

Cloudera Runtime 7.2.17

## Iceberg support for Atlas

Date published: 2023-06-26

Date modified:

The Cloudera logo is displayed in a bold, orange, sans-serif font. The word "CLOUDERA" is written in all caps, with the letter 'E' in "CLouDERA" featuring a unique design where the top bar is a horizontal line with a small triangle pointing downwards from its center.

<https://docs.cloudera.com/>

# Legal Notice

© Cloudera Inc. 2024. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

# Contents

<b>Iceberg support for Atlas (Technical Preview).....</b>	<b>4</b>
How Atlas works with Iceberg.....	4
<b>Using the Spark shell.....</b>	<b>5</b>
<b>Using the Hive shell.....</b>	<b>15</b>
<b>Using the Impala shell.....</b>	<b>22</b>

## Iceberg support for Atlas (Technical Preview)

Atlas integration with Iceberg helps you identify the Iceberg tables to scan data and provide lineage support. Learn how Atlas works with Iceberg and what schema evolution, partition specification, partition evolution are with examples.

### How Atlas works with Iceberg

You can use Atlas to find, organize, and manage different aspects of data about your Iceberg tables and how they relate to each other. This enables a range of data stewardship and regulatory compliance use cases.

The Atlas connectors distinguish between Hive and Iceberg tables. The Iceberg table is available in a “typedef” format which implies that the underlying data can be retrieved by querying the Iceberg table. All attributes of the Hive table are available in the Iceberg table and this equivalence is achieved by creating the Iceberg table as a sub-type of the underlying Hive table. Optionally, the Iceberg table can also be queried by Hive or Impala engine. For more information about Iceberg and related concepts, see [Apache Iceberg features](#) and [Apache Iceberg in CDP](#).

Both Iceberg and Hive tables have equality in Atlas in terms of data tagging. Data evolution and transformation are features unique to Iceberg tables. Iceberg adds tables to compute engines including Spark, Hive and Impala using a high-performance table format that works just like a SQL table. Also, the lineage support for Iceberg table is available. For example, when a Hive table is converted to Iceberg format, the lineage is displayed for the conversion process in Atlas UI.

- Migration of Hive tables to Iceberg is achieved with the following:
  - Using in-place migration by running a Hive query with the ALTER TABLE statement and setting the table properties
  - Executing CTAS command from Hive table to the Iceberg table.
- Schema evolution allows you to easily change a table's current schema to accommodate data that changes over time. Schema evolution enables you to update the schema that is used to write new data while maintaining backward compatibility with the schemas of your old data. Later the data can be read together assuming all of the data has one schema.
  - Iceberg tables supports the following schema evolution changes:
    - Add – add a new column to the table or to a nested struct
    - Drop– remove an existing column from the table or a nested struct
    - Rename– rename an existing column or field in a nested struct
    - Update– widen the type of a column, struct field, map key, map value, or list element
    - Reorder – change the order of columns or fields in a nested struct
  - Partition specification allows you to initiate queries faster by grouping similar rows together when writing.

As an example, queries for log entries from a logs table usually include a time range, like the following query for logs between 10 A.M. and 12 A.M.

```
SELECT level, message FROM logs
WHERE event_time BETWEEN '2018-12-01 10:00:00' AND '2018-12-01 12:00:00'
```

Configuring the logs table to partition by the date of event\_time groups log events into files with the same event date. Iceberg keeps track of that date and uses it to skip files for other dates that do not have useful data.

- Partition evolution across Iceberg table partitioning can be updated in an existing table because queries do not reference partition values directly.

When you evolve a partition specification, the old data written with an earlier specification remains unchanged. New data is written using the new specification in a new layout. The metadata for each of the partition versions is stored separately.

Due to this nature of partition evolution, when you start writing queries, you get split planning. This is where each partition layout plans files separately using the filter it derives for that specific partition layout.

## Using the Spark shell

Using Spark, you can create an Iceberg table followed by schema evolution, partition specification, and partition evolution.

### Before you begin

You must configure the Spark shell as such you have included the valid Spark runtime version.

