

Data Sheet

# Working with Event Tables in Snowflake

A Detailed Guide

5201 GREAT AMERICAN PARKWAY, SUITE 320

SANTA CLARA, CA 95054

Tel: (855) 695-8636

E-mail: [info@lumendata.com](mailto:info@lumendata.com)

Website: [www.lumendata.com](http://www.lumendata.com)

**Event tables in Snowflake** help us to capture log entries and trace events. The table's structure is defined in a way that it can hold both predefined data and data you have.

To capture log entries and trace events, we need to associate the event table to the account.

To set up an event table for logging, we need to set the level for log and trace events.

### Setting the Log Level

We can set the log level on the following objects:

- A stored procedure.
- A user-defined function (UDF) or a user-defined table function (UDTF).
- A database or schema containing procedures and functions.

We can set the Log Level by using the `LOG_LEVEL` Parameter.

### Setting the Trace Level

We can set the log level on the following objects:

- A stored procedure.
- A user-defined function (UDF) or a user-defined table function (UDTF).
- A database or schema containing procedures and functions.

We can set the Trace Level by using the `TRACE_LEVEL` Parameter.

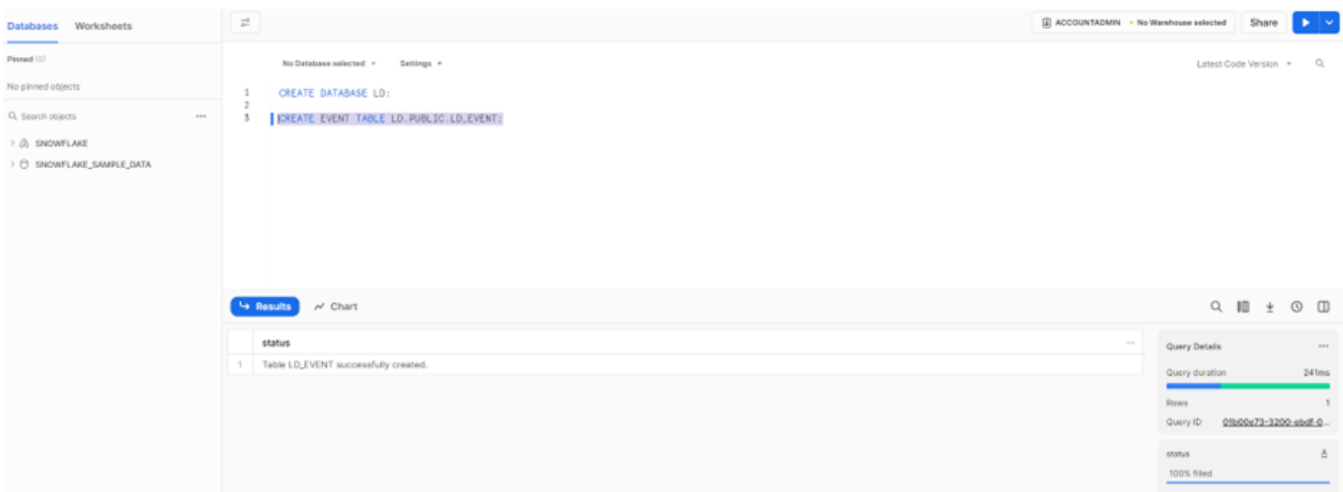
With an event table, we can perform only the following operations:

- `SHOW EVENT TABLES`
- `DESCRIBE EVENT TABLE`
- `DROP TABLE`
- `UNDROP TABLE`
- `TRUNCATE TABLE`
- `DELETE`
- `ALTER TABLE`

