Cloudera Runtime 7.3.1

Atlas Search

Date published: 2020-07-28
Date modified: 2024-12-10

https://docs.cloudera.com/

Legal Notice

Contents

Using Basic search

Search using an entity type in Basic Search.

With **Basic Search**, Atlas returns all of the entities of the type you select.

There are many ways you can define a **Basic Search**. Setting a value in more than one search field builds a logical AND condition for the search. To repeat the same search, click the Refresh button. 

Search By Type

- Choose an entity type to limit the search.
- Choose _ALL_ENTITY_TYPES to apply an attribute filter across all entity types.

Search By Type and specify attribute values using the Entity Attribute Filter 

The **Attribute Filter** dialog box lists all the attributes that correspond to the selected entity type, including:

- Technical attributes specific to the entity type
- System attributes, including classifications, labels, and user-defined properties
- Business Metadata attributes
- Terms

Note: When searching on classifications in the Search By Type filter, use "contains" rather than "=" for the filter operator. If there is more than one classification assigned to an entity, "contains" matches a single classification name; equal only matches the entire list of classifications names.

Search By Classification

- Choose an existing classification; the search returns all entities that have that classification assigned to them.
- Choose _ALL_CLASSIFICATION_TYPES to apply an attribute filter across all classifications.
- Choose _CLASSIFIED or _NOT_CLASSIFIED with an entity type selected to find entities of that type with any or no classifications assigned.

Search By Classification and specify attribute values using the Filter 

The **Attribute Filter** dialog box lists all the attributes for the selected classification; set a value to one or more attributes to define the search. You can choose to match partial strings using the "contains", "begins with", and "ends with" operators.

Search by Term

Choose an existing glossary term. You can enter the first few letters to select a term from a list of matching terms. This filter is case-sensitive.

Search by Text

Search on string values for technical, system, Business Metadata, and classification attribute values. Labels and terms are also included. This search is the same as the Free-Text search; note that when you enter text in the Free-Text search box, it fills in this Search By Text field also.

You can also save these searches when they are useful to run more than once.

Related Information

Using Free-text Search
Searching for entities using Business Metadata attributes
Searching for entities using terms
Searching for entities using classifications
Saving searches
Apache Atlas metadata attributes
Using Search filters

Basic search enhancement

While performing basic search in Atlas, you can exclude header attributes of entities from the response to reduce latency.

The Basic Search feature in Atlas has AtlasEntityHeader data type of each entity in the response.

The AtlasEntityHeader data type has multiple attributes including classification and terms. AtlasEntityHeader requests the Janusgraph database to provide the information for each attribute. This process can be time consuming increasing the response latency.

To overcome this situation, you can add a flag to exclude generic attributes and add only the selected attributes from the attributes field in the response.

In the request payload, including the following improves the search experience:

- Attributes having entityTypes
- excludeHeaderAttributes=true
- Valid entity attributes (not relationship) in the attributes field

Note: The excludeHeaderAttributes attribute overrides other attributes' fields such as the includeClassificatio nAttributes in the request payload.

An example payload request:

Request

```
{
"excludeDeletedEntities": true,
"includeSubClassifications": true,
"includeSubTypes": true,
"includeClassificationAttributes": true,
"limit": 25,
"offset": 0,
"typeName": "hdfs_path",
"attributes": ["path", "name"],
"excludeHeaderAttributes": "true"
}
```

Response

```
{
"queryType": "BASIC",
"searchParameters":
{ "typeName": "hdfs_path", "excludeDeletedEntities": true, "includeClassific
ationAttributes": true, "includeSubTypes": true, "includeSubClassifications"
: true, "limit": 25, "offset": 0, "attributes": ["path", "name"] }
,
"attributes": {
```

```
"name": ["path", "name"],
"values": [
["/data/warehouse/customer", "customer"],
["/data/warehouse/sales", "sales"]
]
},
"approximateCount": 2
}
```

Using Relationship Search

Entities in Atlas can be searched based on the relationships that describe various metadata between a couple of entity end-points.

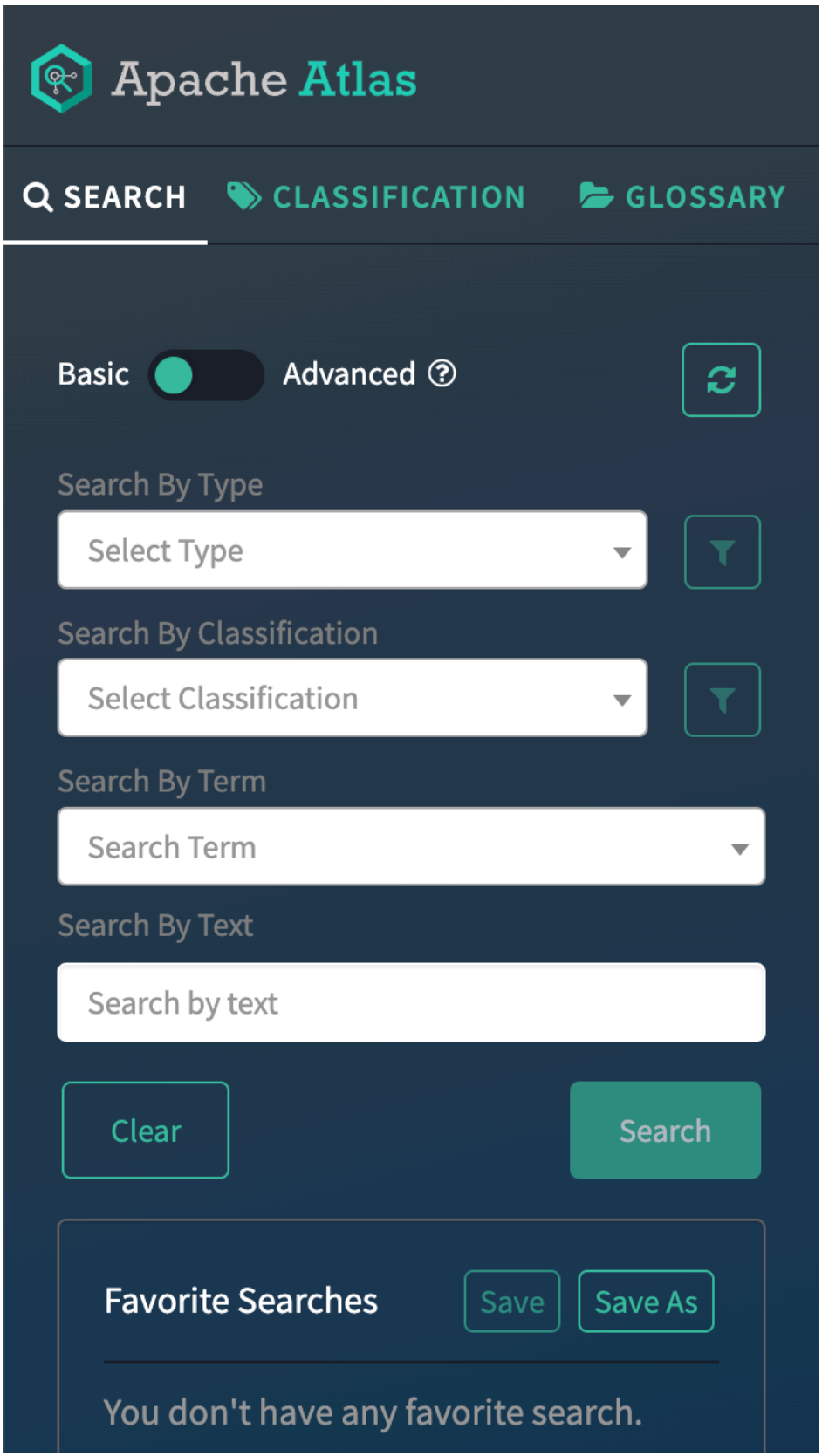
Important: By default, the Relationship Search feature is disabled in Atlas to reduce the start and restart time of the service.

As an example, a relationship between entities hive_table and hive_db can be attributed or defined as hive_table_db, which has a standalone metadata that can be added as an attribute to this format. By searching for hive_table_db, you can retrieve the relationship between hive_table and hive_db entities. This enhancement ensures that those relationships which are tied to the entities and that match the filter criteria on attributes of the relationships, can be searched.

Enabling relationship search

You can enable Relationship Search in Cloudera Manager under Atlas Server Advanced Configuration Snippet for conf/atlas-application.properties by setting the atlas.relationship.search.enabled property to true.

Until the property is enabled, Relationship Search is not visible in the UI:


Apache Atlas
SEARCH
CLASSIFICATION
GLOSSARY
Basic
Advanced
Search By Type
Select Type
Search By Classification
Select Classification
Search By Term
Search Term
Search By Text
Search by text
Clear
Search
Favorite Searches
Save
Save As
You don't have any favorite search.

Configuring Relationship Search

For the entities to be searchable in the relationship definition model, the attributes must be added and marked as indexable before starting the Atlas service.

Use the POST API typedef for the following configuration:

As an example of the configuration setup is as follows:

```
{
“attributeDefs”: [{
“Name”: “edge_property1”,
“isIndexable”: true
}]
```

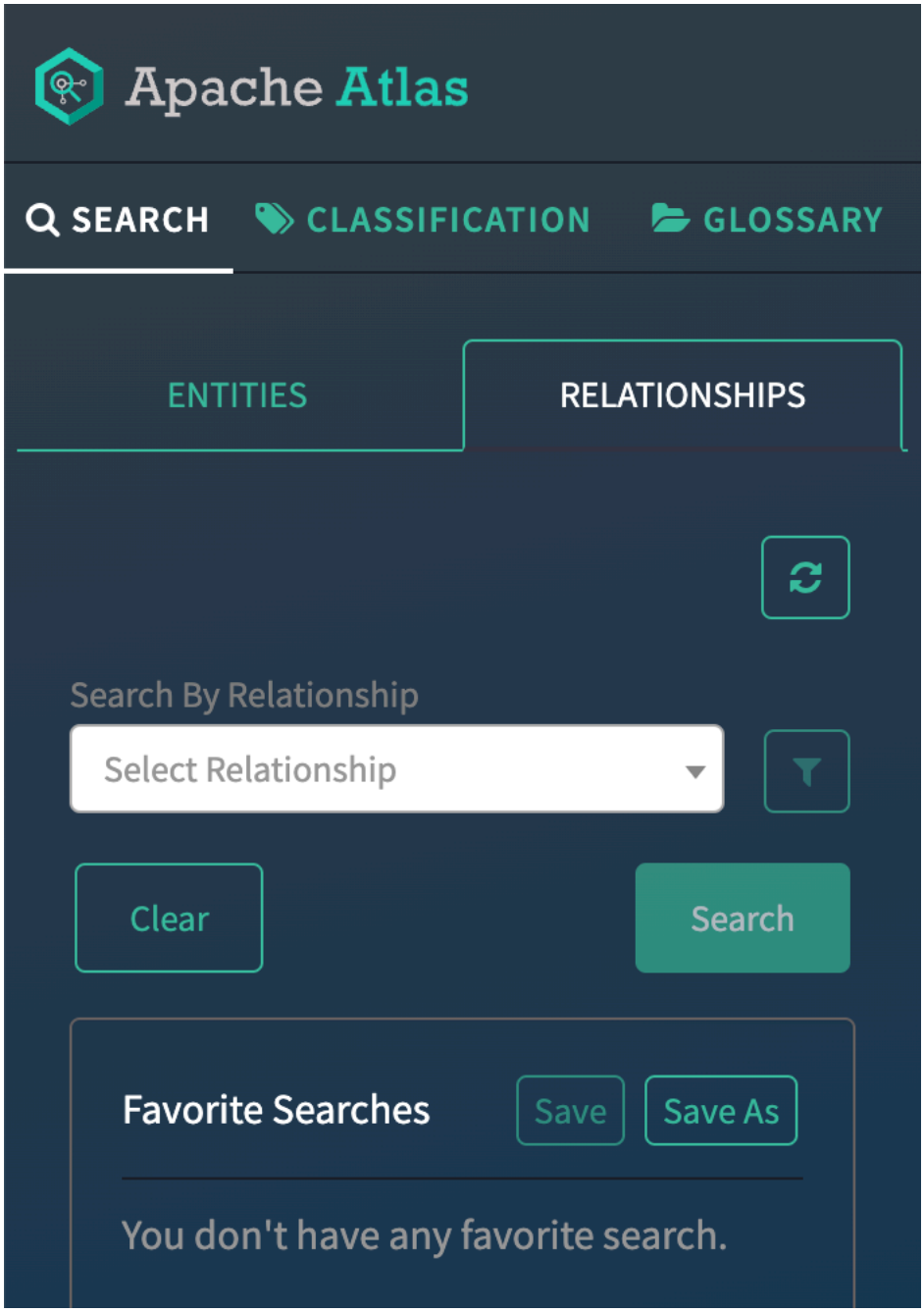
Note:

The following API calls are enabled if the isIndexable flag is set to true:

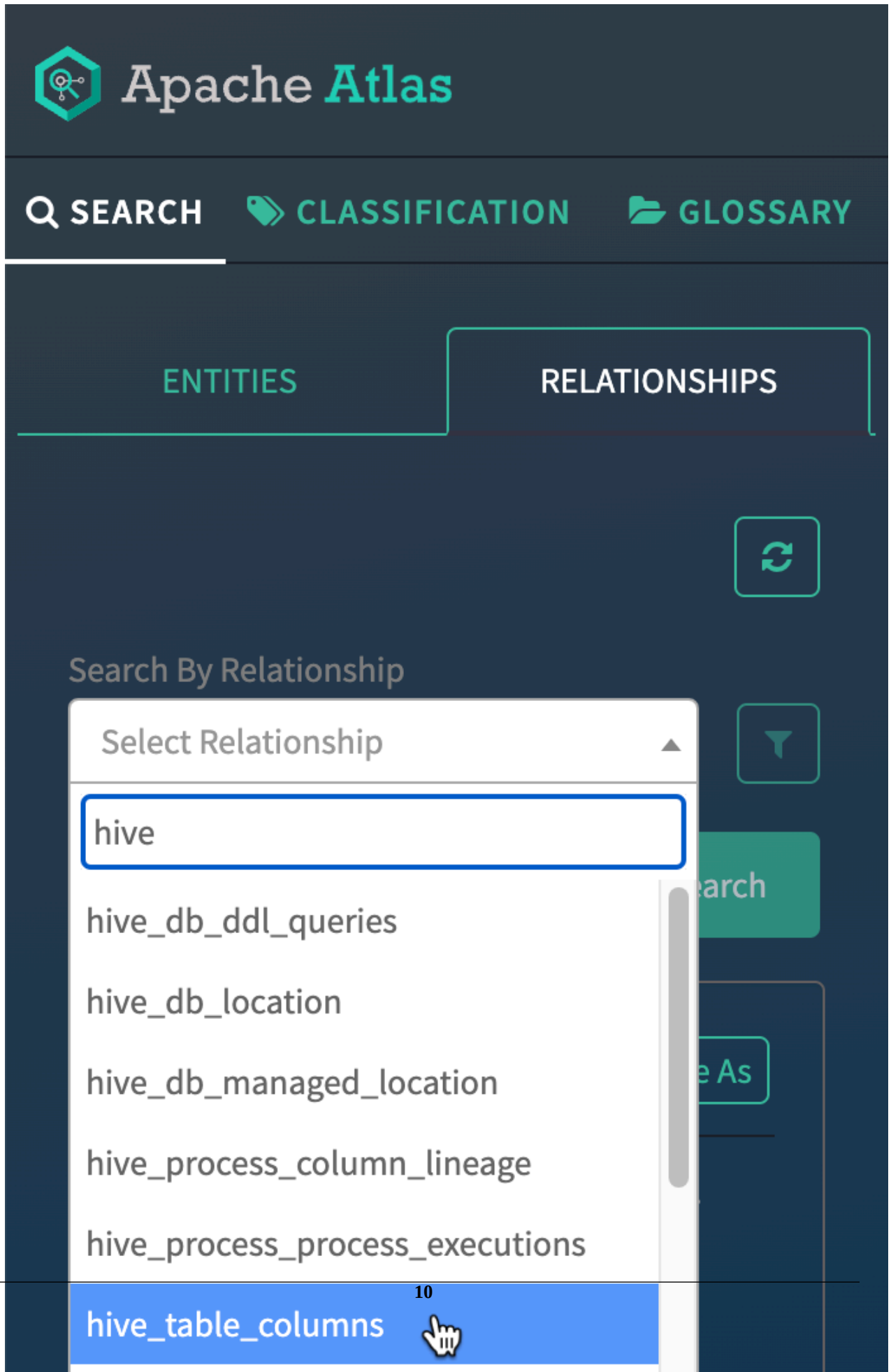
- GET/POST /v2/search/relations
- GET /v2/search/relationship

These API calls are not available by default.

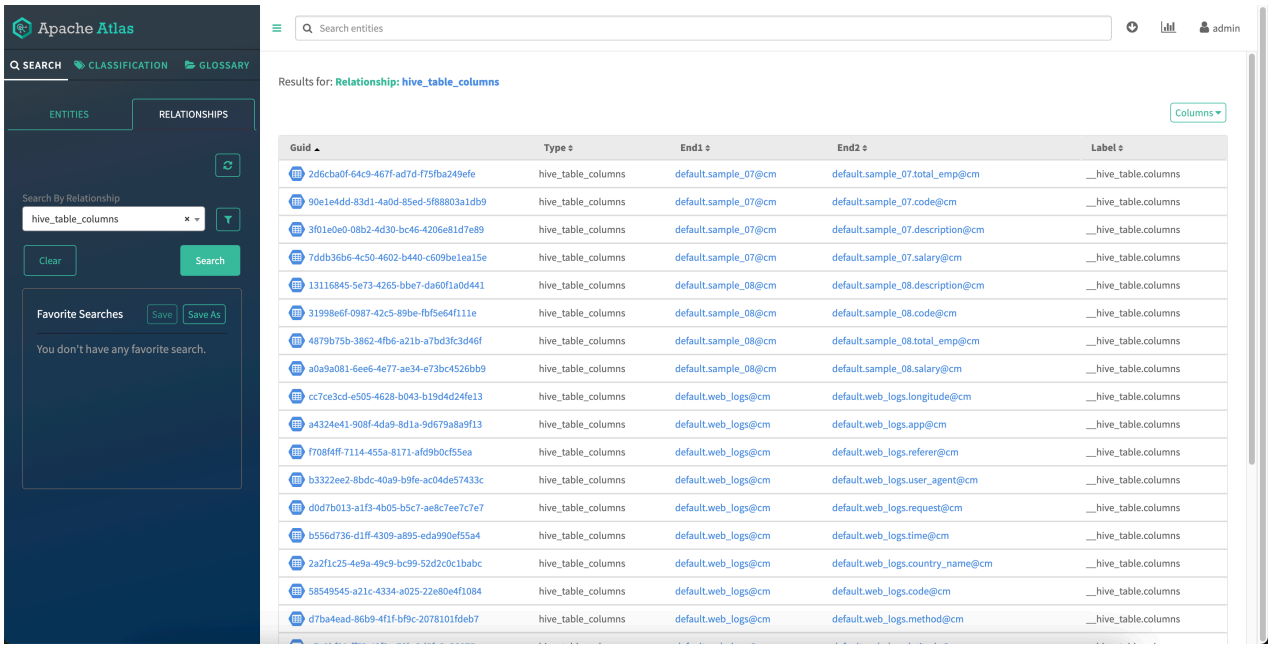
In Atlas, once the atlas.relationship.search.enabled property is set to true, you can switch between Entity Search and Relationship Search:



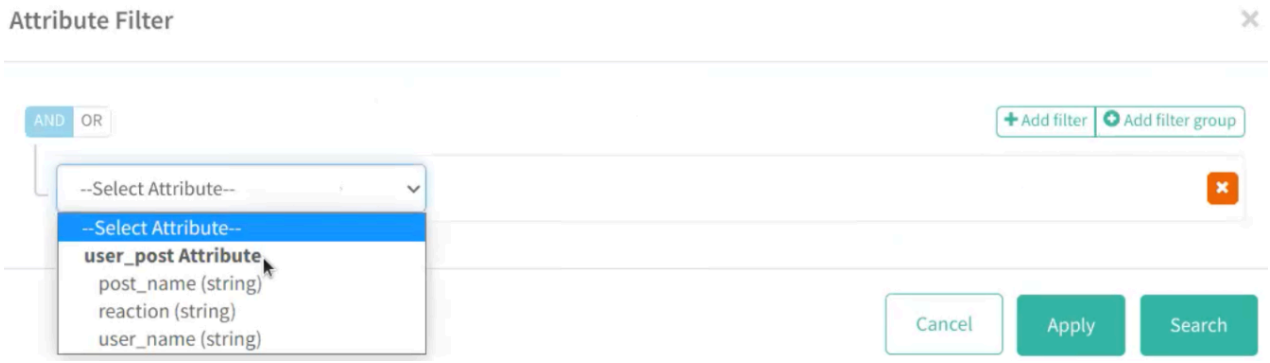
You can search for the relationship using the drop-down list to select a relationship between entities and explore them in Atlas.


