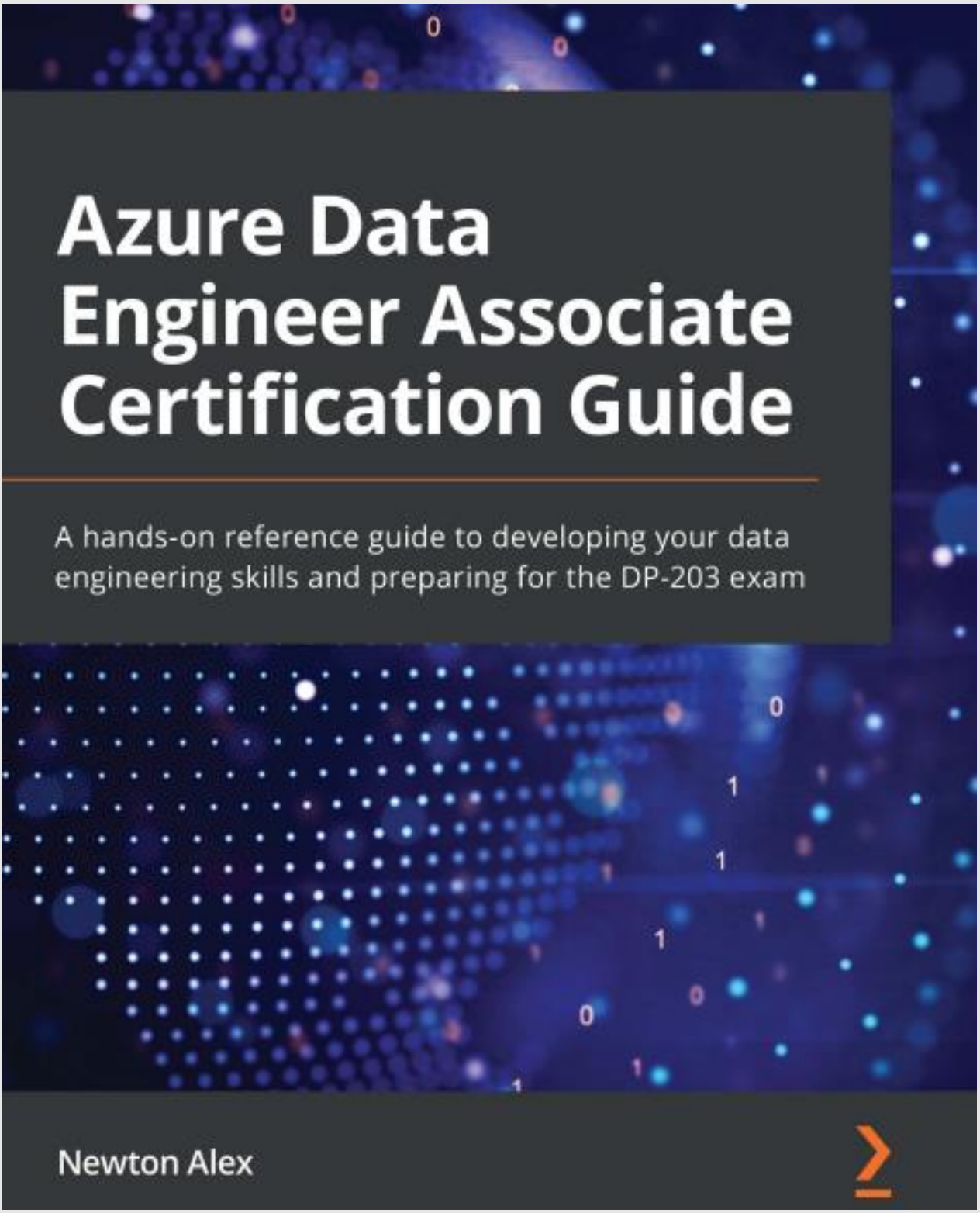
Azure Data Engineer Associate Certification Guide
A hands-on reference guide to developing your data engineering skills and preparing for the DP-203 exam
Newton Alex

Parte 2: Data Storage

Esta parte se sumerge en los detalles de los diferentes tipos de almacenamiento, las estrategias de partición de datos, los esquemas, los tipos de archivos, la alta disponibilidad, la redundancia, etc.

Esta sección comprende los siguientes capítulos:

- Capítulo 2, Diseño de una estructura de almacenamiento de datos
- Capítulo 3, Diseño de una estrategia de partición
- Capítulo 4, Diseño de la capa de servicio
- Capítulo 5, Implementación de estructuras físicas de almacenamiento de datos
- Capítulo 6, Implementación de estructuras lógicas de datos
- Capítulo 7, Implementación de la capa de servicio

Capítulo 3: Diseñar una estrategia de partición

La partición de datos se refiere al proceso de dividir los datos y almacenarlos en lugares físicamente diferentes. La partición de los datos se realiza principalmente por razones de rendimiento, escalabilidad, capacidad de gestión y seguridad. El particionamiento en sí es un término genérico, pero los métodos y las técnicas de partición varían de un servicio a otro; por ejemplo, las técnicas de partición utilizadas para el almacenamiento de Azure Blob pueden no ser las mismas que las aplicadas para los servicios de bases de datos como Azure SQL o Azure Synapse Dedicated SQL pool. Del mismo modo, las bases de datos de documentos como Cosmos DB tienen técnicas de partición diferentes a las de Azure Queues o Azure Tables. En este capítulo, exploraremos algunas de las técnicas de particionamiento importantes y cuándo utilizarlas.

