



Spatial

SELF-GUIDED DEMONSTRATION

Version 1.2

September 7, 2017

Demo purpose

This script walks you through how to demonstrate the Spatial feature in SAP HANA. It is intended to give you a quick overview of how spatial data works and outlines some of the available SQL functions that can be used with this feature.

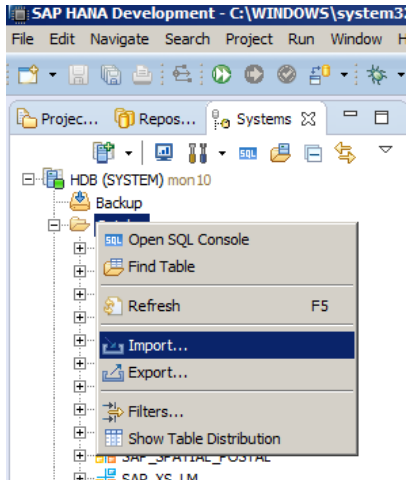
Using the script

The demo is presented in a four-column format. This format is designed to accommodate your time restrictions and interests. The script is meant to be followed from beginning to end.

Step	What to do	What you should see	Notes
Step number	Step by step instructions	Screen-shots of what you'll see	Details about what you are doing/seeing

Preparation

This script was written for use on SAP HANA 1 SPS10 or higher. If you have not done so already, you need to [download the Spatial demo files](#).



Open HANA studio logging in as the System user or a user that has rights to create schemas, import data, and create tables.

There is a HANA binary import (in the spatial_demo_data.zip file) that needs to be added to your HANA instance by right clicking on your Catalog folder and choosing Import. Browse to where you've extracted the aforementioned .zip file and then add the single "census" table while in the import dialogue.

Note that this table is not a complete US Census table. If you do wish to have a full US Census database for demos etc. please see the [HANA Academy video here](#) for more information. This census data video is part of the [Live 4 / ERP Agility](#) project. This project uses HANA Spatial in views and .xsjs web services and also features a mapping application.

After importing the CENSUS table, copy the syntax from the spatial_schema_prep.sql file that was in the download folder. In your HANA Studio, open a new SQL Console, paste in this syntax, and run the syntax in its entirety.

Step-by-step live demo script

It is imperative that you have already completed all of the steps in the previous Preparation section.

	What to do	What you should see	Notes																												
1	<p>Launch your HANA Studio and connect to your instance as the System user or as the user that ran the steps in the Preparation section.</p> <p>Copy the syntax from the geo_spatial_workflow.sql file and paste this into a new SQL Console in HANA Studio.</p> <p>Do not run this syntax yet as you will be running different parts of the file throughout this demo.</p>	<pre>1 ----- 2 -- 3 -- 4 -- Geo-Spatial Overview 5 -- 6 -- 7 ----- 8 9 10 ----- 11 -- 12 -- constructing geo-spatial data from decimal type data 13 -- 14 ----- 15 16 SELECT TOP 10 "ROWID", 17 NAME, 18 COORDINATES_LON AS LON, 19 COORDINATES_LAT AS LAT, 20 NEW ST_POINT(COORDINATES_LON, COORDINATES_LAT) AS LON_LAT_POINT 21 FROM GEO_SPATIAL.CENSUS 22 ORDER BY "ROWID" ASC; 23 24 25 ----- 26 -- 27 -- create table with 3 different geo-spatial systems 28 -- 29 ----- 30 31 DROP TABLE GEO_SPATIAL.CENSUS_GEO; 32 33 CREATE COLUMN TABLE GEO_SPATIAL.CENSUS_GEO 34 LIKE GEO_SPATIAL.CENSUS WITH NO DATA;</pre>	<p>What is SAP HANA Spatial?</p> <p>Column-oriented data structures and in-memory computing have developed into powerful components of today's enterprise applications. While the focus of these developments has primarily been on analyzing sales data, the potential for using these technologies to analyze geographic information is significant. Support for the processing of spatial data represents a key evolution in SAP HANA.</p> <p>To deliver vastly improved performance and results in everything from modeling and storage to analysis and presentation of your spatial data, SAP HANA includes a multilayered spatial engine and supports spatial columns, spatial access methods, and spatial reference systems. With these enhanced GIS features, SAP HANA now provides a common database for both your business and spatial data.</p> <p>Spatial data is data that describes the position, shape, and orientation of objects in a defined space. Spatial data is represented as 2D geometries in the form of points, line strings, and polygons. Two common operations performed on spatial data are calculating the distance between geometries, and determining the union or intersection of multiple objects.</p>																												
2	<p>Select and run lines 16 to 22.</p>	<div><pre>-- SELECT TOP 10 "ROWID", -- NAME, -- COORDINATES_LON AS LON, -- COORDINATES_LAT AS LAT, -- NEW ST_POINT(COORDINATES_LON, COORDINATES_LAT) AS L -- FROM GEO_SPATIAL.CENSUS -- ORDER BY "ROWID" ASC</pre><table><tr><th></th><th>ROWID</th><th>NAME</th><th>LON</th></tr><tr><td>1</td><td>1,965</td><td>Census Tract 9427, Apache County, Arizona</td><td>-109.3680582</td></tr><tr><td>2</td><td>1,966</td><td>Census Tract 9449, Apache County, Arizona</td><td>-109.6021168</td></tr><tr><td>3</td><td>1,972</td><td>Census Tract 5, Coconino County, Arizona</td><td>-111.5565226</td></tr><tr><td>4</td><td>1,973</td><td>Census Tract 9, Coconino County, Arizona</td><td>-111.6274758</td></tr><tr><td>5</td><td>1,974</td><td>Census Tract 14, Coconino County, Arizona</td><td>-111.7381682</td></tr><tr><td>6</td><td>1,975</td><td>Census Tract 9411, Coconino County, Ari...</td><td>-110.8683315</td></tr></table></div>		ROWID	NAME	LON	1	1,965	Census Tract 9427, Apache County, Arizona	-109.3680582	2	1,966	Census Tract 9449, Apache County, Arizona	-109.6021168	3	1,972	Census Tract 5, Coconino County, Arizona	-111.5565226	4	1,973	Census Tract 9, Coconino County, Arizona	-111.6274758	5	1,974	Census Tract 14, Coconino County, Arizona	-111.7381682	6	1,975	Census Tract 9411, Coconino County, Ari...	-110.8683315	<p>This syntax shows how to construct a spatial point type in SAP HANA SQL.</p> <p>The LON and LAT columns in this particular case are stored as Decimal types in this CENSUS database. To get started using Geo-Spatial in HANA you can simply:</p> <div><div>a) Create a table that contains a Spatial Type column or</div><div>b) You can use extended SQL functions against existing data, like in this example.</div></div> <p>The raw output of the point we've constructed (LON_LAT_POINT) is in binary format. Later on we'll look at methods to make the output more readable.</p>
	ROWID	NAME	LON																												
1	1,965	Census Tract 9427, Apache County, Arizona	-109.3680582																												
2	1,966	Census Tract 9449, Apache County, Arizona	-109.6021168																												
3	1,972	Census Tract 5, Coconino County, Arizona	-111.5565226																												
4	1,973	Census Tract 9, Coconino County, Arizona	-111.6274758																												
5	1,974	Census Tract 14, Coconino County, Arizona	-111.7381682																												
6	1,975	Census Tract 9411, Coconino County, Ari...	-110.8683315																												

