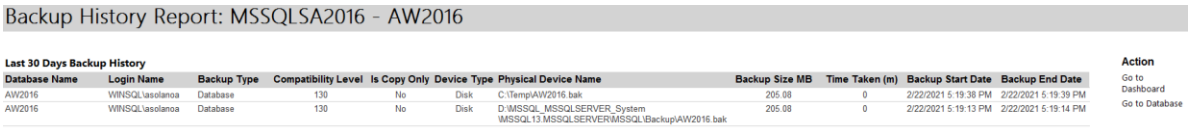
Database File Information

Logica Filel Name	Physical File Name	File Type	File Path	File Size MB
AdventureWorks2016_Log	AW2016_Log.ldf	LOG	L:\MSSQL_MSSQLSERVER_Log\Data\	2
AdventureWorks2016_Data	AW2016_Data.mdf	ROWS	D:\MSSQL_MSSQLSERVER_User\Data\	207.625

Figura 31: Reporte Bases de Datos. Fuente: Tomada del servidor de pruebas SQLMonitor

5.8.2.13. BackupHistoryReport

Este reporte mostró el listado de los respaldos por base de datos que fueron capturados en los últimos 30 días.



Backup History Report: MSSQLSA2016 - AW2016

Last 30 Days Backup History

Database Name	Login Name	Backup Type	Compatibility Level	Is Copy Only	Device Type	Physical Device Name	Backup Size MB	Time Taken (m)	Backup Start Date	Backup End Date
AW2016	WINSQL\asolanoa	Database	130	No	Disk	C:\Temp\AW2016.bak	205.08	0	2/22/2021 5:19:38 PM	2/22/2021 5:19:39 PM
AW2016	WINSQL\asolanoa	Database	130	No	Disk	D:\MSSQL_MSSQLSERVER_System \MSSQL13.MSSQLSERVER\MSSQL\Backup\AW2016.bak	205.08	0	2/22/2021 5:19:13 PM	2/22/2021 5:19:14 PM

Action

Go to Dashboard
Go to Database

Figura 32: Reporte Historial de Respaldos. Fuente: Tomada del servidor de pruebas SQLMonitor

5.8.2.14. RestoreHistoryReport

Este reporte señaló las restauraciones de SQL Server en los últimos 30 días que han sido para ejecutados en la base de datos.


Restore History Report: MSSQLSA2016 - AW2016

Last 30 Days Restore History

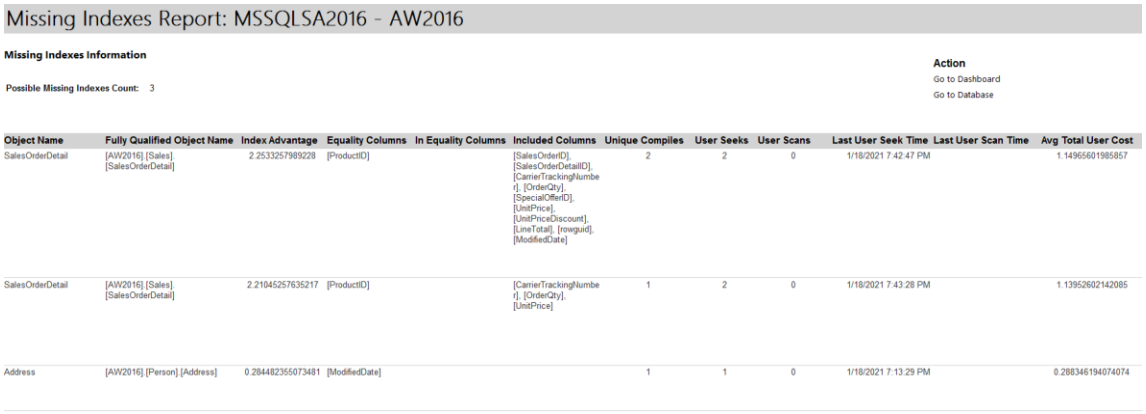
Database Name	Login Name	Restore Type	Replace	Recovery	Backup File Location	Destination Data File	Destination Log File	Restore Date
AW2016	WINSQL\asolanoa	Database	Yes	Yes	C:\Temp\AdventureWorks2016.bak	D:\MSSQL_MSSQLSERVER_User\Data\AW2016_Data.mdf	L:\MSSQL_MSSQLSERVER_Log\Data\AW2016_Log.ldf	2/22/2021 5:19:03 PM

Action
Go to Dashboard
Go to Database

Figura 33: Reporte Historial de Restauraciones. Fuente: Tomada del servidor de pruebas SQLMonitor

5.8.2.15. MissingIndexesReport

El reporte de los índices faltantes es uno de los reportes que puede beneficiar a los usuarios, en donde se mostró la información de los índices que pueden ser implementados dentro de una base de datos para mejorar el rendimiento.


Missing Indexes Report: MSSQLSA2016 - AW2016

Missing Indexes Information

Possible Missing Indexes Count: 3

Action
Go to Dashboard
Go to Database

Object Name	Fully Qualified Object Name	Index Advantage	Equality Columns	In Equality Columns	Included Columns	Unique Compiles	User Seeks	User Scans	Last User Seek Time	Last User Scan Time	Avg Total User Cost
SalesOrderDetail	[AW2016].[Sales].[SalesOrderDetail]	2.2533257989228	[ProductID]		[SalesOrderID], [SalesOrderDetailID], [CarrierTrackingNumber], [OrderQty], [SpecialOfferID], [UnitPrice], [UnitPriceDiscount], [LineTotal], [rowguid], [ModifiedDate]	2	2	0	1/18/2021 7:42:47 PM		1.14965601985857
SalesOrderDetail	[AW2016].[Sales].[SalesOrderDetail]	2.21045257635217	[ProductID]		[CarrierTrackingNumber], [OrderQty], [UnitPrice]	1	2	0	1/18/2021 7:43:28 PM		1.13952602142085
Address	[AW2016].[Person].[Address]	0.284482355073481	[ModifiedDate]			1	1	0	1/18/2021 7:13:29 PM		0.288346194074074

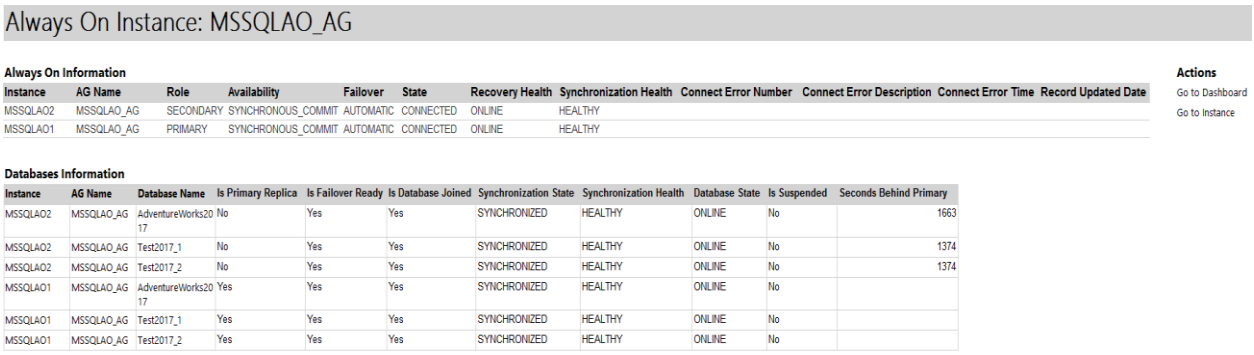
Figura 34: Reporte Índices Faltantes 1. Fuente: Tomada del servidor de pruebas SQLMonitor

Avg User Impact	Proposed Index	Updated Date
98	CREATE INDEX IX_SalesOrderDetail_[ProductID]_D95BC ON [AdventureWorks2016].[Sales].[SalesOrderDetail] ([ProductID]) INCLUDE ([SalesOrderID], [SalesOrderDetailID], [CarrierTrackingNumber], [OrderQty], [SpecialOfferID], [UnitPrice], [UnitPriceDiscount], [LineTotal], [rowguid], [ModifiedDate])	
96.99	CREATE INDEX IX_SalesOrderDetail_[ProductID]_BB426 ON [AdventureWorks2016].[Sales].[SalesOrderDetail] ([ProductID]) INCLUDE ([CarrierTrackingNumber], [OrderQty], [UnitPrice])	
98.66	CREATE INDEX IX_Address_[ModifiedDate]_22785 ON [AW2016].[Person].[Address] ([ModifiedDate])	

Figura 35: Reporte Índices Faltantes 2. Fuente: Tomada del servidor de pruebas SQLMonitor

5.8.2.16. AOInstanceReport

Este reporte se divide en dos secciones, la primera muestra el estado de la replicación de “Always On” a nivel de instancia y la segunda un listado de las bases de datos que forman parte de la replicación.