Apache Atlas
SEARCH
CLASSIFICATION
GLOSSARY
ENTITIES
RELATIONSHIPS
Search By Relationship
Select Relationship
hive
hive_db_ddl_queries
hive_db_location
hive_db_managed_location
hive_process_column_lineage
hive_process_process_executions
hive_table_columns

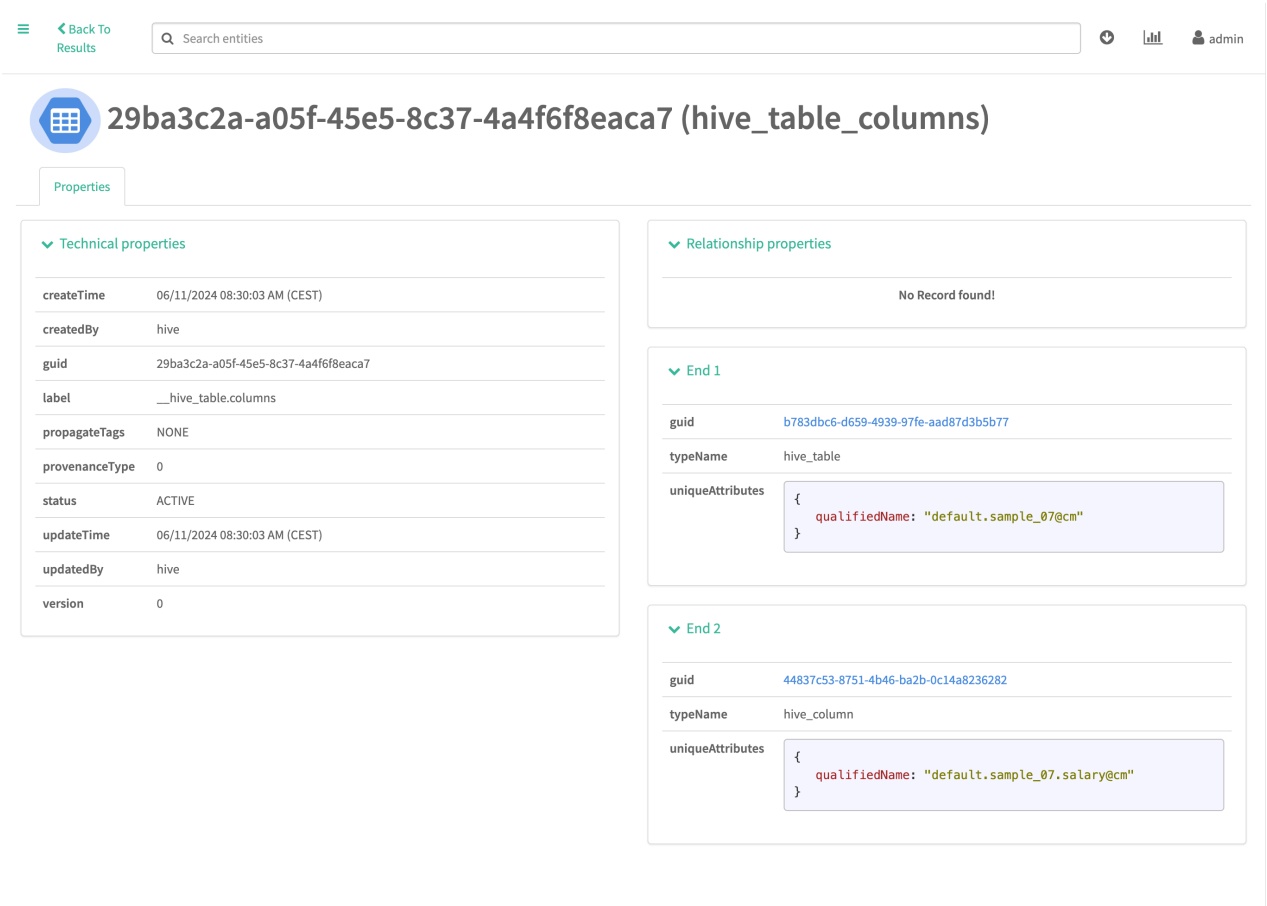
The list of entities with the type hive_table_columns contains additional information about the connected entities.



You can add specific filters to search for relationship entities.



In the example image, end 1 and end 2 are the entities with which the relationship is created.



By clicking on each search result, you can navigate to the details page to view additional details and pertinent information.

Using Search filters

The Basic Search panel includes filter icons that allow you to search for entities based on one or more attribute values.

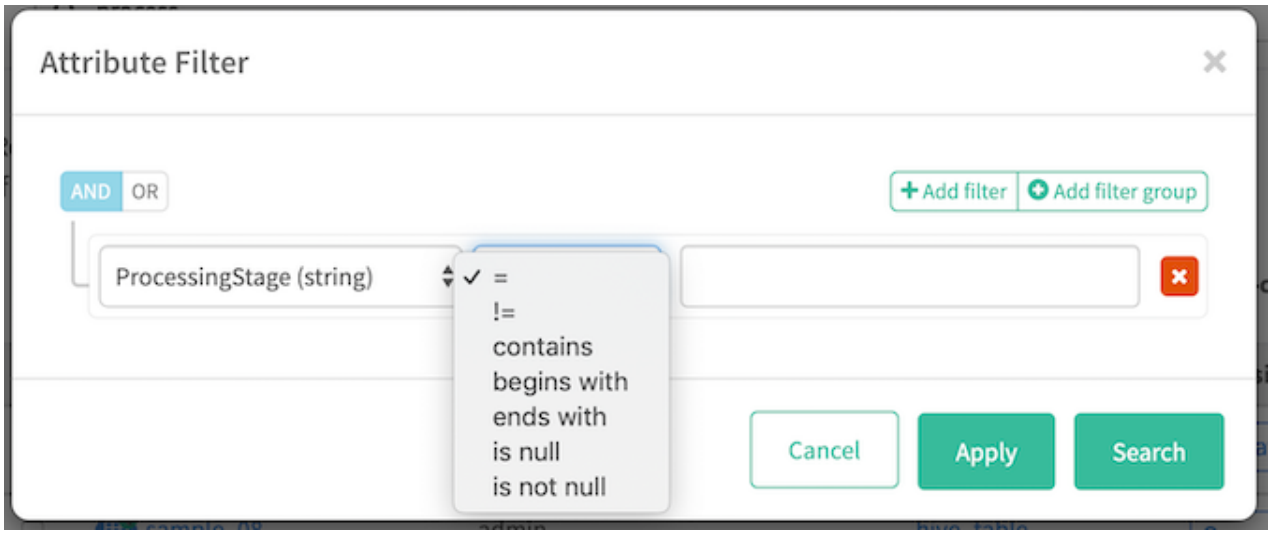
In a filter row, the attribute data type determines which of the following operators can be used to define your search criteria:

Strings	Dates	Enumerations Boolean	Numerics
=		=	=
!=		!=	!=
	>	>	>
	<	<	<
is null	is null	is null	is null
is not null	is not null	is not null	is not null
contains			
begins with			

Strings	Dates	Enumerations Boolean	Numerics
ends with			

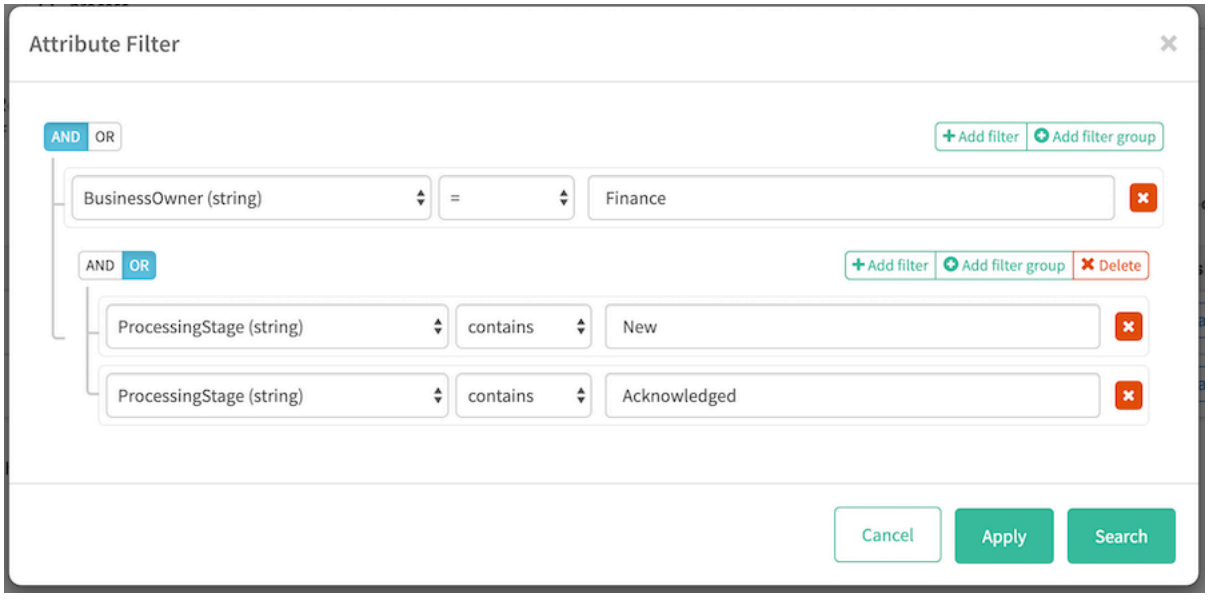
All classification attributes are string values; numerics include byte, short, int, float, double, and long attribute data types.

Note: If the attribute you are searching for could include multiple values, use "contains" rather than "=" to make sure the search finds the individual value out of the list.

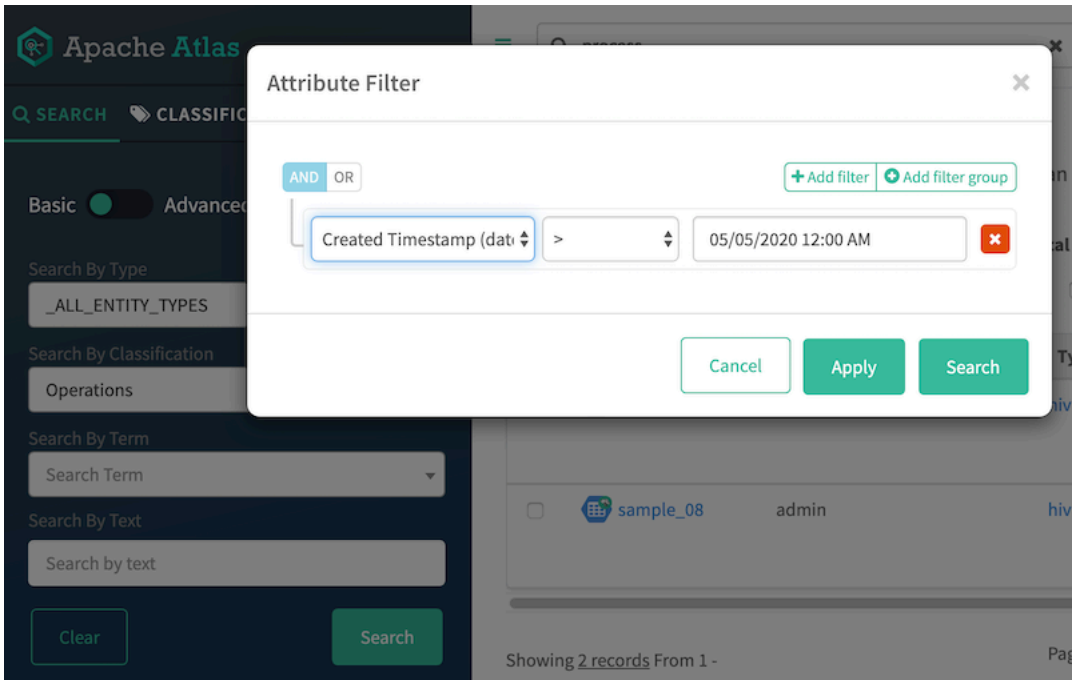


To search on values for more than one attribute, add another filter row to the search filter (click Add filter). The search can find entities matching either filter criteria (logical OR) or matching both criteria (logical AND). Set the logic using the AND / OR buttons at the top-left of the filter rows.

You can combine logical AND and OR criteria using filter groups. The logic is the same within a filter group; use more than one filter group to produce both AND and OR logic. For example, the following Classification attribute filter searches for entities that are at "new" or "acknowledged" stages in their processing and are owned by the Finance business team.



If you wanted to further limit the search results to the entities that were created in Atlas in the last 24 hours, you would open the attribute filter for Search by Entity Type and set the system attribute "Created Timestamp" less than 24 hours. To open the Search by Entity Type filter, you would need to select an entity type or "_ALL_ENTITY_TYPES".



Ability to download search results from Atlas UI

Atlas supports improved search results capabilities. You can download search results for both basic and advanced search options. You can configure the parameters to specify a directory path and configure automatic cleaning up of files to manage space efficiently.