	What to do	What you should see	Notes																																																																	
3	Select and run lines 31 to 46.	<pre>----- -- create table with 3 different geo-spatial systems ----- DROP TABLE GEO_SPATIAL.CENSUS_GEO; CREATE COLUMN TABLE GEO_SPATIAL.CENSUS_GEO LIKE GEO_SPATIAL.CENSUS WITH NO DATA; ALTER TABLE GEO_SPATIAL.CENSUS_GEO ADD (LONLAT_POINT_0 ST_GEOMETRY(0)); ALTER TABLE GEO_SPATIAL.CENSUS_GEO ADD (LONLAT_POINT_1000004326 ST_GEOMETRY(1000004326)); ALTER TABLE GEO_SPATIAL.CENSUS_GEO ADD (LONLAT_POINT_4326 ST_GEOMETRY(4326));</pre>	<p>We're creating a copy of the CENSUS table, without data, and then adding 3 different Spatial Type (ST_GEOMETRY) columns to the new table.</p> <p>The purpose of this and the next couple of steps is to show how to create a new or alter an existing table to include a Spatial Type column. While this is not 100% necessary, as you can construct geo-spatial data on the fly like in Step 2 above, this will make the creation of SQL queries, views, and web services a lot easier and potentially lead to better performance. We'll look at the performance of the different Spatial Reference Systems later on.</p> <p>Three main spatial systems for storing data in HANA are:</p> <ol style="list-style-type: none">1) The default which is a Cartesian system (SRID 0),2) WGS 84 Planar (SRID 1000004326) which is a flat projected system (e.g. a map), and3) WGS 84 (SRID 4326) which is the spheroidal system used for GPS. <p>Each system has different advantages and disadvantages which we'll also look at later on. Please see the Spatial Reference Guide for more information on these different systems.</p>																																																																	
4	Select and run lines 50 to 58.	<pre>SELECT * FROM GEO_SPATIAL.CENSUS_GEO</pre> <table><thead><tr><th></th><th>ROWID</th><th>TYPE</th><th>DOMAIN_ID</th><th>NAME</th></tr></thead><tbody><tr><td>1</td><td>3,981</td><td>mu.acs_topline</td><td>06073006200</td><td>Census Tract 62, San Diego Coun</td></tr><tr><td>2</td><td>96,990</td><td>mu.acs_topline</td><td>32001940200</td><td>Census Tract 9402, Churchill Cour</td></tr><tr><td>3</td><td>178,...</td><td>mu.acs_topline</td><td>04019004900</td><td>Census Tract 49, Pima County, Ar</td></tr><tr><td>4</td><td>179,...</td><td>mu.acs_topline</td><td>06037106606</td><td>Census Tract 1066.06, Los Angele</td></tr><tr><td>5</td><td>97,071</td><td>mu.acs_topline</td><td>32029940100</td><td>Census Tract 9401, Storey Count</td></tr><tr><td>6</td><td>120,...</td><td>mu.acs_topline</td><td>06029005504</td><td>Census Tract 55.04, Kern County</td></tr><tr><td>7</td><td>155,...</td><td>mu.acs_topline</td><td>35006946000</td><td>Census Tract 9460, Cibola County</td></tr><tr><td>8</td><td>304,...</td><td>mu.acs_topline</td><td>49051940300</td><td>Census Tract 9403, Wasatch Coui</td></tr><tr><td>9</td><td>238,...</td><td>mu.acs_topline</td><td>06085504700</td><td>Census Tract 5047, Santa Clara C</td></tr><tr><td>10</td><td>178,...</td><td>mu.acs_topline</td><td>04021000203</td><td>Census Tract 2.03, Pinal County,</td></tr><tr><td>11</td><td>9,154</td><td>mu.acs_topline</td><td>08001008701</td><td>Census Tract 87.01, Adams Coun</td></tr><tr><td>12</td><td>96,617</td><td>mu.acs_topline</td><td>35043940800</td><td>Census Tract 9408, Sandoval Cou</td></tr></tbody></table>		ROWID	TYPE	DOMAIN_ID	NAME	1	3,981	mu.acs_topline	06073006200	Census Tract 62, San Diego Coun	2	96,990	mu.acs_topline	32001940200	Census Tract 9402, Churchill Cour	3	178,...	mu.acs_topline	04019004900	Census Tract 49, Pima County, Ar	4	179,...	mu.acs_topline	06037106606	Census Tract 1066.06, Los Angele	5	97,071	mu.acs_topline	32029940100	Census Tract 9401, Storey Count	6	120,...	mu.acs_topline	06029005504	Census Tract 55.04, Kern County	7	155,...	mu.acs_topline	35006946000	Census Tract 9460, Cibola County	8	304,...	mu.acs_topline	49051940300	Census Tract 9403, Wasatch Coui	9	238,...	mu.acs_topline	06085504700	Census Tract 5047, Santa Clara C	10	178,...	mu.acs_topline	04021000203	Census Tract 2.03, Pinal County,	11	9,154	mu.acs_topline	08001008701	Census Tract 87.01, Adams Coun	12	96,617	mu.acs_topline	35043940800	Census Tract 9408, Sandoval Cou	<p>Using existing longitude and latitude columns (stored as Decimal or Float type data) to create a new ST_GEOMETRY or ST_POINT column is relatively easy as it simply requires constructing a NEW ST_Point() like we used earlier in Step 2.</p> <p>This means that any table brought into HANA that contains longitude and latitude columns can be easily altered to contain a spatial column.</p> <p>Note that the order of the two dimensions in the constructor is longitude and then latitude which corresponds to an X and then a Y dimension. Some other databases and some mapping tools use the order of latitude and then longitude.</p>
	ROWID	TYPE	DOMAIN_ID	NAME																																																																
1	3,981	mu.acs_topline	06073006200	Census Tract 62, San Diego Coun																																																																
2	96,990	mu.acs_topline	32001940200	Census Tract 9402, Churchill Cour																																																																
3	178,...	mu.acs_topline	04019004900	Census Tract 49, Pima County, Ar																																																																
4	179,...	mu.acs_topline	06037106606	Census Tract 1066.06, Los Angele																																																																
5	97,071	mu.acs_topline	32029940100	Census Tract 9401, Storey Count																																																																
6	120,...	mu.acs_topline	06029005504	Census Tract 55.04, Kern County																																																																
7	155,...	mu.acs_topline	35006946000	Census Tract 9460, Cibola County																																																																
8	304,...	mu.acs_topline	49051940300	Census Tract 9403, Wasatch Coui																																																																
9	238,...	mu.acs_topline	06085504700	Census Tract 5047, Santa Clara C																																																																
10	178,...	mu.acs_topline	04021000203	Census Tract 2.03, Pinal County,																																																																
11	9,154	mu.acs_topline	08001008701	Census Tract 87.01, Adams Coun																																																																
12	96,617	mu.acs_topline	35043940800	Census Tract 9408, Sandoval Cou																																																																