Always On Instance: MSSQLAO_AG

Always On Information

Instance	AG Name	Role	Availability	Failover	State	Recovery Health	Synchronization Health	Connect Error Number	Connect Error Description	Connect Error Time	Record Updated Date
MSSQLAO2	MSSQLAO_AG	SECONDARY	SYNCHRONOUS_COMMIT	AUTOMATIC	CONNECTED	ONLINE	HEALTHY				
MSSQLAO1	MSSQLAO_AG	PRIMARY	SYNCHRONOUS_COMMIT	AUTOMATIC	CONNECTED	ONLINE	HEALTHY				

Actions
Go to Dashboard
Go to Instance

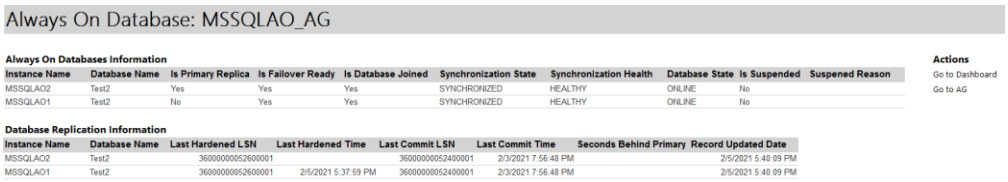
Databases Information

Instance	AG Name	Database Name	Is Primary Replica	Is Failover Ready	Is Database Joined	Synchronization State	Synchronization Health	Database State	Is Suspended	Seconds Behind Primary
MSSQLAO2	MSSQLAO_AG	AdventureWorks2017	No	Yes	Yes	SYNCHRONIZED	HEALTHY	ONLINE	No	1663
MSSQLAO2	MSSQLAO_AG	Test2017_1	No	Yes	Yes	SYNCHRONIZED	HEALTHY	ONLINE	No	1374
MSSQLAO2	MSSQLAO_AG	Test2017_2	No	Yes	Yes	SYNCHRONIZED	HEALTHY	ONLINE	No	1374
MSSQLAO1	MSSQLAO_AG	AdventureWorks2017	Yes	Yes	Yes	SYNCHRONIZED	HEALTHY	ONLINE	No	
MSSQLAO1	MSSQLAO_AG	Test2017_1	Yes	Yes	Yes	SYNCHRONIZED	HEALTHY	ONLINE	No	
MSSQLAO1	MSSQLAO_AG	Test2017_2	Yes	Yes	Yes	SYNCHRONIZED	HEALTHY	ONLINE	No	

Figura 36: Reporte Instancias Always On. Fuente: Tomada del servidor de pruebas SQLMonitor

5.8.2.17. AODatabaseReport

Por último, este reporte mostró el estado de la replicación de las bases de datos que forman parte de los grupos de disponibilidad de “Always On”. El reporte se divide en dos: la primera sección despliega los datos sobre la replicación de las bases de datos y la segunda sección muestra información sobre las restauraciones que se han realizada para continuar con la replicación.



Always On Database: MSSQLAO_AG

Always On Databases Information

Instance Name	Database Name	Is Primary Replica	Is Failover Ready	Is Database Joined	Synchronization State	Synchronization Health	Database State	Is Suspended	Suspened Reason
MSSQLAO2	Test2	Yes	Yes	Yes	SYNCHRONIZED	HEALTHY	ONLINE	No	
MSSQLAO1	Test2	No	Yes	Yes	SYNCHRONIZED	HEALTHY	ONLINE	No	

Actions
Go to Dashboard
Go to AG

Database Replication Information

Instance Name	Database Name	Last Hardened LSN	Last Hardened Time	Last Commit LSN	Last Commit Time	Seconds Behind Primary	Record Updated Date
MSSQLAO2	Test2	36000000052600001		36000000052400001	2/3/2021 7:56:48 PM		2/5/2021 5:40:09 PM
MSSQLAO1	Test2	36000000052600001	2/5/2021 5:37:59 PM	36000000052400001	2/3/2021 7:56:48 PM		2/5/2021 5:40:09 PM

Figura 37: Reporte Bases de Datos Always On. Fuente: Tomada del servidor de pruebas SQLMonitor

Capítulo 6. Análisis de Resultados

Como resultado de la investigación y durante el desarrollo de proceso de monitoreo se definieron las bases necesarias para monitorear las instancias y las bases de datos de SQL Server más importante o al menos las más útiles para proveer mayor facilidad a los administradores de bases de datos durante sus trabajos al momento de revisar el rendimiento de los servidores y obtener información desde un inventario para así manejar las consultas y la carga de trabajo más sencilla, además de tener un control sobre los sistemas los cuales son soportados por el equipo de administración de bases de datos.

Una vez que se conocieron las necesidades de los administradores de bases de datos se realizó un análisis de los datos que fueron necesarios para completar el desarrollo de este proyecto, como las vistas o tablas de sistemas de SQL Server para obtener información sobre las instancias y las bases de datos o en otros casos, utilizando consultas en PowerShell para extraer los datos a nivel del sistema operativo.

Después del análisis sobre los datos imprescindibles para alimentar el proceso, se creó un almacén de datos en el cual se guardó toda la información extraída de las instancias y los sistemas operativos de SQL Server, en donde se siguieron las mejores prácticas para crear una base de datos que fue estandarizada para que su uso fuera sencillo, por lo mismo se crearon dos esquemas en los cuales se utilizaron para diferentes objetivos en el proceso de monitoreo, cabe destacar que se crearon también vistas para facilitar las consultas de los reportes, procedimientos almacenados para facilitar la manipulación de los datos durante la carga de los mismos con la base de datos, también procedimientos que filtran y funcionan de forma especial para el envío de notificación.

Seguidamente, se creó un proceso de extracción, transformación y carga de datos para alimentar el almacén de datos; ya al determinar las fuentes se tomaron varias decisiones para conocer la mejor opción para crear dicho proceso. En la extracción de los datos de SQL Server la mejor opción fue crear los paquetes de “Integration Services” para el consumo y la carga de los datos, en cuanto a los datos

que llegaron desde el sistema operativo Windows la mejor opción fue utilizar un proceso por medio consultas PowerShell.

Una vez que la base de datos fue cargada y los reportes fueron desarrollados, se realizaron pruebas para verificar el funcionamiento y la disponibilidad de los datos; a ventaja de un proyecto como este es que pueden ser modificados y actualizados constantemente para la mejora continua, sin duda alguna revisando el proceso en un ambiente controlado los datos fueron útiles y funcionales, sin embargo, el comportamiento de estos puede variar en un ambiente de producción. No obstante, la información y el uso de la reportería además de las alertas facilita la distribución del trabajo para enfocar esfuerzos en otros objetivos.

Capítulo 7. Conclusiones y Recomendaciones

7.1. Conclusiones

Se definió las necesidades de los administradores de bases de datos para monitorear instancias y bases de datos en Microsoft SQL Server por medio de un análisis y conversaciones con diferentes equipos de bases de datos que dieron a conocer los puntos más importantes según su perspectiva que debería de contener un sistema de monitoreo de instancias y bases de datos SQL Server.

En cuanto a la comprensión de los datos que son necesarios para extraer desde cada instancia de Microsoft SQL Server y Windows, partiendo de la definición de datos se llevó a cabo una ardua investigación para obtener las vistas y tablas de sistema en SQL Server y los comandos necesarios en PowerShell para la obtención de los datos que fueron incluidos en el sistema de monitoreo de bases de datos.

Se elaboró el diseño del almacén de datos en donde se agregó la información, al utilizar las mejores prácticas y por medio del lenguaje de programación de T-SQL se crearon las relaciones que debían tener las diferentes tablas para así lograr un diseño viable que fue utilizado como almacén de datos, al final se creó un diagrama por medio de la herramienta SQL Server.

