Trabajo Práctico de Investigación NoSQL

Dada la experiencia laboral y/o académica adquirida en su entorno de trabajo o bien a lo largo de la cursada, se solicita desarrollar un Trabajo Práctico de Investigación orientado a un motor o al concepto de NoSQL de su elección.

El mismo deberá contar con al menos 6 aspectos de la materia como alcance del trabajo (Introducción, trabajos relacionados, ventajas, desventajas, etc.).

Para ello se le solicita un archivo en formato PDF con los siguientes contenidos:

* Portada (con datos principales del alumno/a);
* Índice de temas propuestos;
* Contenidos del tema tratado;
* Conclusión/reflexión del estudiante y cómo se integran los contenidos de la materia;
* Referencias bibliográficas utilizadas.

Para este trabajo, no deben superar las 45 hojas.

Como proceso final, se deberán grabar y subir la presentación a un enlace privado en YouTube (su URL deberá figurar en el archivo de la carpeta entregada).

Tabla de contenido

[Introducción 3](#_Toc107958968)

[Conceptos básicos de DocumentDB 5](#_Toc107958969)

[Propiedades anidadas maestro-detalle 9](#_Toc107958970)

[Cuando usar una base de datos de documentos 11](#_Toc107958971)

[¿Por qué DocumentDB? 12](#_Toc107958972)

[Consultas enriquecidas en SQL 14](#_Toc107958973)

[Desarrollo del lado del cliente 15](#_Toc107958974)

[Desarrollo del lado del servidor 16](#_Toc107958975)

[Escalabilidad 17](#_Toc107958976)

[Consistencia 18](#_Toc107958977)

[Costos 19](#_Toc107958978)

[Resumen 20](#_Toc107958979)

# Introducción

Azure Cosmos DB de Microsoft es un servicio de datos multimodal distribuido globalmente que le permite escalar de forma elástica el rendimiento y el almacenamiento en cualquier número de regiones geográficas con baja latencia, alta disponibilidad y coherencia. El motor de base de datos de Azure Cosmos DB admite de forma nativa el dialecto SQL de DocumentDB, la API de MongoDB, la API de Gremlin (gráfico) y las API de almacenamiento de tablas de Azure. En este libro, nos centraremos en la parte DocumentDB de Cosmos DB, que es la plataforma de base de datos de documentos NoSQL de Microsoft que se ejecuta en Azure. SQL son las siglas de Structured Query Language y es un acrónimo asociado a las bases de datos relacionales, ya que es el lenguaje estándar y más utilizado para trabajar con bases de datos relacionales. Al usar el término NoSQL, lo que realmente queremos expresar es que nos estamos refiriendo a bases de datos no relacionales. Las bases de datos no relacionales abandonan muchos de los conceptos que utilizan las bases de datos relacionales tradicionales, en particular, cómo se estructuran y organizan los datos en un formato tabular donde cada columna tiene un tipo de datos específico. Las tablas contienen filas, todas con el mismo número de columnas, cada una de las cuales representa un registro. Existen varios tipos de bases de datos no relacionales, como almacenes de clave-valor, almacenes de columnas y bases de datos de gráficos y documentos. Azure Table Storage es un almacén de clave-valor, Cassandra es un almacén de columnas y Neo4j es una base de datos de grafos. MongoDB y DocumentDB son bases de datos de documentos. Aunque existen diferencias clave entre los distintos tipos de bases de datos NoSQL, también tienen características en común. Las bases de datos NoSQL están diseñadas para escalar horizontalmente y no solo hacia arriba, lo que significa que, si bien las bases de datos relacionales se pueden escalar simplemente agregando recursos de hardware adicionales, es más difícil escalar horizontalmente una base de datos relacional. No es fácil distribuir los datos entre varias particiones una vez que comienza a tocar el techo en la CPU, el disco y la memoria, y ya no puede escalar hacia arriba. A diferencia de las bases de datos relacionales, las bases de datos NoSQL están diseñadas para escalar tanto como sea necesario, lo que hace que sea mucho más fácil alcanzar la escala de Internet. No están sujetos a una estructura tabular y columnar rígida, por lo que los datos se pueden organizar sin esquemas. Libre de esquemas significa que ya no necesitamos lo que tradicionalmente consideraríamos como registros en la base de datos. Somos libres de almacenar información de una manera que puede no ser la misma para cada registro, por lo que podríamos tener diferentes números y tipos de columnas, no solo con diferentes valores, sino también con diferentes tipos de datos. Por diseño, las bases de datos NoSQL se crearon para simplificar, en términos de qué tipo de información se puede escribir. No son tan sólidas como las bases de datos relacionales tradicionales como SQL Server u Oracle, y ni siquiera intentan reflejar o proporcionar la funcionalidad completa de estas. Esta es la razón por la que pueden superar a las bases de datos relacionales a gran escala, escalando fácilmente y no solo hacia arriba.

