Bi-temporal HDF5 Documentation

Release 0.0.3

Bi-temporal HDF5 Developers

Oct 10, 2019

CONTENTS

A generic bitemporal model built on HDF5 (h5py)

CHAPTER

ONE

MODEL

The basic model for a bitemporal is an HDF5 dataset that is extensible along a single dimension with named columns and different dtypes for each column. In-memory, this will be represented by a numpy structured array. We will call this structure a *Table*, for purposes here.

Note that HDF5 has its own Table data structure in the high-level interface (hdf5-hl). We will not be using the high-level table here for a couple of reasons. The first is that `h5py` does not support HDF5's high-level constructs. The second is that we plan on eventually swapping out the value column with a deduplicated cache. Relying on low-level HDF5 constructs grants us this flexibility in the future.

The columns present in the table are as follows:

- `transaction_id (uint64)`: This is a monotonic integer that represents the precise write action that caused this row to be written. Multiple rows may be written at the same time, so this value is not unique among rows, though presumably all rows with a given transaction id are contiguous in the table. This value is zero-indexed. The current largest transaction id should be written to the table's attributes as `max_transaction_id` (also uint64). Write operations should bump the `max_transaction_id` by one.
- `transaction_time (datetime64)`: This is a timestamp (sec since epoch). Any metadata about the timezones should be stored as a string attribute of the dataset as `transaction_time_zone`. This represents the time at which the data was recorded by the write operation. All rows with the same `transaction_id` should have the same value here.
- `valid_time (datetime64)`: This is a timestamp (sec since epoch). Any metadata about the timetzones should be stored as a string attribute of the dataset as `valid_time_zone`. This is the primary axis of the time series. It represents the data stored in the `value` column.
- `value ((I,J,K,...)<scalar-type>|)`: This column represents the actual values of a time series. This may be an N-dimensional array of any valid dtype. It is likely sufficient to restrict ourselves to floats and ints, but the model should be general enough to accept any scalar dtypes. Additionally, the typical usecase will be for this column to be a scalar float value.

Therefor an example numpy dtype with float values and a shape of `(1, 2, 3)` is:

```
np.dtype([
    ('transaction_id', '<uint64'),
    ('transaction_time', '<M8'),
    ('valid_time', '<M8'),
    ('value', '<f8', (1, 2, 3))
])
```

CHAPTER

TWO

QUICKSTART API

The interface for writing to the bitemporal HDF5 storage is as follows:

```
>>> with bth5.open(temp_h5, '/', mode='w', value_dtype=np.int64) as ds:
...     ds.write(np.datetime64("2018-06-21 12:26:47"), 1.0)
...     ds.write(np.datetime64("2018-06-21 12:26:49"), 2.0)
...     ds.write([
...         np.datetime64("2018-06-21 12:26:51"),
...         np.datetime64("2018-06-21 12:26:53"),
...     ], [3.0, 4.0])
>>> with bth5.open(temp_h5, '/', mode='r', value_dtype=np.int64) as ds:
...     ds.records[:]
array([(0, '2018-06-21T12:26:47.000000', 1),
       (0, '2018-06-21T12:26:49.000000', 2),
       (0, '2018-06-21T12:26:51.000000', 3),
       (0, '2018-06-21T12:26:53.000000', 4)],
      dtype=[('transaction_id', '<u8'), ('valid_time', '<M8[us]'), ('value', '<i8')])
>>> with bth5.open(temp_h5, '/', mode='r', value_dtype=np.int64) as ds:
...     ds.transactions[:]
array([('2019-09-30T15:35:31.009517', '2018-06-21T12:26:47.000000', '2018-06-
↪21T12:26:53.000000', 0, 4)],
      dtype=[('transaction_time', '<M8[us]'), ('start_valid_time', '<M8[us]'), ('end_
↪valid_time', '<M8[us]'), ('start_idx', '<u8'), ('end_idx', '<u8')])
```

Dataset(filename, path[, mode, value_dtype])	Represents a bitemporal dataset as a memory-mapped structure stored in HDF5.
open(filename, path[, mode, value_dtype])	Opens a bitemporal HDF5 dataset.

CHAPTER

THREE

DATASET

class bth5.**Dataset**(*filename*, *path*, *mode='r'*, *value_dtype=None*)

Represents a bitemporal dataset as a memory-mapped structure stored in HDF5.