Uno de los objetivos fue el desarrollo del proceso de extracción, transformación y carga de datos (ETL), en el cual se utilizaron las mismas herramientas incluidas SQL Server mediante el servicio de Integration Services y PowerShell se crearon diferentes procesos de ETL los cuales alimentaron el almacén de datos.

Además, se examinó el funcionamiento del proceso de extracción, transformación y carga de datos, por medio de pruebas al utilizar diferentes ambientes, versiones de SQL Server, cantidad de datos y reportes, se validó que el proceso fuera funcional y operativo y cumpliera las funciones necesarias para monitorear ambientes de bases de datos.

Parte fundamental de este proyecto fue la creación de un ambiente de pruebas que pudiese replicar diferentes escenarios que pueden existir en un ambiente de producción para así validar el buen funcionamiento del proceso, esto fue aplicado con la creación de un servidor de Active Directory para así simular la

creación de usuarios y el dominio de un ambiente real, seguidamente de la instalación de diferentes versiones de SQL Server para verificar las versiones aceptadas por el proceso y por último la creación de un ambiente de alta disponibilidad que utilizara las tecnologías ofrecidas por Microsoft como son el Windows Server Failover Clustering y el SQL Server Always On.

La utilización de estándares y mejores prácticas durante el desarrollo de este proyecto fueron fundamentales para la normalización y estandarización de los diferentes componentes que forman parte del proceso de monitoreo para las instancias y las bases de datos de SQL Server, de esta forma permitiendo que el proceso sea sencillo de entender para los diferentes usuarios involucrados en su implementación y uso.

Se crearon diferentes pruebas para la validación de los datos recibidos por el sistema de monitoreo, de manera, que se probaron diferentes escenarios en donde los datos cambiaban y así se realizó una revisión del sistema de reportería y las alertas recibidas por medio de correo electrónico, además, de la revisión de las consultas que realizaban conexiones dinámicas verificando el rendimiento de los diferentes servidores.

Por último, sin duda alguna este proceso busca potenciar la utilización de SQL Server para así encontrar la forma en la cual se pueda aprovechar al máximo la licencia de SQL Server, duranta la utilización de este proceso de monitoreo evitaría la inversión de una herramienta de terceros, sin embargo, su uso es limitado ya que no ofrece la complejidad que ofrecen las herramientas de un proveedor, no obstante, el proceso y el sistema de reportería muestran datos que son útiles para la revisión de las diferentes instancias y bases de datos de SQL Server y así facilitar la toma de acciones para proveer un soporte preventivo y de esta forma facilitar el trabajo diario de un administrador de bases de datos.

7.2. Recomendaciones

Como recomendación es importante implementar el proyecto en una versión de SQL Server 2019, tomando en cuenta que se debe utilizar una licencia estándar o empresarial, ya que el mismo debe incluir servicio de Integration Services y el

sistema operativo debe de ser Windows con una versión superior a Windows Server 2012.

No obstante, el proyecto puede ser creado y desarrollado en cualquier otra versión de SQL Server siempre y cuando se apliquen las consultas y genere la base de datos en una versión superior a SQL Server 2008 R2, ya que esta versión tiene estructura diferente a las versiones de SQL Server 2012 o superiores.

Debido a que el proyecto fue desarrollo en la última versión de SQL Server, en este momento se descarta la implementación del proceso en una versión superior a SQL Server 2019.

Es importante mencionar como buena práctica crear una cuenta de servicio que tenga los permisos necesarios para ejecutar la extracción de datos y los reportes, una cuenta única permite el control de los servicios y del proceso.

Se recomienda permitir la comunicación del servidor de monitoreo con los otros servidores de bases de datos SQL Server.

Por último, es importante validar la calendarización de los paquetes de Integration Services para evitar la degradación de los diferentes ambientes que sean incluidos dentro del proceso de monitoreo.

Capítulo 8. Reflexiones Finales

Basado en la experiencia y escenarios existentes para las pequeñas y medianas empresas donde la inversión de una herramienta muy completa puede llegar a ser un capricho para el alcance de la organización, un proyecto como este basado en el aprovechamiento del uso de la licencia de SQL Server que ya fue pagada, beneficia a las organizaciones al evitar una inversión extra de un sistema de terceros para el monitoreo.

Algo tan simple como el monitoreo de la replicación de SQL Server en ambientes con “Always On” o recibir una alerta sobre el estado de los diferentes servicios utilizados de SQL Server, puede evitar cualquier inconveniente y evita que los administradores de bases de datos de forma reactiva esperen la notificación de las organizaciones de las diferentes empresas.

Capítulo 9. Trabajos a Futuro

A continuación, se describe una lista de los posibles trabajos a futuro que pueden impulsar y mejorar este proyecto en un nivel superior:

- Análisis de datos para la identificación de futuras fallas.
- Creación de tablas e interfaz para facilidad de modificación de parámetros.
- Creación de validación de errores y registros de ejecución más precisos.
- Implementar la base de datos SQLMonitor en un ambiente de alta disponibilidad.
- Según el lanzamiento de nuevas versiones de SQL Server analizar si el proceso de monitoreo puede ser implementado.
- Generación de reportes con datos de interés para el negocio donde el enfoque no solo sea una persona con habilidades técnicas en SQL Server.
- Creación de base de datos de conocimientos la cual facilite la solución de problemas a una persona que no sea experta en SQL Server.

Referencias

Chaudhuri, S., Konig, A., & Narasayya, V. (2004). *SQLCM: a continuous monitoring framework for relational database engines.* Obtenido de ieeexplore.ieee.org: https://ieeexplore.ieee.org/abstract/document/1320020

Database Files and Filegroups. (29 de 05 de 2020). Obtenido de https://docs.microsoft.com/: https://docs.microsoft.com/en-us/sql/relational-databases/databases/database-files-and-filegroups?view=sql-server-ver15

Davis, T. (14 de 07 de 2013). *Eight Steps to Effective SQL Server Monitoring*. Obtenido de www.red-gate.com: https://www.red-gate.com/simple-talk/sql/database-administration/eight-steps-to-effective-sql-server-monitoring/

Fritchey, G. (2018). *SQL Server 2017 Query Performance Tunning.* Massachusetts: Apress.

Hernández, R., Fernández, C., & Baptista, P. (2014). *Metodología de la investigación.* México D.F: McGrawHill.

Idera. (s.f.). *Idera Online Store*. Obtenido de www.idera.com: https://www.idera.com/buynow/onlinestore

Jorgensen, A., Wort, S., LoForte, R., & Knight, B. (2012). *Professional Microsoft SQL Server 2012 Administration.* Indianapolis: John Wiley & Sons, Inc.

Kehayias, J., & Krueger, T. (2011). *Troubleshooting SQL Server: A Guide for the Accidental DBA.* Simple Talk Publishing.

Korotkevitch, D. (2016). *Pro SQL Server Internals.* Florida: Apress.

Mateen, A., Raza, B., & Awais, M. (2008). *Autonomic computing in SQL Server.* Obtenido de computer.org: https://www.computer.org/csdl/proceedings-article/icis/2008/3131a113/12OmNxeM49Y

Microsoft. (04 de 06 de 2013). *Database Lifecycle Management (DLM)*. Obtenido de docs.microsoft.com: https://docs.microsoft.com/es-es/previous-versions/sql/sql-server-guides/jj907294(v=sql.110)?redirectedfrom=MSDN

Microsoft. (18 de 07 de 2016). *Monitor and Tune for Performance*. Obtenido de dosc.microsoft.com: ttps://docs.microsoft.com/en-us/sql/relational-databases/performance/monitor-and-tune-for-performance?view=sql-server-ver15

Microsoft. (10 de 11 de 2016). *msdb Database*. Obtenido de docs.microsoft.com: https://docs.microsoft.com/en-us/sql/relational-databases/databases/msdb-database?view=sql-server-ver15

Microsoft. (18 de 01 de 2017). *Always On Failover Cluster Instances (SQL Server)*. Obtenido de www.docs.microsoft.com: https://docs.microsoft.com/en-us/sql/sql-server/failover-clusters/windows/always-on-failover-cluster-instances-sql-server?view=sql-server-ver15

