02_Indexing_S3_MDFs_With_PonyORM-SQLite

April 21, 2020

1 Indexing MDFs with Pony ORM.

"SDK"/Guide to developing your own MDF management & analysis solution.

This can scale from one project with a bunch of log files sitting on your laptop to a full corporate S3 data-lake.

[1]:
```
import fsspec
from asammdf import MDF
```

[2]:
```
import fsspec
import os
s3_cfg = {
    "key": "mdf_minio_access_key",
    "secret": "mdf_minio_secret_key",
    "client_kwargs": {
        "endpoint_url": "http://minio:9000",
    },
}
fs = fsspec.filesystem("s3", **s3_cfg)
buckets = fs.ls("")
print(f"Buckets: {fs.ls('')}")
```

```
Buckets: ['mdfbucket-0', 'mdfbucket-1', 'mdfbucket-2', 'mdfbucket-3',
'mdfbucket-4', 'mdfbucket-5', 'mdfbucket-6', 'mdfbucket-7', 'mdfbucket-8',
'mdfbucket-9', 'test']
```

2 Walk Through All Files:

Walk through all S3 files and find the first one.

[3]:
```
import os
mdf_paths=list()

fs = fsspec.filesystem("s3", **s3_cfg)
for bucket in fs.ls(""):
    for root, dirs, files in fs.walk(bucket):
```

```
        for file in files:
            if file.lower().endswith(".mf4") or file.lower().endswith(".mdf"):
                mdf_paths.append(os.path.join(root, file))
```

Randomly pick a file for analysis/postprocessing.

[4]:
```
import random
mdf_path = random.choice(mdf_paths)
fs.info(mdf_path)
```

[4]:
```
{'Key':
'mdfbucket-2/DäsCarGmbh/MarsColonizer/f035d466-1a00-4914-a831-53aad01cfee5.mf4',
 'LastModified': datetime.datetime(2020, 4, 21, 20, 17, 51, 252000,
tzinfo=tzlocal()),
 'ETag': '"8c74d264743b47a2bae8b7254d57d207"',
 'Size': 1803648,
 'StorageClass': 'STANDARD',
 'Owner': {'DisplayName': '', 'ID': ''},
 'type': 'file',
 'size': 1803648,
 'name':
'mdfbucket-2/DäsCarGmbh/MarsColonizer/f035d466-1a00-4914-a831-53aad01cfee5.mf4'}
```

Determine what keys we need to make:

[5]:
```
for k,v in fs.info(mdf_path).items():
    print(f"{k}{type(v)}")
```

```
Key<class 'str'>
LastModified<class 'datetime.datetime'>
ETag<class 'str'>
Size<class 'int'>
StorageClass<class 'str'>
Owner<class 'dict'>
type<class 'str'>
size<class 'int'>
name<class 'str'>
```

Minimal skeleton Pony classes.

These can be adapted to suit your needs. For example the MDF class could contain the vehicle S/N or other information.

[6]:
```
import os

import pony.orm
from pony.orm.core import EntityMeta
from datetime import datetime
```

```
pony.orm.set_sql_debug(False)
db = pony.orm.Database()

if True:
    # Inmemory datatabase
    filename=":memory:"
else:
    # Or not.
    filename = os.path.abspath("mdf_index.sqlite")
    if os.path.exists(filename):
        os.unlink(filename)
# Bind
db.bind(
    provider="sqlite", filename=filename, create_db=True,
)

#For S3 Indexing
class MDF(db.Entity):
    # fs
    key = pony.orm.Required(str, unique=True,)
    last_modified = pony.orm.Optional(datetime,)
    etag = pony.orm.Optional(str,)
    size = pony.orm.Optional(int,)
    size_mb = pony.orm.Optional(float,)
    storage_class = pony.orm.Optional(str,)
    type = pony.orm.Optional(str,)
    name = pony.orm.Optional(str,)
    # Basename.
    basename = pony.orm.Optional(str,)
    channels = pony.orm.Set("Channel",)

class Channel(db.Entity):
    """Channel entity to represent a

    """
    name = pony.orm.Required(str, unique=True,)
    mdfs = pony.orm.Set("MDF",)


db.generate_mapping(create_tables=True)

def upsert(cls, get, set=None):
    """
    Interacting with Pony entities.

    :param cls: The actual entity class
    :param get: Identify the object (e.g. row) with this dictionary
```

```
    :param set: Additional fields to set if ```get``` returns nothing.
    :return:
    """
    # does the object exist
    assert isinstance(cls, EntityMeta), f"{cls} is not a database entity"

    # if no set dictionary has been specified
    set = set or {}

    if not cls.exists(**get):
        # make new object
        obj = cls(**set, **get)
        db.commit()
        return obj
    else:
        # get the existing object
        obj = cls.get(**get)
        for key, value in set.items():
            obj.__setattr__(key, value)
        return obj
```

[7]:
```
info = fs.info(mdf_path)
```

[8]:
```
info
```

[8]:
```
{'Key':
'mdfbucket-2/DäsCarGmbh/MarsColonizer/f035d466-1a00-4914-a831-53aad01cfee5.mf4',
 'LastModified': datetime.datetime(2020, 4, 21, 20, 17, 51, 252000,
tzinfo=tzlocal()),
 'ETag': '"8c74d264743b47a2bae8b7254d57d207"',
 'Size': 1803648,
 'StorageClass': 'STANDARD',
 'Owner': {'DisplayName': '', 'ID': ''},
 'type': 'file',
 'size': 1803648,
 'name':
'mdfbucket-2/DäsCarGmbh/MarsColonizer/f035d466-1a00-4914-a831-53aad01cfee5.mf4'}
```

