Python Client for Pilosa Documentation

Release 0.3.30

Pilosa Engineering

Oct 04, 2017

Contents:

Python client for Pilosa high performance distributed bitmap index.

CHAPTER 1

pilosa package

Submodules

pilosa.client module

class pilosa.client.**Client**(*cluster_or_uri=None*, *connect_timeout=30000*, *socket_timeout=300000*, *pool_size_per_route=10*, *pool_size_total=100*, *retry_count=3*)

Bases: object

Pilosa HTTP client

This client uses Pilosa's http+protobuf API.

Usage:

```
import pilosa

# Create a Client instance
client = pilosa.Client()

# Create an Index instance
index = pilosa.Index("repository")

stargazer = index.frame("stargazer")
response = client.query(stargazer.bitmap(5))

# Act on the result
print(response.result)
```

- See Pilosa API Reference.
- See Query Language.

create_frame(*frame*)
Creates a frame on the server using the given Frame object.

Parameters **frame** (*pilosa.Frame*) –

Raises **pilosa.FrameExistsError** – if there already is a frame with the given name

create_index(*index*)
Creates an index on the server using the given Index object.

Parameters **index** (*pilosa.Index*) –

Raises **pilosa.IndexExistsError** – if there already is a index with the given name

delete_frame(*frame*)
Deletes the given frame on the server.

Parameters **frame** (*pilosa.Frame*) –

Raises **pilosa.PilosaError** – if the frame does not exist

delete_index(*index*)
Deletes the given index on the server.

Parameters **index** (*pilosa.Index*) –

Raises **pilosa.PilosaError** – if the index does not exist

ensure_frame(*frame*)
Creates a frame on the server if it does not exist.

Parameters **frame** (*pilosa.Frame*) –

ensure_index(*index*)
Creates an index on the server if it does not exist.

Parameters **index** (*pilosa.Index*) –

import_frame(*frame*, *bit_reader*, *batch_size=100000*)
Imports a frame using the given bit reader

Parameters

- **frame** –
- **bit_reader** –
- **batch_size** –

query(*query*, *columns=False*, *exclude_bits=False*, *exclude_attrs=False*)
Runs the given query against the server with the given options.

Parameters

- **query** (*pilosa.PqlQuery*) – a PqlQuery object with a non-null index
- **columns** (*bool*) – Enables returning column data from bitmap queries
- **exclude_bits** (*bool*) – Disables returning bits from bitmap queries
- **exclude_attrs** (*bool*) – Disables returning attributes from bitmap queries

Returns Pilosa response

Return type pilosa.Response

schema()

status()

sync_schema(*schema*)

class pilosa.client.**Cluster**(**hosts*)
Contains hosts in a Pilosa cluster.

Parameters **hosts** – URIs of hosts. Leaving out hosts creates the default cluster

add_host(*uri*)
Makes a host available.

Parameters **uri** (*pilosa.URI*) –

copy()

get_host()
Returns the next host in the cluster.

Returns next host

Return type pilosa.URI

remove_host(*uri*)
Makes a host unavailable.

Parameters **uri** (*pilosa.URI*) –

class pilosa.client.**URI**(*scheme='http'*, *host='localhost'*, *port=10101*)
Represents a Pilosa URI

A Pilosa URI consists of three parts:

•Scheme: Protocol of the URI. Default: `http`

•Host: Hostname or IP URI. Default: `localhost`

•Port: Port of the URI. Default `10101`

All parts of the URI are optional. The following are equivalent:

•`http://localhost:10101`

•`http://localhost`

•`http://:10101`

•`localhost:10101`

•`localhost`

•`:10101`

Parameters

- **scheme** (*str*) – is the scheme of the Pilosa Server. Currently only `http` is supported
- **host** (*str*) – is the hostname or IP address of the Pilosa server
- **port** (`int`) – is the port of the Pilosa server

classmethod address(*address*)
Creates a URI from an address.

Parameters **address** (*str*) – of the form `${SCHEME}://${HOST}:{$PORT}`

Returns a Pilosa URI

Type pilosa.URI

pilosa.exceptions module

exception pilosa.exceptions.**PilosaError**
Bases: exceptions.Exception

exception pilosa.exceptions.**ValidationError**
Bases: *pilosa.exceptions.PilosaError*

exception pilosa.exceptions.**PilosaURIError**
Bases: *pilosa.exceptions.PilosaError*

exception pilosa.exceptions.**IndexExistsError**
Bases: *pilosa.exceptions.PilosaError*

exception pilosa.exceptions.**FrameExistsError**
Bases: *pilosa.exceptions.PilosaError*

pilosa.orm module

class pilosa.orm.**TimeQuantum**(*value*)
Valid time quantum values for frames having support for that.

