**¿Qué es la opción Hiperescala (Citus) de Azure Database for PostgreSQL?**

Azure Database for PostgreSQL es un servicio de base de datos relacional en la nube de Microsoft diseñado para desarrolladores. Se basa en la versión Community del motor de base de datos [PostgreSQL](https://www.postgresql.org/) de código abierto.

Hiperescala (Citus) es una opción de implementación que escala horizontalmente las consultas entre varias máquinas mediante el particionamiento. Su motor de consultas paraleliza las consultas SQL entrantes en estos servidores para agilizar las respuestas en conjuntos de datos grandes. Proporciona servicios a las aplicaciones que requieren mayor escala y mejor rendimiento. Por ejemplo, suele ser el caso de las cargas de trabajo que se aproximan a los 100 GB de datos (o que ya los superan).

Hiperescala (Citus) ofrece lo siguiente:

* Escalado horizontal entre varias máquinas mediante particionamiento
* Paralelización de consultas entre estos servidores, lo que agiliza las respuestas en conjuntos de datos grandes
* Una excelente compatibilidad con aplicaciones de varios inquilinos, análisis operativos en tiempo real y cargas de trabajo transaccionales de alto rendimiento

Las aplicaciones compiladas para PostgreSQL puede ejecutar consultas distribuidas en Citus con las [bibliotecas de conexiones](https://docs.microsoft.com/es-es/azure/postgresql/concepts-connection-libraries) estándar y unos cambios mínimos.

## **Nodos (SERVIDORES DE CITUS)**

El tipo de hospedaje Hiperescala (Citus) permite a los servidores de Azure Database for PostgreSQL (llamados "nodos") coordinarse entre sí en una arquitectura "nada compartido". Los nodos de un grupo de servidores almacenan colectivamente más datos y usan más núcleos de CPU de lo que sería posible en un único servidor. La arquitectura también permite escalar la base de datos al agregar más nodos al grupo de servidores.

### Coordinación y trabajos

Cada grupo de servidores tiene un nodo de coordinación y varios trabajos. Las aplicaciones envían sus consultas al nodo de coordinación, que las retransmite a los trabajos pertinentes y acumula los resultados. Las aplicaciones no son capaces de conectarse directamente a los trabajos.

Hiperescala (Citus) permite al administrador de bases de datos distribuir tablas y almacenar distintas filas en nodos de trabajo diferentes. Las tablas distribuidas son la clave para el rendimiento de Hiperescala (Citus). Si no se distribuyen las tablas, las deja totalmente en el nodo de coordinación y no pueden aprovechar el paralelismo entre máquinas.

Para cada consulta sobre las tablas distribuidas, el coordinador la enruta a un único nodo de trabajo o la paraleliza en varias según si los datos necesarios se encuentren en un único nodo o en varios. El coordinador decide qué hacer tras consultar las tablas de metadatos. Estas tablas hacen un seguimiento de los nombres DNS y del estado de los nodos de trabajo, y de la distribución de datos entre nodos.

## Tipos de tablas

Hay tres tipos de tablas en un grupo de servidores Hiperescala (Citus); cada uno se almacena de forma distinta y se usa con distintos fines.

### **Tipo 1: Tablas distribuidas**

El primer tipo, y el más común, es la tabla distribuida. Aparece como una tabla normal para las instrucciones SQL, pero está particionada horizontalmente en nodos de trabajo. Esto significa que las filas de la tabla se almacenan en nodos diferentes, en tablas de fragmentos llamadas particiones de base de datos.

Hiperescala (Citus) no solo ejecuta instrucciones SQL sino también DDL en un clúster. El cambio del esquema de una tabla distribuida se propaga para actualizar todas las particiones de la tabla en los nodos de trabajo.

#### **Columna de distribución**

Hiperescala (Citus) usa el particionamiento algorítmico para asignar filas a las particiones de base de datos. La asignación se realiza de forma determinista en función del valor de una columna de la tabla denominada columna de distribución. El administrador de clústeres debe designar esta columna al distribuir una tabla. Realizar la elección correcta es importante para el rendimiento y la funcionalidad.