[9]:
```
def index_mdf(mdf_path):
    info = fs.info(mdf_path)
    # Local File
    MDF_ = upsert(
    cls=MDF,
    get={"key": info["Key"]},
    set={
        "last_modified": info["LastModified"],
        "etag": info["ETag"],
```

```
        "size": info["size"],
        "size_mb": info["size"] / 1024 ** 2,
        "storage_class": info["StorageClass"],
        "type": info["type"],
        "name": info["name"],
        "basename": os.path.basename(info["name"])
        },
    )
    try:
        db.commit()
        return MDF_
    except:
        db.rollback()
        return None
```

[10]:
```
print(f"Indexing {len(mdf_paths)} MDF files")
```

```
Indexing 975 MDF files
```

Benchmark indexing times:

[11]:
```
%%timeit
for idx, mdf_path in enumerate(mdf_paths):
    index_mdf(mdf_path)
```

```
451 ms ± 1.31 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)
```

Index one file and get the orm object:

[12]:
```
mdf = index_mdf(mdf_path)
mdf
```

[12]:
```
MDF[265]
```

[13]:
```
# Add features to the class.
def __repr__(self):
    return f"MDF<{self.basename}>"
setattr(MDF, "__repr__", __repr__)
mdf
```

[13]:
```
MDF<f035d466-1a00-4914-a831-53aad01cfee5.mf4>
```

3 Adding Other Attributes

[To adapt to your own needs].

In the data example the company name & product are embedded in the mdf file path, we can extract them out to add to the database.

[14]:
```
os.path.basename(os.path.dirname(mdf.name))
```

[14]:
```
'MarsColonizer'
```

[15]:
```
os.path.basename(
    os.path.dirname(
        os.path.dirname(
            mdf.name
        )
    )
)
```

[15]:
```
'DäsCarGmbh'
```

[16]:
```
product = os.path.basename(os.path.dirname(mdf.name))
company = os.path.basename(
    os.path.dirname(
        os.path.dirname(
            mdf.name
        )
    )
)
```

Create property functions to return the product & company given a filepath:

[17]:
```
def product(self):
    return os.path.basename(os.path.dirname(self.name))
def company(self):
    return os.path.basename(
        os.path.dirname(
            os.path.dirname(
                mdf.name
            )
        )
    )
```

Doesn't exist:

[18]:
```
mdf.product
```

```
␣
↪---------------------------------------------------------------------------

AttributeError                            Traceback (most recent call␣
↪last)

<ipython-input-18-b2328f6df9d4> in <module>
```

```
----> 1 mdf.product


AttributeError: 'MDF' object has no attribute 'product'
```

Attach the properties to the MDF class:

[19]:
```
setattr(MDF, 'product', property(product))
setattr(MDF, 'company', property(company))
```

[20]:
```
mdf.product
```

[20]:
```
'MarsColonizer'
```

[21]:
```
mdf.company
```

[21]:
```
'DäsCarGmbh'
```

[22]:
```
db.rollback()
mdfs = pony.orm.select(m for m in MDF)

for mdf in mdfs:
    break
mdf
```

[22]:
```
MDF<085de752-222e-44ce-a1d9-53c6352cd038.mf4>
```

4 MDF Analysis.

The above analysis was based on just the file attributes and attributes embedded in the filename (Company, Product).

This step uses `asammdf` to index MDF attributes such as which channels each file contains.

[23]:
```
import asammdf
```

[24]:
```
with fs.open(mdf.name) as fid:
    mdf_ = asammdf.MDF(fid)
```

[25]:
```
mdf_.channels_db
```

[25]:
```
{'time': ((0, 0),),
 'engine_speed': ((0, 1),),
 'engine_speed_desired': ((0, 2),),
 'vehicle_speed': ((0, 3),),
 'coolant_temp': ((0, 4),),
```

```
 'longitude': ((0, 5),),
 'latitude': ((0, 6),),
 'power': ((0, 7),),
 'efficiency': ((0, 8),),
 'ADAS5_failure': ((0, 9),),
 'X': ((0, 10),),
 'Y': ((0, 11),),
 'Z': ((0, 12),)}
```

[26]:
```
channels=list()
for channel in mdf_.channels_db.keys():
    channel_ = upsert(Channel, {"name": channel})
    channels.append(channel_)
```

[27]:
```
channels
```

[27]:
```
[Channel[1],
 Channel[2],
 Channel[3],
 Channel[4],
 Channel[5],
 Channel[6],
 Channel[7],
 Channel[8],
 Channel[9],
 Channel[10],
 Channel[11],
 Channel[12],
 Channel[13]]
```

[28]:
```
mdf.key
```

[28]:
```
'mdfbucket-0/ABMøøse/BoatyMcBoatface/085de752-222e-44ce-a1d9-53c6352cd038.mf4'
```

[29]:
```
def process_channels(mdfs):
    """Given a list of MDF files, process the channels

    """
    for mdf in mdfs:
        # Open the MDF file.
        try:
            with fs.open(mdf.name, "rb") as fid:
                mdf_ = asammdf.MDF(fid)
        except asammdf.mdf.MdfException:
            # Bad file!
            continue
        #
```

```
        channels=list()
        # Loop through each of the channels in the database.
        for channel in mdf_.channels_db.keys():
            channel_ = upsert(Channel, {"name": channel})
            channels.append(channel_)
        MDF_ = upsert(
        cls=MDF,
        get={"key": mdf.name},
        set={
            "channels": channels
            },
        )
        db.commit()
```

[30]:
```
mdf
```

[30]: MDF<085de752-222e-44ce-a1d9-53c6352cd038.mf4>

[31]:
```
process_channels([mdf])
```

[32]:
```
%%timeit
# How long does it take to insert channels for 10 mdfs
process_channels(mdfs[:10])
```

```
175 ms ± 2.17 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)
```

[33]:
```
# Insert channel information for each of the MDFs
process_channels(mdfs)
```

[34]:
```
channels = pony.orm.select(m for m in Channel)
```

[35]:
```
for channel in channels:
    break
```

[36]:
```
len(list(channel.mdfs))
```

[36]: 975

How many MDFs contain each channel?