MIcrosoft. (03 de 02 de 2017). *Configure the cost threshold for parallelism Server Configuration Option*. Obtenido de docs.microsoft.com: https://docs.microsoft.com/en-us/sql/database-engine/configure-windows/configure-the-cost-threshold-for-parallelism-server-configuration-option?view=sql-server-ver15

Microsoft. (14 de 03 de 2017). *Monitoring the Error Logs.* Obtenido de docs.microsoft.com: https://docs.microsoft.com/en-us/sql/tools/configuration-manager/monitoring-the-error-logs?view=sql-server-ver15

Microsoft. (24 de 05 de 2017). *Server-Level Roles*. Obtenido de www.docs.microsoft.com: https://docs.microsoft.com/en-us/sql/relational-databases/security/authentication-access/server-level-roles?view=sql-server-ver15#:~:text=Members%20of%20the%20processadmin%20fixed,an%20ins

tance%20of%20SQL%20Server.&text=Members%20of%20the%20setupadmin%20f

MIcrosoft. (19 de 01 de 2017). *SQL Server Agent*. Obtenido de www.SQL Server Agent: https://docs.microsoft.com/en-us/sql/ssms/agent/sql-server-agent?view=sql-server-ver15

Microsoft. (02 de 11 de 2019). *Clustered and Nonclustered Indexes Described*. Obtenido de www.docs.microsoft.com: https://docs.microsoft.com/en-us/sql/relational-databases/indexes/clustered-and-nonclustered-indexes-described?view=sql-server-ver15

Microsoft. (13 de 12 de 2019). *Editions and supported features of SQL Server*. Obtenido de docs.microsoft.com: https://docs.microsoft.com/en-us/sql/sql-server/editions-and-components-of-sql-server-2017?view=sql-server-ver15

Microsoft. (26 de 07 de 2019). *Install SQL Server Database Engine*. Obtenido de www.docs.microsoft.com: https://docs.microsoft.com/en-us/sql/database-engine/install-windows/install-sql-server-database-engine?view=sql-server-ver15#:~:text=The%20Database%20Engine%20component%20of,consuming%20applications%20in%20your%20enterprise.

Microsoft. (28 de 01 de 2019). *master Database*. Obtenido de docs.microsoft.com: https://docs.microsoft.com/en-us/sql/relational-databases/databases/master-database?view=sql-server-ver15

Microsoft. (28 de 01 de 2019). *System Databases*. Obtenido de docs.microsoft.com: https://docs.microsoft.com/en-us/sql/relational-databases/databases/system-databases?view=sql-server-ver15

Microsoft. (02 de 12 de 2020). *Configure the max degree of parallelism Server Configuration Option*. Obtenido de docs.microsoft.com: https://docs.microsoft.com/en-us/sql/database-engine/configure-windows/configure-the-max-degree-of-parallelism-server-configuration-option?view=sql-server-ver15#Recommendations

Microsoft. (22 de 07 de 2020). *model Database*. Obtenido de docs.microsoft.com: https://docs.microsoft.com/en-us/sql/relational-databases/databases/model-database?view=sql-server-ver15

Microsoft. (17 de 04 de 2020). *tempdb Database*. Obtenido de docs.microsoft.com: https://docs.microsoft.com/en-us/sql/relational-databases/databases/tempdb-database?view=sql-server-ver15

Microsoft. (29 de 04 de 2020). *What is an Always On availability group?* Obtenido de www.docs.microsoft.com: https://docs.microsoft.com/en-us/sql/database-engine/availability-groups/windows/overview-of-always-on-availability-groups-sql-server?view=sql-server-ver15

Microsoft. (22 de 05 de 2020). *What is PowerShell?* Obtenido de docs.microsoft.com: https://docs.microsoft.com/en-us/powershell/scripting/overview?view=powershell-7

Microsoft. (s.f.). *SQL Server 2019 pricing*. Obtenido de www.miscrosoft.com: https://www.microsoft.com/en-us/sql-server/sql-server-2019-pricing

Microsoft SQL Server Versions List. (s.f.). Obtenido de sqlserverbuilds.blogspot.com: https://sqlserverbuilds.blogspot.com/

Pearl, R. (2015). *Healthy SQL.* Apress.

RedGate. (s.f.). *Redgate SQL Monitor*. Obtenido de www.red-gate.com: https://www.red-gate.com/dynamic/purchase/product/sqlmonitor

Anexos

Anexo 1. Diccionario de Datos

Índice Diccionario de Datos

1. Tablas pertenecientes al esquema monitor

1.1. Tabla monitor.tAODatabase

Nombre	tAODatabase	**Esquema**	monitor
Definición	Tabla que contiene los datos relacionados con las bases de datos que forman parte de una configuración de “Always On”.		
Nombre Columna	**Tipo de Dato**	**Longitud**	**Descripción**
ReplicaId	uniqueidentifier	16	Identificador único para la réplica.
GroupId	uniqueidentifier	16	Identificador único para el grupo de alta disponibilidad.
GroupDatabaseId	uniqueidentifier	16	Identificador único para las bases de datos del grupo de alta disponibilidad.
AvailabilityGroupName	varchar	50	Nombre del nombre del grupo de alta disponibilidad.
DatabaseName	varchar	100	Nombre de base de datos.
IsPrimaryReplica	varchar	4	Indica si la réplica está en el nodo primario.
IsFailoverReady	varchar	4	Indica si el grupo de alta disponibilidad está listo para realizar un Failover.
IsDatabaseJoined	varchar	4	Indica si la base de datos fue agregada a

			un grupo de alta disponibilidad.
SynchronizationState	varchar	20	Estado de la sincronización entre las réplicas.
SynchronizationStateHealth	varchar	20	Estado de la salud sincronización entre las réplicas.
DatabaseState	varchar	10	Estado de las bases de datos.
IsSuspended	varchar	4	Indica si la sincronización está suspendida.
SuspenedReason	varchar	50	Razón por la cual la sincronización está suspendida.
LastHardenedLSN	numeric	20	Último LSN reforzado en una base de datos secundaria.
LastHardenedTime	datetime	16	Hora del identificador de bloque de registro para el último LSN reforzado.
LastCommitLSN	numeric	20	Número de secuencia de registro real correspondiente al último registro de confirmación en el registro de transacciones.

LastCommitTime	datetime	16	Hora correspondiente al último registro de aplicado.
SecondsBehindPrimary	int	4	Tiempo en segundos que la réplica secundaria está por detrás de la réplica primaria durante la sincronización.
RecordCreatedDate	datetime	16	Fecha en la que se generó el registro.
RecordUpdatedDate	datetime	16	Fecha en la que se actualizó el registro.

1.2. Tabla monitor.tAOInstance

Nombre	tAOInstance	**Esquema**	monitor
Definición	Tabla que contiene los datos relacionados a las instancias que contienen una configuración de "Always On".		
Nombre Columna	**Tipo de Dato**	**Longitud**	**Descripción**
ReplicaId	uniqueidentifier	16	Identificador único para la réplica.
GroupId	uniqueidentifier	16	Identificador único para el grupo de alta disponibilidad.
AvailabilityGroupName	varchar	50	Nombre del nombre del grupo de alta disponibilidad
Role	varchar	10	Estado en el cual se encuentra el nodo.

AvailabilityMode	varchar	20	Tipo de sincronización.
FailoverMode	varchar	10	Tipo del Failover.
ConnectedState	varchar	15	Estado de la conexión del grupo de alta disponibilidad.
RecoveryHealth	varchar	10	Estado de la salud de la recuperación.
SynchronizationHealth	varchar	15	Estado de salud de la sincronización.
LastConnectErrorNumber	int	4	Número de la última conexión de error.
LastConnectErrorDescription	varchar	MAX	Mensaje de la última conexión de error.
LastConnectErrorTimeStamp	datetime	16	Fecha y hora cuando falló la conexión.
InstanceId	int	4	Identificador único en donde se encuentra el grupo de alta disponibilidad.
RecordCreatedDate	datetime	16	Fecha en la que se generó el registro.
RecordUpdatedDate	datetime	16	Fecha en la que se actualizó el registro.