	What to do	What you should see	Notes																																												
5	Select and run lines 69 to 76.	<pre>SELECT TOP 10 "ROWID", COORDINATES_LON, COORDINATES_LAT, LONLAT_POINT_0, LONLAT_POINT_1000004326, LONLAT_POINT_4326 FROM GEO_SPATIAL.CENSUS_GEO ORDER BY "ROWID" ASC</pre> <table><thead><tr><th></th><th>ROWID</th><th>COORDINATES_LON</th><th>COORDINATES_LAT</th><th>LONLAT_POINT</th></tr></thead><tbody><tr><td></td><td>1,965</td><td>-109.3680582</td><td>36.7552563</td><td>0101000000000000</td></tr><tr><td></td><td>1,966</td><td>-109.6021168</td><td>35.5989197</td><td>0101000000000000</td></tr><tr><td></td><td>1,972</td><td>-111.5565226</td><td>35.2275283</td><td>0101000000000000</td></tr><tr><td></td><td>1,973</td><td>-111.6274758</td><td>35.1555465</td><td>0101000000000000</td></tr><tr><td></td><td>1,974</td><td>-111.7381682</td><td>35.5528939</td><td>0101000000000000</td></tr><tr><td></td><td>1,975</td><td>-110.8683315</td><td>35.9334895</td><td>0101000000000000</td></tr></tbody></table>		ROWID	COORDINATES_LON	COORDINATES_LAT	LONLAT_POINT		1,965	-109.3680582	36.7552563	0101000000000000		1,966	-109.6021168	35.5989197	0101000000000000		1,972	-111.5565226	35.2275283	0101000000000000		1,973	-111.6274758	35.1555465	0101000000000000		1,974	-111.7381682	35.5528939	0101000000000000		1,975	-110.8683315	35.9334895	0101000000000000	<p>The 3 new ST_Geometry columns in our new table are returned by default in a binary format. The binary output of the 3 columns is identical as, even though these points are stored in different Spatial Reference Systems, the underlying point data remains unchanged.</p> <p>Earlier we created these 3 new columns as ST_Geometry Type columns. We could have also created ST_Point Type columns as we know in this case that these columns will only contain point (longitude and latitude) data.</p> <p>ST_Geometry is the super-type for all geometries and includes points, polygons, line-strings etc. We'll look at these sub-types later on.</p>									
	ROWID	COORDINATES_LON	COORDINATES_LAT	LONLAT_POINT																																											
	1,965	-109.3680582	36.7552563	0101000000000000																																											
	1,966	-109.6021168	35.5989197	0101000000000000																																											
	1,972	-111.5565226	35.2275283	0101000000000000																																											
	1,973	-111.6274758	35.1555465	0101000000000000																																											
	1,974	-111.7381682	35.5528939	0101000000000000																																											
	1,975	-110.8683315	35.9334895	0101000000000000																																											
6	Select and run lines 80 to 90.	<table><thead><tr><th>LONLAT_POINT_0.ST_ASWKT()</th><th>LONLAT_POINT_0.ST_ASEWKT()</th></tr></thead><tbody><tr><td>POINT (-109.368058 36.755257)</td><td>SRID=0;POINT (-109.368058 36.755257)</td></tr><tr><td>POINT (-109.602117 35.59892)</td><td>SRID=0;POINT (-109.602117 35.59892)</td></tr><tr><td>POINT (-111.556522 35.227529)</td><td>SRID=0;POINT (-111.556522 35.227529)</td></tr><tr><td>POINT (-111.627476 35.155546)</td><td>SRID=0;POINT (-111.627476 35.155546)</td></tr><tr><td>POINT (-111.738168 35.552894)</td><td>SRID=0;POINT (-111.738168 35.552894)</td></tr><tr><td>POINT (-110.868332 35.93349)</td><td>SRID=0;POINT (-110.868332 35.93349)</td></tr><tr><td>POINT (-111.561132 34.303856)</td><td>SRID=0;POINT (-111.561132 34.303856)</td></tr></tbody></table>	LONLAT_POINT_0.ST_ASWKT()	LONLAT_POINT_0.ST_ASEWKT()	POINT (-109.368058 36.755257)	SRID=0;POINT (-109.368058 36.755257)	POINT (-109.602117 35.59892)	SRID=0;POINT (-109.602117 35.59892)	POINT (-111.556522 35.227529)	SRID=0;POINT (-111.556522 35.227529)	POINT (-111.627476 35.155546)	SRID=0;POINT (-111.627476 35.155546)	POINT (-111.738168 35.552894)	SRID=0;POINT (-111.738168 35.552894)	POINT (-110.868332 35.93349)	SRID=0;POINT (-110.868332 35.93349)	POINT (-111.561132 34.303856)	SRID=0;POINT (-111.561132 34.303856)	<p>This query contains several methods of changing the output of our geometry columns. Two of the more readable methods are AsWKT() or 'As Well Known Text' and AsEWKT() or as 'Extended Well Known Text'.</p> <p>There is also a geo-JSON output that can be consumed in web applications as well as an SVG output.</p>																												
LONLAT_POINT_0.ST_ASWKT()	LONLAT_POINT_0.ST_ASEWKT()																																														
POINT (-109.368058 36.755257)	SRID=0;POINT (-109.368058 36.755257)																																														
POINT (-109.602117 35.59892)	SRID=0;POINT (-109.602117 35.59892)																																														
POINT (-111.556522 35.227529)	SRID=0;POINT (-111.556522 35.227529)																																														
POINT (-111.627476 35.155546)	SRID=0;POINT (-111.627476 35.155546)																																														
POINT (-111.738168 35.552894)	SRID=0;POINT (-111.738168 35.552894)																																														
POINT (-110.868332 35.93349)	SRID=0;POINT (-110.868332 35.93349)																																														
POINT (-111.561132 34.303856)	SRID=0;POINT (-111.561132 34.303856)																																														
7	Select and run line 99 to 100.	<pre>SELECT * FROM GEO_SPATIAL.CENSUS_BINARY</pre> <table><thead><tr><th></th><th>ROWID</th><th>NAME</th><th>LONLAT_BINARY</th></tr></thead><tbody><tr><td>1</td><td>1,965</td><td>Census Tract 9427, Apache County, Arizona</td><td></td></tr><tr><td>2</td><td>1,966</td><td>Census Tract 9449, Apache County, Arizona</td><td></td></tr><tr><td>3</td><td>1,972</td><td>Census Tract 5, Coconino County, Arizona</td><td></td></tr><tr><td>4</td><td>1,973</td><td>Census Tract 9, Coconino County, Arizona</td><td></td></tr><tr><td>5</td><td>1,974</td><td>Census Tract 14, Coconino County, Arizona</td><td></td></tr><tr><td>6</td><td>1,975</td><td>Census Tract 9411, Coconino County, Ari...</td><td></td></tr><tr><td>7</td><td>1,976</td><td>Census Tract 1, Gila County, Arizona</td><td></td></tr><tr><td>8</td><td>1,977</td><td>Census Tract 5, Gila County, Arizona</td><td></td></tr><tr><td>9</td><td>1,978</td><td>Census Tract 10, Gila County, Arizona</td><td></td></tr><tr><td>10</td><td>1,979</td><td>Census Tract 9402, Gila County, Arizona</td><td></td></tr></tbody></table>		ROWID	NAME	LONLAT_BINARY	1	1,965	Census Tract 9427, Apache County, Arizona		2	1,966	Census Tract 9449, Apache County, Arizona		3	1,972	Census Tract 5, Coconino County, Arizona		4	1,973	Census Tract 9, Coconino County, Arizona		5	1,974	Census Tract 14, Coconino County, Arizona		6	1,975	Census Tract 9411, Coconino County, Ari...		7	1,976	Census Tract 1, Gila County, Arizona		8	1,977	Census Tract 5, Gila County, Arizona		9	1,978	Census Tract 10, Gila County, Arizona		10	1,979	Census Tract 9402, Gila County, Arizona		<p>Spatial data may exist as point data stored in binary format in a table. You may wish to convert these types of columns into a Spatial Type column so that it is easier to use the converted data in SQL queries etc.</p> <p>We'll alter this table to use HANA Spatial in the next step.</p>
	ROWID	NAME	LONLAT_BINARY																																												
1	1,965	Census Tract 9427, Apache County, Arizona																																													
2	1,966	Census Tract 9449, Apache County, Arizona																																													
3	1,972	Census Tract 5, Coconino County, Arizona																																													
4	1,973	Census Tract 9, Coconino County, Arizona																																													
5	1,974	Census Tract 14, Coconino County, Arizona																																													
6	1,975	Census Tract 9411, Coconino County, Ari...																																													
7	1,976	Census Tract 1, Gila County, Arizona																																													
8	1,977	Census Tract 5, Gila County, Arizona																																													
9	1,978	Census Tract 10, Gila County, Arizona																																													
10	1,979	Census Tract 9402, Gila County, Arizona																																													
8	Select and run lines 105 to 117.	<table><thead><tr><th>LONLAT_POINT</th><th>LONLAT_POINT.ST_ASWKT()</th></tr></thead><tbody><tr><td>010100000000000000448E575BC000000040AC604240</td><td>POINT (-109.368058 36.755257)</td></tr><tr><td>0101000000000000001489665BC000000068A9CC4140</td><td>POINT (-109.602117 35.59892)</td></tr><tr><td>010100000000000000109EE35BC0000000A81F9D4140</td><td>POINT (-111.556522 35.227529)</td></tr><tr><td>0101000000000000009028E85BC0000000F0E8934140</td><td>POINT (-111.627476 35.155546)</td></tr><tr><td>010100000000000000243EEF5BC000000038C5C64140</td><td>POINT (-111.738168 35.552894)</td></tr><tr><td>010100000000000000C092B75BC0000000987CF74140</td><td>POINT (-110.868332 35.93349)</td></tr><tr><td>01010000000000000098E9E35BC0000000C0E4264140</td><td>POINT (-111.561132 34.303856)</td></tr><tr><td>010100000000000000A047CF5BC0000000B841154140</td><td>POINT (-111.238747 34.166068)</td></tr><tr><td>01010000000000000024D6AD5BC0000000D0D3CF4...</td><td>POINT (-110.716195 33.623652)</td></tr><tr><td>01010000000000000080A9955BC00000005024F04040</td><td>POINT (-110.33847 33.876108)</td></tr></tbody></table>	LONLAT_POINT	LONLAT_POINT.ST_ASWKT()	010100000000000000448E575BC000000040AC604240	POINT (-109.368058 36.755257)	0101000000000000001489665BC000000068A9CC4140	POINT (-109.602117 35.59892)	010100000000000000109EE35BC0000000A81F9D4140	POINT (-111.556522 35.227529)	0101000000000000009028E85BC0000000F0E8934140	POINT (-111.627476 35.155546)	010100000000000000243EEF5BC000000038C5C64140	POINT (-111.738168 35.552894)	010100000000000000C092B75BC0000000987CF74140	POINT (-110.868332 35.93349)	01010000000000000098E9E35BC0000000C0E4264140	POINT (-111.561132 34.303856)	010100000000000000A047CF5BC0000000B841154140	POINT (-111.238747 34.166068)	01010000000000000024D6AD5BC0000000D0D3CF4...	POINT (-110.716195 33.623652)	01010000000000000080A9955BC00000005024F04040	POINT (-110.33847 33.876108)	<p>In this step we've altered the table with a binary to include a true ST_Point type column. Note that spatial type columns can be either ST_Geometry or ST_Point.</p> <p>We then converted the existing binary column using this syntax:</p> <pre>ST_GeomFromWKB(LONLAT_BINARY)</pre> <p>This conversion method output is then inserted into the new ST_Point column. We've now converted a table that had a binary column to a table with a HANA Spatial column.</p>																						
LONLAT_POINT	LONLAT_POINT.ST_ASWKT()																																														
010100000000000000448E575BC000000040AC604240	POINT (-109.368058 36.755257)																																														
0101000000000000001489665BC000000068A9CC4140	POINT (-109.602117 35.59892)																																														
010100000000000000109EE35BC0000000A81F9D4140	POINT (-111.556522 35.227529)																																														
0101000000000000009028E85BC0000000F0E8934140	POINT (-111.627476 35.155546)																																														
010100000000000000243EEF5BC000000038C5C64140	POINT (-111.738168 35.552894)																																														
010100000000000000C092B75BC0000000987CF74140	POINT (-110.868332 35.93349)																																														
01010000000000000098E9E35BC0000000C0E4264140	POINT (-111.561132 34.303856)																																														
010100000000000000A047CF5BC0000000B841154140	POINT (-111.238747 34.166068)																																														
01010000000000000024D6AD5BC0000000D0D3CF4...	POINT (-110.716195 33.623652)																																														
01010000000000000080A9955BC00000005024F04040	POINT (-110.33847 33.876108)																																														

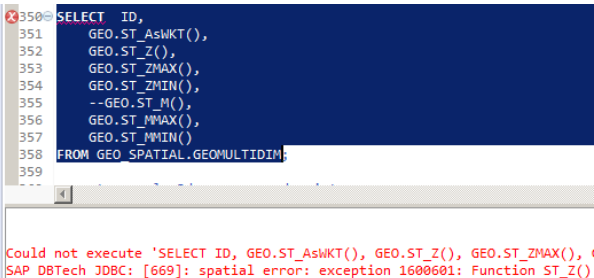
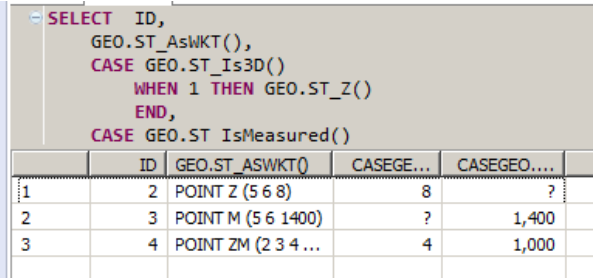
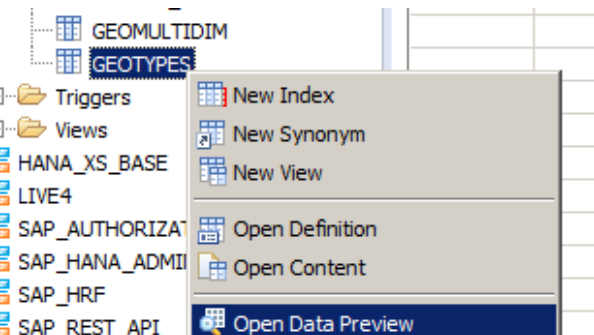
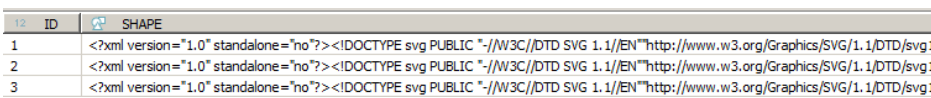
	What to do	What you should see	Notes										
9	Select and run lines 128 to 144.	<pre>WITH A AS (SELECT -- Palo Alto office in 3 different SRS NEW ST_Point('POINT(-122.1463709 37.3989354)') AS PA, NEW ST_Point('POINT(-122.1463709 37.3989354)', 1000004326) AS PA_1000004326, NEW ST_Point('POINT(-122.1463709 37.3989354)', 4326) AS PA_4326, -- Vancouver office in 3 different SRS NEW ST_Point('POINT(-123.1208974 49.2766576)') AS VN, NEW ST_Point('POINT(-123.1208974 49.2766576)', 1000004326) AS VN_1000004326, NEW ST_Point('POINT(-123.1208974 49.2766576)', 4326) AS VN_4326 FROM DUMMY) SELECT PA.ST_ASRKT(), VN.ST_ASRKT(), VN.ST_DISTANCE(PA, 'kilometer') AS DIST_0, VN_1000004326.ST_DISTANCE(PA_1000004326, 'kilometer') AS DIST_1000004326, VN_4326.ST_DISTANCE(PA_4326, 'kilometer') AS DIST_4326 FROM A</pre> <table><tr><th>PA.ST_ASRKT()</th><th>VN.ST_ASRKT()</th><th>DIST_0</th><th>DIST_1000004326</th><th>DIST_4326</th></tr><tr><td>POINT (-122.146371 37.398935)</td><td>POINT (-123.120897 49.276658)</td><td>0.011917633960108231</td><td>1.324.2874266880424</td><td>1.321.9430160618783</td></tr></table>	PA.ST_ASRKT()	VN.ST_ASRKT()	DIST_0	DIST_1000004326	DIST_4326	POINT (-122.146371 37.398935)	POINT (-123.120897 49.276658)	0.011917633960108231	1.324.2874266880424	1.321.9430160618783	<p>In the top SELECT statement we're returning some dummy records for the locations of the SAP Vancouver and Palo Alto offices. There is one point for each of the 3 systems: Cartesian, WGS 84 Planar, and WGS 84 Spheroid.</p> <p>The bottom SELECT statement performs 3 distance calculations, using the 3 different spatial systems, between the two offices.</p> <p>Which of these is the most accurate given that the output could be in kilometers?</p> <p>The real distance is about 1321+ kilometers so the most accurate is SRS 4326...remember that this is the system used for GPS.</p> <p>WGS Planar 1000004326 is close in this case but distances are distorted in this system due to the projection of a spherical system onto a flat system. These calculations are more distorted as one uses points that are close to either of Earth's two poles.</p> <p>The Cartesian calculation is far off as it is using a measurement derived from the Pythagoras Theorem. It is not therefore using a geo-spatial calculation but instead is using a spatial distance calculation.</p> <p>The conclusion to this particular exercise is to utilize SRS 4326 (WGS 84 Spheroid) when accurate distance calculations are required.</p> <p>NOTE: The differences in the calculated distances are not unique to SAP HANA as they are typical of the spatial systems used.</p>
PA.ST_ASRKT()	VN.ST_ASRKT()	DIST_0	DIST_1000004326	DIST_4326									
POINT (-122.146371 37.398935)	POINT (-123.120897 49.276658)	0.011917633960108231	1.324.2874266880424	1.321.9430160618783									