[37]:
```
for channel in channels:
    n = len(list(channel.mdfs))
    print(f"{channel.name}: {n}")
```

```
time: 975
engine_speed: 967
engine_speed_desired: 964
vehicle_speed: 969
```

```
coolant_temp: 965
longitude: 959
latitude: 969
power: 964
efficiency: 965
ADAS5_failure: 967
X: 959
Y: 966
Z: 962
```

5 Further Development.

Take the minimal skeleton from above and develop it into a more 'useful' example with some additional python.

[38]:
```python
import os

import pony.orm
from pony.orm.core import EntityMeta
from datetime import datetime

pony.orm.set_sql_debug(False)
db = pony.orm.Database()

# Work in memory, or not.
if True:
    # Inmemory datatabase
    filename=":memory:"
else:
    # Or not.
    filename = os.path.abspath("mdf_index.sqlite")
    if os.path.exists(filename):
        os.unlink(filename)
# Bind
db.bind(
    provider="sqlite", filename=filename, create_db=True,
)


# For Local Indexing.
class MDF(db.Entity):
    # Filesystem Bits.
    key = pony.orm.Required(str, unique=True,)
    last_modified = pony.orm.Optional(datetime,)
    etag = pony.orm.Optional(str,)
    size = pony.orm.Optional(int,)
```

```
    size_mb = pony.orm.Optional(float,)
    storage_class = pony.orm.Optional(str,)
    type = pony.orm.Optional(str,)
    name = pony.orm.Optional(str,)

    # Pre-calculated bits.
    basename = pony.orm.Optional(str,)

    product = pony.orm.Optional(str,)
    company = pony.orm.Optional(str,)

    # ASAM MDF Bits.
    version = pony.orm.Optional(str,)
    channels = pony.orm.Set("Channel",)

    def __repr__(self):
        return f"MDF<{self.id},{self.product},{self.company},Ch:{len(self.
↪channels)}>"

class Channel(db.Entity):
    """Channel entity to represent a

    """
    name = pony.orm.Required(str, unique=True,)
    mdfs = pony.orm.Set("MDF",)

    def __repr__(self):
        return f"Channel<{self.id},{self.name}>"

def upsert(cls, get, set=None):
    """
    Interacting with Pony entities.

    :param cls: The actual entity class
    :param get: Identify the object (e.g. row) with this dictionary
    :param set: Additional fields to set if ```get``` returns nothing.
    :return:
    """
    # does the object exist
    assert isinstance(cls, EntityMeta), f"{cls} is not a database entity"

    # if no set dictionary has been specified
    set = set or {}

    if not cls.exists(**get):
        # make new object
        return cls(**set, **get)
```

```
    else:
        # get the existing object
        obj = cls.get(**get)
        for key, value in set.items():
            obj.__setattr__(key, value)
        return obj


db.generate_mapping(create_tables=True)

def index_mdf(mdf_path):
    info = fs.info(mdf_path)
    # Local File
    MDF_ = upsert(
    cls=MDF,
    get={"key": info["Key"]},
    set={
        "last_modified": info["LastModified"],
        "etag": info["ETag"],
        "size": info["size"],
        "size_mb": info["size"] / 1024 ** 2,
        "storage_class": info["StorageClass"],
        "type": info["type"],
        "name": info["name"],
        "basename": os.path.basename(info["name"])
        },
    )
    try:
        db.commit()
        return MDF_
    except:
        db.rollback()
        return None
import asammdf
def index_channels(mdf):
    """Given a MDF files, process the channels

    """
    # Open the MDF file.
    mdf_ = asammdf.MDF(mdf.name)
    #
    channels=list()
    # Loop through each of the channels in the database.
    for channel in mdf_.channels_db.keys():
        channel_ = upsert(Channel, {"name": channel})
        channels.append(channel_)
    MDF_ = upsert(
```

```
    cls=MDF,
    get={"name": mdf.name},
    set={
        "channels": channels
        },
    )
    try:
        db.commit()
        return channels
    except:
        db.rollback()
        return None

def index_mdf_info(mdf):
    """ Index company and product information in the database from the filename.
 ↪"""
    product = os.path.basename(os.path.dirname(mdf.name))
    company = os.path.basename(
        os.path.dirname(
            os.path.dirname(
                mdf.name
            )
        )
    )
    # Local File
    MDF_ = upsert(
    cls=MDF,
    get={"name": mdf.name},
    set={
        "product": product,
        "company": company,
        },
    )
    try:
        db.commit()
        return MDF_
    except:
        db.rollback()
        return None
```

[39]:
```
import os
mdf_paths=list()

fs = fsspec.filesystem("s3", **s3_cfg)
for bucket in fs.ls(""):
    for root, dirs, files in fs.walk(bucket):
        for file in files:
```

```
            if file.lower().endswith(".mf4") or file.lower().endswith(".mdf"):
                mdf_paths.append(os.path.join(root, file))
print(f"Found {len(mdf_paths)} MDF files")
```

```
Found 975 MDF files
```

Randomly pick a file for analysis.