1.3. Tabla monitor.tBackupHistory

Nombre	tBackupHistory	**Esquema**	monitor
Definición	Tabla que contiene los datos relacionados al historial de respaldos de las bases de datos.		
Nombre Columna	**Tipo de Dato**	**Longitud**	**Descripción**
BackupHistoryId	int identity	4	Identificador único para cada respaldo.
LoginName	varchar	255	Nombre del usuario que creó el respaldo.
BackupType	varchar	20	Tipo de respaldo.
RecoveryModel	varchar	30	Modelo de recuperación.
CompatibilityLevel	int	4	Nivel de compatibilidad.
Collation	varchar	50	Tipo de colación utilizada.
IsCopyOnly	varchar	5	Indica si se utilizó la opción de “CopyOnly”.
DeviceType	varchar	25	Tipo de dispositivo en donde se almacena el respaldo
PhysicalDeviceName	nvarchar	MAX	Directorio en donde se almacenó el archivo de respaldo.
BackupSizeMB	float	8	Tamaño del archivo de respaldo.
TimeTakenMin	int	4	Duración de tiempo que tomó el proceso de creación del respaldo.
BackupStartDate	datetime	16	Fecha de inicio de creación del respaldo.
BackupEndDate	datetime	16	Fecha final de la creación del respaldo.
DatabaseId	int	4	Identificador único de las bases de datos que fueron respaldadas.

RecordCreatedDate	datetime	16	Fecha en la que se generó el registro.
RecordUpdatedDate	datetime	16	Fecha en la que se actualizó el registro.

1.4. Tabla monitor.tDatabase

Nombre	tDatabase	**Esquema**	monitor
Definición	Tabla que contiene los datos relacionados con la información de cada base de datos.		
Nombre Columna	**Tipo de Dato**	**Longitud**	**Descripción**
DatabaseId	int identity	4	Identificador único para cada base de datos.
InstanceDatabaseId	int	4	Identificador único para cada base de datos.
DatabaseName	varchar	255	Nombre de la base de datos.
CompatibilityLevel	int	4	Nivel de compatibilidad.
Collation	varchar	50	Tipo de colación utilizada.
UserAccess	varchar	25	Nivel de acceso de usuarios.
RecoveryModel	varchar	25	Modelo de recuperación.
State	varchar	15	Estado de la base de datos.
LogReuseWait	varchar	25	Espera de reutilización de registros.
IsReadOnly	bit	1	Indica si la base de datos está en solo lectura.
IsAutoCloseOn	bit	1	Indica si la base de datos tiene habilitada la función “Auto Close”.

IsAutoShrink	bit	1	Indica si la base de datos tiene habilitada la función “Auto Shrink”.
IsStandBy	bit	1	Indica si la base de datos está en “Stand By”.
DatabaseCreatedDate	datetime	16	Fecha de creación de la base de datos.
InstanceId	int	4	Identificador único de la instancia en donde se encuentra la base de datos.
RecordCreatedDate	datetime	16	Fecha en la que se generó el registro.
RecordUpdatedDate	datetime	16	Fecha en la que se actualizó el registro.

1.5. Tabla monitor.tDatabaseFiles

Nombre	tDatabaseFiles	**Esquema**	monitor
Definición	Tabla que contiene los datos relacionados con la información de los archivos pertenecientes a las bases de datos.		
Nombre Columna	**Tipo de Dato**	**Longitud**	**Descripción**
DatabaseFileId	int identity	4	Identificador único para cada uno de los archivos de base de datos.
TypeDescription	varchar	10	Tipo de archivo.
LogicaFilelName	varchar	255	Nombre lógico del archivo.
PhysicalFileName	varchar	255	Nombre físico del archivo.
FilePath	nvarchar	MAX	Directorio en el cual se encuentra los archivos de datos o log.
FileState	varchar	25	Estado del archivo.

FileSizeMB	float	8	Tamaño del archivo.
DatabaseId	int	4	Identificador único de la base de datos a la cual le pertenece el archivo.
RecordCreatedDate	datetime	16	Fecha en la que se generó el registro.
RecordUpdatedDate	datetime	16	Fecha en la que se actualizó el registro.

1.6. Tabla monitor. tDriveSize

Nombre	tDriveSize	**Esquema**	monitor
Definición	Tabla que contiene los datos relacionados con la información del espacio en disco.		
Nombre Columna	**Tipo de Dato**	**Longitud**	**Descripción**
DriveSizeId	int identity	4	Identificador único para cada disco.
DriveLetter	char	2	Letra que representa el disco.
DriveName	varchar	30	Nombre del disco.
FreeSpaceGB	decimal	7	Espacio libre en Gigabytes.
FreeSpacePercentage	decimal	7	Porcentaje de espacio libre.
UsedSpaceGB	decimal	7	Espacio utilizado en Gigabytes.
UsedSpacePercentage	decimal	7	Porcentaje de espacio utilizado.
TotalSizeGB	decimal	7	Tamaño total del disco.
HostId	int	4	Identificador único del host en donde se encuentran los discos.
RecordCreatedDate	datetime	16	Fecha en la que se generó el registro.

RecordUpdatedDate	datetime	16	Fecha en la que se actualizó el registro.

1.7. Tabla monitor. tHost

Nombre	tHost		**Esquema** monitor
Definición	Tabla que contiene los datos relacionados con la información del host.		
Nombre Columna	**Tipo de Dato**	**Longitud**	**Descripción**
HostId	int	4	Identificador único del host.
HostName	varchar	100	Nombre del host.
Environment	varchar	15	Tipo de ambiente.
WindowsEdition	varchar	255	Edición de Windows
WindowsVersion	varchar	50	Versión de Windows.
PhysicalCores	float	8	Cantidad de núcleos en el procesador.
LogicalCores	float	8	Cantidad de núcleos lógicos en el procesador.
TotalMemoryGB	float	8	Cantidad de memoria total en el servidor.
RecordCreatedDate	datetime	16	Fecha en la que se generó el registro.
RecordUpdatedDate	datetime	16	Fecha en la que se actualizó el registro.

1.8. Tabla monitor. tInstance

Nombre	tInstance	**Esquema**	monitor
Definición	Tabla que contiene los datos relacionados con la información de la instancia.		

Nombre Columna	Tipo de Dato	Longitud	Descripción
InstanceId	int	4	Identificador único para las instancias.
InstanceName	varchar	100	Nombre de la instancia.
Collation	varchar	50	Tipo de colación.
TcpPort	char	4	Puerto de la instancia.
Version	varchar	100	Versión de SQL Server.
Edition	varchar	30	Edición de SQL Server.
Environment	nvarchar	30	Tipo de ambiente.
DefaultDataFilePath	nvarchar	MAX	Directorio en el cual se encuentran los archivos de datos (.mdf y/o .ndf).
DefaultLogPath	nvarchar	MAX	Directorio en el cual se encuentran los archivos de log (.log).
IsAlwaysOnAGEnable	varchar	25	Indica si la configuración de “Always On” está habilitada.
AuthenticationMode	varchar	50	Modo de autenticación.
HostId	int	4	Identificador único del host en donde se encuentran instaladas las instancias de SQL Server.
RecordCreatedDate	datetime	16	Fecha en la que se generó el registro.
RecordUpdatedDate	datetime	16	Fecha en la que se actualizó el registro.

1.9. Tabla monitor. tInstanceConfiguration

Nombre	tInstanceConfiguration	Esquema	monitor

Definición	Tabla que contiene los datos relacionados con la información de las configuraciones avanzadas de las instancias.		
Nombre Columna	**Tipo de Dato**	**Longitud**	**Descripción**
InstanceConfigurationId	int identity	4	Identificador único para la configuración avanzada de las instancias.
ConfigurationId	int	4	Identificador de las configuraciones.
ConfigurationName	varchar	30	Nombre de la configuración.
ConfigurationDescription	varchar	48	Descripción de la configuración.
Value	int	4	Valor configurado.
InstanceId	int	4	Identificador único de la instancia en donde se encuentra la configuración.
RecordCreatedDate	datetime	16	Fecha en la que se generó el registro.
RecordUpdatedDate	datetime	16	Fecha en la que se actualizó el registro.

1.10. Tabla monitor. tJobStatus

Nombre	tJobStatus	**Esquema**	monitor
Definición	Tabla que contiene los datos relacionados con la información de los “jobs".		
Nombre Columna	**Tipo de Dato**	**Longitud**	**Descripción**
JobStatusId	int identity	4	Identificador único del “job”.