## Logging and Tracing Limitations

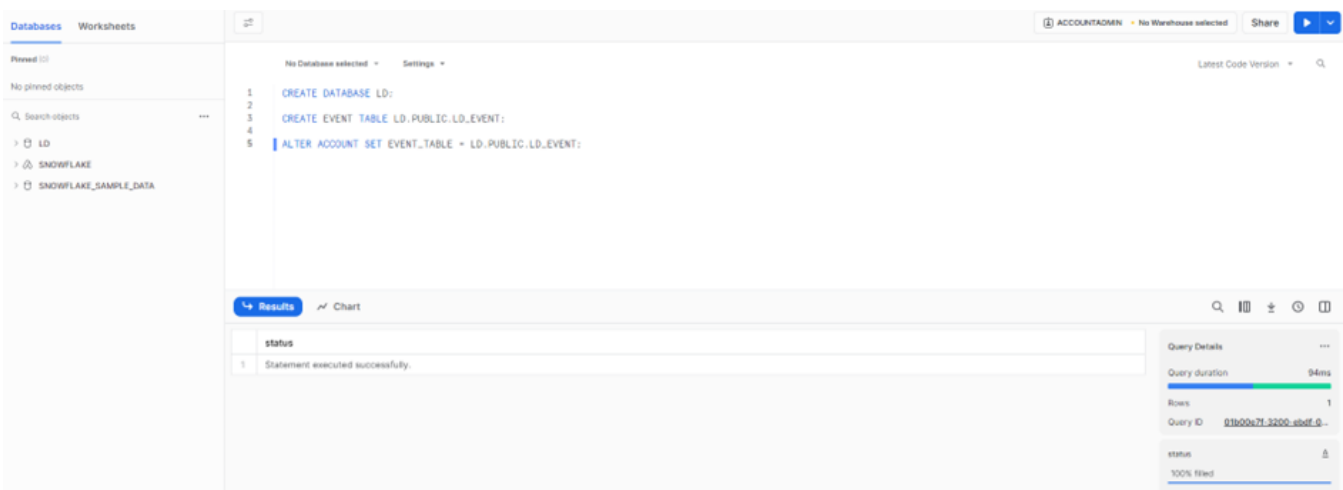
There is a 1MB limit for log and trace event payloads. If the payload is over the **1MB** threshold, the record in the event table will be incomplete and only contain values for the following columns: **TIMESTAMP**, **RECORD\_TYPE**, and **RESOURCE\_ATTRIBUTES**.

1. Create an event table named "LD\_EVENT" by using the below command:  
**CREATE EVENT TABLE LD.PUBLIC.LD\_EVENT;**



The screenshot shows the Snowflake SQL Editor interface. The SQL editor contains three lines of code: 1. CREATE DATABASE LD; 2. (blank line) 3. CREATE EVENT TABLE LD.PUBLIC.LD\_EVENT; The third line is highlighted in blue. Below the editor, the 'Results' tab is active, showing a single row with the status 'Table LD\_EVENT successfully created.' The 'Query Details' panel on the right shows a query duration of 241ms, 1 row, and a status of '100% filled'.

2. Now, we will associate the event table that we created with the account:  
**ALTER ACCOUNT SET EVENT\_TABLE = LD.PUBLIC.LD\_EVENT;**



The screenshot shows the Snowflake SQL Editor interface. The SQL editor contains five lines of code: 1. CREATE DATABASE LD; 2. (blank line) 3. CREATE EVENT TABLE LD.PUBLIC.LD\_EVENT; 4. (blank line) 5. ALTER ACCOUNT SET EVENT\_TABLE = LD.PUBLIC.LD\_EVENT; The fifth line is highlighted in blue. Below the editor, the 'Results' tab is active, showing a single row with the status 'Statement executed successfully.' The 'Query Details' panel on the right shows a query duration of 94ms, 1 row, and a status of '100% filled'.

3. We can confirm whether EVENT\_TABLE values with SHOW PARAMETERS Command:  
**SHOW PARAMETERS LIKE 'event\_table' IN ACCOUNT;**

The screenshot shows the Snowflake SQL Editor interface. The top bar indicates the user is 'ACCOUNTADMIN' and 'No Warehouse selected'. The main editor area contains the following SQL code:

```
1 CREATE DATABASE LD;  
2  
3 CREATE EVENT TABLE LD.PUBLIC.LD_EVENT;  
4 ALTER ACCOUNT SET EVENT_TABLE = LD.PUBLIC.LD_EVENT;  
5  
6  
7 SHOW PARAMETERS LIKE 'event_table' IN ACCOUNT;
```