•See: Data Model

DAY = <pilosa.orm.TimeQuantum instance>

DAY_HOUR = <pilosa.orm.TimeQuantum instance>

HOUR = <pilosa.orm.TimeQuantum instance>

MONTH = <pilosa.orm.TimeQuantum instance>

MONTH_DAY = <pilosa.orm.TimeQuantum instance>

MONTH_DAY_HOUR = <pilosa.orm.TimeQuantum instance>

NONE = <pilosa.orm.TimeQuantum instance>

YEAR = <pilosa.orm.TimeQuantum instance>

YEAR_MONTH = <pilosa.orm.TimeQuantum instance>

YEAR_MONTH_DAY = <pilosa.orm.TimeQuantum instance>

YEAR_MONTH_DAY_HOUR = <pilosa.orm.TimeQuantum instance>

class pilosa.orm.**CacheType**(*value*)

DEFAULT = <pilosa.orm.CacheType instance>

LRU = <pilosa.orm.CacheType instance>

RANKED = <pilosa.orm.CacheType instance>

class pilosa.orm.**Schema**
Schema is a container for index objects

index(*name*, *column_label='columnID'*, *time_quantum=<pilosa.orm.TimeQuantum instance>*)
Returns an index object with the given name and options.

If the index didn't exist in the schema, it is added to the schema.

Parameters

- **name** (*str*) – index name
- **column_label** (*str*) – a valid column label. This field is deprecated and will be removed in a future release.
- **time_quantum** (*pilosa.TimeQuantum*) – Sets the time quantum

Returns Index object

•See Data Model

•See Query Language

•‘‘column_label‘ field is deprecated.

class pilosa.orm.**Index**(*name*, *column_label='columnID'*, *time_quantum=<pilosa.orm.TimeQuantum instance>*)

The purpose of the Index is to represent a data namespace.

You cannot perform cross-index queries. Column-level attributes are global to the Index.

Parameters

- **name** (*str*) – index name
- **column_label** (*str*) – a valid column label
- **time_quantum** (*pilosa.TimeQuantum*) – Sets the time quantum

•See Data Model

•See Query Language

batch_query(**queries*)

Creates a batch query.

Parameters **queries** (*pilosa.PQLQuery*) – the queries in the batch

Returns Pilosa batch query

Return type pilosa.PQLBatchQuery

copy(*frames=True*)

count(*bitmap*)

Creates a Count query.

`Count` returns the number of set bits in the BITMAP_CALL passed in.

Parameters **bitmap** (*pilosa.PQLQuery*) – the bitmap query

Returns Pilosa query

Return type pilosa.PQLQuery

difference(**bitmaps*)

Creates a `Difference` query.

`Difference` returns all of the bits from the first BITMAP_CALL argument passed to it, without the bits from each subsequent BITMAP_CALL.

Parameters **bitmaps** (*pilosa.PQLBitmapQuery*) – 1 or more bitmap queries to differentiate

Returns Pilosa bitmap query

Return type pilosa.PQLBitmapQuery

Raises ***PilosaError*** – if the number of bitmaps is less than 1

frame(*name*, *row_label='rowID'*, *time_quantum=<pilosa.orm.TimeQuantum instance>*, *inverse_enabled=False*, *cache_type=<pilosa.orm.CacheType instance>*, *cache_size=0*, *fields=None*)

Creates a frame object with the specified name and defaults.

Parameters

- **name** (*str*) – frame name
- **row_label** (*str*) – a valid row label. This field is deprecated and will be removed in a future release.
- **time_quantum** (*pilosa.TimeQuantum*) – Sets the time quantum for the frame. If a Frame has a time quantum, then Views are generated for each of the defined time segments.
- **inverse_enabled** (*bool*) –
- **cache_type** (*pilosa.CacheType*) – `CacheType.DEFAULT`, `CacheType.LRU` or `CacheType.RANKED`
- **cache_size** (`int`) – Values greater than 0 sets the cache size. Otherwise uses the default cache size
- **fields** (*list(IntField)*) – List of `IntField` objects. E.g.: `[IntField.int("rate", 0, 100)]`

Returns Pilosa frame

Return type pilosa.Frame

•`row_label` field is deprecated.

intersect(**bitmaps*)

Creates an `Intersect` query.

`Intersect` performs a logical AND on the results of each BITMAP_CALL query passed to it.

