

Azure Data Engineer Associate Certification Guide

A hands-on reference guide to developing your data engineering skills and preparing for the DP-203 exam

Newton Alex



Capítulo 7: Implementación de la capa de servicio	4
7.1. Requisitos técnicos.....	4
7.2. Entrega de datos en un esquema relacional en estrella	5
7.3. Implementación de una jerarquía dimensional	9
7.3.1. Entrega de datos en archivos Parquet	9
Synapse SQL serverless	9
Synapse Spark.....	10
Azure Databricks	11
7.4. Mantener los metadatos.....	14
7.4.1. Metadatos usando Synapse SQL y Spark pools.....	14
7.4.2. Metadatos utilizando Azure Databricks	16
Resumen.....	24

Parte 2: Data Storage

Esta parte se sumerge en los detalles de los diferentes tipos de almacenamiento, las estrategias de partición de datos, los esquemas, los tipos de archivos, la alta disponibilidad, la redundancia, etc.

Esta sección comprende los siguientes capítulos:

- ❖ Capítulo 2, Diseño de una estructura de almacenamiento de datos
- ❖ Capítulo 3, Diseño de una estrategia de partición
- ❖ Capítulo 4, Diseño de la capa de servicio
- ❖ Capítulo 5, Implementación de estructuras físicas de almacenamiento de datos
- ❖ Capítulo 6, Implementación de estructuras lógicas de datos
- ❖ Capítulo 7, Implementación de la capa de servicio

Capítulo 7: Implementación de la capa de servicio

Espero que hayas disfrutado aprendiendo sobre los detalles de implementación de las estructuras lógicas de datos en el capítulo anterior. En este capítulo, aprenderemos sobre la implementación de la capa de servicio, que implica la implementación de esquemas en estrella, técnicas para leer y escribir diferentes formatos de datos, compartir datos entre servicios como SQL y Spark, y más. Una vez que hayas completado este capítulo, deberías ser capaz de entender las diferencias entre un Synapse dedicated SQL pool frente a los sistemas SQL tradicionales para implementar el esquema de estrella, las distintas formas de acceder a los datos de Parquet utilizando tecnologías como Spark y SQL, y los detalles involucrados en el almacenamiento de metadatos entre servicios. Todos estos conocimientos deberían ayudarte a construir una capa de servicio práctica y mantenible en un data lake y, por supuesto, también a obtener la certificación.

En este capítulo cubriremos los siguientes temas:

- Entrega de datos en un esquema relacional en estrella
- Implementación de una jerarquía dimensional
- Entrega de datos en archivos Parquet
- Mantener los metadatos

7.1. Requisitos técnicos

Para este capítulo, necesitará lo siguiente

Una cuenta de Azure (gratuita o de pago)

Un espacio de trabajo Synapse activo

Comencemos.

7.2. Entrega de datos en un esquema relacional en estrella

Hemos aprendido sobre el esquema de estrella en el Capítulo 4, Diseño de la capa de servicio. Aquí tomaremos el mismo ejemplo y mostraremos cómo implementar un esquema de estrella en Synapse SQL y entregar datos desde él.

Los esquemas en estrella tienen dos tipos de tablas, tablas de hechos y tablas dimensionales. Las tablas de hechos suelen tener un volumen mucho mayor que las tablas de dimensiones y, por lo tanto, se beneficiarían del uso de una distribución hash con indexación clustered columnstore. Por otro lado, las tablas dimensionales son más pequeñas y pueden beneficiarse del uso de tablas replicadas.

NOTA IMPORTANTE

Synapse dedicated SQL pools no soportan las restricciones de foreign key en el momento de escribir este libro. Por lo tanto, la responsabilidad de mantener la integridad de los datos recae en las aplicaciones.

Consideremos el mismo ejemplo de viajes en taxi de Imaginary Airport Cabs (IAC) para nuestro esquema estrella del Capítulo 4, Diseño de la capa de servicio. En ese diseño teníamos las siguientes tablas

- **FactTrips**
- **DimDriver**
- **DimCustomer**
- **DimCab**
- **DimDate**

Veamos cómo implementar estas tablas.

1. En la pantalla de Synapse, cree un nuevo pool SQL desde la pestaña Manage, como se muestra en la siguiente captura de pantalla. Haz clic en el símbolo +Nuevo y rellena los detalles para crear un nuevo dedicated SQL pool.

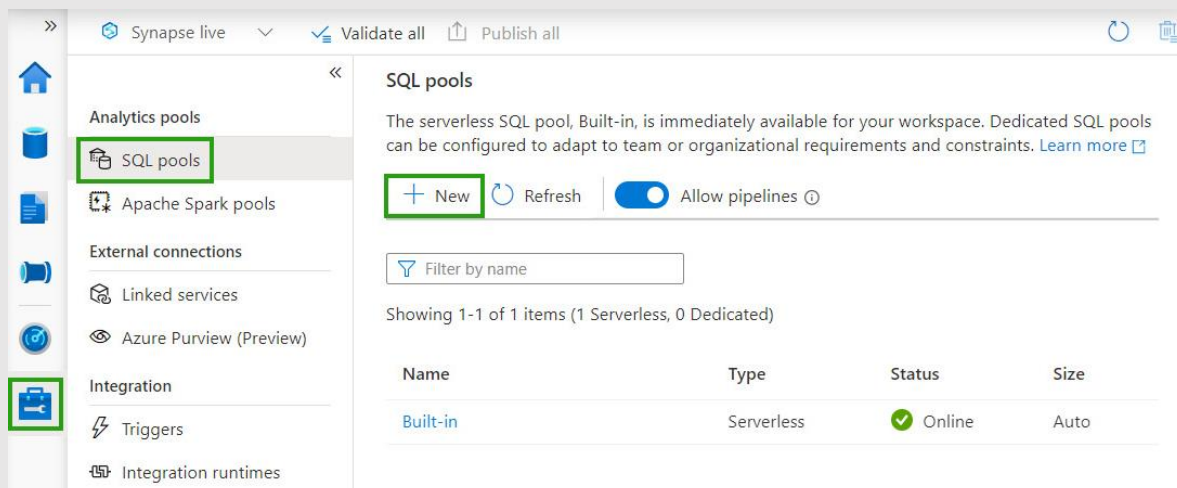


Figura 7.1 - Creación de un nuevo dedicated SQL Pool

A continuación, cree un nuevo script SQL desde la pestaña Editor haciendo clic en el signo +, como se muestra en la siguiente captura de pantalla:

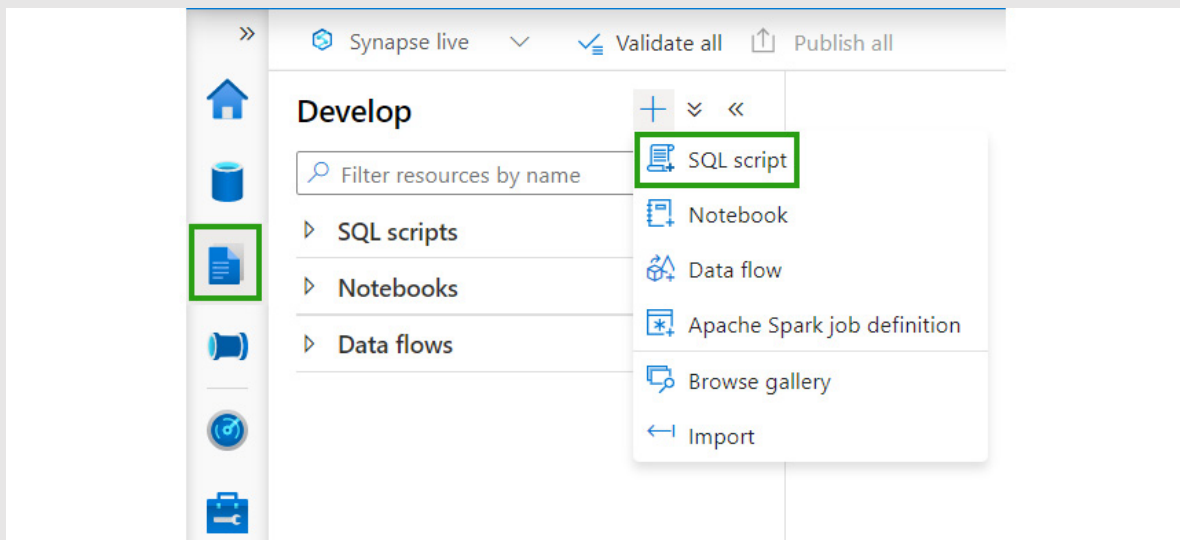


Figura 7.2 - Creación de un nuevo script SQL

En el editor SQL, puede introducir comandos SQL para crear tablas que representen el esquema de la estrella. A continuación se muestra un ejemplo de cómo crear una tabla de hechos de ejemplo, FactTrips:

```
CREATE TABLE dbo.FactTrips
(
    [tripId] INT NOT NULL,
    [driverId] INT NOT NULL,
    [customerId] INT NOT NULL,
    [tripdate] INT,
    [startLocation] VARCHAR(40),
    [endLocation] VARCHAR(40)
)
WITH
(
    CLUSTERED COLUMNSTORE INDEX,
    DISTRIBUTION = HASH ([tripId])
)
```

