MobilityDB SQLAlchemy

Release 0.4

Bommakanti Krishna Chaitanya

Sep 02, 2020
CONTENTS:

1 Quickstart 1
2 Inserting temporal data 3
3 Inserting TGeomPoint data 5
4 Inserting TGeomPoint data, using movingpandas 7
5 Using MobilityDB functions 9
6 Using MobilityDB operators 11
7 Using MobilityDB ranges 13
8 Making use of movingpandas Trajectory data structure 15
9 Operators 17
10 Installation 21
11 Indices and tables 23
Python Module Index 25
Index 27
Listing 1: Example usage of the TGeomPoint class as a column in a table defined using SQLAlchemy’s declarative API

```python
from mobilitydb_sqlalchemy import TGeomPoint

from sqlalchemy import Column, Integer
from sqlalchemy.ext.declarative import declarative_base

Base = declarative_base()

class Trips(Base):
    __tablename__ = "test_table_trips_01"
    car_id = Column(Integer, primary_key=True)
    trip_id = Column(Integer, primary_key=True)
    trip = Column(TGeomPoint)

trips = session.query(Trips).all()

# Querying using MobilityDB functions, for example - valueAtTimestamp
session.query(
    Trips.car_id,
    func.asText(func.valueAtTimestamp(Trips.trip, datetime.datetime(2012, 1, 8, 10, 0)))
).all()
```
mobilitydb-sqlalchemy lets you use pandas DataFrame (which are great for timeseries data) while you are in the Python world, and translates it back and for to temporal types defined in mobilitydb.

A point to note here is that we assume that the DataFrame’s columns are named “value” (except in case of TGeomPoint where it is “geometry”) and “t” for the data and the timestamp respectively.

Here we show how we can store numeric data which changes over time (i.e. tfloat), using the mobilitydb_sqlalchemy.types.TFloat.TFloat class.

Running the following code will create a new table with a tfloat column, and insert one row of hardcoded data into it.

```python
import datetime
import pandas as pd
from mobilitydb_sqlalchemy import TFloat
from sqlalchemy import Column, Integer, create_engine
from sqlalchemy.ext.declarative import declarative_base
from sqlalchemy.orm import sessionmaker

# Setup the engine and session, make sure you set the right url to connect to your mobilitydb instance
engine = create_engine("postgresql://docker:docker@localhost:25432/mobilitydb", echo=True)
session = sessionmaker(bind=engine)()

# Setup and create the tables (only one in our case here)
Base = declarative_base()
class TemporalFloats(Base):
    __tablename__ = "tfloat_test_001"
    id = Column(Integer, primary_key=True)
    tdata = Column(TFloat(True, False))

Base.metadata.create_all(engine)

# Prepare and insert the data
df = pd.DataFrame(["value": 0, "t": datetime.datetime(2018, 1, 12, 12, 0, 0)},{"value": 8.2, "t": datetime.datetime(2018, 1, 12, 6, 0)},{"value": 6.6, "t": datetime.datetime(2018, 1, 12, 10, 0)},{"value": 9.1, "t": datetime.datetime(2018, 1, 12, 15, 0)}]).set_index("t")
row = TemporalFloats(tdata=df,)
```

(continues on next page)
session.add(row)
session.commit()
While creating the DataFrame, make sure the column is named “geometry” and not “value”. This is to maintain compatibility with movingpandas. We can use Point objects from shapely for preparing the geometry data.

```python
from mobilitydb_sqlalchemy import TGeomPoint
from shapely.geometry import Point

class Trips(Base):
    __tablename__ = "trips_test_001"
    car_id = Column(Integer, primary_key=True)
    trip_id = Column(Integer, primary_key=True)
    trip = Column(TGeomPoint)

Base.metadata.create_all(engine)

# Prepare and insert the data
df = pd.DataFrame([{
    "geometry": Point(0, 0), "t": datetime.datetime(2012, 1, 1, 8, 0, 0),},
    {"geometry": Point(2, 0), "t": datetime.datetime(2012, 1, 1, 8, 10, 0),},
    {"geometry": Point(2, -1.9), "t": datetime.datetime(2012, 1, 1, 8, 15, 0),},
]).set_index("t")

trip = Trips(car_id=1, trip_id=1, trip=trip)
session.add(trip)
session.commit()
```
movingpandas is an optional dependency, but if installed, you can insert TGeomPoint data with Trajectory objects directly. Just be sure to enable the flag use_movingpandas on the column beforehand.

```python
from mobilitydb_sqlalchemy import TGeomPoint
from shapely.geometry import Point
from fiona.crs import from_epsg

CRS_METRIC = from_epsg(31256)

class Trips(Base):
    __tablename__ = "trips_test_001"
    car_id = Column(Integer, primary_key=True)
    trip_id = Column(Integer, primary_key=True)
    trip = Column(TGeomPoint(use_movingpandas=True))

Base.metadata.create_all(engine)

# Prepare and insert the data
df = pd.DataFrame(
    [
        
        
        "geometry": Point(0, 0), "t": datetime.datetime(2012, 1, 1, 8, 0, 0),
        "geometry": Point(2, 0), "t": datetime.datetime(2012, 1, 1, 8, 10, 0),
        "geometry": Point(2, -1.9), "t": datetime.datetime(2012, 1, 1, 8, 15, 0),
    ]
).set_index("t")
geo_df = GeoDataFrame(df, crs=CRS_METRIC)

# Note: In case you are depending on movingpandas 0.1 or lower,
# you might need to do mpd.Trajectory(1, geo_df) instead

traj = mpd.Trajectory(geo_df, 1) # Note: In case you are depending on movingpandas 0.1 or lower,

trip = Trips(car_id=1, trip_id=1, trip=traj)
session.add(trip)
session.commit()
```
SQLAlchemy’s `func` is pretty generic and flexible, allowing us to use MobilityDB’s functions without needing any new constructs.