Parameters **bitmaps** (*pilosa.PQLBitmapQuery*) – 1 or more bitmap queries to intersect

Returns Pilosa bitmap query

Return type pilosa.PQLBitmapQuery

Raises ***PilosaError*** – if the number of bitmaps is less than 1

raw_query(*query*)

Creates a raw query.

Note that the query is not validated before sending to the server.

Parameters **query** (*str*) –

Returns Pilosa query

Return type pilosa.PQLQuery

set_column_attrs(*column_id*, *attrs*)

Creates a SetColumnAttrs query.

`SetColumnAttrs` associates arbitrary key/value pairs with a column in an index.

Following object types are accepted:

•int

- str
- bool
- float

Parameters

- **column_id** (int) –
- **attrs** (*dict*) – column attributes

Returns Pilosa query

Return type pilosa.PQLQuery

union(**bitmaps*)

Creates a Union query.

Union performs a logical OR on the results of each BITMAP_CALL query passed to it.

Parameters **bitmaps** (*pilosa.PQLBitmapQuery*) – 0 or more bitmap queries to union

Returns Pilosa bitmap query

Return type pilosa.PQLBitmapQuery

xor(**bitmaps*)

Creates a Xor query.

Parameters **bitmaps** (*pilosa.PQLBitmapQuery*) – 2 or more bitmap queries to xor

Returns Pilosa bitmap query

Return type pilosa.PQLBitmapQuery

Raises ***PilosaError*** – if the number of bitmaps is less than 2

class pilosa.orm.**PQLQuery**(*pql*, *index*)

serialize()

class pilosa.orm.**PQLBatchQuery**(*index*)

add(**queries*)

serialize()

class pilosa.orm.**IntField**(*attrs*)

classmethod int(*name*, *min=0*, *max=100*)

pilosa.response module

class pilosa.response.**BitmapResult**(*bits=None*, *attributes=None*)

Represents a result from Bitmap, Union, Intersect, Difference and Range PQL calls.

- See Query Language

classmethod from_internal(*obj*)

class pilosa.response.**CountResultItem**(*id*, *count*)
Represents a result from TopN call.

•See Query Language

class pilosa.response.**QueryResult**(*bitmap=None*, *count_items=None*, *count=0*, *sum=0*)
Represents one of the results in the response.

•See Query Language

classmethod from_internal(*obj*)

class pilosa.response.**ColumnItem**(*id*, *attributes*)
Contains data about a column.

Column data is returned from QueryResponse.getColumns() method. They are only returned if Client.query was called with columns=True.

class pilosa.response.**QueryResponse**(*results=None*, *columns=None*, *error_message=''*)
Bases: object

Represents the response from a Pilosa query.

•See Query Language

column

result

pilosa.validator module

pilosa.validator.**valid_index_name**(*index_name*)

pilosa.validator.**validate_index_name**(*index_name*)

pilosa.validator.**valid_frame_name**(*frame_name*)

pilosa.validator.**validate_frame_name**(*frame_name*)

pilosa.validator.**valid_label**(*label*)

pilosa.validator.**validate_label**(*label*)

pilosa.version module

pilosa.version.**get_version**()
Returns the version being used

Module contents

CHAPTER 2

Requirements

- Python 2.6 and higher or Python 3.3 and higher

CHAPTER 3

Install

Pilosa client is on PyPI. You can install the library using `pip`:

```
pip install pilosa
```

CHAPTER 4

Quick overview

Assuming Pilosa server is running at `localhost:10101` (the default):

```
import pilosa

# Create the default client
client = pilosa.Client()

# Create an Index object
myindex = pilosa.Index("myindex")

# Make sure the index exists on the server
client.ensure_index(myindex)

# Create a Frame object
myframe = myindex.frame("myframe")

# Make sure the frame exists on the server
client.ensure_frame(myframe)

# Send a SetBit query. PilosaError is thrown if execution of the query fails.
client.query(myframe.setbit(5, 42))

# Send a Bitmap query. PilosaError is thrown if execution of the query fails.
response = client.query(myframe.bitmap(5))

# Get the result
result = response.result

# Act on the result
if result:
    bits = result.bitmap.bits
    print("Got bits: ", bits)

# You can batch queries to improve throughput
response = client.query(
```

```
    myindex.batch_query(
        myframe.bitmap(5),
        myframe.bitmap(10),
    )
)
for result in response.results:
    # Act on the result
    print(result)
```

CHAPTER 5

Indices and tables

- genindex
- modindex
- search

Python Module Index

p

Index

Q

R

S

T

U

V

X

Y