Al igual que en el capítulo anterior, volveremos a centrarnos más en los aspectos de diseño, según el programa de estudios. Los detalles de implementación serán cubiertos en el Capítulo 5, Implementación de Estructuras de Almacenamiento de Datos Físicos.

Este capítulo cubrirá los siguientes temas:

- Entender los fundamentos del particionamiento
- Diseño de una estrategia de partición para archivos
- Diseño de una estrategia de partición para cargas de trabajo analíticas
- Diseño de una estrategia de partición para la eficiencia/rendimiento
- Diseño de una estrategia de partición para Azure Synapse Analytics
- Identificación de cuándo se necesita la partición en Azure Data Lake Storage Gen2 (ADLS Gen2)

Casi todos los principales servicios de Azure Storage y Azure Analytics soportan el particionamiento de una manera u otra, pero en el interés de la certificación, vamos a cubrir sólo los servicios que son relevantes para la certificación. Si estás interesado en aprender sobre el particionamiento en los otros servicios de Azure, puedes consultar el siguiente enlace: https://docs.microsoft.com/en-us/azure/architecture/best-practices/data-partitioning-strategies.

¡Vamos a sumergirnos en el mundo del particionamiento!

3.1. Entendiendo los fundamentos del particionamiento

En el capítulo anterior, introdujimos brevemente el concepto de particionamiento como parte de la sección Diseño del almacenamiento para una consulta eficiente. Exploramos los conceptos de particionamiento del lado del almacenamiento, como la replicación de datos, la reducción de las operaciones entre particiones, como los joins, y la consistencia final para mejorar el rendimiento de las consultas. En este capítulo, profundizaremos de forma más sistemática en las técnicas de particionamiento tanto de almacenamiento como de análisis. Comencemos con los beneficios del particionamiento.

3.1.1. Beneficios del particionamiento

El particionamiento tiene varios beneficios además del rendimiento de las consultas. Veamos algunos de los más importantes.

Mejora del rendimiento

Como hemos comentado en el capítulo anterior, **el particionado ayuda a mejorar la paralelización de las consultas** al dividir los datos monolíticos masivos en trozos más pequeños y fácilmente consumibles.

Además de la paralelización, **el particionamiento también mejora el rendimiento a través de la poda de datos**, otro concepto que ya discutimos en el capítulo anterior. **Utilizando la poda de datos, las consultas pueden ignorar las particiones no relevantes, reduciendo así la entrada/salida (I/O) requerida para las consultas**.

Las particiones también ayudan a archivar o eliminar datos antiguos. Por ejemplo, supongamos que necesitamos eliminar todos los datos de más de 12 meses. Si particionamos los datos en unidades de datos mensuales, podemos borrar los datos de un mes completo con un solo comando DELETE, en lugar de borrar todos los archivos uno por uno o todas las entradas de una tabla fila por fila.

Veamos a continuación cómo la partición ayuda a la escalabilidad.

Mejora de la escalabilidad

En el mundo del procesamiento de big data, hay dos tipos de escalado: **vertical** y **horizontal**.

El **escalado vertical** se refiere a la técnica de aumentar la capacidad de las máquinas individuales añadiendo más memoria, CPU, almacenamiento o red para mejorar el rendimiento. Esto suele ayudar a corto plazo, pero al final se llega a un límite más allá del cual no se puede escalar.

El segundo tipo de escalado se denomina **escalado horizontal**. Se refiere a la técnica de aumentar la capacidad de procesamiento y almacenamiento añadiendo más y más máquinas a un clúster, con especificaciones de hardware regulares que están fácilmente disponibles en el mercado (hardware básico). A medida que los datos crecen, basta con añadir más máquinas y redirigir los nuevos datos a las nuevas máquinas. En teoría, este método no tiene límites máximos y puede crecer eternamente. Los data lakes se basan en el concepto de escalado horizontal.

La partición de los datos ayuda de forma natural al escalado horizontal. Por ejemplo, supongamos que almacenamos los datos en un intervalo por día en particiones, por lo que tendremos unas 30 particiones por mes. Ahora, si necesitamos generar un informe mensual, podemos configurar el clúster para que tenga 30 nodos, de modo que cada nodo pueda procesar los datos de un día. Si la necesidad aumenta para procesar informes trimestrales (es decir, informes

cada 3 meses), podemos añadir más nodos, por ejemplo, 60 nodos más a nuestro tamaño original de clúster de 30 para procesar 90 días de datos en paralelo. Por lo tanto, si podemos diseñar nuestra estrategia de partición de datos de tal manera que podamos dividir los datos fácilmente entre nuevas máquinas, esto nos ayudará a escalar más rápido.

Veamos a continuación cómo la partición ayuda a la gestión de los datos.

