pygcat Documentation

Release 0.2.0

Rodolphe Quiédeville

November 28, 2015

Contents

Make access to postgresql schema informations easy, is the column **C** is present in an index, is the table **T** contains a primary key, PygCat can answers this.

Contents:

CHAPTER 1

Introduction

PygCatalog simply access to schema informations, you can easily know if a column is present in an index or use of special type in a table.

The only one requirements is to have a psycopg2 connection open to a database, PygCatalog will explore all schemas

CHAPTER 2

PygCatalog

Read information in Postgresql system catalog

exception pygcat.**ColumnDoesNotExists**
A column does not exists

Raised when a column is specificaly requested as a parameter in a function

class pygcat.**PygCatalog**(*conn=None, default_schemas=['public']*)
Python library to read PostgreSQL system catalog

analyze(*table=None*)
Run an ANALYZE over the database or a table

biggest_table()
Return the biggest table in term of total size

The size is compute all disk usage used by the table, it includes datas, indexes and TOAST data. Sizes are express in Bytes.

Example

```
>>> cat.biggest_table()
('foo', 163840L, 1000L)
```

biggest_tables(*max=1*, ***kwargs*)
Return the biggest table in term of total size

The size is compute all disk usage used by the table, it includes datas, indexes and TOAST data.

Example

```
>>> cat.biggest_table()
('foo', 163840L, 1000L)
```

get_indexes(*schema='public'*, ***kwargs*)
Return all indexes in a schema

Return all indexes defined in the schemas, each indexex is associated with the table oid, it's own oid, the number of tuples present in it and the name of the columns.

Example

```
>>> cat.get_indexes()
{'foo_name_idx': {'table_oid': 121090,
                  'oid': 121093,
                  'columns': None,
                  'tuple': 1000L},
```

```
         'foo_name_ratio_idx': {'table_oid': 121090,
                                'oid': 121094,
                                'columns': None,
                                'tuple': 1000L}
        }
```

Returns dict that contains all indexes

Return type dict

get_operator_class(***kwargs*)
Return information on oeprator class

http://www.postgresql.org/docs/current/static/catalog-pg-opclass.html

get_table_columns(*table*, *schema='public'*)
Return all existing columns in a table Won't return dropped columns

Return type list

get_table_columns_extended(*tablename*, *schema='public'*)
Return all existing columns in a table Won't return dropped columns

Return type list

get_tables(***kwargs*)
Return tables list

You may specify a single schema to look in by specifying the keyword argumeent *schema*

Example

```
        >>> cat.get_tables(schema='public')
```

get_triggers(*tablename*, ***kwargs*)
Return information on triggers

http://www.postgresql.org/docs/current/static/catalog-pg-trigger.html

Example

```
        >>> cat.get_triggers('foobar')
        [{'name': 'car_insert_trigger','event': 'INSERT'
          'timing', 'BEFORE'},
         {'name': 'car_update_trigger','event': 'UPDATE',
          'timing': 'AFTER'}
        ]
```

Returns all triggers on a table

Return type array

is_column_exists(*column_name*, *table_name*, *schema='public'*)
Check if a column exists in a table

Parameters

- **column_name** – the column's name to look for
- **table_name** – the table's name to look in

is_column_indexed(*column_name*, *table_name*, *schema='public'*)
Check if a column is indexed

Check if the column is present in at least one index.

Parameters

- **column_name** (*string*) – The column's name to look for
- **table_name** (*string*) – The table's name to look in

Returns The result of the addition

Return type boolean

Example

```
>>> is_table_exists('foobar')
true
```

is_table_exists(*table_name*, *schema='public'*)
Check if a table exists

Parameters **table_name** (*string*) – The table's name to look for

Returns The result of the addition

Return type boolean

Example

```
>>> is_table_exists('foobar')
true
```

pgversion()
Run the version of PostgreSQL

reset_cache()
Reset the cache

schemas()
Return schemas

Return the list of all schemas present in the database

Example

```
>>> cat.get_schemas()
['pg_toast', 'pg_temp_1', 'pg_toast_temp_1', 'pg_catalog',
 'public', 'information_schema', 'alice']
```

Return type list

set_default_schema(*schema*)
Define the default schema to work on

Parameters **schema** (*string*) – The schema's name to work on

Returns The result of the addition

Return type boolean

set_default_schemas(*schemas*)
Define as set of schemas to work on

Remove schemas set twice or more

table_tuples(*table*, ***kwargs*)
Return the table's number of tuples

exception pygcat.**SchemaDoesNotExists**
A schema does not exists

Raised when a schema is specificaly requested as a parameter in a function

exception pygcat.**TableDoesNotExists**
A table does not exists

Raised when a table is specificaly requested as a parameter in a function

CHAPTER 3

Exceptions

class pygcat.**SchemaDoesNotExists**
A schema does not exists

Raised when a schema is specificaly requested as a parameter in a function

class pygcat.**TableDoesNotExists**
A table does not exists

Raised when a table is specificaly requested as a parameter in a function

class pygcat.**ColumnDoesNotExists**
A column does not exists

Raised when a column is specificaly requested as a parameter in a function

CHAPTER 4

Contributing

- Source code
- Issues

CHAPTER 5

Indices and tables

- genindex
- modindex
- search

p

A

B

C

G

I

P

R

S

T