JobId	uniqueidentifier	16	Identificador único del “job” en la instancia.
JobName	varchar	255	Nombre del “job”.
JobOwner	varchar	100	Nombre del dueño del “job”.
IsEnabled	varchar	4	Indica si el “job” está habilitado.
StepName	varchar	100	Nombre del “step” del “job”.
Message	nvarchar	MAX	Último mensaje del “job”.
RunStatus	varchar	20	Último estado del “job”.
LastRunDurationSeconds	int	4	Última duración de tiempo de ejecución en segundos.
LastRunDuration	varchar	20	Última duración de tiempo de ejecución.
LastRunDate	datetime	16	Última fecha de tiempo de ejecución
JobDateCreated	datetime	16	Fecha de creación.
JobDateModified	datetime	16	Fecha de modificación.
InstanceId	int	4	Identificador único de la instancia en la cual se encuentran los “jobs”.
RecordCreatedDate	datetime	16	Fecha en la que se generó el registro.
RecordUpdatedDate	datetime	16	Fecha en la que se actualizó el registro.

1.11. Tabla monitor.tMissingIndex

Nombre	tMissingIndex	**Esquema**	monitor
Definición	Tabla que contiene los datos relacionados con la información de los posibles índices faltantes.		
Nombre Columna	**Tipo de Dato**	**Longitud**	**Descripción**
MissingIndexId	int identity	4	Identificador único para los índices faltantes.
ObjectName	varchar	100	Nombre del objeto.
FullyQualifiedObjectName	varchar	MAX	Nombre completo del objeto incluyendo el nombre de la base de datos, esquema y nombre de la tabla.
IndexAdvantage	float	8	Ventaja del índice.
EqualityColumns	varchar	MAX	Lista de columnas que contribuyen a las consultas de igualdad.
InEqualityColumns	varchar	MAX	Lista de columnas que contribuyen a las consultas de desigualdad.
IncludedColumns	varchar	MAX	Lista de columnas necesarias como columnas de cobertura para la consulta.
UniqueCompiles	int	4	Número de compilaciones y recopilaciones que se beneficiarían con el índice recomendando.
UserSeeks	int	4	Número de búsquedas provocadas por consultas para las que se podría haber utilizado el índice recomendado.

UserScans	int	4	Número de escaneos provocados por consultas para las que se podría haber utilizado el índice.
LastUserSeekTime	datetime	16	Fecha y hora de la última búsqueda provocada por consultas de usuario para las que podría haberse utilizado el índice recomendado.
LastUserScanTime	datetime	16	Fecha y hora del último análisis provocado por consultas de usuarios para las que se podría haber utilizado el índice faltante.
AvgTotalUserCost	float	8	Costo promedio de las consultas que podría reducirse con el índice.
AvgUserImpact	float	8	Beneficio promedio que las consultas podrían experimentar si se implementara el índice faltante.
ProposedIndex	varchar	MAX	“Script” de creación del índice faltante.
DatabaseId	int	4	Identificador único de la base de datos en donde puede ser implementado el índice.
RecordCreatedDate	datetime	16	Fecha en la que se generó el registro.
RecordUpdatedDate	datetime	16	Fecha en la que se actualizó el registro.

1.12. Tabla monitor.tMonitorHost

Nombre	tMonitorHost		**Esquema**	monitor
Definición	Tabla que contiene los nombres de los diferentes hosts que forman parte del proceso de monitoreo.			
Nombre Columna	**Tipo de Dato**	**Longitud**	**Descripción**	
HostId	int identity	4	Identificador único del host.	
HostName	varchar	255	Nombre del host.	
Environment	varchar	15	Tipo de ambiente.	
IsEnable	bit	1	Indica si está habilitado.	
RecordCreatedDate	datetime	16	Fecha en la que se generó el registro.	
RecordUpdatedDate	datetime	16	Fecha en la que se actualizó el registro.	

1.13. Tabla monitor.tMonitorInstance

Nombre	tMonitorInstance		**Esquema**	monitor
Definición	Tabla que contiene los nombres de las diferentes instancias que forman parte del proceso de monitoreo.			
Nombre Columna	**Tipo de Dato**	**Longitud**	**Descripción**	
InstanceId	int identity	4	Identificador único de la instancia.	
InstanceName	varchar	255	Nombre de la instancia.	
TcpPort	char	4	Puerto de la instancia.	
Environment	varchar	15	Tipo de ambiente.	
HostId	int	4	Identificador único del host en donde está instalado SQL Server.	
IsEnable	bit	1	Indica si está habilitado.	

RecordCreatedDate	datetime	16	Fecha en la que se generó el registro.
RecordUpdatedDate	datetime	16	Fecha en la que se actualizó el registro.

1.14. Tabla monitor.tPrivAccess

Nombre	tPrivAccess	**Esquema**	monitor
Definición	Tabla que contiene los datos relacionados con la información de los altos privilegios que tienen los diferentes usuarios en las instancias.		
Nombre Columna	**Tipo de Dato**	**Longitud**	**Descripción**
LoginId	int identity	4	Identificador único de los usuarios.
LoginName	Varchar	255	Nombre del usuario.
LoginType	Varchar	25	Tipo de usuario.
IsSysadmin	Varchar	5	Indica si el usuario es sysadmin.
IsServeradmin	Varchar	5	Indica si el usuario es serveradmin.
IsSecurityadmin	Varchar	5	Indica si el usuario es securityadmin.
IsProcessadmin	Varchar	5	Indica si el usuario es processadmin.
IsDisabled	Varchar	5	Indica si el usuario está habilitado.
AccountCreatedDate	Datetime	16	Fecha de creación de la cuenta.
AccountUpdatedDate	Datetime	16	Fecha de actualización de la cuenta.

InstanceId	Int	4	Identificador único de la instancia en la cual se encuentran los usuarios.
RecordCreatedDate	Datetime	16	Fecha en la que se generó el registro.
RecordUpdatedDate	Datetime	16	Fecha en la que se actualizó el registro.

1.15. Tabla monitor.tRestoreHistory

Nombre	tRestoreHistory	**Esquema**	monitor
Definición	Tabla que contiene los datos relacionados al historial de restauraciones de las bases de datos.		
Nombre Columna	**Tipo de Dato**	**Longitud**	**Descripción**
RestoreHistoryId	int identity	4	Identificador único para cada restauración.
LoginName	varchar	255	Usuario que restauró la base de datos.
RestoreType	varchar	20	Tipo de restauración.
Replace	varchar	4	Indica si una base de datos fue reemplazada.
Recovery	varchar	4	Modelo de recuperación.
BackupFileLocation	varchar	MAX	Directorio en el cual se encuentra el archivo de respaldo utilizado.
DestinationDataFile	varchar	MAX	Directorio en donde se creó el archivo de datos restaurado.
DestinationLogFile	Varchar	MAX	Directorio en donde se creó el archivo de log restaurado.
RestoreDate	datetime	16	Fecha de la restauración.

DatabaseId	int	4	Identificador único de las bases de datos que fue respaldada.
RecordCreatedDate	datetime	16	Fecha en la que se generó el registro.
RecordUpdatedDate	datetime	16	Fecha en la que se actualizó el registro.

1.16. Tabla monitor.tSQLServiceStatus

Nombre	tSQLServiceStatus	**Esquema**	monitor
Definición	Tabla que contiene los datos relacionados con los servicios de SQL Server.		
Nombre Columna	**Tipo de Dato**	**Longitud**	**Descripción**
SQLServiceId	int identity	4	Identificador único para el servicio de SQL Server.
ServiceName	varchar	50	Nombre del servicio.
StartMode	varchar	50	Modo de inicio del servicio.
Status	varchar	15	Estado del servicio
HostId	int	4	Identificador único del host en donde se encuentra instalado los servicios de SQL Server
RecordCreatedDate	datetime	16	Fecha en la que se generó el registro.
RecordUpdatedDate	datetime	16	Fecha en la que se actualizó el registro.