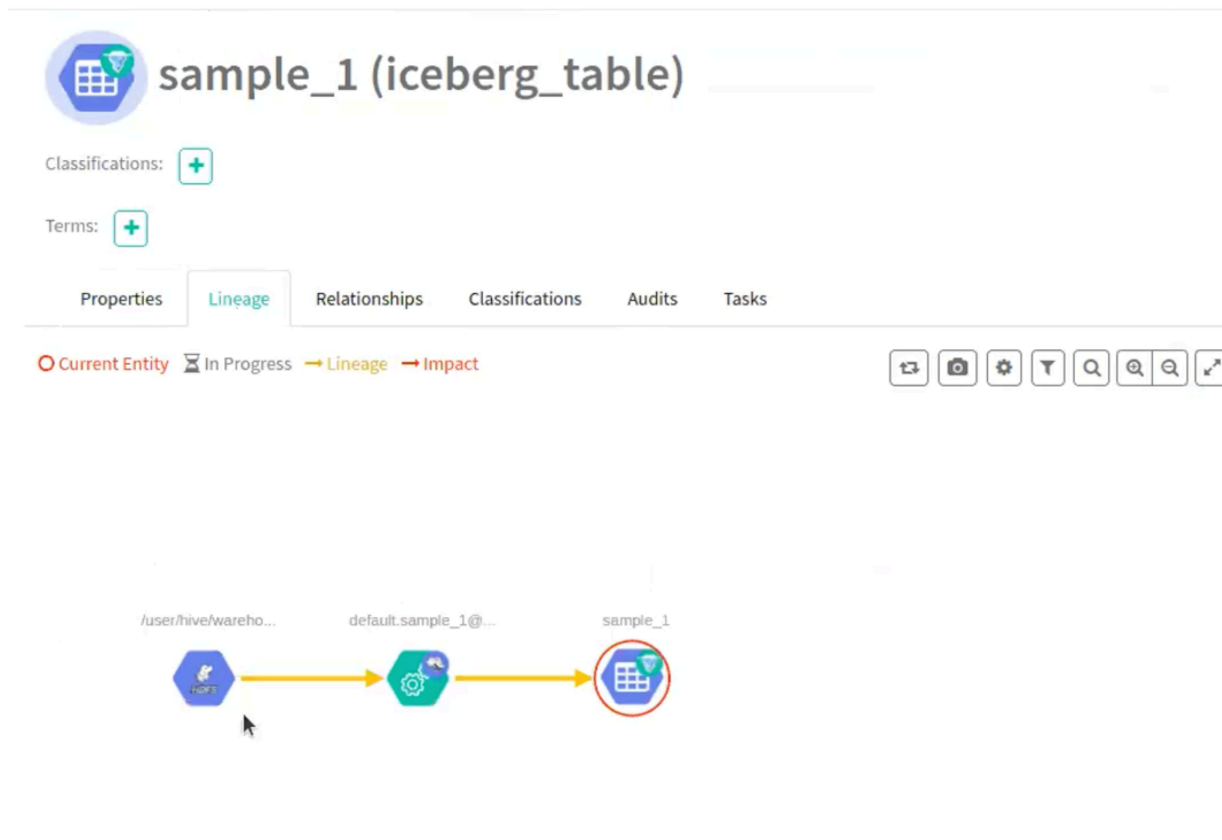
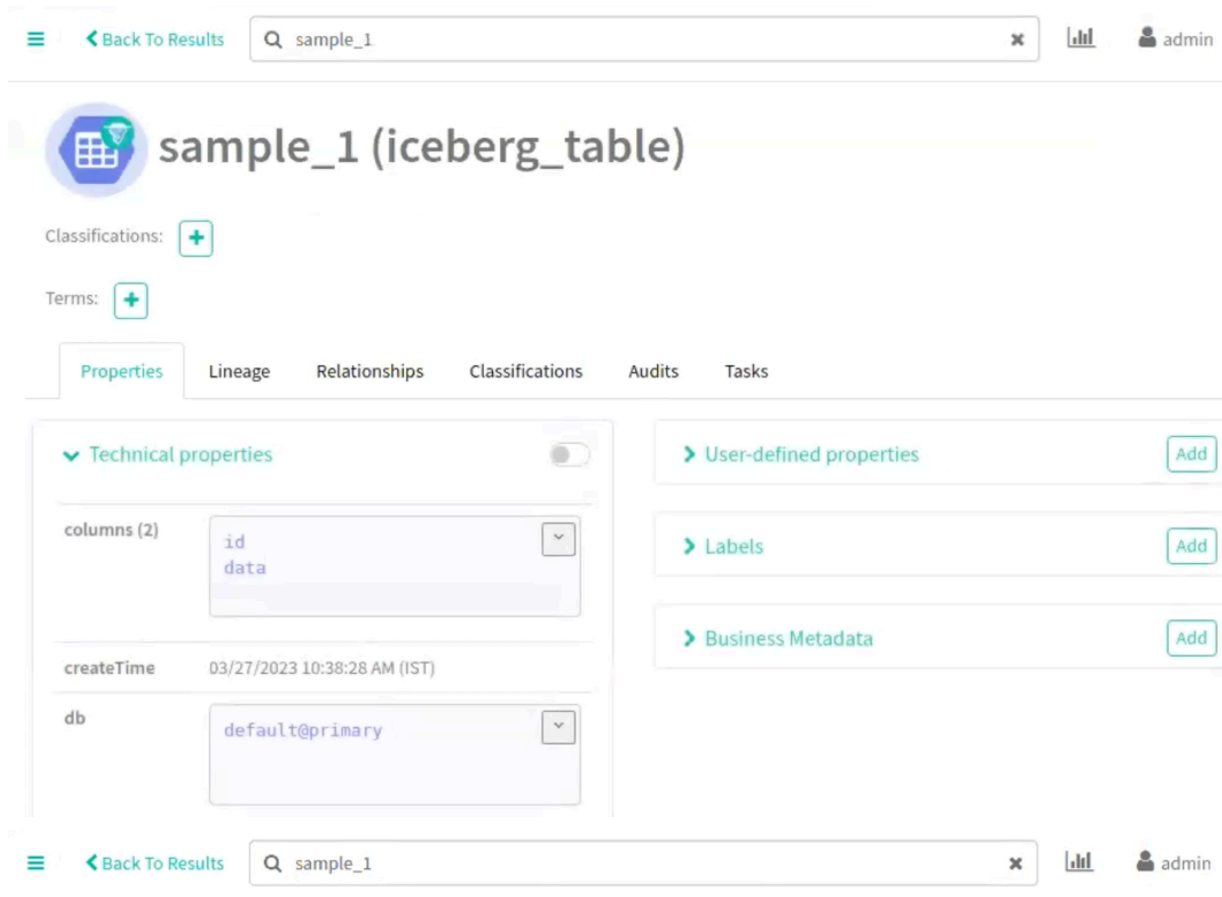
Run the following command in your Spark shell to create a new Iceberg table