### **Tipo 2: tablas de referencia**

Una tabla de referencia es un tipo de tabla distribuida en la que todo su contenido se concentra en una sola partición de base de datos. La partición se replica en todos los nodos de trabajo. Las consultas en cualquier nodo pueden acceder a la información de referencia localmente, sin la sobrecarga de red que supone solicitar filas de otro nodo. Las tablas de referencia no tienen columna de distribución porque no es necesario distinguir particiones de base de datos independientes por cada fila.

Las tablas de referencia suelen ser pequeñas y se usan para almacenar los datos que son pertinentes para las consultas que se ejecutan en cualquier nodo de trabajo. Por ejemplo, valores enumerados, como los estados de pedidos o las categorías de productos.

### **Tipo 3: tablas locales**

Cuando se usa Hiperescala (Citus), el nodo de coordinación al que se conecta es una base de datos PostgreSQL normal. Puede crear tablas normales en el coordinador y elegir no particionarlas.

Un buen candidato para las tablas locales serían pequeñas tablas administrativas que no participan en las consultas de combinación. Por ejemplo, una tabla de usuarios para el inicio de sesión en una aplicación y la autenticación.

## **Particiones de base de datos**

En la sección anterior, se describe la manera en que las tablas distribuidas se almacenan como particiones de base de datos en nodos de trabajo. En esta sección se tratan detalles más técnicos.

La tabla de metadatos pg\_dist\_shard del coordinador contiene una fila para cada partición de base de datos de cada tabla distribuida en el sistema. La fila asocia un identificador de partición con un intervalo de enteros en un espacio de hash (shardminvalue, shardmaxvalue).

SQLCopiar

SELECT \* from pg\_dist\_shard;

logicalrelid | shardid | shardstorage | shardminvalue | shardmaxvalue

---------------+---------+--------------+---------------+---------------

github\_events | 102026 | t | 268435456 | 402653183

github\_events | 102027 | t | 402653184 | 536870911

github\_events | 102028 | t | 536870912 | 671088639

github\_events | 102029 | t | 671088640 | 805306367

(4 rows)

Si el nodo de coordinación quiere determinar qué partición de base de datos contiene una fila de github\_events, aplica un algoritmo hash al valor de la columna de distribución de la fila. A continuación, comprueba qué intervalo de la partición de base de datos contiene el valor con hash. Los intervalos se definen de modo que la imagen de la función hash sea su unión independiente.

### Colocaciones de particiones

Supongamos que la partición de base de datos 102027 está asociada a la fila en cuestión. La fila se leerá o escribirá en una tabla denominada github\_events\_102027 de uno de los nodos de trabajo. ¿Qué trabajo? Esto lo determinan por completo las tablas de metadatos. La asignación de la partición de base de datos a un nodo de trabajo se conoce como colocación de la partición.

El nodo de coordinación vuelve a escribir consultas en fragmentos que hacen referencia a las tablas específicas como github\_events\_102027, y ejecuta esos fragmentos en los nodos de trabajo adecuados. Este es un ejemplo de una consulta que se ejecuta en segundo plano para buscar el nodo que contiene el identificador de partición de base de datos 102027.

SQLCopiar

SELECT

shardid,

node.nodename,

node.nodeport

FROM pg\_dist\_placement placement

JOIN pg\_dist\_node node

ON placement.groupid = node.groupid

AND node.noderole = 'primary'::noderole

WHERE shardid = 102027;

ResultadosCopiar

┌─────────┬───────────┬──────────┐

│ shardid │ nodename │ nodeport │

├─────────┼───────────┼──────────┤

│ 102027 │ localhost │ 5433 │

└─────────┴───────────┴──────────┘

### Tablas Distribuidas

1. Fragmentadas en diferentes lugares
2. tamaño superior a 10GB para gran volumen de datos
3. Reto de diseño: elegir la forma de distribuir o fragmentar, de esto depende la eficiencia (velocidad) de las consultas

### Tablas de Referencia

1. Pequeñas < 10GB
2. Se replican en todos los nodos para facilitar consultas entre “TablasFragmentadas” <-> “TablasReferencias”

### Tablas Locales

1. Para administración interna
2. contienen metadatos para uso del nodo cordinador