You can use the new Download option to capture the search results of both Basic and Advanced (DSL) searches, in CSV file format. The CSV files are stored on the local file system. This download option queues up the search results and downloads the files in the following format: User type_type_of_search_timestamp. For example: admin_DSL_2023-04-29_04-56.csv indicates that the file contains results of search performed using the advanced search option.

Atlas employs the API path to route the requests to queue up the search results and later convert them to CSV files. The following are commonly used for the search and download operations.

- /atlas/v2/search/<basic/dsl>/download/create_file
- /atlas/v2/search/download/status - Status API
- /atlas/v2/search/download/{filename} - Download request API

Search results configuration

You can configure the search result size The default value of search result size is 10000 and only 50 simultaneous requests can be done. You can configure the search request using the atlas.download.search.max.pending.tasks property.

You can configure the parameters in such a way that the downloaded CSV files are available in a specified directory path for a fixed time period and they are automatically cleaned up to free up the storage space. You can search for specific type definitions in Atlas and download them as appropriate.

The following APIs can be used to manage the download options and can be included in the **Configuration** tab in the Cloudera Manager instance under Atlas Server Advanced Configuration Snippet (Safety Valve) for conf/atlas-application.properties.

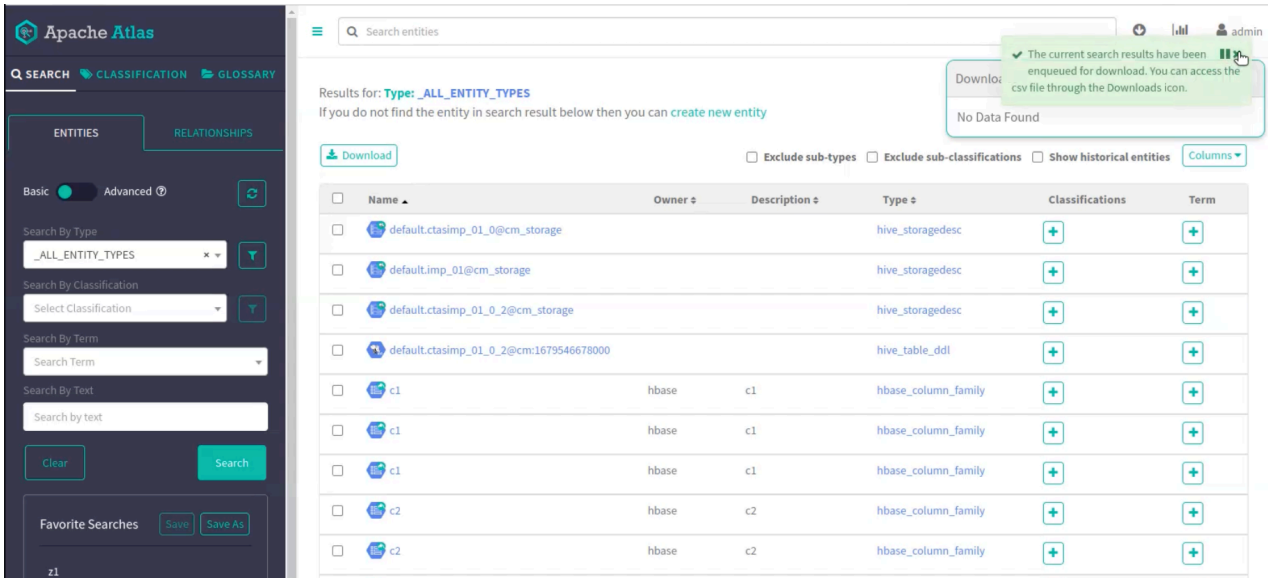
- atlas.download.search.dir.path - Use this parameter to define the path for storing the downloaded CSVs.
- atlas.download.search.file.expiry.millis - Use this parameter to set the time limit for the downloaded file to live in the storage. As an example: 180000 milliseconds # 3 minutes.

Note: For every logged in user, a separate directory is created automatically under the path defined by the atlas.download.search.dir.path property and the user's requested search files are stored in that directory. Users can only download files created by them using UI / API.

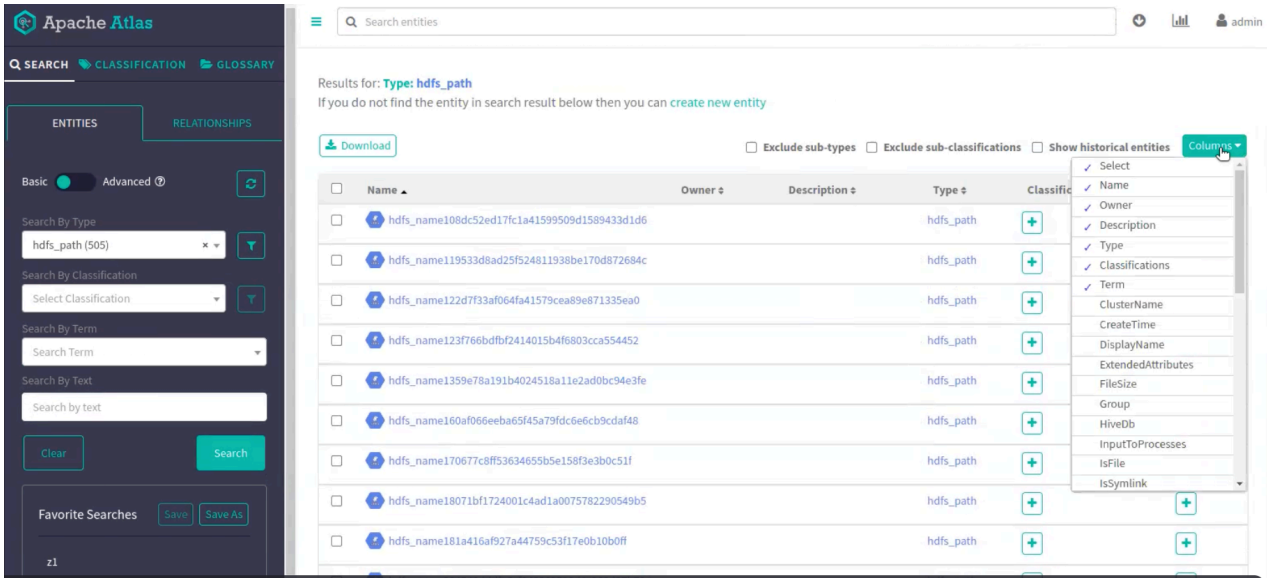
How to download results using Basic and Advanced search options

Using the Basic search option you can search and download the results. You must select the type of query and search for the results and later click the Download option.

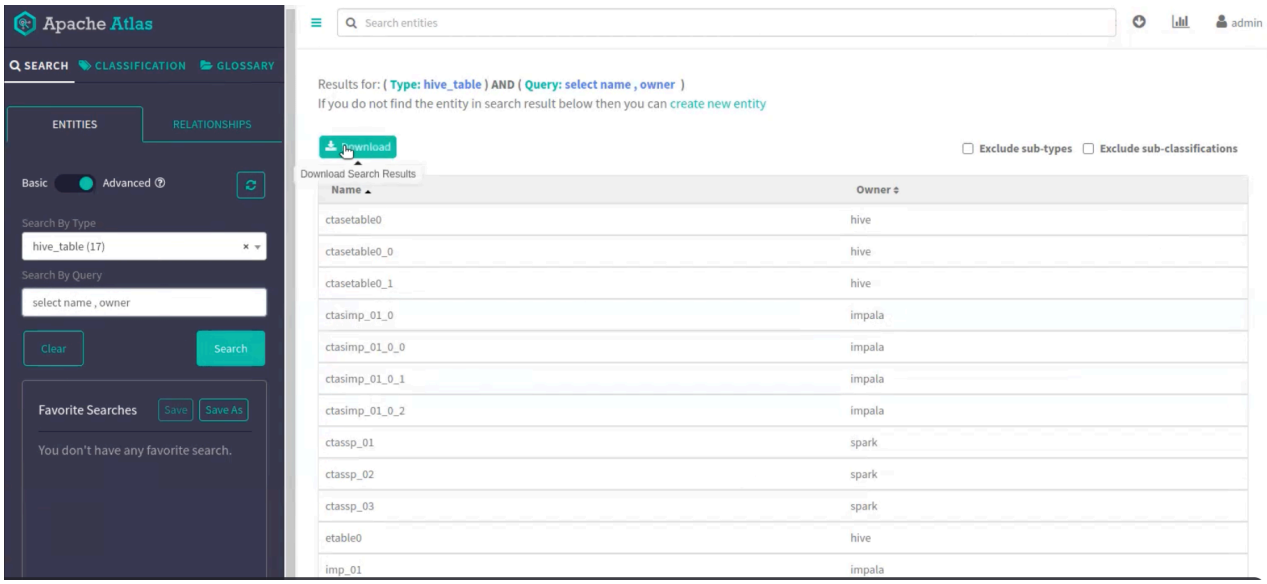
The search query queues up the results. Click to view the resultant queue.

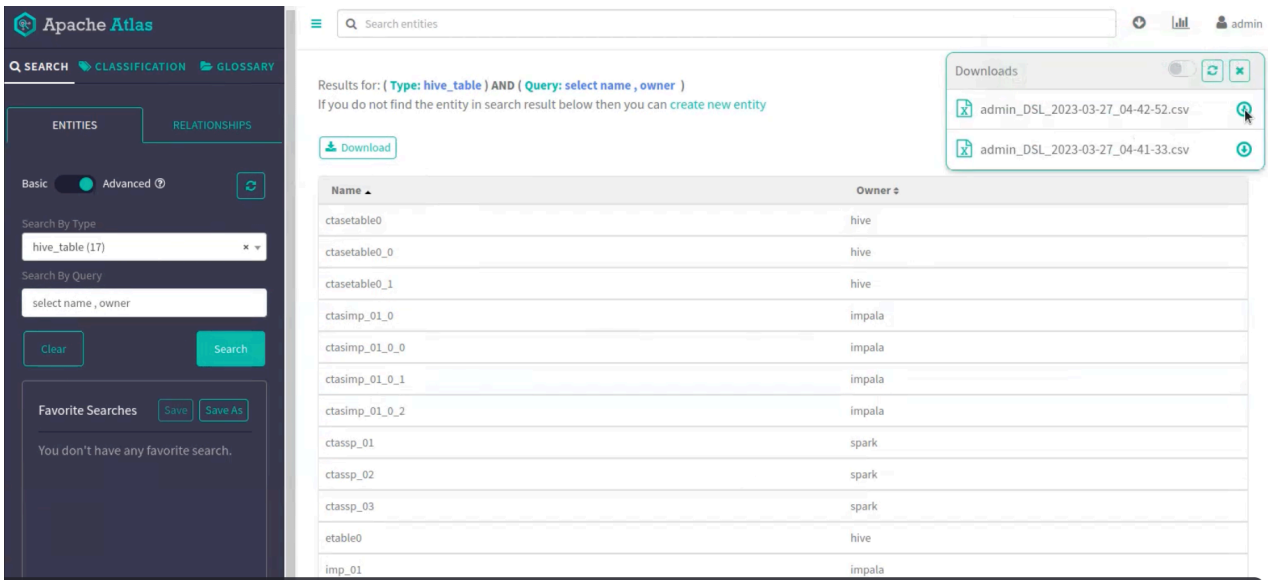


You can expand the scope of the downloaded search results by using the Column drop-down menu to select the columns that you want to get include in the CSV files.