A continuación se muestra un ejemplo de tabla de dimensión de clientes:

```
CREATE TABLE dbo.DimCustomer
(
    [customerId] INT NOT NULL,
    [name] VARCHAR(40) NOT NULL,
    [emailId] VARCHAR(40),
    ...
    [city] VARCHAR(40)
```

```
)
WITH
(
    CLUSTERED COLUMNSTORE INDEX,
    DISTRIBUTION = REPLICATE
)
```

Este es un ejemplo de una tabla de dimensión de fecha:

```
CREATE TABLE dbo.DimDate
(
    [dateId] INT NOT NULL,
    [date] DATETIME NOT NULL,
    [dayOfWeek] VARCHAR(40),
    [fiscalQuarter] VARCHAR(40)
)
WITH
(
    CLUSTERED COLUMNSTORE INDEX,
    DISTRIBUTION = REPLICATE
)
```

Ahora, veamos cómo cargar los datos en estas tablas. Puede utilizar la sentencia COPY INTO para rellenar las tablas. Las tablas de hechos y dimensiones tendrán la misma sintaxis para cargar la información:

```
COPY INTO dbo.DimCustomer
FROM 'https://path/to/customer.csv'
WITH (
    FILE_TYPE='CSV',
    FIELDTERMINATOR=',',
    FIELDQUOTE='"',
    ROWTERMINATOR='\n',
    ENCODING = 'UTF8',
    FIRSTROW = 2 // To skip the header line
)
```

Y, por último, consulta desde las tablas del esquema estrella. He aquí una consulta de ejemplo para obtener la lista de todos los clientes cuya ubicación final era "San José":

```
SELECT trip.[tripId], customer.[name]
FROM dbo.FactTrips AS trip
JOIN dbo.DimCustomer AS customer
ON trip.[customerId] = customer.[customerId]
WHERE trip.[endLocation] = 'San Jose';
```

Como puedes ver, una vez que entendemos el concepto de esquemas en estrella, crearlos es tan sencillo como crear tablas en Synapse SQL.

Puedes aprender más sobre Synapse SQL y los esquemas aquí: <https://docs.microsoft.com/en-us/azure/synapse-analytics/sql-data-warehouse/sql-data-warehouse-tables-overview>.

Veamos ahora la implementación de una jerarquía dimensional.

7.3. Implementación de una jerarquía dimensional

Como ya hemos explorado la jerarquía dimensional en el Capítulo 4, Diseño de la capa de servicio, en la sección Diseño de una jerarquía dimensional, no lo repetiremos aquí. Por favor, echa un vistazo al Capítulo 4 para refrescar tu comprensión de la jerarquía dimensional.

Veamos ahora las técnicas disponibles para leer y escribir datos en archivos Parquet.

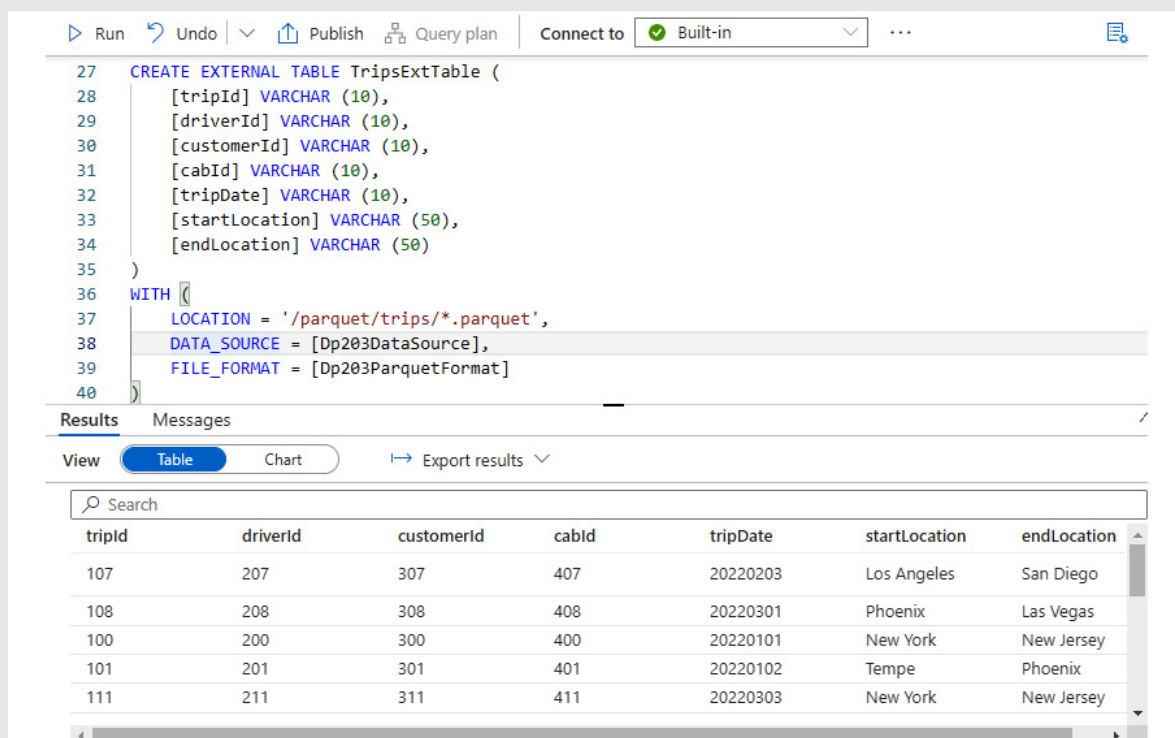
7.3.1. Entrega de datos en archivos Parquet

En esta sección, aprenderemos a entregar datos presentes en archivos Parquet tanto a través de Synapse SQL como de Spark. Para ello, utilizaremos el concepto de tablas externas que aprendimos en el capítulo anterior.

Consideremos un ejemplo en el que tenemos los datos de los trips almacenados como archivos Parquet. Utilizaremos Synapse SQL serverless pools y Spark para acceder a los datos y ejecutar algunas consultas de ejemplo sobre ellos.