Debido a que las bases de datos NoSQL no tienen esquemas, son escalables y fáciles de usar, nos brindan más opciones para nuestras aplicaciones que las que teníamos antes. Las bases de datos relacionales tradicionales definitivamente llegaron para quedarse. Sin embargo, ya no tienen un monopolio firme como opción predeterminada cuando necesitamos elegir una base de datos para nuestras necesidades de desarrollo de aplicaciones. DocumentDB es una base de datos no relacional de documentos que ha sido diseñada para objetos (documentos) de notación de objetos de JavaScript (JSON) con elementos etiquetados que pueden serializarse y deserializarse, generalmente sin esquema. Un documento en términos NoSQL se refiere a un objeto JSON y no debe confundirse con el significado típico de la palabra "documento", que tradicionalmente se refiere a PDF, Word y otros formatos de archivos de documentos similares. En este libro electrónico, exploraremos las características de DocumentDB, cómo se compara con las bases de datos relacionales tradicionales, cómo se puede usar para crear aplicaciones escalables y cómo el uso de muchas de las funcionalidades integradas de Azure permite que sea una excelente opción para las aplicaciones de Internet. Los ejemplos de este libro electrónico se realizarán principalmente con Visual Studio 2017 usando C#; sin embargo, algunas partes se cubrirán con el lenguaje de consulta incorporado de DocumentDB, DocumentDB SQL, que es muy similar al SQL estándar, y también se usará JSON. En el lado del servidor, se utilizará JavaScript. Para maximizar su potencial de aprendizaje, se supone que tiene conocimientos previos de bases de datos relacionales y SQL, así como dominio de C#, y está familiarizado con JSON y JavaScript. Los ejemplos de código de Visual Studio se pueden encontrar aquí. A lo largo de este libro, los términos Cosmos DB y DocumentDB se pueden usar indistintamente, ya que Cosmos DB se originó a partir de DocumentDB. El viaje debe ser divertido de seguir, los ejemplos fáciles de implementar y las habilidades útiles para ayudarlo a comprender lo que es posible con esta increíble base de datos NoSQL. ¡Disfrutar!

# Conceptos básicos de DocumentDB

DocumentDB, como su nombre lo indica, almacena datos como "documentos", que en realidad son objetos JSON. En una base de datos relacional, los registros se almacenan como filas en una tabla con columnas específicas utilizando un esquema definido. Sin embargo, en un almacén de documentos, los registros son documentos y cada uno contiene propiedades específicas, sin esquema. Para comprender las diferencias entre una base de datos relacional y una de documentos, consideremos la siguiente tabla.

Tabla : Principales diferencias entre una base de datos relacional y documental

|  |  |
| --- | --- |
| Base de datos relacional | Base de datos de documentos |
| Filas | Documentos |
| Columnas y tipos de datos | Propiedades |
| Esquemas definidos y tipificados | Sin esquemas |
| Altamente normalizado | Mayormente desnormalizado |
| Robusto y maduro | Simple y delgado |
| Escala verticalmente (más hardware) | Escala horizontalmente |

Una tabla de base de datos relacional tiene una estructura fija y para realizar cambios en una tabla, como agregar una nueva columna, cambiar una columna específica para permitir valores NULL o cambiar un tipo de datos, es necesario modificar el esquema de la tabla, lo que puede crear efectos secundarios para los datos existentes ya almacenados en la tabla. Debido a que DocumentDB no tiene esquemas, no está sujeto a las restricciones que tienen las bases de datos relacionales. En el mundo actual, donde el desarrollo de aplicaciones y software está asociado con esquemas de datos que evolucionan con frecuencia, las bases de datos de documentos son una gran combinación. Veamos cómo se almacenarían algunos datos de ejemplo en una tabla relacional tradicional y cómo se almacenarían en una base de datos de documentos.