You can search using the Advanced option and download the search results. For example, you can search by using the name and owner criteria. Click Download and follow the same process as described while performing the Basic search operations.

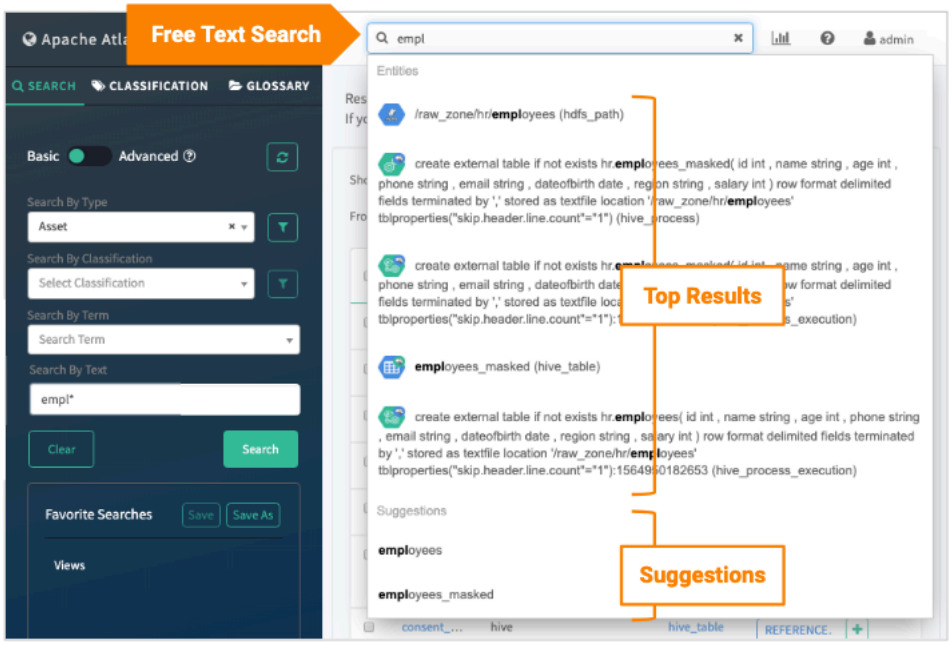




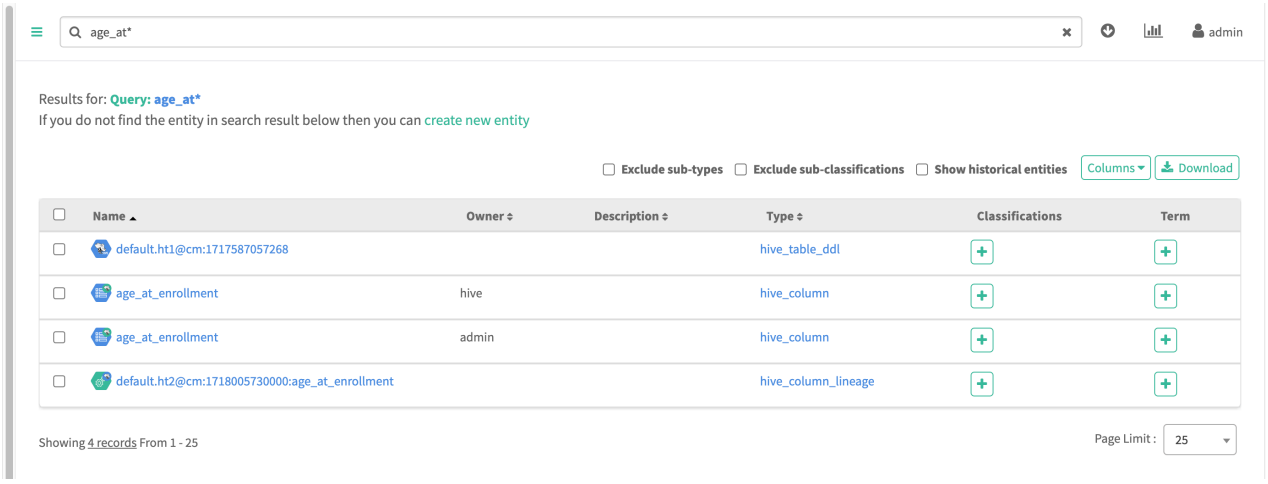
Using Free-text Search

Apache Atlas builds a ranked index of metadata values so you can search for values across all metadata.

The search field in the top of the dashboard lets you search for entities, classifications, or terms by entering any full or partial text to match any entity metadata values. Atlas searches all metadata fields that have string data types, so you can use this search tool to find entities by their labels, descriptions, locations, or other metadata.



Searches are case insensitive. You can add the asterisk (*) wildcard to the search term's start or end or terms to find partial strings anywhere they occur in the metadata value.



The following single and double characters have special meaning:

```
+ - && || ! ( ) { } [ ] ^ " ~ * ? : \
```

If your search string includes one of these characters, surround the string in double quotation marks or escape the special character with a backslash.

You can see that the search terms you use in the search at the top of the dashboard are also inserted into the free text search field in the left Search panel: you can combine the free text search with other selections to narrow the search results. The combination acts as an “AND” with other search criteria.

Search result ordering: The search results are ranked by how well they match the search terms, with entities that match on more than one metadata value being ranked higher.

Different metadata have different scores, where the highest scoring metadata fields are entity names, including Kafka topic names. Descriptions, users/owners, query text, and comments rank next. Locations, namespaces, domains, etc. come next. Search results are not ordered in any specific way among results that have the same search ranking.

Suggestions: As you enter your search text, you see the five highest-ranked matching items and as many as five suggestions.

The matching items are ranked in the same way as the general search results, case-sensitive (at the moment) terms that “contain” the search terms; If there are more than five search results with the same search ranking, the five shown are randomly ordered from the highest scoring results.

The suggested items are chosen from search results that match with a “starts with” behavior.

Limitations

The suggestions under the Free-text Search field work independently from the search results. Also, certain languages have different behaviors when they are used for searching:

Language or Alphabet	Limitations

• English and languages with Latin alphabets
• Chinese
• Japanese
• Japanese Kana
	• Searching entity names (indexable attributes)
	  • Suggestions are provided for multicharacter searches.
	  • Partial search terms return results.
	• Searching descriptions or comments and other (non-indexable attributes)
	  • Suggestions are not provided for multiword searches.
	  • Partial search terms return results.

Language or Alphabet	Limitations

• Korean	• Searching entity names (indexable attributes)
	  • Suggestions are provided for multicharacter searches.
	  • Partial search terms do not return results. Only partial suggestions return search results.
	• Searching descriptions or comments and other (non-indexable attributes)
	  • Suggestions are provided for multicharacter searches.
	  • Partial search terms do not return results. Only partial suggestions return search results.
	Note: For Korean, you need to add manually the asterisk character to enable partial search results. However, suggestions are provided for partial searches if your search matches the start of an existing term.

Enhancements with search query

When you perform search operations in Atlas, note some of the new changes that are in effect.

First use case

Performing freetext / quick search does not require pre-fixing the attribute name in the search text query.

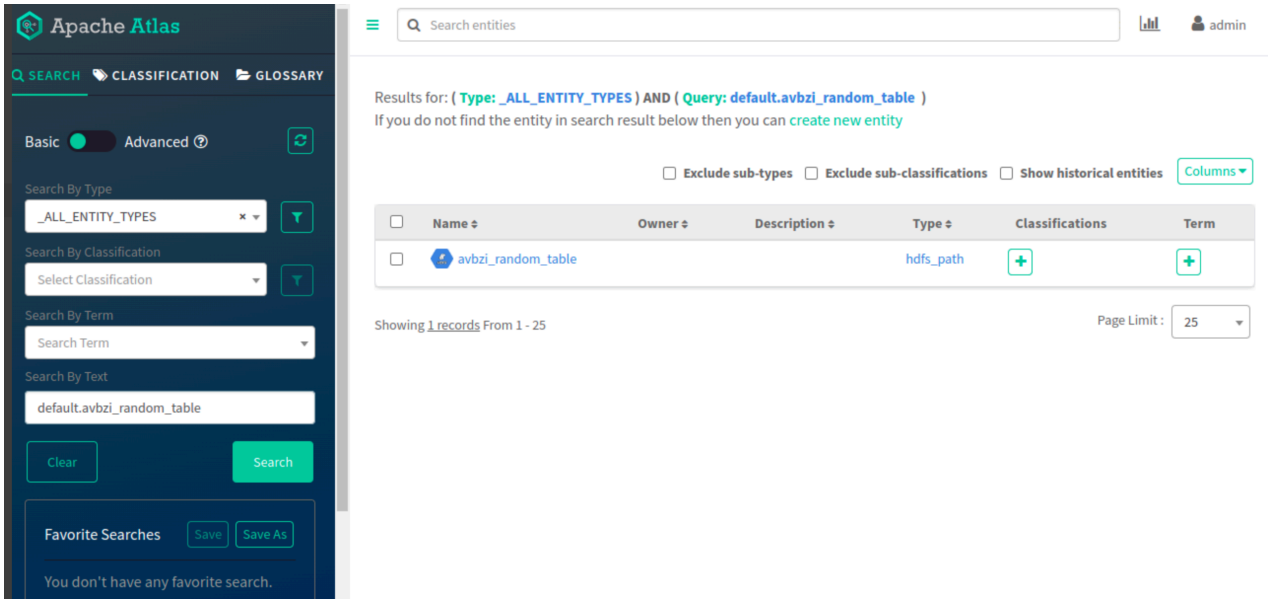
You can directly search for the entities by searching with its value.

For example:

To search entity having:

qualifiedName="default.avbzi_random_table@cm"

You can directly add "default.avbzi_random_table@cm" in the search bar.



Second use case

While using the basic / quick search in Atlas, the characters which are not alphabetic and numeric are considered as special characters except for the following characters:

_, ., :, and ‘

Note: When using ., :, and ‘ characters, if the prefix or suffix does not contain alphabet character, it is considered a special character.

