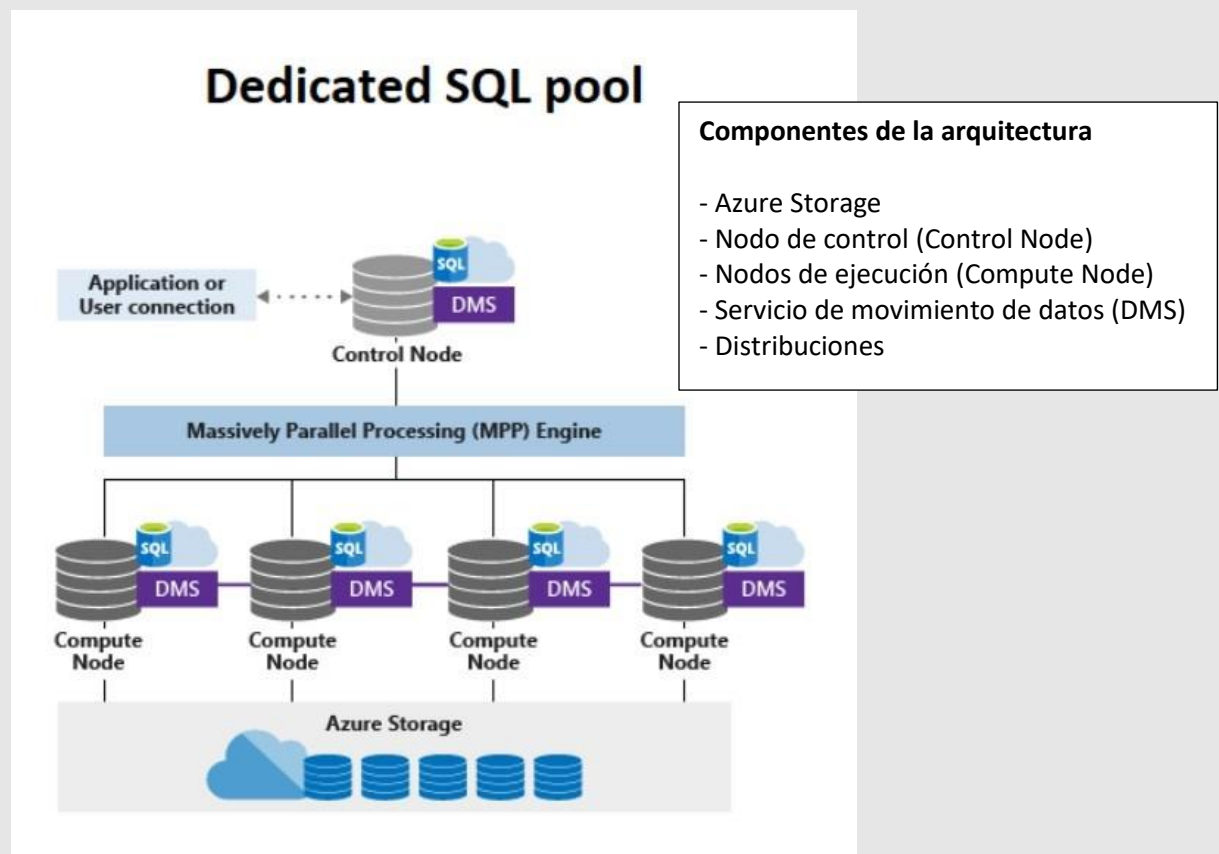


Azure Synapse MPP Architecture (Dedicated SQL Pool)

Una arquitectura MPP (massively parallel processing, por sus siglas en inglés) está diseñada para gestionar operaciones múltiples de manera simultánea mediante varias unidades de procesamiento.

En ese sentido, el Dedicated SQL Pool de Synapse utiliza una arquitectura MPP que está basada en nodos. Los usuarios y/o aplicaciones se conectan mediante un nodo de control, que sirve como único punto de entrada que optimiza las consultas para el procesamiento en paralelo y envía las operaciones a los nodos de ejecución para llevar a cabo el trabajo en paralelo.

Los nodos de ejecución almacenan todos los datos en Azure Storage y ejecutan las consultas en paralelo. El Servicio de movimiento de datos (DMS) es un servicio interno de nivel de sistema que mueve datos entre los nodos según sea necesario para ejecutar consultas en paralelo.