### Procedure

1. `spark.sql("CREATE TABLE spark_catalog.default.sample_1 ( id bigint COMMENT 'unique id', data string) USING iceberg");`

- 2. Navigate accordingly in the Atlas UI to view the changes.

The following images provide information about Iceberg table creation process.



Cloudera Runtime interface showing the lineage of a table named `sample_1`. The search bar contains `sample_1` and the user is `admin`.

Classifications: +  
Terms: +

Properties **Lineage** Relationships Classifications Audits Tasks

○ Current Entity ⌚ In Progress → Lineage → Impact

Diagram illustrating the lineage of the table `sample_1`:

```

graph LR
    A["/user/hive/wareho..."] --> B["default.sample_1@..."]
    B --> C["sample_1"]
  
```

The diagram shows a flow from a source (represented by a blue icon) to a table (represented by a green icon) and finally to the target table `sample_1` (represented by a blue icon with a red border).

Properties of the `iceberg_table` (highlighted in blue):

Property	Value
guid	3b7b9d10-d773-4281-a318-7d30c6886c9f
typeName	iceberg_table
name	sample_1
qualifiedName	default.sample_1@primary
owner	spark
createTime	1679893708000
status	ACTIVE
classifications	N/A

Run the following command in your Spark shell to create a Schema Evolution in a new table. For example - `sample_2`.

3. `spark.sql("CREATE TABLE spark_catalog.default.sample_2 ( id bigint COMMENT 'unique id', data string) USING iceberg");`

4. Navigate accordingly in the Atlas UI to view the changes.

The following image provide information about Iceberg schema evolution process.

The screenshot shows the Cloudera Atlas interface for a table named 'sample\_2 (iceberg\_table)'. The 'Lineage' tab is selected, and the 'Table' view is active. The table's properties are displayed in a table format:

Key	Value	Show Empty Values
columns (2)	data id	<input type="checkbox"/>
db	default@primary	<input type="checkbox"/>
outputFromProcesses (1)	default.sample_2@primary:1679893777000	<input type="checkbox"/>

Run the following command in your Spark shell to include a column:

5. `spark.sql("ALTER TABLE spark_catalog.default.sample_2 ADD COLUMN (add_col_1 string)");`



- 6. Navigate accordingly in the Atlas UI to view the changes.

The following images provide information about Iceberg schema creation process.

The screenshot displays the Cloudera Atlas interface for an Iceberg table. At the top, there is a navigation bar with a 'Back To Results' link, a search bar for entities, and a user profile for 'admin'. The main header shows the table name 'sample\_2 (iceberg\_table)' with a table icon. Below the header, there are sections for 'Classifications' and 'Terms', both with a plus sign to add more. A horizontal menu allows switching between 'Properties', 'Lineage', 'Relationships', 'Classifications', 'Audits', and 'Tasks'. The 'Properties' tab is active, showing a 'Technical properties' section with a toggle switch. This section lists: 'columns (3)' with a dropdown menu containing 'data', 'add\_col\_1', and 'id'; 'createTime' as '03/27/2023 10:39:37 AM (IST)'; 'db' as 'default@primary' with a dropdown menu; and 'lastAccessTime' as '01/07/1970 05:29:24 PM (IST)'. To the right, three expandable sections are visible: 'User-defined properties', 'Labels', and 'Business Metadata', each with an 'Add' button.