Examples

```
>>> ds = bth5.Dataset(temp_h5, '/path/to/group', mode='a', value_dtype=np.float64)
>>> with ds:
...     ds.write(np.datetime64("2018-06-21 12:26:47"), 2.0)
>>> # Write happens here.
>>> with ds:
...     ds.valid_times[:]
array([(0, '2018-06-21T12:26:47.000000', 2.)],
      dtype=[('transaction_id', '<u8'), ('valid_time', '<M8[us]'), ('value', '<f8
↪')])
```

Attributes

Dataset.dtype	The dtype of this dataset.
Dataset.file_dtype	The dtype stored in the file.

3.1 Dataset.dtype

property Dataset.**dtype**

The dtype of this dataset.

3.2 Dataset.file_dtype

property Dataset.**file_dtype**

The dtype stored in the file.

Methods

Dataset.__init__(filename, path[, mode, ...])	Creates a *Dataset*.

Continued on next page

Table 2 – continued from previous page

Dataset.close()	Close the current file handle.
Dataset.interpolate_values(interp_times)	Interpolates the values at the given valid times.
Dataset.open([mode])	Opens the file for various operations
Dataset.write(valid_time, value)	Appends data to a dataset.

3.3 Dataset.__init__

Dataset.**__init__**(*filename*, *path*, *mode='r'*, *value_dtype=None*)
Creates a *Dataset*.

Parameters

- **filename** (*str*) – The path to the h5 file, on disk.
- **path** (*str*) – The path to the group within the HDF5 file.
- **mode** (*str*) – The mode to open a file with.
- **value_dtype** (*str, optional*) – The dtype of the value that is attached to

3.4 Dataset.close

Dataset.**close**()
Close the current file handle.

3.5 Dataset.interpolate_values

Dataset.**interpolate_values**(*interp_times*)
Interpolates the values at the given valid times.

3.6 Dataset.open

Dataset.**open**(*mode='r'*, ***kwargs*)
Opens the file for various operations

3.7 Dataset.write

Dataset.**write**(*valid_time*, *value*)
Appends data to a dataset.

Examples

```
>>> with bth5.open(temp_h5, '/', mode='w', value_dtype=np.int64) as ds:
...     ds.write(np.datetime64("2018-06-21 12:26:47"), 1.0)
...     ds.write(np.datetime64("2018-06-21 12:26:49"), 2.0)
...     ds.write([
```

(continues on next page)

(continued from previous page)

```
...          np.datetime64("2018-06-21 12:26:51"),
...          np.datetime64("2018-06-21 12:26:53"),
...      ], [3.0, 4.0])
>>> with bth5.open(temp_h5, '/', mode='r', value_dtype=np.int64) as ds:
...      ds.records[:]
array([(0, '2018-06-21T12:26:47.000000', 1),
       (0, '2018-06-21T12:26:49.000000', 2),
       (0, '2018-06-21T12:26:51.000000', 3),
       (0, '2018-06-21T12:26:53.000000', 4)],
      dtype=[('transaction_id', '<u8'), ('valid_time', '<M8[us]'), ('value', '
↪<i8')])
```

Indexers

Dataset.records	Index into the dataset by record ID.
Dataset.transaction_times	Index into the transaction index by transaction time.
Dataset.transactions	Index into the transaction index by transaction ID.
Dataset.transaction_idx	Indexes into the dataset by transaction ID.
Dataset.valid_times	Indexes into the dataset by valid time.

3.8 Dataset.records

Dataset.**records**
Index into the dataset by record ID.

Examples

```
>>> with bth5.open(temp_h5, '/', mode='w', value_dtype=np.int64) as ds:
...      ds.write(np.datetime64("2018-06-21 12:26:47"), 2.0)
...      ds.write(np.datetime64("2018-06-21 12:26:49"), 2.0)
>>> with bth5.open(temp_h5, '/', mode='r', value_dtype=np.int64) as ds:
...      ds.records[:]
array([('2019-09-19T10:32:00.210817', '2018-06-21T12:26:47.000000', '2018-06-
↪21T12:26:49.000000', 0, 2)],
      dtype=[('transaction_time', '<M8[us]'), ('start_valid_time', '<M8[us]'),
↪ ('end_valid_time', '<M8[us]'), ('start_idx', '<u8'), ('end_idx', '<u8')])
```

3.9 Dataset.transaction_times

Dataset.**transaction_times**
Index into the transaction index by transaction time.

