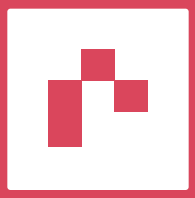

TIME SERIES IN RDM 15.0

Introduction

We will first take a look at a high-level view of time series before diving into specifics for RDM. Time series is a data model for data that is collected periodically over some time. The data will typically be indexed, listed, analyzed, and graphed. There are mainly two domains for time series, which we will cover in the following subsections.

The Time Domain

The first of the two domains is the time domain, where the data consist of discrete measurements with time stamps that are used to extract certain characteristics. This data can then be used for monitoring, reporting, or making future predictions. For efficient retrieval, storage, and lookup, the data is indexed, downsampled, and/or aggregated.

The Frequency Domain

The other domain is the frequency domain, where the data consist of power spectrums. Instead of discrete measurements with respect to time, we have a power spectrum with respect to frequency over a time interval. Power spectrums with high frequencies will normally be over a short time interval while lower frequencies will be over a longer time interval. Data in the time domain can be converted to the frequency domain and vice versa by doing a Discrete Fourier Transform (DFT) or an Inverse DFT.

Fast Fourier Transform (FFT)

Fast Fourier Transform (FFT) is a class of algorithms for the Discrete Fourier Transform (DFT). If you are unfamiliar with FFT, we would recommend the following resource:

- But what is the Fourier Transform? A visual introduction

RDM implements a simple, reasonably efficient algorithm with some limitations for converting data between the time domain and the frequency domain. If a more general solution or higher performance is needed than RDM offers, we suggest FFTW[1]:

- FFTW Home Page

The RDM FFT implementation has been integrated with the time series template classes presented later.

[1] Please be aware that the license for RDM is incompatible with the free software license for FFTW. You will need a commercial license for FFTW.

Raima Data Manager (RDM)

RDM 15.0 supports time series in the time domain and the frequency domain by use of template classes that can be used to process time series data before it is inserted into the database. We will come back to this later as we will start out with some of the general features of RDM and look into how they are particularly suited for time series.

The nature of time series data is that the data needs to be keyed using a timestamp and as data is collected, the timestamps always increase in value since the data is expected to be collected in real time. The same is the case for data that is deleted. We always delete the oldest data and therefore the timestamps of the deleted rows will also increase in value over time as we delete more rows.

Therefore, the main feature that a database needs for efficient handling of time series data is the capability to handle keyed data for which the keys for inserts and those for deletes both increase over time. RDM 15.0 implements circular tables with optimized primary keys. We will not cover this topic here; instead, we will jump straight to the discussion of template classes for time series.

C++ Template Classes for Time Series

RDM provides a set of C++ template classes that can be used to process time series data in a chain of statically instantiated classes. We will first take a look at the data layout.

Data or Table Layout

In its simplest form, the data consist of rows of the following format2:

```
create table measurement (
    time_stamp_current uint64 primary key,
    value_current double not null
);
```

Data for this form will typically be raw measurements or downsampled measurements. The above table definition uses an unsigned 64-bit integer instead of the SQL timestamp type. Furthermore, the value store is just a double. These data types can be replaced with any other type since the template classes are agnostic about these types. However, the operators used in the template classes must be supported for the corresponding C++ types. To meet this requirement, these operators may need to be overloaded with additional operator functions. The columns have fixed names.

[2] Use rdm-compile with the -s option to convert these SQL specified tables to C/C++ structs/classes

Statistics of the data consists of rows of the following format:

```
create table stat (
time_stamp_first uint64 not null, time_stamp_last uint64 primary key, n uint64 not null,
 value_sum double not null, value_sum2 double not null
);
```

Data using this format represents an aggregation of data for a time period instead of data for a particular time. We will not discuss this format any further except to mention that there are template classes that can convert data from the first form to this form.

A third form consists of rows of the following format:

```
create table range (
value_range double array [32] not null, time_stamp_first uint64 not null, time_stamp_last uint64 primary key
);
```

Data using this format represents an aggregation or a transform of data for a time period. Compared to the first form, this one has a range with an array of values and two timestamps: One timestamp for the first value and one for the last value. This form can be used as a more compact representation of the first form, or it could be some kind of transformation of the original data. One such transformation is FFT.

Template Classes

RDM 15.0 includes the RDM Time Series API. It contains a set of template classes used for time series data manipulation. The template classes use data of the form presented in the previous subsection. Each of these template classes implements methods for passing data from one class to the next class in a chain.

Methods Implemented

The first public method that must be called after the instantiation of these template classes is the init method. The init method takes a database handle as the parameter. There are also a number of put methods, one method for each row format applicable to a template class. The put methods are used to pass data on for processing or insert data into the database. Some of these template classes collect a certain amount of data before passing its aggregated information on to the next class in the chain. Many of those template classes allow the aggregated information to be passed on before a threshold for passing on information has been reached by explicitly calling a flush method.

There are also a few other public methods we will not cover in this article. To implement the template classes, some private methods were needed. For instance, in the event of an error, there are private

methods used to restore the state. There are also methods that communicate which tables need to be locked when a transaction is started.

Actual Template Classes

We will give a brief overview of some of the template classes implemented in RDM. Many of these template classes can be extended to suit a specific purpose.

General Template Classes

A template class that is often the first class in a chain is the transaction template class. Strictly speaking, this class is not part of the time series template classes but is often used in conjunction with them. The transaction template class has the next class in the chain as the parameter. This class has extra methods for starting, committing, and rolling back a transaction. This class knows which tables need to be locked based on the next class in the chain (either directly the next class or indirectly any of the next classes). When rolling back a transaction, it also knows how to rewind the state of the next class (and its next classes) back to the state it had when the transaction was first started.

A time series chain without any processing would just have the insert_row template class as the parameter to the transaction template class. In the general case, the insert_row template class would often be the last class in a chain of classes. This class does not hold any state. Instead, it relies on a transaction class at the beginning of a chain to roll back the transaction if necessary.

Time Series Template Classes

Template classes for converting measurements to a range are; collect, fft_abs, and fft_abs_positive. The collect template class simply produces a more compact representation of a number of measurements, where we use an array of values with a timestamp for only the first and the last measurement. The two fft template classes, on the other hand, will perform an FFT, producing the output that therefore represents a range in the frequency domain.

There are template classes for calculating arithmetic mean, geometric mean, and harmonic mean. A number of values or ranges are collected, and their mean values are computed and passed on to the next class in the chain.

Sometimes a chain needs to be split. This can be done using the split template class. A split template class passes on what it receives to two next classes. There are also template classes for doing 3-way, 4-way, and 5-way splits.

There are also template classes for doing statistics, scaling, downsampling, and custom computations. We will not cover those in this article.

Lastly, there is a template class for restoring the state of aggregate classes. This may be needed when a chain is shut down. When it comes up again, we may want to re-feed the data that was last collected in order to reproduce the state those aggregate classes had

before it was shut down. That can be done for some use cases of these template classes by using this template class early in the chain.

Conclusion

We hope we have helped you learn the basics of how RDM supports time series. There are many details left out. To learn more about the RDM time series, refer to the reference manual and our examples.