Mejorar la capacidad de gestión

En muchos sistemas analíticos, tendremos que tratar con datos procedentes de una gran variedad de fuentes, y cada una de ellas puede tener asignadas diferentes políticas de gobierno de datos. Por ejemplo, algunos datos pueden ser confidenciales, por lo que debemos restringir su acceso; otros pueden ser datos transitorios que pueden regenerarse a voluntad; otros pueden ser datos de registro que pueden eliminarse después de unos meses; otros pueden ser datos de transacciones que deben archivarse durante años; etc. Del mismo modo, puede haber datos que necesiten un acceso más rápido, por lo que podríamos optar por conservarlos en almacenes de unidades de estado sólido (SSD) de primera calidad y otros datos en una unidad de disco duro (HDD) para ahorrar costes.

Si almacenamos estos diferentes conjuntos de datos en sus propias particiones de almacenamiento, entonces resulta fácil aplicar reglas separadas -como restricciones de acceso, o configurar diferentes actividades de gestión del ciclo de vida de los datos, como la eliminación o el archivo de datos, etc.- para las particiones individuales. Por lo tanto, la partición reduce la sobrecarga de gestión, especialmente cuando se trata de varios tipos de datos diferentes, como en un data lake.

Veamos a continuación cómo la partición de datos ayuda a la seguridad.

Mejora de la seguridad

Como hemos visto en la sección anterior sobre la mejora de la capacidad de gestión, los conjuntos de datos confidenciales pueden tener diferentes niveles de acceso y privacidad. Los datos de los clientes suelen tener los niveles más altos de seguridad y privacidad. Por otro lado, los catálogos de productos pueden no necesitar niveles muy altos de seguridad y privacidad.

Por lo tanto, **al dividir los datos en función de los requisitos de seguridad, podemos aislar los datos seguros y aplicar reglas de control de acceso y auditoría independientes a esas particiones**, permitiendo así que sólo los usuarios con privilegios accedan a esos datos.

Veamos ahora cómo podemos mejorar la disponibilidad de los datos mediante la partición.

Mejorar la disponibilidad

Si nuestros datos se dividen en múltiples particiones que se almacenan en diferentes máquinas, las aplicaciones pueden seguir sirviendo al menos parcialmente los datos aunque algunas particiones estén caídas. Sólo un subconjunto de clientes cuyas particiones se han caído puede verse afectado, mientras que el resto de los clientes no verá ningún impacto. Esto es mejor que la caída de toda la aplicación. Por lo tanto, la partición física de los datos ayuda a mejorar la disponibilidad de los servicios.

En general, si planificamos correctamente nuestra estrategia de partición, los beneficios pueden ser significativos. Espero que ahora hayas entendido los beneficios de la partición de datos. Veamos ahora algunas estrategias de partición desde la perspectiva del almacenamiento/archivos.

3.2. Diseñando una estrategia de partición para archivos

En esta sección, veremos las técnicas de partición disponibles para Azure Storage, que también deberían cubrir los archivos. Los servicios de Azure Storage son genéricos y muy flexibles a la hora de particionar. Podemos implementar cualquier lógica de partición que queramos, utilizando las mismas interfaces de programación de aplicaciones (API) de Crear, Leer, Actualizar y Eliminar (CRUD) que están disponibles públicamente. No hay APIs o características especiales disponibles para el particionamiento. Con estos antecedentes, exploremos ahora las opciones de partición disponibles en Azure Blob storage y ADLS Gen2.

3.2.1. Azure Blob storage

En el Azure Blob storage, primero creamos cuentas Azure; luego, dentro de las cuentas, creamos contenedores; y dentro de los contenedores, creamos blobs de almacenamiento reales. Estos contenedores son entidades lógicas, por lo que incluso cuando creamos blobs de datos dentro de los contenedores, no hay garantía de que los datos aterricen dentro de la misma partición. Pero hay un truco para mejorar nuestras posibilidades de almacenar los blobs en la misma partición.

Azure utiliza algo llamado partición de rango para almacenar blobs. En una partición de rango, los archivos que están en una secuencia léxica terminan juntos en una partición.

Por ejemplo, si tenemos nombres de archivo como Cab1-20211220, Cab1-20211221, Cab1-20211222 y Cab1-20211223, estos archivos terminarán en su mayoría en la misma partición.

Del mismo modo, los nombres de archivos como IAC-Cabs, IAC-Routes, IAC-Customers y IAC-Drivers terminarán en su mayoría en la misma partición.

Azure Storage utiliza <nombre de la cuenta + nombre del contenedor + nombre del blob> como clave de la partición. Continúa almacenando blobs en la misma partición hasta que alcanza el límite interno de la partición. En ese momento, Azure Storage reparticiona y reequilibra los datos y los distribuye uniformemente entre las particiones. Todo este proceso de repartición y reequilibrio será realizado automáticamente por Azure Storage, sin intervención del usuario.

Si la cantidad de datos almacenados es grande y Azure Storage comienza a reparticionar y reequilibrar los datos, la latencia de las APIs CRUD se verá afectada. Puede evitar o retrasar estas reparticiones añadiendo un valor hash de tres dígitos (valor aleatorio) a sus nombres de archivo. Esto hará que los datos se distribuyan en varias particiones.

Por ejemplo, sus nombres de archivo podrían utilizar el siguiente formato

```
New York/cabs/cab1234/customers/{XYZYYYYMMDD}
```

Aquí, XYZ podría ser el valor hash aleatorio.

