Managed vs External Table

February 4, 2025

1 Managed vs External Tables in Spark

This notebook explains **Managed vs External Tables in Spark** using an example dataset (`customers_1mb.csv`). It also demonstrates how to configure the **default warehouse path** and verify tables using **Hive Metastore**.

1.1 Key Concepts

- **Managed Table**: Spark fully controls the **data and metadata**.
- **External Table**: Spark manages only **metadata**, while the data remains **outside** Spark's control.
- **Hive Metastore**: Stores **table definitions** to enable SQL-like querying in Spark.

---

[1]:
```
#  Step 1: Import necessary libraries
from pyspark.sql import SparkSession


#  Step 3: Create Spark session with Hive support
spark = SparkSession.builder \
    .appName('ManagedVsExternalTables') \
    .enableHiveSupport() \
    .getOrCreate()

# print(f"Spark session created with warehouse location: {warehouse_location}")
```

```
25/02/02 03:33:03 WARN SparkSession: Using an existing Spark session; only
runtime SQL configurations will take effect.
```

1.1.1 Checking Current Warehouse Directory

You can verify the current **Spark SQL warehouse directory** using the command below.

[2]:
```
# Show the configured warehouse directory
spark.conf.get('spark.sql.warehouse.dir')
```

[2]:
```
'file:/spark-warehouse'
```

[ ]:
```

```

1.2 Loading the Dataset

Now, we load the **customers_1mb.csv** dataset.

[3]:
```
# Load CSV data into a DataFrame
df = spark.read \
    .format('csv') \
    .option('header', 'True') \
    .option('inferSchema', 'True') \
    .load('/tmp/customers_100.csv')

# Display DataFrame schema
df.printSchema()
```

```
[Stage 1:>                                                          (0 + 1) / 1]
```

```
root
 |-- customer_id: integer (nullable = true)
 |-- name: string (nullable = true)
 |-- city: string (nullable = true)
 |-- state: string (nullable = true)
 |-- country: string (nullable = true)
 |-- registration_date: timestamp (nullable = true)
 |-- is_active: boolean (nullable = true)
```

1.2.1 Creating a Temporary View

We'll create a **temporary view** to allow SQL-like queries before creating tables.

[4]:
```
# Create a temporary view for querying
df.createOrReplaceTempView("temp_customers")

# Query the view
spark.sql("SELECT * FROM temp_customers LIMIT 5").show()
```

```
+-----------+----------+---------+-----------+-------+------------------+------
---+
|customer_id|      name|     city|      state|country|
registration_date|is_active|
+-----------+----------+---------+-----------+-------+------------------+------
---+
|          0|Customer_0|     Pune|Maharashtra|  India|2023-06-29 00:00:00|
false|
|          1|Customer_1|Bangalore| Tamil Nadu|  India|2023-12-07 00:00:00|
true|
```

```
|          2|Customer_2|Hyderabad|    Gujarat|  India|2023-10-27 00:00:00|
true|
|          3|Customer_3|Bangalore|  Karnataka|  India|2023-10-17 00:00:00|
false|
|          4|Customer_4|Ahmedabad|  Karnataka|  India|2023-03-14 00:00:00|
false|
+-----------+----------+---------+-----------+-------+-------------------+------
---+
```

1.3 Creating a Managed Table

- Spark **stores** the data inside the **warehouse directory** (`/tmp/mera/warehouse`).
- If you **drop** this table, the **data is also deleted**.

[5]:
```
# Creating a Managed Table
spark.sql("DROP TABLE IF EXISTS managed_customers")
spark.sql("""
    CREATE TABLE managed_customers AS
    SELECT * FROM temp_customers
""")
print("  Managed table 'managed_customers' created.")
```

```
ivysettings.xml file not found in HIVE_HOME or
HIVE_CONF_DIR,/etc/hive/conf.dist/ivysettings.xml will be used
25/02/02 03:35:50 WARN ResolveSessionCatalog: A Hive serde table will be created
as there is no table provider specified. You can set
spark.sql.legacy.createHiveTableByDefault to false so that native data source
table will be created instead.
25/02/02 03:35:50 WARN SessionState: METASTORE_FILTER_HOOK will be ignored,
since hive.security.authorization.manager is set to instance of
HiveAuthorizerFactory.

  Managed table 'managed_customers' created.
```

[9]:
```
spark.sql('describe extended managed_customers').show(truncate=False)
```

```
+----------------------------+------------------------------------------------
-------+-------+
|col_name                    |data_type
|comment|
+----------------------------+------------------------------------------------
-------+-------+
|customer_id                 |int
|null   |
|name                        |string
|null   |
|city                        |string
```

```
|null   |
|state                        |string
|null   |
|country                      |string
|null   |
|registration_date            |timestamp
|null   |
|is_active                    |boolean
|null   |
|                             |
|       |
|# Detailed Table Information|
|       |
|Database                     |default
|       |
|Table                        |managed_customers
|       |
|Owner                        |root
|       |
|Created Time                 |Sun Feb 02 03:35:50 UTC 2025
|       |
|Last Access                  |UNKNOWN
|       |
|Created By                   |Spark 3.3.2
|       |
|Type                         |MANAGED
|       |
|Provider                     |hive
|       |
|Table Properties             |[transient_lastDdlTime=1738467352]
|       |
|Statistics                   |6315 bytes
|       |
|Location                     |hdfs://my-
cluster-m/user/hive/warehouse/managed_customers|       |
+-----------------------------+-------------------------------------------------
-------+-------+
only showing top 20 rows
```

[7]:
```
!hdfs dfs -ls /user/hive/warehouse/managed_customers
```

```
Found 1 items
-rwxr-xr-x   2 root hadoop       6315 2025-02-02 03:35
/user/hive/warehouse/managed_customers/part-00000-695ee9bd-
dbce-4dda-8f90-c66e35b19d20-c000
```

1.4 Creating an External Table

- The data **remains** in `/tmp/customers_1mb.csv`.
- If you **drop** this table, the **data is not deleted**.

[10]:
```
# Creating an External Table
spark.sql("DROP TABLE IF EXISTS external_customers")
spark.sql("""
    CREATE EXTERNAL TABLE external_customers
    USING CSV
    LOCATION '/tmp/customers_1mb.csv'
""")
print(" External table 'external_customers' created.")
```

```
 External table 'external_customers' created.
```

```
25/02/02 03:41:16 WARN HiveExternalCatalog: Couldn't find corresponding Hive
SerDe for data source provider CSV. Persisting data source table
`default`.`external_customers` into Hive metastore in Spark SQL specific format,
which is NOT compatible with Hive.
```

[12]:
```
spark.sql('describe extended external_customers').show(truncate=False)
```

```
+----------------------------+--------------------------------------------------
+-------+
|col_name                    |data_type
|comment|
+----------------------------+--------------------------------------------------
+-------+
|_c0                         |string
|null   |
|_c1                         |string
|null   |
|_c2                         |string
|null   |
|_c3                         |string
|null   |
|_c4                         |string
|null   |
|_c5                         |string
|null   |
|_c6                         |string
|null   |
|                            |
|       |
|# Detailed Table Information|
|       |
|Database                    |default
|       |
|Table                       |external_customers
```

```
|       |
|Owner                         |root
|       |
|Created Time                  |Sun Feb 02 03:41:16 UTC 2025
|       |
|Last Access                   |UNKNOWN
|       |
|Created By                    |Spark 3.3.2
|       |
|Type                          |EXTERNAL
|       |
|Provider                      |CSV
|       |
|Location                      |hdfs://my-cluster-m/tmp/customers_1mb.csv
|       |
|Serde Library
|org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe|       |
|InputFormat                   |org.apache.hadoop.mapred.SequenceFileInputFormat
|       |
+------------------------------+--------------------------------------------------
+-------+
only showing top 20 rows
```

1.4.1 Verify External Table

Run the following command to check its location:

```
!hdfs dfs -ls /tmp/customers_1mb.csv
```

1.5 Verifying Tables in Hive Metastore

You can check all available tables in Spark using:

[13]:
```
# Show tables in Spark
spark.sql("SHOW TABLES").show()
```

```
+---------+--------------------+-----------+
|namespace|           tableName|isTemporary|
+---------+--------------------+-----------+
|  default|customers_persistent|      false|
|  default|  external_customers|      false|
|  default|   managed_customers|      false|
|         |      temp_customers|       true|
+---------+--------------------+-----------+
```

1.5.1 Dropping the Tables

Dropping a **Managed Table** deletes both **metadata and data**, while dropping an **External Table** deletes only **metadata**.

[14]:
```
# Drop managed table (Data is deleted!)
spark.sql("DROP TABLE IF EXISTS managed_customers")

# Drop external table (Data is NOT deleted!)
spark.sql("DROP TABLE IF EXISTS external_customers")

print(" Tables dropped successfully.")
```

```
Tables dropped successfully.
```

1.6 Summary

Feature	Managed Table	External Table
Data Location	Inside warehouse	Custom location
Dropping Table	Deletes data	Only deletes metadata
Performance	Optimized by Spark	Depends on external storage

[ ]:
```
# Why Spark uses Hive Metastore for Sceh
```

[18]:
```
spark.stop()
```

[ ]:
```
!hadoop fs -mkdir /tmp/mera
```

[12]:
```
!hadoop fs -mkdir /tmp/mera/warehouse
```

```
mkdir: `/tmp/mera/warehouse': File exists
```

[11]:
```
#  Step 1: Import necessary libraries
from pyspark.sql import SparkSession


warehouse_location = '/tmp/mera/warehouse'

#  Step 3: Create Spark session with Hive support
spark_self = spark \
    .config('spark.sql.warehouse.dir', warehouse_location)\
    .enableHiveSupport() \
    .newSession()

# print(f"Spark session created with warehouse location: {warehouse_location}")
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
```

```
/tmp/ipykernel_21247/3499198947.py in <cell line: 8>()
      7 #   Step 3: Create Spark session with Hive support
      8 spark_self = spark \
----> 9     .config('spark.sql.warehouse.dir', warehouse_location)\
     10     .enableHiveSupport() \
     11     .newSession()

AttributeError: 'SparkSession' object has no attribute 'config'
```

[2]:
```
# Show the configured warehouse directory
spark.conf.get('spark.sql.warehouse.dir')
```

[2]: 'file:/spark-warehouse'