[40]:
```
mdf_path = random.choice(mdf_paths)
mdf_path
```

[40]:
```
'mdfbucket-2/ABMøøse/Transmission/d83018c5-d19a-434b-8e2f-cfc1009d950a.mf4'
```

Insert the MDF file into the database.

[Notice the **repr** string isn't fully populated, the data isn't yet in the database]

[41]:
```
mdf = index_mdf(mdf_path)
mdf
```

[41]:
```
MDF<1,,,Ch:0>
```

Index the product and company name of the mdf

[42]:
```
index_mdf_info(mdf)
mdf
```

```
      ␣
↪---------------------------------------------------------------------------

       UnrepeatableReadError                     Traceback (most recent call␣
↪last)

       <ipython-input-42-136b14ffb051> in <module>
   ----> 1 index_mdf_info(mdf)
         2 mdf


       <ipython-input-38-06e6b36ea2c2> in index_mdf_info(mdf)
       152     set={
       153         "product": product,
   --> 154         "company": company,
       155         },
       156     )


       <ipython-input-38-06e6b36ea2c2> in upsert(cls, get, set)
        73     set = set or {}
```

```
        74
   ---> 75     if not cls.exists(**get):
        76         # make new object
        77         return cls(**set, **get)


       /opt/conda/lib/python3.7/site-packages/pony/orm/core.py in␣
↪exists(entity, *args, **kwargs)


       /opt/conda/lib/python3.7/site-packages/pony/utils/utils.py in␣
↪cut_traceback(func, *args, **kwargs)
        75             module_name = tb.tb_frame.f_globals.get('__name__') or ''
        76             if module_name.startswith('pony.utils') and tb.tb_frame.
↪f_code.co_name == 'throw':
   ---> 77                 reraise(exc_type, exc, last_pony_tb)
        78             reraise(exc_type, exc, full_tb)
        79         finally:


       /opt/conda/lib/python3.7/site-packages/pony/utils/utils.py in␣
↪reraise(***failed resolving arguments***)
        93 else:
        94     def reraise(exc_type, exc, tb):
   ---> 95         try: raise exc.with_traceback(tb)
        96         finally: del exc, tb
        97


       /opt/conda/lib/python3.7/site-packages/pony/orm/core.py in _db_set_(obj,␣
↪avdict, unpickling)
      4934                 throw(UnrepeatableReadError,
      4935                       'Value of %s.%s for %s was updated outside of␣
↪current transaction (was: %r, now: %r)'
   -> 4936                       % (obj.__class__.__name__, attr.name, obj,␣
↪old_dbval, new_dbval))
      4937
      4938             if attr.reverse: attr.db_update_reverse(obj, old_dbval,␣
↪new_dbval)


       /opt/conda/lib/python3.7/site-packages/pony/utils/utils.py in␣
↪throw(exc_type, *args, **kwargs)
       106             raise exc
       107         else:
   --> 108             raise exc  # Set "pony.options.CUT_TRACEBACK = False" to␣
↪see full traceback
```

```
109     finally: del exc
110
```

```
UnrepeatableReadError: Value of MDF.last_modified for MDF<1,,,Ch:0> was␣
↪updated outside of current transaction (was: datetime.datetime(2020, 4, 21,␣
↪20, 18, 13, 396000, tzinfo=tzlocal()), now: datetime.datetime(2020, 4, 21, 20,␣
↪18, 13, 396000))
```

Index the channels.

[43]:
```
index_channels(mdf)
mdf
```

```
␣
↪---------------------------------------------------------------------------
```

```
MdfException                                Traceback (most recent call␣
↪last)

<ipython-input-43-4f52964d6366> in <module>
----> 1 index_channels(mdf)
      2 mdf


<ipython-input-38-06e6b36ea2c2> in index_channels(mdf)
    115     """
    116     # Open the MDF file.
--> 117     mdf_ = asammdf.MDF(mdf.name)
    118     #
    119     channels=list()


/opt/conda/lib/python3.7/site-packages/asammdf/mdf.py in __init__(self,␣
↪name, version, **kwargs)
    106                     file_stream = open(name, "rb")
    107                 else:
--> 108                     raise MdfException(f'File "{name}" does not␣
↪exist')
    109             file_stream.seek(0)
    110             magic_header = file_stream.read(8)


MdfException: File "mdfbucket-2/ABMøøse/Transmission/
↪d83018c5-d19a-434b-8e2f-cfc1009d950a.mf4" does not exist
```

[44]:
```
%%timeit
for mdf_path in mdf_paths[:10]:
    index_mdf(mdf_path)
```

```
4.77 ms ± 42.4 µs per loop (mean ± std. dev. of 7 runs, 100 loops each)
```