Tabla : Datos de ejemplo almacenados en una tabla de base de datos relacional

|  |  |  |  |
| --- | --- | --- | --- |
| Nombre | Apellido | Edad | Activo |
| Juan | Pérez | 44 | Activo |
| Cristiano | Ronaldo | 33 | NULL |
| Peter | Pan | 67 | Retirado |

Como podemos ver, los datos están estructurados en un formato tabular con todos los registros teniendo el mismo número de columnas. Tenga en cuenta que utilizo una columna Edad por simplicidad; en una base de datos real, usaría una columna de fecha de nacimiento. Sin embargo, considere el registro resaltado en amarillo. Este registro tiene valores en las columnas Nombre, Apellido y Edad, pero la columna Activo no tiene ningún valor; por lo tanto, se establece en NULL. Los demás registros, a diferencia del amarillo, tienen valores en todas sus columnas. Aunque el formato tabular puede acomodar registros que no necesariamente tienen valores en el mismo número de columnas (al usar valores NULL en columnas que no tienen un valor), todavía requiere que esos registros tengan esa columna, ya que el esquema lo exige. Si tuviéramos que acomodar estos mismos datos en una base de datos de documentos no relacionales como DocumentDB, se vería de la siguiente manera.

Tabla : Datos de ejemplo almacenados en una base de datos de documentos no relacionales

|  |  |
| --- | --- |
| Documento | Datos del documento |
| 1 | {  “Nombre”: “Juan”,  “Apellido”: “Pérez”,  “Edad”: 44,  “Activo”: “Active”  } |
| 2 | {  “Nombre”: “Cristiano”,  “Apellido”: “Ronaldo”,  “Edad”: 33  } |
| 3 | {  “Nombre”: “Peter”,  “Apellido”: “Pan”,  “Edad”: 67,  “Activo”: “Retired”  } |

Solo con fines ilustrativos, los datos de la tabla anterior se muestran en un formato tabular, que se asemeja al formato que utiliza una tabla de base de datos relacional. Sin embargo, una base de datos de documentos no almacena los datos en un formato tabular, sino como una representación binaria de los datos JSON. Lo que se conoce como una tabla en el contexto de una base de datos relacional se llama una colección de documentos en una base de datos de documentos o, simplemente, una colección. Tenga en cuenta que las principales diferencias entre las bases de datos de documentos relacionales y no relacionales son que en las bases de datos de documentos no relacionales los registros se denominan documentos y solo almacenan valores para las propiedades que se utilizan (que corresponderían a columnas en una tabla relacional). Por lo tanto, no es necesario representar las propiedades NULL dentro de los documentos, ya que no se imponen mediante un esquema. Además, una base de datos de documentos no relacionales permite que los documentos almacenen subpropiedades, que no se pueden almacenar explícitamente dentro de una tabla relacional. Un ejemplo de esto sería reemplazar las propiedades Nombre y Apellido con una propiedad NombreCompleto con dos subpropiedades denominadas Nombre y Apellido.

Tabla : Un documento con subpropiedades

|  |  |
| --- | --- |
| Documento | Datos del documento |
| 2 | {  “NombreCompleto”:  {  “Nombre”: “Cristiano”,  “Apellido”: “Ronaldo”  },  “Edad”: 31  } |

Este ejemplo muestra cómo, al tener subpropiedades, el documento no está normalizado con respecto a otros documentos presentes en la misma colección, pero una base de datos de documentos puede acomodar esto perfectamente. Esto significa que cada documento puede tener propiedades y subpropiedades que pueden no estar necesariamente presentes en otros documentos, lo que permite un alto grado de flexibilidad y desnormalización de los datos almacenados.

# Propiedades anidadas maestro-detalle

En las bases de datos no relacionales, las relaciones maestro-detalle se definen mediante propiedades anidadas. En una base de datos relacional, la única forma de lograr esto es tener una relación maestro-detalle entre dos tablas, compartiendo un campo común entre ellas. Consideremos este ejemplo de una relación maestro-detalle dentro de una base de datos relacional.