Synapse SQL serverless

Vamos a crear una tabla externa desde el editor de consultas de Synapse SQL que apunte a los archivos Parquet en el storage de Azure. La siguiente captura de pantalla muestra un ejemplo:



The screenshot shows the Synapse SQL editor interface. The top toolbar includes buttons for Run, Undo, Publish, Query plan, and Connect to (set to Built-in). The main area contains SQL code for creating an external table named 'TripsExtTable' with columns: tripId, driverId, customerId, cabId, tripDate, startLocation, and endLocation. The table is defined with a LOCATION, DATA_SOURCE, and FILE_FORMAT. Below the code editor, the 'Results' tab is active, showing a table view of the data. The table has 7 columns and 5 rows of data.

tripid	driverid	customerid	cabid	tripDate	startLocation	endLocation
107	207	307	407	20220203	Los Angeles	San Diego
108	208	308	408	20220301	Phoenix	Las Vegas
100	200	300	400	20220101	New York	New Jersey
101	201	301	401	20220102	Tempe	Phoenix
111	211	311	411	20220303	New York	New Jersey

Figura 7.3 - Creación de SQL pool external tables

Ahora, vamos a ejecutar una consulta de ejemplo, como la forma en que una herramienta de BI accedería a estos datos. Intentemos encontrar el número de viajes por ubicación a partir de los datos cargados en el paso anterior y veámoslo como un gráfico:

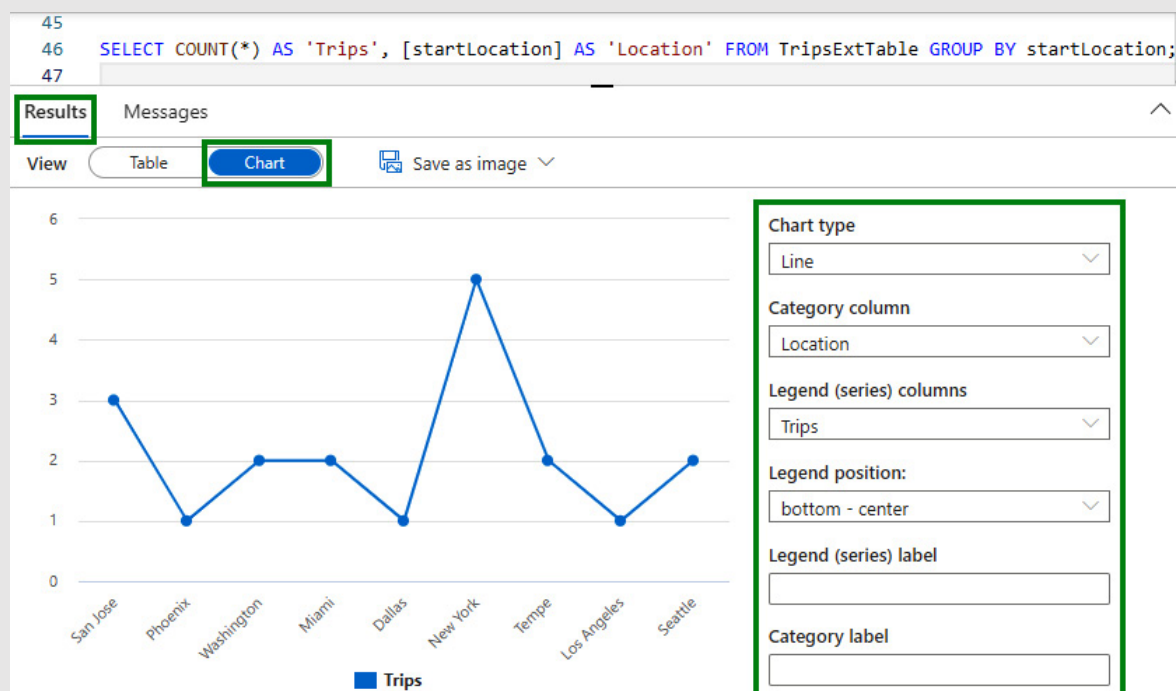


Figura 7.4 - Ejemplo de consulta sobre datos de Parquet visualizados en forma de gráfico

Para ver los gráficos, hay que seleccionar la opción Gráfico en la pantalla de Resultados. Synapse SQL proporciona varias opciones para configurar los gráficos, como se muestra en la parte derecha de la imagen anterior.

Como puede ver, leer y servir datos de archivos Parquet es tan sencillo como definir una tabla externa y consultarla.

Puede aprender más sobre el uso de tablas externas en Synapse SQL aquí: <https://docs.microsoft.com/en-us/azure/synapse-analytics/sql/develop-tables-external-tables>.

Ahora, vamos a intentar el mismo ejercicio usando Spark.

Synapse Spark

En un notebook de Synapse Spark, puede leer archivos Parquet utilizando el comando `spark.read.load('path/to/parquet/files', format='Parquet')`. La siguiente captura de pantalla muestra un ejemplo:

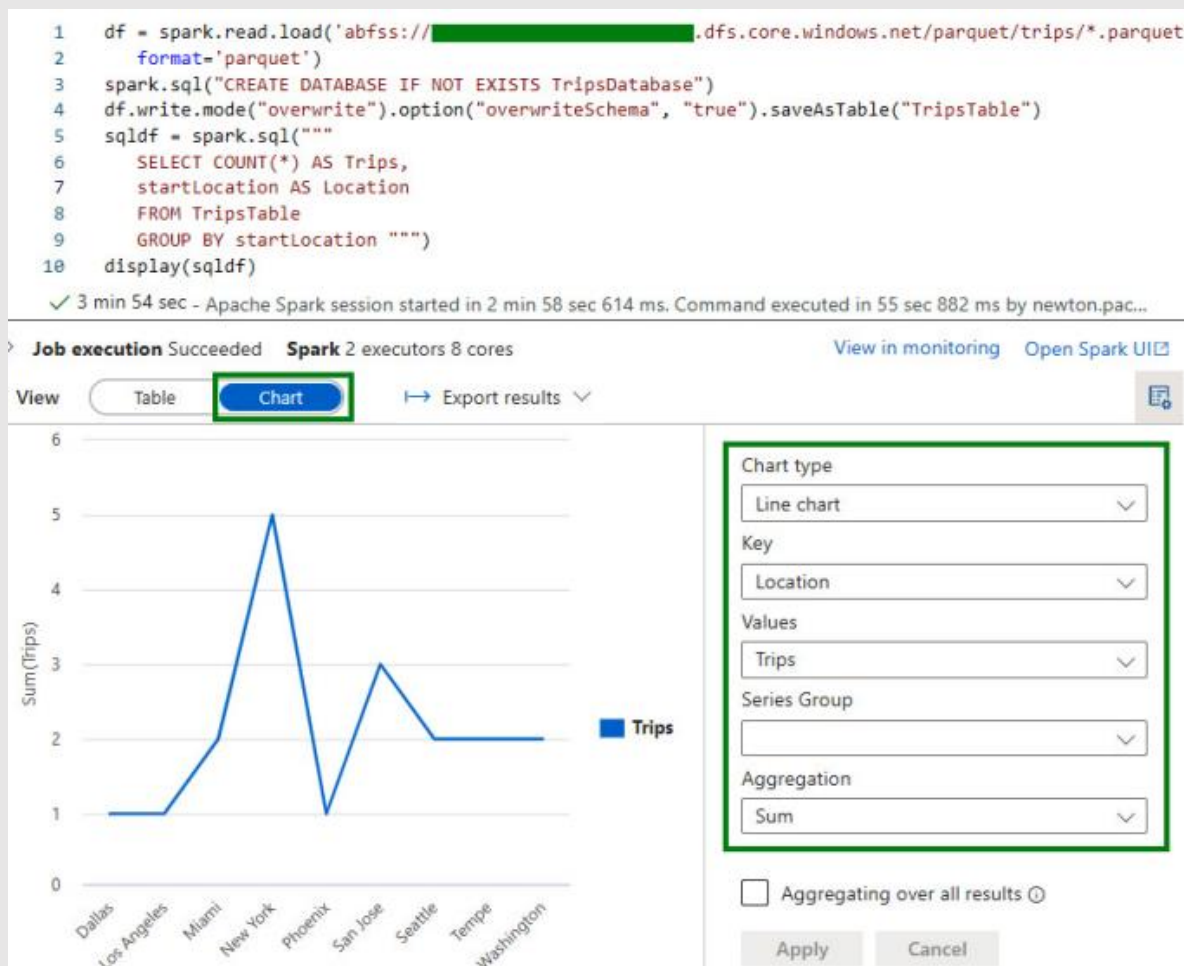


Figura 7.5 - Uso de Spark para consultar archivos Parquet

Spark también proporciona opciones de gráficos como se muestra en la imagen anterior.

A continuación vamos a explorar cómo podemos lograr lo mismo utilizando Azure Databricks.

Azure Databricks

Los ejemplos de Spark de Azure Databricks son similares a los de Synapse Spark. En el caso de Azure Databricks, tendrás que definir un service principal para que Azure Databricks se comunice con Azure Data Lake Storage Gen2.

Puedes encontrar información detallada sobre la configuración del service principal aquí: <https://docs.microsoft.com/en-us/azure/databricks/scenarios/databricks-extract-load-sql-data-warehouse>.