Examples

```
>>> with bth5.open(temp_h5, '/', mode='w', value_dtype=np.int64) as ds:
...      ds.write(np.datetime64("2018-06-21 12:26:47"), 2.0)
```

(continues on next page)

(continued from previous page)

```
...     ds.write(np.datetime64("2018-06-21 12:26:49"), 2.0)
>>> with bth5.open(temp_h5, '/', mode='r', value_dtype=np.int64) as ds:
...     ds.transaction_times[:]
array([('2019-09-19T10:32:00.210817', '2018-06-21T12:26:47.000000', '2018-06-
↪21T12:26:49.000000', 0, 2)],
      dtype=[('transaction_time', '<M8[us]'), ('start_valid_time', '<M8[us]'),
↪ ('end_valid_time', '<M8[us]'), ('start_idx', '<u8'), ('end_idx', '<u8')])
```

3.10 Dataset.transactions

Dataset.**transactions**
Index into the transaction index by transaction ID.

Examples

```
>>> with bth5.open(temp_h5, '/', mode='w', value_dtype=np.int64) as ds:
...     ds.write(np.datetime64("2018-06-21 12:26:47"), 2.0)
...     ds.write(np.datetime64("2018-06-21 12:26:49"), 2.0)
>>> with bth5.open(temp_h5, '/', mode='r', value_dtype=np.int64) as ds:
...     ds.transactions[:]
array([('2019-09-30T13:52:44.216755', '2018-06-21T12:26:47.000000', '2018-06-
↪21T12:26:49.000000', 0, 2)],
  dtype=[('transaction_time', '<M8[us]'), ('start_valid_time', '<M8[us]'), (
↪'end_valid_time', '<M8[us]'), ('start_idx', '<u8'), ('end_idx', '<u8')])
```

3.11 Dataset.transaction_idx

Dataset.**transaction_idx**
Indexes into the dataset by transaction ID.

3.12 Dataset.valid_times

Dataset.**valid_times**
Indexes into the dataset by valid time.

Examples

```
>>> with bth5.open(temp_h5, '/', mode='w', value_dtype=np.int64) as ds:
...     ds.write(np.datetime64("2018-06-21 12:26:47"), 2.0)
...     ds.write(np.datetime64("2018-06-21 12:26:49"), 2.0)
>>> with bth5.open(temp_h5, '/', mode='r', value_dtype=np.int64) as ds:
...     ds.valid_times[:]
...     ds.valid_times[np.datetime64("2018-06-21 12:26:47"):np.datetime64(
↪"2018-06-21 12:26:48")]
...     ds.valid_times[np.datetime64("2018-06-21 12:26:48"):]
...     ds.valid_times[:np.datetime64("2018-06-21 12:26:48")]
```

(continues on next page)

(continued from previous page)

```
...     ds.valid_times[np.datetime64("2018-06-21 12:26:49")]
array([(0, '2018-06-21T12:26:47.000000', 2),
       (0, '2018-06-21T12:26:49.000000', 2)],
      dtype=[('transaction_id', '<u8'), ('valid_time', '<M8[us]'), ('value', '
↪<i8')])
array([(0, '2018-06-21T12:26:47.000000', 2)],
      dtype=[('transaction_id', '<u8'), ('valid_time', '<M8[us]'), ('value', '
↪<i8')])
array([(0, '2018-06-21T12:26:49.000000', 2)],
      dtype=[('transaction_id', '<u8'), ('valid_time', '<M8[us]'), ('value', '
↪<i8')])
array([(0, '2018-06-21T12:26:47.000000', 2)],
      dtype=[('transaction_id', '<u8'), ('valid_time', '<M8[us]'), ('value', '
↪<i8')])
(0, '2018-06-21T12:26:49.000000', 2)
>>> with bth5.open(temp_h5, '/', mode='r', value_dtype=np.int64) as ds:
...     ds.valid_times[np.datetime64("2018-06-21 12:26:48")]
Traceback (most recent call last):
    ...
ValueError: The specified date was not found in the dataset, use interpolate_
↪value.
```

CHAPTER

FOUR

OPEN

bth5.**open**(*filename*, *path*, *mode='r'*, *value_dtype=None*, ***kwargs*)

Opens a bitemporal HDF5 dataset.

CHAPTER

FIVE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

b

INDEX

Symbols

B

C

D

F

I

O

R

T

V

W