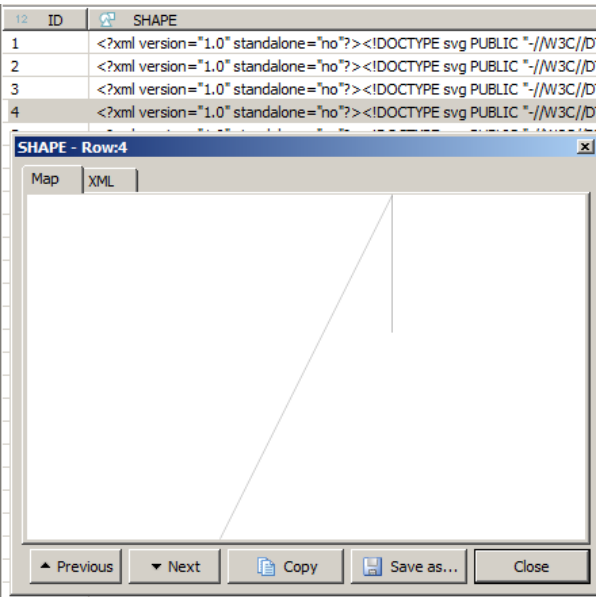
	What to do	What you should see	Notes						
10	Select and run lines 156 to 209.	<pre>SELECT SUM(POPULATION_TOTAL) AS POP, COUNT(*) AS POINTS FROM CENSUS_GEO WHERE LENGTH(CENSUS_GEO_ID) = 11 AND LONLAT_POINT_0.ST_Within(NEW ST_Polygon('Polygon((-138.9343338012695 40.47433077320648, -138.9343338012695 32.89135906381192, -100.5920486450195 32.89135906381192, -100.5920486450195 40.47433077320648, -138.9343338012695 40.47433077320648))',0)) = 1</pre> <table><thead><tr><th></th><th>POP</th><th>POINTS</th></tr></thead><tbody><tr><td>1</td><td>49,306,668</td><td>9,656</td></tr></tbody></table> <pre>Statement 'SELECT SUM(POPULATION_TOTAL) AS POP, COUNT(*) AS POINTS successfully executed in 204 ms 915 µs (server processing time: 50 ms 0 µs) Fetched 1 row(s) in 0 ms 11 µs (server processing time: 0 ms 0 µs) Statement 'SELECT SUM(POPULATION_TOTAL) AS POP, COUNT(*) AS POINTS successfully executed in 193 ms 142 µs (server processing time: 40 ms 0 µs) Fetched 1 row(s) in 0 ms 13 µs (server processing time: 0 ms 0 µs) Statement 'SELECT SUM(POPULATION_TOTAL) AS POP, COUNT(*) AS POINTS successfully executed in 1.664 seconds (server processing time: 1 ms 0 µs) Fetched 1 row(s) in 0 ms 12 µs (server processing time: 0 ms 0 µs) Statement 'SELECT SUM(POPULATION_TOTAL) AS POP, COUNT(*) AS POINTS successfully executed in 2.474 seconds (server processing time: 2 ms 0 µs) Duration of 4 statements: 4.537 seconds Fetched 1 row(s) in 0 ms 11 µs (server processing time: 0 ms 0 µs)</pre>		POP	POINTS	1	49,306,668	9,656	<p>In this set of queries we'll look at the performance of each of the 3 systems we've been using by calculating the number of geographical points within a pre-determined polygon.</p> <p>We're using 3 different methods for the calculation: ST_Within, ST_Contains, & ST_CoveredBy. ST_Within is equivalent to ST_Contains in that all geometries must be completely within the interior & not intersect the boundary. Neither of these two can be used in round earth (SRID 4326) calculations. There are other methods that cannot be used with the 4326 system. Please see the Spatial Reference Guide for more information on the individual methods.</p> <p>Syntax overview...</p> <ul style="list-style-type: none">- first two queries use ST_Within function to return a population within a polygon- first one uses data in a Cartesian system & the ST_Within function- second uses WGS Planar & the ST_Within function- third uses WGS Planar & ST_Contains- fourth uses WGS 84 Spheroid / 4326 & ST_CoveredBy <p>Results...</p> <ul style="list-style-type: none">- First 3 provide accurate results- ST_Within provides the best speed- SRS 4326 is by far the slowest and results are not what we expect <p>Given the performance results from this simple test it would seem to make sense to use SRS 1000004326 / WGS 84 Planar or SRS 0 / Cartesian for storing geo-spatial data in HANA. However, we've seen with SRS 0 that the distance calculations are not accurate for LON LAT points.</p> <p>NOTE: The differences and results of the calculations are not unique to SAP HANA as they are typical of the spatial systems used.</p> <p>In the next steps we'll see how to effectively deal with having best performance, the greatest accuracy for distance calculations, as well as the greatest accuracy for other methods such as ST_Within.</p>
	POP	POINTS							
1	49,306,668	9,656							