Una vez que tengas la sesión configurada, puedes utilizar exactamente la misma sintaxis de Spark que vimos en el caso de uso de Synapse Spark aquí también. A continuación se muestra un ejemplo para ver el contenido de todos los archivos Parquet en la carpeta parquet/trips:

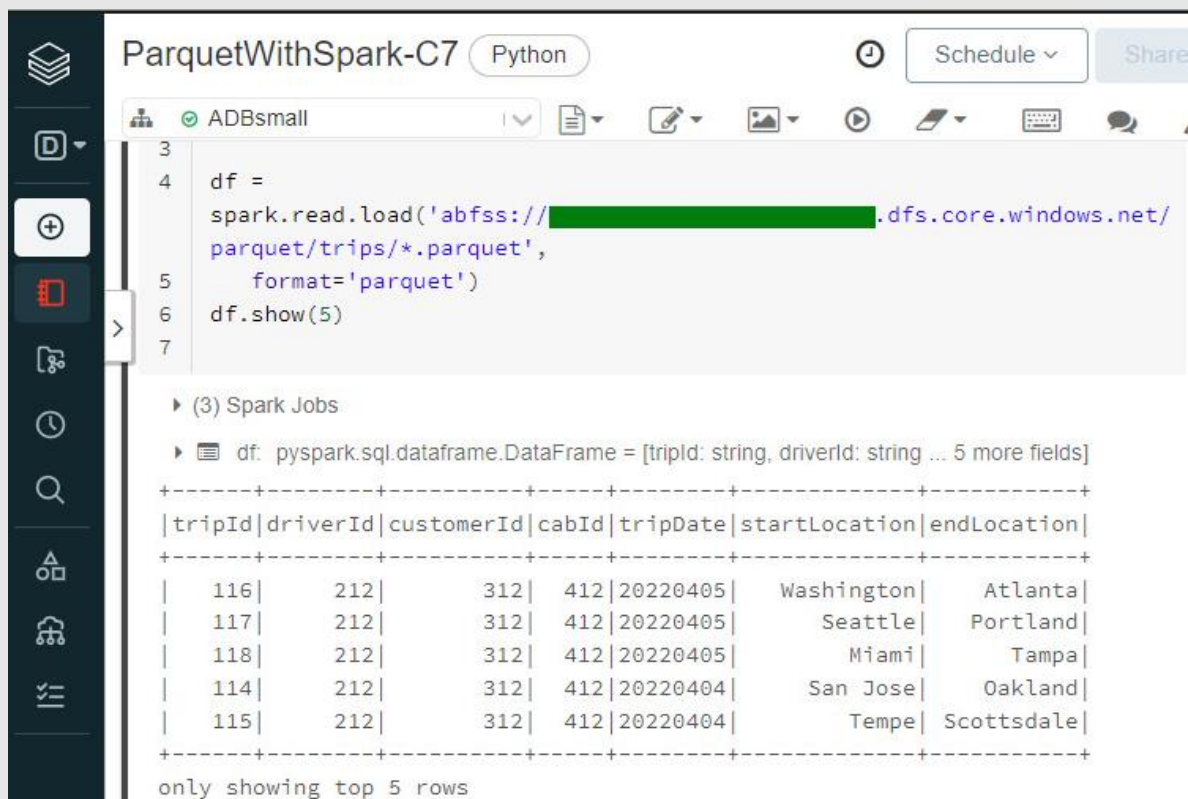


Figura 7.6 - Azure Databricks notebook con el ejemplo de acceso a Azure Data Lake Storage Gen2

A continuación se muestra un ejemplo de la misma consulta de transformación que ejecutamos anteriormente en Synapse Spark, en Databricks:

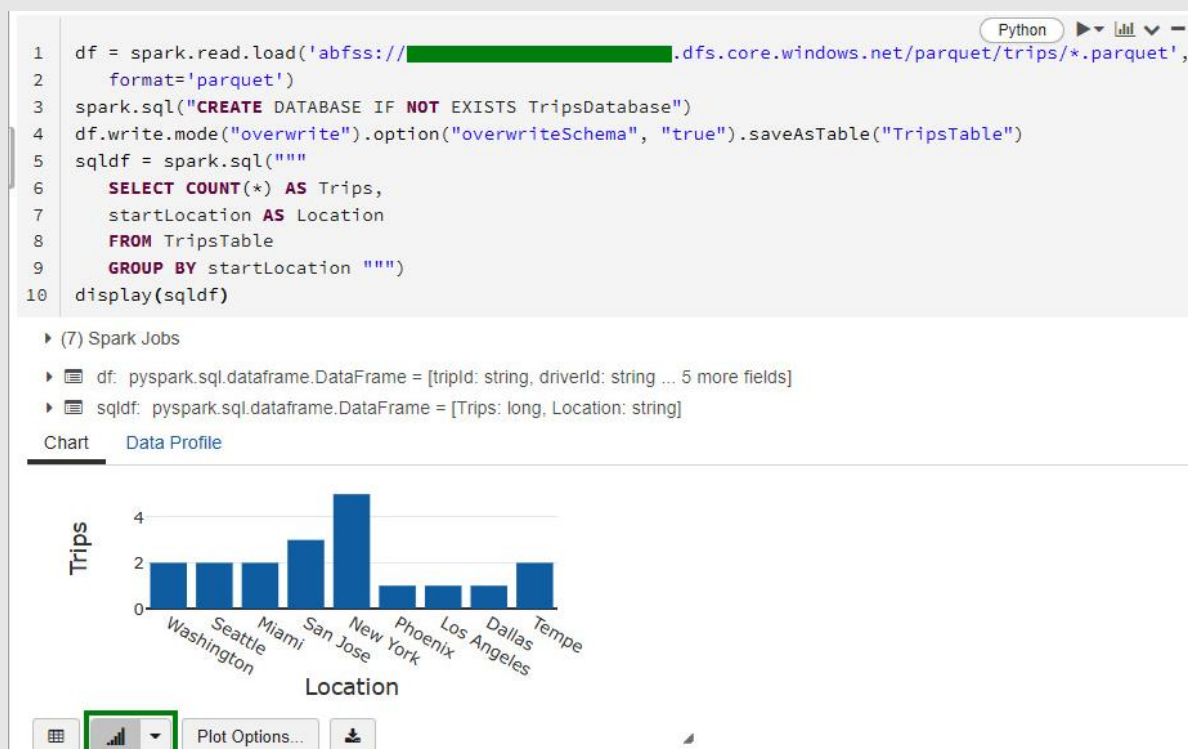


Figura 7.7 - Consulta simple de Spark sobre datos Parquet

Azure Databricks también proporciona un rico conjunto de opciones de gráficos que se pueden utilizar para visualizar los datos. Puedes acceder a las opciones de gráficos haciendo clic en el icono de gráfico situado debajo de los resultados.

Puedes aprender más sobre el uso de Spark y Parquet aquí: <https://spark.apache.org/docs/latest/sql-data-sources-parquet.html>.

Espero que ahora tengas una idea de cómo servir los datos de Parquet utilizando los servicios de SQL y Spark en Azure. Veamos ahora las opciones de metadatos disponibles en Azure.

7.4. Mantener los metadatos

Como hemos visto en el Capítulo 4, Diseño de una estrategia de partición, los metastores son como catálogos de datos que contienen información sobre todas las tablas que tienes, los esquemas de las tablas, las relaciones entre ellas, dónde están almacenadas, etc. En ese capítulo, aprendimos a un alto nivel sobre cómo acceder a los metadatos en Synapse y Databricks. Ahora, vamos a aprender los detalles de su implementación.

7.4.1. Metadatos usando Synapse SQL y Spark pools

Synapse soporta un modelo de metadatos compartido. Las bases de datos y las tablas que utilizan formatos de almacenamiento Parquet o CSV se comparten automáticamente entre los pools de cómputo, como SQL y Spark.

NOTA IMPORTANTE

Los datos creados desde Spark sólo pueden ser leídos y consultados por los pools de SQL pero no pueden ser modificados en el momento de escribir este libro.