[45]:
```
pony.orm.select(m for m in MDF)[0:10]
```

```
␣
↪---------------------------------------------------------------------------

       UnrepeatableReadError                     Traceback (most recent call␣
↪last)

       <ipython-input-45-47701663352c> in <module>
   ----> 1 pony.orm.select(m for m in MDF)[0:10]


       /opt/conda/lib/python3.7/site-packages/pony/orm/core.py in␣
↪__getitem__(query, key)


       /opt/conda/lib/python3.7/site-packages/pony/utils/utils.py in␣
↪cut_traceback(func, *args, **kwargs)
        75             module_name = tb.tb_frame.f_globals.get('__name__') or ''
        76             if module_name.startswith('pony.utils') and tb.tb_frame.
↪f_code.co_name == 'throw':
   ---> 77                 reraise(exc_type, exc, last_pony_tb)
        78             reraise(exc_type, exc, full_tb)
        79         finally:


       /opt/conda/lib/python3.7/site-packages/pony/utils/utils.py in␣
↪reraise(***failed resolving arguments***)
        93 else:
        94     def reraise(exc_type, exc, tb):
   ---> 95         try: raise exc.with_traceback(tb)
        96         finally: del exc, tb
        97


       /opt/conda/lib/python3.7/site-packages/pony/orm/core.py in _db_set_(obj,␣
↪avdict, unpickling)
      4934                 throw(UnrepeatableReadError,
      4935                       'Value of %s.%s for %s was updated outside of␣
↪current transaction (was: %r, now: %r)'
```

```
   -> 4936                         % (obj.__class__.__name__, attr.name, obj,␣
↪old_dbval, new_dbval))
      4937
      4938             if attr.reverse: attr.db_update_reverse(obj, old_dbval,␣
↪new_dbval)


      /opt/conda/lib/python3.7/site-packages/pony/utils/utils.py in␣
↪throw(exc_type, *args, **kwargs)
      106             raise exc
      107         else:
   --> 108             raise exc  # Set "pony.options.CUT_TRACEBACK = False" to␣
↪see full traceback
      109     finally: del exc
      110


      UnrepeatableReadError: Value of MDF.last_modified for MDF<1,,,Ch:0> was␣
↪updated outside of current transaction (was: datetime.datetime(2020, 4, 21,␣
↪20, 18, 13, 396000, tzinfo=tzlocal()), now: datetime.datetime(2020, 4, 21, 20,␣
↪18, 13, 396000))
```

[46]:
```
%%timeit
for mdf in pony.orm.select(m for m in MDF)[0:10]:
    index_mdf_info(mdf)
```

```
      ␣
↪---------------------------------------------------------------------------

      UnrepeatableReadError                     Traceback (most recent call␣
↪last)

      <ipython-input-46-127b8e2d94fc> in <module>
   ----> 1 get_ipython().run_cell_magic('timeit', '', 'for mdf in pony.orm.
↪select(m for m in MDF)[0:10]:\n    index_mdf_info(mdf)\n')


      /opt/conda/lib/python3.7/site-packages/IPython/core/interactiveshell.py␣
↪in run_cell_magic(self, magic_name, line, cell)
      2360             with self.builtin_trap:
      2361                 args = (magic_arg_s, cell)
   -> 2362                 result = fn(*args, **kwargs)
      2363             return result
      2364
```

```
       <decorator-gen-60> in timeit(self, line, cell, local_ns)


       /opt/conda/lib/python3.7/site-packages/IPython/core/magic.py in␣
↪<lambda>(f, *a, **k)
       185     # but it's overkill for just that one bit of state.
       186     def magic_deco(arg):
   --> 187         call = lambda f, *a, **k: f(*a, **k)
       188
       189         if callable(arg):


       /opt/conda/lib/python3.7/site-packages/IPython/core/magics/execution.py␣
↪in timeit(self, line, cell, local_ns)
      1158             for index in range(0, 10):
      1159                 number = 10 ** index
   -> 1160                 time_number = timer.timeit(number)
      1161                 if time_number >= 0.2:
      1162                     break


       /opt/conda/lib/python3.7/site-packages/IPython/core/magics/execution.py␣
↪in timeit(self, number)
       167         gc.disable()
       168         try:
   --> 169             timing = self.inner(it, self.timer)
       170         finally:
       171             if gcold:


       <magic-timeit> in inner(_it, _timer)


       /opt/conda/lib/python3.7/site-packages/pony/orm/core.py in␣
↪__getitem__(query, key)


       /opt/conda/lib/python3.7/site-packages/pony/utils/utils.py in␣
↪cut_traceback(func, *args, **kwargs)
        75             module_name = tb.tb_frame.f_globals.get('__name__') or ''
        76             if module_name.startswith('pony.utils') and tb.tb_frame.
↪f_code.co_name == 'throw':
   ---> 77                 reraise(exc_type, exc, last_pony_tb)
        78             reraise(exc_type, exc, full_tb)
        79         finally:
```

```
       /opt/conda/lib/python3.7/site-packages/pony/utils/utils.py in␣
↪reraise(***failed resolving arguments***)
        93 else:
        94     def reraise(exc_type, exc, tb):
   ---> 95         try: raise exc.with_traceback(tb)
        96         finally: del exc, tb
        97

       /opt/conda/lib/python3.7/site-packages/pony/orm/core.py in _db_set_(obj,␣
↪avdict, unpickling)
      4934                 throw(UnrepeatableReadError,
      4935                       'Value of %s.%s for %s was updated outside of␣
↪current transaction (was: %r, now: %r)'
   -> 4936                       % (obj.__class__.__name__, attr.name, obj,␣
↪old_dbval, new_dbval))
      4937
      4938             if attr.reverse: attr.db_update_reverse(obj, old_dbval,␣
↪new_dbval)

       /opt/conda/lib/python3.7/site-packages/pony/utils/utils.py in␣
↪throw(exc_type, *args, **kwargs)
       106             raise exc
       107         else:
   --> 108             raise exc  # Set "pony.options.CUT_TRACEBACK = False" to␣
↪see full traceback
       109     finally: del exc
       110

       UnrepeatableReadError: Value of MDF.last_modified for MDF<1,,,Ch:0> was␣
↪updated outside of current transaction (was: datetime.datetime(2020, 4, 21,␣
↪20, 18, 13, 396000, tzinfo=tzlocal()), now: datetime.datetime(2020, 4, 21, 20,␣
↪18, 13, 396000))
```