Tabla : Una relación típica de base de datos relacional maestro-detalle

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nombre | Apellido | Edad | Activo | JugadorId |
| Juan | Pérez | 44 | Activo | 1 |
| Cristiano | Ronaldo | 33 | NULL | 2 |
| Peter | Pan | 67 | Retirado | 3 |

|  |  |
| --- | --- |
| JugadorId | NombreEquipo |
| 2 | Real Madrid |
| 2 | Manchester United |
| 1 | Mars Galactic Soccer |
| 3 | Fantasy FC |

En una base de datos relacional, como en el ejemplo anterior, la única forma de indicar que Cristiano Ronaldo ha jugado tanto con el Real Madrid como con el Manchester United es tener una segunda tabla (de detalles) que contenga los nombres de ambos equipos y agregar una columna adicional a la tabla maestra con un JugadorId único que identifica el registro. Esta columna JugadorId también existirá en la tabla de detalle y será responsable de vincular los registros de detalle con el registro correspondiente en la tabla maestra. Esta es una relación de base de datos relacional maestro-detalle estándar. En una base de datos de documentos, esta relación maestro-detalle se representa como una propiedad con propiedades anidadas. Exploremos cómo se vería esto para el registro del jugador Cristiano Ronaldo.

Tabla : Un documento con propiedades anidadas de maestro-detalle

|  |  |
| --- | --- |
| Documento | Datos del documento |
| 2 | {  “Nombre: “Cristiano”,  “Apellido”: “Ronaldo”,  “Edad”: 31,  “Equipos”: [  {  “Equipo”: “Real Madrid”  },  {  “Equipo”: “Manchester United”  }  ]  } |

Podemos ver que los equipos que se almacenarían en la tabla de detalles en una base de datos relacional tradicional ahora se almacenan dentro de una matriz denominada Equipos, en la que cada equipo es un objeto JSON anidado que contiene una subpropiedad denominada Equipo. Básicamente, la relación maestro-detalle ahora se describe como una matriz de objetos JSON. Sin embargo, el hecho de que la relación maestro-detalle se pueda describir fácilmente como una matriz de objetos JSON en este caso no significa que siempre deba describirse como tal.

# Cuando usar una base de datos de documentos

El poder y la belleza de las bases de datos no relacionales es que usted es libre de implementar y representar la relación entre registros maestro-detalle de la manera que mejor se adapte a sus requisitos comerciales y de aplicación. Tampoco es raro en las bases de datos de documentos repetir algunos datos para que cada documento tenga los datos que necesita sin tener que localizar otros documentos. Si los datos se repiten demasiado, puede optar por organizar los datos repetidos en diferentes documentos, de forma similar a una base de datos relacional tradicional. En cualquier caso, puede organizar la estructura de sus documentos JSON según lo que mejor se adapte a sus requisitos y aplicaciones. En esencia, una base de datos de documentos le brinda la libertad de modelar sus datos de la manera que mejor se adapte a sus necesidades. A pesar de esta flexibilidad, es importante comprender que las bases de datos de documentos son más adecuadas cuando se trabaja con datos que se pueden organizar en documentos jerárquicos ricos que pueden ser casi totalmente autónomos. Si se encuentra modelando una base de datos que contiene muchos documentos relacionados o documentos con una estructura plana, esta es una señal de que una base de datos de documentos probablemente no sea la mejor opción para su aplicación. Cuando necesita una base de datos escalable, una base de datos de documentos es una excelente opción. Las razones principales por las que las bases de datos de documentos pueden escalar horizontalmente son que no imponen reglas complejas o rígidas sobre los datos y que son simples y sencillas por diseño. Por otro lado, las bases de datos relacionales son más adecuadas para manejar requisitos complejos que no necesariamente tienen que escalar horizontalmente.

# ¿Por qué DocumentDB?

DocumentDB es la API de base de datos de documentos altamente escalable de Microsoft que se ejecuta en Azure Cosmos DB. Aunque tiene todas las características de una base de datos de documentos típica, también tiene características que no están disponibles en ninguna otra base de datos de documentos. Exploremos estas características. Con DocumentDB, a diferencia de otras bases de datos de documentos donde necesita definir índices explícitamente, todas las propiedades se indexan automáticamente tan pronto como el documento se agrega a la base de datos. Esto le permite buscar cualquier propiedad dentro de la jerarquía del documento, por muy anidada que esté.