Veamos un ejemplo de creación de una base de datos y una tabla utilizando Spark y accediendo a ella mediante SQL:

1. En el notebook de Synapse Spark, crea una tabla de ejemplo, como se muestra en la siguiente captura de pantalla:

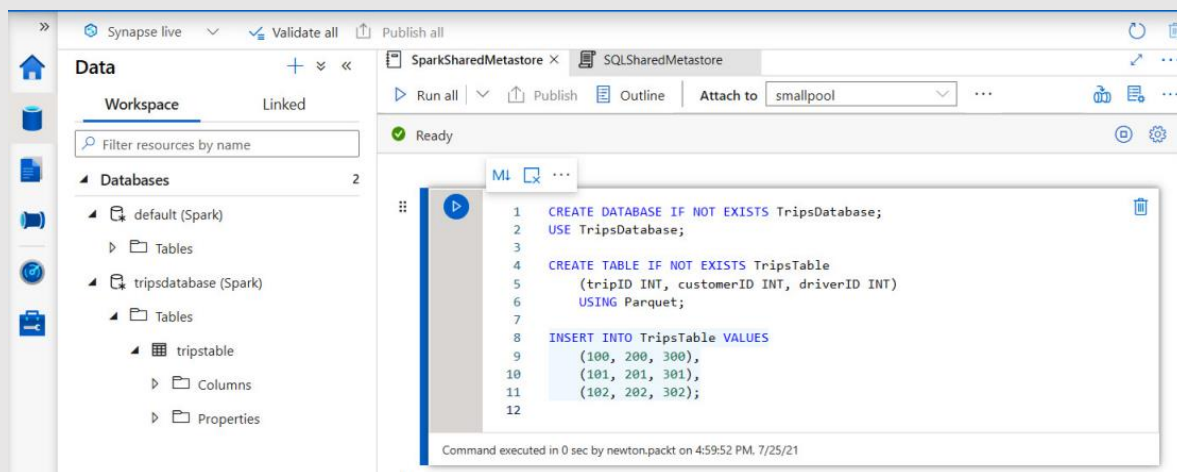


Figura 7.8 - Creación de una tabla de ejemplo en Spark

2. Ahora, vamos a consultar el contenido de la tabla desde el SQL serverless pool.

NOTA IMPORTANTE

Esta base de datos se sincronizará de forma asíncrona, por lo que puede haber un ligero retraso antes de ver las bases de datos y las tablas en el pool de SQL.

3. SQL serverless pool es un servicio bajo demanda, por lo que todo lo que tiene que hacer es hacer clic en el signo + en la página del área de trabajo de Synapse y seleccionar SQL script para crear un nuevo editor de SQL, como se muestra en la siguiente captura de pantalla:

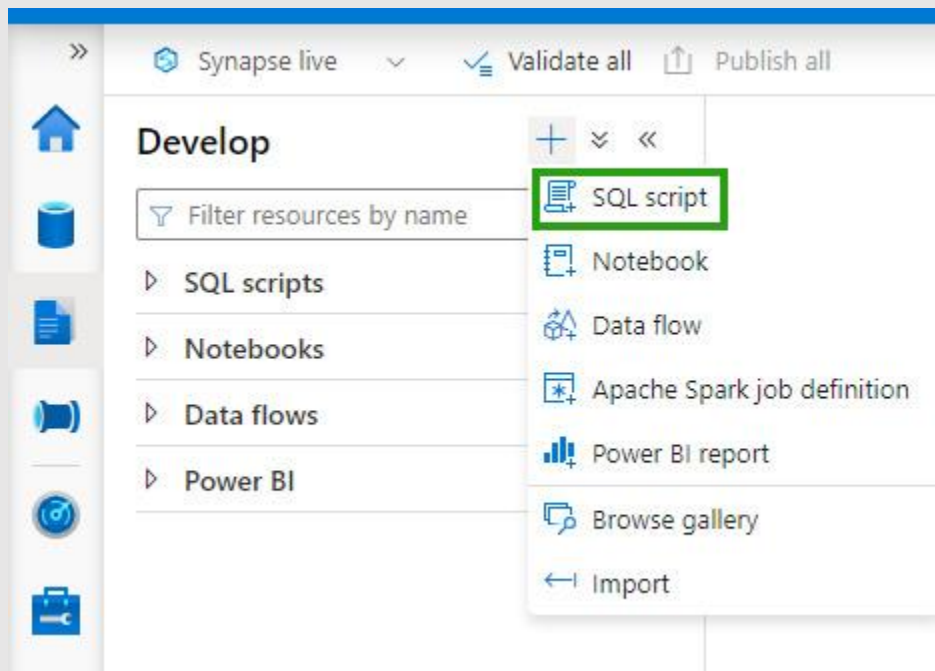


Figura 7.9 - Creación de un nuevo script SQL

4. A continuación, en el campo Connect to (Conectar a), seleccione Built-in pool (Grupo incorporado), como se muestra en la siguiente captura de pantalla:

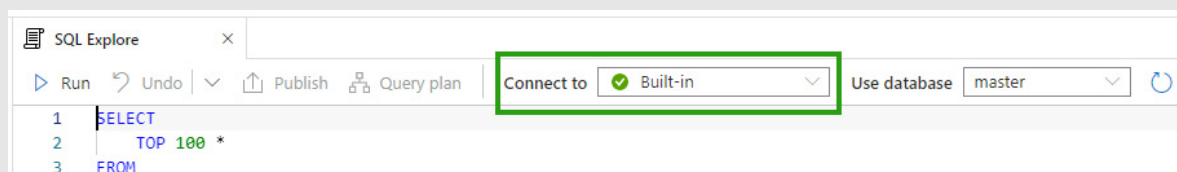


Figura 7.10 - Conectando al serverless SQL pool

5. Ahora, basta con ejecutar un sencillo script para consultar la tabla compartida; en el ejemplo, la tabla compartida sería la tabla tripID:

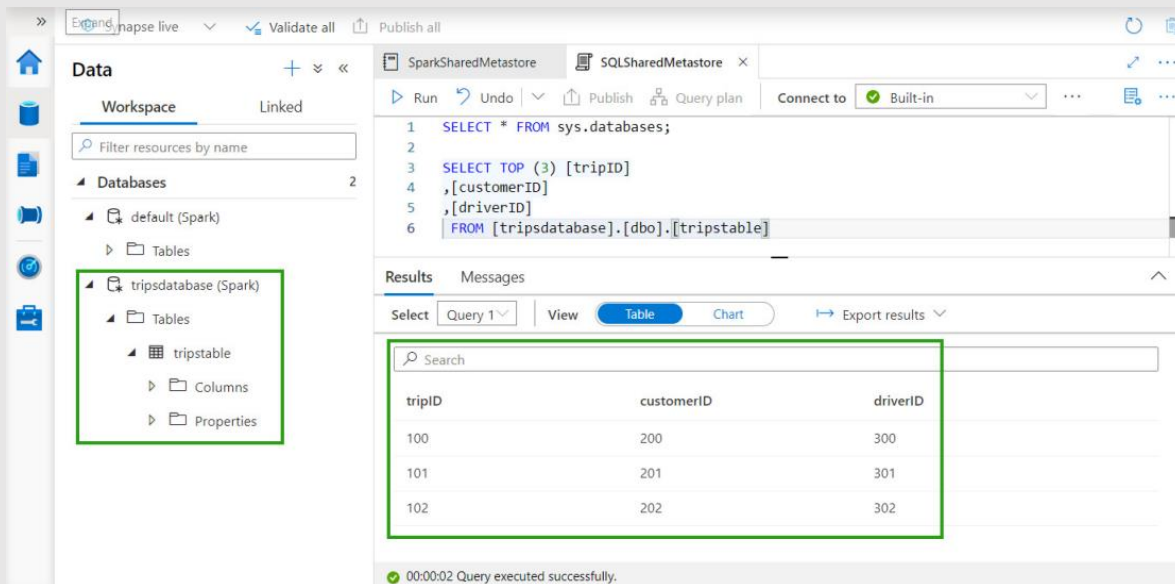


Figura 7.11 - Acceso a la tabla Spark en SQL

Como acaba de observar, el modelo de datos compartidos de Synapse hace que sea muy fácil compartir datos entre los pools de SQL y Spark. Synapse ya se encarga de todo y los datos están a nuestra disposición.

Puedes aprender más sobre cómo mantener los metadatos en Synapse aquí: <https://docs.microsoft.com/en-us/azure/synapse-analytics/metadata/overview>.

A continuación vamos a explorar cómo trabajar con los metastores en Azure Databricks.

