



6.1k

Get Started



Apache Pinot™ 0.11 - How do I see my indexes?

By: Mark Needham

November 8th, 2022 • 4 min read

We recently released [Pinot 0.11.0](#), which has lots of goodies for you to play with. This is the first in a series of blog posts showing off some of the new features in this release.

A common question from the community is: how can you work out which indexes are currently defined on a Pinot table? This information has always been [available via the REST API](#), but sometimes you simply want to see it on the UI and not have to parse your way through a bunch of JSON. Let's see how it works!

Spinning up Pinot

We're going to spin up the Batch [QuickStart](#) in Docker using the following command:

```
docker run \  
  -p 8000:8000 \  
  -p 9000:9000 \  
  apachepinot.docker.scarf.sh/apachepinot/pinot:0.11.0 \  
  QuickStart -type BATCH
```

Or if you're on a Mac M1, change the name of the image to have the arm-64 suffix, like this:

```
docker run \
  -p 8000:8000 \
  -p 9000:9000 \
  apachepinot.docker.scarf.sh/apachepinot/pinot:0.11.0-arm64 \
  QuickStart -type BATCH
```

Once that's up and running, navigate to <http://localhost:9000/#/> and click on Tables. Under the tables section click on airlineStats_OFFLINE. You should see a page that looks like this:

OPERATIONS

Edit Table

Delete Table

Edit Schema

Delete Schema

Truncate Table

Reload All Segments

Reload Status

Rebalance Servers

Rebalance Brokers

☒ Enable

SUMMARY

Table Name: airlineStats_OFFLINE

Reported Size: 3.57 MB

Estimated Size: 3.57 MB

TABLE CONFIG

```

1 {
2   "OFFLINE": {
3     "tableName": "airlineStats_OFFLINE",
4     "tableType": "OFFLINE",
5     "segmentsConfig": {
6       "timeType": "DAYS",
7       "replication": "1",
8       "segmentAssignmentStrategy": "BalanceNumSegmentAssig
9       "timeColumnName": "DaysSinceEpoch",
10      "segmentPushType": "APPEND",
11      "minimizeDataMovement": false
12    },
13    "tenants": {
14      "broker": "DefaultTenant",
15      "server": "DefaultTenant"
16    },
17    "tableIndexConfig": {
18      "invertedIndexColumns": [
19        "FlightNum"
20      ],
21      "loadMode": "MMAP",
22      "enableDefaultStarTree": false,
23      "enableDynamicStarTreeCreation": false

```

TABLE SCHEMA

☐ JSON Format

Search...

Column	Type	Field Type
ActualElapsedTime	INT	Dimension
AirTime	INT	Dimension
AirlineID	INT	Dimension
ArrDel15	INT	Dimension
ArrDelay	INT	Dimension
ArrDelayMinutes	INT	Dimension
ArrTime	INT	Dimension
ArrTimeBlk	STRING	Dimension
ArrivalDelayGroups	INT	Dimension
CRSArrTime	INT	Dimension
CRSDepTime	INT	Dimension

Click on Edit Table. This will show a window with the config for this table.

Edit Config

```
1 {
2   "OFFLINE": {
3     "tableName": "airlineStats_OFFLINE",
4     "tableType": "OFFLINE",
5     "segmentsConfig": {
6       "timeType": "DAYS",
7       "replication": "1",
8       "segmentAssignmentStrategy": "BalanceNumSegmentAssignmen
9       "timeColumnName": "DaysSinceEpoch",
10      "segmentPushType": "APPEND",
11      "minimizeDataMovement": false
12    },
13    "tenants": {
14      "broker": "DefaultTenant",
15      "server": "DefaultTenant"
16    },
17    "tableIndexConfig": {
18      "invertedIndexColumns": [
19        "FlightNum"
20      ],
21      "loadMode": "MMAP",
22      "enableDefaultStarTree": false,
23      "enableDynamicStarTreeCreation": false,
24      "aggregateMetrics": false,
25      "nullHandlingEnabled": false,
26      "optimizeDictionaryForMetrics": false,
27      "noDictionarySizeRatioThreshold": 0,
28      "rangeIndexVersion": 2,
29      "autoGeneratedInvertedIndex": false,
30      "createInvertedIndexDuringSegmentGeneration": false
31    },
32  }
```

CANCEL

SAVE

Indexing Config

We're interested in the `tableIndexConfig` and `fieldConfigList` sections. These sections are responsible for defining indexes, which are applied to a table on a per segment basis.

- `tableIndexConfig` is responsible for inverted, JSON, range, Geospatial, and StarTree indexes.

- fieldConfigList is responsible for timestamp and text indexes.

tableIndexConfig is defined below:

```
"tableIndexConfig": {
  "rangeIndexVersion": 2,
  "autoGeneratedInvertedIndex": false,
  "createInvertedIndexDuringSegmentGeneration": false,
  "loadMode": "MMAP",
  "enableDefaultStarTree": false,
  "enableDynamicStarTreeCreation": false,
  "aggregateMetrics": false,
  "nullHandlingEnabled": false,
  "optimizeDictionaryForMetrics": false,
  "noDictionarySizeRatioThreshold": 0
},
```

From reading this config we learn that no indexes have been explicitly defined.