[47]:
```
pony.orm.select(m for m in MDF)[0:10]
```

```
      ␣
↪---------------------------------------------------------------------------

       UnrepeatableReadError                     Traceback (most recent call␣
↪last)
```

```
       <ipython-input-47-47701663352c> in <module>
   ----> 1 pony.orm.select(m for m in MDF)[0:10]


       /opt/conda/lib/python3.7/site-packages/pony/orm/core.py in␣
↪__getitem__(query, key)


       /opt/conda/lib/python3.7/site-packages/pony/utils/utils.py in␣
↪cut_traceback(func, *args, **kwargs)
        75             module_name = tb.tb_frame.f_globals.get('__name__') or ''
        76             if module_name.startswith('pony.utils') and tb.tb_frame.
↪f_code.co_name == 'throw':
   ---> 77                 reraise(exc_type, exc, last_pony_tb)
        78             reraise(exc_type, exc, full_tb)
        79         finally:


       /opt/conda/lib/python3.7/site-packages/pony/utils/utils.py in␣
↪reraise(***failed resolving arguments***)
        93 else:
        94     def reraise(exc_type, exc, tb):
   ---> 95         try: raise exc.with_traceback(tb)
        96         finally: del exc, tb
        97


       /opt/conda/lib/python3.7/site-packages/pony/orm/core.py in _db_set_(obj,␣
↪avdict, unpickling)
      4934                 throw(UnrepeatableReadError,
      4935                       'Value of %s.%s for %s was updated outside of␣
↪current transaction (was: %r, now: %r)'
   -> 4936                       % (obj.__class__.__name__, attr.name, obj,␣
↪old_dbval, new_dbval))
      4937
      4938             if attr.reverse: attr.db_update_reverse(obj, old_dbval,␣
↪new_dbval)


       /opt/conda/lib/python3.7/site-packages/pony/utils/utils.py in␣
↪throw(exc_type, *args, **kwargs)
       106             raise exc
       107         else:
   --> 108             raise exc  # Set "pony.options.CUT_TRACEBACK = False" to␣
↪see full traceback
       109     finally: del exc
       110
```

```
        UnrepeatableReadError: Value of MDF.last_modified for MDF<1,,,Ch:0> was␣
↪updated outside of current transaction (was: datetime.datetime(2020, 4, 21,␣
↪20, 18, 13, 396000, tzinfo=tzlocal()), now: datetime.datetime(2020, 4, 21, 20,␣
↪18, 13, 396000))
```

[48]:
```
%%timeit
for mdf in pony.orm.select(m for m in MDF)[0:10]:
    index_channels(mdf)
```

```
      ␣
↪---------------------------------------------------------------------------

        UnrepeatableReadError                     Traceback (most recent call␣
↪last)

        <ipython-input-48-8351b7959225> in <module>
      ----> 1 get_ipython().run_cell_magic('timeit', '', 'for mdf in pony.orm.
↪select(m for m in MDF)[0:10]:\n    index_channels(mdf)\n')


        /opt/conda/lib/python3.7/site-packages/IPython/core/interactiveshell.py␣
↪in run_cell_magic(self, magic_name, line, cell)
        2360             with self.builtin_trap:
        2361                 args = (magic_arg_s, cell)
      -> 2362                 result = fn(*args, **kwargs)
        2363             return result
        2364


        <decorator-gen-60> in timeit(self, line, cell, local_ns)


        /opt/conda/lib/python3.7/site-packages/IPython/core/magic.py in␣
↪<lambda>(f, *a, **k)
        185     # but it's overkill for just that one bit of state.
        186     def magic_deco(arg):
      --> 187         call = lambda f, *a, **k: f(*a, **k)
        188
        189         if callable(arg):


        /opt/conda/lib/python3.7/site-packages/IPython/core/magics/execution.py␣
↪in timeit(self, line, cell, local_ns)
```

```
      1158             for index in range(0, 10):
      1159                 number = 10 ** index
   -> 1160                 time_number = timer.timeit(number)
      1161                 if time_number >= 0.2:
      1162                     break


      /opt/conda/lib/python3.7/site-packages/IPython/core/magics/execution.py␣
↪in timeit(self, number)
       167         gc.disable()
       168         try:
   --> 169             timing = self.inner(it, self.timer)
       170         finally:
       171             if gcold:


      <magic-timeit> in inner(_it, _timer)


      /opt/conda/lib/python3.7/site-packages/pony/orm/core.py in␣
↪__getitem__(query, key)


      /opt/conda/lib/python3.7/site-packages/pony/utils/utils.py in␣
↪cut_traceback(func, *args, **kwargs)
        75             module_name = tb.tb_frame.f_globals.get('__name__') or ''
        76             if module_name.startswith('pony.utils') and tb.tb_frame.
↪f_code.co_name == 'throw':
   ---> 77                 reraise(exc_type, exc, last_pony_tb)
        78             reraise(exc_type, exc, full_tb)
        79         finally:


      /opt/conda/lib/python3.7/site-packages/pony/utils/utils.py in␣
↪reraise(***failed resolving arguments***)
        93 else:
        94     def reraise(exc_type, exc, tb):
   ---> 95         try: raise exc.with_traceback(tb)
        96         finally: del exc, tb
        97


      /opt/conda/lib/python3.7/site-packages/pony/orm/core.py in _db_set_(obj,␣
↪avdict, unpickling)
      4934                 throw(UnrepeatableReadError,
      4935                       'Value of %s.%s for %s was updated outside of␣
↪current transaction (was: %r, now: %r)'
```

```
   -> 4936                         % (obj.__class__.__name__, attr.name, obj,␣
↪old_dbval, new_dbval))
      4937
      4938             if attr.reverse: attr.db_update_reverse(obj, old_dbval,␣
↪new_dbval)


       /opt/conda/lib/python3.7/site-packages/pony/utils/utils.py in␣
↪throw(exc_type, *args, **kwargs)
       106             raise exc
       107         else:
   --> 108             raise exc  # Set "pony.options.CUT_TRACEBACK = False" to␣
↪see full traceback
       109     finally: del exc
       110


       UnrepeatableReadError: Value of MDF.last_modified for MDF<1,,,Ch:0> was␣
↪updated outside of current transaction (was: datetime.datetime(2020, 4, 21,␣
↪20, 18, 13, 396000, tzinfo=tzlocal()), now: datetime.datetime(2020, 4, 21, 20,␣
↪18, 13, 396000))
```