The screenshot displays the Cloudera Runtime interface for a table named 'sample\_2'. At the top, there is a navigation bar with a 'Back To Results' link, a search box for entities, and a user profile for 'admin'. Below this, a table header shows 'Users', 'Timestamp', and 'Actions'. The main content area shows the table 'sample\_2' with a timestamp of '03/27/2023 10:39:57 AM (IST)' and an action of 'Entity Updated'. The table details are divided into two sections: 'Technical properties' and 'Relationship properties'. The 'Technical properties' section includes fields for 'comment' (N/A), 'createTime' (03/27/2023 10:39:37 AM (IST)), 'lastAccessTime' (01/07/1970 05:29:24 PM (IST)), 'name' (sample\_2), 'owner' (spark), 'parameters' (a JSON object with 'owner: "spark"'), 'qualifiedName' (default.sample\_2@primary), 'retention' (2147483647), and 'tableType' (EXTERNAL\_TABLE). The 'Relationship properties' section includes 'columns (3)' (id, data, add\_col\_1), 'db' (default), 'partitionKeys' (N/A), and 'sd' (default.sample\_2@primary\_storage).

Run the following command in your Spark shell to include the second column:

7. `spark.sql("ALTER TABLE spark_catalog.default.sample_2 ADD COLUMN (add_col_2 string)");`

8. Navigate accordingly in the Atlas UI to view the changes.

The following image provide information about Iceberg schema creation process.

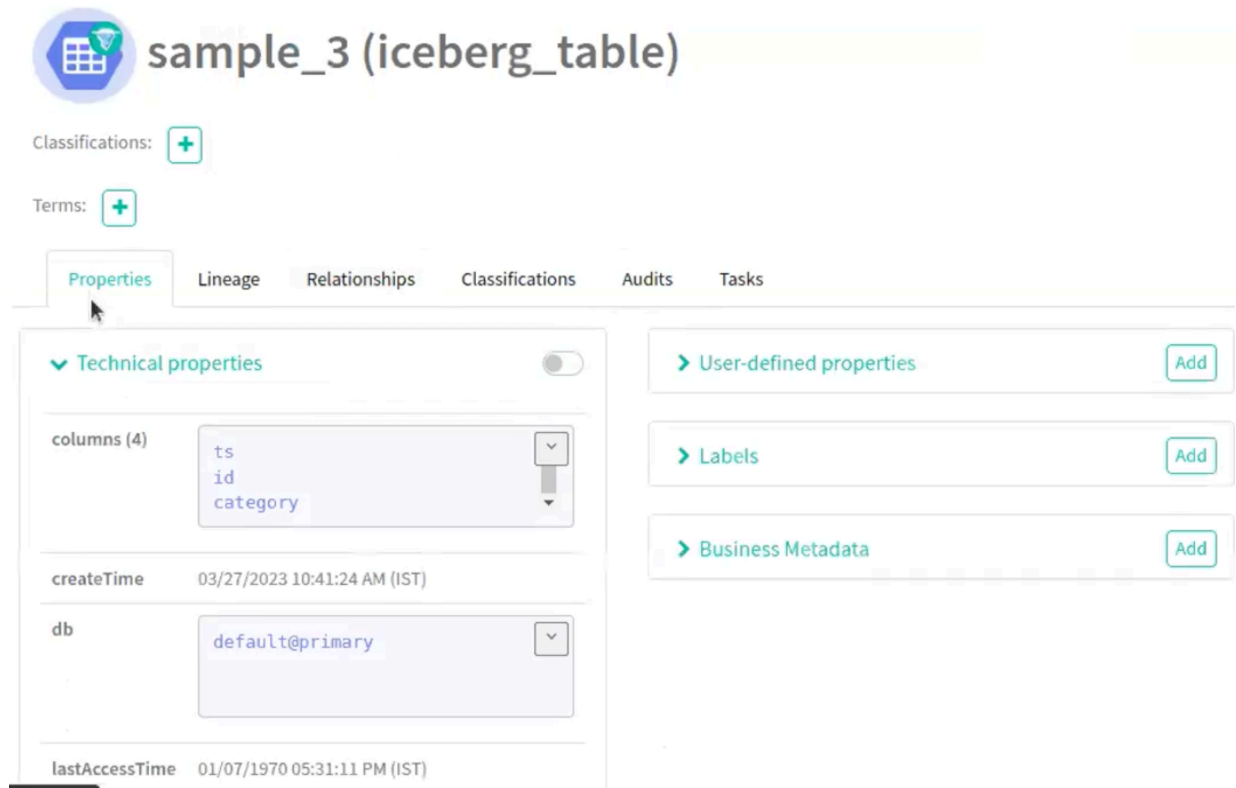
The screenshot displays the Cloudera Atlas UI for an Iceberg table named 'sample\_2'. At the top, there is a navigation bar with a 'Back To Results' link, a search bar for entities, and a user profile for 'admin'. Below this, the table's name 'sample\_2 (iceberg\_table)' is prominently displayed. Underneath, there are sections for 'Classifications' and 'Terms', each with a plus sign icon. A horizontal menu contains tabs for 'Properties', 'Lineage', 'Relationships', 'Classifications', 'Audits', and 'Tasks', with 'Audits' currently selected. The main content area shows an audit log table with columns for 'Users', 'Timestamp', and 'Actions'. A single entry is visible: user 'spark' at timestamp '03/27/2023 10:40:27 AM (IST)' with the action 'Entity Updated'. Below the audit log, the table's name 'Name: sample\_2' is shown. Two expandable sections are present: 'Technical properties' and 'Relationship properties'. The 'Technical properties' section lists fields: 'comment' (N/A), 'createTime' (03/27/2023 10:39:37 AM (IST)), 'lastAccessTime' (01/07/1970 05:29:24 PM (IST)), and 'name' (sample\_2). The 'Relationship properties' section shows 'columns (4)' with a list of 'id', 'data', 'add\_col\_1', and 'add\_col\_2', and a 'db' dropdown menu set to 'default'.