Let’s take few example queries from MobilityDB’s documentation, and explain how we can achieve the same using this package.

```python
from sqlalchemy import func
from shapely.wkt import loads

# Value at a given timestamp
session.query(
    Trips.car_id,
    func.asText(func.valueAtTimestamp(Trips.trip, datetime.datetime(2012, 1, 1, 8, 10, 0))
).all()

# Restriction to a given value
session.query(
    Trips.car_id,
    func.asText(func.atValue(Trips.trip, 'Point(2 0)'))
).all()

# Restriction to a period
session.query(
    Trips.car_id,
    func.asText(func.atPeriod(Trips.trip, '[2012-01-01 08:05:00, 2012-01-01 08:10:00]'))
).all()

# Temporal distance
session.query(
    T1.car_id, T2.car_id, T1.trip <-> T2.trip
).filter(T1.car_id < T2.car_id)

(continues on next page)
Trip.car_id,
func.asText(func.atValue(Trips.trip, Point(2, 0).wkt)),
).all()

# Restriction to a period
session.query(
    Trips.car_id,
    func.asText(
        func.atPeriod(Trips.trip, "[2012-01-01 08:05:00,2012-01-01 08:10:00]"
    ),
).all()

# Temporal distance
session.query(
    T1.c.car_id,
    T2.c.car_id,
    T1.c.trip.distance(T2.c.trip),
) \ 
.filter(T1.c.car_id < T2.c.car_id,) 
.all()
CHAPTER SIX

USING MOBILITYDB OPERATORS

Listing 1: Example usage of the distance operator (‘<->’)

```python
session.query(
    T1.c.car_id,
    T2.c.car_id,
    T1.c.trip.distance(T2.c.trip),
) \
.filter(T1.c.car_id < T2.c.car_id,)
.all()
```

For exhaustive listing of operators, see operators page.
MobilityDB also allows you to store the temporal data in either open or closed intervals on either site. While this is supported by the package at the column level, because we use pandas DataFrame to hold the values once we load them into python runtime, this data is lost, and hence not of much use. In future, this can be avoided with a better suiting data structure to hold this data instead of relying on pandas.

However, to define a column which stores temporal data as a left closed, right open interval, ie. ‘[\)’, it can be done as shown below:

```python
class Trips(Base):
    trip_id = Column(Integer, primary_key=True)
    trip = Column(TGeomPoint(True, False))
```
MAKING USE OF MOVINGPANDAS TRAJECTORY DATA STRUCTURE

TGeomPoint objects can also be optionally mapped to movingpandas Trajectory objects. For this the optional dependency “movingpandas” needs to be installed.

```
poeetry install -E movingpandas
```

After this, movingpandas can be enabled with a flag on the TGeomPoint column