7.4.2. Metadatos utilizando Azure Databricks

Para compartir datos entre Spark y otros servicios fuera de Synapse, tenemos que hacer uso del metastore Hive. Spark utiliza el metastore Hive para compartir información con otros servicios. Veamos un ejemplo de compartir datos entre Azure Databricks Spark y Azure HDInsight Hive. La lógica y los pasos para utilizar un metastore Hive externo serían similares para Synapse Spark también. Estos son los pasos:

1. Necesitaremos una base de datos independiente que pueda ser utilizada por Spark para almacenar los metadatos. Vamos a utilizar Azure SQL para este propósito. Crea una base de datos Azure SQL desde el portal de Azure. Busca Azure SQL desde el cuadro de búsqueda del portal y selecciónala. Haz clic en la opción + Crear. Verás una pantalla, como la que se muestra en la siguiente captura:

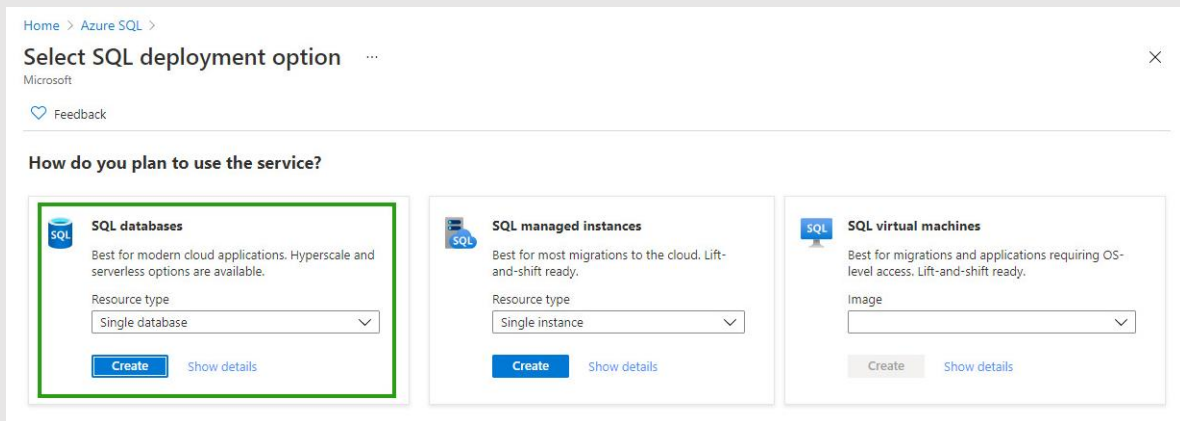


Figura 7.12 - Planes de Azure SQL

2. Selecciona la casilla de bases de datos SQL y haz clic en la opción Base de datos única del desplegable.
3. Se puede crear la base de datos con datos de muestra prepopulados, de forma que tengamos datos listos para la experimentación. Seleccione la opción Muestra para el campo Usar datos existentes, como se muestra en la siguiente captura de pantalla:

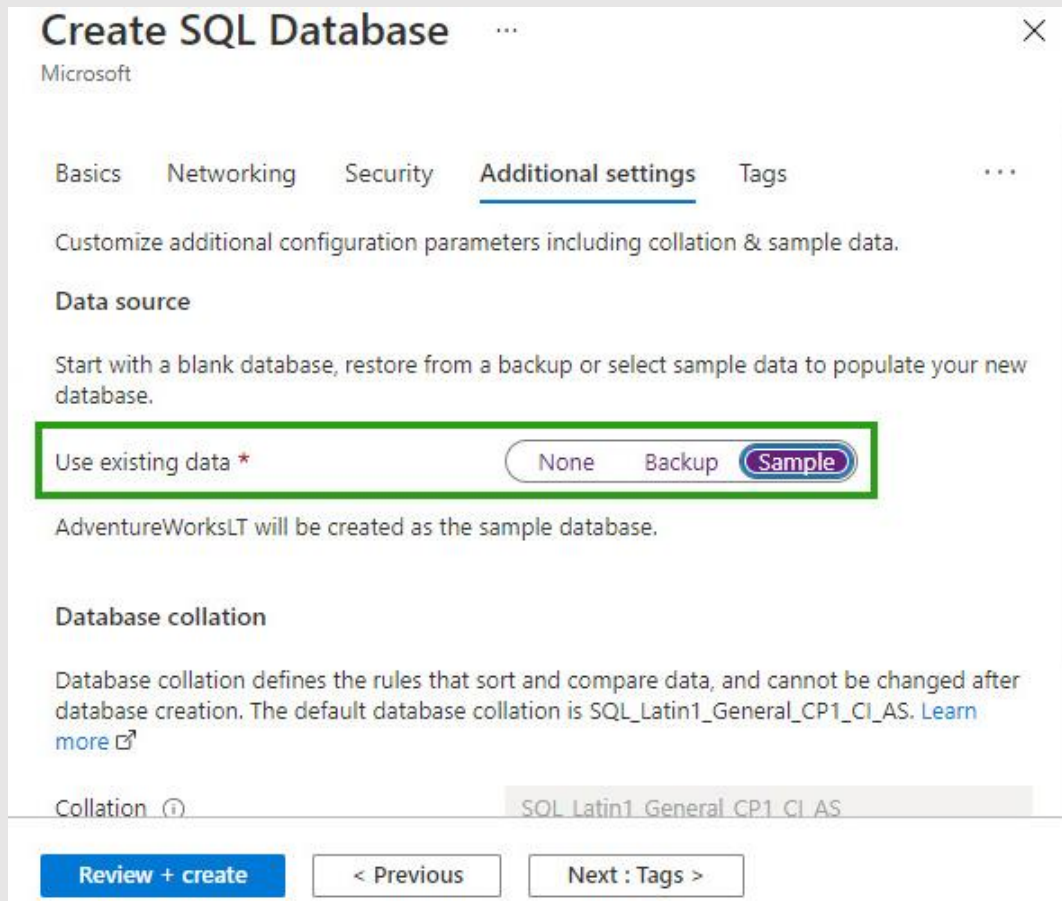


Figura 7.13 - Rellenar la base de datos con datos de ejemplo

4. Rellena el resto de las pestañas y haz clic en Revisar + crear para crear la base de datos Azure SQL.
5. Recupera la cadena de conexión JDBC de la pestaña Connection Strings, como se muestra en la siguiente captura de pantalla, y guárdala en el Bloc de notas. Necesitaremos esta información más adelante:

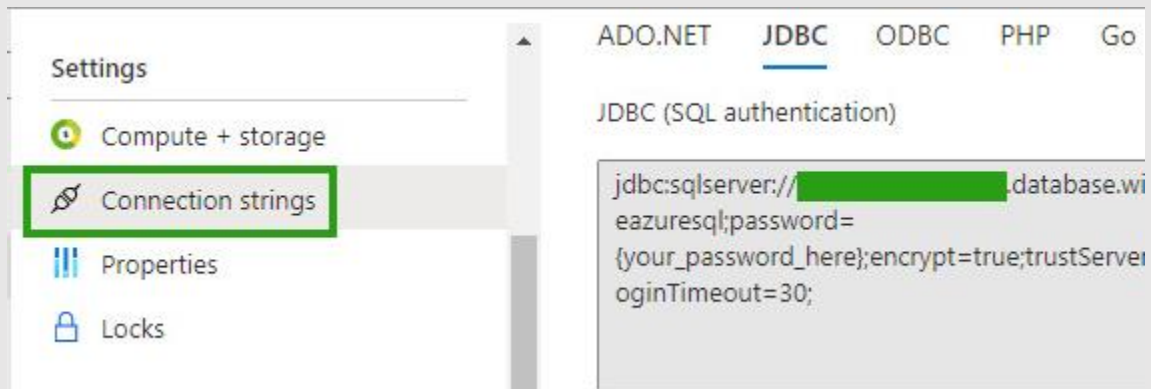


Figura 7.14 - Obtención de las cadenas de conexión para Azure SQL

6. A continuación, tenemos que crear un clúster HDInsight Hive. A estas alturas, es posible que ya conozcas el proceso para instanciar cualquier servicio de Azure. Sólo tienes que buscar HDInsight en la barra de búsqueda del portal y hacer clic en él. En la página de inicio del portal de HDInsight, haz clic en el enlace + Crear para abrir el formulario de creación y rellena los datos.
7. En la pantalla de creación del clúster de HDInsight, puede elegir la opción Hadoop o la opción Interactive Query para el tipo de clúster, ya que ambas instalarán Hive. Consulte la siguiente captura de pantalla para ver las opciones disponibles:

Create HDInsight cluster

Basics | Storage | Security + networking

New to HDInsight? Get started with our [training](#)
Create a managed HDInsight cluster. Select from

Project details
Select the subscription to manage deployed resources.