Below the code, the 'Results' tab is active, displaying a table with the following data:

key	value	default	level	description	type
EVENT_TABLE	LD.PUBLIC.LD_EVENT		ACCOUNT	Event destination for the given target.	STRING

On the right side of the results table, there is a 'Query Details' panel showing:

- Query duration: 79ms
- Rows: 1
- Query ID: 01600e81-3200-4bb...
- key: 100% filled
- value: 100% filled
- default: 100% filled

4. Will set the Log and Trace levels to the session by using the below command.

```
ALTER SESSION SET LOG_LEVEL = DEBUG;  
ALTER SESSION SET TRACE_LEVEL = ALWAYS;
```

5. Now, we will create a Stored Procedure to capture the CURRENT\_TIMESTAMP:

```
CREATE OR REPLACE PROCEDURE LD.PUBLIC.CURRENT_TIMESTAMP_1()  
RETURNS VARCHAR(16777216)  
LANGUAGE JAVASCRIPT  
EXECUTE AS CALLER  
AS '
```

```
    var start_time_command= "SELECT "" + start_time_query +  
    ""::TIMESTAMP_NTZ";  
    var start_time_command_create = snowflake.createStatement({sqlText:  
start_time_command});  
    var start_time_command_result = start_time_command_create.execute();  
return start_time_command_result;  
';
```

```
CALL CURRENT_TIMESTAMP_1();
```

- Once we have the stored Procedure, we need to ALTER the procedure and set the Log and Trace Logs by using the below commands:

```
ALTER PROCEDURE LD.PUBLIC.CURRENT_TIMESTAMP_1() SET LOG_LEVEL = INFO;  
ALTER PROCEDURE LD.PUBLIC.CURRENT_TIMESTAMP_1() SET TRACE_LEVEL = ALWAYS;
```

- Now, if we call the stored procedure again and do a SELECT query on the event table, we view a few logs and traces that got captured.

```
CALL CURRENT_TIMESTAMP_1();  
SELECT * FROM LD.PUBLIC.LD_EVENT;
```

The screenshot shows the Snowflake SQL Editor interface. The SQL code in the editor is as follows:

```
LD.PUBLIC - Settings +  
16 EXECUTE AS CALLER  
17 AS *  
18  
19 var start_time_command = "SELECT '' + start_time_query + ''::TIMESTAMP_NTZ";  
20 var start_time_command_create = snowflake.createStatement({sqlText: start_time_command});  
21 var start_time_command_result = start_time_command_create.execute();  
22 return start_time_command_result;  
23  
24  
25 CALL CURRENT_TIMESTAMP_1();  
26  
27 ALTER PROCEDURE LD.PUBLIC.CURRENT_TIMESTAMP_1() SET LOG_LEVEL = INFO;  
28 ALTER PROCEDURE LD.PUBLIC.CURRENT_TIMESTAMP_1() SET TRACE_LEVEL = ALWAYS;  
29  
30 SELECT * FROM LD.PUBLIC.LD_EVENT;  
31
```

The Results tab shows a table with the following columns: **TIMESTAMP**, **START\_TIMESTAMP**, **OBSERVED\_TIMESTAMP**, **TRACE**, and **RESOURCE**. The first row contains the following data:

TIMESTAMP	START_TIMESTAMP	OBSERVED_TIMESTAMP	TRACE	RESOURCE
2023-11-02 09:35:42.211	2023-11-02 09:35:42.210		[{"span_id": "796848e755ec16cf", "trace_id": "01b00e9f3200ebdf00000006d75e4859"}]	

The TRACE column contains a JSON array with the following structure:

```
{  
  "span_id":  
    "796848e755ec16cf",  
  "trace_id":  
    "01b00e9f3200ebdf00000006d75e4859"  
}
```

The screenshot shows the Snowflake SQL Editor interface. The SQL code in the editor is as follows:

```
No Database selected - Settings +  
27 ALTER PROCEDURE LD.PUBLIC.CURRENT_TIMESTAMP_1() SET LOG_LEVEL = INFO;  
28 ALTER PROCEDURE LD.PUBLIC.CURRENT_TIMESTAMP_1() SET TRACE_LEVEL = ALWAYS;  
29  
30 SELECT * FROM LD.PUBLIC.LD_EVENT;  
31
```

