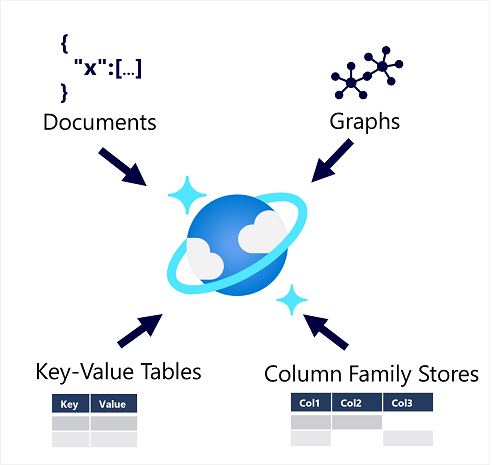
**Descripción de Azure Cosmos DB**

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 |

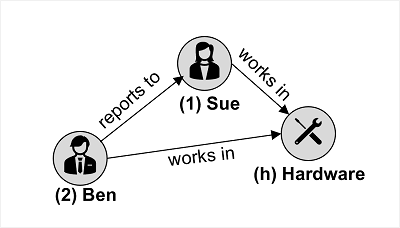
Cassandra admite una sintaxis basada en SQL, por lo que una aplicación cliente podría recuperar el registro de *Ben Chan* como se muestra a continuación:

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')