3.2.2. ADLS Gen2

La solución para la partición de archivos en un data lake es la misma que para el diseño de la estructura de carpetas, que ya comentamos en el capítulo anterior. Podemos aplicar diferentes políticas de seguridad o diferentes configuraciones de gestión del ciclo de vida de los datos a las carpetas individuales en ADLS Gen2. Dado que ya hemos explorado el tema de la estructura de carpetas en detalle, no lo volveremos a tratar aquí.

Aparte de la estructura de carpetas, también podemos particionar los datos para obtener los beneficios que hemos discutido anteriormente en este capítulo, como el rendimiento, la escalabilidad, la capacidad de gestión, la seguridad, la disponibilidad, etc. El proceso de segregación de los datos en particiones puede hacerse manualmente o puede automatizarse utilizando Azure Data Factory (ADF). Profundizaremos en la automatización mediante ADF en los capítulos posteriores orientados a la implementación.

Ahora que tenemos una idea bastante buena sobre las optimizaciones de particionamiento basadas en el almacenamiento y en los archivos, vamos a explorar el particionamiento desde el punto de vista de las cargas de trabajo analíticas.

3.3. Diseñando una estrategia de partición para cargas de trabajo analíticas

Hay tres tipos principales de estrategias de partición para cargas de trabajo analíticas. Éstas se enumeran a continuación:

- **Partición horizontal, que también se conoce como sharding**
- **Partición vertical**
- **Partición funcional**

Exploremos cada una de ellas en detalle.

3.3.1. Partición horizontal

En una **partición horizontal**, dividimos los datos de la tabla horizontalmente, y los subconjuntos de filas se almacenan en diferentes data stores. Cada uno de estos subconjuntos de filas (con el mismo esquema que la tabla principal) se denominan **fragmentos** (**shards**). Esencialmente, **cada uno de estos fragmentos se almacena en diferentes instancias de la base de datos**.

Aquí puede ver un ejemplo de partición horizontal:

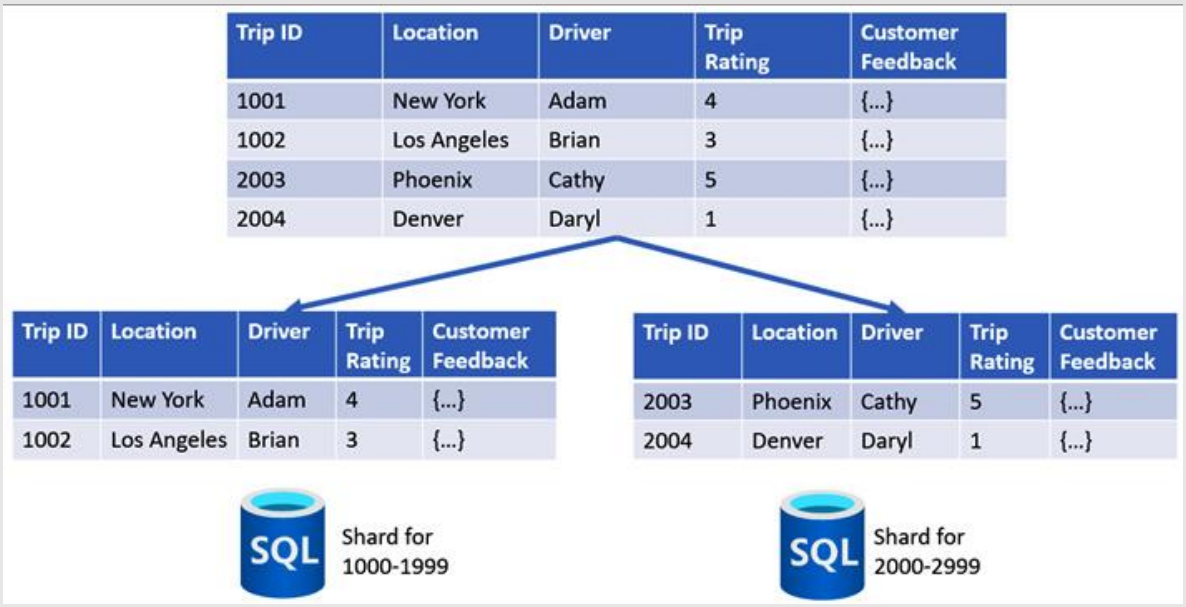

Figura 3.1 - Ejemplo de partición horizontal

En el ejemplo anterior, puede ver que los datos de la tabla superior se distribuyen horizontalmente en función del rango de Trip ID.

Selección de la clave de partición correcta

Es muy importante que seleccionemos la clave de fragmento correcta (también llamada clave de partición) para la partición de los datos, ya que cambiarla posteriormente será una operación muy costosa. Las siguientes pautas nos ayudarán a ello:

- Seleccionar una clave que reparta los datos de tal manera que el tráfico de la aplicación a las particiones de datos se distribuya uniformemente.

- Seleccionar una clave que no cambie demasiado a menudo. Las buenas claves son estáticas y están muy extendidas, es decir, el rango de esa clave no debe ser ni demasiado pequeño ni demasiado grande. Como regla general, una clave que pueda generar cientos de particiones es buena. Evite las claves que generan muy pocas particiones (decenas de particiones) o demasiadas particiones (miles de particiones).