Now for fieldConfigList, which is defined below:

```
"fieldConfigList": [
  {
    "name": "ts",
    "encodingType": "DICTIONARY",
    "indexType": "TIMESTAMP",
    "indexTypes": [
      "TIMESTAMP"
    ],
    "timestampConfig": {
      "granularities": [
        "DAY",
        "WEEK",
        "MONTH"
      ]
    }
  }
],
```

From reading this config we learn that a timestamp index is being applied to the *ts* column. It is applied at DAY, WEEK, and MONTH granularities, which means that the derived columns *tsDAY*, *tsWEEK*, and *tsMONTH* will be created for the segments in this table.

Viewing Indexes

Now, close the table config modal, and under the segments section, open `airlineStats_OFFLINE_16071_16071_0` and `airlineStats_OFFLINE_16073_16073_0` in new tabs.

If you look at one of those segments, you'll see the following grid that lists columns/field names against the indexes defined on those fields.

INDEXES					
Search...					
Field Name	Bloom Filter	Dictionary	Forward Index	Sorted	Inverted Index
FlightNum	false	true	true	false	false
Origin	false	true	true	false	false
Quarter	false	true	true	true	true
LateAircraftDelay	false	true	true	false	false
DivActualElapsedTime	false	true	true	false	false
DivWheelsOns	false	true	true	false	false
DivWheelsOffs	false	true	true	false	false
AirTime	false	true	true	false	false
ArrDel15	false	true	true	false	false
Rows per page: 10 ▾ 1-10 of 84 < < > > 					

All the fields on display are persisting their values using the dictionary/forward [index format](#)). Still, we can also see that the Quarter column is sorted and has an inverted index, neither of which we explicitly defined.

This is because Pinot will automatically create sorted and inverted indexes for columns whose data is sorted when the segment is created.

So the data for the Quarter column was sorted, and hence it has a sorted index.

I've written a couple of blog posts explaining how sorted indexes work on offline and real-time tables:

Adding an Index

Next, let's see what happens if we add an explicit index. We're going to add an inverted index to the FlightNum column. Go to Edit Table config again and update tableIndexConfig to have the following value:

INDEXES					
Search...					
Field Name	Bloom Filter	Dictionary	Forward Index	Sorted	Inverted Index
FlightNum	false	true	true	false	true
Origin	false	true	true	false	false
Quarter	false	true	true	true	true
LateAircraftDelay	false	true	true	false	false
DivActualElapsedTime	false	true	true	false	false
DivWheelsOns	false	true	true	false	false
DivWheelsOffs	false	true	true	false	false
AirTime	false	true	true	false	false
ArrDel15	false	true	true	false	false

Rows per page: 10 ▼ 1-10 of 84

|< < > >|

If you go back to the page for segment airlineStats_OFFLINE_16073_16073_0, notice that it does not have an inverted index for this field.

INDEXES ^					
 Search...					
Field Name	Bloom Filter	Dictionary	Forward Index	Sorted	Inverted Index
FlightNum	false	true	true	false	false
Origin	false	true	true	false	false
Quarter	false	true	true	true	true
LateAircraftDelay	false	true	true	false	false
DivActualElapsedTime	false	true	true	false	false
DivWheelsOns	false	true	true	false	false
DivWheelsOffs	false	true	true	false	false
AirTime	false	true	true	false	false
ArrDel15	false	true	true	false	false
Rows per page: 10 ▼ 1-10 of 84 < < > >					

This is because indexes are applied on a per segment basis. If we want the inverted index on the FlightNum column in this segment, we can click *Reload Segment* on this page, or we can go back to the table page and click *Reload All Segments*.

If we do that, all the segments in the airlineStats_OFFLINE table will eventually have an inverted index on FlightNum.

Summary

As I mentioned in the introduction, information about the indexes on each segment has always been available via the REST API, but this feature democratizes that information.

If you have any questions about this feature, feel free to join us on [Slack](#), where we'll be happy to help you out.



Resources

[Docs](#)

[Getting Started](#)

[ThirdEye](#)

[Company Stories](#)

[Download](#)

[Blog](#)

Apache

[Foundation](#)

[License](#)

[Security](#)

[Sponsorship](#)

[Events](#)

[Thanks](#)



Join our newsletter



Copyright © 2026 The Apache Software Foundation. Apache Pinot, Pinot, Apache, the Apache feather logo, and the Apache Pinot project logo are registered trademarks of The Apache Software Foundation. This page has references to third party software - Presto, PrestoDB, ThirdEye, Trino, TrinoDB, that are not part of the Apache Software Foundation and are not covered under the Apache License.