parquetranger

Endre Márk Borza

May 30, 2023

CONTENTS:

CHAPTER

ONE

INSTALLATION:

1.1 using pip

```
pip install parquetranger
```

CHAPTER

TWO

QUICKSTART

```
import pandas as pd

from parquetranger import TableRepo
```

```
df = pd.DataFrame(
    {
        "A": [1, 2, 3, 4, 5, 6],
        "B": ["x", "y", "z", "x1", "x2", "x3"],
        "C": [1, 2, 1, 1, 1, 2],
        "C2": ["a", "a", "b", "a", "c", "c"],
    },
    index=["a1", "a2", "a3", "a4", "a5", "a6"],
)
```

```
df
```

```
trepo = TableRepo("some_tmp_path", group_cols="C2")  # this creates the directory
```

```
trepo.extend(df)
```

```
trepo.get_full_df()
```

```
df2 = pd.DataFrame(
    {
        "A": [21, 22, 23],
        "B": ["X", "Y", "Z"],
        "C": [10,20,1],
        "C2": ["a", "b", "a"],
    },
    index=["a1", "a4", "a7"]
    )
```

```
trepo.replace_records(df2)  # replaces based on index
```

```
trepo.get_full_df()
```

```
trepo.replace_groups(df2)
```

```
trepo.get_full_df()  # replaced the whole groups where C2==a and C2==b with the records␣
↪that were present in df2
```

```
trepo.replace_all(df2)  # erases everything and puts df2 in. all traces of df are lost
```

```
trepo.get_full_df()
```

```
trepo.replace_records(df, by_groups=True)  # replaces records based on index, but only␣
↪looks for indices within groups, so this way duplicate a4 index is possible
# as they are in different groups, with different values in C2
```

```
trepo.get_full_df()
```

```
trepo.purge()  # deletes everything
```

CHAPTER

THREE

API

3.1 parquetranger Package

Read and write parquet files

3.1.1 Classes

DfBatchWriter(trepo, record_limit, ...)	
HashPartitioner([col, num_groups])	
ObjIngestor(root[, root_id_key, force_key, ...])	
RecordWriter(trepo, record_limit, ...)	
TableRepo(root_path[, max_records, ...])	helps with storing, extending and reading tabular data in parquet format

DfBatchWriter

class parquetranger.**DfBatchWriter**(*trepo: parquetranger.core.TableRepo*, *record_limit: int = 1000000*, *writer_function: Callable = <function TableRepo.extend at 0x7ff654390f70>*, *batch: list = <factory>*)

Bases: *RecordWriter*

HashPartitioner

class parquetranger.**HashPartitioner**(*col: str | None = None*, *num_groups: int = 128*)

Bases: object

Attributes Summary

col
key
num_groups

Methods Summary

__call__(df)	Call self as a function.
h(elem)	

Attributes Documentation

col: str | None = None

key

num_groups: int = 128

Methods Documentation

__call__(*df: DataFrame*) → Series

Call self as a function.

h(*elem: str*)

ObjIngestor

class parquetranger.**ObjIngestor**(*root: pathlib.Path*, *root_id_key: str | None = None*, *force_key: bool = False*, *forward_uuids: bool = False*, *total_atoms: int = 0*, *largest_size: int = 0*)

Bases: object

Attributes Summary

force_key
forward_uuids
largest_size
root_id_key
size_limit
total_atoms

Methods Summary

dump_all()
dump_largest()
ingest(obj[, parents, parent_id])

Attributes Documentation

force_key: bool = False

forward_uuids: bool = False

largest_size: int = 0

root_id_key: str | None = None

size_limit = 1000000

total_atoms: int = 0

Methods Documentation

dump_all()

dump_largest()

ingest(*obj*, *parents=()*, *parent_id=None*)

RecordWriter

class parquetranger.**RecordWriter**(*trepo: parquetranger.core.TableRepo*, *record_limit: int = 1000000*, *writer_function: Callable = <function TableRepo.extend at 0x7ff654390f70>*, *batch: list = <factory>*)

Bases: `object`

Attributes Summary

record_limit

Methods Summary

add_to_batch(element)
close()
writer_function(df)

Attributes Documentation

record_limit: int = 1000000

Methods Documentation

add_to_batch(*element*)

close()

writer_function(*df: DataFrame*)