This task can easily be distributed with celery or rq

Asynchronous Task Execution In Python

[49]:
```
for mdf_path in mdf_paths:
    mdf = index_mdf(mdf_path)
    index_mdf_info(mdf)
    index_channels(mdf)
```

```
      ␣
↪---------------------------------------------------------------------------

       UnrepeatableReadError                       Traceback (most recent call␣
↪last)

       <ipython-input-49-0cf54cb5e9ce> in <module>
         1 for mdf_path in mdf_paths:
         2     mdf = index_mdf(mdf_path)
   ----> 3     index_mdf_info(mdf)
         4     index_channels(mdf)


       <ipython-input-38-06e6b36ea2c2> in index_mdf_info(mdf)
       152     set={
```

```
      153         "product": product,
--> 154         "company": company,
      155         },
      156     )


      <ipython-input-38-06e6b36ea2c2> in upsert(cls, get, set)
       73     set = set or {}
       74
---> 75     if not cls.exists(**get):
       76         # make new object
       77         return cls(**set, **get)


      /opt/conda/lib/python3.7/site-packages/pony/orm/core.py in␣
↪exists(entity, *args, **kwargs)


      /opt/conda/lib/python3.7/site-packages/pony/utils/utils.py in␣
↪cut_traceback(func, *args, **kwargs)
       75             module_name = tb.tb_frame.f_globals.get('__name__') or ''
       76             if module_name.startswith('pony.utils') and tb.tb_frame.
↪f_code.co_name == 'throw':
---> 77                 reraise(exc_type, exc, last_pony_tb)
       78             reraise(exc_type, exc, full_tb)
       79         finally:


      /opt/conda/lib/python3.7/site-packages/pony/utils/utils.py in␣
↪reraise(***failed resolving arguments***)
       93 else:
       94     def reraise(exc_type, exc, tb):
---> 95         try: raise exc.with_traceback(tb)
       96         finally: del exc, tb
       97


      /opt/conda/lib/python3.7/site-packages/pony/orm/core.py in _db_set_(obj,␣
↪avdict, unpickling)
     4934                 throw(UnrepeatableReadError,
     4935                       'Value of %s.%s for %s was updated outside of␣
↪current transaction (was: %r, now: %r)'
-> 4936                       % (obj.__class__.__name__, attr.name, obj,␣
↪old_dbval, new_dbval))
     4937
     4938             if attr.reverse: attr.db_update_reverse(obj, old_dbval,␣
↪new_dbval)
```

```
       /opt/conda/lib/python3.7/site-packages/pony/utils/utils.py in␣
↪throw(exc_type, *args, **kwargs)
       106             raise exc
       107         else:
   --> 108             raise exc  # Set "pony.options.CUT_TRACEBACK = False" to␣
↪see full traceback
       109     finally: del exc
       110

       UnrepeatableReadError: Value of MDF.last_modified for MDF<2,,,Ch:0> was␣
↪updated outside of current transaction (was: datetime.datetime(2020, 4, 21,␣
↪20, 17, 40, 220000, tzinfo=tzlocal()), now: datetime.datetime(2020, 4, 21, 20,␣
↪17, 40, 220000))
```

[50]:
```
channels = pony.orm.select(c for c in Channel)
```

[51]:
```
for channel in channels:
    break
```

How many MDF files have been indexed?

[52]:
```
len(channel.mdfs)
```

[52]: 962

How many bytes of MDF files have been indexed?

[53]:
```
pony.orm.sum(m.size for m in MDF)
```

[53]: 22815376

How many GB of MDF files have been indexed?

[54]:
```
pony.orm.sum(m.size for m in MDF)/1024**3
```

[54]: 0.021248474717140198

Find the biggest MDF file to analyze:

[55]:
```
q = pony.orm.select(mdf for mdf in MDF).order_by(lambda: pony.orm.desc(mdf.
↪size))
```