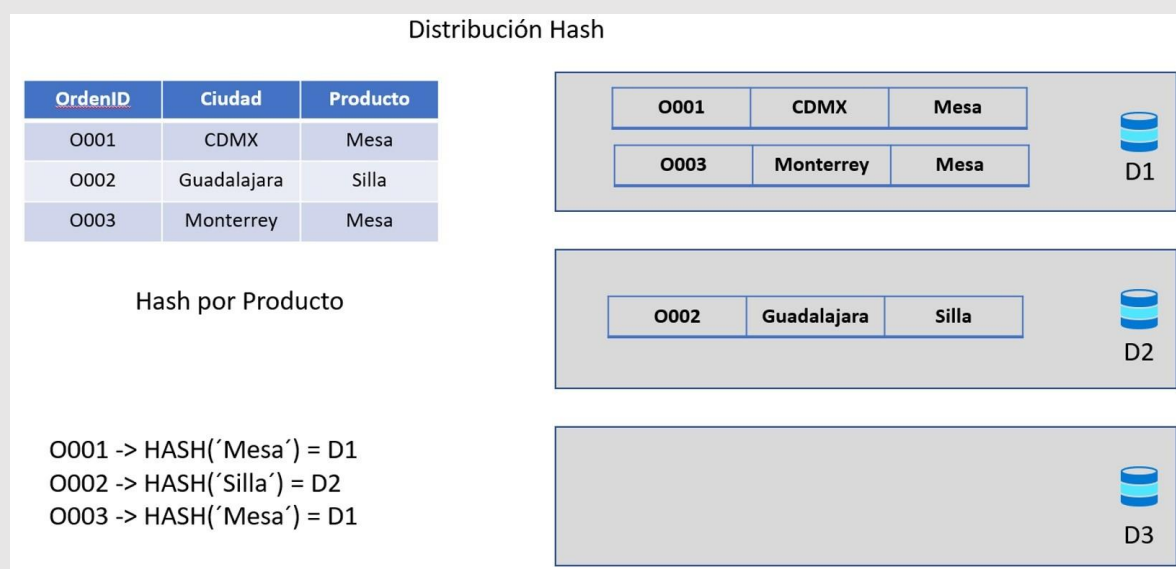


Un punto importante por notar aquí es el uso de las **distribuciones**, lo que significa que las filas de las tablas están distribuidas entre las diferentes distribuciones según el algoritmo elegido para dicho caso, como lo veremos a continuación:

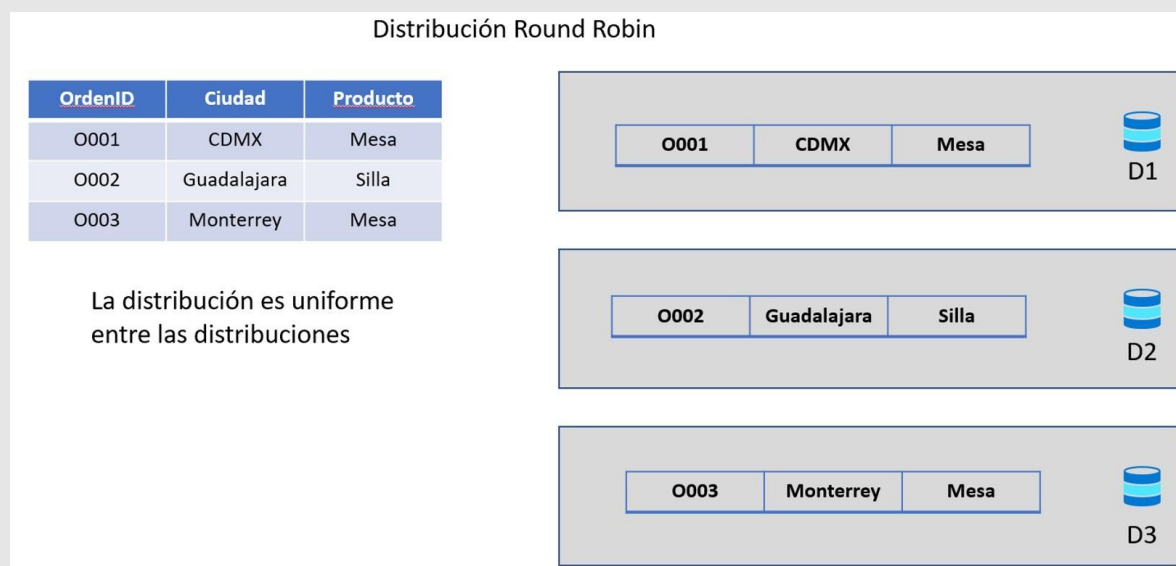
Distribuciones de tablas (Hash | Round robin | Replicadas)

Una tabla distribuida aparece como una sola tabla, pero las filas se almacenan realmente en 60 distribuciones. Las filas se distribuyen con un algoritmo **hash** o **round robin**.