NOTA

No intente equilibrar los datos para que se distribuyan uniformemente entre las particiones a menos que lo requiera específicamente su caso de uso, porque normalmente se accederá más a los datos más recientes que a los más antiguos. Por lo tanto, las particiones con datos recientes terminarán convirtiéndose en cuellos de botella debido al alto acceso a los datos.

Cubriremos el patrón de fragmentación en profundidad en la siguiente sección cuando hablemos de la partición para la eficiencia y el rendimiento.

3.3.2. Partición vertical

En una **partición vertical**, dividimos los datos verticalmente, y cada subconjunto de las columnas se almacena por separado en un data store diferente. En el caso del particionamiento vertical, normalizamos parcialmente la tabla para dividirla en tablas más pequeñas (menos columnas). Este tipo de partición es ideal en los casos en que una tabla puede tener un subconjunto de datos al que se accede con más frecuencia que al resto. El particionamiento vertical puede ayudar a acelerar las consultas, ya que sólo se puede recuperar selectivamente el subconjunto de datos necesario, en lugar de leer filas enteras. Esto es ideal para los data stores orientados a columnas, como HBase, Cosmos DB, etc.

Puede ver un ejemplo de partición vertical aquí:

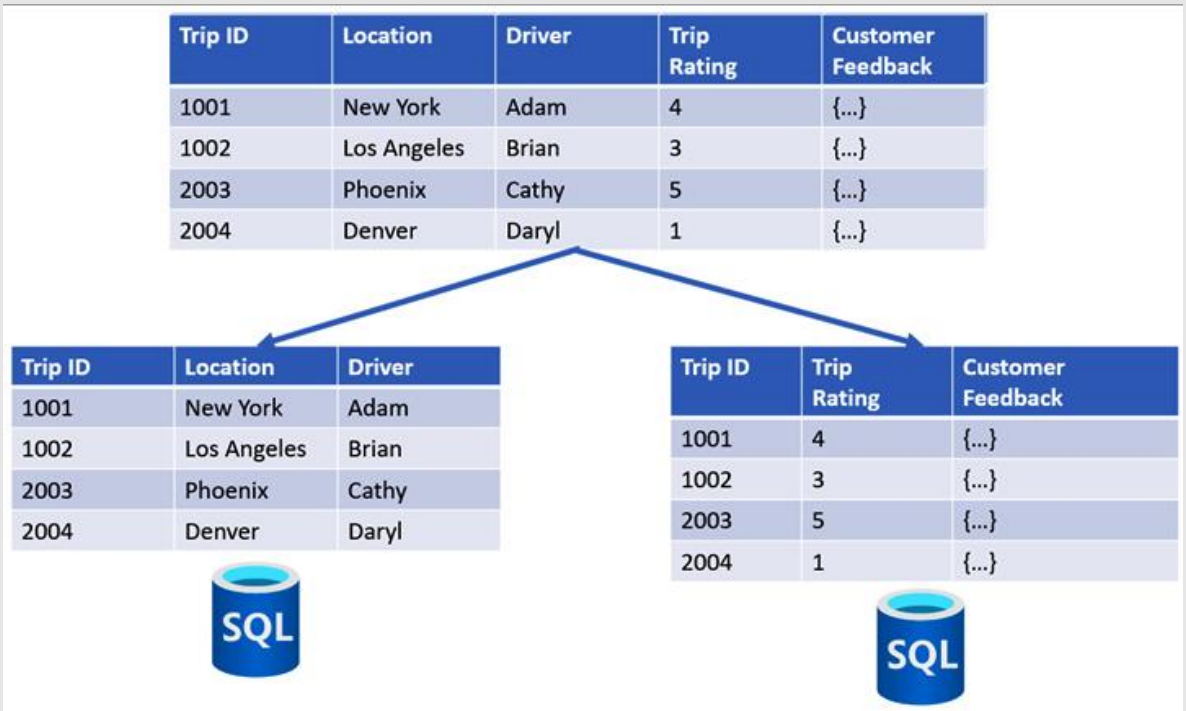

Figura 3.2 - Ejemplo de partición vertical

En el ejemplo anterior, puede ver que las columnas como Trip Rating y Customer Feedback que podrían no ser utilizadas con frecuencia se dividen en particiones separadas. Esto ayudará a reducir la cantidad de datos que se leen para las consultas.

3.3.3. Particiones funcionales

Las particiones funcionales son similares a las particiones verticales, salvo que aquí almacenamos tablas o entidades enteras en diferentes data stores. Pueden utilizarse para separar los datos que pertenecen a diferentes organizaciones, las tablas de uso frecuente de las de uso infrecuente, las tablas de lectura y escritura de las de sólo lectura, los datos sensibles de los datos generales, etc.