Run the following command in your Spark shell to create a Partition Specification in a new table (sample\_3):

9. `spark.sql("CREATE TABLE spark_catalog.default.sample_3 (id bigint,data string,category string,ts timestamp) USING iceberg PARTITIONED BY (bucket(16, id), days(ts), category)");`

10. Navigate accordingly in the Atlas UI to view the changes.

The following images provide information about Iceberg partition specification process.



createTime	03/27/2023 10:41:24 AM (IST)
db	default@primary
lastAccessTime	01/07/1970 05:31:11 PM (IST)
name	sample_3
owner	spark
parameters	{ owner: "spark", "current-schema": "
partitionSpec (3)	category, id_bucket, ts_day
qualifiedName	default.sample_3@primary
retention	2147483647
sd	default.sample_3@primary_stora

Run the following command in your Spark shell to create a Partition Evolution in a new table (sample\_3):

11. `spark.sql("ALTER TABLE spark_catalog.default.sample_3 ADD PARTITION FIELD years(ts)");`

12. Navigate accordingly in the Atlas UI to view the changes.

The following images provide information about Iceberg partition evolution process.

name	sample_3
owner	spark
parameters	<pre>{   owner: "spark",   previous_metadata_location:</pre>
partitionSpec (4)	<pre>category, ts_year, id_bucket, ts_day</pre>
qualifiedName	default.sample_3@primary
retention	2147483647
sd	<pre>default.sample_3@primary_stora</pre>
tableType	EXTERNAL_TABLE
temporary	false
typeName	iceberg_table

## Using the Hive shell

Using Hive, you can create an Iceberg table followed by using the CTAS command to alter or copy the existing Hive table and its properties into the Iceberg table.

### Before you begin

In this case, you create an external table and alter an existing Hive table to Iceberg table using the Hive engine.

Run the following command in your Hive shell to create an Iceberg table.

### Procedure

1. create external table if not exists hive\_ice\_1 (CountryID int, CountryName string, Capital string, Population string) STORED BY ICEBERG STORED AS PARQUET;

- 2. Navigate accordingly in the Atlas UI to view the changes.

The following images provide information about Iceberg table creation process.

hive\_ice\_1 (iceberg\_table)

Classifications: +

Terms: +

Properties Lineage Relationships Classifications **Audits** Tasks

Users	Timestamp	Actions
hive	03/27/2023 10:44:04 AM (IST)	Entity Updated
hive	03/27/2023 10:44:03 AM (IST)	Entity Created

Showing 2 records From 1 - 25 Page Limit: 25

hive\_ice\_1 (iceberg\_table)

Classifications: +

Terms: +

Properties **Lineage** Relationships Classifications Audits Tasks

Current Entity In Progress Lineage Impact

user/hive/wareho... default.hive\_ice\_... hive\_ice\_1

```
graph LR; A[warehouse] --> B[table]; B --> C[hive_ice_1];
```



## hive\_ice\_1 (iceberg\_table)

Classifications:

Terms:

Properties Lineage Relationships Classifications Audits Tasks

iceberg\_table

guid	23c32c99-ca4a-4a26-8dbb-4ea8c93a4772
typeName	iceberg_table
name	hive_ice_1
qualifiedName	default.hive_ice_1@primary
owner	hive
createTime	1679894043000
status	ACTIVE

Run the following commands in your Hive shell to copy the contents of one table (hive\_ice\_3) to another newly created table (hive\_ice\_4).

- create external table if not exists hive\_ice\_3 (CountryID int, CountryName string, Capital string, Population string) STORED BY ICEBERG STORED AS PARQUET;

- 4. create external table if not exists hive\_ice\_4 STORED BY ICEBERG STORED AS PARQUET as select \* from hive\_ice\_3;

The following images provide information about copying contents from one table to another.



Technical properties

---

clusterName primary

---

endTime 03/27/2023 10:45:17 AM (IST)

---

inputs (2) ▼

```
hdfs://atlas-  
hadoop:9000/user/hive/warehouse/hive_ice_4@primary
```

---

name default.hive\_ice\_4@primary:1679894103000

---

operationType **CREATETABLE\_AS\_SELECT**

---

outputs (2) ▼

```
default.hive_ice_4@primary  
hdfs://atlas-  
hadoop:9000/user/hive/warehouse
```

---

qualifiedName default.hive\_ice\_4@primary:1679894103000

---

queryId

---

queryPlan Not Supported

---

queryText

You can alter an existing Hive table to Iceberg table.