Cuando hablamos de **hash**, la distribución se lleva a cabo de manera determinista, lo que significa que valores idénticos de la columna (de distribución), siempre irán a la misma distribución.



Por otro lado, **round robin** distribuye las filas de manera aleatoria, pero de manera uniforme entre todas las distribuciones.



NOTA: También existe la opción de usar [tablas replicadas](#), que se refiere a tener una copia completa de la tabla para que esté disponible en cada nodo de proceso. Esto se recomienda para tablas pequeñas.

NOTA: Las distribuciones se dividen equitativamente entre los **nodos de cómputo**, por ejemplo, si tenemos un solo nodo, las 60 distribuciones son manejadas por ese único nodo, pero si tenemos 3 nodos, entonces cada uno maneja 20 distribuciones y así sucesivamente según el nivel de desempeño elegido en la infraestructura dedicada del grupo de SQL ([ver detalle](#)).

El dedicated SQL pool divide cada tabla en 60 bases de datos distribuidas

DWU	# Nodos de cómputo	Distribución por nodo
100	1	60
200	2	30
300	3	20
400	4	15
500	5	12
600	6	10
1000	10	6
1200	12	5
1500	15	4
2000	20	3
3000	30	2
6000	60	1

Dependiendo del uso que se vaya a dar a la tabla, así como las características de esta, se puede tomar como guía la siguiente tabla para identificar qué método de distribución es el más conveniente:

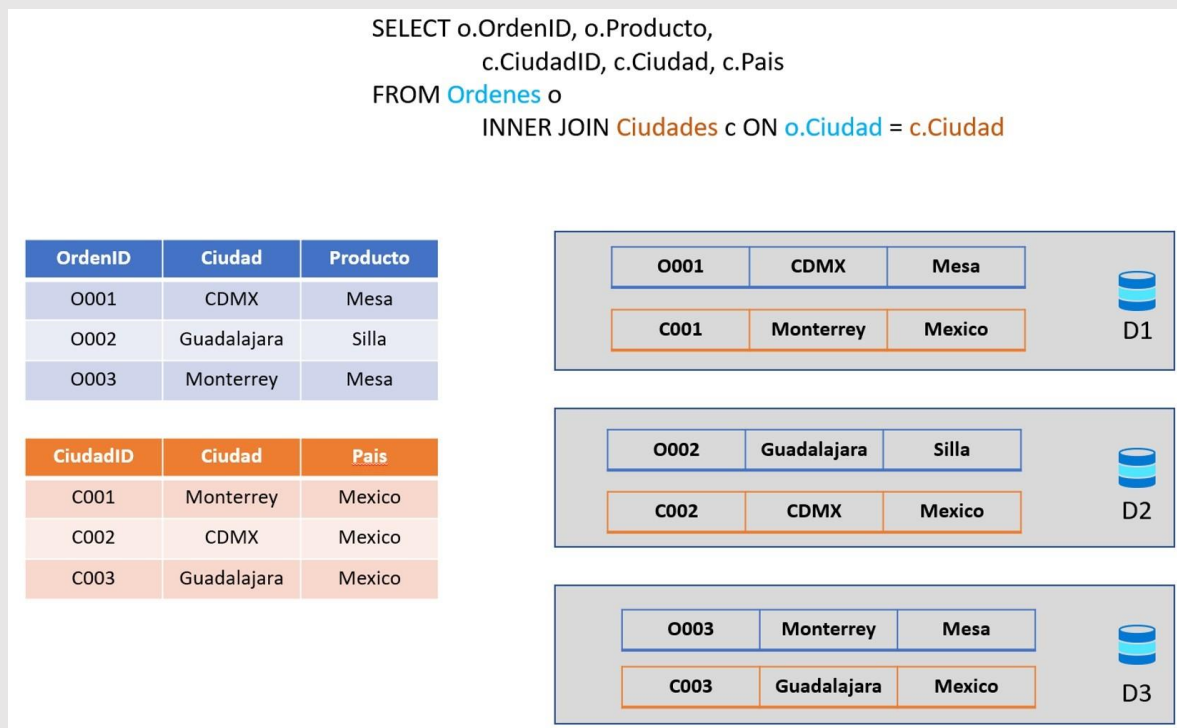
Métodos de distribución			
Nombre	Desempeño cargas	Desempeño consultas	Excelente para
Round-Robin	Rápido (distribución aleatoria)	Lento (Movimiento de datos)	Staging y tablas temporales
Hash	Lento (Hash de los valores)	Rápido (dependiendo de la distribución)	Tablas de hechos y Tablas grandes de dimensiones
Replicada	Lento (actualización en todos los nodos)	Rápido (elimina el movimiento)	Tablas pequeñas de dimensiones