Además, los documentos se pueden buscar utilizando una versión especial de SQL que cualquier persona con experiencia en SQL puede comprender y relacionar fácilmente de una manera intuitiva: este es el propio dialecto de SQL de DocumentDB. Dado que DocumentDB se ejecuta en Azure Cosmos DB, proporciona un entorno del lado del servidor en el que puede ejecutar código JavaScript que puede actualizar varios documentos con un procesamiento transaccional completo. Esta es una manera excelente y fácil de garantizar la coherencia de los datos entre varios documentos. Además, DocumentDB permite un rendimiento ajustable para los requisitos de su aplicación, como mejorar el rendimiento, la indexación y la consistencia. El rendimiento se puede escalar hacia arriba o hacia abajo instantáneamente cambiando el nivel de rendimiento a través de Azure Portal. Aunque DocumentDB indexa automáticamente cada propiedad, aún puede ajustar el sistema para excluir cualquier propiedad o documento que no necesite indexarse, lo que incluso podría ayudar a mejorar el rendimiento en escenarios muy específicos. Aunque DocumentDB es compatible con la consistencia sólida tradicional y eventual (una forma ligeramente diferente asociada con los sistemas de datos distribuidos), también proporciona dos opciones adicionales para brindarle un mayor control sobre las ventajas y desventajas entre el rendimiento y la consistencia. Toda esta funcionalidad está muy bien empaquetada como una solución de plataforma como servicio (PaaS) escalable masivamente y completamente administrada que es muy fácil de configurar y comenzar. No hay nada que instalar, ni sistema operativo ni actualizaciones que administrar, ni réplicas que configurar. A través de Azure Portal, puede comenzar a usar DocumentDB en cuestión de minutos usando un navegador y teniendo una suscripción de Azure. Tómese el tiempo necesario para configurar su suscripción de Azure correctamente. Suena emocionante, así que exploremos algunas de estas características con algunos detalles adicionales.

# Consultas enriquecidas en SQL

Una de las mejores características de DocumentDB es que su lenguaje de consulta nativo es muy similar a SQL. Para aquellos que desarrollan con .NET, también hay un proveedor LINQ. Aunque las consultas modificadas de DocumentDB están escritas en SQL, están profundamente arraigadas en la semántica JSON y JavaScript. Le permiten consultar matrices y datos anidados jerárquicos dentro de documentos y también compartir proyecciones personalizadas de los resultados de sus consultas. Echemos un vistazo a un ejemplo.

Tabla : Una consulta SQL enriquecida de DocumentDB

|  |
| --- |
| -- Consulta enriquecida SQL.  SELECT h.Nombre, h.Apellido  FROM Jugadores AS j  JOIN h IN j.Hijos  WHERE j.Edad = 33 |

En este ejemplo de consulta SQL con sabor, estamos haciendo algunas cosas. La cláusula JOIN básicamente permite que DocumentDB itere a través de todos los elementos secundarios (propiedades) anidados dentro del documento Players. La cláusula WHERE filtra comprobando los documentos que tienen un valor de propiedad Edad igual a 31. Observe que la notación punteada se utiliza para hacer referencia a las propiedades dentro del documento. La notación punteada se puede anidar hasta donde llega la jerarquía del documento. Debido a que en la cláusula SELECT estamos seleccionando dos propiedades en lugar de SELECT \*, DocumentDB proyecta un nuevo objeto JSON que solo contiene las propiedades que se consultan, en lugar de todo el documento como resultado. Una vez que se inserta un documento en DocumentDB, se puede buscar prácticamente al instante, ya que se indexa automáticamente. Este comportamiento de indexación se puede ajustar. Sin embargo, generalmente no hay necesidad.

# Desarrollo del lado del cliente

DocumentDB proporciona varios SDK, que permiten una fácil integración desde su plataforma de desarrollo preferida. Hay SDK para varias de las plataformas de desarrollo más comunes, como .NET, Node.js, JavaScript, Java y Python. No es diferente a otras plataformas, ya que también proporciona una API REST/HTTP que se puede usar siempre que los encabezados de la solicitud HTTP contengan información de autenticación válida y la solicitud apunte al recurso DocumentDB HTTP correcto. Trabajar con la API REST/HTTP es la forma más primitiva de interactuar con DocumentDB; puede volverse muy tedioso cuando necesita concentrar la mayor parte de su atención en la lógica de su aplicación. Por lo tanto, nos centraremos en el uso de .NET SDK en este libro electrónico al realizar el desarrollo del lado del cliente. Si no hay un SDK disponible para su plataforma de desarrollo, puede usar la API REST/HTTP. Sin embargo, también debe ponerse en contacto con el soporte técnico de Azure, informarles qué plataforma está utilizando y proporcionarles sus comentarios. Con suerte, pueden tener en cuenta su opinión e incluir un SDK para su plataforma de desarrollo en su hoja de ruta. El equipo de Azure es muy activo y está comprometido con la plataforma, por lo que si no hay un SDK para su plataforma, es probable que nadie lo haya pedido todavía.