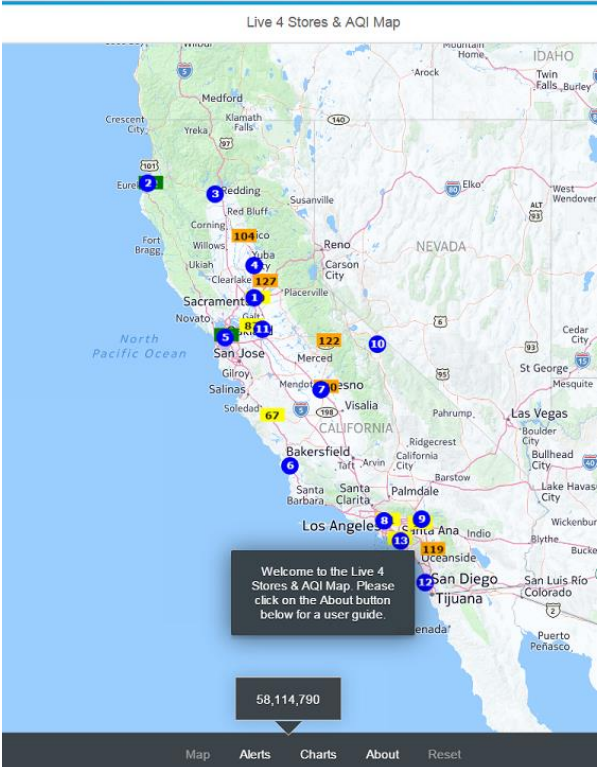
	What to do	What you should see	Notes																																																							
11	Select and run lines 220 to 229.	<div><pre>SELECT TOP 10 "ROWID", LONLAT_POINT_1000004326.ST_SRID() AS SRID, LONLAT_POINT_1000004326.ST_Transform(4326).ST_SRID() AS TRANS_SRID, NEW ST_Point('POINT(-123.1208974 49.2766576)', 4326) AS DIST_TO_VAN FROM CENSUS_GEO ORDER BY "ROWID" ASC;</pre></div> <table><thead><tr><th></th><th>ROWID</th><th>SRID</th><th>TRANS_SRID</th></tr></thead><tbody><tr><td>1</td><td>1,965</td><td>1,000,004,326</td><td>4,326</td></tr><tr><td>2</td><td>1,966</td><td>1,000,004,326</td><td>4,326</td></tr><tr><td>3</td><td>1,972</td><td>1,000,004,326</td><td>4,326</td></tr><tr><td>4</td><td>1,973</td><td>1,000,004,326</td><td>4,326</td></tr><tr><td>5</td><td>1,974</td><td>1,000,004,326</td><td>4,326</td></tr><tr><td>6</td><td>1,975</td><td>1,000,004,326</td><td>4,326</td></tr><tr><td>7</td><td>1,976</td><td>1,000,004,326</td><td>4,326</td></tr><tr><td>8</td><td>1,977</td><td>1,000,004,326</td><td>4,326</td></tr><tr><td>9</td><td>1,978</td><td>1,000,004,326</td><td>4,326</td></tr><tr><td>10</td><td>1,979</td><td>1,000,004,326</td><td>4,326</td></tr></tbody></table>		ROWID	SRID	TRANS_SRID	1	1,965	1,000,004,326	4,326	2	1,966	1,000,004,326	4,326	3	1,972	1,000,004,326	4,326	4	1,973	1,000,004,326	4,326	5	1,974	1,000,004,326	4,326	6	1,975	1,000,004,326	4,326	7	1,976	1,000,004,326	4,326	8	1,977	1,000,004,326	4,326	9	1,978	1,000,004,326	4,326	10	1,979	1,000,004,326	4,326	<p>Prior to SPS10 of SAP HANA there was no method to convert data between two different Spatial Reference Systems (SRS). In theory this would mean that data would have to be stored in several different columns, each with a different SRS. The CENSUS_GEO table that we created had such a structure. However, this meant taking up extra database storage for what is essentially the same underlying data.</p> <p>New to SPS10 of HANA is the ST_Transform method that allows data to be converted between different SRS and thus eliminate the need to have multiple columns of the same geometry with different SRS.</p> <p>In this syntax we've converted data that is stored in WGS 84 Planar (flat earth) / 1000004326 to WGS 84 Spheroid / 4326. This is done on the fly / in the result set only and does not alter the data stored in HANA.</p>											
	ROWID	SRID	TRANS_SRID																																																							
1	1,965	1,000,004,326	4,326																																																							
2	1,966	1,000,004,326	4,326																																																							
3	1,972	1,000,004,326	4,326																																																							
4	1,973	1,000,004,326	4,326																																																							
5	1,974	1,000,004,326	4,326																																																							
6	1,975	1,000,004,326	4,326																																																							
7	1,976	1,000,004,326	4,326																																																							
8	1,977	1,000,004,326	4,326																																																							
9	1,978	1,000,004,326	4,326																																																							
10	1,979	1,000,004,326	4,326																																																							
12	Uncomment lines 224 to 226 only. Select and rerun lines 220 to 229.	<div><pre>SELECT TOP 10 "ROWID", LONLAT_POINT_1000004326.ST_SRID() AS SRID, LONLAT_POINT_1000004326.ST_Transform(4326).ST_SRID() AS TRANS_SRID, LONLAT_POINT_1000004326.ST_Transform(4326).ST_DISTANCE(NEW ST_Point('POINT(-123.1208974 49.2766576)', 4326), 'kilometer') AS DIST_TO_VAN FROM CENSUS_GEO ORDER BY "ROWID" ASC;</pre></div> <table><thead><tr><th></th><th>ROWID</th><th>SRID</th><th>TRANS_SRID</th><th>DIST_TO_VAN</th></tr></thead><tbody><tr><td>1</td><td>1,965</td><td>1,000,004,326</td><td>4,326</td><td>1,780.7232493663578</td></tr><tr><td>2</td><td>1,966</td><td>1,000,004,326</td><td>4,326</td><td>1,876.8015077760917</td></tr><tr><td>3</td><td>1,972</td><td>1,000,004,326</td><td>4,326</td><td>1,824.4841150347802</td></tr><tr><td>4</td><td>1,973</td><td>1,000,004,326</td><td>4,326</td><td>1,828.5760404431765</td></tr><tr><td>5</td><td>1,974</td><td>1,000,004,326</td><td>4,326</td><td>1,784.8536787011453</td></tr><tr><td>6</td><td>1,975</td><td>1,000,004,326</td><td>4,326</td><td>1,786.0441934751934</td></tr><tr><td>7</td><td>1,976</td><td>1,000,004,326</td><td>4,326</td><td>1,915.620004794432</td></tr><tr><td>8</td><td>1,977</td><td>1,000,004,326</td><td>4,326</td><td>1,942.5423859182545</td></tr><tr><td>9</td><td>1,978</td><td>1,000,004,326</td><td>4,326</td><td>2,018.0236133489232</td></tr><tr><td>10</td><td>1,979</td><td>1,000,004,326</td><td>4,326</td><td>2,009.2119050402418</td></tr></tbody></table>		ROWID	SRID	TRANS_SRID	DIST_TO_VAN	1	1,965	1,000,004,326	4,326	1,780.7232493663578	2	1,966	1,000,004,326	4,326	1,876.8015077760917	3	1,972	1,000,004,326	4,326	1,824.4841150347802	4	1,973	1,000,004,326	4,326	1,828.5760404431765	5	1,974	1,000,004,326	4,326	1,784.8536787011453	6	1,975	1,000,004,326	4,326	1,786.0441934751934	7	1,976	1,000,004,326	4,326	1,915.620004794432	8	1,977	1,000,004,326	4,326	1,942.5423859182545	9	1,978	1,000,004,326	4,326	2,018.0236133489232	10	1,979	1,000,004,326	4,326	2,009.2119050402418	<p>In lines 222 to 224 we're transforming data that is stored in WGS 84 Planar so that it is in WGS 84 Spheroid. We then use this transformed data in a distance calculation with the SAP Vancouver office. The Vancouver office data is already constructed as WGS 84 Spheroid / 4326.</p> <p>We can see how we can store data as WGS Planar and then use the ST_Transform method so that accurate distances can be calculated when the points are transformed in HANA SQL to WGS Spheroid.</p> <p>By storing the data in WGS Planar we are then utilizing the SRS that has the best performance as well as having access to the most spatial methods / geo-spatial SQL functions.</p>
	ROWID	SRID	TRANS_SRID	DIST_TO_VAN																																																						
1	1,965	1,000,004,326	4,326	1,780.7232493663578																																																						
2	1,966	1,000,004,326	4,326	1,876.8015077760917																																																						
3	1,972	1,000,004,326	4,326	1,824.4841150347802																																																						
4	1,973	1,000,004,326	4,326	1,828.5760404431765																																																						
5	1,974	1,000,004,326	4,326	1,784.8536787011453																																																						
6	1,975	1,000,004,326	4,326	1,786.0441934751934																																																						
7	1,976	1,000,004,326	4,326	1,915.620004794432																																																						
8	1,977	1,000,004,326	4,326	1,942.5423859182545																																																						
9	1,978	1,000,004,326	4,326	2,018.0236133489232																																																						
10	1,979	1,000,004,326	4,326	2,009.2119050402418																																																						
13	Uncomment line 225 and rerun lines 218 to 227.	<div><pre>SELECT TOP 10 "ROWID", LONLAT_POINT_1000004326.ST_SRID() AS SRID, LONLAT_POINT_1000004326.ST_Transform(4326).ST_SRID() AS TRANS_SRID, LONLAT_POINT_1000004326.ST_Transform(4326).ST_DISTANCE(NEW ST_Point('POINT(-123.1208974 49.2766576)', 4326), 'kilometer') AS DIST_TO_VAN, LONLAT_POINT_0.ST_Transform(4326) FROM CENSUS_GEO ORDER BY "ROWID" ASC;</pre></div> <div>not execute 'SELECT TOP 10 "ROWID", LONLAT_POINT_1000004326.ST_SRID() AS SRID, .. DBTech JDBC: [669]: spatial error: exception 1610047: Invalid transform definition unction st transform()</div>	<p>In line 225 we're trying to transform a geometry stored in SRS 0 / Cartesian to SRS 4326. However an error is returned. In the next step we'll find out why.</p> <p>Could not execute 'SELECT TOP 10 "ROWID", LONLAT_POINT_1000004326.ST_SRID() AS SRID, ...' in 143 ms 412 μs . SAP DBTech JDBC: [669]: spatial error: exception 1610047: Invalid transform definition ''. at function st_transform()</p>																																																							