¿Por qué es importante entender todo esto? Veamos, cuando el dedicated SQL pool ejecuta una consulta, el trabajo se divide en 60 consultas más pequeñas que se ejecutan en paralelo. Cada una de estas 60 consultas más pequeñas se ejecuta en una de las distribuciones de datos. Por lo tanto, existe la posibilidad de que los datos con los que estamos trabajando se encuentren en diferentes distribuciones y se requiera tener que moverlos (aquí entra en juego el DMS) para completar la consulta.

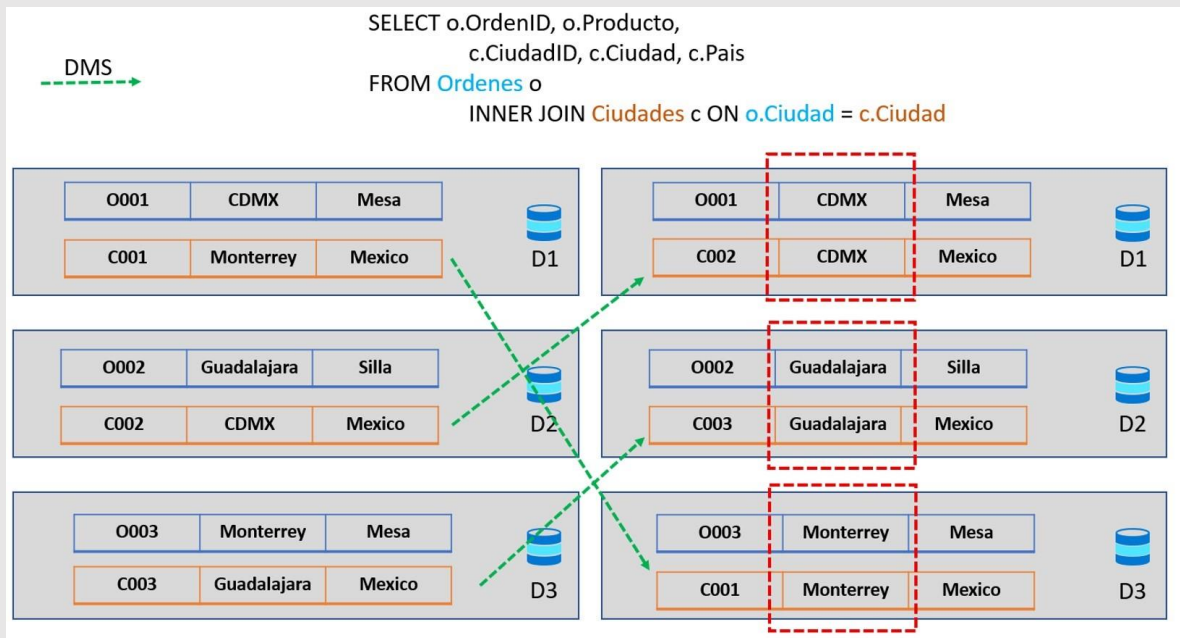
DMS

El servicio de movimiento de datos (DMS) es la tecnología de transporte de datos del Dedicated SQL Pool, que coordina el movimiento de los datos entre los nodos de proceso. Algunas consultas requieren el movimiento de datos para asegurarse de que las consultas paralelas devuelven resultados precisos.

Supongamos que además de la tabla de Ordenes, mostrada arriba, tenemos la tabla de Ciudades y queremos mostrar un cruce de información de ambas tablas. También supongamos que los datos están distribuidos como se ve en la siguiente imagen:



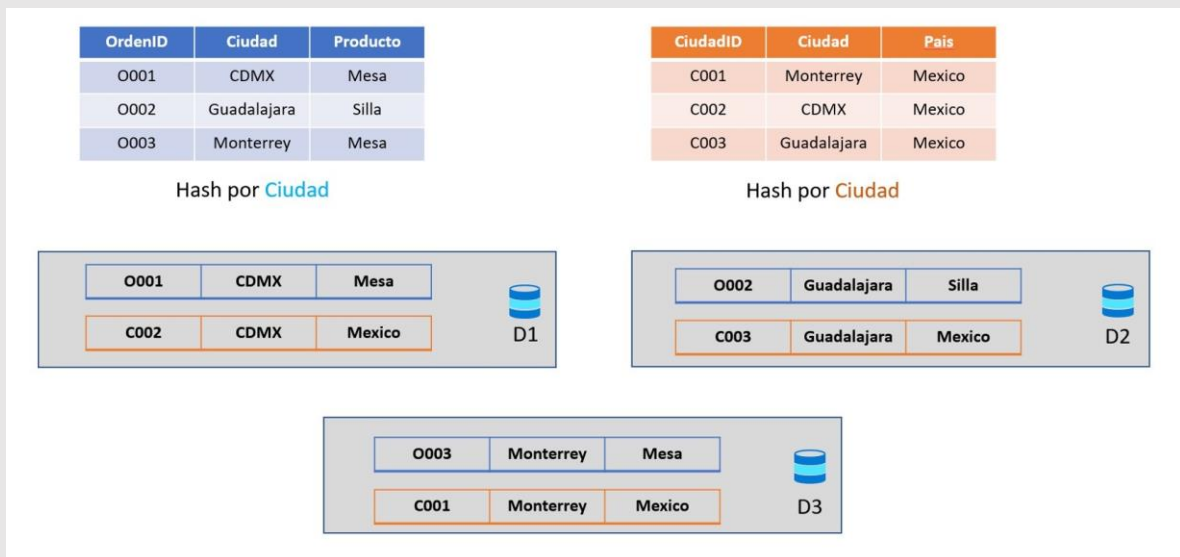
La consulta anterior implica mover los datos entre distribuciones para poder regresar los resultados (hacer el INNER JOIN). Dicho movimiento conlleva un gran uso de recursos, que impactan en el desempeño, sobre todo si estamos trabajando con tablas muy grandes.



¿Cómo resolverlo?

Un punto para tener en cuenta es la siguiente [guía de diseño](#) de tablas distribuidas, que nos ayudará a identificar las opciones que tenemos y las consideraciones que se deben tomar para evitar en lo posible el movimiento de datos.


Para el ejemplo anterior, veamos cómo al definir la columna de la distribución **hash** adecuada, se evita el movimiento de datos y, por consiguiente, se obtiene un mejor desempeño con los mismos recursos aprovisionados.



Este ejemplo es muy sencillo, pero es importante entender la idea de la distribución de datos y las implicaciones que tiene el movimiento de datos en el desempeño, una buena elección en cuanto a los métodos de distribución puede hacer toda la diferencia.


Otro ejemplo:

```
CREATE TABLE [dbo]. DimProducto
(
    ProductID INT NOT NULL,
    ProductName VARCHAR(50) NOT NULL,
    Precio DECIMAL(5,2) NOT NULL,
    CategoryID INT NOT NULL
)
WITH
(
    ÍNDICE DE ALMACÉN DE COLUMNAS AGRUPADO,
    DISTRIBUTION = ROUND_ROBIN
)
```



Y ahora con un algoritmo hash:

```
CREATE TABLE [dbo]. DimProducto
(
    ProductID INT NOT NULL,
    ProductName VARCHAR(50) NOT NULL,
    Precio DECIMAL(5,2) NOT NULL,
    CategoryID INT NOT NULL
)
WITH
(
    ÍNDICE DE ALMACÉN DE COLUMNAS AGRUPADO,
    DISTRIBUTION = HASH(CategoryID)
)
```



En esta parte, especificamos que el hash se toma de la columna CategoryID. Como todos los productos de la misma categoría se almacenarán en la misma distribución.

Entonces, ¿qué ganamos al asegurarnos de que los productos de las mismas categorías se almacenen en la misma distribución?

Si queremos obtener la suma del número de productos por categoría, ahora podemos hacerlo sin movimiento de datos porque estamos seguros de que las filas para una categoría determinada estarán todas en la misma distribución.

Además, si queremos adjuntar datos de otra tabla a CategoryID, esta unión puede ocurrir "localmente" si la otra tabla tiene una distribución hash en CategoryID. Tienes que pensar en el tipo de consultas que vas a tener y también asegurarte de que los datos se distribuirán de manera uniforme.

Se recomienda utilizar el control de distribución en columnas que no están actualizadas y distribuir los datos de manera uniforme, evitando el sesgo de datos para minimizar el movimiento de datos.

[Azure Synapse Analytics y procesamiento paralelo masivo \(linkedin.com\)](#)

[MPP & Distribution in Azure SQL Data warehouse - Artículos de TechNet - España \(Inglés\) - TechNet Wiki \(microsoft.com\)](#)