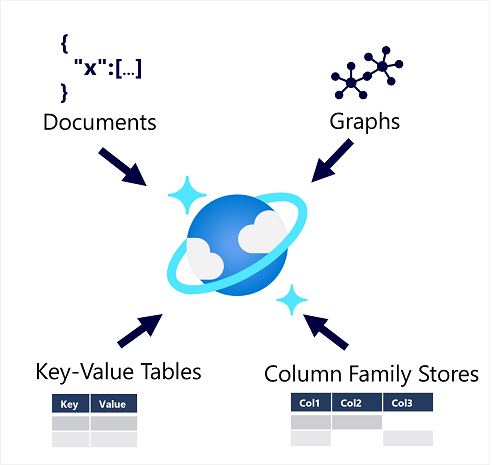
**Descripción de Azure Cosmos DB**

Completado100 XP

* 5 minutos



Azure Cosmos DB admite varias interfaces de programación de aplicaciones (API) que permiten a los desarrolladores usar la semántica de programación de muchos tipos comunes de almacén de datos para trabajar con datos en una base de datos Cosmos DB. La estructura de datos interna se abstrae, lo cual permite a los desarrolladores usar Cosmos DB para almacenar y consultar datos mediante API con las que ya están familiarizados.

**Nota**

Una *API* es una *interfaz de programación de aplicaciones*. Los sistemas de administración de bases de datos (y otros marcos de software) proporcionan un conjunto de API que los desarrolladores pueden usar para escribir programas que necesitan tener acceso a datos. Las API serán diferentes para los distintos sistemas de administración de bases de datos.

Cosmos DB usa índices y particiones para proporcionar un rendimiento rápido de lectura y escritura y se puede escalar a volúmenes masivos de datos. Puede habilitar escrituras en varias regiones, agregando las regiones de Azure que prefiera a su cuenta de Cosmos DB para que los usuarios distribuidos globalmente puedan trabajar con datos en su réplica local.

**Cuándo usar Cosmos DB**

Cosmos DB es un sistema de administración de bases de datos muy escalable. Cosmos DB asigna automáticamente espacio para las particiones en un contenedor y cada partición puede crecer hasta un tamaño de 10 GB. Los índices se crean y se mantienen de forma automática. No hay prácticamente ninguna sobrecarga administrativa.

Cosmos DB es un servicio fundamental de Azure. Muchos de los productos de Microsoft usan Cosmos DB para aplicaciones críticas a escala global, como Skype, Xbox, Microsoft 365 y Azure, entre muchos otros. Cosmos DB es muy recomendable para los escenarios siguientes:

* *IoT y telemática*. Estos sistemas suelen ingerir grandes cantidades de datos en ráfagas de actividad frecuentes. Cosmos DB puede aceptar y almacenar esta información con rapidez. lo que permite que servicios analíticos como Azure Machine Learning, Azure HDInsight o Power BI puedan hacer uso de esos datos. Además, los datos se pueden procesar en tiempo real a través de funciones de Azure Functions que se activan a medida que los datos van llegando a la base de datos.
* *Comercio y marketing*. Microsoft usa Cosmos DB en sus plataformas de comercio electrónico propias que se ejecutan como parte de la Tienda Windows y Xbox Live. También se usa en el sector comercial para almacenar los datos de catálogo y para el suministro de eventos en las canalizaciones de procesamiento de pedidos.
* *Juegos*. El nivel de base de datos es un componente fundamental de las aplicaciones de juegos. Los juegos modernos realizan el procesamiento de los elementos grafos en los clientes de consola o dispositivos móviles, pero utilizan la nube para ofrecer contenido personalizado y a medida, como estadísticas dentro del juego, integración con las redes sociales y los marcadores de puntuaciones. A menudo, los juegos requieren latencias de un solo milisegundo en las lecturas y escrituras para proporcionar una experiencia de juego inmersiva. Una base de datos de un juego debe ser rápida y capaz de manejar los picos masivos en la velocidad de las solicitudes cuando se inicia un nuevo juego y se actualizan las características.
* *Aplicaciones web y para dispositivos móviles*. Azure Cosmos DB se usa normalmente en aplicaciones web y móviles y sirve para modelar interacciones sociales, para la integración con servicios de terceros y para la creación de experiencias personalizadas enriquecidas. Se pueden usar SDK de Cosmos DB con el fin de compilar aplicaciones para iOS y Android completas con el marco Xamarin Framework, muy popular.

Para obtener más información sobre los usos de Cosmos DB, consulte [Casos de uso comunes de Azure Cosmos DB](https://docs.microsoft.com/es-ES/azure/cosmos-db/use-cases).