5. create external table if not exists hive\_ice\_5 (CountryID int, CountryName string, Capital string, Population string) STORED AS PARQUET;

6. ALTER TABLE hive\_ice\_5 SET TBLPROPERTIES ('storage\_handler'='org.apache.iceberg.mr.hive.HiveIcebergStorageHandler');

The following images provide information about alter tables operations.

The screenshot displays the Cloudera Data Browser interface for the 'hive\_ice\_5' table. The search bar at the top contains 'hive\_ice\_5'. Under the 'Entities' section, several items are listed, with 'hive\_ice\_5 (hive\_table)' selected and highlighted in blue. Below this, the 'Suggestions' section shows 'hive\_ice\_5'. At the bottom, a navigation bar includes tabs for 'Properties', 'Lineage', 'Relationships', 'Classifications', 'Audits', and 'Tasks'. The 'Lineage' tab is active, showing a diagram with a yellow arrow pointing from 'default.hive\_ice\_5@primary\_storage' to 'hive\_ice\_5'. A legend below the diagram indicates: 'O Current Entity', '⌚ In Progress', '→ Lineage', and '→ Impact'. On the right side, a detailed view for the 'hive\_table' entity is shown, listing the following properties:

Property	Value
guid	5bce52d0-cc48-4d84-bbe8-54db00a22345
typeName	hive_table
name	hive_ice_5
qualifiedName	default.hive_ice_5@primary
owner	hive
createTime	03/27/2023 10:46:10 AM (IST)
status	ACTIVE

The image displays two screenshots of the Cloudera Impala shell interface, showing lineage information for a table.

**Top Screenshot: hive\_process**

- Current Entity: **hive\_process**
- Lineage: **hive\_process**
- Impact: **Impact**
- Lineage Diagram: **/user/hive/wareho...** → **default.hive\_ice\_...** → **hive\_ice\_5**
- Properties:
  - guid: c17a03db-221d-42e5-a6a2-59bc8e58832b
  - typeName: **hive\_process**
  - name: ALERTTABLE\_PROPERTIES:default.hive\_ice\_5@primary:1679894170000->:default.hive\_ice\_5@primary:1679894170000
  - qualifiedName: ALERTTABLE\_PROPERTIES:default.hive\_ice\_5@primary:1679894170000->:default.hive\_ice\_5@primary:1679894170000

**Bottom Screenshot: iceberg\_table**

- Current Entity: **iceberg\_table**
- Lineage: **iceberg\_table**
- Impact: **Impact**
- Lineage Diagram: **/user/hive/wareho...** → **default.hive\_ice\_...** → **hive\_ice\_5**
- Properties:
  - guid: 16d64519-bc28-466e-beb9-b7809e4c2318
  - typeName: iceberg\_table
  - name: **hive\_ice\_5**
  - qualifiedName: default.hive\_ice\_5@primary
  - owner: hive
  - createTime: 1679894170000
  - status: ACTIVE

## Using the Impala shell

Using Impala, you can create an Iceberg table followed by Schema evolution, partition specification, partition evolution and CTAS operation.

### Before you begin

Run the following command in your Impala shell to create a new Iceberg table

### Procedure

1. `CREATE TABLE ice_t (i INT) STORED AS ICEBERG;`

2. Navigate accordingly in the Atlas UI to view the changes.

The following images provide information about Iceberg table creation process.

The screenshot displays the Cloudera Atlas user interface for an Iceberg table named 'ice\_t'. At the top, there is a navigation bar with a 'Back To Results' link, a search bar containing 'ice\_t', and a user profile for 'admin'. Below the search bar, the main header shows the table name 'ice\_t (iceberg\_table)' with a table icon. Underneath, there are sections for 'Classifications' and 'Terms', each with a plus sign icon. A horizontal menu contains tabs for 'Properties', 'Lineage', 'Relationships', 'Classifications', 'Audits', and 'Tasks'. The 'Lineage' tab is selected and highlighted. Below the menu, a status bar indicates 'Current Entity' (with a red circle icon), 'In Progress' (with a clock icon), and a lineage path: 'Lineage' (yellow arrow) followed by 'Impact' (red arrow). To the right of this bar are several utility icons: a refresh icon, a camera icon, a settings icon, a filter icon, and three search icons. The main content area shows a lineage diagram with three nodes connected by yellow arrows. The first node is a blue hexagon with a table icon and the text '/user/hive/wareho...'. The second node is a green hexagon with a gear icon and the text 'default.ice\_t@pri...'. The third node is a blue hexagon with a table icon and the text 'ice\_t', which is circled in red. The diagram illustrates the data flow from the warehouse to the Iceberg table.

Run the following command in your Impala shell to create a scheme evolution:

**3. CREATE TABLE ice\_t\_2 (i INT) STORED AS ICEBERG;**

The screenshot shows the Cloudera Data Browser interface for the table `ice_t_2 (iceberg_table)`. At the top, there is a search bar with the text `ice_t_2` and a user profile for `admin`. Below the search bar, the table name is displayed with a grid icon. There are sections for `Classifications` and `Terms`, each with a plus sign icon. A navigation bar includes `Properties` (selected), `Lineage`, `Relationships`, `Classifications`, `Audits`, and `Tasks`. The `Technical properties` section is expanded, showing a toggle switch, a `columns (1)` field with the value `i`, a `createTime` field with the value `03/27/2023 10:48:21 AM (IST)`, and a `db` field with the value `default@primary`. To the right, there are three sections: `User-defined properties`, `Labels`, and `Business Metadata`, each with an `Add` button.

Run the following command in your Impala shell to add a column to the existing table (`ice_t_2`):



4. `alter table ice_t_2 ADD COLUMNS (add_col_1 string );`



[← Back To Results](#)



Search entities



# ice\_t\_2 (iceberg\_tab

Classifications:



Terms:



Properties

Lineage

Relationships

Classification

## Technical properties

columns (2)

i  
add\_col\_1

createTime

03/27/2023 10:48:21 AM (IST)

db

default@primary

Run the following command in your Impala shell to create a partition specification.

5. CREATE TABLE ice\_part\_spec (s string , b string ) PARTITIONED BY SPEC (truncate(3, s)) STORED AS ICEBERG ;

The screenshot shows the Cloudera Impala Admin console interface. At the top, there is a search bar with the text 'ice\_part\_spec' and a user profile for 'admin'. Below the search bar, the main heading is 'ice\_part\_spec (iceberg\_table)'. There are tabs for 'Properties', 'Lineage', 'Relationships', 'Classifications', 'Audits', and 'Tasks'. The 'Audits' tab is selected, showing a table with columns 'Users', 'Timestamp', and 'Actions'. One record is visible, showing 'impala' as the user, '03/27/2023 10:49:28 AM (IST)' as the timestamp, and 'Entity Created' as the action. Below the table, it says 'Showing 1 records From 1 - 25' and 'Page Limit: 25'. Below the audit log, there is a detailed view of the table's properties. The 'name' is 'ice\_part\_spec', 'owner' is 'impala', 'parameters' is a JSON object with 'external.table.purge': 'TRUE', 'partitionSpec (1)' is 's.trunc', 'qualifiedName' is 'default.ice\_part\_spec@primary', 'retention' is '2147483647', 'sd' is 'default.ice\_part\_spec@primary\_orage', 'tableType' is 'EXTERNAL\_TABLE', 'temporary' is 'false', and 'typeName' is 'iceberg\_table'.

Back To Results  admin

## ice\_part\_spec (iceberg\_table)

Classifications:

Terms:

Properties Lineage Relationships Classifications **Audits** Tasks

Users	Timestamp	Actions
impala	03/27/2023 10:49:28 AM (IST)	Entity Created

Showing 1 records From 1 - 25 Page Limit: 25

Back To Results  admin

lastAccessTime 01/07/1970 05:39:15 PM (IST)