# Desarrollo del lado del servidor

DocumentDB es un entorno de espacio aislado de servidor que le brinda la capacidad de ejecutar la lógica interna, donde residen los datos. La lógica del lado del servidor en DocumentDB se puede envolver en procedimientos almacenados y disparadores, y funciones definidas por el usuario (UDF), que es sorprendentemente familiar al trabajar con sistemas de bases de datos relacionales como Oracle y SQL Server.

Sin embargo, existe una sutil diferencia entre la lógica del lado del servidor escrita en DocumentDB y la lógica del lado del servidor escrita con bases de datos relacionales tradicionales. En DocumentDB, la lógica del lado del servidor está escrita en JavaScript en lugar de SQL, lo que lo convierte en el compañero perfecto para manejar objetos JSON. DocumentDB adopta JavaScript como una especie de SQL moderno al admitir la ejecución de secuencias de comandos transaccionales de forma nativa dentro del motor de la base de datos. Debido a que DocumentDB es un servicio totalmente alojado, no puede permitir que los scripts que funcionan mal se ejecuten indefinidamente, ya que esto podría poner en riesgo la integridad de todo el servicio. Por lo tanto, aplica un paradigma conocido como ejecución limitada, que básicamente determina cuánto tiempo puede ejecutarse su lógica antes de que se agote el tiempo de espera. Toda la ejecución de la lógica del lado del servidor es completamente transaccional, lo que significa que si actualiza algunos documentos y se produce un error, o uno de sus scripts se agota debido a una ejecución limitada antes de que realmente se complete, entonces todas las actualizaciones hasta ese momento se revierten automáticamente. . Si su código del lado del servidor se completa con éxito, entonces se garantiza que todas las actualizaciones se confirmarán juntas. Esto hace que DocumentDB sea una opción aún más convincente.

# Escalabilidad

Con las bases de datos NoSQL, la escalabilidad es clave para el éxito y DocumentDB lo cumple. Ya es el back-end elegido por servicios como Xbox y Office OneNote, que dependen de DocumentDB para bases de datos que contienen decenas de terabytes de documentos JSON, más de un millón de usuarios y una disponibilidad de tiempo de actividad del 99,9 %. Para darle una idea de cuánto puede escalar DocumentDB: puede crecer tanto como pueda pagar o hasta el final del hardware disponible en los centros de datos de Azure, el límite que alcance primero. Esta es una declaración profunda y una indicación clara de lo que DocumentDB puede manejar y cuánto puede escalar. También sirve como base utilizada para la creación de Cosmos DB. En términos simples, DocumentDB puede escalar masivamente a cientos de terabytes e incluso petabytes a través de miles de nodos. Puede escalar hacia arriba y hacia abajo y también hacia afuera. Es capaz de escalar hacia arriba y hacia abajo sin problemas, lo que consiste en una combinación de potencia informática y capacidad de almacenamiento. DocumentDB también puede escalar agregando más colecciones. Una colección de documentos puede verse esencialmente como una unidad de escala. Si su base de datos crece más allá de 10 GB, puede escalar horizontalmente simplemente agregando más colecciones y luego dividiendo sus datos en varias colecciones.

# Consistencia

Otra característica importante que hace que DocumentDB sea excelente es su capacidad para ajustar la consistencia. Por lo general, hay una compensación entre el rendimiento y la consistencia.