For example:

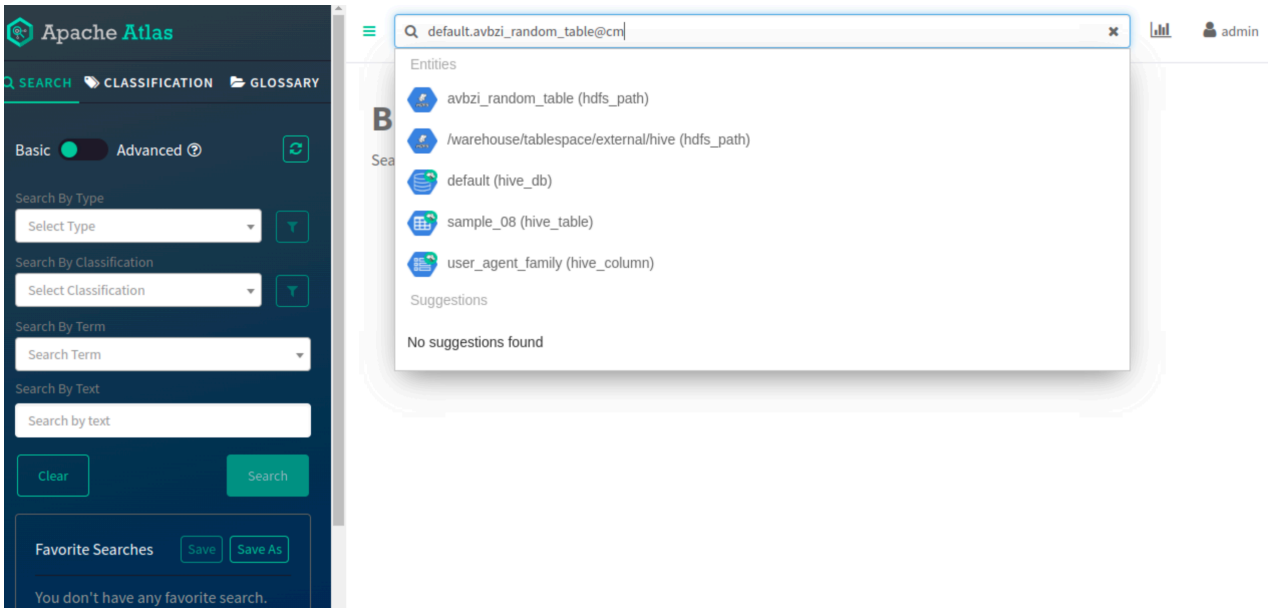
default.1marketing > . will be considered as a special character because suffix is numeric not alphabetic

default.marketing > . will not be considered as a special character because prefix and suffix are alphabetic

Additionally, while performing the search operation, when the search string has special characters, SOLR tokenizes the string enabling the search result query to provide OR condition of each tokenized string.

For example: if search string is provided as default.avbzi_random_table@cm

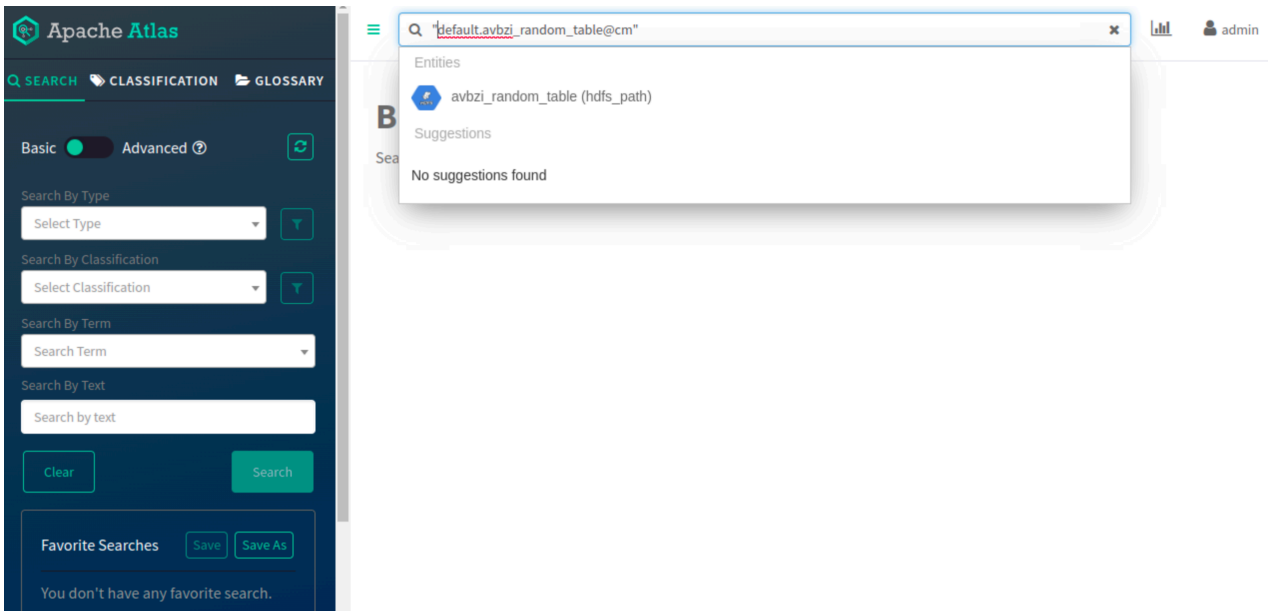
results in, picking up all the matching documents or attributes, where, attribute contains default.avbzi_random_table OR cm.



If a string is enclosed with double quotes, SOLR does not tokenize and behave as a single string.

For example: if search string is like "default.avbzi_random_table@cm"

results in, picking up all the matching documents or attributes, where, attribute contains default.avbzi_random_table AND cm.

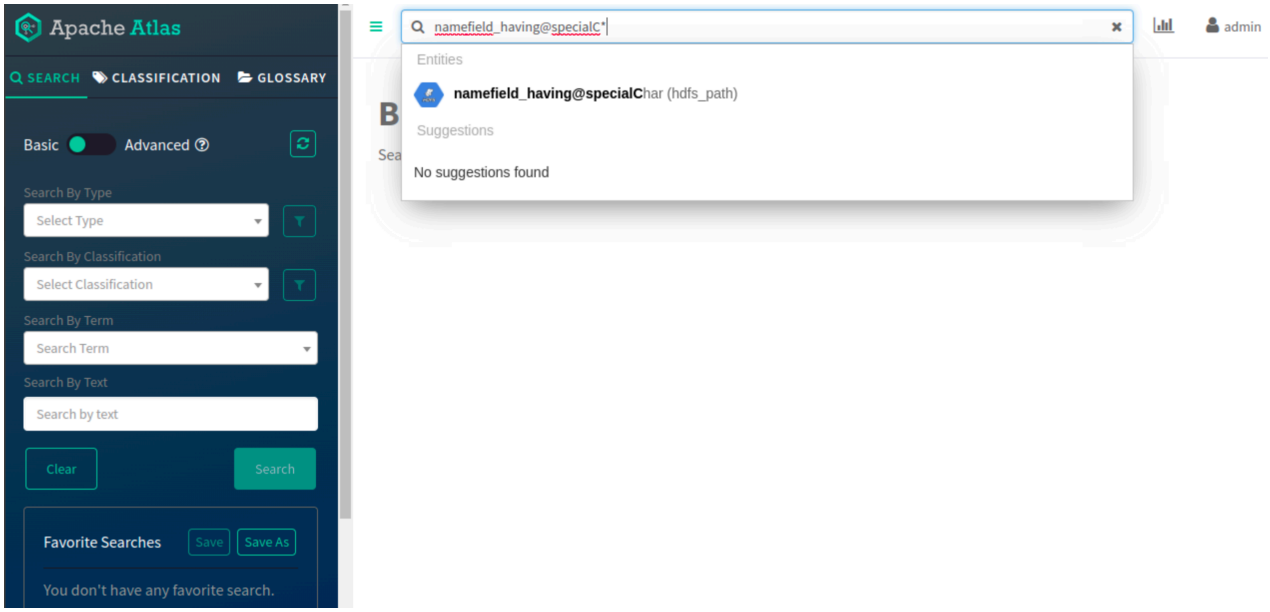


Third use case

In Atlas, name and qualifiedName attributes are different in the way they operate in Atlas Properties. The qualifiedName is a text-based attribute while the name is a string-based attribute. SOLR does not tokenize any string type attribute and does tokenize the text-based attribute.

When you search for 'STRING' type attribute called: 'name', the following conditions must be met:

- Client has to escape 'space'
- For the partial search, client has to append "{*}



For example: While searching for string with name = finance department@axis, user has to search like finance\ department*

Whenever you search for a 'TEXT' type attribute called: 'qualifiedName', the following conditions must be met:

- No Need of escape 'space'. If it is escaped partial search will not work
- For partial search, no need of appending "{*}

For example: While searching for string with qualifiedName = finance department@axis, user has to search like finance department or finance\ department@axis

Ignore or Prune pattern to filter Hive metadata entities

Atlas supports metadata and lineage updates from services like HBase, Hive, Impala, and Spark. You can selectively ignore or prune these incoming metadata messages to reduce resource consumption and focus on the important metadata for your scenario.

These updates are in the form of messages that are posted by these services. The messages contain Atlas entities specific to the service. The notification processing module within Atlas processes these messages.

Typically, most of the metadata is tracked. Sometimes, a part of the schema changes more often than not and tracking these frequent changes creates metadata that is insignificant. The Atlas notification processing system gets overloaded with the frequently changing schema updates. The resultant outcome might be that the low-value messages are processed at the expense of messages that contain critical schema updates.

To overcome such a pattern within a data processing pipeline, you can employ a couple of options:

- Ignore schema updates.
- Preserve an abbreviated form of the entity.

The Ignore and Prune feature within Atlas addresses this scenario for Hive Metastore and Hive Server2 (HS2) hooks. This feature is a mechanism to specify which Hive tables should be ignored and which ones should be pruned. This feature helps regulate data that is posted to Atlas. The user is able to choose data that is important for metadata management and lineage capture.

Note: This mechanism does not exist for other hooks.

Attention: The Ignore / Prune configurations feature is not supported when the configurations are provided in upper case or mixed case. You must use the lower case while setting up the Ignore / Prune configurations.

Tables whose lifecycle is of no consequence are targeted for being ignored. Tables whose lifecycle need not be tracked closely or for garnering minute details are targeted for pruning.

Attention: Atlas tracks the table and table-level lineage; however, columns of pruned table and their column level lineage are not tracked in Atlas.

Use case

As a part of the Extract/Transform/Load (ETL) data pipeline, services such as Hive use a number of temporary and/or staging tables that are short-lived. These temporary and/or staging tables are generally employed during the extract or transform phase before the data is loaded. Once the processing is complete, these tables are not used anymore and are deleted.

With Atlas Hive Hook enabled, Atlas captures metadata events, lifecycle, and lineage of all the Hive entities.

Temporary tables that are created only to aid the development process are safe to be ignored. Metadata for these tables are not generated or reported into Atlas.

For staging tables, tracking details like columns and column-lineage in Atlas may not be useful. By not tracking the information in Atlas, it can significantly reduce the time it takes to process notification and can help the overall performance of Atlas.

You can ignore temporary tables completely. Just the minimum details of these staging tables can be stored in Atlas, to capture data lineage from source to target table through all the intermediate staging tables.

Setting Ignore and Prune Properties

The ignore and prune configuration properties can be set both at Atlas server-side and Hive hooks configuration.

Setting it at Hive Hook side prevents Atlas' metadata from being generated.

If the metadata for ignored and pruned elements is generated and posted on Atlas' Kafka topic, setting this property on Atlas' server side handles these elements before they get stored within Atlas.

Both these properties accept Java regex expressions. For more information, see documentation.

How Ignore and Prune feature works

The configurations are matched against the Hive table's qualifiedName attribute.

Within the Hive hook, qualifiedName attribute value has this format: database.table@namespace

The namespace is the value specified by the atlas.metadata.namespace property.

For example, for a Hive hook, the property atlas.metadata.namespace is set to glv.

On that server, for a table t1 which is a part of database db1, the qualifiedName value is: db1.tb1@glv

Ignore Pattern

Hook-side

atlas.hook.hive.hive_table.ignore.pattern

Atlas server side

atlas.notification.consumer.preprocess.hive_table.ignore.pattern

Prune pattern

Hook-side

atlas.hook.hive.hive_table.prune.pattern

Atlas server side

atlas.notification.consumer.preprocess.hive_table.prune.pattern

Using Ignore and Prune patterns

You can configure both Ignore and Prune patterns to manage your data.