**Identificación de las API de Azure Cosmos DB**

Completado100 XP

* 6 minutos

Azure Cosmos DB admite varias API, lo cual permite a los desarrolladores migrar fácilmente datos desde almacenes NoSQL usados habitualmente y aplicar sus aptitudes de programación existentes. Al aprovisionar una nueva instancia de Cosmos DB, seleccione la API que quiere usar. La elección de la API depende de muchos factores, como el tipo de datos que se van a almacenar, la necesidad de admitir aplicaciones existentes y las aptitudes de API de los desarrolladores que trabajarán con el almacén de datos.

**API Core (SQL)**

La API nativa de Cosmos DB administra los datos en formato de documento JSON y, a pesar de ser una solución de almacenamiento de datos NoSQL, usa sintaxis SQL para trabajar con los datos.

Una consulta SQL para una base de datos de Cosmos DB que contiene datos del cliente podría ser similar a esta:

SQLCopiar

SELECT \*

FROM customers c

WHERE c.id = "joe@litware.com"

El resultado de esta consulta consta de uno o varios documentos JSON, como se muestra aquí:

JSONCopiar

{

"id": "joe@litware.com",

"name": "Joe Jones",

"address": {

"street": "1 Main St.",

"city": "Seattle"

}

}

**MongoDB API**

MongoDB es una base de datos de código abierto popular en la que los datos se almacenan en formato JSON binario (BSON). La API de MongoDB de Azure Cosmos DB permite a los desarrolladores usar bibliotecas de cliente de MongoDB en y código para trabajar con datos en Azure Cosmos DB.

El lenguaje de consulta de MongoDB (MongoDB Query Language, MQL) usa una sintaxis compacta orientada a objetos en la que los desarrolladores usan *objetos* para llamar a *métodos*. Por ejemplo, la consulta siguiente usa el método **find** para consultar la colección **products** en el objeto **db**:

JavaScriptCopiar

db.products.find({id: 123})

Los resultados de esta consulta constan de documentos JSON, similares a los siguientes:

JSONCopiar

{

"id": 123,

"name": "Hammer",

"price": 2.99}

}

**Table API**

Table API se usa para trabajar con datos en tablas de clave-valor, de forma similar a Azure Table Storage. Table API de Azure Cosmos DB ofrece mayor escalabilidad y rendimiento que Azure Table Storage.

Por ejemplo, puede definir una tabla denominada **Clientes** de la siguiente forma:

| **PartitionKey** | **RowKey** | **Nombre** | **Email** |
| --- | --- | --- | --- |
| 1 | 123 | Joe Jones | joe@litware.com |
| 1 | 124 | Samir Nadoy | samir@northwind.com |

Posteriormente, puede usar Table API de Cosmos DB a través de uno de los SDK específicos del lenguaje para realizar llamadas al punto de conexión de servicio para recuperar datos de la tabla. Por ejemplo, la siguiente solicitud devuelve la fila que contiene el registro de *Samir Nadoy* en la tabla anterior:

textCopiar

https://endpoint/Customers(PartitionKey='1',RowKey='124')

**Cassandra API**

Cassandra API es compatible con Apache Cassandra, que es una base de datos de código abierto popular que usa una estructura de almacenamiento de familia de columnas. Las familias de columnas son tablas, similares a las de una base de datos relacional, con la excepción de que no es obligatorio que cada fila tenga las mismas columnas.

Por ejemplo, puede crear una tabla de **Empleados** como esta:

| **id** | **Nombre** | **Manager** |
| --- | --- | --- |
| 1 | Sue Smith |  |
| 2 | Ben Chan | Sue Smith |

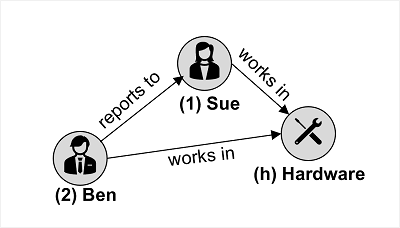
Cassandra admite una sintaxis basada en SQL, por lo que una aplicación cliente podría recuperar el registro de *Ben Chan* como se muestra a continuación:

SQLCopiar

SELECT \* FROM Employees WHERE ID = 2

**API de Gremlin**

Gremlin API se usa con datos en una estructura de grafos. en el que las entidades se definen como *vértices* que forman nodos en el gráfico conectado. Los nodos se conectan mediante *bordes* que representan relaciones, como esta:



En el ejemplo de la imagen se muestran dos tipos de vértices (empleado y departamento) y bordes que los conectan (el empleado "Ben" depende de la empleada "Sue" y ambos empleados trabajan en el departamento "Hardware").

La sintaxis de Gremlin incluye funciones para operar en vértices y bordes, y esto permite insertar, actualizar, eliminar y consultar datos en el gráfico. Por ejemplo, puede usar el código siguiente para agregar un nuevo empleado llamado *Alice* que dependa de la empleada con el identificador **1** (*Sue*)

Copiar

g.addV('employee').property('id', '3').property('firstName', 'Alice')

g.V('3').addE('reports to').to(g.V('1'))

La consulta siguiente devuelve todos los vértices de *empleado*, por orden de identificador.