Básicamente, la consistencia fuerte ralentiza las lecturas y escrituras y la consistencia eventual no siempre devuelve los datos más actuales. Con una gran consistencia, obtiene resultados de consulta consistentes a medida que los escritores realizan cambios en la base de datos, pero paga un precio en rendimiento, ya que todas las consultas deben esperar hasta que todas las réplicas se hayan actualizado con los últimos cambios, lo que obviamente ralentiza un poco las cosas. Por el contrario, la consistencia eventual le brinda el mejor rendimiento, pero no puede confiar completamente en los resultados de la consulta. Pueden devolver datos que no son del todo coherentes con lo que otros usuarios pueden estar actualizando, dado que no todas las réplicas están necesariamente actualizadas. DocumentDB es compatible con estos dos métodos de consistencia y con tres métodos adicionales que se encuentran en medio de consistencias fuertes y eventuales. Estos tres métodos adicionales se denominan obsolescencia limitada, sesión y prefijo coherente. La obsolescencia limitada le permite tolerar resultados de consulta inconsistentes al garantizar que esos resultados sean al menos lo suficientemente consistentes dentro de un período de tiempo específico. La consistencia de la sesión, que en realidad es el método de consistencia predeterminado que se usa en DocumentDB, se puede considerar como una experiencia híbrida. Se garantiza que los escritores tendrán una gran consistencia para los datos que ellos mismos escribieron, mientras que todos los demás operan con consistencia eventual. El prefijo consistente garantiza que, en ausencia de más escrituras, las réplicas dentro del grupo finalmente convergen. Las cuentas de Azure Cosmos DB que están configuradas con coherencia de prefijo coherente pueden asociar cualquier número de regiones de Azure con su cuenta de Azure Cosmos DB. DocumentDB permite ajustar y cambiar la consistencia, brindándole la flexibilidad para trabajar con el enfoque que mejor se adapte a los requisitos y necesidades de su negocio. Puede encontrar más información sobre los niveles de coherencia de datos ajustables en Azure Cosmos DB mediante la API de DocumentDB aquí.

# Costos

Antes de la introducción de Cosmos DB, cuando DocumentDB se ofrecía como un servicio independiente, los costos se basaban en niveles y colecciones preestablecidos. En DocumentDB antes de Cosmos DB, una colección no solo era una unidad de escala, sino que también estaba directamente relacionada con los costos y los precios. Pagarías por colección, y cada colección tenía una capacidad de almacenamiento de hasta 10 GB. Con la introducción de Azure Cosmos DB, el antiguo esquema de precios basado en la combinación de colecciones y niveles se volvió irrelevante y no completamente elástico. Los antiguos niveles de rendimiento de DocumentDB S1, S2 y S3 no ofrecían la flexibilidad que ahora ofrecen las colecciones de la API de DocumentDB en el futuro con Cosmos DB. Esto se debió a que en los niveles de rendimiento S1, S2 y S3, tanto el rendimiento como la capacidad de almacenamiento estaban preestablecidos y no ofrecían elasticidad.

Azure Cosmos DB ahora ofrece la capacidad de personalizar su rendimiento y almacenamiento, ofreciéndole mucha más flexibilidad para escalar a medida que cambian las necesidades de su aplicación. Puede encontrar más información sobre este tema aquí.

# Resumen

En este capítulo, aprendimos rápidamente sobre algunas de las características más destacadas de la base de datos de documentos NoSQL y específicamente sobre DocumentDB y cómo se compara con las bases de datos relacionales tradicionales. En particular, nos enfocamos en describir cómo DocumentDB está diseñado desde cero para escalar y trabajar con documentos JSON jerárquicos sin esquema, a diferencia de las tablas de bases de datos tabulares tradicionales que requieren un esquema y uniones complejas para ensamblar. También describimos rápidamente cómo DocumentDB permite la búsqueda instantánea en documentos y cómo cada propiedad se indexa automáticamente cuando se agrega un documento a una colección, lo que le permite consultarlos rápida y fácilmente utilizando una sintaxis familiar similar a SQL. Además, discutimos cómo la programación del lado del cliente y del servidor es posible a través de varios SDK específicos de la plataforma y el uso de JavaScript con procesamiento transaccional completo. Finalmente, describimos cómo DocumentDB permite la consistencia ajustable y admite el escalado elástico. También hablamos brevemente sobre los costos. Esto prepara el escenario para los siguientes capítulos y nos brinda un punto de partida sólido y una comprensión de alto nivel de lo que DocumentDB puede permitirnos lograr. En los siguientes capítulos, nos centraremos en profundizar en cada una de las funciones de DocumentDB y explorar cada una de ellas escribiendo código. ¡Gracias por leer!