2. Tablas pertenecientes al esquema staging

2.1. Tabla staging.tAODatabase

Nombre	tAODatabase	**Esquema**	staging

Definición	Tabla que contiene de forma temporal los datos relacionados a las bases de datos que forman parte de una configuración de “Always On”.		
Nombre Columna	**Tipo de Dato**	**Longitud**	**Descripción**
ReplicaId	uniqueidentifier	16	Identificador único para la réplica.
GroupId	uniqueidentifier	16	Identificador único para el grupo de alta disponibilidad.
GroupDatabaseId	uniqueidentifier	16	Identificador único para las bases de datos del grupo de alta disponibilidad.
AvailabilityGroupName	varchar	50	Nombre del nombre del grupo de alta disponibilidad.
DatabaseName	varchar	100	Nombre de la base de datos.
IsPrimaryReplica	varchar	4	Indica si la réplica está en el nodo primario.
IsFailoverReady	varchar	4	Indica si el grupo de alta disponibilidad está listo para realizar un “Failover”.
IsDatabaseJoined	varchar	4	Indica si la base de datos fue agregada a un grupo de alta disponibilidad.

SynchronizationState	varchar	20	Estado de la sincronización entre las réplicas.
SynchronizationStateHealth	varchar	20	Estado de la salud de sincronización entre las réplicas.
DatabaseState	varchar	10	Estado de las bases de datos.
IsSuspended	varchar	4	Indica si la sincronización está suspendida.
SuspenedReason	varchar	50	Razón por la cual la sincronización está suspendida.
LastHardenedLSN	numeric	20	Último LSN reforzado en una base de datos secundaria.
LastHardenedTime	datetime	16	Hora del identificador de bloque de registro para el último LSN reforzado.
LastCommitLSN	numeric	20	Número de secuencia de registro real correspondiente al último registro de confirmación en el registro de transacciones.
LastCommitTime	datetime	16	Hora correspondiente al último registro aplicado.

SecondsBehindPrimary	int	4	Tiempo en segundos en que la réplica secundaria está por detrás de la réplica primaria durante la sincronización.
RecordCreatedDate	datetime	16	Fecha en la que se generó el registro.

2.2. Tabla staging.tAOInstance

Nombre	tAOInstance	**Esquema**	"staging"
Definición	Tabla que contiene de forma temporal los datos relacionados a las instancias que contienen una configuración de "Always On".		
Nombre Columna	**Tipo de Dato**	**Longitud**	**Descripción**
AOInstanceId	int identity	4	Identificador único para la réplica.
ReplicaId	uniqueidentifier	16	Identificador único para el grupo de alta disponibilidad.
GroupId	uniqueidentifier	16	Nombre del nombre del grupo de alta disponibilidad.
AvailabilityGroupName	varchar	50	Estado en el cual se encuentra el nodo.
Role	varchar	10	Tipo de sincronización.
AvailabilityMode	varchar	20	Tipo del Failover.

FailoverMode	varchar	10	Estado de la conexión del grupo de alta disponibilidad.
ConnectedState	varchar	15	Estado de la salud de la recuperación.
RecoveryHealth	varchar	10	Estado de salud de la sincronización.
SynchronizationHealth	varchar	15	Número de la última conexión de error.
LastConnectErrorNumber	int	4	Mensaje de la última conexión de error.
LastConnectErrorDescription	nvarchar	MAX	Fecha y hora de la falla de la conexión.
LastConnectErrorTimeStamp	datetime	16	Identificador único en donde se encuentra el grupo de alta disponibilidad.
InstanceId	int	4	Fecha en la que se generó el registro.
RecordCreatedDate	datetime	16	Fecha en la que se generó el registro.

2.3. Tabla staging.tBackupHistory

Nombre	tBackupHistory	**Esquema**	"staging"
Definición	Tabla que contiene de forma temporal los datos relacionados a el historial de respaldos de las bases de datos.		
Nombre Columna	**Tipo de Dato**	**Longitud**	**Descripción**

BackupHistoryId	int identity	4	Identificador único para el respaldo.
DatabaseName	varchar	255	Nombre del usuario que creó el respaldo.
LoginName	varchar	255	Tipo de respaldo.
BackupType	varchar	20	Modelo de recuperación.
RecoveryModel	varchar	30	Nivel de compatibilidad.
CompatibilityLevel	int	4	Tipo de colación utilizada.
Collation	varchar	50	Indica si se utilizó la opción de “CopyOnly”.
IsCopyOnly	varchar	5	Tipo de dispositivo en donde se almacena el respaldo.
DeviceType	varchar	25	Directorio en donde se almacenó el archivo de respaldo.
PhysicalDeviceName	varchar	MAX	Tamaño del archivo de respaldo.
BackupSizeMB	float	8	Duración de tiempo que tomó el proceso de creación del respaldo.
TimeTakenMin	int	4	Fecha de inicio de creación del respaldo.
BackupStartDate	datetime	16	Fecha final de la creación del respaldo.
BackupEndDate	datetime	16	Identificador único de las bases de datos que fue respaldada.
InstanceId	int	4	Fecha en la que se generó el registro.
RecordCreatedDate	datetime	16	Fecha en la que se generó el registro.

2.4. Tabla staging.tDatabase

Nombre	tDatabase	**Esquema**	“staging”

Definición	Tabla que contiene de forma temporal los datos relacionados con la información de cada base de datos.		
Nombre Columna	**Tipo de Dato**	**Longitud**	**Descripción**
DatabaseId	int identity	4	Identificador único para cada base de datos.
InstanceDatabaseId	int	4	Identificador único para cada base de datos.
DatabaseName	varchar	255	Nombre de la base de datos.
CompatibilityLevel	int	4	Nivel de compatibilidad.
Collation	varchar	50	Tipo de colación utilizada.
UserAccess	varchar	25	Nivel de acceso de usuarios.
RecoveryModel	varchar	25	Modelo de recuperación.
State	varchar	15	Estado de la base de datos.
LogReuseWait	varchar	25	Espera de reutilización de registros.
IsReadOnly	bit	1	Indica si la base de datos está en solo lectura.
IsAutoCloseOn	bit	1	Indica si la base de datos tiene habilitada la función “Auto Close”.
IsAutoShrink	bit	1	Indica si la base de datos tiene habilitada la función “Auto Shrink”.
IsStandBy	bit	1	Indica si la base de datos está en “Stand By”.
DatabaseCreatedDate	datetime	16	Fecha de creación de la base de datos
InstanceId	int	4	Identificador único de la instancia en donde se encuentra la base de datos.

RecordCreatedDate	datetime	16	Fecha en la que se generó el registro.

2.5. Tabla staging.tDatabaseFiles

Nombre	tDatabaseFiles	**Esquema**	“staging”
Definición	Tabla que contiene los datos relacionados con la información de los archivos pertenecientes a las bases de datos.		
Nombre Columna	**Tipo de Dato**	**Longitud**	**Descripción**
DatabaseFileId	int identity	4	Identificador único para cada uno de los archivos de base de datos.
InstanceDatabaseId	int	4	Tipo de archivo.
DatabaseName	varchar	255	Nombre lógico del archivo.
TypeDescription	varchar	10	Nombre físico del archivo.
LogicaFilelName	varchar	255	Directorio en el cual se encuentra los archivos de datos o “log”.
PhysicalFileName	nvarchar	MAX	Estado del archivo.
FileState	varchar	25	Tamaño del archivo.
Size	float	8	Identificador único de la base de datos a la que le pertenece el archivo.
InstanceId	int	4	Fecha en la que se generó el registro.
RecordCreatedDate	datetime	16	Identificador único para cada uno de los archivos de base de datos.

2.6. Tabla staging.tDriveSize

Nombre	tDriveSize	**Esquema**	“staging”

Definición	Tabla que contiene de forma temporal los datos relacionados con la información del espacio en disco.		
Nombre Columna	**Tipo de Dato**	**Longitud**	**Descripción**
DriveSizeId	int identity	4	Identificador único para cada disco.
DriveLetter	char	2	Letra que representa el disco.
DriveName	varchar	30	Nombre del disco.
FreeSpaceGB	float	8	Espacio libre en gigabytes.
TotalSizeGB	float	8	Porcentaje de espacio libre.
HostId	int	4	Espacio utilizado en gigabytes.
RecordCreatedDate	datetime	16	Fecha en la que se generó el registro.
RecordUpdatedDate	datetime	16	Fecha en la que se generó el registro.