```
class Trips(Base):
    trip_id = Column(Integer, primary_key=True)
    trip = Column(TGeomPoint(use_movingpandas=True))
```
More details can be found in MobilityDB’s documentation itself: https://docs.mobilitydb.com/nightly/index.html

class mobilitydb_sqlalchemy.comparator.Comparator(expr)
    Bases: sqlalchemy.sql.type_api.Comparator

    A custom comparator base class. It adds the ability to call spatial and temporal functions on columns that use
    this kind of comparator. It also defines functions that map to operators supported by TBool, TInt, TFloat
    and TGeomPoint columns.

    always_different_from(other)
        The “@<>” operator.

    always_equal_to(other)
        The “%=” operator.

        Is lhs always equal to the rhs?

        The function does not take into account whether the bounds are inclusive or not.

    always_greater_than(other)
        The “%>” operator.

    always_greater_than_or_equal_to(other)
        The “%>=” operator.

    always_less_than(other)
        The “%<” operator.

    always_less_than_or_equal_to(other)
        The “%<=” operator.

    bbox_always_after(other)
        The “<<#” operator.

    bbox_always_before(other)
        The “<<#” operator.

    bbox_always_strictly_greater_than(other)
        The “>>” operator.

    bbox_always_strictly_less_than(other)
        The “<<” operator.

    bbox-contained(other)
        The “_<@” operator.

    bbox_contains(other)
        The “@>” operator.
bbox_does_not_extend_above (other)
The “|&>” operator.

bbox_does_not_extend_below (other)
The “&<|” operator.

bbox_does_not_extend_in_back (other)
The “/&>” operator.

bbox_does_not_extend_in_front (other)
The “&<|” operator.

bbox_does_not_extend_to_left (other)
The “&>” operator.

bbox_does_not_extend_to_right (other)
The “&<” operator.

bbox_never_after (other)
The “&<#” operator.

bbox_never_before (other)
The “#&>” operator.

bbox_never_greater_than (other)
The “&<” operator.

bbox_never_less_than (other)
The “&>” operator.

bbox_strictly_above (other)
The “|>>” operator.

bbox_strictly_below (other)
The “<<|” operator.

bbox_strictly_in_back (other)
The “/>>” operator.

bbox_strictly_in_front (other)
The “<<|” operator.

bbox_strictly_to_left (other)
The “<<” operator.

bbox_strictly_to_right (other)
The “|>>” operator.

bboxes_equal (other)
The “~=” operator.

bbox_overlap (other)
The “&&” operator.

distance (other)
The “<>” operator.

ever_different_from (other)
The “?<>” operator.

ever_equal_to (other)
The “?=” operator.

Is lhs ever equal to the rhs?
The function does not take into account whether the bounds are inclusive or not.

**ever_greater_than**(other)
The “?>” operator.

**ever_greater_than_or_equal_to**(other)
The “?>=” operator.

**ever_less_than**(other)
The “?<” operator.

**ever_less_than_or_equal_to**(other)
The “?<=” operator.

**smallest_distance_ever_between**(other)
The “|==|” operator.

**temporal_equal**(other)
The “#=” operator.

**temporal_greater_than**(other)
The “#>” operator.

**temporal_greater_than_or_equal_to**(other)
The “#>=” operator.

**temporal_less_than**(other)
The “#<” operator.

**temporal_less_than_or_equal_to**(other)
The “#<=” operator.

**temporal_not_equal**(other)
The “#<>” operator.

**MobilityDB extensions for SQLAlchemy.**

Specifically, the following column types are provided, which can be used in SQLAlchemy models defined using the declarative API:

- **class mobilitydb_sqlalchemy.types.TGeomPoint**
- **class mobilitydb_sqlalchemy.types.TFloat**
- **class mobilitydb_sqlalchemy.types.TInt**
- **class mobilitydb_sqlalchemy.types.TBool**
For getting started with MobilityDB SQLAlchemy, read our Quickstart
CHAPTER
ELEVEN

INDICES AND TABLES

- genindex
- modindex
- search
m

mobilitydb_sqlalchemy.comparator, 17
always_different_from() (mobilitydb_sqlalchemy.comparator.Comparator method), 17
always_equal_to() (mobilitydb_sqlalchemy.comparator.Comparator method), 17
always_greater_than() (mobilitydb_sqlalchemy.comparator.Comparator method), 17
always_greater_than_or_equal_to() (mobilitydb_sqlalchemy.comparator.Comparator method), 17
always_less_than() (mobilitydb_sqlalchemy.comparator.Comparator method), 17
always_less_than_or_equal_to() (mobilitydb_sqlalchemy.comparator Comparator method), 17
bbox_always_after() (mobilitydb_sqlalchemy.comparator.Comparator method), 18
bbox_always_before() (mobilitydb_sqlalchemy.comparator.Comparator method), 18
bbox_always_strictly_greater_than() (mobilitydb_sqlalchemy.comparator.Comparator method), 18
bbox_always_strictly_less_than() (mobilitydb_sqlalchemy.comparator.Comparator method), 18