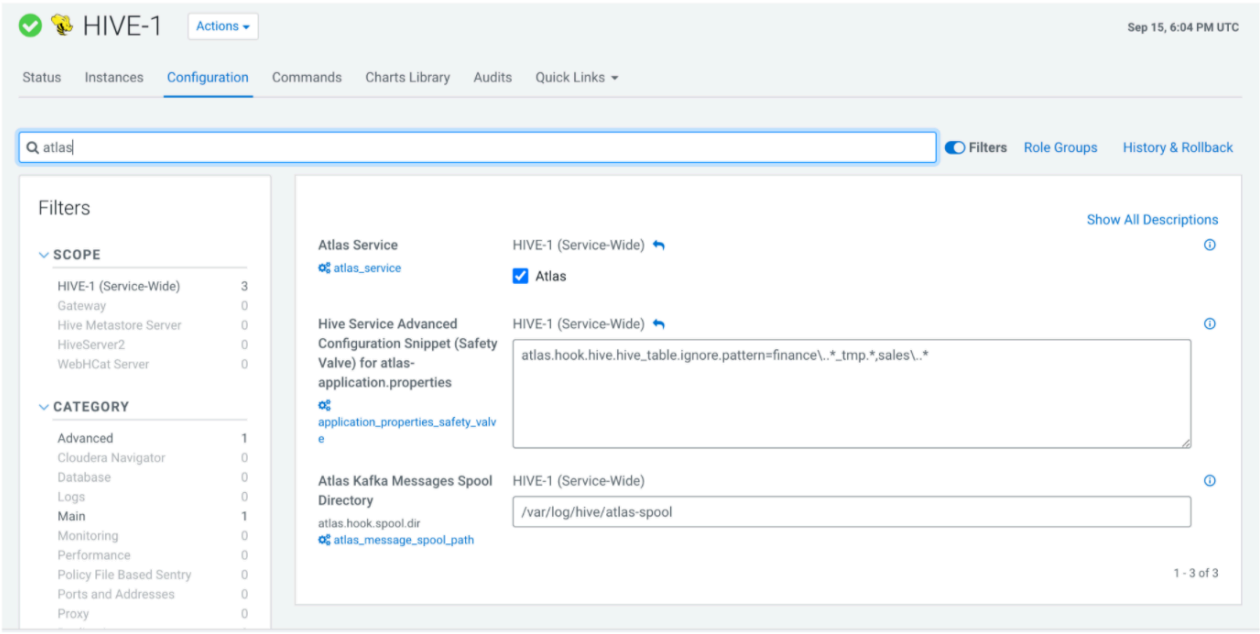
Using the Ignore pattern

Atlas ignores temporary managed tables by default. But an external temporary table is captured because the table uses the HDFS path for storage and Atlas creates a lineage in between.

To disregard the temporary table and avoid Atlas processing it, you can set up appropriate configurations in Hive and Atlas and later restart the services.

For example, if all tables in the ‘sales’ database and the tables that contain ‘_tmp’ in the ‘finance’ database should be ignored, the property can be set as follows in your Cloudera Manager instance.

Hive Metastore Server and Hive settings:



atlas.hook.hive.hive_table.ignore.pattern=finance\..*_tmp.*,sales\..* is set in Cloudera Manager Hive Service Advanced Configuration Snippet (Safety Valve) for atlas-application properties in Hive(HMS) and Hiveserver2.

Atlas server

atlas.notification.consumer.preprocess.hive_table.ignore.pattern=finance\..*_tmp.*

With the above configurations, tables having _tmp in their names, in the finance database are ignored.

Note: The "." is a special regular expression character, hence had to be escaped with a backslash (\).

Using the Prune pattern

Staging tables are created to hold data temporarily during a query execution and are manually dropped once the processing is completed. It might be insignificant to track the details of the staging tables.

For example, in the below images, the finance table contains 333 columns (column names blurred) and the staging tables are created frequently by running an “INSERT OVERWRITE TABLE” query on the finance table. Processing is executed on the data in staging tables and later the staging tables are deleted as observed in the table level lineage diagram.

Every time you run a query to create lineage between tables, column-level lineage is also created along with table-level lineage.

Note: If the query involves all the columns, 333 hive_column_lineage entities are created and pushed to the ATLAS_HOOK Kafka topic.

Using generic ignore patterns

Next to Hive metadata entities, generic ignore patterns can be set up to reduce resource consumption caused by the incoming metadata from any type of service.

Generic ignore patterns allow you to specify entities using regex patterns to be excluded from metadata capture and processing. This helps you to focus its metadata management efforts on relevant data entities, reducing clutter and improving the accuracy and efficiency of metadata operations.

The supported metadata sources include the following:

- Apache Hbase Hook
- Apache Hive Metastore Hook
- Apache HiveServer2 Hook
- Apache Impala Hook
- Apache Spark Hook

Note: When using both generic and Apache Hive specific ignore patterns or prune patterns, the generic ignore patter configuration takes precedence.

Generic ignore patterns on the server side

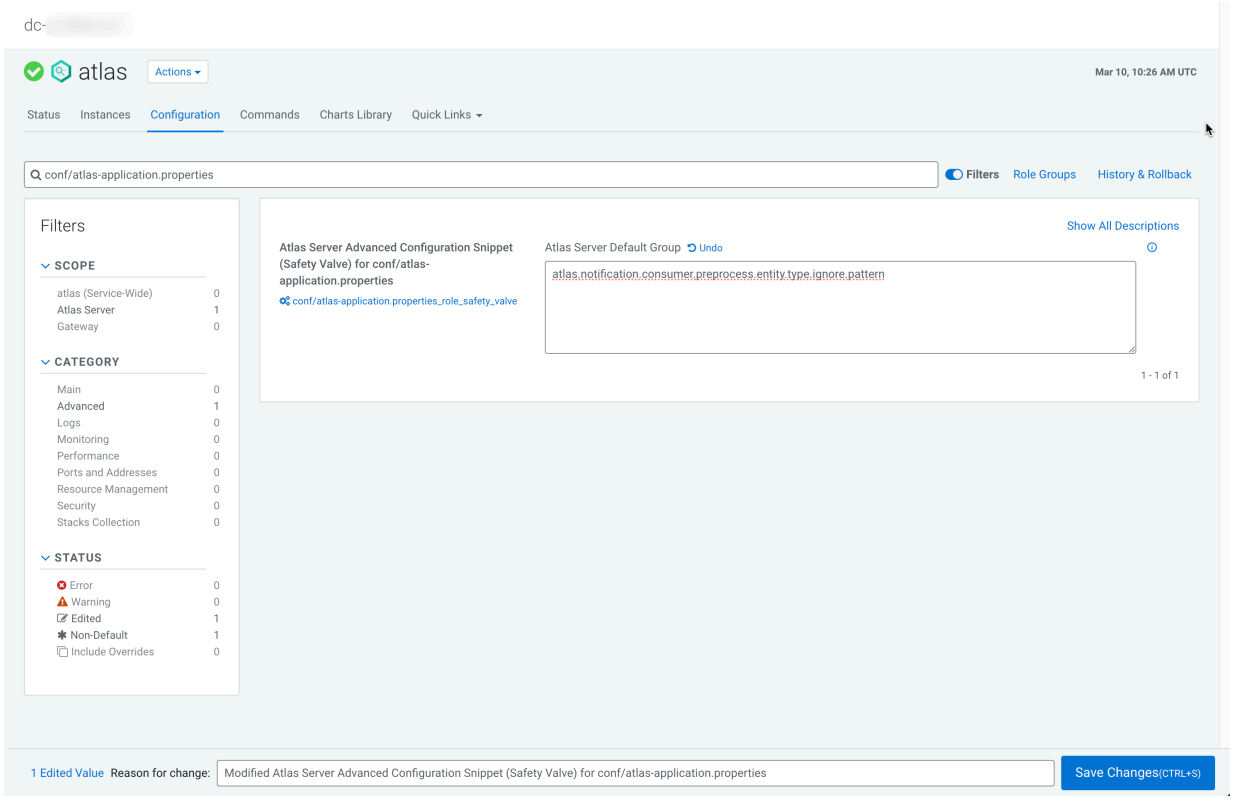
Ignore pattern can be configured to ignore entities based on Type Name, qualifiedName or both. You can configure the generic ignore pattern in Atlas Server Advanced Configuration Snippet (Safety Valve) for conf/atlas-application.properties:

1. Go to Cloudera Manager Clusters Atlas Configuration .
2. Search for conf/atlas-application.properties.
3. Enter the following configurations combined with your regex expression:
   Configuration for Type Name

   ```
   atlas.notification.consumer.preprocess.entity.type.ignore.pattern
   ```

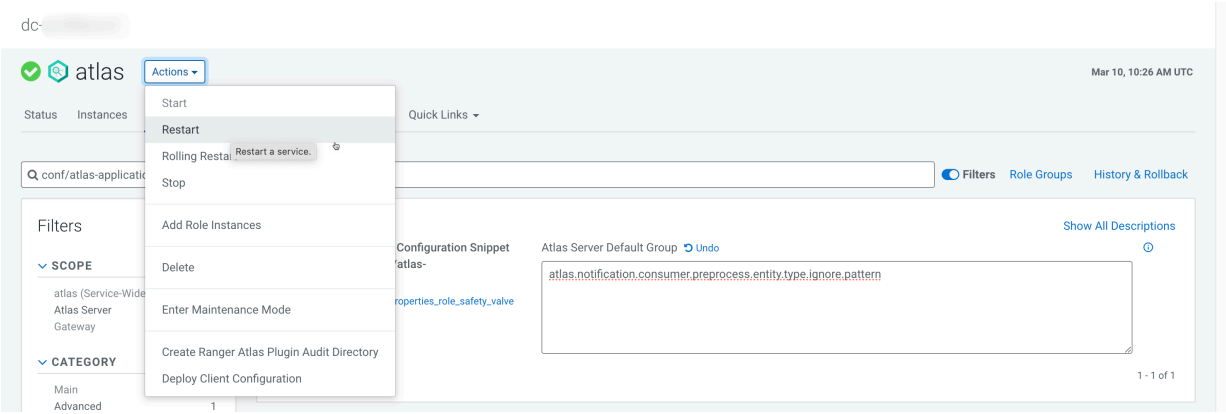
   Configuration for qualifiedName

   ```
   atlas.notification.consumer.preprocess.entity.ignore.pattern
   ```



4. Click Save Changes.

5. Click Actions Restart to apply your changes.



Ignoring all Apache Hive entities

```
atlas.notification.consumer.preprocess.entity.type.ignore.pattern=hive_.*
```

Ignoring all entities with suffix _tmp in their name

```
atlas.notification.consumer.preprocess.type.ignore.pattern=.*\\..*_tmp.*
```

Ignoring all Apache Hive entities with column name "password" and "confidential" as table name

```
atlas.notification.consumer.preprocess.entity.ignore.pattern=.*\\..*\\..*pas
sword.*,.*\\..*confidential.*
atlas.notification.consumer.preprocess.entity.type.ignore.pattern=hive_col
umn,hive_table_ddl
```

Note:

- Multiple regex patterns can be used by dividing them with ",".
- Use "\\" to escape a "." (period) character.

