Chapter 19: Other SnowFlake Objects. File Formats & Sequences

SnowFlake File Formats and Sequences explanation

As we saw in the chapter on SnowFlake architecture, there were several objects in SnowFlake. Tables, views, stages, pipes, stored procedures, UDFs... We have seen all of them during this course, although we missed two file formats and sequences. In this chapter, we will see these two new objects!

FILE FORMATS

A named file format object describes and stores the format information required to load data into SnowFlake tables. You can specify different parameters, for example, the delimiter of the file, whether you want to skip the header or not...

Some supported File Formats in SnowFlake.

SnowFlake supports both Structured and Semi-Structured Data, so just as an example, you can store JSON files into tables. Let's see the differences between them:

Structured Data

CSV → You can Load and Unload files in CSV format. It's the fastest file format to load data.

Semi-structured Data

Semi-structured data is saved as Variant type in SnowFlake tables, with a maximum limit size of 16MB, and it can be queried using JSON notation. You can store arrays, objects... Non-native values, such as dates and timestamps, are stored as strings when loaded into a VARIANT column.

- **JSON** → It's used for both loading & unloading data.
- **Parquet** → Binary format used for both loading & unloading data.
- **XML** → You can only load data in SnowFlake using the XML format.
- **Avro** → Binary format used to load data.
- **ORC** → Binary format used to load data.

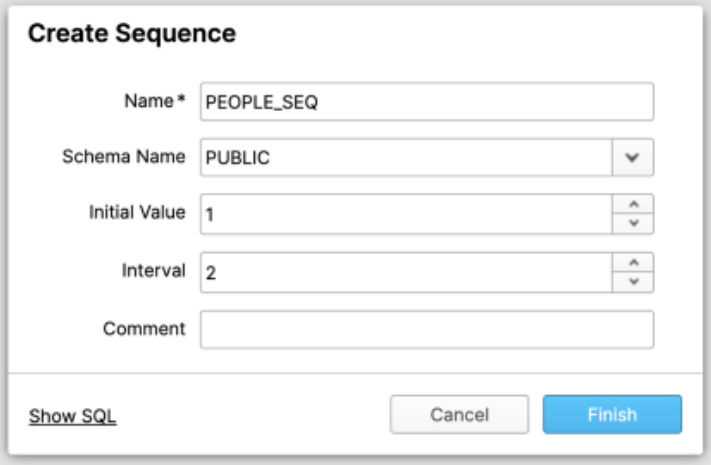
Format	Type	Load	Unload	Binary format
CSV	Structured	Yes	Yes	No
JSON	Semi-structured	Yes	Yes	No
Parquet	Semi-structured	Yes	Yes	Yes
XML	Semi-structured	Yes	No	No
AVRO	Semi-structured	Yes	No	Yes
ORC	Semi-structured	Yes	No	Yes

Different file formats in SnowFlake

You can use the function FLATTEN to convert semi-structured data to a relational representation. It takes a Variant, Object, or Array column and produces a lateral view.

SEQUENCES

We use sequences to generate unique numbers across sessions and statements, including concurrent statements. You can use them to generate values for a primary key or any column that requires a unique value. They have an initial value and an interval.



Creating sequences in SnowFlake.

You can access sequences in queries as expressions. Using the function "nextval", will generate a set of distinct values. I don't think that the syntax will appear in the SnowFlake SnowPro Core exam, but let's make an example. Having the previous sequence "PEOPLE_SEQ" this is how we can increase it:

```
INSERT INTO PEOPLE (ID, NAME) VALUES
(PEOPLE_SEQ.nextval, "Gonzalo"),
(PEOPLE_SEQ.nextval, "Nacho"),
(PEOPLE_SEQ.nextval, "Megan"),
(PEOPLE_SEQ.nextval, "Angel")
```

We created a sequence with the initial value of 1 and an interval of 2. So the first value is going to be 1, and the second one 3. The result is shown in the next table:

```
+--------+---------+
| ID     |  NAME   |
+--------+---------+
|   1    | Gonzalo |
+--------+---------+
|   3    |  Nacho  |
+--------+---------+
|   5    |  Megan  |
+--------+---------+
|   7    |  Angel  |
+--------+---------+
```

We can also use the "Default" statement when creating the tables, which will increase the sequence automatically.

```
CREATE OR REPLACE TABLE PEOPLE
(
ID NUMBER DEFAULT PEOPLE_SEQ.nextval,
NAME VARCHAR(50)
)

-------------

INSERT INTO PEOPLE (NAME) VALUES
("Gonzalo"),
("Nacho"),
("Megan"),
("Angel")
```

This will generate the same result as before.

TYPICAL EXAM QUESTIONS

1. Does SnowFlake allow only the load of structure data?

1. True
2. False

Solution: 2.

2. Which of the following file formats are supported by SnowFlake?

1. CSV
2. XML
3. TXT
4. Avro

Solution: 1, 2, 4.

3. How will you store JSON data in SnowFlake?

1. Using a column of the JSON type
2. Using a column of the VARCHAR type
3. Using a column of the VARIANT type

4. Using a column of the NULL type

Solution: 3.

4. Which of the following object types are stored within a schema?

1. Tables
2. Views
3. File Formats
4. UDFs
5. Roles
6. Users
7. Sequences

Solution: 1, 2, 3, 4, 7.

6. Which table function allows you to convert semi-structured data to a relational representation?

1. FLATTEN
2. CHECK_JSON
3. PARSE_JSON

Solution: 1.

5. You have the following data in a variant column from the table "myTable". How can you query the favorite technology that Gonzalo uses?

```
{
  "name": "Gonzalo",
  "country": "Spain",
  "favouriteTechnology": SnowFlake,
  "hobbies": [
    {
      "name": "soccer",
      "since": "2000",
    },{
      "name": "music",
      "since": "2005",
    },{
      "name": "technology",
      "since": "1996",
    }
  ]
}
```

1. SELECT favouriteTechnology FROM myTable;
2. SELECT src:favouriteTechnology FROM myTable;
3. SELECT src:$favouriteTechnology FROM myTable;
4. SELECT CONVERT_JSON(src:favouriteTechnology) FROM myTable;

Solution: 2. In the SnowFlake SnowPro Core exam, they won't probably ask you about syntax; that's why I didn't explain it before. But it's really useful to know this command.

7. What file format provides the fastest load performance?

1. Parquet
2. JSON
3. Avro
4. CSV

Solution: 4.

Thanks for Reading!