bbox_contained() (mobilitydb_sqlalchemy.comparator.Comparator method), 18
bbox_contains() (mobilitydb_sqlalchemy.comparator.Comparator method), 18
bbox_does_not_extend_above() (mobilitydb_sqlalchemy.comparator.Comparator method), 18
bbox_does_not_extend_below() (mobilitydb_sqlalchemy.comparator.Comparator method), 18
bbox_does_not_extend_in_back() (mobilitydb_sqlalchemy.comparator.Comparator method), 18
bbox_does_not_extend_in_front() (mobilitydb_sqlalchemy.comparator.Comparator method), 18
bbox_does_not_extend_to_left() (mobilitydb_sqlalchemy.comparator.Comparator method), 18
bbox_does_not_extend_to_right() (mobilitydb_sqlalchemy.comparator.Comparator method), 18
bbox_never_after() (mobilitydb_sqlalchemy.comparator.Comparator method), 18
bbox_never_before() (mobilitydb_sqlalchemy.comparator.Comparator method), 18
bbox_never_greater_than() (mobilitydb_sqlalchemy.comparator.Comparator method), 18
bbox_never_less_than() (mobilitydb_sqlalchemy.comparator.Comparator method), 18
bbox_strictly_above() (mobilitydb_sqlalchemy.comparator.Comparator method), 18
bbox_strictly_below() (mobilitydb_sqlalchemy.comparator.Comparator method), 18
bbox_strictly_in_back() (mobilitydb_sqlalchemy.comparator.Comparator method), 18
bbox_strictly_in_front() (mobilitydb_sqlalchemy.comparator.Comparator method), 18
bbox_strictly_to_left() (mobilitydb_sqlalchemy.comparator.Comparator method), 18
bbox_strictly_to_right() (mobilitydb_sqlalchemy.comparator.Comparator method), 18
bbox_strictly_to_right() (mobilitydb_sqlalchemy.comparator.Comparator method), 18
bboxes_equal() (mobilitydb_sqlalchemy.comparator.Comparator method), 18
bboxes_overlap() (mobilitydb_sqlalchemy.comparator.Comparator method), 18

C
Comparator (class in mobilitydb_sqlalchemy.comparator), 17

D
distance() (mobilitydb_sqlalchemy.comparator.Comparator method), 18

ever_different_from() (mobilitydb_sqlalchemy.comparator.Comparator method), 18
ever_equal_to() (mobilitydb_sqlalchemy.comparator.Comparator method), 18
ever_greater_than() (mobilitydb_sqlalchemy.comparator.Comparator method), 19
ever_greater_than_or_equal_to() (mobilitydb_sqlalchemy.comparator.Comparator method), 19
ever_less_than() (mobilitydb_sqlalchemy.comparator.Comparator method), 19
ever_less_than_or_equal_to() (mobilitydb_sqlalchemy.comparator.Comparator method), 19
ever_not_equal() (mobilitydb_sqlalchemy.comparator.Comparator method), 19

temporal_greater_than() (mobilitydb_sqlalchemy.comparator.Comparator method), 19
temporal_greater_than_or_equal_to() (mobilitydb_sqlalchemy.comparator.Comparator method), 19
temporal_less_than() (mobilitydb_sqlalchemy.comparator.Comparator method), 19
temporal_less_than_or_equal_to() (mobilitydb_sqlalchemy.comparator.Comparator method), 19
temporal_not_equal() (mobilitydb_sqlalchemy.comparator.Comparator method), 19

temporal_equal() (mobilitydb_sqlalchemy.comparator.Comparator method), 19
temporal_greater_than() (mobilitydb_sqlalchemy.comparator.Comparator method), 19
temporal_greater_than_or_equal_to() (mobilitydb_sqlalchemy.comparator.Comparator method), 19

temporal_less_than() (mobilitydb_sqlalchemy.comparator.Comparator method), 19
temporal_less_than_or_equal_to() (mobilitydb_sqlalchemy.comparator.Comparator method), 19
temporal_not_equal() (mobilitydb_sqlalchemy.comparator.Comparator method), 19

temporal_equal() (mobilitydb_sqlalchemy.comparator.Comparator method), 19

temporal_greater_than() (mobilitydb_sqlalchemy.comparator.Comparator method), 19
temporal_greater_than_or_equal_to() (mobilitydb_sqlalchemy.comparator.Comparator method), 19

temporal_less_than() (mobilitydb_sqlalchemy.comparator.Comparator method), 19
temporal_less_than_or_equal_to() (mobilitydb_sqlalchemy.comparator.Comparator method), 19
temporal_not_equal() (mobilitydb_sqlalchemy.comparator.Comparator method), 19

temporal_equal() (mobilitydb_sqlalchemy.comparator.Comparator method), 19

temporal_greater_than() (mobilitydb_sqlalchemy.comparator.Comparator method), 19
temporal_greater_than_or_equal_to() (mobilitydb_sqlalchemy.comparator.Comparator method), 19

temporal_less_than() (mobilitydb_sqlalchemy.comparator.Comparator method), 19
temporal_less_than_or_equal_to() (mobilitydb_sqlalchemy.comparator.Comparator method), 19
temporal_not_equal() (mobilitydb_sqlalchemy.comparator.Comparator method), 19

M
mobilitydb_sqlalchemy.comparator (module), 17

S
smallest_distance_ever_between() (mobilitydb_sqlalchemy.comparator.Comparator method), 19

T
temporal_equal() (mobilitydb_sqlalchemy.comparator.Comparator method), 19