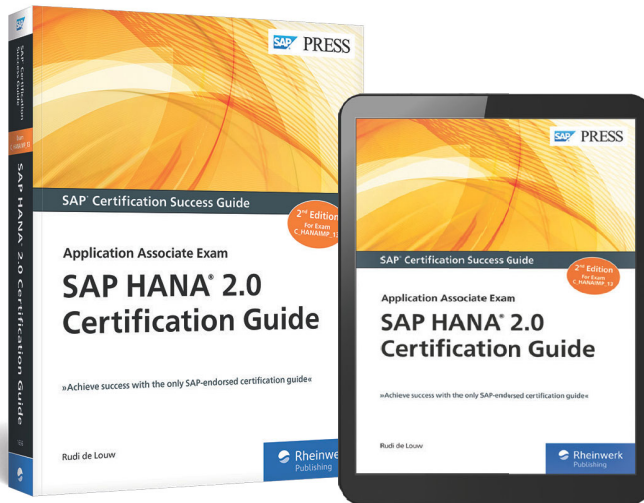


# Modeling with Series Data

from the book

## SAP HANA® 2.0 Certification Guide: Application Associate Exam

by Rudi de Louw



“The comprehensive coverage you need to advance your career, hone your SAP HANA skills, and make the grade!”

# Modeling with Series Data

## Techniques You'll Master

- Get an overview of SAP HANA series data
- Understand the advantages of using SAP HANA for series data
- Get to know some of the SAP HANA series data functionalities

The C\_HANAIMP\_13 exam, based on SAP HANA 2.0 SPS 01, expired in December 2018. Readers will be happy to hear that the *SAP HANA 2.0 Certification Guide*, second edition, for C\_HANAIMP\_13 can also be used to prepare for the next exam—C\_HANAIMP\_14. Some readers have confirmed that they passed the C\_HANAIMP\_14 exam using this book. The C\_HANAIMP\_14 exam is based on SAP HANA 2.0 SPS 02 and will most likely be valid until December 2019.

In fact, the book was already updated for SAP HANA 2.0 SPS 02 before getting published. The screenshots used were actually from an SAP HANA 2.0 SPS 02 system. The only topic that is missing is SAP HANA series data, which contributes about 1% of the exam. That missing topic is addressed in this appendix.

In Chapter 9, we covered four different topics: text, spatial, graph, and predictive modeling. *Series data* is the fifth topic that will complement your knowledge of advanced SAP HANA modeling techniques.

When searching for information on series data, you'll notice that it's normally referred to as "time series" data. SAP HANA uses "series data" because it can also include any data in a series form, including events and transactions. Time series data is often associated with the Internet of Things (IoT), where sensors capture and send out monitoring or event data. However, series data can also include data such as database log files.

In the database world, interest in the topic of *time series databases* has seen the most growth of all types of databases over the past two years. At [https://db-engines.com/en/ranking\\_categories](https://db-engines.com/en/ranking_categories), you can scroll down the page to see the TREND OF THE LAST 24 MONTHS. Time series databases have clearly seen the fastest growth in interest. Many new databases were released into the market that focus exclusively on time series data. Many of them basically store the data using a label (or "tag") with key-value pairs. This is typical for many NoSQL databases.

This appendix will just provide an overview on the topic. You can combine series data with aggregation, analysis, and predictive modeling to create valuable insights into real-world operations.

### Real-World Scenario

You're working on an SAP HANA project with a company that has a fleet of vehicles. All the vehicles are fitted with tracking devices, as well as a variety of sensors to monitor the vehicles.

The company wants a dashboard that shows where the vehicles are (time series data), what they are doing (e.g., loading goods or driving), and when they will be at their destinations. This dashboard shows a combination of real-time data, inferred status information, and arrival time predictions.

You also notice that by analyzing the historical data, you can add planned dates for servicing the vehicles based on their mileage, the way the drivers handled their vehicles, and warnings from sensors in the vehicles. You can then calculate projected running costs of the fleet and even show if the company needs fewer or more vehicles in its fleet.

## Objectives of This Portion of the Test

The purpose of this portion of the SAP HANA certification exam is to test your knowledge of the SAP HANA series data modeling capabilities.

The certification exam expects you to have a good understanding of the following topics:

- Basic series data modeling techniques
- How to create a series table



#### Note

This portion contributes 1% to 2% of the total certification exam score.

## Key Concepts Refresher

Let's say you want to keep track of your weight. Every morning after waking up, you joyfully jump on the bathroom scale and record your weight. After a while, the data will look something like that shown in [Table 1](#).

You've created a time series. The *series data* is a sequence of regular measurements taken over a period of time.

Person	Date1	Weight
You	1 January	176 lbs
You	2 January	175 lbs
You	3 January	174 lbs
You	4 January	173.3 lbs
You	...	...
You	14 February	159 lbs
You	15 February	160 lbs

**Table 1** Tracking Your Weight

There are only three columns, but you'll have hundreds of records in just a few months, resulting in a very long, thin weight table over time. SAP HANA is great for such tables because it's column-based by default and also uses lossless data compression. For time series, SAP HANA can use additional compression methods to reduce the storage requirements even further.

Series data uses the following various columns:

- **Profile column**

This is the sensor or object that you're collecting data from. In this case, it's the name of the person. You can also add other members of the family to the table.

- **Series column**

This column is the time interval for the data collection. Here you're recording your weight daily.

- **Value columns**

The value in this table is your weight on that specific day. In SAP HANA, you can have multiple value columns; for example, you can also record how many hours you slept that night or your waist circumference.

Let's look at how SAP HANA can be used with series data.

## Series Data

If you think about it, most data is series data. In a database, transactions are a series of data points. Many times, these transactions are recorded with a time stamp (date and time) field. In a database, all changes to the data are recorded in database log files, where each change is recorded with a time stamp. Log files are therefore series data. Log files record the changes to our data and are everywhere.



### Tip

Because we're working with time stamps, it's recommended to store them using a single standard, for example, UTC, or UTC with a constant offset. Changes to daylight savings or differences in time zones between countries can introduce complexities when combining series data with regular business data.

As series data looks similar to normal data, you could use normal tables for series data. But sometimes the workload for series data isn't exactly the same as a normal transactional workload. The volumes of data are normally also much larger. Hence, we need some special-purpose optimized features for managing series data. SAP HANA provides such features.

## Examples of Series Data

Before diving into the details, let's look at some use cases for series data and why companies are so interested in this data. In the previous Real-World Scenario section, you already saw one such use case.

With the rise of sensors and IoT, the amount of generated data is increasing dramatically. For example, many cities around the world have installed *smart meters* at each home. Even cities like Johannesburg, South Africa, you could have 1 million smart meters for electricity or water. These smart meters generate a reading every 15 minutes. Each meter therefore creates 2,920 readings per year. For 1 million meters, there are 2.9 billion readings per year! With such volumes of data, efficient storage is vital.

The reason that cities are interested in such data is that they can analyze it to see usage patterns and make predictions for planning infrastructure and providing services for the population. With readings every 15 minutes, they can deduce what devices were used in each 15-minute interval. Assume a stove uses 500 W in 15 minutes, a hot water heater uses 250 W in 15 minutes, a TV uses 25 W, and an LED light bulb uses 2 W. If they get a reading of 29 W in the past 15 minutes, they can

deduce that this home probably had a TV and two light bulbs on. If they get a reading of 512 W, the people were probably cooking and had a few lights on. The city then aggregates this data to see when people wake up, shower, and have breakfast. They use this to predict traffic patterns and electricity demand, see the effects of rain and cold weather, and so on.

Another use case is seen in vehicles. All modern cars have a slot for an On-Board Diagnostics (OBD2) device. You can buy a fairly cheap device and plug it into your car to see the data from your car, for example, speed and fuel consumption, as well as diagnostic and warning messages. As an exercise, you can build a tool in SAP HANA to analyze your driving, and use this information to decrease your fuel consumption.

With the move to cloud and micro-services architectures, the field of DevOps is growing daily. *DevOps* is a software development methodology that combines software development (Dev) with operations (Ops). DevOps aims to automate many of the processes in the software development cycle, reducing the time to market for new services. To ensure service uptime and optimal performance, you need to constantly monitor these micro-services. Analyzing log files has become so important that new time series databases, such as Prometheus, have sprung up to address this need.

Other use cases are daily currency conversion rates, share prices, and rainfall per month.

As you can see, series data allows you to collect large volumes of data, analyze the way this data behaves over time, and use the information for patterns, trends, and forecasts. The fields of engineering, technology, electronics, development, operations, statistics, signal processing, and machine learning are coming together when we work with series data.



---

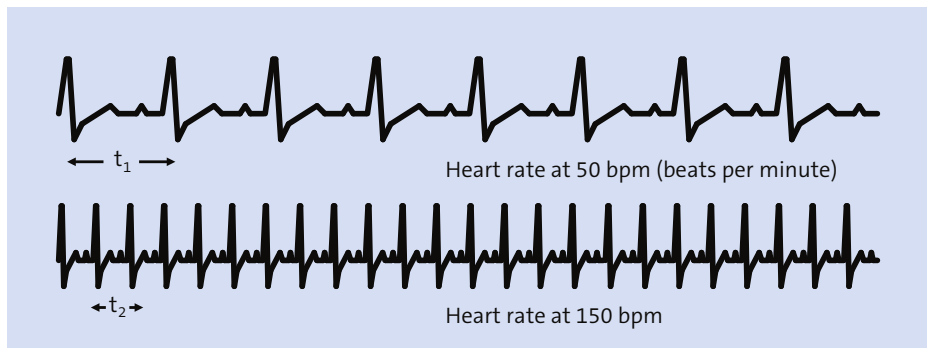
**Note**

It's important to realize that data in the real world isn't always perfect or complete. You'll always get some missing readings or some abnormal readings due to network errors, faulty sensors, buffer overflows, bad code, or myriad other reasons.

### Internet of Things, Edge Processing, and Equidistant Series Data

We started looking at IoT, smart meters, and OBD2 devices in the previous section. To better understand IoT, we can also look at a common IoT scenario—exercise and smart watches. As you move, run, or exercise, the watch saves your GPS location, the elevation, and your heart rate every second.

So how does the watch know what your heart rate is at any moment of time? The electronics of the watch actually detect the individual heartbeats and measure the time between the heartbeats. The top graph in [Figure 1](#) illustrates resting heartbeats. The time between heartbeats, shown as  $t_1$ , is relatively long—just over 1 second. The bottom graph in [Figure 1](#) illustrates heartbeats during exercise, and the time ( $t_2$ ) is much shorter.



**Figure 1** Heart Rates Measured by a Smart Watch

The watch now counts the number of heartbeats in a time period to *calculate* your heart rate. If it measures 20 heartbeats in 15 seconds, it calculates that this would result in 80 heartbeats per minute, and thus your heart rate is 80 bpm (beats per minute).

The important thing to realize is that almost all IoT devices do some calculations on the data. This is called IoT *edge processing*.

The watch processes your heartbeat data, averages it over the past few seconds, calculates your heart rate, and displays this average to you every second. The watch doesn't store the "raw" data, which is, in this case, the time at which each heartbeat occurs. Rather, the watch stores the calculated value of your heart rate each second.

The series data that the watch provides every second is called *equidistant*. Equidistant is when the time increments between successive records in the series data are a constant—in this case, 1 second. When you look at the heartbeat data that the watch actually measured (refer to [Figure 1](#)), you’ll see that  $t_1$  and  $t_2$  aren’t the same. Using edge processing, the watch changed the nonequidistant heartbeat series data into equidistant heart rate series data.

The heartbeat data is showing the actual *events*, that is, the exact time when each heartbeat occurred. The heart rate data, on the other side, shows aggregate summaries at regular intervals of 1 second. From this, you see that recording *actual* event data provides nonequidistant series data. Aggregated data normally provides equidistant series data. For example, actual financial transactions (events) are nonequidistant, whereas daily or monthly balances (aggregates) are equidistant.

It’s not just edge processing that performs aggregation of series data. You can also aggregate series data in SAP HANA, as discussed later in this appendix.

SAP HANA can also convert nonequidistant series data into equidistant data. Using series data functions, you can specify which rounding rules you want to apply. For example, you can round 8:30 down to 8:00 or up to 9:00.

You might be concerned that you’re losing some data when IoT devices are using aggregations. Most of the time, this isn’t a concern because you normally don’t want that much data. When exercising, you don’t want to see each heartbeat. The aggregated heart rate information is exactly what you need at that point. If, however, you’re a medical practitioner, you might be interested in the actual heartbeats, for example, to detect irregular heartbeats or atrial fibrillation.

In manufacturing plants, “historian” databases are used to store the IoT series data. These databases can then be used later to do predictive maintenance using predictive algorithms like those we discussed in Chapter 9.



#### Note

IoT and series data are very often combined with streaming data, predictive modeling, and machine learning. Everything you learned in Chapter 9 is used together in powerful ways to create business insights. SAP HANA provides everything you need for addressing these requirements.



## SAP HANA for Series Data

Due to the increased popularity of time series databases, quite a few of these new databases have appeared in the market. As the market is maturing, users of these time series databases run into some limitations; for example, some of them only allow a single value field (like [Table 1](#)). This is due to the key-value pair structure they received from their NoSQL heritage. You have to create another new table to add a new value column. In SAP HANA, you can have multiple value columns—for example, you can also record how many hours you slept that night or your waist circumference—all in the same table.

One of the biggest limitations of dedicated time series databases is that they can only store time series data. SAP HANA databases can contain both series data and regular business data. From a modeling perspective, there is no difference between these different types of database tables. You can integrate series data with any other type of data to produce powerful applications and analytics by using calculation views or standard SQL statements to query any table, including series data tables.

Many of the dedicated time series databases use their own (limited) query languages with SQL-like syntax that just introduces mental friction. Existing SQL-based tools don't work with these databases. SAP HANA series data can be accessed using standard SQL, in which simple queries stay simple, complex queries are powerful, and joins are easy. Data modelers and business users can get started immediately using familiar business intelligence (BI) tools, using legacy systems to query this data, and inheriting a vast ecosystem of functionality, for example, streaming data, extract, transform, load (ETL), enterprise resource planning (ERP), BI, visualization, and so on.

While there is no difference in using a series table in SAP HANA from a modeling perspective, technically, there is obviously a difference between series data tables and regular column tables. You've seen that series data can have large volumes and that the tables tend to be long and thin. SAP HANA can use additional compression methods to reduce the storage requirements even further.

In [Table 1](#) previously, you saw a series data table. While you can store this data in a normal column table in SAP HANA, which will already provide quite good compression, you can do even better. Instead of actually storing the series (`Date1`) column, you can *calculate* these dates. The table is an equidistant series data table, meaning that the time period between dates of successive records is a constant. SAP HANA only has to store the start date and the equidistant time period instead of the entire column! To calculate the date for the fourth record, SAP HANA takes

the start date (1 January), and adds three days to give you 4 January. The formula for calculating is therefore as follows:

$$\text{Calculated time stamp} = \text{Starting time stamp} + (\text{Record number} - 1) \times \text{Equidistant time period}$$

SAP HANA does this compression automatically in the background when you use an equidistant series table, so you won't see the compression or the calculations. You can use calculation views and normal SQL statements, as if the data is actually stored in the table instead of being calculated.

You can find out how to calculate how much space you save when using an equidistant series table instead of a column table by watching the SAP HANA Academy videos about series data on YouTube. In the specific case given in these videos, they only used about 12% of the space. You can find these videos on YouTube at <http://bit.ly/2TevdQB>. The source code is at <https://github.com/saphanaacademy/SeriesData>.



#### Note

Calculation views or SQL queries that use compressed equidistant (calculated) series columns will be slower than if the data were stored in nonequidistant tables. You should use nonequidistant tables if your series column (time) data has no pattern or when trading better performance for more memory is preferred and acceptable.

Sometimes, when you keep adding a lot of data to a series table over time, the performance might decrease. You can then run an `ALTER TABLE SERIES REORGANIZE` statement to improve the compression.

## Creating Series Tables

Let's say you want to create the series table shown in [Table 1](#). You can do so using [Listing 1](#).

```
CREATE COLUMN TABLE SCHEMA_NAME.TABLE_NAME (
    Person VARCHAR(25) NOT NULL,
    Date1 DATE NOT NULL,
    Weight DECIMAL(5,2),
    PRIMARY KEY (Person, Date1)
)
SERIES (
    SERIES KEY(Person)
    PERIOD FOR SERIES(Date1)
```

```

EQUIDISTANT INCREMENT BY INTERVAL 1 DAY
MINVALUE '2019-01-01'
MAXVALUE '2019-12-31'
);

```

**Listing 1** Creating the Series Table for Table 1

You'll notice that this code starts with creating a normal column table, but then adds the `SERIES` clause to specify that this is a series table. Inside this `SERIES` clause, note the following:

- The `SERIES KEY` is used to define the `Person` profile column. In this case, it's the name of the person whose weight is measured. The `SERIES KEY` can be multiple columns.
- `PERIOD FOR SERIES` is used to specify the `Date1` series column, where the time values are stored.
- `EQUIDISTANT INCREMENT BY` defines that this series is equidistant, and the distance between data points is one day.
- `MINVALUE` and `MAXVALUE` specifies the range of acceptable dates for the `Date1` series column. `MINVALUE` specifies the start point, and `MAXVALUE` the specifies the end point.

SAP HANA will also create a trigger for the table, which will return errors if you try to insert dates that aren't in the specified range or if the date doesn't align with the daily interval (in this case).

You can use the `SERIES_GENERATE` function to prepopulate the entire series column (`Date1`). The rows are then ready for just adding values (`Weight`) at a later point.

You can alter a series table using the `ALTER TABLE` statement. However, the `SERIES` clause of an existing series table can't be altered.

## Horizontal and Vertical Aggregation

A challenge you'll often face when working with series tables is how to `JOIN` two series tables with different time intervals. To solve this, you must use horizontal aggregation.

*Horizontal aggregation* is when you take granular data and aggregate it to a less granular level. For example, you can aggregate hourly data to a daily level. Or you can disaggregate the data to a more granular level, for example, breaking up hourly data to 15-minute intervals.

**Note**

Horizontal aggregation doesn't change the data in the series table. It's merely a "view" that displays the data at a higher or lower level of granularity and allows you to JOIN series tables with different time intervals.

You can use normal aggregation functions, such as SUM and AVG, for aggregating the value columns. When taking the `Date1` series column, for example, to a less granular level, you use the `SERIES_ROUND` function. In [Listing 2](#), horizontal aggregation is applied to the data from [Table 1](#). In this case, you use the `ROUND_DOWN` option, meaning that 14 February gets rounded down to the month of February, rather than rounded up to the month of March. In this case, this is quite obvious, but sometimes you'll want 8:59 to be rounded up to 9:00, rather than rounded down to 8:00.

```
SELECT
    Person,
    SERIES_ROUND(Date1, 'INTERVAL 1 MONTH', ROUND_DOWN) AS Monthly,
    AVG(Weight) AS MonthlyWeight
FROM SCHEMA_NAME.TABLE_NAME
GROUP BY Person, SERIES_ROUND(Date1, 'INTERVAL 1 MONTH', ROUND_DOWN);
```

**Listing 2** Horizontal Aggregation

[Listing 2](#) is the classical SQL format. You might not like the fact that the `SERIES_ROUND` function is called twice. [Listing 3](#) shows an alternative way of accomplishing the same horizontal aggregation by using a `WITH ... SELECT` statement, which only calls the `SERIES_ROUND` function once.

```
WITH TMP AS (
    SELECT Person,
    SERIES_ROUND(Date1, 'INTERVAL 1 MONTH', ROUND_DOWN) AS Monthly,
    Weight
    FROM SCHEMA_NAME.TABLE_NAME
)
SELECT Person, TIMER_HOUR, AVG(Weight) AS MonthlyWeight
FROM TMP
GROUP BY Person, Monthly;
```

**Listing 3** Horizontal Aggregation Using a WITH Clause

You use the `SERIES_DISAGGREGATE` function instead of the `SERIES_ROUND` function when you move from coarse units (e.g., daily) to finer units (e.g., hourly).