name **ice\_part\_spec**

owner impala

parameters 

```
{
  "external.table.purge":
  "TRUE",
}
```

partitionSpec (1) **s.trunc**

qualifiedName default.ice\_part\_spec@primary

retention 2147483647

sd default.ice\_part\_spec@primary\_orage

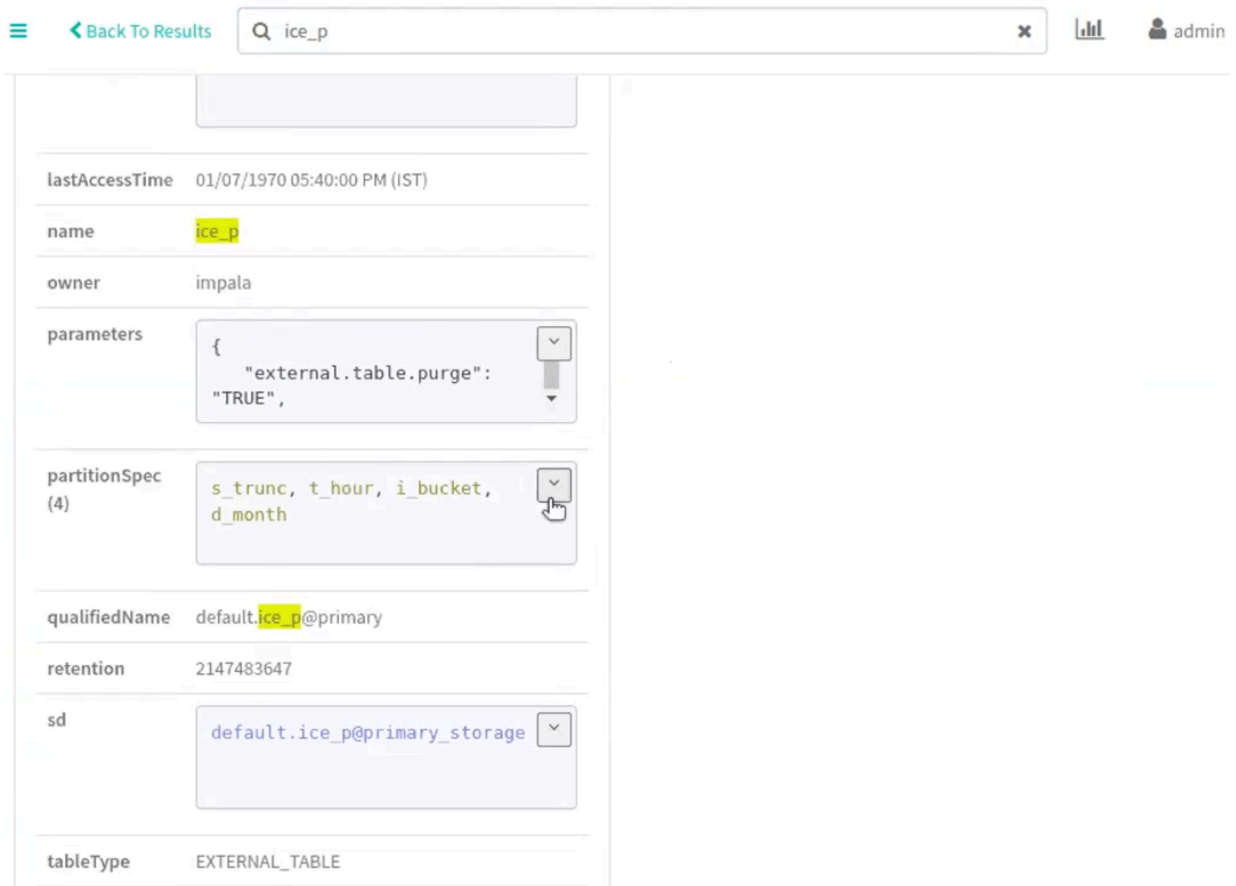
tableType EXTERNAL\_TABLE

temporary false

typeName iceberg\_table

Run the following command in your Impala shell to create a partition evolution.

6. CREATE TABLE ice\_p (i INT, d DATE, s STRING, t TIMESTAMP) PARTITIONED BY SPEC (BUCKET(5, i), MONTH(d), TRUNCATE(3, s), HOUR(t)) STORED AS ICEBERG;



The screenshot shows the Cloudera Impala web interface. At the top, there is a navigation bar with a hamburger menu, a "Back To Results" link, a search bar containing "ice\_p", and a user profile icon labeled "admin". Below the navigation bar, the details for the table "ice\_p" are displayed in a card-like format. The details include:

- lastAccessTime**: 01/07/1970 05:40:00 PM (IST)
- name**: ice\_p
- owner**: impala
- parameters**: {"external.table.purge": "TRUE", ...}
- partitionSpec (4)**: s\_trunc, t\_hour, i\_bucket, d\_month
- qualifiedName**: default.ice\_p@primary
- retention**: 2147483647
- sd**: default.ice\_p@primary\_storage
- tableType**: EXTERNAL\_TABLE

Run the following command in your Impala shell to modify the partition specification

7. ALTER TABLE ice\_p SET PARTITION SPEC (VOID(i), VOID(d), TRUNCATE(3, s), HOUR(t), i);

[← Back To Results](#)

Search entities

lastAccessTime 01/07/1970 05:40:00 PM (IST)

name ice\_p

owner impala

parameters

```
{  
  previous_metadata_location:  
  "hdfs://atlas-
```

partitionSpec  
(5)

```
d_null, s_trunc, t_hour, i_nul  
i
```

qualifiedName default.ice\_p@primary

retention 2147483647

sd

```
default.ice_p@primary_storage
```

Run the following commands in your Impala shell to create the contents of one table (ice\_t\_3) to another table (ice\_t\_4).

**8.** CREATE TABLE ice\_t\_3 (i INT) STORED AS ICEBERG;

9. CREATE TABLE ice\_t\_4 STORED AS ICEBERG as select \* from ice\_t\_3;



☰ < Back To Results






partitionSpec (5)

✓ impala

Name: ice\_p

Q ice\_t\_4

Entities

-  /user/hive/warehouse/ice\_t\_4
-  default.ice\_t\_4@primary:-100
-  **ice\_t\_4 (iceberg\_table)**
-  default.ice\_t\_4@primary\_stor
-  default.ice\_t\_4@primary:1679

Suggestions

**ice\_t\_4**

✓ Technical properties

comment	N/
createTime	03/27/2023 10:50:13 AM (IS
lastAccessTime	01/07/1970 05:40:00 PM (IS
name	ice_

← Back To Results  admin

# ice\_t\_4 (iceberg\_table)

Classifications: +  
Terms: +

Properties Lineage Relationships Classifications Audits Tasks

○ Current Entity ⌛ In Progress → Lineage → Impact



The process is completed.