Copiar

g.V().hasLabel('employee').order().by('id')

**Objetivos de aprendizaje**

En este módulo aprenderá a:

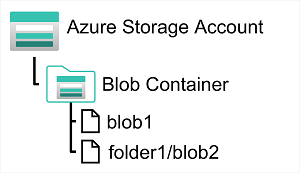
* Descripción de las características y funcionalidades de Azure Blob Storage
* Descripción de las características y funcionalidades de Azure Data Lake Gen2
* Descripción de las características y funcionalidades de Azure File Storage
* Descripción de las características y funcionalidades de Azure Table Storage
* Aprovisionamiento y uso de una cuenta de Azure Storage

**Exploración de Azure Blob Storage**

Completado100 XP

* 4 minutos

Azure Blob Storage es un servicio que le permite almacenar grandes cantidades de datos no estructurados como objetos binarios grandes, o *blobs*, en la nube. Los blobs son una manera eficaz de almacenar archivos de datos en un formato optimizado para el almacenamiento basado en la nube, y las aplicaciones pueden leerlos y escribirlos mediante la API de Azure Blob Storage.



En una cuenta de Azure Storage, los blobs se almacenan en *contenedores*. Un contenedor proporciona una manera cómoda de agrupar blobs relacionados. Puede controlar quién puede leer y escribir blobs dentro de un contenedor en el nivel de contenedor.

Dentro de un contenedor, puede organizar los blobs en una jerarquía de carpetas virtuales, similares a los archivos de un sistema de archivos en un disco. Sin embargo, de manera predeterminada, estas carpetas no son más que una forma de utilizar un carácter "/" en el nombre de un blob para organizar los blobs en espacios de nombres. Las carpetas son puramente virtuales y no es posible hacer operaciones de nivel de carpeta para controlar el acceso ni hacer operaciones masivas.

Azure Blob Storage admite tres tipos de blobs diferentes:

* **Blobs en bloques**. Un blob en bloques se trata como un conjunto de bloques. Cada bloque puede tener un tamaño distinto, de hasta 100 MB. Un blob en bloques puede contener hasta 50 000 bloques, con un tamaño máximo de más de 4,7 TB. El bloque es la cantidad más pequeña de datos que se puede leer o escribir como una unidad individual. Los blobs en bloques se recomiendan especialmente para almacenar objetos binarios grandes discretos que cambian con poca frecuencia.
* **Blobs en páginas**. Un blob en páginas se organiza como una colección de páginas de tamaño fijo de 512 bytes. Un blob en páginas está optimizado para admitir operaciones de lectura y escritura aleatorias; puede capturar y almacenar datos para una sola página si es necesario. Un blob en páginas puede contener hasta 8 TB de datos. Azure usa blobs en páginas para implementar el almacenamiento de discos virtuales de las máquinas virtuales.
* **Blobs en anexos**. Un blob en anexos es un blob en bloques optimizado para admitir operaciones de anexión. Solo puede agregar bloques al final de un blob en anexos; no se admite la actualización o eliminación de bloques existentes. Cada bloque puede tener un tamaño distinto, de hasta 4 MB. El tamaño máximo de un blob en anexos es de algo más de 195 GB.

El almacenamiento de blobs proporciona tres niveles de acceso, que ayudan a equilibrar la latencia de acceso y el costo de almacenamiento:

* El nivel de *acceso frecuente* es el predeterminado. Este nivel se usa para los blobs a los que se accede con frecuencia. Los datos de blob se almacenan en medios de alto rendimiento.
* El nivel *Esporádico* tiene un rendimiento inferior e incurre en cargos de almacenamiento reducidos en comparación con el nivel Frecuente. Use el nivel de acceso esporádico para los datos a los que se accede con poca frecuencia. Es habitual que el acceso a los blobs recién creados sea más frecuente al principio y menos frecuente a medida que pasa el tiempo. En estas situaciones, puede crear el blob en el nivel de acceso frecuente, pero migrarlo al nivel de acceso esporádico más adelante. Puede migrar un blob del nivel de acceso esporádico al frecuente.
* El nivel *Archivo* proporciona el menor costo de almacenamiento, pero una mayor latencia. El nivel de acceso de archivo está pensado para los datos históricos que no deben perderse, pero que raramente se necesitan. Los blobs del nivel de acceso de archivo se almacenan de forma eficaz en un estado sin conexión. La latencia de lectura típica para los niveles de acceso frecuente y esporádico es de unos milisegundos, pero para el nivel de acceso de archivo los datos pueden tardar horas en estar disponibles. Para recuperar un blob desde el nivel de acceso de archivo, debe cambiar el nivel de acceso a acceso frecuente o esporádico. Con ello, el blob se rehidratará. Solo puede leer el blob una vez que se ha completado el proceso de rehidratación.