TableRepo

class parquetranger.**TableRepo**(*root_path: Path | str*, *max_records: int = 0*, *group_cols: str | list | HashPartitioner | None = None*, *env_parents: dict[str, pathlib.Path | str] | None = None*, *mkdirs=True*, *extra_metadata: dict | None = None*, *drop_group_cols: bool = False*, *fixed_metadata: dict | None = None*, *allow_metadata_extension: bool = False*)

Bases: `object`

helps with storing, extending and reading tabular data in parquet format

tries dividing based on group_cols, if that is None tries dividing based on max_records, if max records is 0 just writes the file to root_path.parquet

if both group_cols and max_records is given, it will create directories for the groups (nested directories if multiple columns given)

Attributes Summary

dfs
full_metadata
main_path
n_files
paths
vc_path

Methods Summary

batch_extend(df_iterator, **para_kwargs)	
env_ctx(env_name)	
extend(df)	
get_extending_df_batch_writer([max_records])	
get_extending_dict_batch_writer([max_record	
get_extending_fixed_dict_batch_writer(cols)	
get_full_df()	
get_full_table()	
get_partition_paths(partition_col)	
get_replacing_df_batch_writer([max_records])	
get_replacing_dict_batch_writer([max_record	
map_partitions(fun[, level])	
purge()	purges everything
read_df_from_path(path[, lock, release])	
read_table_from_path(path[, lock, release])	
replace_all(df)	purges everything and writes df instead
replace_groups(df)	replace files based on file name, only viable if *group_cols* is set
replace_records(df[, by_groups])	replace records in files based on index
set_env(env)	
set_env_to_default()	

Attributes Documentation

dfs

full_metadata

main_path

n_files

paths

vc_path

Methods Documentation

batch_extend(*df_iterator*, ***para_kwargs*)

env_ctx(*env_name*)

extend(*df: DataFrame*)

get_extending_df_batch_writer(*max_records=1000000*)

get_extending_dict_batch_writer(*max_records=1000000*)

get_extending_fixed_dict_batch_writer(*cols*, *max_records=1000000*)

get_full_df() → DataFrame

get_full_table() → Table

get_partition_paths(*partition_col: str*) → Iterable[tuple[str, Iterable[pathlib.Path]]]

get_replacing_df_batch_writer(*max_records=1000000*)

get_replacing_dict_batch_writer(*max_records=1000000*)

map_partitions(*fun*, *level=None*, ***para_kwargs*)

purge()
purges everything

read_df_from_path(*path: Path*, *lock: allocate_lock | None = None*, *release=True*) → DataFrame

read_table_from_path(*path*, *lock: allocate_lock | None = None*, *release=True*) → Table

replace_all(*df: DataFrame*)
purges everything and writes df instead

replace_groups(*df: DataFrame*)
replace files based on file name, only viable if *group_cols* is set

replace_records(*df: DataFrame*, *by_groups=False*)
replace records in files based on index

set_env(*env: str*)

set_env_to_default()

3.1.2 Class Inheritance Diagram

CHAPTER

FOUR

RELEASE NOTES

4.1 v0.0.0

- first release of parquetranger, yay!!

4.2 v0.0.1

- fix specific dask

4.3 v0.0.10

4.4 v0.0.11

4.5 v0.0.12

- add extra metadata feature

4.6 v0.0.2

- add accessors

4.7 v0.0.3

- add col stretching

4.8 v0.0.4

- fix axis error

4.9 v0.0.5

- fix names indices

4.10 v0.0.6

- add env handling

4.11 v0.0.7

- add name prop

4.12 v0.0.8

- remove duplicate indices

4.13 v0.0.9

- params and renames

4.14 v0.1.0

- add extra metadata feature

4.15 v0.1.1

4.16 v0.1.2

4.17 v0.1.3

lazy client loading

4.18 v0.1.4

4.19 v0.1.5

4.20 v0.1.6

4.21 v0.1.7

4.22 v0.2.0

- first release of parquetranger, yay!!

4.23 v0.2.1

return partition map

4.24 v0.2.2

try_dask default change

4.25 v0.2.3

partition gb path

4.26 v0.3.0

smarter lock business

4.27 v0.4.0

dump dask

4.28 v0.4.1

expose vc path

4.29 v0.4.2

add group col deleting

4.30 v0.4.3

fit new atqo

4.31 v0.4.4

no col issue

4.32 v0.4.5

map partition with level

4.33 v0.4.6

add batch writers

4.34 v0.5.0

massive speedup

4.35 v0.5.1

refix mysteries

4.36 v0.5.2

fixed thing

CHAPTER

FIVE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

INDEX

Symbols

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

R

S

T

V

W