Ignoring all Apache Hive entities with suffix _tmp in their name

```
atlas.notification.consumer.preprocess.entity.type.ignore.pattern=hive_.*
atlas.notification.consumer.preprocess.entity.ignore.pattern=.*\\..*_tmp.*
```

Generic ignore patterns on the hook side

Ignore pattern can be configured on the hook side to ignore entities based on qualifiedName. You can configure the hook side generic ignore patterns Cloudera Manager Advanced Configuration Snippet (Safety Valve) for atlas-application properties section of each individual hook:

1. Go to Cloudera Manager Clusters ***HOOK SERVICE*** Configuration .
2. Search for atlas-application.properties.
3. Enter the following configurations combined with your regex expression:
   Configuration for qualifiedName

   ```
   atlas.hook.entity.ignore.pattern
   ```
4. Click Save Changes.
5. Click Actions Restart to apply your changes.

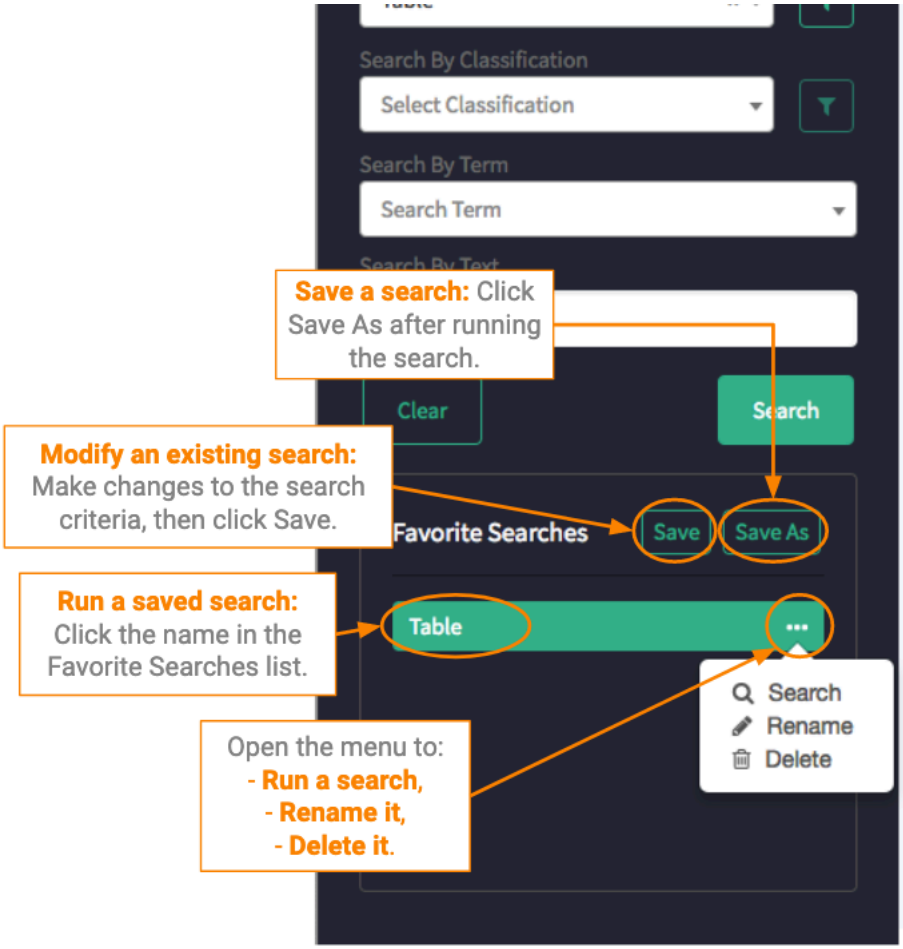
Ignoring all entities with “test” in their name

```
atlas.hook.entity.ignore.pattern=.*\\.test.*
```

Saving searches

Saving a search saves the search criteria with a name that will help you remember what the search returns.

After you run a search, you can save it under a name in the list of Favorites. Here's what you can do to save a search and to use a search you've already saved:



Using advanced search

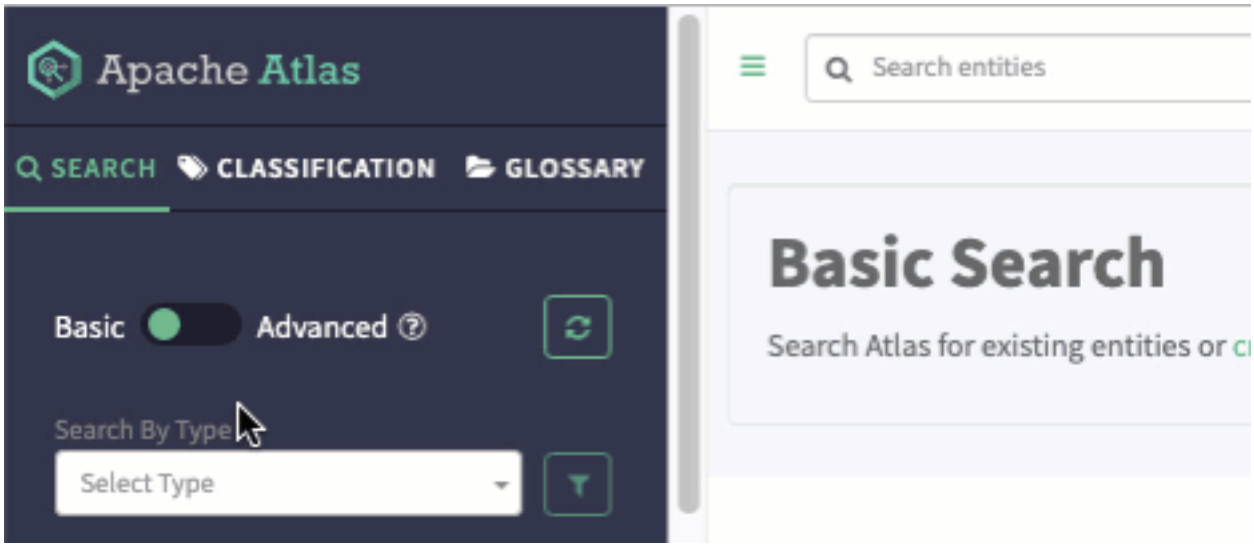
Apache Atlas advanced search lets you use a query language to combine criteria and refine search results.

Important: In Atlas, the Basic Search feature is the recommended search method because of its more efficient use of resources. Advanced Search is recommended to be used in the following scenarios:

- Querying relationship attributes, for example: hive_table select outputFromProcesses.name
- Using aggregate functions, such as sum, min, max, count, groupby

Advanced search gives you more control over search criteria through the Atlas domain-specific query language.

In the left navigator pane, Search tab, switch to Advanced Search mode by sliding the green toggle button from Basic to Advanced.



Select an entity type if appropriate, then add your query to refine the search results. Here are some examples of advanced search queries:

- Search for partial names

```
from hive_table where name LIKE '*_dim'
```

- Search in date ranges

  Note that the entity attributes may contain date fields that are populated from the source while the system attributes contain date fields that are populated when the entity is created in Atlas. The values can be different.

Entity attribute createTime	`from hive_table where createTime > '2019-01-01' and createTime < '2019-01-03'`
System attribute Create Timestamp	`from hive_table where __timestamp > '2019-01-01' and __timestamp < '2019-01-03'`

- Search for deleted entitiesSystem attributes (with two underscores before the name) are available on all entity types.

```
from hive_table where __state = DELETED
```

- Search for multiple criteria

```
from hdfs_path where isFile = true and name = "Invoice"
```

- Return specific metadata

```
from hive_table where name = 'customer_dim' select owner, name,
        qualifiedName, createTime
```

- Search for entities with classifications

```
from hive_table where hive_table isa Dimension select owner, name, quali
fiedName
```

See the advanced search reference for information about the query language and for more examples.

Improved search capabilities for Glossary terms and Relationships

In Atlas, while using the Advanced Search feature, you can now search for entities based on the glossary term, by using the newly introduced hasTerm keyword that allows users to search the entities which are tagged with them. You can also search the entities based on relationship attributes using the where clause.

In order to search for those entities having a specific glossary term, you must add a fully qualified name. For example: {termName}@{glossaryName}. This term gets compared with the qualifiedName attribute of glossary type.

Where as, when you add only the term name, the resultant output will be the available entities with the specific term name. This is irrespective of what type of glossary it is in and would compare with the name attribute of the glossary type.

Additionally, to search for entities related to the referenced entities, you must add the relationship attribute and value to search for in the where clause. For example: To search for tables under a specific database. For example: {relationshipName}.{attributeName} = {value}

Examples of Glossary term filtering:

- Table hasTerm savingAccount1234
- Table hasTerm “savingAccount1234@Banking"
- Table hasTerm “savingAccount1234@Banking" where Table.name = "customer_dim" and tableType = "external"
- Table hasTerm “savingAccount1234@Banking" select name orderby name desc
- Table hasTerm “savingAccount1234@Banking" limit 2
- Table hasTerm “savingAccount1234@Banking" or Table hasTerm "salesTerm@salesGlossary"
- Table hasTerm “savingAccount1234@Banking" and Table isA Dimension
- Table hasTerm “savingAccoun1234t@Banking" and db.name = “Sales" or (Table.qualifiedName like "customer")
- Table where Table hasTerm “savingAccount1234@Banking"
- Table where (name = "customer_dim" and Table hasTerm “savingAccount1234@Banking")
- Table hasTerm "savingAccount1234@Banking" select count() as terms

Examples of Relationship attributes filtering:

-
```
Table where db.name = “Sales4321”
```

- Table where name = “customer_dim” select columns
- Table where columns.name like “sales” and Table isA Dimension
- Table where db.name = “Sales4321” limit 2
- Table where db.name = “Sales4321” orderby name asc
- Table where db.name = “Sales4321” and columns.name like “sales” and Table hasTerm “salesTerm@salesGlossary” - (Combination of both where and hasTerm attribute and keyword respectively.)

Related Information

Atlas Advanced Search language reference
Apache Atlas Advanced Search (atlas.apache.org)

Atlas index repair configuration

You can use reindexing to troubleshoot Apache Atlas basic search inconsistency.

Rebuilding the whole Atlas index

In your Cloudera Manager instance running the Atlas service, add the following in Atlas Server Advanced Configuration Snippet (Safety Valve) for conf/atlas-application.properties.

atlas.rebuild.index=true

atlas.patch.numWorkers=3

atlas.patch.batchSize=300

Later, restart the Atlas Service.

Attention:

- You must revert back this configuration once the reindexing is completed, else the reindexing takes place on every restart.
- The reindexing process will be done during Atlas restart, so Atlas will not be reachable till reindexing process is completed.
- The time taken for reindexing depends upon the amount of data.

Rebuilding the index for particular GUID

Incorrect search results related to a particular GUID can be repaired by limiting the reindex to that element.

atlas-index-repair/repair_index.py [-g <***GUID***>]

Note:

Atlas will use REST APIs to fetch the entity, which will need the correct authentication mechanism to be specified based on the installation.

For an Atlas installation with username and password use the following:

atlas-index-repair/repair_index.py [-g <***GUID***>] [-u <***USER***>] [-p <***PASSWORD***>] * guid: [optional]

Example:

atlas-index-repair/repair_index.py -u admin -p admin123 -g 13d77457-2a45-4e92-ad53-a172c7cb70a5

For Atlas installations using Kerberos as authentication mode, use the following:

kinit -kt /etc/security/keytabs/atlas.service.keytab atlas/fqdn@DOMAIN

Example:

kinit -kt /etc/security/keytabs/atlas.service.keytab atlas/fqdn@EXAMPLE.com

atlas-index-repair/repair_index.py -g 13d77457-2a45-4e92-ad53-a172c7cb70a5

Note: In case of many affected entities, it is recommended to rebuild the whole index instead.