Puede crear directivas de administración del ciclo de vida para los blobs de una cuenta de almacenamiento. Una directiva de administración del ciclo de vida puede trasladar automáticamente un blob de acceso frecuente a acceso esporádico y, a continuación, al nivel de acceso de archivo, a medida que pasa el tiempo y se usa con menos frecuencia (la directiva se basa en el número de días transcurridos desde la última modificación). Una directiva de administración del ciclo de vida también puede organizarse para eliminar blobs obsoletos.

**Creación de una cuenta de Cosmos DB**

Para usar Cosmos DB, debe aprovisionar una cuenta de Cosmos DB en su suscripción de Azure. En este ejercicio, aprovisionará una cuenta de Cosmos DB que usa la API de núcleo (SQL).

1. En Azure Portal, seleccione **+ Crear un recurso** en la parte superior izquierda y busque *Azure Cosmos DB*. En los resultados, seleccione **Azure Cosmos DB** y seleccione **Crear**.
2. En el mosaico **Núcleo (SQL): Recomendado**, seleccione **Crear**.
3. Escriba los detalles siguientes y seleccione **Revisar y crear**:
   * **Suscripción**: si usa un espacio aislado, seleccione la opción *Concierge Subscription* (Suscripción de Concierge). En caso contrario, seleccione su suscripción de Azure.
   * **Grupo de recursos**: si usa un espacio aislado, seleccione el grupo de recursos existente (que tendrá un nombre como *learn-xxxx…*). De lo contrario, cree un grupo de recursos con el nombre que prefiera.
   * **Nombre de cuenta**: escriba un nombre único
   * **Ubicación**: cualquier ubicación disponible
   * **Capacity mode** (Modo de capacidad): rendimiento aprovisionado
   * **Apply Free-Tier Discount** (Aplicar descuento de nivel Gratis): seleccione Aplicar si está disponible
   * **Limit total account throughput** (Limitar el rendimiento total de la cuenta): no seleccionado
4. Una vez validada la configuración, seleccione **Crear**.
5. Espere a que la implementación finalice. A continuación, vaya al recurso implementado.

**Crear una base de datos de ejemplo**

1. En la página de la nueva cuenta de Cosmos DB, en el panel de la izquierda, seleccione **Explorador de datos**.
2. En la página del **Explorador de datos**, seleccione **Start with Sample** (Iniciar con ejemplo) y, posteriormente, observe el estado en el panel de la parte inferior de la pantalla hasta que se haya creado la base de datos **SampleDB** y el contenedor de **Personas** de ejemplo (esta acción puede tardar unos minutos).
3. Seleccione **Cerrar** en el mensaje de notificación.

**Visualización y creación de elementos**

1. En la página del Explorador de datos, expanda la base de datos **SampleDB** y el contenedor de Personas y seleccione **Elementos** para ver una lista de elementos del contenedor. Los elementos representan a personas, cada una con un identificador único, un nombre, una edad y otras propiedades.
2. Seleccione cualquiera de los elementos de la lista para ver una representación JSON de los datos del elemento.
3. En la parte superior de la página, seleccione **Nuevo elemento** para crear un nuevo elemento en blanco.
4. Modifique el JSON del nuevo elemento como se muestra a continuación y, posteriormente, seleccione **Guardar**.

JSONCopiar

{

"id": "123",

"firstname": "Bob",

"age": 54

}

1. Después de guardar el nuevo elemento, observe que las propiedades de metadatos adicionales se agregan automáticamente.

**Consulta de la base de datos**

1. En la página del **Explorador de datos**, seleccione el icono **Nueva consulta de SQL**.
2. En el editor de consultas SQL, revise la consulta predeterminada (SELECT \* FROM c) y use el botón SELECT \* FROM c para ejecutarla.
3. Revise los resultados, que incluyen la representación JSON completa de todos los elementos.
4. Modifique la consulta del siguiente modo:

SQLCopiar

SELECT c.id, c.firstname, c.age

FROM c

WHERE c.age > 40

1. Use el botón **Ejecutar consulta** para ejecutar la consulta revisada y revisar los resultados, que incluye JSON que contiene los campos de identificador, nombre y edad para los elementos de persona con una edad superior a 40.
2. Cierre el editor de consultas SQL y descarte los cambios.

Ha visto cómo crear y consultar entidades JSON en una base de datos de Cosmos DB mediante la interfaz del explorador de datos de Azure Portal. En un escenario real, un desarrollador de aplicaciones utilizará uno de los muchos kits de desarrollo de software (SDK) específicos del lenguaje de programación para llamar a la API de núcleo (SQL) y trabajar con datos en la base de datos.