intake_sql Documentation

Release 0.1.1

Martin Durant

Jul 25, 2018

Contents:

Access any SQL data service which *sqlalchemy* can talk to from Intake.

CHAPTER 1

Quickstart

`intake-sql` provides quick and easy access to tabular data stored in sql data sources.

1.1 Installation

To use this plugin for intake, install with the following command:

```
conda install -c intake intake-sql
```

In addition, you will also need other packages, depending on the database you wish to talk to. For example, if your database if Hive, you will also need to install *pyhive*.

CHAPTER 2

API Reference

intake_sql.SQLManualPartition	
intake_sql.SQLSource(uri, sql_expr[, ...])	One-shot SQL to dataframe reader (no partitioning)
intake_sql.SQLSourceAutoPartition(uri, ...)	SQL table reader with automatic partitioning
intake_sql.SQLSourceManualPartition(uri, ...)	SQL expression reader with explicit partitioning
intake_sql.SQLCatalog(uri[, views])	Makes data sources out of known tables in the given SQL service

class intake_sql.**SQLSource**(*uri*, *sql_expr*, *sql_kwargs={}*, *metadata={}*)

One-shot SQL to dataframe reader (no partitioning)

Caches entire dataframe in memory.

Parameters

uri: str or None Full connection string in sqlalchemy syntax

sql_expr: str Query expression to pass to the DB backend

sql_kwargs: dict Further arguments to pass to pandas.read_sql

Attributes

datashape

description

hvplot Returns a hvPlot object to provide a high-level plotting API.

plot Returns a hvPlot object to provide a high-level plotting API.

Methods

close()	Close open resources corresponding to this data source.
discover()	Open resource and populate the source attributes.
read()	Load entire dataset into a container and return it
read_chunked()	Return iterator over container fragments of data source
read_partition(i)	Return a (offset_tuple, container) corresponding to i-th partition.
to_dask()	Return a dask container for this data source
yaml()	Return YAML representation of this data-source

read()
Load entire dataset into a container and return it

class intake_sql.**SQLSourceAutoPartition**(*uri*, *table*, *index*, *sql_kwargs={}*, *metadata={}*)
SQL table reader with automatic partitioning

Only reads existing tables, not arbitrary SQL expressions.

For partitioning, require to provide the column to be used, which should be indexed in the database. Can then provide list of boundaries, number of partitions or target partition size; see dask.dataframe.read_sql_table and examples for a list of possibilities.

Parameters

uri: str or None Full connection string in sqlalchemy syntax

table: str Table to read

index: str Column to use for partitioning and as the index of the resulting dataframe

sql_kwargs: dict Further arguments to pass to dask.dataframe.read_sql

Attributes

datashape

description

hvplot Returns a hvPlot object to provide a high-level plotting API.

plot Returns a hvPlot object to provide a high-level plotting API.

Methods

close()	Close open resources corresponding to this data source.
discover()	Open resource and populate the source attributes.
read()	Load entire dataset into a container and return it
read_chunked()	Return iterator over container fragments of data source
read_partition(i)	Return a (offset_tuple, container) corresponding to i-th partition.
to_dask()	Return a dask container for this data source
yaml()	Return YAML representation of this data-source

read()
Load entire dataset into a container and return it

to_dask()
Return a dask container for this data source

class intake_sql.**SQLSourceManualPartition**(*uri*, *sql_expr*, *where_values*, *where_template=None*, *sql_kwargs={}*, *metadata={}*)

SQL expression reader with explicit partitioning

Reads any arbitrary SQL expressions into pa5titioned data-frame, but requires a full specification of the boundaries.

The boundaries are specified as either a set of strings with *WHERE* clauses to be applied to the main SQL expression, or a string to be formatted with a set of values to produce the comlete SQL expressions.

Note, if not supplying a *meta* argument, dask will load the first partition in order to determine the schema. If some of the partitions are empty, loading without a meta will likely fail.

Parameters

uri: str or None Full connection string in sqlalchemy syntax

sql_expr: str SQL expression to evaluate

where_values: list of str or list of values/tuples Either a set of explicit partitioning statements (e.g., *"WHERE index_col < 50"...*) or pairs of valued to be entered into where_template, if using

where_template: str (optional) Template for generating partition selection clauses, using the values from where_values, e.g., *"WHERE index_col >= {} AND index_col < {}"*

sql_kwargs: dict Further arguments to pass to pd.read_sql_query

Attributes

datashape

description

hvplot Returns a hvPlot object to provide a high-level plotting API.

plot Returns a hvPlot object to provide a high-level plotting API.

Methods

close()	Close open resources corresponding to this data source.
discover()	Open resource and populate the source attributes.
read()	Load entire dataset into a container and return it
read_chunked()	Return iterator over container fragments of data source
read_partition(i)	Return a (offset_tuple, container) corresponding to i-th partition.
to_dask()	Return a dask container for this data source
yaml()	Return YAML representation of this data-source

read()
Load entire dataset into a container and return it

to_dask()
Return a dask container for this data source

class intake_sql.**SQLCatalog**(*uri*, *views=False*, ***kwargs*)
Makes data sources out of known tables in the given SQL service

Attributes

datashape

description

hvplot Returns a hvPlot object to provide a high-level plotting API.

plot Returns a hvPlot object to provide a high-level plotting API.

Methods

close()	Close open resources corresponding to this data source.
discover()	Open resource and populate the source attributes.
force_reload()	Imperative reload data now
read()	Load entire dataset into a container and return it
read_chunked()	Return iterator over container fragments of data source
read_partition(i)	Return a (offset_tuple, container) corresponding to i-th partition.
reload()	Reload catalog if sufficient time has passed
to_dask()	Return a dask container for this data source
walk([sofar, prefix, depth])	Get all entries in this catalog and sub-catalogs
yaml()	Return YAML representation of this data-source

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

Index