Aquí puede ver un ejemplo de partición funcional:

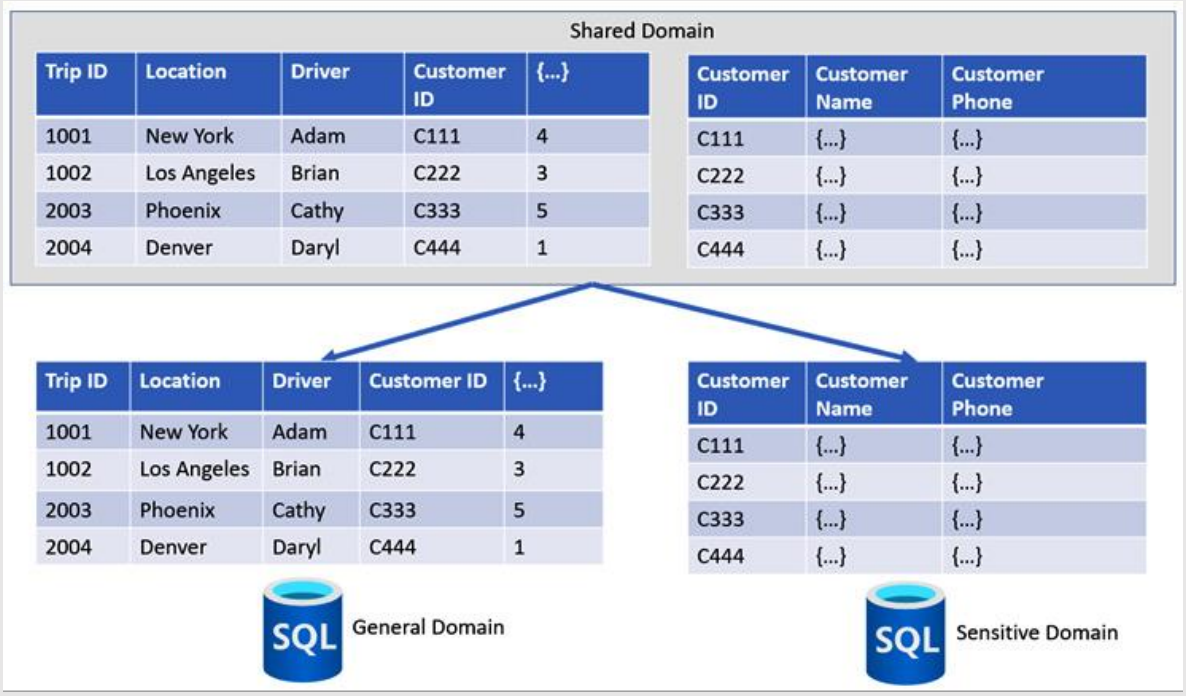

Figura 3.3 - Ejemplo de partición funcional

En este ejemplo, puede ver cómo los datos de los clientes se trasladan a su propia partición. Esta segregación nos ayudará a aplicar diferentes reglas de privacidad y seguridad para las diferentes particiones.

Los servicios de Azure como Azure SQL y Azure Synapse Dedicated pool soportan todos los formatos de partición comentados en esta sección.

3.4. Diseñando una estrategia de partición para la eficiencia/rendimiento

En las últimas secciones, hemos explorado las diversas opciones de almacenamiento y partición analítica y hemos aprendido cómo la partición ayuda al rendimiento, la escala, la seguridad, la disponibilidad, etc. En esta sección, recapitularemos los puntos que aprendimos sobre el rendimiento y la eficiencia y conoceremos algunos patrones de rendimiento adicionales.

Estas son algunas estrategias que hay que tener en cuenta a la hora de diseñar la eficiencia y el rendimiento:

- Particionar los conjuntos de datos en trozos más pequeños que puedan ejecutarse con un paralelismo óptimo para múltiples consultas.
- Particionar los datos de forma que las consultas no acaben requiriendo demasiados datos de otras particiones. Minimizar las transferencias de datos entre particiones.
- Diseñar estructuras de carpetas eficaces para mejorar la eficiencia de las lecturas y escrituras de datos.
- Particionar los datos de forma que se pueda podar una cantidad significativa de datos mientras se ejecutan las consultas.
- Particionar en unidades de datos que puedan añadirse, eliminarse, intercambiarse o archivarse fácilmente. Esto ayuda a mejorar la eficiencia de la gestión del ciclo de vida de los datos.
- Los tamaños de archivo en el rango de 256 **megabytes** (MB) a 100 **gigabytes** (GB) funcionan realmente bien con motores analíticos como HDInsight y Azure Synapse. Por lo tanto, agregue los archivos a estos rangos antes de ejecutar los motores analíticos en ellos.
- Para los trabajos de I/O intensivos, intente mantener los tamaños óptimos de los búferes de E/S en el rango de 4 a 16 MB; cualquier cosa demasiado grande o demasiado pequeña se volverá ineficiente.
- Ejecute más contenedores o ejecutores por **máquina virtual** (**VM**) (como los ejecutores Apache Spark o los contenedores Apache **Yet Another Resource Negotiator** (**YARN**)).

Intente recordar las buenas prácticas anteriores cuando diseñe su próxima estrategia de partición. A continuación, tratemos de entender cómo encontrar realmente los datos correctos para particionar y cuántos datos particionar.

3.4.1. Proceso iterativo de mejora del rendimiento de las consultas

He aquí un proceso iterativo de alto nivel para mejorar el rendimiento de las consultas:

1. Haga una lista de las consultas críticas para el negocio, las consultas que se ejecutan con más frecuencia y las consultas más lentas.

2. Compruebe los planes de consulta para cada una de estas consultas utilizando la palabra clave EXPLAIN y vea la cantidad de datos que se utilizan en cada etapa (aprenderemos a ver los planes de consulta en los capítulos posteriores).

3. Identifique los joins o filtros que más tiempo le están llevando. Identifique las particiones de datos correspondientes.

4. Intente dividir las correspondientes particiones de datos de entrada en particiones más pequeñas, o cambie la lógica de la aplicación para realizar un procesamiento aislado sobre cada partición y posteriormente fusionar sólo los datos filtrados.

5. También puede probar a ver si otras claves de particionamiento funcionan mejor y si es necesario repartir los datos para obtener un mejor rendimiento del trabajo en cada partición.

6. Si alguna tecnología de partición en particular no funciona, puede explorar tener más de una pieza de lógica de partición-por ejemplo, podría aplicar la partición horizontal dentro de la partición funcional, y así sucesivamente.

7. Supervise la partición regularmente para comprobar si los patrones de acceso a la aplicación están equilibrados y bien distribuidos. Intente identificar los puntos calientes desde el principio.

8. Repita este proceso hasta que consiga el tiempo de ejecución de consulta preferido.

También utilizaremos estas directrices en nuestros ejemplos de futuros capítulos. A continuación, exploremos los detalles del particionamiento en Azure Synapse Analytics.

3.5. Diseño de una estrategia de partición para Azure Synapse Analytics

Aprendimos sobre Azure Synapse Analytics en el Capítulo 2, Diseño de una estructura de almacenamiento de datos. Synapse Analytics contiene dos motores de cómputo, que se describen aquí:

- Un pool de **Structured Query Language (SQL)** que consiste en pools de SQL serverless and dedicated (anteriormente conocido como SQL Data Warehouse).

- Un pool de Spark que consiste en pools de Synapse Spark

Pero cuando la gente se refiere a Azure Synapse Analytics, suele referirse a la opción de Dedicated SQL pool. En esta sección, veremos la estrategia de partición disponible para Synapse Dedicated SQL pool.

NOTA

Ya hemos cubierto brevemente la partición en Spark como parte de la sección de poda de datos en el capítulo anterior. Los mismos conceptos se aplican también a Synapse Spark.

Antes de explorar las opciones de partición, vamos a recapitular las técnicas de distribución de datos de un Synapse dedicated pool del capítulo anterior, ya que esto jugará un papel importante en nuestra estrategia de partición, como sigue:

Un dedicated SQL pool es un sistema de procesamiento paralelo masivo (MPP) que divide las consultas en 60 consultas paralelas y las ejecuta en paralelo. Cada una de estas consultas más pequeñas se ejecuta en algo llamado distribución. ***Una distribución es una unidad básica de procesamiento y almacenamiento para un dedicated SQL pool****. Hay tres formas diferentes de distribuir (fragmentar) los datos entre las distribuciones, como se indica a continuación:*

- *Tablas Round-robin*
- *Tablas Hash*
- *Tablas replicadas*

El particionamiento es compatible con todos los tipos de distribución de la lista anterior. Aparte de los tipos de distribución, Dedicated SQL pool también soporta tres tipos de tablas: **clustered columnstore**, **clustered index** y **heap** tables. Exploraremos estas opciones de tablas en detalle en los próximos capítulos. **El particionamiento también es compatible con todos estos tipos de tablas**.

En un dedicated SQL pool, los datos ya están distribuidos en sus 60 distribuciones, por lo que debemos ser cuidadosos a la hora de decidir si necesitamos particionar más los datos. **Las tablas clustered columnstore funcionan de forma óptima cuando el número de filas por tabla en una distribución es de alrededor de 1 millón**.

Por ejemplo, si pensamos dividir más los datos por los meses de un año, estamos hablando de 12 particiones x 60 distribuciones = 720 subdivisiones. Cada una de estas divisiones debe tener al menos 1 millón de filas; en otras palabras, la tabla (normalmente una tabla de hechos) deberá tener más de 720 millones de filas. Por lo tanto, **tendremos que tener cuidado de no sobreparticionar los datos cuando se trate de dedicated SQL pool**.

Dicho esto, el particionamiento en los pools dedicados de Synapse tiene dos ventajas distintas, como veremos ahora.

3.5.1. Mejora del rendimiento durante la carga de datos

El particionamiento ayuda a cargar los datos para las consultas en los dedicated SQL pools. Esta es una técnica que ya discutimos en la sección de Beneficios del particionamiento de este capítulo. Si podemos agrupar los datos que pertenecen a un periodo de tiempo concreto en una partición, entonces añadir o eliminar esos datos será tan sencillo como ejecutar un simple comando ADD o DELETE. Por ejemplo, supongamos que necesitamos generar un informe de 12 meses. Al final de cada mes, eliminamos el mes más antiguo y añadimos un nuevo mes al informe. Si hemos particionado los datos con la granularidad de los meses, podemos eliminar fácilmente los datos antiguos y añadir los nuevos utilizando una técnica de cambio de partición en un dedicated SQL pools.

3.5.2. Mejora del rendimiento de las consultas de filtrado

El particionamiento también puede ayudar a mejorar el rendimiento de las consultas al poder filtrar los datos basándose en las particiones. En particular, las particiones ayudan con la cláusula WHERE en las consultas. Por ejemplo, si hemos particionado los datos en base a los meses de un año, podemos especificar qué mes exacto buscar en nuestras consultas, saltándonos así el resto de meses.

Espero que hayáis entendido las características y restricciones del particionamiento en los Azure Synapse dedicated pools. Pasemos a nuestra siguiente sección, que habla de cuándo empezar a particionar los datos en ADLS Gen2.

3.6. Identificar cuándo es necesario el particionamiento en ADLS Gen2

Como hemos aprendido en el capítulo anterior, podemos particionar los datos en función de nuestros requisitos -como el rendimiento, la escalabilidad, la seguridad, la sobrecarga operativa, etc.- pero hay **otra razón por la que podríamos acabar particionando nuestros datos, y son los distintos límites de ancho de banda de I/O que impone Azure en los niveles de suscripción. Estos límites se aplican tanto al almacenamiento Blob como al ADLS Gen2**.

La tasa a la que ingerimos datos en un sistema de Azure Storage se llama **tasa de entrada (ingress rate)**, y la tasa a la que movemos los datos fuera del sistema de Azure Storage se llama **tasa de salida (egress rate)**.

La siguiente tabla muestra una instantánea de algunos de los límites impuestos por Azure Blob storage. Esta tabla es sólo para dar una idea de los límites que impone Azure Storage. Cuando diseñamos nuestras aplicaciones de data lake, debemos tener en cuenta estas restricciones como parte de nuestro propio diseño:

Resource	Limit
Number of storage accounts per region per subscription, including standard, and premium storage accounts.	250
Maximum storage account capacity	5 PiB (can be increased by calling Azure Support)
Maximum request rate[1] per storage account	20,000 requests per second
Maximum ingress[1] per storage account (US, Europe regions)	10 Gbps
Maximum ingress[1] per storage account (regions other than US and Europe)	5 Gbps if RA-GRS/GRS is enabled, 10 Gbps for LRS/ZRS
Maximum egress for general-purpose v2 and Blob storage accounts (all regions)	50 Gbps
Maximum number of IP address rules per storage account	200
Maximum number of virtual network rules per storage account	200

Figura 3.4 - Algunos de los límites de Azure Blob storage en el momento de la publicación de este libro

Así, por ejemplo, si tu tasa de salida es superior a 50 gigabits por segundo (Gbps) para tu data lake, tendrás que crear varias cuentas y dividir tus datos entre esas cuentas.

CONSEJO

Si utiliza una configuración híbrida de sistemas locales y en la nube, y si transfiere datos entre dichos sistemas con frecuencia, asegúrese de que sus máquinas de origen/destino y la red pública real puedan soportar el nivel de tasas de transferencia de datos de entrada y salida que ofrece Azure Storage. Si está moviendo datos desde fuentes on-prem, considere el uso de Azure ExpressRoute.

Para conocer los límites completos y actualizados de Azure Storage, consulte la siguiente documentación: https://docs.microsoft.com/en-us/azure/storage/common/scalability-targets-standard-account.

Tenga en cuenta que algunos de los límites de la tabla (como las tasas de entrada y los límites de capacidad de almacenamiento) son límites blandos, lo que significa que puede ponerse en contacto con el servicio de asistencia de Azure para aumentar esos límites hasta cierto punto; sin embargo, al final llegará a los límites duros de cada opción. Otros recursos como el Protocolo de Internet (IP), las direcciones, las redes virtuales (VNets), etc. son límites duros, por lo que debe planificar las particiones teniendo en cuenta estos números.

NOTA

Los mayores requisitos de entrada/salida también podrían provenir de las aplicaciones que se ejecutan sobre Azure Storage y no sólo a través de las cargas y descargas directas de datos en el almacén de Azure. Por ejemplo, si tenemos un Azure SQL o Azure Synapse dedicated pool (data warehouse) que tiene un shard muy ocupado, podría superar los límites de lectura (egress) de esa cuenta de almacenamiento. En estos casos, tendremos que repartir ese shard para dividir los datos entre varias cuentas.

En general, para cualquier servicio de Azure, mantén un ojo en los límites de recursos para que no sea una sorpresa cuando tu producto se despliegue en producción.

Resumen

Con esto, hemos llegado al final de nuestro tercer capítulo. Espero que hayas disfrutado aprendiendo sobre las diferentes técnicas de partición disponibles en Azure. Empezamos con los fundamentos de la partición, donde aprendiste sobre los beneficios de la partición; luego pasamos a las técnicas de partición para el almacenamiento y las cargas de trabajo analíticas. Exploramos las mejores prácticas para mejorar la eficiencia y el rendimiento del particionamiento. Comprendimos el concepto de tablas de distribución y cómo afectan al particionamiento de Azure Synapse Analytics y, por último, aprendimos sobre las limitaciones de almacenamiento, que desempeñan un papel importante a la hora de decidir cuándo particionar para ADLS Gen2. Esto cubre el temario del examen DP-203, Diseño de una estrategia de partición. Reforzaremos lo aprendido en este capítulo mediante detalles de implementación y consejos en los siguientes capítulos.

Vamos a explorar la capa de servicio en el próximo capítulo.