2.7. Tabla staging.tHost

Nombre	tHost	Esquema	“staging
Definición	Tabla que contiene de forma temporal los datos relacionados con la información del host.		
Nombre Columna	**Tipo de Dato**	**Longitud**	**Descripción**
HostId	int	4	Identificador único del host.
HostName	varchar	100	Nombre del host.
Environment	varchar	15	Tipo de ambiente.
WindowsEdition	varchar	255	Edición de Windows
WindowsVersion	varchar	50	Versión de Windows.
PhysicalCores	varchar	255	Cantidad de núcleos en el procesador.
LogicalCores	varchar	255	Cantidad de núcleos lógicos en el procesador.

TotalMemoryGB	varchar	255	Cantidad de memoria total en el servidor.
RecordCreatedDate	datetime	16	Fecha en la que se generó el registro.

2.8. Tabla staging.tInstance

Nombre	tInstance	**Esquema**	"staging"
Definición	Tabla que contiene de forma temporal los datos relacionados		
Nombre Columna	**Tipo de Dato**	**Longitud**	**Descripción**
InstanceId	int	4	Identificador único para las instancias.
InstanceName	varchar	100	Nombre de instancia.
Collation	varchar	50	Tipo de colación.
TcpPort	char	4	Puerto de la instancia.
Version	varchar	100	Versión de SQL Server.
Edition	varchar	30	Edición de SQL Server.
Environment	nvarchar	30	Tipo de ambiente.
DefaultDataFilePath	nvarchar	MAX	Directorio en el cual se encuentran los archivos de datos (.mdf y/o .ndf).
DefaultLogPath	nvarchar	MAX	Directorio en el cual se encuentran los archivos de "log" (.log).
IsAlwaysOnAGEnable	varchar	25	Indica si la configuración de "Always On" está habilitada.
AuthenticationMode	varchar	50	Modo de autenticación.
HostId	int	4	Identificador único del host en donde se encuentran instaladas las instancias de SQL Server.

RecordCreatedDate	datetime	16	Fecha en la que se generó el registro.

2.9. Tabla staging.tInstanceConfiguration

Nombre	tInstanceConfiguration	**Esquema**	"staging"
Definición	Tabla que contiene de forma temporal los datos relacionados de las configuraciones avanzadas de las instancias.		
Nombre Columna	**Tipo de Dato**	**Longitud**	**Descripción**
InstanceConfigurationId	int identity	4	Identificador único para la configuración avanzada de las instancias.
ConfigurationId	int	4	Identificador de las configuraciones.
ConfigurationName	varchar	30	Nombre de la configuración.
ConfigurationDescription	varchar	48	Descripción de la configuración.
Value	int	4	Valor configurado.
InstanceId	int	4	Identificador único de la instancia en donde se encuentra la configuración.
RecordCreatedDate	datetime	16	Fecha en la que se generó el registro.

2.10. Tabla staging.tJobStatus

Nombre	tJobStatus	**Esquema**	staging
Definición	Tabla que contiene de forma temporal los datos relacionados con la información de los "jobs".		
Nombre Columna	**Tipo de Dato**	**Longitud**	**Descripción**

JobStatusId	int identity	4	Identificador único del “job”.
JobId	uniqueidentifier	16	Identificador único del “job” en la instancia.
JobName	varchar	255	Nombre del “job”.
JobOwner	varchar	100	Nombre del dueño del “job”.
IsEnabled	varchar	4	Indica si el “job” está habilitado.
StepName	varchar	100	Nombre del “step” del “job”.
Message	nvarchar	MAX	Último mensaje del “job.
RunStatus	varchar	20	Último estado del job”.
LastRunDuration	int	4	Última duración de tiempo de ejecución en segundos.
LastRunDate	int	4	Última duración de tiempo de ejecución.
LastRunTime	int	4	Última fecha de tiempo de ejecución.
JobDateCreated	datetime	16	Fecha de creación.
JobDateModified	datetime	16	Fecha de modificación.
InstanceId	int	4	Identificador único de la instancia en la cual se encuentran los “jobs”.
RecordCreatedDate	datetime	16	Fecha en la que se generó el registro.

2.11. Tabla staging.tMissingIndex

Nombre	tMissingIndex	**Esquema**	staging
Definición	Tabla que contiene de forma temporal los datos relacionados con la información de los posibles índices faltantes.		

Nombre Columna	Tipo de Dato	Longitud	Descripción
MissingIndexId	int identity	4	Identificador único para los índices faltantes.
DatabaseName	varchar	100	Nombre del objeto.
ObjectName	varchar	100	Nombre completo del objeto incluyendo el nombre de la base de datos, esquema y nombre de la tabla.
FullyQualifiedObjectName	varchar	MAX	Ventaja del índice.
IndexAdvantage	float	8	Lista de columnas que contribuyen a las consultas de igualdad.
EqualityColumns	varchar	MAX	Lista de columnas que contribuyen a las consultas de igualdad.
InEqualityColumns	varchar	MAX	Lista de columnas que contribuyen a las consultas de desigualdad.
IncludedColumns	varchar	MAX	Lista de columnas necesarias como columnas de cobertura para la consulta.
UniqueCompiles	int	4	Número de compilaciones y recopilaciones que se beneficiarían con el índice recomendando.
UserSeeks	int	4	Número de búsquedas provocadas por consultas para las que se podría haber utilizado el índice recomendado.

UserScans	int	4	Número de escaneos provocadas por consultas para las que se podría haber utilizado el índice.
LastUserSeekTime	datetime	16	Fecha y hora de la última búsqueda provocada por consultas de usuario para las que podría haberse utilizado el índice recomendado.
LastUserScanTime	datetime	16	Fecha y hora del último análisis provocado por consultas de usuarios para las que se podría haber utilizado el índice faltante.
AvgTotalUserCost	float	8	Costo promedio de las consultas que podría reducirse con el índice.
AvgUserImpact	float	8	Beneficio promedio que las consultas podrían experimentar si se implementara el índice faltante.
ProposedIndex	varchar	MAX	“Script” de creación del índice faltante.
InstanceId	int	4	Identificador único de la base de datos en donde puede ser implementado el índice.
RecordCreatedDate	datetime	16	Fecha en la que se generó el registro.

2.12. Tabla staging.tRestoreHistory

Nombre	tRestoreHistory	**Esquema**	“staging”
Definición	Tabla que contiene de forma temporal los datos relacionados al historial de restauraciones de las bases de datos.		
Nombre Columna	**Tipo de Dato**	**Longitud**	**Descripción**
RestoreHistoryId	int identity	4	Identificador único para cada restauración.
DatabaseName	varchar	255	Usuario que restauró la base de datos.
LoginName	varchar	255	Tipo de restauración.
RestoreType	varchar	20	Indica si una base de datos fue reemplazada.
Replace	varchar	4	Modelo de recuperación.
Recovery	varchar	4	Directorio en el cual se encuentra el archivo de respaldo utilizado.
BackupFileLocation	varchar	MAX	Directorio en donde se creó el archivo de datos restaurado.
DestinationDataFile	varchar	MAX	Directorio en donde se creó el archivo de “log” restaurado.
DestinationLogFile	varchar	MAX	Fecha de la restauración.
RestoreDate	datetime	16	Identificador único de las bases de datos que fue respaldada.
InstanceId	int	4	Identificador único en el cual se encuentra el histórico de restauraciones.
RecordCreatedDate	datetime	16	Fecha en la que se generó el registro.

2.13. Tabla staging.tSQLServiceStatus

Nombre	tSQLServiceStatus	**Esquema**	staging

Definición	Tabla que contiene de forma temporal los datos relacionados con la información de los servicios de SQL Server.		
Nombre Columna	**Tipo de Dato**	**Longitud**	**Descripción**
SQLServiceId	int identity	4	Identificador único para el servicio de SQL Server.
ServiceName	varchar	50	Nombre del servicio.
StartMode	varchar	50	Modo de inicio del servicio.
Status	varchar	15	Estado del servicio
HostId	int	4	Identificador único del host en donde se encuentra instalado los servicios de SQL Server
RecordCreatedDate	datetime	16	Fecha en la que se generó el registro.
RecordUpdatedDate	datetime	16	Fecha en la que se actualizó el registro.