

---

**parquetranger**

**Endre Márk Borza**

**May 30, 2023**



## CONTENTS:

<b>1</b>	<b>Installation:</b>	<b>3</b>
1.1	using pip . . . . .	3
<b>2</b>	<b>Quickstart</b>	<b>5</b>
<b>3</b>	<b>API</b>	<b>7</b>
3.1	parquettranger Package . . . . .	7
<b>4</b>	<b>Release Notes</b>	<b>15</b>
4.1	v0.0.0 . . . . .	15
4.2	v0.0.1 . . . . .	15
4.3	v0.0.10 . . . . .	15
4.4	v0.0.11 . . . . .	15
4.5	v0.0.12 . . . . .	15
4.6	v0.0.2 . . . . .	15
4.7	v0.0.3 . . . . .	15
4.8	v0.0.4 . . . . .	16
4.9	v0.0.5 . . . . .	16
4.10	v0.0.6 . . . . .	16
4.11	v0.0.7 . . . . .	16
4.12	v0.0.8 . . . . .	16
4.13	v0.0.9 . . . . .	16
4.14	v0.1.0 . . . . .	16
4.15	v0.1.1 . . . . .	16
4.16	v0.1.2 . . . . .	16
4.17	v0.1.3 . . . . .	16
4.18	v0.1.4 . . . . .	17
4.19	v0.1.5 . . . . .	17
4.20	v0.1.6 . . . . .	17
4.21	v0.1.7 . . . . .	17
4.22	v0.2.0 . . . . .	17
4.23	v0.2.1 . . . . .	17
4.24	v0.2.2 . . . . .	17
4.25	v0.2.3 . . . . .	17
4.26	v0.3.0 . . . . .	17
4.27	v0.4.0 . . . . .	17
4.28	v0.4.1 . . . . .	18
4.29	v0.4.2 . . . . .	18
4.30	v0.4.3 . . . . .	18
4.31	v0.4.4 . . . . .	18

4.32	v0.4.5 . . . . .	18
4.33	v0.4.6 . . . . .	18
4.34	v0.5.0 . . . . .	18
4.35	v0.5.1 . . . . .	18
4.36	v0.5.2 . . . . .	19
<b>5</b>	<b>Indices and tables</b>	<b>21</b>
	<b>Python Module Index</b>	<b>23</b>
	<b>Index</b>	<b>25</b>





## INSTALLATION:

### 1.1 using pip

```
pip install parquetranger
```





## 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
```

## 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

<i>col</i>
<i>key</i>
<i>num_groups</i>

Methods Summary

<i>__call__</i> (df)	Call self as a function.
<i>h</i> (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

<i>force_key</i>
<i>forward_uuids</i>
<i>largest_size</i>
<i>root_id_key</i>
<i>size_limit</i>
<i>total_atoms</i>

### Methods Summary

<i>dump_all()</i>
<i>dump_largest()</i>
<i>ingest(obj[, parents, parent_id])</i>

### 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

<code>record_limit</code>
---------------------------

### Methods Summary

<code>add_to_batch(element)</code>
------------------------------------

<code>close()</code>
----------------------

<code>writer_function(df)</code>
----------------------------------

### 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, makedirs=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

<i>dfs</i>
<i>full_metadata</i>
<i>main_path</i>
<i>n_files</i>
<i>paths</i>
<i>vc_path</i>

## Methods Summary

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

## 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



## 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





## INDICES AND TABLES

- `genindex`
- `modindex`
- `search`



## PYTHON MODULE INDEX

### p

parquettranger, [7](#)



## Symbols

`__call__()` (*parquettranger.HashPartitioner* method), 8

## A

`add_to_batch()` (*parquettranger.RecordWriter* method), 10

## B

`batch_extend()` (*parquettranger.TableRepo* method), 13

## C

`close()` (*parquettranger.RecordWriter* method), 10  
`col` (*parquettranger.HashPartitioner* attribute), 8

## D

`DfBatchWriter` (class in *parquettranger*), 7  
`dfs` (*parquettranger.TableRepo* attribute), 12  
`dump_all()` (*parquettranger.ObjIngestor* method), 9  
`dump_largest()` (*parquettranger.ObjIngestor* method), 9

## E

`env_ctx()` (*parquettranger.TableRepo* method), 13  
`extend()` (*parquettranger.TableRepo* method), 13

## F

`force_key` (*parquettranger.ObjIngestor* attribute), 9  
`forward_uuids` (*parquettranger.ObjIngestor* attribute), 9  
`full_metadata` (*parquettranger.TableRepo* attribute), 12

## G

`get_extending_df_batch_writer()` (*parquettranger.TableRepo* method), 13  
`get_extending_dict_batch_writer()` (*parquettranger.TableRepo* method), 13  
`get_extending_fixed_dict_batch_writer()` (*parquettranger.TableRepo* method), 13  
`get_full_df()` (*parquettranger.TableRepo* method), 13  
`get_full_table()` (*parquettranger.TableRepo* method), 13  
`get_partition_paths()` (*parquettranger.TableRepo* method), 13

`get_replacing_df_batch_writer()` (*parquettranger.TableRepo* method), 13  
`get_replacing_dict_batch_writer()` (*parquettranger.TableRepo* method), 13

## H

`h()` (*parquettranger.HashPartitioner* method), 8  
`HashPartitioner` (class in *parquettranger*), 7

## I

`ingest()` (*parquettranger.ObjIngestor* method), 9

## K

`key` (*parquettranger.HashPartitioner* attribute), 8

## L

`largest_size` (*parquettranger.ObjIngestor* attribute), 9

## M

`main_path` (*parquettranger.TableRepo* attribute), 12  
`map_partitions()` (*parquettranger.TableRepo* method), 13  
module  
    *parquettranger*, 7

## N

`n_files` (*parquettranger.TableRepo* attribute), 12  
`num_groups` (*parquettranger.HashPartitioner* attribute), 8

## O

`ObjIngestor` (class in *parquettranger*), 8

## P

*parquettranger*  
    module, 7  
`paths` (*parquettranger.TableRepo* attribute), 12  
`purge()` (*parquettranger.TableRepo* method), 13

## R

`read_df_from_path()` (*parquettranger.TableRepo* method), 13

`read_table_from_path()` (*parquettranger.TableRepo* method), 13  
`record_limit` (*parquettranger.RecordWriter* attribute), 10  
`RecordWriter` (class in *parquettranger*), 10  
`replace_all()` (*parquettranger.TableRepo* method), 13  
`replace_groups()` (*parquettranger.TableRepo* method), 13  
`replace_records()` (*parquettranger.TableRepo* method), 13  
`root_id_key` (*parquettranger.ObjIngestor* attribute), 9

## S

`set_env()` (*parquettranger.TableRepo* method), 13  
`set_env_to_default()` (*parquettranger.TableRepo* method), 13  
`size_limit` (*parquettranger.ObjIngestor* attribute), 9

## T

`TableRepo` (class in *parquettranger*), 10  
`total_atoms` (*parquettranger.ObjIngestor* attribute), 9

## V

`vc_path` (*parquettranger.TableRepo* attribute), 12

## W

`writer_function()` (*parquettranger.RecordWriter* method), 10