Subscription *
Resource group * [Create new](#)

Cluster details
Name your cluster, pick a region, and choose a cluster type.

Cluster name *
Region *
Cluster type *
Version

Cluster credentials
Enter new credentials that will be used to administer the cluster.

Cluster login username *

[Review + create](#) [Previous](#)

Select cluster type

Hadoop
Petabyte-scale processing with Hadoop components like MapReduce, Hive (SQL on Hadoop), Pig, Sqoop and Oozie. [Select](#)

Spark
Fast data analytics and cluster computing using in-memory processing. [Select](#)

Kafka
Build a high throughput, low-latency, real-time streaming platform using a fast, scalable, durable, and fault-tolerant publish-subscribe messaging system. [Select](#)

HBase
Fast and scalable NoSQL database. Available with both standard and premium (SSD) storage options. [Select](#)

Interactive Query
Build Enterprise Data Warehouse with in-memory analytics using Hive (SQL on Hadoop) and LLAP (Low Latency Analytical Processing). Note that this feature requires high memory instances. [Select](#)

Storm
Reliably process infinite streams of data in real-time. [Select](#)

ML Services (R Server)
Analyze data at scale, build intelligent apps and discover valuable insights across your business using both R and Python. [Select](#)
Adds 1.15272416 INR per Core-Hour.

Figura 7.15 - Selección del tipo de clúster para los clústeres de HDInsight

8. Una vez seleccionado el tipo de clúster, rellena el resto de campos de la pantalla Basics.
9. En la pestaña Storage, en la sección External metadata stores, proporciona la base de datos Azure SQL que creamos en los pasos anteriores como base de datos SQL para Hive. La siguiente captura de pantalla muestra la ubicación:

Home > HDInsight clusters >

Create HDInsight cluster

Additional Azure Storage
Link additional Azure Storage accounts to the cluster.
[Add Azure Storage](#)

Custom Ambari DB
Use an external Ambari database for greater flexibility, control, and customization. [Learn More](#)

SQL database for Ambari ⓘ

External metadata stores
To store your Hive and Oozie metadata outside of this cluster, select a SQL database. [Learn More](#)

SQL database for Hive ⓘ

SQL database for Oozie ⓘ

[Review + create](#) [« Previous](#) [Next: Security + networking »](#)

Figura 7.16 - Creación de un cluster Hive en HDInsight con Azure SQL como metastore

10. Completa el resto de los campos y luego haz clic en Review + Create para crear el cluster de HDInsight.
11. Una vez creado el clúster, ve a la página de inicio de Ambari desde el portal de HDInsight haciendo clic en el enlace Ambari home, como se muestra en la siguiente captura de pantalla:

Overview [Activity log](#) [Access control \(IAM\)](#) [Tags](#) [Diagnose and solve problems](#)

Settings

[Cluster size](#) [Quota limits](#)

URL : <https://testhdicluster.azurehdinsight.net>

Cluster ID : dd7a2a05a31b4a2ebbf29182b391b89f

Tags ([change](#)) : [Click here to add tags](#)

Overview [Get started](#)

Dashboards

[Ambari home](#) [Ambari views](#)

Figura 7.17 - Enlace a la página de inicio de Ambari desde la página de inicio del portal HDInsight

12. Desde el panel de control de Ambari, haz clic en Hive view 2.0, como se muestra en la siguiente captura de pantalla:

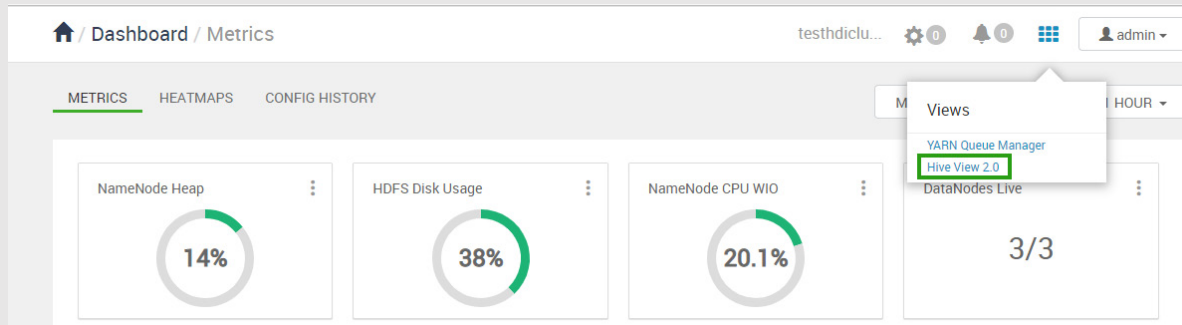


Figura 7.18 - Vista Hive 2.0 desde el panel de control de Ambari

13. Ahora, deberías poder ver la base de datos Hivesampletable en el dashboard de Hive, como se muestra en la siguiente captura de pantalla:

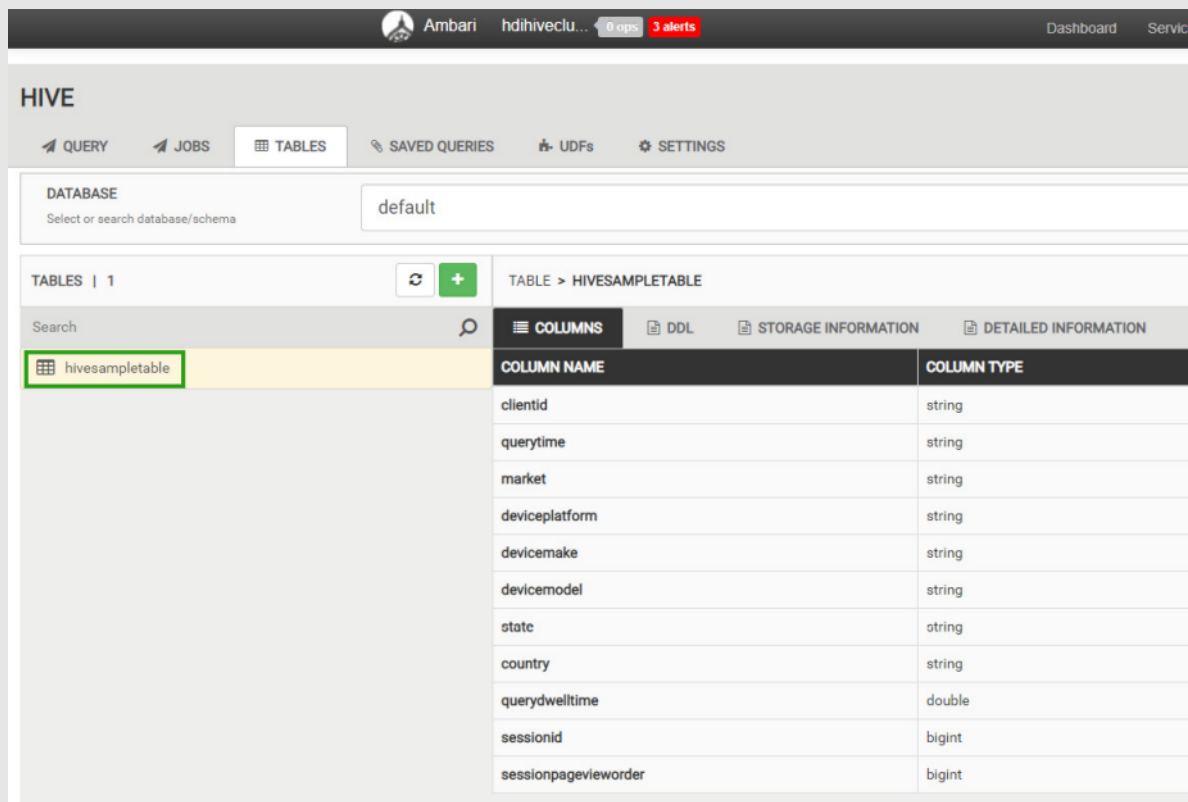


Figura 7.19 - Vista de la consulta Hive de la tabla de ejemplo

14. Ahora que tenemos el cluster de HDInsight, tenemos que crear un cluster de Azure Databricks. Tenemos que crear un nuevo cluster con las siguientes configuraciones. Vamos a ver cómo introducir estas configuraciones en la pantalla de creación de Spark en el siguiente paso:

```
spark.sql.hive.metastore.version 2.1.1 // For HDInsight Interactive 2.1 version
spark.hadoop.javax.jdo.option.ConnectionUserName <Your Azure SQL Database Username>
spark.hadoop.javax.jdo.option.ConnectionURL <Your Azure SQL Database JDBC connection string>
spark.hadoop.javax.jdo.option.ConnectionPassword <Your Azure SQL Database Password>
spark.hadoop.javax.jdo.option.ConnectionDriverName
com.microsoft.sqlserver.jdbc.SQLServerDriver
spark.sql.hive.metastore.jars <Location where you have copied the Hive Metastore Jars>
datanucleus.autoCreateSchema true
datanucleus.fixedDatastore false
```

Ten en cuenta que tendrás que usar el enlace JDBC que habías guardado antes, para la configuración que dice spark.hadoop.javax.jdo.option.ConnectionURL.

15. Tendrás que introducir las configuraciones en el campo Spark Config de la página Create Cluster, como se muestra en la siguiente captura de pantalla:

Create Cluster

Cluster Mode: Standard

Databricks Runtime Version: Runtime: 8.3 (Scala 2.12, Spark 3.1.1)

Note: Databricks Runtime 8.x and later use Delta Lake as the default table format.

Autopilot Options

- ☒ Enable autoscaling
- ☒ Terminate after 120 minutes of inactivity

Worker Type: Standard_DS3_v2 (14 GB Memory, 4 Cores) | Min Workers: 2 | Max Workers: 8 | ☐ Spot instances

Driver Type: Same as worker (14 GB Memory, 4 Cores)

DBU / hour: 2.25 - 6.75

Advanced Options

Azure Data Lake Storage Credential Passthrough: ☐ Enable credential passthrough for user-level data access

Spark | Tags | Logging | Init Scripts

Spark Config

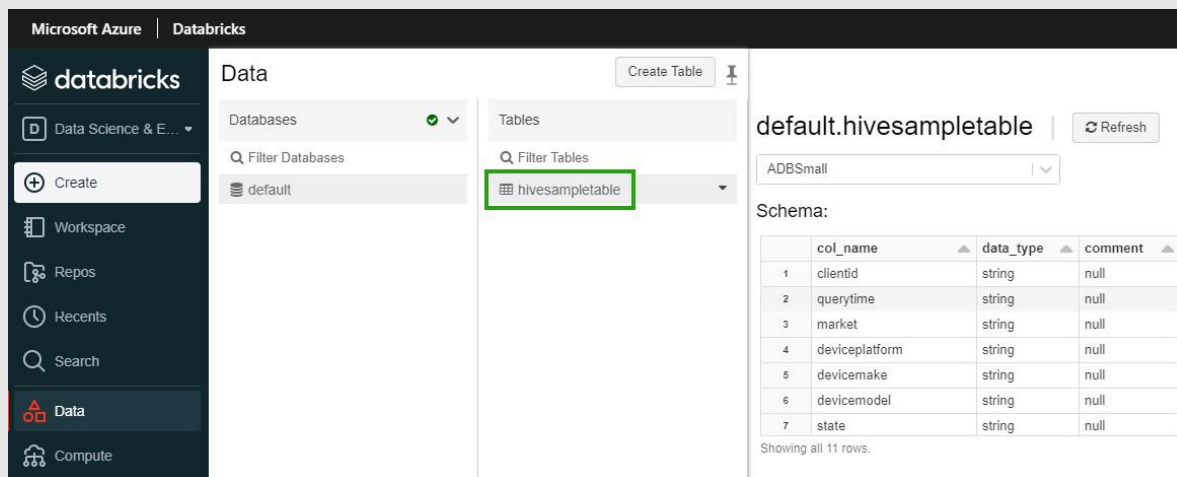
```
spark.sql.hive.metastore.version 2.1.1 // For HDInsight Interactive 2.1 version
spark.hadoop.javax.jdo.option.ConnectionUserName <Your Azure SQL Database Username>
spark.hadoop.javax.jdo.option.ConnectionURL <Your Azure SQL Database JDBC connection string>
spark.hadoop.javax.jdo.option.ConnectionPassword <Your Azure SQL Database Password>
spark.hadoop.javax.jdo.option.ConnectionDriverName
com.microsoft.sqlserver.jdbc.SQLServerDriver
```

Figura 7.20 - Adición de la configuración personalizada durante la creación del clúster Azure Databricks

NOTA IMPORTANTE

Aparte de los campos de configuración, también tendrás que descargar los archivos JAR del metastore de Hive y proporcionarles una ubicación donde los clusters de Azure Databricks puedan acceder a ellos. Azure proporciona instrucciones paso a paso junto con scripts listos para descargar fácilmente los archivos JAR aquí: https://docs.microsoft.com/en-us/azure/databricks/_static/notebooks/setup-metastore-jars.html.

Una vez creado el cluster de Spark, podrás acceder a las tablas de Hive Metastore directamente desde un notebook de Spark. En la siguiente captura de pantalla, puedes ver cómo el cluster Spark de Databricks es capaz de acceder a la tabla HiveSampletable que vimos anteriormente utilizando la vista de consulta Hive:



The screenshot displays the Microsoft Azure Databricks web interface. On the left is a dark sidebar with navigation options: 'Data Science & E...', 'Create', 'Workspace', 'Repos', 'Recents', 'Search', 'Data' (highlighted), and 'Compute'. The main area is titled 'Data' and contains two panels: 'Databases' and 'Tables'. Under 'Databases', 'default' is selected. Under 'Tables', 'hivesampletable' is selected and highlighted with a green rectangular box. To the right of these panels, the details for 'default.hivesampletable' are shown, including a 'Refresh' button, a dropdown for 'ADBSmall', and a 'Schema:' section. The schema is presented as a table with columns 'col_name', 'data_type', and 'comment'.

	col_name	data_type	comment
1	clientid	string	null
2	querytime	string	null
3	market	string	null
4	deviceplatform	string	null
5	devicemake	string	null
6	devicemodel	string	null
7	state	string	null

Showing all 11 rows.

Figura 7.21 - Spark es capaz de acceder a los datos del metastore externo de Hive

¡Viva! Ahora ya sabes cómo acceder a los metadatos entre los clusters de Spark y Hive utilizando un metastore externo de Hive.

Puedes saber más sobre los metastores de Azure Databricks aquí: <https://docs.microsoft.com/en-us/azure/databricks/data/metastores/external-hive-metastore>. Con esto, hemos llegado al final de esta sección y del capítulo. Ahora deberías estar familiarizado con las diferentes formas en las que se pueden compartir los metadatos entre los servicios SQL, Spark y Hive en Azure.

Resumen

Este fue otro pequeño pero interesante capítulo. Comenzamos con la implementación del esquema de estrella, luego aprendimos sobre la entrega de datos en formato Parquet y, finalmente, vimos cómo podemos compartir datos entre los servicios SQL-Spark y Spark-Hive en Azure. Con todos estos conocimientos, ahora deberías ser capaz de diseñar e implementar una capa de servicio básica para una arquitectura de data lake utilizando los servicios de Azure. Para saber más, sigue los enlaces que he proporcionado al final de los temas importantes.

Con esto, hemos llegado al final de la primera sección importante del programa de estudios DP-203, Diseño e Implementación de Almacenamiento de Datos. Esto cubre alrededor del 40-45% del examen de certificación. Nos acercamos a la mitad del camino. ¡Buen trabajo!

En la próxima sección, aprenderemos a diseñar y desarrollar sistemas de procesamiento de datos y, más concretamente, a ingerir y transformar datos en data lakes.