[56]:
```
q[0:5]
```

```
      ␣
↪--------------------------------------------------------------------------

      UnrepeatableReadError                     Traceback (most recent call␣
↪last)

      <ipython-input-56-7507018d6cd7> in <module>
   ----> 1 q[0:5]


      /opt/conda/lib/python3.7/site-packages/pony/orm/core.py in␣
↪__getitem__(query, key)


      /opt/conda/lib/python3.7/site-packages/pony/utils/utils.py in␣
↪cut_traceback(func, *args, **kwargs)
       75             module_name = tb.tb_frame.f_globals.get('__name__') or ''
       76             if module_name.startswith('pony.utils') and tb.tb_frame.
↪f_code.co_name == 'throw':
   ---> 77                 reraise(exc_type, exc, last_pony_tb)
       78             reraise(exc_type, exc, full_tb)
       79         finally:


      /opt/conda/lib/python3.7/site-packages/pony/utils/utils.py in␣
↪reraise(***failed resolving arguments***)
       93 else:
       94     def reraise(exc_type, exc, tb):
   ---> 95         try: raise exc.with_traceback(tb)
       96         finally: del exc, tb
       97


      /opt/conda/lib/python3.7/site-packages/pony/orm/core.py in _db_set_(obj,␣
↪avdict, unpickling)
     4934                 throw(UnrepeatableReadError,
     4935                       'Value of %s.%s for %s was updated outside of␣
↪current transaction (was: %r, now: %r)'
   -> 4936                       % (obj.__class__.__name__, attr.name, obj,␣
↪old_dbval, new_dbval))
     4937
     4938             if attr.reverse: attr.db_update_reverse(obj, old_dbval,␣
↪new_dbval)
```

```
      /opt/conda/lib/python3.7/site-packages/pony/utils/utils.py in␣
↪throw(exc_type, *args, **kwargs)
      106             raise exc
      107         else:
  --> 108             raise exc  # Set "pony.options.CUT_TRACEBACK = False" to␣
↪see full traceback
      109     finally: del exc
      110

      UnrepeatableReadError: Value of MDF.last_modified for MDF<7,,,Ch:0> was␣
↪updated outside of current transaction (was: datetime.datetime(2020, 4, 21,␣
↪20, 19, 3, 712000, tzinfo=tzlocal()), now: datetime.datetime(2020, 4, 21, 20,␣
↪19, 3, 712000))
```

[57]:
```
mdf_ = asammdf.MDF(list(q)[0].name)
```

```
     ␣
↪---------------------------------------------------------------------------

      UnrepeatableReadError                     Traceback (most recent call␣
↪last)

      <ipython-input-57-ca350ca81e3f> in <module>
  ----> 1 mdf_ = asammdf.MDF(list(q)[0].name)

      /opt/conda/lib/python3.7/site-packages/pony/orm/core.py in __len__(query)

      /opt/conda/lib/python3.7/site-packages/pony/utils/utils.py in␣
↪cut_traceback(func, *args, **kwargs)
       75             module_name = tb.tb_frame.f_globals.get('__name__') or ''
       76             if module_name.startswith('pony.utils') and tb.tb_frame.
↪f_code.co_name == 'throw':
  ---> 77                 reraise(exc_type, exc, last_pony_tb)
       78             reraise(exc_type, exc, full_tb)
       79         finally:

      /opt/conda/lib/python3.7/site-packages/pony/utils/utils.py in␣
↪reraise(***failed resolving arguments***)
       93 else:
       94     def reraise(exc_type, exc, tb):
  ---> 95         try: raise exc.with_traceback(tb)
```

```
 96         finally: del exc, tb
 97

/opt/conda/lib/python3.7/site-packages/pony/orm/core.py in _db_set_(obj,␣
↪avdict, unpickling)
   4934                 throw(UnrepeatableReadError,
   4935                       'Value of %s.%s for %s was updated outside of␣
↪current transaction (was: %r, now: %r)'
-> 4936                       % (obj.__class__.__name__, attr.name, obj,␣
↪old_dbval, new_dbval))
   4937
   4938             if attr.reverse: attr.db_update_reverse(obj, old_dbval,␣
↪new_dbval)

/opt/conda/lib/python3.7/site-packages/pony/utils/utils.py in␣
↪throw(exc_type, *args, **kwargs)
    106             raise exc
    107         else:
--> 108             raise exc  # Set "pony.options.CUT_TRACEBACK = False" to␣
↪see full traceback
    109     finally: del exc
    110

UnrepeatableReadError: Value of MDF.last_modified for MDF<7,,,Ch:0> was␣
↪updated outside of current transaction (was: datetime.datetime(2020, 4, 21,␣
↪20, 19, 3, 712000, tzinfo=tzlocal()), now: datetime.datetime(2020, 4, 21, 20,␣
↪19, 3, 712000))
```

[58]:
```
for chan in mdf_.iter_channels():
    chan.plot()
```

```
  ␣
↪---------------------------------------------------------------------------

KeyError                                  Traceback (most recent call␣
↪last)

<ipython-input-58-71a12ac4107e> in <module>
----> 1 for chan in mdf_.iter_channels():
      2     chan.plot()
```

```
       /opt/conda/lib/python3.7/site-packages/asammdf/mdf.py in␣
↪iter_channels(self, skip_master, copy_master)
      1876             ]
      1877
   -> 1878             channels = self.select(channels, copy_master=copy_master)
      1879
      1880             yield from channels


       /opt/conda/lib/python3.7/site-packages/asammdf/mdf.py in select(self,␣
↪channels, record_offset, raw, copy_master, ignore_value2text_conversions,␣
↪record_count, validate)
      2276             indexes.append(self._validate_channel_selection(*item))
      2277
   -> 2278         signals = [output_signals[pair] for pair in indexes]
      2279
      2280         if copy_master:


       /opt/conda/lib/python3.7/site-packages/asammdf/mdf.py in <listcomp>(.0)
      2276             indexes.append(self._validate_channel_selection(*item))
      2277
   -> 2278         signals = [output_signals[pair] for pair in indexes]
      2279
      2280         if copy_master:


       KeyError: (0, 1)
```

[ ]: