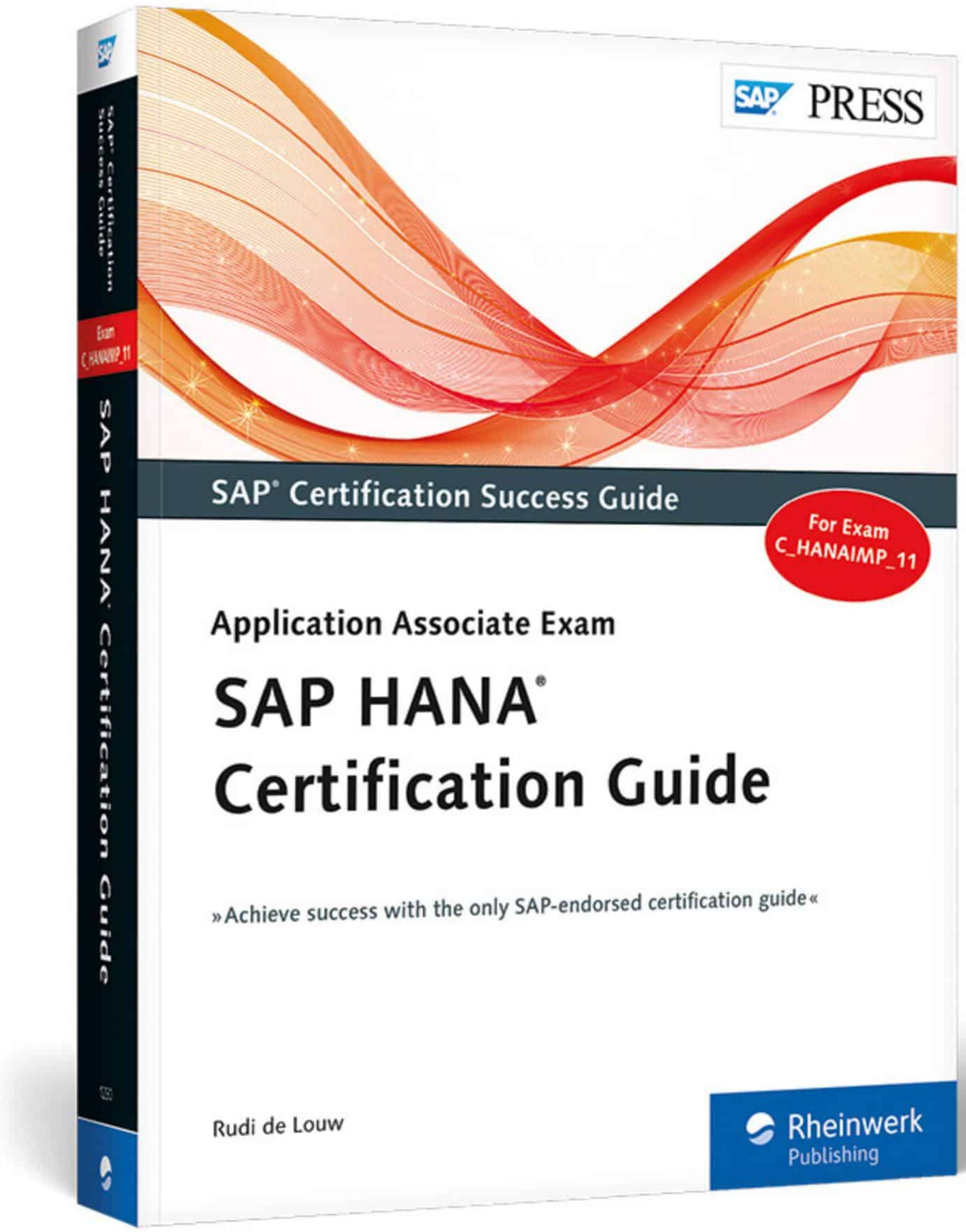


Reading Sample

In this reading sample, we'll use Chapter 4 to show you how the book is structured to prepare you for the exam. Chapter 4 provides details on the core modeling concepts you will find throughout the book and the exam. Understanding the basics, such as views, joins, cubes, fact tables, hierarchies, CDS views, and the different information views, is critical in developing more advanced data modeling skills.

"Information Modeling Concepts"

Contents

Index

The Authors

Rudi de Louw

SAP HANA Certification Guide

559 Pages, 2016, $79.95
ISBN 978-1-4932-1230-9

www.sap-press.com/3859

4 Information Modeling Concepts

Techniques You'll Master

- Understand general data modeling concepts like views, cubes, fact tables, and hierarchies
- Know when to use the different types of joins in SAP HANA
- Examine the differences between attributes and measures
- Learn about the different types of information views that SAP HANA uses
- Get to know projections, aggregations, and unions used in SAP HANA information views
- Describe Core Data Services (CDS) in SAP HANA
- Get a glimpse of best practices and general guidelines for data modeling in SAP HANA

Before moving onto the more advanced modeling concepts, it's important to understand the basics. In this chapter, we'll discuss the general concepts of information modeling in SAP HANA. We'll start off by reviewing views, join types, cubes, fact tables, hierarchies, the differences between attributes and measures, the different types of information views that SAP HANA offers, and CDS views. From there, we'll take a deeper dive into SAP HANA's information views, as well as concepts like calculated columns, how to perform currency conversions, input parameters, decision tables, and so on.

Real-World Scenario

You start a new project. Many of the project team members have some knowledge of traditional data modeling concepts and describe what they need in those terms. You need to know how to take those terms and ideas, translate them into SAP HANA modeling concepts, and implement them in SAP HANA in an optimal way.

Some of the new concepts in SAP HANA are quite different than what you're used to—for example, not storing the values of a cube, instead calculating it when required. If you understand these modeling concepts, you can quickly understand and create real-time applications and reports in SAP HANA.

Objectives of This Portion of the Test

The objective of this portion of the SAP HANA certification is to test your understanding of basic SAP HANA modeling artifacts.

For the certification exam, SAP HANA modelers must have a good understanding of the following topics:

- The different types of joins available in SAP HANA and when to use each one
- How the SAP HANA information views correspond to traditional modeling artifacts
- The different types of SAP HANA views and when to use them
- The purpose of Core Data Services (CDS) in SAP HANA

Note

This portion contributes up to 5% of the total certification exam score.

Key Concepts Refresher

This section looks at some of the core data modeling concepts used when working with SAP HANA and that will be covered on the exam. Let's start from where the data is stored in SAP HANA by looking at tables.

Tables

Tables allow you to store information in *rows* and *columns*. Inside SAP HANA, there are different ways of storing data. We can either store it in a table as *row-oriented* or *column-oriented*. In a normal disk-based database, we use row-oriented storage because it works faster with transactions in which we are reading and updating single records. However, because SAP HANA works in memory, we prefer the column-oriented method of storing data in the tables, with which we can make use of compression in memory, remove the storage overhead of indexes, save a lot of space, and optimize for performance by loading compressed data into computer processors.

Once we have our tables, we can begin combining them in SAP HANA information models.

Views

The first step in building information models is building *views*. A *view* is a database entity that is not persistent and is defined as the projection of other entities, like tables.

For a view, we take two or more tables, link these different tables together on matching columns, and select certain output fields. Figure 4.1 illustrates this process.

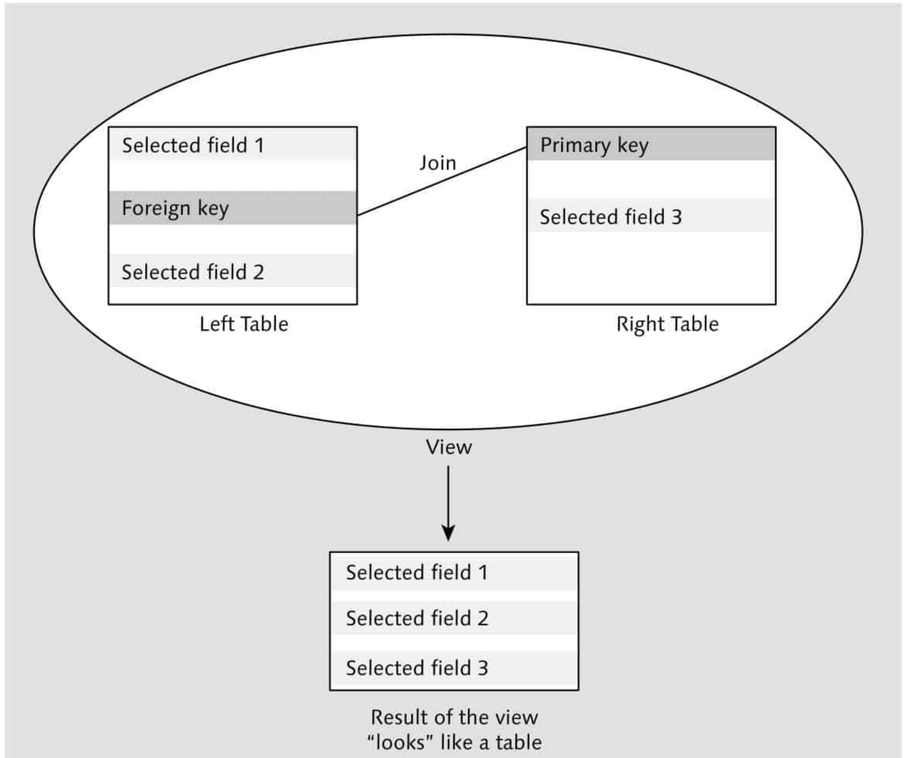


Figure 4.1 Database View

There are four steps when creating database views:

1. **Select the tables**
   Select two or more tables that have related information—for example, two tables listing the groceries you bought. One table contains the shop where you bought each item, the date, the total amount, and how you paid for your groceries. The other table contains the various grocery items you bought.
2. **Link the tables**
   Next, link the selected tables on matching columns. In our example, perhaps our two tables both have shopping spree numbers, should ideally a unique number for every shopping trip you took. We call this a *key field*.
   We can link the tables together with *joins* or *unions*.

3. **Select the output fields**
   We want to show a certain number of columns that are of interest to us—for example, to use in our grocery analytics. Normally, you don't want all the available columns as part of the output.
4. **Save the view**
   Finally, save your view, and it's created in the database. These views are sometimes called *SQL views*. We can even add some filters on our data in the view—for example, to show only the shopping trips in 2015 or all the shopping trips in which we bought apples.

When we call this view, the database gathers the data. The database view pulls the tables out, links them together, chooses the selected output fields, reads the data in the combined fields, and returns the result set. After this, the view "disappears" until we call it again. Then, the database deletes all of the output from the view. It does not store this data from the view inside the database storage.

Tip

It's important to realize that database views don't store the result data. Each time you call a view, it performs the full process again.

You might have also heard about *materialized views*, in which people store the output created by a view in another table. However, in our definition of *views*, we stated that a view is a database entity that *is not persistent*; that is the way we use the term *view* in SAP HANA.

The data stays in the database tables. When we call the view, the database gathers the subset of data, showing us only the data that we asked for. If we ask a database view for that same data a few minutes later, the database will regather and recalculate the data again.

This concept is quite important going forward, because you make extensive use of views in SAP HANA. SAP HANA creates a cube-like view, for example, sends the results back to us, and "disappears" again. SAP HANA performs this process fast, and we avoid consuming a lot of extra memory by not storing static results. What really makes this concept important is the way it enables us to perform *real-time reporting*.

In the few minutes between running the same view twice, our data in the table might have changed. If we use the same output from the view every time, we will not get the benefit of seeing the effect of the updated data. Extensive caching of result sets does not help when we want real-time reporting.

Note

Real-time reporting requires recalculating results, because the data can keep changing. Views are designed to handle this requirement elegantly.

We have discussed the basic type of database views here, but SAP HANA takes the concept of views to an entirely new level!

Before we get there, we'll look at a few more concepts that we will need for our information modeling journey.

Cardinality

When we join tables together, we need to define how much data we expect to get from the various joined tables. This part of the relationship is called the *cardinality*.

There are four basic cardinalities that you will typically work with. The cardinality is expressed in terms of how many records on the left side join to how many records on the right side.

- **One-to-one**
  Each record on the left side matches with one record on the right side. For example, say that you have a lookup table of country codes and country names. In your left table, you have a country code called ZA. In the right table (the lookup table), this matches with exactly one record, showing that the country name of South Africa is associated with ZA.
- **One-to-many**
  In this case, each record on the left side matches with many records on the right side. For example, say you have a list of publishers in the left table. For one publisher—such as SAP PRESS—we find many published books.

- **Many-to-one**
  This is the inverse of one-to-many. For example, this case may apply when we have many people in a small village all buying items at the local corner shop.
- **Many-to-many**
  For example, this case may apply when many people read many web pages.

Joins

Before we look at the different types of joins, let's quickly discuss the idea of "left" and "right" tables. Sometimes, it does not make much of difference which table is on the left of the join and which table is on the right, because some join types are symmetrical. However, with a few join types it does make a difference.

We recommend putting the most important table, seen as the *main table*, on the left of the join. An easy way to remember which table should be the table on the right side of the join is to determine which table can be used for a lookup or for a dropdown list in an application (see Figure 4.2).

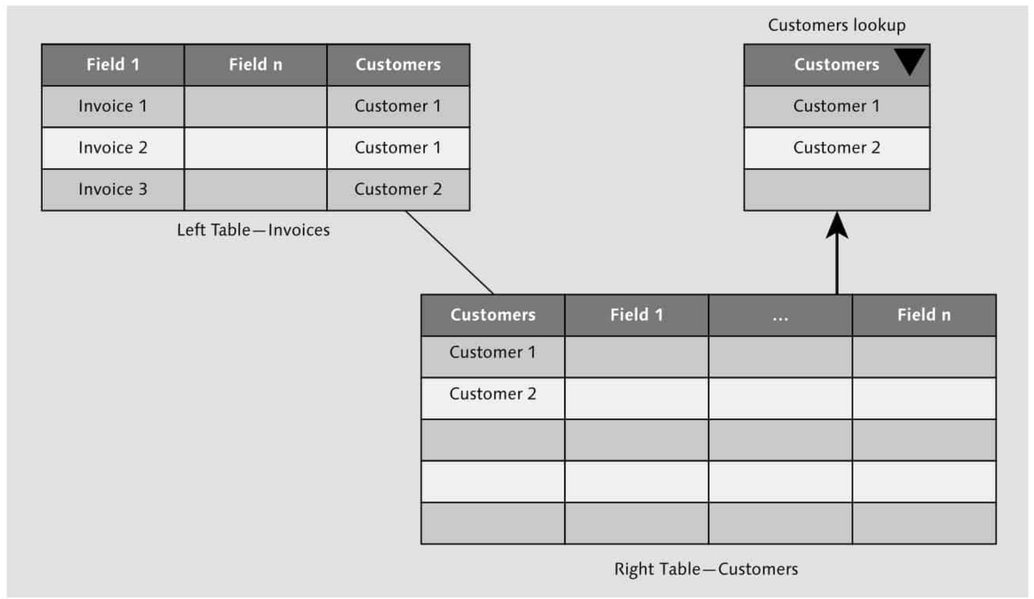


Figure 4.2 Table Used for Dropdown List on Right-Hand Side of a Join

This does not always mean that this table has to be physically positioned on the left of the screen in our graphical modeling tools. When we create a join in SAP HANA, the table we start the join from (by dragging and dropping) becomes the "left table" of the join.

Databases are highly optimized to work with joins; they are much better at it than you can hope to be in your application server or reporting tool. SAP HANA takes this to the next level with its new in-memory and parallelization techniques.

In SAP HANA, there are a number of different types of joins. Some of these are the same as what you would find in traditional databases, but others are unique to SAP HANA. Let's start by looking at the basic join types.

Basic Join Types

The first basic join types that you will find in most databases are inner joins, left outer joins, right outer joins, and full outer joins. We will discuss SQL in more detail in Chapter 8.

The easiest way to visualize these join types is to use circles, as shown in Figure 4.3. This is a simplified illustration of what these join types do. (The assumption is that the tables illustrated here contain unique rows; that is, we join the tables via primary and foreign keys, as illustrated in Figure 4.1. If the tables contain duplicate records, this visualization does not hold.)

The left circle represents the table on the left side of the join. In Figure 4.3, the left table contains the two values A and B. The right circle represents the table on the right side of the join and contains the values B and C.

Inner Join

The *inner join* is the most widely used join type in most databases. When in doubt and working with a non-SAP HANA database, try an inner join first.

The inner join returns the results from the area where the two circles overlap. In effect, this means that a value is shown only if it is found to be present in both the left and the right tables. In our case, in Figure 4.3, you can see that the value B is found in both tables.

Because we are only working with the "overlap" area, it does not matter for this join type which table is on the left or on the right side of the join.

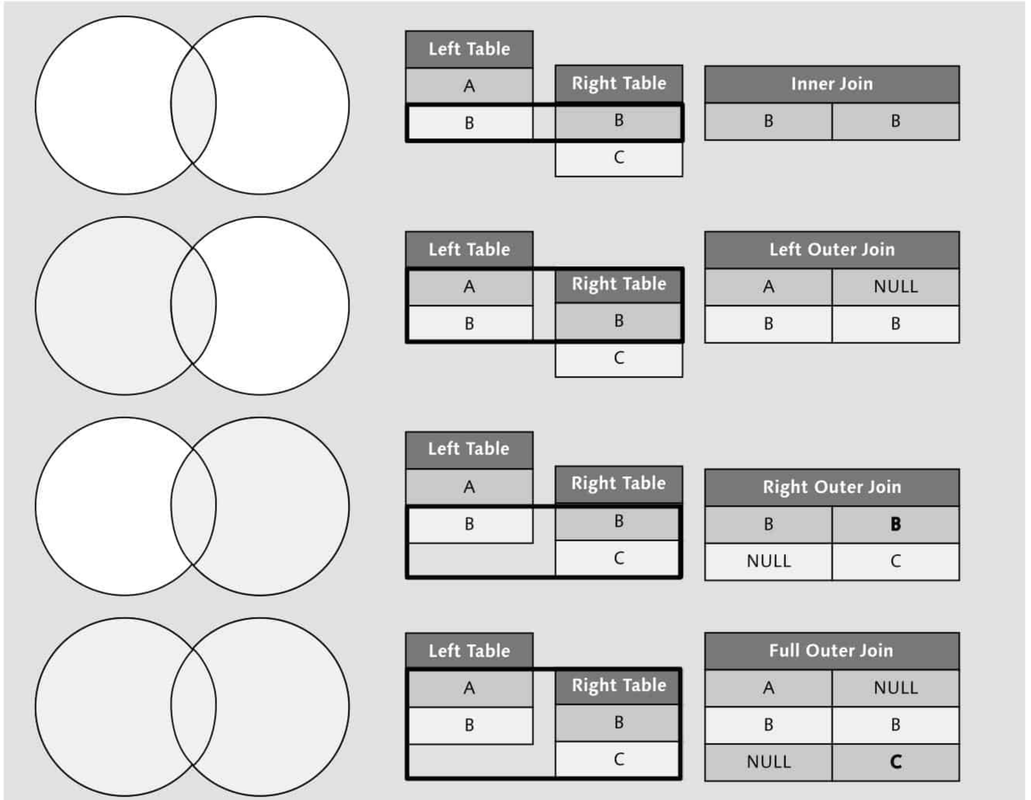


Figure 4.3 Basic Join Types and Their Result Sets

Left Outer Join

When using an inner join you may find that you're not getting all the data back that you require. To retrieve the data that was *left out*, you would use a *left outer* join. You will only use this join type when this need arises.

With a left outer join, everything in the table on the left side is shown (first table). If there are matching values in the right table, these are shown. If there are any "missing" values in the right hand table, these are shown with a `NULL` value.

For this join type, it is important to know which tables are on the left and the right side of the join.

Right Outer Join

The inverse of the left outer join is the right outer join, This shows everything from the right table (second table). If there are matching values in the left table, these are shown. If there are any "missing" values in the left hand table, these are shown with a `NULL` value.

Full Outer Join

As of SAP HANA SPS 11, we now have the full outer join. Many other databases also have this join type, so it is still regarded as one of the four basic join types.

This join type combines the results sets of both the left outer join and right outer join into a single result set.

Self-Joins

There isn't really a join type called a *self-join*; the term refers to special cases in which a table is joined to itself. The same table is both the left table and the right table in the join relationship. The actual join type would still be something like an inner join.

This configuration is used mostly in recursive tables—that is, tables that refer back to themselves, such as in HR organizational structures. All employee team members have a manager, but managers are also employees of their companies and have someone else as their manager. In this case, team members and managers are all employees of the same company and are stored in the same table. Other examples include cost and profit center hierarchies or bills of materials (BOMs).

We return to this concept when we look at the Hierarchies section later in this chapter.

SAP HANA Join Types

The join types on the right side of Figure 4.4 are unique to SAP HANA: referential join, text join, and temporal join.

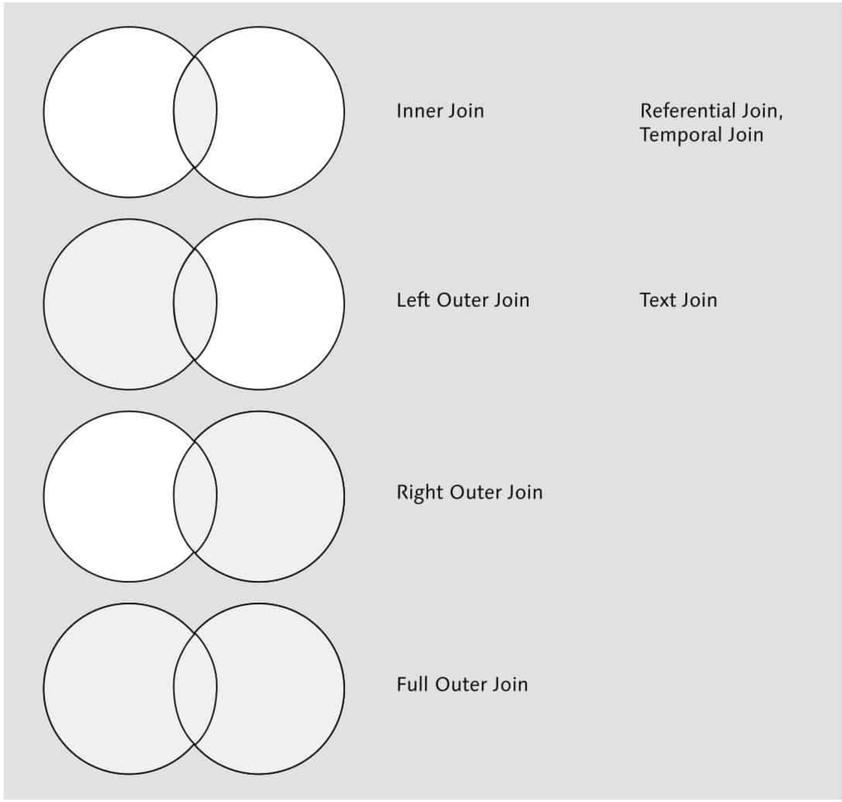


Figure 4.4 Basic Join Types and Some SAP HANA-Specific Join Types

Referential Join

The referential join type normally returns exactly the same results as an inner join. So, what's the difference? There are many areas SAP HANA tries to optimize the performance of queries and result sets. In this case, even though a referential join gives the same results as an inner join, it provides a performance optimization under certain circumstances.

The referential join uses *referential integrity* between the tables in the join. Referential integrity is used in a business system to ensure that data in the left and right tables match. For example, we can't have an invoice that isn't linked to a customer, and the customer must be inserted into the database before we are allowed to create an invoice for that customer. Note that you do not need a value

on both sides of the join: You may have a customer without an invoice, but you may not have an invoice without a customer.

If we can prove that the data in our tables has referential integrity, which most data in financial business systems has, then we can use a referential join. In this case, if we're not reading any data from the right table, then SAP HANA can quite happily ignore the entire right table and the join itself, and it doesn't have to do any checking, which speeds up the whole process.

Tip

A referential join is the optimal join type in SAP HANA.

Text Join

Another SAP HANA-specific joint type is a text join. The name gives a clue as to when we will use this type of join.

The right-hand table, as shown in Figure 4.5, would be something like a text lookup table—for example, a list of country names. On one side, we would have a country code such as US or DE, and in the text lookup table we would have the same country code and would link it with the key, and also would have the actual name of the country (e.g., United States or Germany).

SAP sells software in many countries, and SAP business systems are available in multiple languages, and the name of the country and be translated into different languages. The fourth column in the text lookup table, called the language code, indicates into which language the country name has been translated. For example, for the country called DE and a language code of EN, the name of the country is in English and thus would be Germany. If the language code was DE (for German), the name of the country would be Deutschland, and if the language code was ES (for Español, indicating Spanish) then the name of the country would be Alemania.

Therefore, the same country can have totally different names depending on what language you speak.

In such a case, SAP HANA does something clever: By looking at the browser, the application server, or the machine that you are working on, it determines the language that you are logged on in. If you are logged in using English, it knows to use the name Germany. If you are logged on using German, it provides the German country name Deutschland for you.

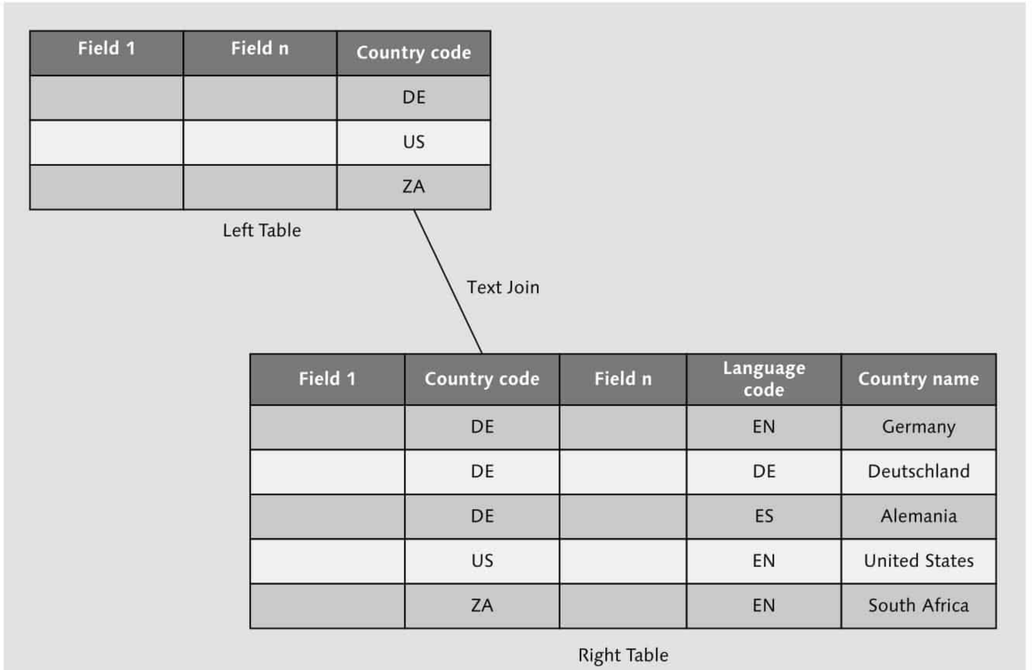


Figure 4.5 Text Join

The text join behaves like a left outer join but always has a cardinality of one-to-one (1:1). In other words, it only returns a single country name based on the language you're logged on with. Even if you're logged in via mobile phone, your mobile phone has a certain language setting.

Temporal Join

We use the temporal join when we've got `FROM` and `TO` date and time fields, or integers.

For example, say that you're storing the fuel price for gasoline in a specific table. From the beginning of this month to the end of this month, gasoline sells for a certain price. Next month, the price differs, and maybe two weeks later, it's adjusted again. Later, you'll have a list of all the different gasoline prices over time.

As a car owner, you now want to perform calculations for how much you've paid for your gasoline each time you filled up your tank. However, you just have the number of gallons and the dates of when you filled up.

You can say, "OK, on this date, I filled up the tank." You then have to look up which date range your filling-up date falls within and find the price for that specific date. You can then calculate what you paid to fill your tank for each of your dates. Figure 4.6 illustrates this example.

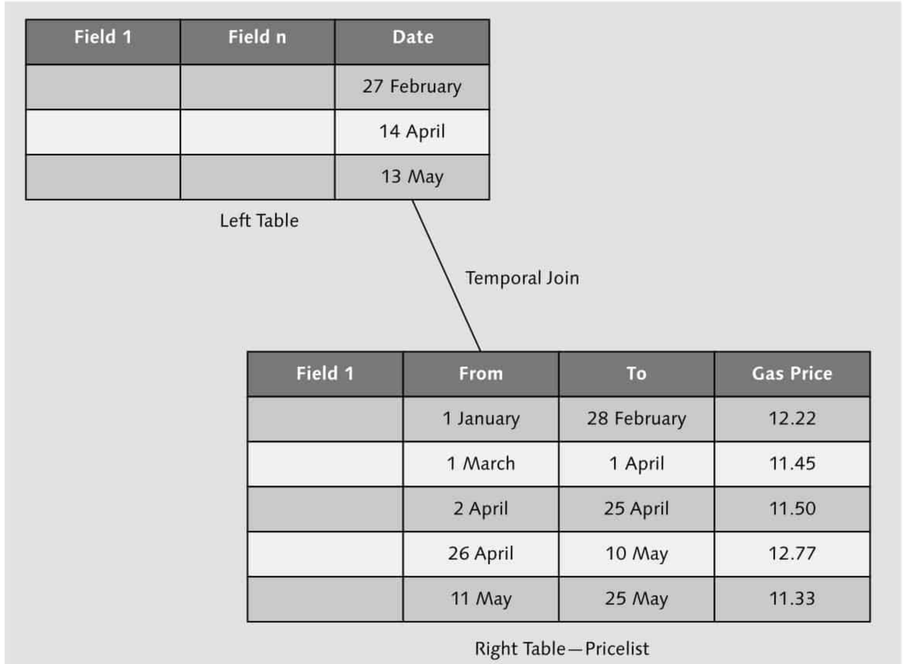


Figure 4.6 Temporal Join for Gasoline Prices

This date range lookup can be a little more complicated in a database. Sometimes, programmers read the data into an application server and then loop through the different records. In this case, a temporal join makes it easy, because SAP HANA will perform the date lookups in the `FROM` and the `TO` fields automatically for you, compare it to the date you've supplied, and get you the right fuel price at that specific date. It will perform all the calculations automatically for you—a great time and effort saver.

A temporal join uses either a referential join; or an inner join to do the work. A temporal join requires valid to and from dates (in the gasoline price table) and a valid date-time column (in your car log book table).

Spatial Join

Spatial joins became available in SAP HANA SPS 09. SAP HANA provides spatial data types, which we can use, for example, with maps. These all have the prefix *ST_*. A location on a map, with longitude and latitude, would be an `ST_POINT` data type. (We will discuss spatial data and analytics further in Chapter 9.)

We can use special spatial joins between tables in SAP HANA (see Figure 4.7). Say that you have a map location stored in a `ST_POINT` data type in one table. In the other table, you have a list of suburbs described in a `ST_POLYGON` data type. You can now join these two tables with a spatial join and define the spatial join to use the `ST_CONTAINS` method. This will calculate for each of the locations (`ST_POINT`) in which suburb (`ST_POLYGON`) they are located (`ST_CONTAINS`).

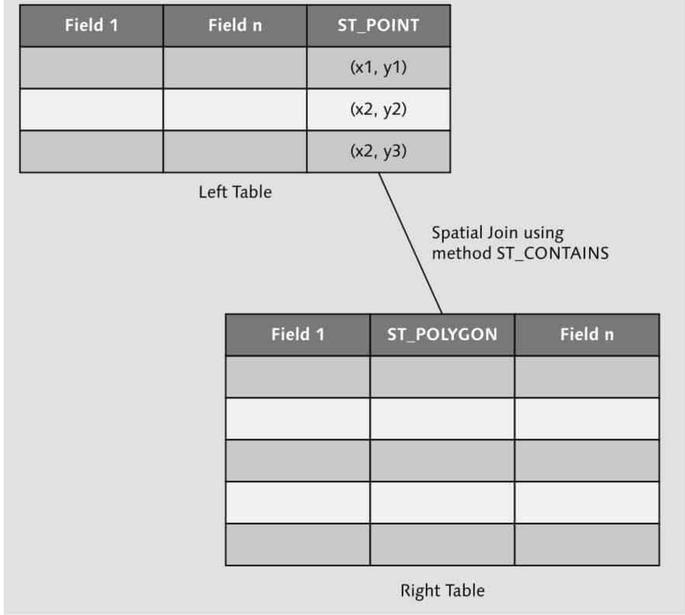


Figure 4.7 Spatial Join

There are about a dozen methods like `ST_CONTAINS` available for spatial joins. For example, you can test if a road crosses a river, if one area overlaps another (e.g., mobile phone tower reception areas), or how close certain objects are to each other.

We have only mentioned the two-dimensional aspects in relation to maps, but many of these spatial functions can be used in three dimensions.

Dynamic Joins

A dynamic join is not really a join type; it's a join property. It is also a performance enhancement available for SAP HANA. Once you have defined a join, you can mark it as a dynamic join. For this to work, you have to join the tables on multiple columns.

Let's assume you define a static (normal) join on columns A, B, and C between two tables, as shown in Figure 4.8. In this case, the join criteria on all three columns will be evaluated every time the view is called in a query.

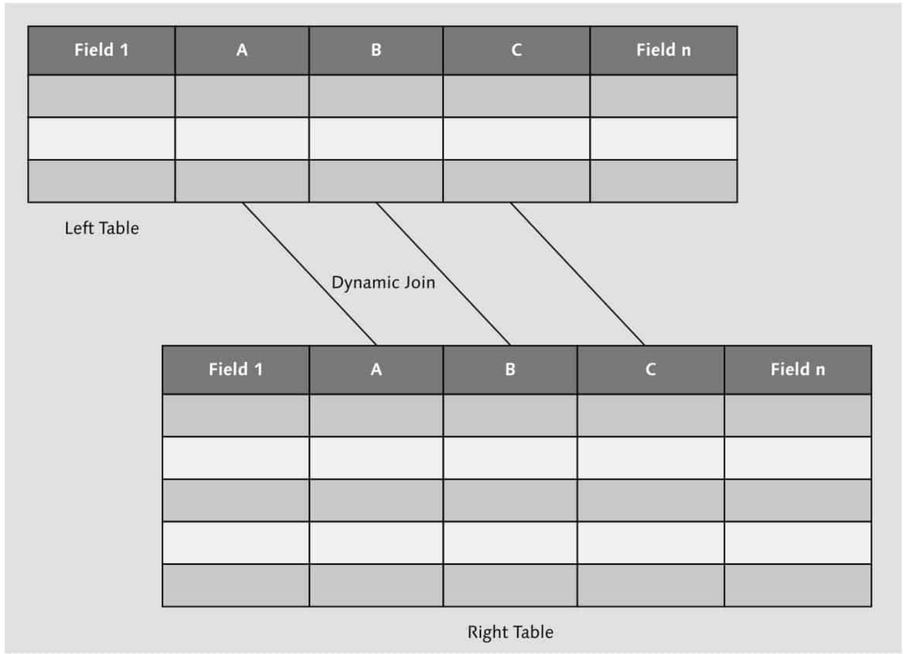


Figure 4.8 Dynamic Join on Multiple Columns

If you now mark the join as *dynamic*, the join criteria will only be evaluated on the columns involved in the query to the view. If you only ask for column A in your query of the view, then only column A's join criteria will be evaluated. The join criteria on columns B and C will not be evaluated, and your system performance improves.

Tip

If your query does not request any of the join columns in the dynamic join, it will produce a runtime error.

Core Data Services Views

CDS is a modeling concept that is becoming more important with every new release of SAP HANA. To understand why we need CDS, it is important to remember the wider context of how SAP HANA systems are deployed.

While learning SAP HANA, you work with a single system, but productive environments normally include development, quality assurance, and production systems. You perform development in the development environment, in which you have special modeling and development privileges. Your models and code then enter the quality assurance system for testing. When everyone is satisfied with the results, the models and code move to the production system. You do not have any modeling and development privileges in the quality assurance and production systems; you need a way to create views and other database objects in the production system, ideally without giving people special privileges.

An SAP HANA system can be deployed as the database for an SAP business system. In such a case, ABAP developers do not have any modeling and development privileges, even in the development system. We need a way to allow ABAP developers to create views and other database objects in the production system without these special privileges.

With that in mind, let's look at why CDS is a good idea with the following example:

You need to create a new schema in your SAP HANA database. It's easy to use a short SQL statement to do this, but remember the wider context. Someone will need the same create privileges in the production system! We want to avoid this, because giving anyone too much authorization in a production system can weaken security and encourage hackers.

CDS allows for a better way to create such a schema: Specify the schema creation statement once by creating a CDS (text) file with the `schema_name="DEMO";`

instruction. When we take this CDS file to the production system, the SAP HANA system automatically creates the schema for us. In the background, SAP HANA automatically converts the CDS file into the equivalent schema creation SQL statement and executes this statement.

The system administrators do not get privileges to create any schemas in the production system. They have rights to import CDS files, but they have no control over the contents of the CDS files; that's decided by the business. CDS thus gives us a nice way to separate what the business needs from what database administration can do.

There's much more to CDS. You can specify table structures (metadata) and associations (relationships) between tables with CDS, and taking this CDS file to production creates a new table. Even more impressive, if you want to modify the table structure later, SAP HANA does not delete the existing tables and recreate them; it automatically generates and executes the correct SQL statements to modify only the minimal changes to the existing table structures—for example, adding only a new column. CDS does not replace SQL statements, because it ultimately translates back into SQL statements. CDS adds the awareness to use a table creation SQL statement the first time it's run but a table modification SQL statement the second time it's run.

We can also create views using CDS. These CDS views can be used as data sources in our SAP HANA information views. Please note that these CDS views do not replace the more powerful information views we can create in SAP HANA.

ABAP developers can also use CDS inside ABAP. When a CDS view is sent to the SAP HANA system, it creates the necessary SAP HANA database objects without the ABAP developer having to log into the SAP HANA database. Because ABAP is database-independent, the same CDS file sent to another database will create the database objects relevant to that database. Because of this database independence, the versions of CDS for ABAP and for SAP HANA have slight differences.

The example screens in this book make use of the SHINE demo package described in Chapter 2. SHINE is a good example of CDS in action. You do not have to create a schema, create tables, or import data into these tables. When SHINE is imported into your SAP HANA system, it automatically creates all that content for you by using CDS statements.

Now, let's take our knowledge of these concepts to the next "dimension."

Cube

One of the most important modeling concepts we will cover is the *cube*. We'll expand on this concept when we discuss the information views available in SAP HANA. In this section, we'll use a very simplistic approach to describing cubes to give you an understanding of the important concepts.

When you look at a cube the first time, it looks like multiple tables that are joined together—and at a database level, that might be true. What makes a cube a cube are the "rules" for joining these tables together.

The tables you join together contain two types of data: transactional data and master data. As shown in Figure 4.9, we have a main table in the middle called a *fact table*. This is where our transactional data is stored. The transactional data stored in the fact table is referred to as either *key figures* or *measures*. In SAP HANA, we just use the term *measures*.

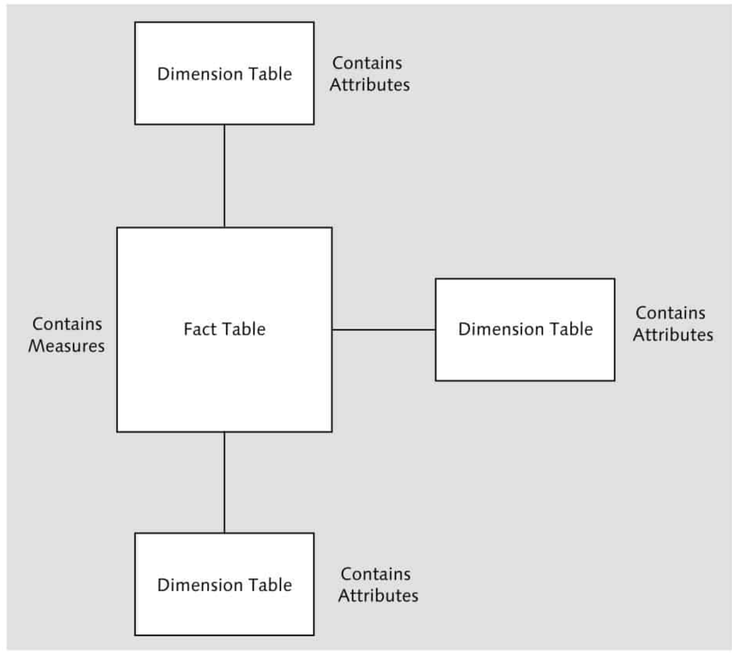


Figure 4.9 Cube Features

The side tables are called *dimension tables*, and this is where our master data is stored. The master data that is stored in the dimension table are referred to as

characteristics or *attributes*, or sometimes even *facets*. In SAP HANA, we just use the term *attributes*.

To clear up these statements, let's walk through an example. Our cube example will be simplified to communicate the most basic principles.

In our example, we'll work with the sentence "Laura buys 10 apples." You will agree that this sentence describes a financial transaction: Laura went to the shop to buy some apples, she paid for them, and the shop owner made some profit.

Where will we will store this transaction's data in the cube? There are a couple of "rules" that we have to follow. The simplest rule is that you store the data with which you can perform calculations on in the fact table, and you store everything else that you can't perform calculations on in the dimension tables.

In our transaction of "Laura buys 10 apples," we first look at *Laura*. Can we perform a calculation on Laura? No, we can't. Therefore, we should put her name into one of the dimension tables. We put her name in the top dimension table in Figure 4.10. Laura is not the only customer of that shop; her name is stored with those of many other people in this dimension table. We will call this table the Customer dimension table.

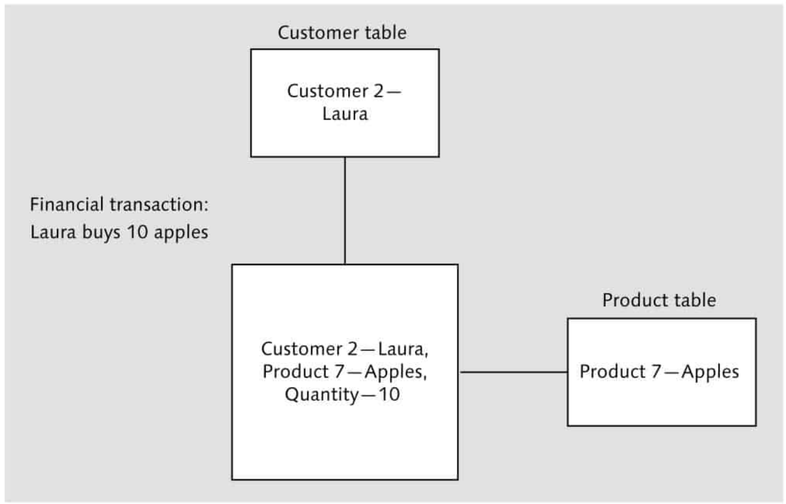


Figure 4.10 Financial Transaction Data Stored in a Cube

Next, let's look at the apples. Can we perform calculations on apples? No, we can't, so again this data goes into a dimension table. The shop doesn't sell only apples; it also sells a lots of other products. The second dimension table is therefore called the Product dimension table.

In an enterprise data warehouse, we would normally limit the number of dimension tables that we link to the fact table for performance reasons. Typically, in an SAP BW system, we limit the number of dimension tables to a maximum of 16 or 18.

Finally, let's look at the number of apples, 10. Can we perform a calculation on that? Yes, because it's a number. We therefore store that number in the fact table.

To complete the picture, if Laura buys more apples, then she is a top customer; let's say she's customer number 2. Apples are something the store keeps in stock, so apples are product number 7.

If we store a link (join) to customer 2 and another link (join) to product 7 and the number 10 (number of apples), we would say that that constitutes a complete transaction. We can abbreviate the transaction "Laura buys ten apples" to 2, 10, and 7 in our specific cube.

Laura and apples are part of our master data. The transaction itself is stored in the fact table. We've now put a real-life transaction into the cube.

Hopefully, this gives you a better idea of what a cube is, and how to use it. Let's complete this short tour of cubes by clarifying a few terms ahead.

Attributes and Measures

As previously shown in Figure 4.9, attributes are stored in dimension tables, and measures are stored in fact tables. Let's define attributes and measures:

- **Attributes**
  Attributes describe a transaction, like the customer Laura or the product apples. This is also referred to as master data. We cannot perform calculations on attributes.
- **Measures**
  Measures are something we can *measure* and perform calculations on. Measurable (transactional) information goes into the fact table.

Fact Tables in the Data Foundation

Figure 4.10 showed only a single fact table, but sometimes we have more complex cubes that include multiple fact tables. In SAP HANA, we call a group of fact tables (or the single fact table) the *data foundation*.

When we join the data foundation (fact tables) to the dimension tables, we always put the data foundation on the left side of the join.

Star Joins

The join between the data foundation and the dimension tables is normally a referential join in SAP HANA. We refer to this as a *star join*, which indicates that this is not just a normal referential join between low-level tables, but a join between a data foundation and dimension tables. This is seen as a join at a higher level.

Originally, we called these *logical joins* in SAP HANA, however they were renamed star joins as of SAP HANA SPS 10.

With all the building blocks in place, let's start examining SAP HANA information views.

Information Views

Let's start our introduction to SAP HANA information views by looking back at what we learned when we discussed cubes.

Dimension Views

The first thing we need to build cubes is the master data. Master data is stored in dimension tables. Sometimes, these dimension tables are created for the cubes. In SAP HANA, we do not create dimension tables as separate tables; instead we build them as views.

We join all the low-level database tables that contain the specific master data we are interested in into a view. For example, we might join two tables in a view to build a new Products dimension in SAP HANA, and maybe another three tables to build a new Customers dimension.

In this book, we will refer to these type of views simply as *dimension views*. In SAP HANA, there are currently two types of dimension views available:

- **Attribute views**
  The older version of dimension views are called *attribute views*. This name refers to the data, called *attributes*, stored in dimension tables. Attribute views are still used in exceptional cases in SAP HANA SPS 11.
- **Calculation views of type dimension (dimension calculation views)**
  As of SAP HANA SPS 10, we now use *calculation views of type dimension*, also called *dimension calculation views*. In SAP HANA SPS 11, a new migration tool is available to help migrate your old attribute views to the new dimension calculation views.

Just as the same master data is used in many cubes, so will we reuse dimension views in many of the other SAP HANA information views. You can even build your own "library" of dimension views.

Star Join Views

In the same way, instead of storing data in cubes, we can create another type of view in SAP HANA that produces the same results as a traditional cube. There are currently two types of SAP HANA views that produce the same results as an old-fashioned, traditional cube:

- **Analytic views**
  We still have an older version of these SAP HANA views called *analytic views*. This name refers to the fact that we use cubes for analytics. Analytic views are still used in some cases in SAP HANA SPS 11.
- **Calculation views of type cube with star join**
  As of SAP HANA SPS 10, we now use *calculation views of type cube with star join*. In SAP HANA SPS 11, a new migration tool is available to help you to migrate your old analytic views to the new calculation views of type cube with star join.

Terminology

In this book, we will simply refer to these types of views as *star join views*, not as *cube views*, which would lead to confusion. This is because the third type of SAP HANA views is called *calculation view of type cube*.

The results created by the star join views in SAP HANA are the same as that of a cube in an enterprise data warehouse, except that in a data warehouse, the data is stored in a cube. With SAP HANA, the data is not stored in a cube, but is calculated when required.

Calculation Views of Type Cube

This view type, provides even more powerful capabilities. For example, you can combine multiple star join views ("cubes") to produce a combined result set. In SAP Business Warehouse (BW), we call this combined result set a *MultiProvider*.

In SAP HANA, we call these *calculation views of type cube*. Use cases for this type of view include:

- In your holding company, you want to combine the financial data from multiple subsidiary companies.
- You want to compare your actual expenses for 2015 with your planned budget for 2015.
- You have archived data in an archive database that you want to combine with current data in SAP HANA's memory to compare sales periods over a number of years.

Using Information Views

Let's examine how to use the different types of SAP HANA information views together in information modeling. Figure 4.11 provides an overview of everything you've learned in this chapter and how each topic fits together.

On the top left-hand side, you can see our source systems: an SAP source system and a non-SAP source system. SAP HANA doesn't care from which system the data comes. In many projects, we have more non-SAP systems that we get data from than SAP systems. We extract the data from these systems using various data provisioning tools, which we'll look at in Chapter 14.

The data is stored in row tables in these source systems. When we use our data provisioning methods and tools, the data is put into column tables in the SAP HANA database. This is illustrated in the bottom-left corner of Figure 4.11.

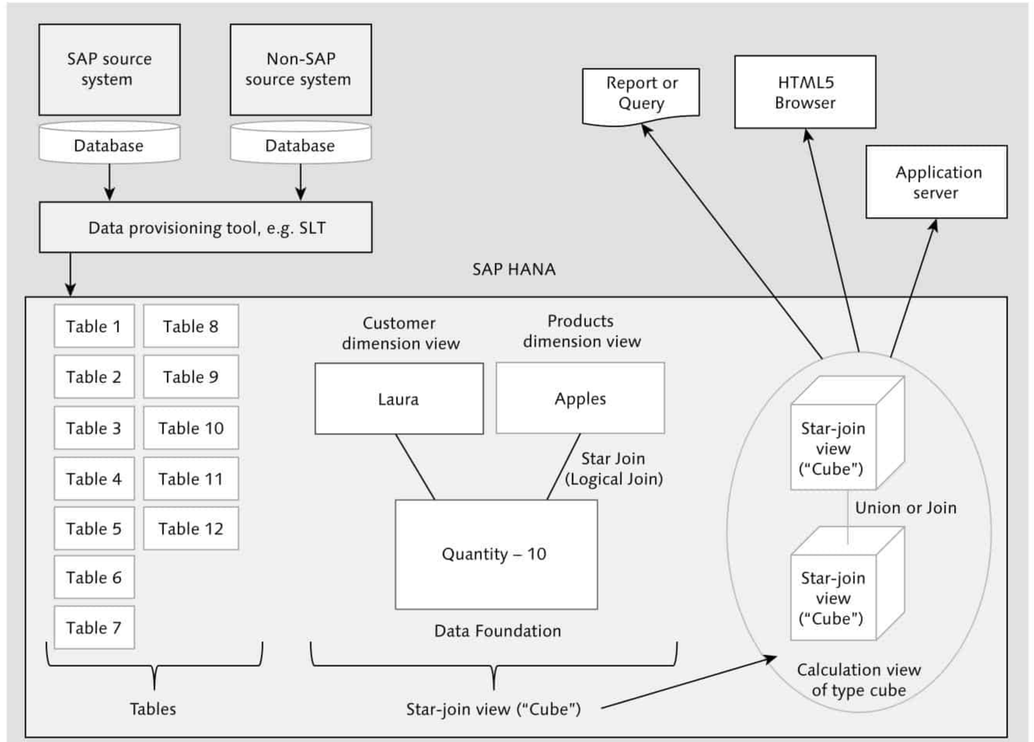


Figure 4.11 Different SAP HANA Information Views Working Together

Building the Information Model

Now, let's start building our information models on top of these tables.

If we want to build something like a cube for our analytics or reporting, we use SAP HANA information views. First, we need the equivalent of a dimension table, such as our Product or Customer dimension table. How would we build this?

We will build a dimension view (a calculation view of type dimension, or an attribute view). This is indicated in Figure 4.11 by Laura and Apples, which represent the Customer and Product dimension views. We create these dimension views the same way we create any other view: Take two tables, join them together, select certain output fields, and save it.

Once we have our dimension views, we build a data foundation for the cube. The data foundation is for all our transactional data. The data foundation stores measures, like the number *10* for the 10 apples that Laura bought. We build a data foundation in a similar fashion to building a dimension view: We take multiple tables or a single table. If using multiple tables, we join them together, select the output fields, and build our data foundation.

To complete the cube, we have to link our data foundation (fact tables) to the dimensions with a star join. Finally, to complete our star join view, we also select the output fields for our cube. The created star join view will produce the same result as a cube, but in this case no data is stored and it performs much better.

In the next step, we can create calculation view of type cube, in which we combine two star join views (cubes). We can combine the star join views using a join or a union. We recommend a union in this case, because it leads to better performance than a join.

Finally, we can build reports, analytics, or applications on our calculation view. This is shown in the top right of Figure 4.11.

Using the Information Model

In the previous section, we looked at how to build an information model from the bottom up. Let's now think about the process from the other side and look at what happens when we call our report or application that built on the calculation view. This is the top-down view of Figure 4.11.

The calculation view does not store the data: It has to gather and calculate all the data required by the report. SAP HANA looks at the calculation view's description and says, "Oh, I've got to build two star join views and union them together. How do I build these star join views?"

SAP HANA then goes down one level and says, "OK, how do I build the first star join view? I need a data foundation, and I need two dimension views, and I join them together with a star join."

Then, it goes down another level: "How do I build the dimension view? I'll read two different tables, join them together, and select the output fields." SAP HANA then builds the two dimension views.

Going back up one level, SAP HANA needs to combine these two dimension tables with a data foundation. It creates the data foundation and joins it with the two dimension tables. It also builds the second star join view in the same way. Going up to the level of the calculation view, it combines the two star join views with a union. Finally, SAP HANA sends the result set out to the report, and then cleans out the memory again. (It does not store this data!) If someone else asks for the exact same data five minutes later, SAP HANA happily performs the same calculations all over again.

At first, this can seem very wasteful. Often we're asked "Why don't you use some kind of caching?" However, *caching* means that we store data in memory. In SAP HANA the entire database is already in memory. We don't need caching because everything is in memory already.

The reason people ask this question is that in their traditional data warehouses, data normally doesn't change until the next evening when the data is refreshed. Inside SAP HANA, the data can be updated continuously, and we always want to use the latest data in our applications, reports, and analytics. That means we have to recalculate the values each time; caching static result sets will not work.

In summary, to use real-time data, you have to continuously recalculate.

Other Modeling Artifacts

We will now discuss some of the other modeling concepts we will use when building our information views.

The power of calculation views of type cube is that you can build them up in many layers. In the bottom layer, you can perhaps have a union of two star join views (cubes), as shown in Figure 4.12. In the next (higher) layer, we sum up all the values from the union. In the level above that, we rank the summed results to show the top 10 most profitable customers.

Projection

A *projection* is used to change an output result set. For example, we can get a result set from a star join view (cube) and find that it has too many output fields for what we need in the next layers. Therefore, we use a projection as an additional in-between layer to create a subset of the results.

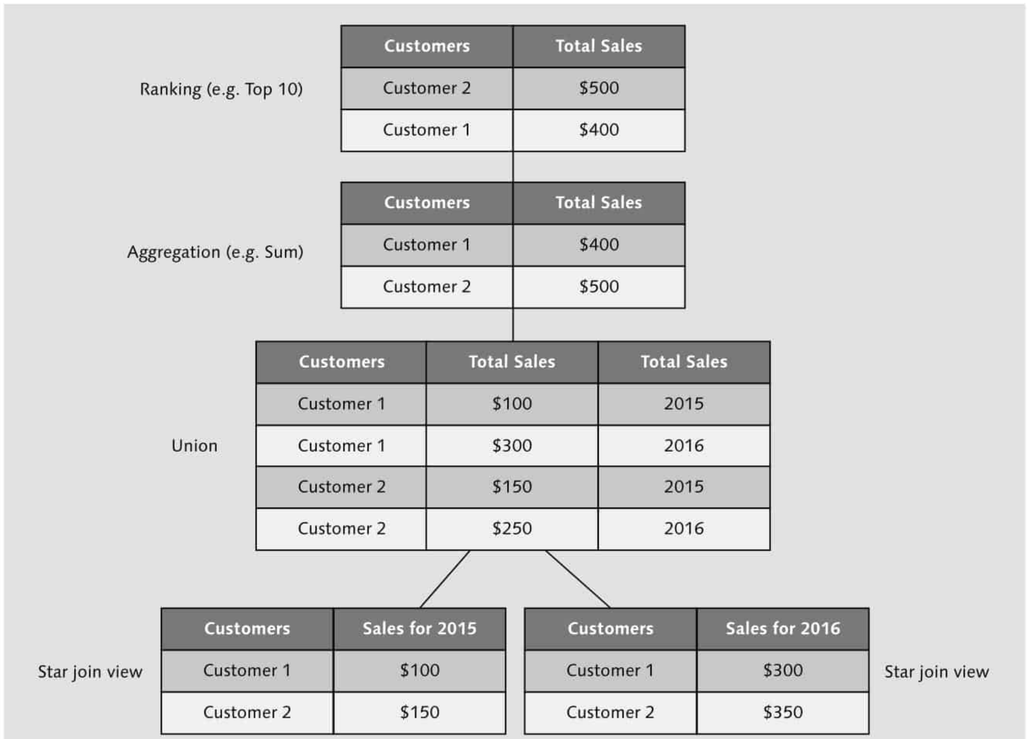


Figure 4.12 Calculation View of Type Cube Built in Various Layers

We can also use the projection to calculate additional fields—for example, sales tax—or we can rename fields to give them more meaningful names for our analytics and reporting users.

Ranking

As illustrated in the top node of Figure 4.12, we can use *ranking* to create useful top-N or bottom-N reports and analytics. This normally returns a filtered list (e.g., only the top 10 companies) and is sorted in the order we specify.

Aggregation

The word *aggregation* means a collection of things. In SQL, we use aggregations to calculate values such as the sum of a list of values. Aggregations in databases provide the following functions:

- **Sum**
  Adds up all the values in the list
- **Count**
  Counts how many values (records) there are in the list
- **Minimum**
  Finds the smallest value in the list
- **Maximum**
  Finds the largest value in the list
- **Average**
  Calculates the arithmetical mean of all the values in the list
- **Standard deviation**
  Calculates the standard deviation of all the values in the list
- **Variance**
  Calculates the variance of all the values in the list

Normally, we would only calculate aggregations of measures. (Remember that measures are values we can perform calculations on, which usually means our transactional data.) For example, we could calculate the total number of apples a shop sold in 2015. However, in the latest versions of SAP HANA, we can also create aggregations of attributes, essentially creating a list of unique values.

Union

A *union* combines two result sets. Figure 4.12 illustrates this by combining the results of two star join views.

In SAP HANA graphical calculation views, we're not restricted to combining two result sets; we can combine multiple result sets. SAP HANA merges these multiple result sets into a single result set.

Let's say we have four result sets—A, B, C, and D—that we union together. Although we union multiple result sets, that doesn't mean the performance will

be bad. If SAP HANA detects that you are not asking for data from A and D, it will not even generate the result sets for A and D; it will only work with B and C.

Note

The union expects the two result sets to have the same number of columns, in the same order, and the matching columns from each result set must have similar data types.

You can have variations in unions, such as the following:

- **Union all**
  Will merge the result sets and return all records
- **Union**
  Will merge the results and only give you the unique results back
- **Union with constant values**
  Lets you "pivot" your merged results to help get the data ready for reporting output

Figure 4.13 compares the output generated by a union to that of a union with constant values.

Customers	Total Sales	Total Sales
Customer 1	$100	2015
Customer 1	$300	2016
Customer 2	$150	2015
Customer 2	$250	2016

Standard Union

Customers	Sales for 2015	Sales for 2016
Customer 1	$100	$300
Customer 2	$150	$350

Union with constant values

Customers	Sales for 2015
Customer 1	$100
Customer 2	$150

Customers	Sales for 2016
Customer 1	$300
Customer 2	$350

Figure 4.13 Standard Union Compared to Union with Constant Values

Often, we want a report that looks like the top right of the figure. If we're given the output from the standard union, it takes more work to create a report with sales for 2015 and 2016 in separate columns. With the output from the union with constant values, writing the report is easy.

In Figure 4.11, we showed that you should use a union rather than a join for performance reasons—but what happens if the two star join views that you want to union together do not have the same outputs? In that case, the union will return an error. How do we solve this problem?

Figure 4.14 illustrates an example. On top, we have the ideal case in which both star join views have matching columns. In the middle, we have a case in which only column B matches. Normally, we would use a join, as shown.

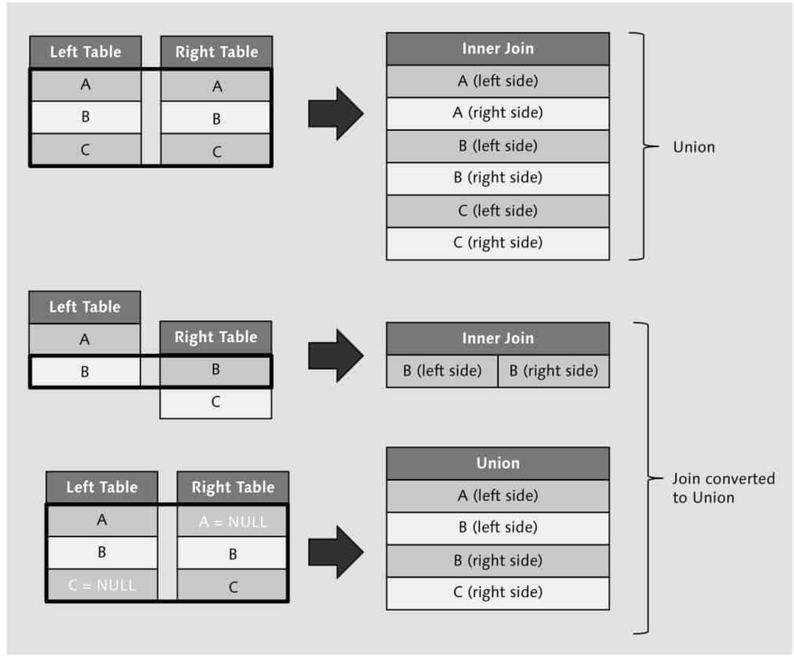


Figure 4.14 Standard Union, Join, and Join Converted to Union

We solve this problem by creating a projection for each of the two star join views. We add the missing columns and assign a `NULL` value to those columns. We then perform the union on the two projections, because they now have matching columns.

Semantics

An important step that many modelers skip is to add more meaning to the output of their information models. *Semantics*, as a general term, is the study of meaning. We can use semantics in SAP HANA to, for example, rename fields in a system so they are more clear. Some database fields have really horrible names. Using semantics, you can change the name of these fields to be more meaningful for your end users. This is especially useful when they create reports. If you have a reporting background, you will know that a reporting universe can fulfill a similar function.

We can also show that certain fields are related to each other—for example, a customer number and a customer name field.

One example that is used often is that of building a hierarchy. By using a hierarchy, you can see how fields are related to each other with regards to "drilling down" for reports.

Hierarchies

Hierarchies are used in reporting to enable intelligent drilldown. Figure 4.15 illustrates two examples: Users can start with a list of sales for all the different countries; they can then drill down to the sales of the provinces (or states) of a specific country, and finally down to the sales of a city.

End users can see the sales figures for just the country, state, or province or for a specific city. Even though countries and cities are two different fields, they're related to each other by use of a hierarchy. This extra relationship enhances the data and gives it more meaning (semantic value).

There are two different types of hierarchies: *level hierarchies* and *parent–child hierarchies*. The hierarchy we just described is the level hierarchy; level 1 is the country, level 2 is the state or province, and level 3 is the city or town.

Another example of a level hierarchy is a *time-based hierarchy* that goes from a year to a month to a day. You can even keep going deeper, down to seconds. This is illustrated on the right side of Figure 4.15.

HR employee organizational charts or a cost center structures are examples of parent–child hierarchy, as illustrated in Figure 4.16. In this case, we join a table to itself, as described in the Self-Joins section.

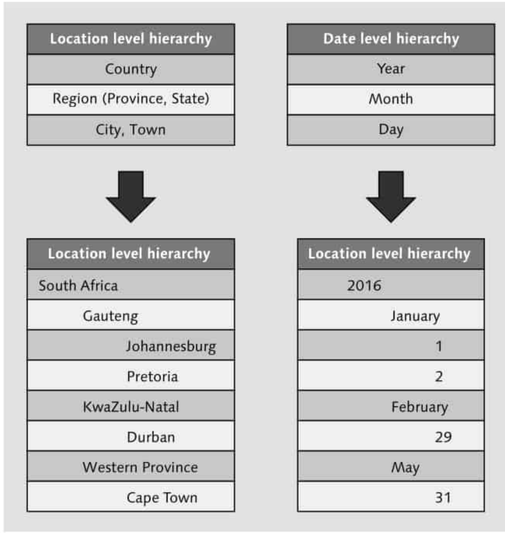


Figure 4.15 Examples of Level Hierarchies

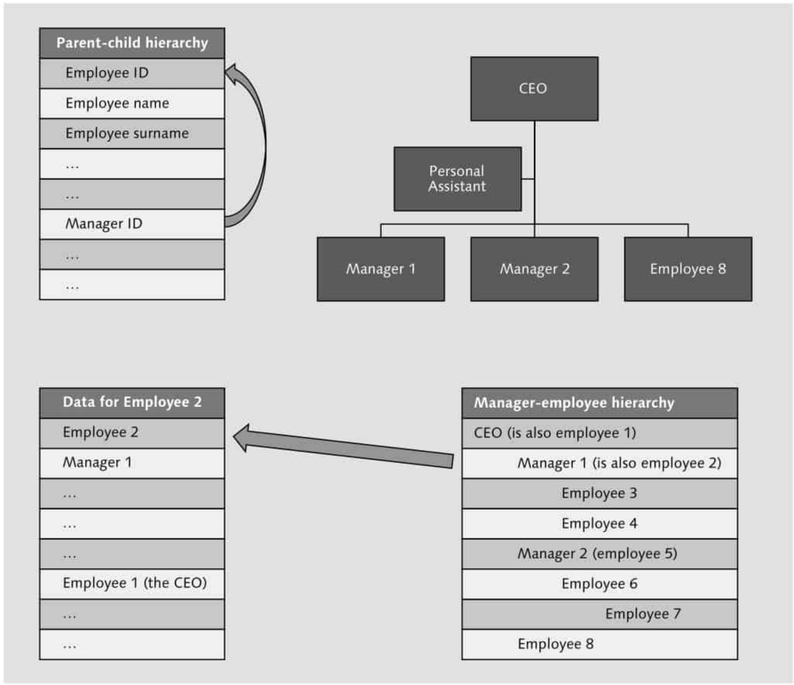


Figure 4.16 Parent–Child Hierarchy Example

Table 4.1 provides a quick summary of the differences between the two types of hierarchies and when you should use which one.

Level Hierarchies	Parent–Child Hierarchies
Fixed (rigid) number of levels in the hierarchy.	Variable hierarchy depth.
Different data fields on every level of the hierarchy (e.g., country, state, city).	The same (two) fields used on every level of the hierarchy.
The fields on each level of the hierarchy can be different data types (e.g., country can be text, while zip/postal code can be numeric).	The parent and child fields have the same data type.

Table 4.1 Comparing Level Hierarchies and Parent–Child Hierarchies

Best Practices and Modeling Guidelines

Let's end our tour of modeling concepts with a summary of best practices for modeling views in SAP HANA. Many of the basic modeling guidelines that we find in traditional data modeling and reporting environments are still applicable in SAP HANA. We can summarize the basic principles as follows:

- **Limit (filter) the data as early as possible**
  It's wasteful to send millions of data records across a slow network to a reporting tool when the report only uses a few of these data records. It makes a lot more sense to send only the data records that the report requires, so we want to filter the data before it hits the network.

  In the same way, we don't want to send data to a star join view when we can filter it in the earlier dimension view. Hence, the rule to filter as early as possible.

- **Perform calculations as late as possible**
  Assume you have column A and column B in a table. The table contains billions of records. We can use the values of column A and column B in a formula for every data record, which means the server performs billions of calculations, and finally add up the sum total of the calculated fields.

  It would be much faster to add up the sum total for column A and the sum total for column B and then calculate the formula. In that case, we perform the calculation once instead of billions of times. We delay the calculation as long as possible for performance reasons.

Important Terminology

In this chapter, we focused on various modeling concepts and how they're used in SAP HANA. We introduced the following important terms:

- **Views**
  We started at by looking at *views* and realized that we can leverage the fact that SAP HANA already has all the data in memory, that the servers have lots of CPU cores available, and that SAP HANA runs everything in parallel. We therefore do not need to store the information in cubes and dimension tables, but can use views to generate the information when we need it. This also gives us the advantage of real-time results.

- **Joins**
  We have both normal database join types, like inner and left outer joins, and SAP HANA-specific types, like referential, text, temporal, and spatial joins. In the process, we also looked at the special case of self-joins and at dynamic joins as an optimization technique.

- **Core Data Services (CDS)**
  CDS provides a way to separate business semantics and intent from database operations.

- **Cubes**
  A cube consists of a data foundation with fact tables that contain the transactional data, linked to dimension tables that contain master data. The data foundation is linked to the dimension tables with star joins.

- **Attributes and measures**
  Attributes describe elements involved in a transaction. Transactional data that we can perform calculations on is called a measure and stored in the fact tables.

- **Information views**
  In SAP HANA, we take the concept of views to higher levels. We can create

dimension tables and cubes as dimension views and star join views. Dimension views are attribute views and calculation views of type dimension. Star join views are analytic views and calculation views of type cube with star join. The final type of information view is the calculation view of type cube, which we can use to perform even more powerful processes.

- **Modeling artifacts**
  Inside our views, we can use unions, projections, aggregations, and ranking. Unions are quite flexible and can be used for a direct merge of result sets, or we can use a union with constant values to create report-ready result sets.

Practice Questions

These practice questions will help you evaluate your understanding of the topics covered in this chapter. 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 that on the exam you must select all correct answers and only correct answers to receive credit for the question.

In this section, we have a few questions with graphics attached. In the certification exam, you might also see a few questions that include graphics.

1. In which SAP HANA views will you find measures? (There are 2 correct answers.)

☐ **A.** Attribute views
☐ **B.** Calculation view of type cube with star join
☐ **C.** Calculation view of type cube
☐ **D.** Database views

2. True or False: A database view stores data in the database.

☐ **A.** True
☐ **B.** False

3. A traditional cube is represented by which SAP HANA view type? (There are 2 correct answers.)

☐ **A.** Attribute view
☐ **B.** Analytic view
☐ **C.** Calculation view of type cube with star join
☐ **D.** Calculation view of type dimension

4. A referential join gives the same results as which other join type?

☐ **A.** Inner join
☐ **B.** Left outer join
☐ **C.** Spatial join
☐ **D.** Star join

5. Which join type makes use of date ranges?

☐ **A.** Spatial join
☐ **B.** Text join
☐ **C.** Temporal join
☐ **D.** Inner join

6. If we change the transaction "Laura buys 10 apples" to "Laura buys 10 green apples," how would we store the color *green*?

☐ **A.** Store *green* as an attribute
☐ **B.** Store *green* as a measure
☐ **C.** Store *green* as a spatial data type
☐ **D.** Store *green* as a CDS view

7. True or False: A view always shows all the available columns of the underlying tables.

☐ **A.** True
☐ **B.** False

8. You have a view with two tables, joined by a left outer join. If you redesign the view and accidently swap the two tables around, what should you do to the join?

- ☐ **A.** Keep the left outer join.
- ☐ **B.** Change the join to a text join.
- ☐ **C.** Change the join to a right outer join.
- ☐ **D.** Change the join to a referential join.

9. What do you call the data displayed in the data foundation of an SAP HANA information view?

- ☐ **A.** Facets
- ☐ **B.** Measures
- ☐ **C.** Characteristics
- ☐ **D.** Key figures

10. You are writing a mobile application for the World Series. You have details about all the baseball players and the baseball scores for all the previous matches. How do you use the data of the player information and the scores?

- ☐ **A.** Both the player information and the scores are used as master data.
- ☐ **B.** Both the player information and the scores are used as transactional data.
- ☐ **C.** The player information is used as master data, and the scores are used as transactional data.
- ☐ **D.** The player information is used as transactional data, and the scores are used as master data.

11. Look at Figure 4.17. You are selling books that have been translated into various languages. What join type should you use?

- ☐ **A.** Left outer join
- ☐ **B.** Text join
- ☐ **C.** Temporal join
- ☐ **D.** Referential join

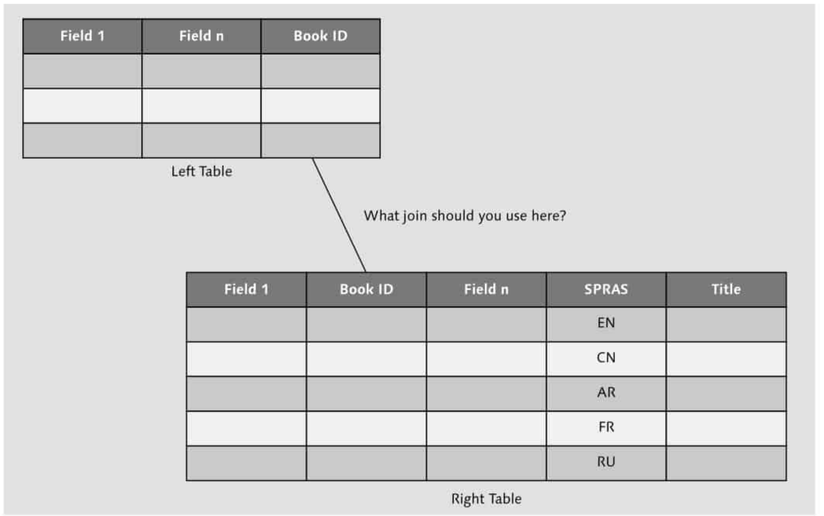


Figure 4.17 What Type of Join Should You Use?

12. Which of the following are reasons you should use Core Data Services (CDS) to create a schema? (There are 2 correct answers.)

- ☐ **A.** The database administrator might type the name wrong.
- ☐ **B.** The database administrator should not get those privileges in the production system.
- ☐ **C.** The focus is on the business requirements.
- ☐ **D.** The focus is on the database administration requirements.

13. Look at Figure 4.18. A company sends out a lot of quotes. Some customers accept the quotes, and they're invoiced. The company asks you to find a list of the customers that did NOT accept the quotes. How do you find the customers that received quotes but did NOT receive invoices?

- ☐ **A.**
  - ▸ Use a right outer join.
  - ▸ Filter to show only the `NULL` values on the right table.
- ☐ **B.**
  - ▸ Use a left outer join.
  - ▸ Filter to show only customers on the right table.

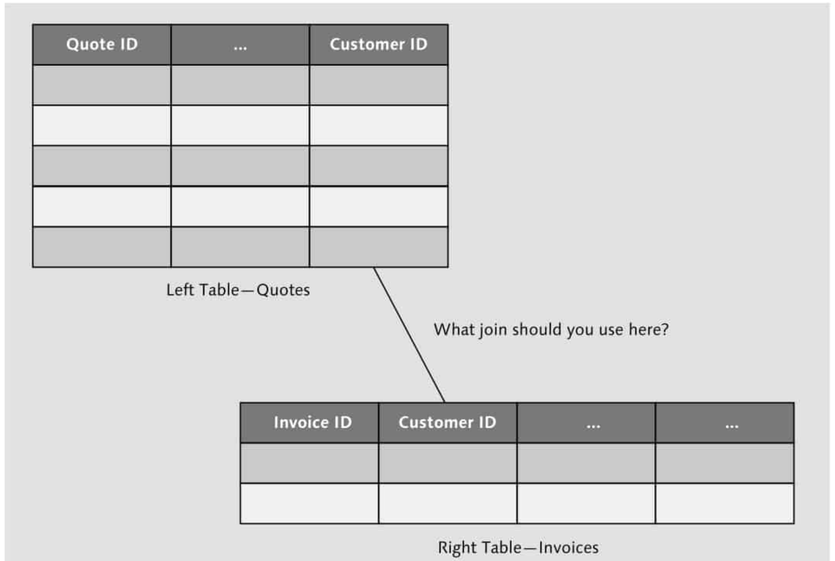


Figure 4.18 Find the Customers That Received Quotes but Not Invoices

☐ **C.**

- Use a left outer join.
- Filter to show only the `NULL` values on the right table.

☐ **D.**

- Use a right outer join.
- Filter to show only customers on the left table.

14. True or False: When you use a parent–child hierarchy, the depth in your hierarchy levels can vary.

☐ **A.** True

☐ **B.** False

15. Look at Figure 4.19. What type of join should you use? (*Hint:* Watch out for distractors.)

☐ **A.** Temporal join

☐ **B.** Text join

☐ **C.** Referential join

☐ **D.** Inner join

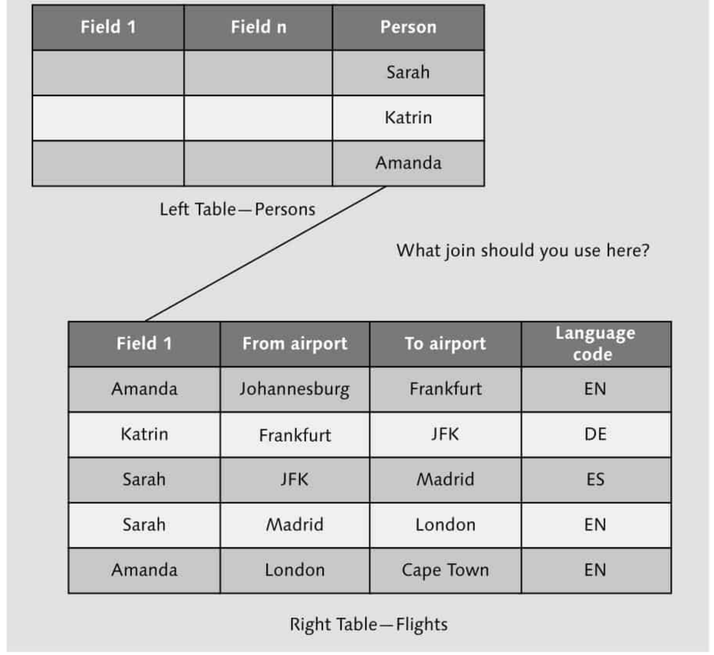


Figure 4.19 What Type of Join Should You Use?

16. You have a list of map locations for clinics. The government wants to build a new clinic, but wants to build it where there is the greatest need. You need to find the largest distance between any two clinics. With your current knowledge, how do you do this?

☐ **A.**

- Use a temporal join to find the distance between clinics.
- Use a union with constant values to "pivot" the values.

☐ **B.**

- Use a spatial join to find the distance between clinics.
- Use a level hierarchy for drilldown.

☐ **C.**

- Use a dynamic join to find the longest distance between clinics.
- Use a ranking to find the top 10 values.

☐ **D.**

- Use a spatial join to find the distance between clinics.
- Use an aggregation with the maximum option.

17. Look at Figure 4.20. What are the values for X, Y, and Z? (*Note:* You will not see a question like this on the certification exam; it's only added here to test your understanding.)

☐ **A.** X = Customer 2, Y = 400, Z = 800.
☐ **B.** X = Customer 1, Y = 400, Z = 400.
☐ **C.** X = Customer 1, Y = 800, Z = 800.
☐ **D.** X = Customer 2, Y = 200, Z = 400.

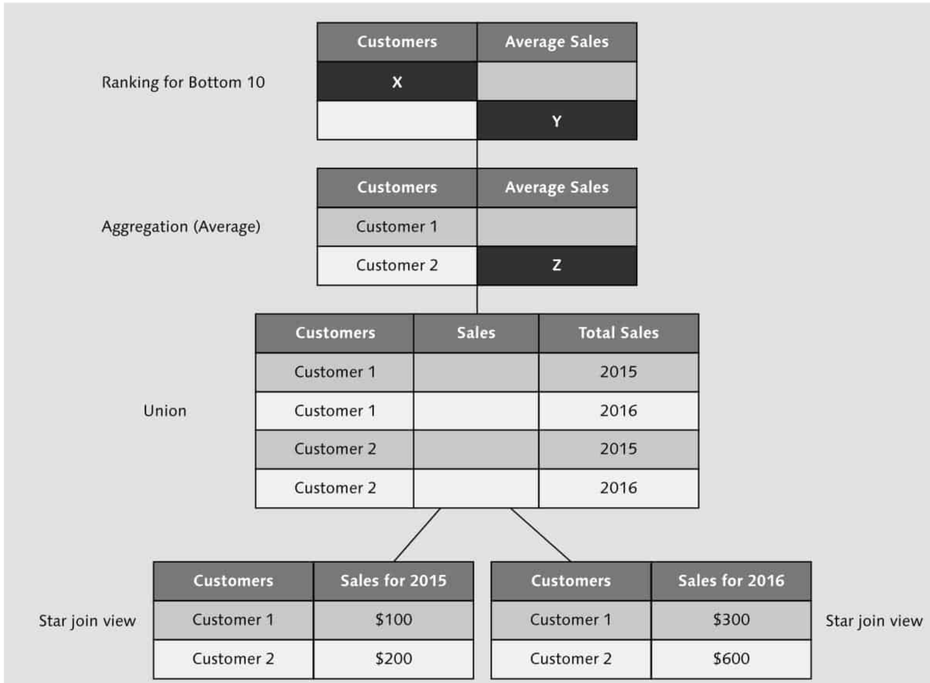


Figure 4.20 Find Values for X, Y, and Z

18. You have to build a parent–child hierarchy. What type of join do you expect to use?

☐ **A.** Relational join
☐ **B.** Temporal join
☐ **C.** Dynamic join
☐ **D.** Self-join

19. Look at Figure 4.21. What join type should you use to see all the suppliers?

☐ **A.** Left outer join
☐ **B.** Right outer join
☐ **C.** Inner join
☐ **D.** Referential join

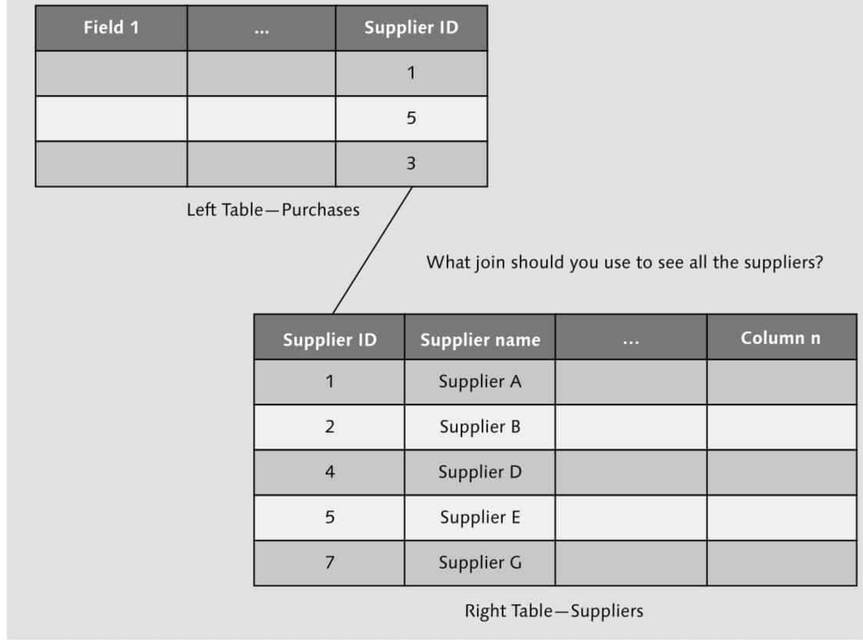


Figure 4.21 What Join Should You Use to See All Suppliers?

Practice Question Answers and Explanations

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

   You find measures in analytic views, calculation views of type cube with star join (the new type of analytic view), and calculation views of type cube. Attribute views do not contain measures—only attributes!

2. Correct answer: **B**

   False. A database view does not store data in the database.

3. Correct answers: **B, C**

   A traditional cube is represented by an analytic view or a calculation view of type cube with star join. Both are different from a cube in that they do not store any data like a cube does. An attribute view and a calculation view of type dimension are similar to a dimension table.

4. Correct answer: **A**

   A referential join gives the same results as an inner join but speeds up the calculations in some cases by assuming referential integrity of the data. It is the optimal join type in SAP HANA.

5. Correct answer: **C**

   A temporal join requires `FROM` and `TO` fields in the table on the right side of the join and a date-time column in the table on the left side of the join.

6. Correct answer: **A**

   We cannot "measure" or perform calculations on the color green. Green is not a spatial data type. CDS views are not relevant in this context.

7. Correct answer: **B**

   False. A view does NOT *always* show all the available columns of the underlying tables. You have to select the output fields of a view. The word *always* in the statement is what makes it false.

8. Correct answer: **C**

   A right outer join is the inverse of the left outer join. Therefore, if you reverse the order of the tables, you can "reverse" the join type. It is important to note that this does not work for a text join. A text join is only equivalent to a left outer join. For a text join, you have to be careful about the order of the tables.

9. Correct answer: **B**

   The data displayed in the data foundation of an SAP HANA information view is called measures. In SAP HANA, we talk about *attributes* and *measures*. The other names might be used in other data modeling environments, but not in SAP HANA.

10. Correct answer: **C**

    The player information is used as master data, and the scores are used as transactional data. You cannot perform calculations on the players. Therefore, this data is used in the dimensions and is seen as master data. You definitely perform calculations on the scores, so these are used in the fact tables, which means they are transactional data.

11. Correct answer: **B**

    Figure 4.17 shows the SPRAS column and shows a list of languages. The need for a translation is also hinted at in the question, meaning this should be a text join. There is not enough information to select any of the other answer options.

12. Correct answers: **B, C**

    You should use CDS to create a schema because the database administrator should not get those privileges in the production system and the focus should be on the business requirements.

13. Correct answer: **C**

    Look carefully at Figure 4.18. The question asks you to find all the customers that received quotes but did NOT receive invoices.

    The answer only gives us left outer or right outer as options. Because the question states they want *all* customers from the quotes tables, you might try a left outer join first.

    Now look at Figure 4.3 to see the difference between the left and right outer joins. Look especially at the where the `NULL` values are shown. With a left outer join, the `NULL` values are found in the right table. With a right outer join, the `NULL` values are found in the left table.

    We want to find the customers that did not get invoices. With a left outer join, the `NULL` values are found on the right side: That group is the one we are looking for. Therefore, filter for the `NULL` values on the right table.

Note

Any time that a negative phrase or word is used in the certification exam, it is shown in capital letters to bring attention to the fact that it is a negative word.

14. Correct answer: **A**

    True. Yes, the depth of your hierarchy levels can vary when you use a parent–child hierarchy.

15. Correct answer: **C**

    You should use a referential join.

    There are two distractors to try and throw you off track:

    - A temporal join uses `FROM` and `TO` fields, but only for date-time fields. This example is for airports.
    - The second distractor was to try to get you to select the text join: The language code has nothing to do with translations, but has to do with the airline's primary language.

    Choose a referential join because you can see the same people traveling in both tables, so it seems that these two tables have referential integrity.

    If the tables did not have referential integrity, you would have chosen an inner join.

16. Correct answer: **D**

    A spatial join can give the distance between different points on a map.

    There are two ways you can find the largest distance between any two clinics:

    - Use a ranking to find the top 10 values.
    - Use an aggregation with the maximum option.

17. Correct answer: **B**

    See the full solution in Figure 4.22.

    X = Customer 1, Y = 400, Z = 400.

    Note that we are using average in the aggregation node, not sum.

    In the ranking node, we are asking for the bottom 10, not the top 10.

18. Correct answer: **D**

    You expect to use a self-join when you build a parent–child hierarchy. See Figure 4.16 for an illustration.

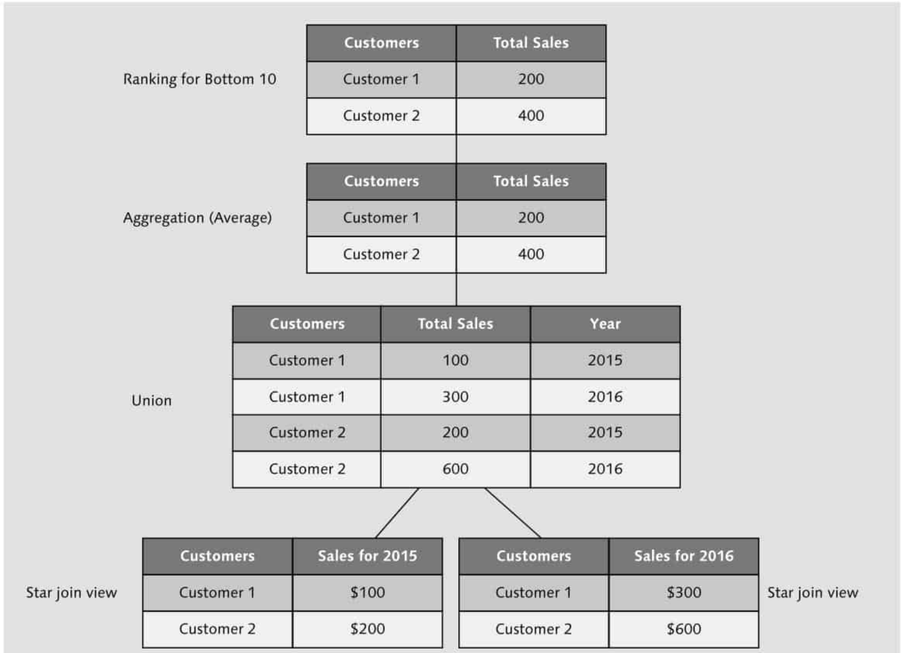


Figure 4.22 Full Solution: X = Customer 1, Y = 400, Z = 400

19. Correct answer: **B**

    To see everything on the right table, use a right outer join.

    Note that these tables do not have referential integrity. If we had asked the question a little differently, this could have been important.

Takeaway

You should now have a general understanding of information modeling concepts: views, joins, cubes, star joins, data foundations, dimension tables, and so on. In this chapter, you learned about the differences between attributes and measures and the value of CDS in the development cycle. You have seen the various types of views that SAP HANA uses, how to use them together, and what options you have available for your information modeling. Finally, we examined a few best practices and guidelines for modeling in SAP HANA in general.

Summary

You have learned about many modeling concepts and how they are applied and implemented in SAP HANA.

You're now ready to go into the practical details of learning how to use all the SAP HANA information views and how to enhance these views with the various modeling artifacts that we have available to us in each of these different views.

Contents

Acknowledgments 15
Preface 17

1 SAP HANA Certification Track—Overview 23

Who This Book Is For 24
SAP HANA Certifications 25
Associate-Level Certification 27
Professional-Level Certification 28
Specialist-Level Certification 28
SAP HANA Application Associate Certification Exam 29
Exam Objective 31
Exam Structure 31
Exam Process 33
Summary 34

2 SAP HANA Training 35

SAP Education Training Courses 36
Training Courses for SAP HANA Certifications 37
Additional SAP HANA Training Courses 38
Other Sources of Information 39
SAP Help 39
hana.sap.com and SAP Community Network 41
SAP HANA Academy 41
openSAP and openHPI 44
Hands-On with SAP HANA 46
Where to Get an SAP HANA System 46
Project Examples 52
Where to Get Data 54
Exam Questions 56
Types of Questions 56
Elimination Technique 58
Bookmark Questions 59
General Examination Strategies 59
Summary 60

3 Architecture and Deployment Scenarios 61

Objectives of This Portion of the Test 62
Key Concepts Refresher 63
In-Memory Technology 63
Architecture and Approach 69
Deployment Scenarios 79
Important Terminology 89
Practice Questions 91
Practice Question Answers and Explanations 96
Takeaway 99
Summary 99

4 Information Modeling Concepts 101

Objectives of This Portion of the Test 102
Key Concepts Refresher 103
Tables 103
Views 103
Cardinality 106
Joins 107
Core Data Services Views 117
Cube 119
Information Views 122
Using Information Views 124
Other Modeling Artifacts 127
Semantics 132
Hierarchies 132
Best Practices and Modeling Guidelines 134
Important Terminology 135
Practice Questions 136
Practice Question Answers and Explanations 144
Takeaway 147
Summary 148

5 Information Modeling Tools 149

Objectives of This Portion of the Test 150
Key Concepts Refresher 151
SAP HANA Studio 151

SAP HANA Web-Based Development Workbench 170
Important Terminology 174
Practice Questions 175
Practice Question Answers and Explanations 178
Takeaway 179
Summary 180

6 Information Views 181

Objectives of This Portion of the Test 182
Key Concepts Refresher 183
Data Sources for Information Views 184
Calculation Views: Type Dimension, Type Cube, and Type Cube with Star Join 185
Working with Nodes 197
Semantics Node 202
Attribute Views and Calculation Views of Type Dimension 205
Analytic Views and Calculation Views of Type Cube with Star Join 208
Migrating Attribute and Analytic Views 209
Important Terminology 213
Practice Questions 214
Practice Question Answers and Explanations 218
Takeaway 220
Summary 221

7 Advanced Information Modeling 223

Objectives of this Portion of the Test 224
Key Concepts Refresher 225
Calculated Columns 225
Restricted Columns 231
Filters 236
Variables 238
Input Parameters 242
Currency 246
Decision Tables 250
Hierarchies 253
Important Terminology 258
Practice Questions 260
Practice Question Answers and Explanations 264

Takeaway 267
Summary 268

8 SQL and SQLScript 269

Objectives of This Portion of the Test 271
Key Concepts Refresher 271
SQL 272
SQLScript 278
Views, Functions, and Procedures 293
Catching Up with SAP HANA SPS 11 295
Important Terminology 299
Practice Questions 300
Practice Question Answers and Explanations 306
Takeaway 309
Summary 309

9 Text, Spatial, and Predictive Modeling 311

Objectives of This Portion of the Test 313
Key Concepts Refresher 313
Text 314
Spatial 323
Predictive 327
Important Terminology 332
Practice Questions 333
Practice Question Answers and Explanations 337
Takeaway 338
Summary 339

10 Optimization of Information Models 341

Objectives of This Portion of the Test 342
Key Concepts Refresher 343
Architecture and Performance 343
Redesigned and Optimized Applications 344
Information Modeling Techniques 345
Optimization Tools 346
Best Practices for Optimization 358

Important Terminology 359
Practice Questions 360
Practice Question Answers and Explanations 363
Takeaway 365
Summary 366

11 Administration of Information Models 367

Objectives of This Portion of the Test 368
Key Concepts Refresher 369
Validating and Activating Information Models 369
Transporting Information Models 374
Core Data Services 383
Refactoring Information Models 383
Documenting Information Models 387
Translating Information Models 389
Important Terminology 390
Practice Questions 392
Practice Question Answers and Explanations 397
Takeaway 400
Summary 401

12 SAP HANA Live 403

Objectives of This Portion of the Test 404
Key Concepts Refresher 405
Background Information 405
Architecture 409
Virtual Data Model 410
SAP HANA Live Views 412
Installation and Administration 413
SAP HANA Live Browser—Browse and Use Views 415
SAP HANA Live Extension Assistant—Modify Views 422
Important Terminology 424
Practice Questions 425
Practice Question Answers and Explanations 428
Takeaway 430
Summary 430

13 Security 433

Objectives of This Portion of the Test 434
Key Concepts Refresher 435
Usage and Concepts 435
Users 439
Roles 443
Privileges 446
Testing Security 458
Important Terminology 459
Practice Questions 461
Practice Question Answers and Explanations 463
Takeaway 465
Summary 465

14 Data Provisioning 467

Objectives of This Portion of the Test 468
Key Concepts Refresher 469
Concepts 470
SAP Data Services 476
SAP LT Replication Server 478
SAP Replication Server 484
SAP Direct Extractor Connection 485
SAP HANA Smart Data Access 488
SAP HANA Enterprise Information Management 495
SAP HANA Smart Data Streaming 497
Flat Files or Microsoft Excel Datasheets 500
Web Services (OData and REST) 502
Important Terminology 502
Practice Questions 505
Practice Question Answers and Explanations 508
Takeaway 510
Summary 511

15 Utilization of Information Models 513

Objectives of this Portion of the Test 514
Key Concepts Refresher 515
Business Intelligence Concepts 515

Business Intelligence Tools for Microsoft Office Integration 522
Business Intelligence Tools for Applications and Dashboards 527
Business Intelligence Tools for Data Discovery 529
Business Intelligence Tools for Reporting 532
Choosing the Right Business Intelligence Tool 533
Alternative Consumption Methods for SAP HANA 534
Important Terminology 538
Practice Questions 538
Practice Question Answers and Explanations 541
Takeaway 542
Summary 542

The Author 543
Index 545

Index

A

ABAP, 477
ABC analysis algorithm, 329
Accelerator deployment, 84
  profitability analysis, 84
ACID-compliant database, 75
Administration, 367, 368
Administration Console, 152, 163, 343, 352, 360, 365
  diagnosis files, 163
  performance, 163
Affinity propagation, 328
Aggregation node, 197, 213, 216, 219
  calculated columns, 226
  restricted columns, 232
Aggregations, 129, 136, 344
Algorithms, 327
Amazon Web Services (AWS), 47, 49, 88, 94
American National Standards Institute (ANSI), 271
Analytic privileges, 449, 460, 465
  assign to role, 450
  create, 450
  migration, 455, 463
Analytic views, 123, 136, 137, 144, 182, 208, 215
  calculate before aggregation, 229
  create, 186
  migration, 209
  temporal joins, 220
  vs. calculation views of type cube with star join, 208
Analytics, 159, 517, 538
API, 318, 333
  SAP HANA simple info access (SINA), 318
  SAP HANA Text Analysis XS JavaScript, 322
  SAP HANA Text Mining XS JavaScript, 322
Application Function Library (AFL), 278, 329
Application Function Modeler (AFM), 179, 330, 331, 332, 336
  editor, 168
  PAL, 168
Applications, 344
  development, 167
  privileges, 457
Apriori algorithm, 328
Architects, 25
Architecture, 69, 90, 343, 359
Association algorithms, 328
Asynchronous replication, 79
Attribute views, 123, 136, 144, 182, 205, 217
  create, 186
  derived, 207
  migration, 185, 209
  vs. calculation view of type dimension, 205
Attributes, 101, 120, 121, 135
  calculated columns, 228
  geographic, 254
  restricted columns, 232
  restrictions, 453
  vocabulary, 252
Authentications, 441
Authorization Assistant, 436
Authorizations, 359, 441, 442
Auto Documentation, 387, 392, 397
Automated predictive library, 531
Average, 129

B

Backups, 76, 154
Best practices, 134
BEx Analyzer, 522
Big data, 493
  limitations, 493
BIGINT, 307
Bookmark questions, 59
Bottlenecks, 69
Branch logic, 359
Bring-your-own-license, 89
Business Explorer (BEx), 519
Business Function Library (BFL), 329
Business information, 513

Business intelligence, 513, 514
applications and dashboards, 527
choosing the right tool, 533
consuming information models, 515
convergence of tools, 517
data discovery, 529
fact sheet, 523
history, 516
reporting, 532
tools, 519
Business Intelligence Consumer Services (BICS), 476, 504, 526, 540
Business rules, 250
BW362, 29, 38

C

C_HANAIMP, 27
C_HANAIMP_11, 17, 24
scoring, 33
C_HANATEC, 27, 29
C4.5 decision tree, 331
Caching, 127
Calculated columns, 102, 223, 225, 226, 258, 260, 276, 278
analytic views, 263
counters, 230
create, 226
input parameters, 244
Calculation engine, 346, 365
Calculation view of type cube, 124, 136, 144, 183, 214, 216
aggregation node, 219
create, 188
Calculation view of type cube with star join, 123, 136, 144, 183, 208, 210, 214, 220
advantages, 209
create, 188
star join node, 200, 219
vs. analytic views, 208
Calculation view of type dimension, 123, 136, 144, 183, 189, 205, 214, 217
create, 187
migration, 210
projection node, 219
Calculation views, 126, 185, 263, 346, 536
adding data sources, 192
adding nodes, 191
create, 186
creating joins, 193
default nodes, 190
output area, 246
save and activate, 196
selecting output fields, 194
variables, 241
Calculations, 134, 345
CALL statement, 290
Cardinality, 106, 193, 356, 358, 361
many-to-many, 107
many-to-one, 107
one-to-many, 106
one-to-one, 106
types, 106
Catalog, 178
SAP HANA studio, 155, 156, 176
SAP HANA web-based development workbench, 171
table context menu, 156
table preview, 172
tables, 156
CDS views, 117, 135, 139
ABAP, 118
benefits, 117
data sources, 118
Certification
prefixes, 26
track, 23
Change and Transport System (CTS), 376
Characteristics, 120
Circularstring, 324
Classical analytic privileges, 204
Classification algorithms, 328
Classroom training, 36
Clients, 168
Client-side JavaScript, 86
Cloud, 170
deployments, 87
Clustering algorithms, 327, 329
Cold data, 79
Column, 103
engine, 346

Column (Cont.)
store, 156
table, 78, 157, 272
views, 156
Column Views folder, 373
Columnar storage, 344, 361
Column-based
databases, 73
input parameters, 243, 259
storage, 73
table, 68, 156, 176
Column-oriented tables, 103
Comments, 387
Complex event processing, 517, 538
Composite Provider, 537
Compression, 62
Computation Continuous Language (CCL), 499
Conditional statements, 278
Consumers, 520
Consumption, 534
Contains predicate, 317
Content, 159
create modeling object, 162
packages, 159
SAP HANA studio, 155, 159
SHINE package, 161
subpackages, 159
Context menu, 156
Controllers, 535
Convergence, 517
Core Data Services (CDS), 101, 117, 145, 184, 213, 315, 383, 393
Count, 129
Counters, 230, 258
new, 231
CPU
in parallel, 71
speed, 64
Create Fulltext Index, 314
Creators, 520
CSS, 86, 151
CSV files, 500
Cubes, 101, 119, 135, 137, 144
Currency, 246, 248
conversion, 247, 248
Currency (Cont.)
conversion options, 249
conversion schema, 249
decimal shift, 248
decimal shift back, 249
source and target, 247
Currency conversions, 102, 223, 259, 261, 278, 299
Customer query views, 411, 424

D

Dashboards, 534
Data
aging, 72, 79
archiving, 494
backups, 76
category, 187, 188
compression, 343, 361
discovery, 519, 529
federation, 488
filter, 276
foundation, 122, 126, 135, 145, 251, 345
integration specialist, 25
lakes, 493
noise, 328
persistence, 77
provisioning specialist, 25
read, 276, 361
records, 272
scientists, 25
sources, 184, 213
streaming, 497
types, 273
volume, 76, 77, 78, 79, 99
Data Control Language (DCL), 273, 306
Data Definition Language (DDL), 273, 284, 306
Data foundation node, 215
Data Manipulation Language (DML), 273, 306
Data modeling
artifacts, 127, 136
concepts, 101
limit and filter, 134
Data preview, 157
analysis, 159

Data preview (Cont.)
distinct values, 159
sort current dataset, 158
sort entire dataset, 158
sorting and exporting, 158
Data provisioning, 467, 502
concepts, 470
replication vs extractors, 486
tools, 469
Data Query Language (DQL), 273, 301, 306
Database
administrators, 25
column-oriented, 68
connections, 475
deployment, 82, 83
development, 167
migration, 82, 90
to platform, 69
views, 104
Datahub, 54
Datasets, 54
DataStore Object (DSO), 475
Decimal shift, 262
Decision
tables, 102, 184, 213
trees, 328
Decision tables, 223, 250, 261, 263
create, 251
nodes, 252
output fields, 252
Declarative logic, 281
Dedicated hardware, 79
Default schema, 284, 289
Default view node, 190
Definer, 284
Delimited identifiers, 274, 306
Delivery units, 367, 375, 376, 391, 397, 462
Delta, 503
buffer, 73, 74, 278, 299, 344
load, 473, 484, 503
merge, 73, 74, 344
store, 74
Deployment, 61
accelerator, 84
cloud, 87
database, 82, 83
Deployment (Cont.)
development platform, 84, 85
scenarios, 61, 79, 93
sidecar solution, 80
virtual machine, 87
Deprecate, 204, 216, 386
Derived attribute views, 207
Derived From Procedure/Scalar Function input parameter, 243, 259
Derived From Table input parameter, 243, 259
Design-time
definitions, 373
objects, 368
roles, 445
Design-time objects, 372, 390, 393
CDS, 383, 398
delivery units, 397
Developer Mode, 377, 391, 399
Developer perspective, 153, 164
Repositories tab, 153
Systems tab, 153
Developers, 25
Development
object, 167
system, 250, 375, 390
Development platform deployment, 84
programming languages, 85
SAP HANA XS, 85
Diagnosis Files tab, 163, 353
Dictionary compression, 68
Dimension tables, 119, 120, 125, 135
Dimension views, 123, 125, 136, 183, 185, 200, 205, 217, 361
Direct input parameter, 243, 258
Disaster recovery, 79
Distinct values, 159
Distributed database, 75
Documentation, 387
Domain fix values, 238, 258, 260
Dynamic
data tiering, 79
join, 110, 116, 135, 194, 345
restrictions, 454
SQL, 282, 295

E

E_HANAAW ABAP certification, 29
E_HANABW SAP BW on SAP HANA certification, 29
E_HANAINS installation certification, 29
Eclipse, 151, 153
Editor, 173
SAP HANA web-based development workbench, 173
E-learning, 36
Elimination technique, 58
Engines, 346, 358, 362
Entity analysis, 320
Exam
objects, 31
process, 33
questions, 56
structure, 31
EXEC statement, 282
Execution plan, 350, 351
Explain Plan, 343, 347, 360, 362, 365
Export, 376, 379, 395
delivery units, 368
information models, 368
Expression Editor, 228
calculated columns, 227
elements, 228
functions area, 228
operators area, 228
restricted columns, 233
Expressions, 223, 273
Extract, Transform, and Load (ETL), 471, 502
Extraction, 471

F

Facets, 120
Fact tables, 101, 119, 121, 122, 126, 135
Fault-tolerant text search, 314, 316
Fields
hide, 216
original, 215
output, 194
rename, 215
Filters, 169, 236, 258, 345, 358, 462
expressions, 223, 237, 263
Filters (Cont.)
operations, 223
operators, 236
variables, 239
Flat files, 500
use cases, 501
Flowgraph, 164, 167
For loop, 281
Free access, 46
Freestyle search, 317
Full outer join, 108, 110, 194
Full-text index, 314, 316, 320, 332, 334
columns, 315
hidden column, 316
Functions, 156, 274
Fuzzy text search, 207, 210, 278, 286, 299, 311, 313, 315, 316, 319, 332, 335
alternative names, 317

G

Generate SLT File, 419
GeoJSON, 326
Geospatial processing, 323
Global Temporary, 275
Global Temporary Column, 275
Google, 313
Grammatical Role Analysis (GRA), 322
Granted Roles tab, 442, 445
Graphical calculation views, 173, 174, 186, 385, 422
decision tables, 250
refactoring and restructuring, 384
Graphical data models, 70, 97
Graphical flowgraph model, 331
Graphical information models, 358
Group by, 197, 216
Grouping sets, 359
Grubbs' test algorithm, 329
Guidelines, 134

H

HA100, 37
HA215, 38
HA300, 38
HA400, 38

HA450, 38, 536
Hardware, 51
Hierarchies, 101, 132, 133, 223, 253
  BICS, 254
  create, 254
  MDX, 254
  time-dependent, 207, 256
  value help, 257
High availability, 75
  shared disk storage, 75
History, 386
  column, 275
Hot data, 79, 495
HTML5, 85, 86, 151, 170
Hybrid cloud, 88, 91

I

Identifiers, 273, 274
if(), 228, 278
Imperative logic, 281, 302, 359
Import, 376, 379
  delivery units, 368
  information models, 368
  objects, 382, 383, 391
Index server, 77
Indexes, 156
InfoCubes, 536
InfoPackage, 475, 508
InfoProvider, 475
Information modelers, 25
Information models, 536
  activate, 368, 369, 372, 394
  administration, 367, 368
  Auto Documentation, 387
  build, 125
  comments, 387
  deprecated, 386, 395
  documentation, 367, 369, 387
  labels, 389
  mass copy, 382
  refactoring, 367, 369, 383, 392
  SAP BW, 536
  SAP HANA XS, 535, 542
  schema mapping, 380
  techniques, 345
  translation, 367, 369, 389
Information models (Cont.)
  transport, 368, 374
  utilization, 126, 513
  validate, 21, 367, 368, 369
Information views, 122, 181
  characteristics, 293
  data sources, 184, 217
  parameterized, 272
  performance, 362
  use, 124
Infrastructure as a Service (IaaS), 87
Initial load, 473, 480, 503
In-memory, 67, 72
  data movement, 70
  technology, 61, 63, 89, 361
In-Memory DataStore Object (IMDSO), 488
Inner join, 108, 109, 111, 135, 137, 144, 194
Input parameters, 102, 223, 242, 258
  create, 243, 244
  date, 246
  expressions, 245
  types, 243, 258
Insert-only, 74
  principle, 344
International Organization for Standardization (ISO), 271
Internet of Things (IoT), 498
Interval, 239, 258
Invoker, 284
  security, 307

J

Java, 151
Java Database Connectivity (JDBC), 476, 503
Java Virtual Machines, 151, 170
JavaScript, 85, 151
  server-side, 86
Join node, 253, 260
  calculated columns, 226
Joins, 102, 104, 107, 110, 135, 181, 193, 294, 356
  basic, 108
  dynamic join, 116
  engine, 346
  node, 191, 200, 213
  performance, 361

Joins (Cont.)
  referential join, 111
  relocation, 490
  self-joins, 110
  spatial join, 115
  star joins, 122
  temporal join, 113
  text join, 112
  types, 108

K

Kerberos, 462, 464
Key
  field, 104
  figures, 119
K-means, 327

L

Language, 284, 289
  detection, 314
Layered information views, 449
Lazy load, 78, 99
Left outer join, 108, 109, 113, 135, 144, 194, 220, 361
Level hierarchies, 132, 134, 254, 262
Line of business (LOB), 520
Linestring, 324
Link prediction algorithm, 329
Load, 471
Local Temporary, 275
Local Temporary Column, 275
Log
  backups, 77, 78
  buffer, 78
  volume, 77, 79, 99
Log-based replication, 484
Logical
  data warehouse, 494
  joins, 122
Loops, 302, 307, 359

M

Main table, 107
Managed Cloud as a Service (MCaaS), 88
Many-to-many cardinality, 107
Many-to-one cardinality, 107
Mapping property, 194, 215, 219
Mass
  copy, 382, 394, 398
  import, 382
Master data, 122, 135, 145, 214
Materialized views, 105
Maximum, 129
Measures, 101, 119, 121, 135, 138, 145
  calculate before aggregation, 229
  calculated columns, 228
  restricted columns, 232
Memory
  blocks, 76
  dump, 78
Microsoft Azure Cloud, 47, 49
Microsoft Excel, 540
  datasheets, 500
  on SAP HANA, 524
Microsoft Live Office, 522
Microsoft Office integration, 519, 534
Microsoft PowerPoint, 522
Migration, 211
Miscellaneous algorithms, 329
Mobile, 170
Modeler perspective, 153, 176
Modeling role, 444, 449, 460
Models, 535
Model-View-Controller (MVC), 85, 535
Monitoring role, 444, 460
Moore's Law, 63
Multicore CPUs, 343, 361
Multidimensional expressions (MDX), 476, 524
Multilevel aggregation, 292
Multilinestring, 324
Multiple choice questions, 56
Multiple response questions, 56
Multipoint, 324
Multipolygon, 324
Multitenancy, 86
Multitenant database container (MDC), 86, 184, 213

N

Native objects, 382, 392
Nodes, 181
NoSQL databases, 75
NULL, 274, 301, 306

O

Object Linking and Embedding, Database (OLE DB), 503
Object Linking and Embedding, Database for Online Analytical Processing, 476
Object privileges, 448, 460
OData, 38, 85, 502, 509
  service, 319
ODBO driver, 524
OLAP, 92, 343, 517, 538
  engine, 346, 347, 413
  vs. OLTP, 67
OLTP, 92, 343
  vs. OLAP, 67
One-to-many cardinality, 106
One-to-one cardinality, 106
Online analytical processing, 406
Online transactional processing, 405
Open Content, 418
Open Cross Reference, 418
Open Database Connectivity (ODBC), 475, 503
Open Definition, 418
Open Geospatial Consortium (OGC), 325
Open View in Analysis Office, 420
Open View in SAP Lumira, 420
openHPI, 44
openSAP, 44, 536
  certifications, 45
Operating system, 51
Operational reporting, 424, 426
Operators, 233, 273, 348, 352
Optimization, 341
  best practices, 358
  tools, 346
Oracle, 272
Outliers, 329
Output fields, 105, 194

P

P_HANAIMP, 28
Package privileges, 456
Page manager, 76, 77, 99
Pages, 76
Parallelism, 71, 344
Parameters, 290
Parent-child hierarchies, 132, 134, 254, 256, 260, 534
Partitioned tables, 75
Partitioning, 75
Performance, 343
  analysis, 492
  enhancements, 341
  tab, 163, 352
Performance Analysis Mode, 179, 343, 354, 355, 360, 363, 365
Persistence layer, 61, 74, 75, 90
Perspectives, 152
  change, 153
  list, 153
  modeler vs. developer, 153
PHP, 151
PL/SQL, 272
Plan visualization, 153
Platform as a Service (PaaS), 87
Point, 324
Polygon, 324
PostGIS, 325
Predicates, 273, 325
Predictive, 327
  analytics, 517, 538
  modeling, 311, 327, 330
Predictive Analysis Library (PAL), 168, 327, 331, 333, 336
  algorithms, 329
Preprocessing algorithms, 328
Prescriptive analytics, 517, 538
Primary storage, 76
Private cloud, 88, 91
Private views, 410, 424
Privileges, 437, 460
  analytic privileges, 449
  application privileges, 457
  object privileges, 448
  package privileges, 456

Privileges (Cont.)
  system privileges, 447
  types, 446
Procedures, 288, 293, 300, 303
  characteristics, 294
  create, 288
  parameters, 290
  read-only, 292
Production system, 250, 375, 390
Professionally authored, 520, 534
Project Explorer, 153, 166
Projection node, 198, 213, 260
  calculated columns, 226
  filters, 237
Projections, 127, 136, 276
Projects, 52, 53
Proof-of-concept (POC), 500
Provisioning, 155
Proxy tables, 488
Pruning configuration table, 201
Public cloud, 88, 91, 98

Q

Quality assurance system, 250, 375, 390
Queries, 274
Query views, 411, 412, 424
Quick View tab, 152, 154, 455

R

R language, 85, 307, 531
Range, 239, 258
Rank node, 198, 213, 216
  columns, 199
  sorting, 198
Ranking, 128, 136
Raw
  data, 157
  tab, 176
RDL_USER, 459
Reads SQL data, 289
Real-time
  computing, 343, 361
  data, 127
  reporting, 105
Recursive tables, 110
Refactoring, 367, 383, 384, 392, 395, 396
  deprecate, 386
  history, 386
  nodes in a graphical calculation view, 384
  Where-Used, 386
Referential integrity, 111, 144
Referential join, 110, 111, 112, 114, 135, 137, 144, 194, 220, 345, 356
  star joins, 122
Regression algorithms, 328
Release notes, 39
Replication, 79, 82, 473
  log-based, 484
  trigger-based, 479
REPO.EXPORT, 448
REPO.IMPORT, 448
REPO.MAINTAIN_DELIVERY_UNITS, 448
REPO.WORK_IN_FOREIGN_WORKSPACE, 448
Report writers, 25
Reporting, 515, 517, 519, 538
  system, 406
Reports, 516
Repository
  create, 165
  icon, 165
  objects, 372, 390, 445
  tab, 165
  workspace, 165
Repository Translation Tool (RTT), 389
REST, 38, 85, 502
Restricted columns, 223, 231, 232, 258
  create, 233
  operators, 233
  using calculated columns, 235
Restrictions, 452, 453
Restructuring, 384
Returns, 284
Reuse views, 410, 412, 424
Right outer join, 108, 110, 138, 144, 194
Right-click menu, 156
Roles, 433, 437, 443, 459
  choosing, 442
  composite role, 437, 459
  create, 444
  new, 444

Roles (Cont.)
template roles, 443, 444
Root package privilege, 457
Row, 103
engine, 346
storage, 344
store, 156
Row-based
databases, 73
storage, 73
tables, 68, 156, 176
Row-oriented tables, 103
Ruby, 151
Runtime information, 157
Runtime objects, 368, 397
CDS, 383
Runtime roles, 445
Runtime version, 373, 390

S

Sales forecasting, 327
Sandbox, 500
SAP Basis, 27
SAP Business Suite, 479
SAP Business Warehouse (BW), 82
SAP BusinessObjects Analysis, edition for Microsoft Office, 519, 522, 525, 539, 541
information, 526
plug-in, 526
SAP BusinessObjects BI Mobile, 529
SAP BusinessObjects Design Studio, 519, 521, 527, 529, 539, 540
dashboard, 528
information, 527
SAP BusinessObjects Web Intelligence, 519, 532
information, 532
SAP BW, 121, 156, 435, 475, 486, 517, 536
consume calculation views, 537
consume information models, 537, 541
generate SAP HANA views, 537
SAP BW on SAP HANA, 38, 51, 82, 156, 464
SAP Cloud Appliance Library, 49, 50
SAP Cloud for Analytics, 522, 529, 540
SAP Community Network (SCN), 41
SAP CRM, 435
SAP Crystal Reports, 519, 532
SAP Crystal Reports 2011/2013, 533, 539
SAP Crystal Reports for Enterprise, 533, 539, 540
information, 533
SAP Data Quality Management, 478
SAP Data Services, 476, 505
benefits, 478
extracting data, 477
transforming and cleaning data, 478
SAP Direct Extractor Connection (DXC), 477, 485, 487
SAP Education, 35
SAP ERP, 408, 435
SAP ERP on SAP HANA, 156
SAP extractors, 474, 503
SAP Fiori, 86
SAP HANA
as a database, 82, 83
as a development platform, 84, 85
as a sidecar solution, 80, 81
as a virtual machine, 87
as an accelerator, 84
clients, 475
in the cloud, 87
reference guides, 40
training courses, 35, 37
SAP HANA Academy, 41
SAP HANA application function modeler (AFM), 168
SAP HANA Application Lifecycle Manager, 329, 378, 391
SAP HANA as a database, 409
security, 435
SAP HANA as a platform, 90
security, 436, 461
SAP HANA as a sidecar, 409
security, 436
SAP HANA Business Function Library (BFL) Reference, 41
SAP HANA certifications
associate level, 27
professional level, 28
specialist level, 28
SAP HANA Cloud Platform, 47, 48, 87, 89, 522
SAP HANA Developer Center, 46

SAP HANA Developer Guide, 41
SAP HANA Developer Quick Start Guide, 41
SAP HANA Enterprise Cloud (HEC), 88, 89, 91, 94
private cloud, 89
SAP HANA Enterprise Information Management, 495
SAP HANA Interactive Education (SHINE), 39, 530
SAP HANA Live, 364, 403, 404, 461, 464
architecture, 410
background information, 405
definition, 405
installation, 413, 425
rapid deployment solution, 415
security, 436
SQL engine, 346
tags, 419
views, 403, 412, 422, 424
SAP HANA Live Browser, 403, 414, 415, 425
all views, 416
for business users, 421, 430
invalid views, 416
my favorites, 416
toolbar, 418
SAP HANA Live Extension Assistant, 403, 414, 422, 425, 426
restrictions, 423
SAP HANA Modeling Guide, 39
SAP HANA Predictive Analysis Library (PAL) Reference, 41
SAP HANA Security Guide, 40
SAP HANA simple info access (SINA) API, 318, 333
SAP HANA smart data access (SDA), 79, 356, 488
adapters, 488
benefits, 495
data archiving, 494
implementation, 490
virtual tables, 488
SAP HANA smart data integration (SDI), 168, 495, 496
SAP HANA smart data quality (SDQ), 495, 496
SAP HANA smart data streaming (SDS), 168, 497, 498, 517
benefits, 499
SAP HANA SPS 10, 322
SAP HANA SPS 11, 283, 295, 319, 455
SAP HANA SQLScript Reference, 40
SAP HANA studio, 149, 150, 151, 175, 445, 514
Administration Console, 163
Backup, 154
Catalog, 155
Content, 155, 159
create calculation view, 187
data preview, 534
development object, 167
installation, 52
main workspace, 154
perspectives, 152
Provisioning, 155
Quick View tab, 152
screen, 152
security, 156
session client, 152, 168
SQL Console, 162
systems view, 154
working areas, 152
XS project, 166
SAP HANA Text Analysis XS JavaScript API, 322
SAP HANA Troubleshooting and Performance Analysis Guide, 40
SAP HANA web-based development workbench, 52, 150, 170, 175, 414, 445
editor, 171
performance analysis mode, 174
table data, 172
SAP HANA XS, 85, 86, 179, 291, 317, 535, 541
application privileges, 457, 460
project, 166
SAP HANA XS DB Utilities JavaScript API Reference, 41
SAP HANA XS JavaScript API Reference, 41
SAP HANA XS JavaScript Reference, 40
SAP HANA XSUnit JavaScript API Reference, 41
SAP Help, 39

SAP Hybris, 88
SAP InfiniteInsight, 530
SAP landscape, 375
SAP Learning Hub, 37
SAP LT Replication Server (SLT), 409, 478
  benefits, 483
  configuration, 482
  features, 481
  replication process, 480
  trigger-based replication, 480
SAP Lumira, 54, 421, 519, 521, 529, 531, 540
  charts, 530
  information, 529
  storyboard, 531
SAP NetWeaver
  old architecture, 65
SAP Predictive Analytics, 529, 530, 540
SAP Replication Server (SRS), 484
  benefits, 485
  log-based replication, 484
SAP resources, 39
SAP S/4HANA, 72, 80, 86, 97, 464
SAP Solution Manager, 376
SAP Store, 53
SAP SuccessFactors, 88
SAP Support Mode, 377, 391
SAPUI5, 25, 38, 41, 85, 86
  Developer Guide for SAP HANA, 41
  views, 535
Savepoint, 76, 78, 99
Scalable Vector Graphic (SVG), 326
Scalar functions, 269, 283, 285
Scale-out architecture, 75
Schema, 156, 162, 394, 396
  mapping, 367, 368, 380, 393
  tables, 156
SCORE() function, 317
Scripted calculation views
  migrate, 297
Search, 314
Seasonal patterns, 327
Secondary storage, 76
Security, 282, 359, 433, 438
  concepts, 435, 437
  SAP HANA studio, 156
Security (Cont.)
  SAP HANA web-based development workbench, 171
  testing, 458
  usage, 438
Segmentation, 328
SELECT *, 301, 306
SELECT statement, 276
Self-joins, 110, 135
Self-service, 520, 523, 534
Semantics node, 202, 208, 214, 215
  Column tab, 202
  hide fields, 203
  hierarchies, 254
  input parameters, 242
  renaming fields, 202
  session client, 204
  top node, 189
  variables, 239
  View Properties tab, 203
Sentiment analysis, 320, 334
Separate statements, 279
Sequential execution, 289
Servers, 75
Session client, 152, 168, 440
  settings, 169
Set-oriented, 272, 300
Shared hardware, 79
SHINE, 53, 118, 160, 161, 287, 326, 347
  datasets, 54
Show Line Numbers, 162
Sidecar deployment, 80
  advantages, 81
  blank system, 82
SINA, 318, 333
Single sign-on, 441, 462
Single value, 239, 258
Slow disk, 67
Social network analysis algorithms, 329
Software, 51
Software as a Service (SaaS), 88
Spatial
  data, 323
  data types, 278, 300
  functions, 326
  import data, 325
  join, 115, 135, 194, 209, 220, 323, 324

Spatial (Cont.)
  join type, 278
  processing, 311, 323, 333
  properties, 324
SQL, 85, 269, 272, 341
  analytic privileges, 204, 452, 455, 462, 463, 465
  button, 162
  conditional statements, 278
  creating calculated columns, 276
  creating projections, 276
  creating tables, 275
  creating unions, 277
  Data Definition Language, 273
  Data Manipulation Language, 273
  dynamic, 282, 302, 307
  engine, 204, 346, 413, 422, 425
  Expression Editor, 228
  filter data, 276
  guidelines, 359
  language, 272
  reading data, 276
  security, 284, 289
  set-oriented, 272
  statements, 162
  Structured Query Language, 271, 299
  views, 105, 184, 213
SQL Console, 152, 162, 290, 347, 353
  schemas, 162
  text editors, 162
SQL Editor, 454
SQL Plan Cache, 365
SQLScript, 85, 269, 272, 278, 292, 307, 331, 341, 359, 362, 384
  compiler, 279
  decision tables, 253
  declarative logic, 281
  dynamic SQL, 282
  for loop, 281
  multilevel aggregation, 292
  optimizer, 279
  procedures, 288
  security, 282, 283
  separate statements, 279
  while loop, 281
Standard
  deviation, 129
Standard (Cont.)
  union, 200
Star join node, 209, 213, 215, 217, 219
  calculated columns, 226
  data foundation, 199
  restricted columns, 232
Star join views, 123, 126, 136
  analytic views, 123
  calculation view of type cube with star join, 123
Star joins, 122
  referential joins, 122
Statements, 274
Static List input parameter, 243, 259
Statistics algorithms, 329
Stored procedures, 70, 97, 288
Structured Query Language, 271, 299
Subtype, 188
Sum, 129
Supervised learning, 328, 336
Synchronous, 99
  replication, 79
_SYS_REPO, 284, 440, 454, 459, 461
SYSTEM, 439, 458, 459
System administrator, 369
System identity (SID), 156
System privileges, 447
  types, 447
Systems tab, 438
Systems view, 154, 162, 177
  folders, 154

T

Tables, 103
  context menu, 176
  create, 275
  data preview, 157
  data source, 184, 213
  definitions, 156
  export data, 378
  functions, 251, 269, 283, 286
  join, 345
  left and right, 107
  link, 104
  partitioned, 75
  recursive, 110

Tables (Cont.)
select, 104
table definitions, 156
types, 251, 275
Technical performance tuning experts, 25
Template roles, 443
modeling role, 444, 460
monitoring role, 444, 460
Temporal join, 110, 113, 114, 135, 137, 144, 220
Tenant, 86
Text, 311, 314
analysis, 333
editors, 162
index, 314
mining, 333
Text analysis, 311, 314
results, 321
usage, 320
Text join, 110, 112, 113, 135, 138, 144, 145, 194, 369
left outer join, 113
Text mining, 311, 313, 314, 315, 322
capabilities, 322, 333
Threshold value, 356
Time dimensions, 188
Time series, 328
Time-based calculation views, 189, 215
Time-dependent hierarchies, 207, 256
value help, 257
Timeline view, 351
Trace configuration, 354
Traces, 171
SAP Web-based Development Workbench, 171
Training courses, 26, 37, 38
BW362, 29, 38
HA100, 27, 32, 37
HA200, 27
HA215, 38
HA300, 27, 32, 38
HA360, 32, 38
HA400, 29, 38
HA450, 38
HA900, 32, 38
SAPX05, 38
Transaction Control Language (TCL), 273
Transaction manager, 77, 99
Transactional data, 135
Transact-SQL (T-SQL), 272
Transform, 471
Transient Provider, 537, 539
Translate, 367, 389
labels, 389
Repository Translation Tool (RTT), 389
Transport, 374, 378, 393, 395
delivery unit, 378
export and import, 376
information models, 368
TREAT expression, 324
Trigger, 479
Trusted data, 521, 534

U

UDF, 283, 284, 288, 300, 302
characteristics, 294
Undelimited identifier, 274
Unicode, 274, 481
Union node, 200, 214
pruning configuration table, 201
Union with constant values, 200
Unions, 104, 126, 129, 130, 136, 154, 345, 361
all, 130
creating, 277
pruning, 362
with constant values, 130
Universe, 534
User, 433, 437, 439
_SYS_REPO user, 440
create, 440
new, 440
SYSTEM, 439

V

Validation rules, 357, 369, 370, 390, 393
errors, 371, 394
examples, 369
Value help, 233
hierarchies, 257
views, 411, 424

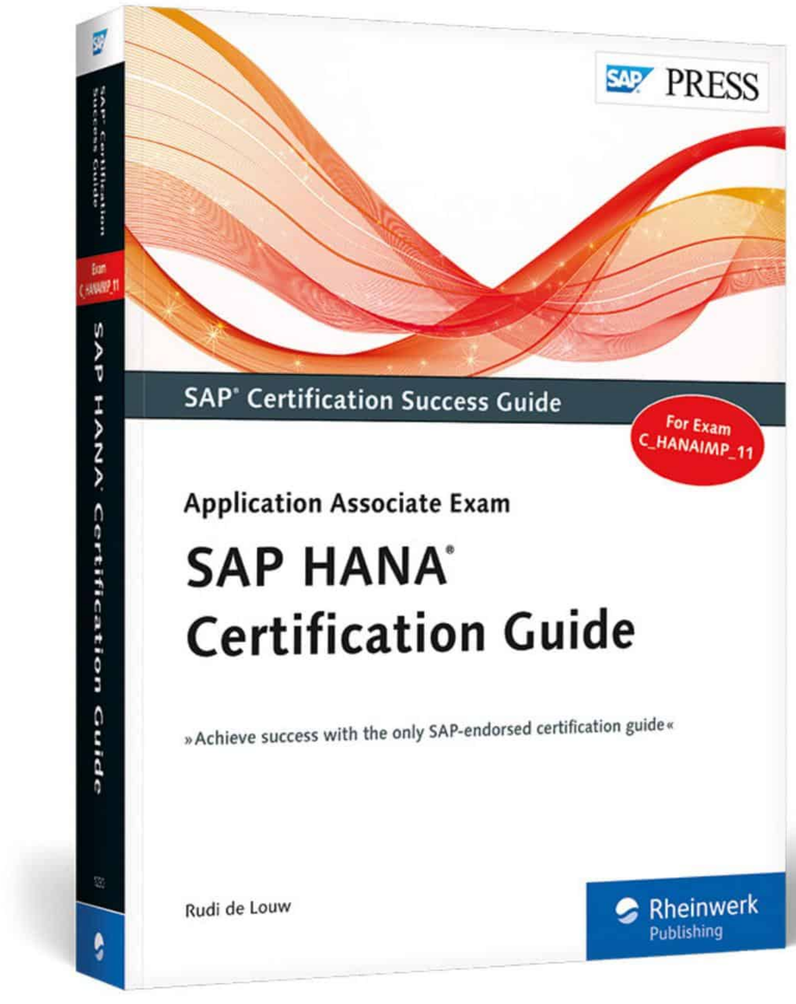
Variables, 223, 238, 258, 261, 262, 279
create, 239, 240
modeling, 239
types, 239, 258
Variance, 129
VDA, 405
View Editor, 355
Views, 101, 102, 103, 135, 136, 293, 449
disappear, 105
save, 105
Virtual classrooms, 36
Virtual data model (VDM), 403, 410
Virtual machines, 94
Virtual tables, 156, 184, 213, 251, 488, 489, 495
add, 491
using, 492
vs native tables, 489
Visualize Plan, 343, 348, 360, 363, 365
Visualize View in PlanViz Editor, 355
VMware, 52
VMware vSphere, 87, 94, 98

W

Warm data, 79, 495
Web
applications, 534
services, 85, 502
Web-based development tool, 175
Weighted score tables, 329
Well-Known Binary (WKB), 325
Well-Known Text (WKT), 325
Where-Used, 385, 395
While loop, 281
With results view, 289
Workspace root, 165

X

XML-based analytic privileges, 452



Rudi de Louw is an SAP HANA architect for SAP South Africa, as well as an international speaker and trainer, principal consultant, and mentor. He has been working with SAP HANA for more than 5 years, and is the SAP HANA market unit champion. He is also the content owner for the SAP HANA professional certification examination. Rudi has provided software and technology solutions to businesses for more than 25 years, and has been with SAP since 1999. He has a passion for knowledge sharing and understanding, new technologies, and finding innovative ways to leverage these to help people in their development.

Rudi de Louw

SAP HANA Certification Guide

559 Pages, 2016, $79.95
ISBN 978-1-4932-1230-9

www.sap-press.com/3859