*Vertical aggregation* is when you aggregate measures over a number of attributes. You can JOIN a series table with other tables in SAP HANA and aggregate the

results to create interesting reports, for example, per factory, per region, per machine type, or per year.

All logical join types are supported when joining two series tables. However, both series tables must be at similar levels of granularity. Both can be nonequidistant or equidistant with the same time interval.

### Functions for Series Data

Special SQL functions are available specifically for series data. You've seen quite a few of these already, for example, `SERIES_ROUND`, `SERIES_DISAGGREGATE`, and `SERIES_GENERATE`. These functions are documented in the SAP HANA Series Data Developer Guide at [http://help.sap.com/hana/SAP\\_HANA\\_Series\\_Data\\_Developer\\_Guide\\_en.pdf](http://help.sap.com/hana/SAP_HANA_Series_Data_Developer_Guide_en.pdf).

More advanced functions, such as `CUBIC_SPLINE_APPROX`, allow you to fit a cubic spline over your series data.

You also have additional analytic functions for analyzing series data, such as `MEDIAN` to compute a median value, or `CORR` to calculate the correlation coefficient.

## Important Terminology

In this appendix, we looked at the following important terms:

- **Time series data**

Time series data is a sequence of regular measurements taken over a period of time.

- **Equidistant**

Equidistant is when the time increments between successive records in the series data are constant, for example, one hour.

- **Horizontal aggregation**

Horizontal aggregation is when you take granular data and aggregate it to a less granular level. For example, you can aggregate hourly data to a daily level or disaggregate the data to a more granular level, such as breaking up hourly data to 15-minute intervals.



## Practice Questions

These practice questions will help you evaluate your understanding of the topics covered in this appendix. The questions shown are similar in nature to those found on the certification examination. Although none of these questions will be found on the exam itself, they will allow you to review your knowledge of the subject. Select the correct answers, and then check the completeness of your answers in the “Practice Question Answers and Explanations” section. Remember, in the exam, you must select *all* correct answers and *only* correct answers to receive credit for the question.

1. What will you use to convert daily series data to an hourly level?  
Note: There are 2 correct answers to this question.
  - ☐ A. Vertical aggregation
  - ☐ B. Horizontal aggregation
  - ☐ C. SERIES\_DISAGGREGATE
  - ☐ D. SERIES\_ROUND
2. What do you use to create an equidistant series table?
  - ☐ A. CREATE TABLE
  - ☐ B. CREATE SERIES TABLE
  - ☐ C. CREATE COLUMN TABLE
  - ☐ D. CREATE HISTORY COLUMN TABLE
3. You create an equidistant series table. In the SERIES clause, what do you use to identify the profile column?
  - ☐ A. SERIES KEY
  - ☐ B. PERIOD FOR SERIES
  - ☐ C. EQUIDISTANT INCREMENT BY
  - ☐ D. PRIMARY KEY

## Practice Question Answers and Explanations

1. Correct answer: **B, C**

You use horizontal aggregation and `SERIES_DISAGGREGATE` to convert daily series data to an hourly level. In this case, you go to hourly, which means there are *more* values. You need to disaggregate to achieve this.

`SERIES_ROUND` is used with aggregation. Aggregation (e.g., `SUM`) takes many values and reduces them to one value.

2. Correct answer: **C**

You use `CREATE COLUMN TABLE` with a `SERIES` clause to create a series table.

There is no `CREATE SERIES TABLE` statement.

`CREATE TABLE` creates a row table.

A history column table is for audit log type tables, where you keep a full history of all changes made to the table.

3. Correct answer: **A**

You identify the profile column, which is the name of the “sensor” in IoT, with `SERIES KEY`).

The `PERIOD FOR SERIES` (used on the series column) is used for the time period.

## Takeaway

You now know how to create a series table, know what an equidistant series is, and have a basic understanding of series data modeling techniques.

## Summary

In this appendix, we discussed important aspects of SAP HANA series data, including the advantages of using SAP HANA. Throughout this overview of SAP HANA series data, you’ve learned more about the SAP HANA series data functionality, as well as the basic concepts of series data in SAP HANA, such as series tables, equidistant series data, and horizontal aggregation.

Best wishes for your C\_HANAIMP\_14 exam!