The Results tab shows a table with the following columns: **RESOURCE**, **RESOURCE\_ATTRIBUTES**, **SCOPE**, **SCOPE\_ATTRIBUTES**, and **RECORD\_TYPE**. The first row contains the following data:

RESOURCE	RESOURCE_ATTRIBUTES	SCOPE	SCOPE_ATTRIBUTES	RECORD_TYPE
01b00e9f3200ebdf00000006d75e4859	[{"db.user": "SAT", "snow.database.id": 7, "snow.database.name": "LD", "snow.schema.id": 2, "snow.schema.name": "PUBLIC", "snow.query.id": "01b00e9f3200ebdf00000006d75e4859", "snow.session.id": 29363866855, "snow.session.role.primary.id": 2, "snow.session.role.primary.name": "ACCOUNTADMIN", "snow.user.id": 1, "snow.warehouse.id": 1}]]			SPAN

The RESOURCE\_ATTRIBUTES column contains a JSON array with the following structure:

```
{  
  "db.user": "SAT",  
  "snow.database.id": 7,  
  "snow.database.name": "LD",  
  "snow.executable.id": 2,  
  "snow.executable.name": "CURRENT_TIMESTAMP_1()",  
  "snow.executable.type": "PROCEDURE",  
  "snow.owner.id": 2,  
  "snow.owner.name": "ACCOUNTADMIN",  
  "snow.query.id": "01b00e9f3200ebdf00000006d75e4859",  
  "snow.schema.id": 2,  
  "snow.schema.name": "PUBLIC",  
  "snow.session.id": 29363866855,  
  "snow.session.role.primary.id": 2,  
  "snow.session.role.primary.name": "ACCOUNTADMIN",  
  "snow.user.id": 1,  
  "snow.warehouse.id": 1  
}
```

- As shown above, we can capture the TIMESATMP, TRACES, and RESOURCES\_ATTRIBUTES. In the RESOURCES\_ATTRIBUTES Snowflake captures all the details of the User who ran the query like database, schema, query\_id, warehouse name, session id, role name, etc as shown below.

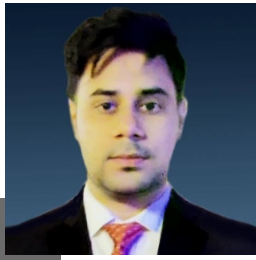
```
{
  "db.user": "SAI",
  "snow.database.id": 7,
  "snow.database.name": "LD",
  "snow.executable.id": 2,
  "snow.executable.name": "CURRENT_TIMESTAMP_1():VARCHAR(16777216)",
  "snow.executable.type": "PROCEDURE",
  "snow.owner.id": 2,
  "snow.owner.name": "ACCOUNTADMIN",
  "snow.query.id": "01b00e9f-3200-ebdf-0000-0006d75e4059",
  "snow.schema.id": 2,
  "snow.schema.name": "PUBLIC",
  "snow.session.id": 29383086085,
  "snow.session.role.primary.id": 2,
  "snow.session.role.primary.name": "ACCOUNTADMIN",
  "snow.user.id": 1,
  "snow.warehouse.id": 1,
  "snow.warehouse.name": "COMPUTE_WH",
  "telemetry.sdk.language": "javascript"
}
```

- Similarly, we can capture different levels of Logs & Traces level by using Event Tables.

## Authors



Sai Bharadwaja  
Consultant



Ankit Kumar  
Technical Lead

### About LumenData

LumenData is a leading provider of **Enterprise Data Management, Cloud & Analytics** solutions. We help businesses navigate their data visualization and analytics anxieties and enable them to accelerate their innovation journeys.

**Founded in 2008**, with locations in multiple countries, LumenData is privileged to serve over 100 leading companies. LumenData is **SOC2 certified** and has instituted extensive controls to protect client data, including adherence to GDPR and CCPA regulations.



Get in touch with us:  
[info@lumendata.com](mailto:info@lumendata.com)

Let us know what you need:  
[lumendata.com/contact-us](https://lumendata.com/contact-us)

