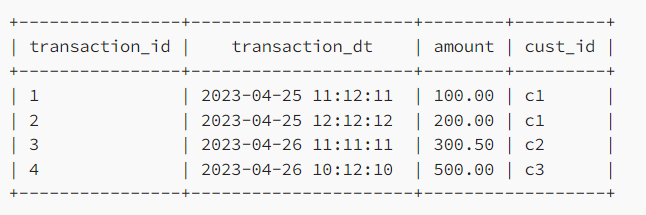
In this blog we will explore “Hidden Partitioning” concept in Apache Iceberg.

1. Let’s first discuss about partitioning in Apache Hive. Suppose we have data like this (refer below) & want to store data in Hive based on daily partition.



We will create table something like this, by explicitly giving the partition value column.

CREATE TABLE sales\_data (

transaction\_id int,

transaction\_dt timestamp,

amount double,

cust\_id string

)

PARTITIONED by (day\_part string)

Also, while ingesting the data into table, it is our responsibility to create partition value by transforming the exisingtransformation\_dt timestamp column to date format yyyy-MM-dd. See below example.

INSERT OVERWRITE TABLE sales\_data

PARTITION (day\_part)

SELECT

transaction\_id,

transaction\_dt,

amount,

cust\_id string,

substring(transaction\_dt,1,10) as part\_day -- creating partition value

FROM temp\_view;

In the above insert query, we have transformed the source timestamp column to day format yyyy-MM-dd in-order to generate partition values.

Now while querying the data from partition table, we have to explicitly pass the partition value column in query to take benefit of partition pruning. Everytime we insert/query table we have to remember the table physical layout.

select \* from sales\_data where part\_day = '2023-04-25' and <other-filters>

For monthly based partitioned table, we should have used a transformation like this while ingesting data for creating a month partition value: substring(transaction\_dt, 1, 7) as part\_month

1. How Apache Iceberg do differently, let's explore. As per the official document of Iceberg.

a. Iceberg handles the tedious and error-prone task of producing partition values for rows in a table.

b. Consumers don’t need to know how the table is partitioned and add extra filters to their queries.

So, we don't have to explicitly provide the partition value while table creation and data ingestion, Iceberg will responsible for creating partition values and tracking the relationship. See below example:

ddl = """

CREATE TABLE dev.sales\_data

(trnsaction\_id int,

transaction\_dt timestamp,

amount double,

cust\_id string)

USING iceberg

PARTITIONED BY (day(transaction\_dt))

"""

spark.sql(ddl)

spark.sql("DESCRIBE EXTENDED dev.sales\_data").show(20,False)

Here we have used source column transaction\_dt of timestamp type & partition transform day to create a partition values. Iceberg support different type of transform like identity, bucket[N], truncate[W], year, month, day, hour, void refer complete transform list here.

Let’s ingest some data in Iceberg table dev.sales\_data & explore.

from datetime import datetime

from pyspark.sql.types import \*

data = [

(1, datetime.strptime("2023-04-25 11:12:11", '%Y-%m-%d %H:%M:%S'), 100.00, "C1"),

(2, datetime.strptime("2023-04-25 12:12:12", '%Y-%m-%d %H:%M:%S'), 200.00, "C1"),

(3, datetime.strptime("2023-04-26 11:11:11", '%Y-%m-%d %H:%M:%S'), 200.00, "C2"),

(4, datetime.strptime("2023-04-26 10:12:10", '%Y-%m-%d %H:%M:%S'), 300.50, "C2"),

(5, datetime.strptime("2023-04-27 11:12:10", '%Y-%m-%d %H:%M:%S'), 300.00, "C3"),

]

schema = StructType([

StructField("trnsaction\_id", IntegerType(), True), \

StructField("transaction\_dt", TimestampType(), True), \

StructField("amount", DoubleType(), True), \

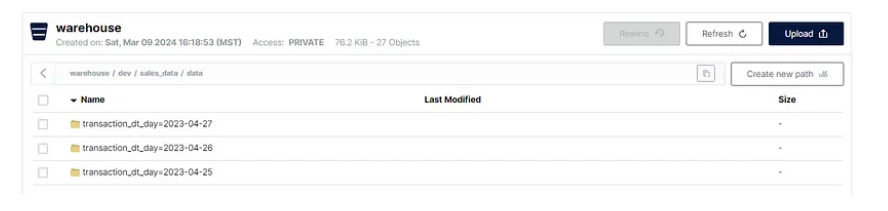
StructField("cust\_id", StringType(), True), \

])

df = spark.createDataFrame(data=data,schema=schema)

df.writeTo("dev.sales\_data").append()

See below Iceberg has created the partitions on day, as we have used (day(transaction\_dt) transform. Note that, we have not added any code in the insert statement explicitly for the partition column.



Now, let’s query the Iceberg table to fetch the records for partition date2023–04–25 .

from pyspark.sql.functions import col

spark.table('dev.sales\_data').filter((col('transaction\_dt')>='2023-04-25 00:00:00') & (col('transaction\_dt')<'2023-04-26 00:00:00')).show()

In above query we haven't explicitly provided the partition value in filters, just provided two timestamps. This concept is nothing but “Hidden Partitioning”, where user doesn’t worry about physical layout.