	What to do	What you should see	Notes																																												
14	Select and run line 231.	<pre>SELECT TRANSFORM_DEFINITION, SRS_ID FROM ST_SPATIAL_REFERENCE_SYSTEMS</pre> <table><thead><tr><th></th><th>TRANSFORM_DEFINITION</th><th>SRS_ID</th></tr></thead><tbody><tr><td>1</td><td></td><td>0</td></tr><tr><td>2</td><td>+proj=longlat +ellps=WGS84 +datum=WGS84 +no_defs</td><td>4,326</td></tr><tr><td>3</td><td>+proj=longlat +ellps=WGS84 +datum=WGS84 +no_defs</td><td>1,000,004,326</td></tr><tr><td>4</td><td></td><td>2,147,483,646</td></tr></tbody></table>		TRANSFORM_DEFINITION	SRS_ID	1		0	2	+proj=longlat +ellps=WGS84 +datum=WGS84 +no_defs	4,326	3	+proj=longlat +ellps=WGS84 +datum=WGS84 +no_defs	1,000,004,326	4		2,147,483,646	<p>There are two main rules to keep in mind when using ST_Transform.</p> <p>The first rule is that the SRS must exist in your HANA system. The query that we just ran shows that we have 4 systems including the ones we've been using: 0, 4326, & 1000004326.</p> <p>The second rule is that the original system and the target system must have the same datum. If you look at the TRANSFORM_DEFINITION you'll see that the 4326 and the planar system are both WGS84 for the datum.</p> <p>SRS 0 does not have a TRANSFORM_DEFINITION and therefore one cannot transform between 0 & 4326 or 0 & 1000004326.</p> <p>Given the inability to transform from SRS 0 to other systems and vice versa, storing geo-spatial data in SRS 1000004326 remains the best option.</p> <p>However, you may encounter existing data that is already stored as SRS 0. In these cases if you require accurate distance calculations then altering that table to include a duplicate of the geometry data but in SRS 1000004326 may be the best solution.</p>																													
	TRANSFORM_DEFINITION	SRS_ID																																													
1		0																																													
2	+proj=longlat +ellps=WGS84 +datum=WGS84 +no_defs	4,326																																													
3	+proj=longlat +ellps=WGS84 +datum=WGS84 +no_defs	1,000,004,326																																													
4		2,147,483,646																																													
15	Select and run lines 242 to 298.	<pre>SELECT *, SHAPE.ST_AsWKT() AS SHAPE FROM GEO_SPATIAL.GEOTYPES</pre> <table><thead><tr><th></th><th>ID</th><th>SHAPE</th><th>SHAPE</th></tr></thead><tbody><tr><td>1</td><td>1</td><td>010800000000300...</td><td>CIRCULARSTRING (0 0,1 1,0 2)</td></tr><tr><td>2</td><td>2</td><td>010800000000500...</td><td>CIRCULARSTRING (0 0,1 1,0 2,-1 1,0 0)</td></tr><tr><td>3</td><td>3</td><td>010200000000200...</td><td>LINESTRING (0 0,5 10)</td></tr><tr><td>4</td><td>4</td><td>010200000000300...</td><td>LINESTRING (0 0,5 10,5 6)</td></tr><tr><td>5</td><td>5</td><td>010500000000200...</td><td>MULTILINESTRING ((10 10,12 12),(14 10,16 12))</td></tr><tr><td>6</td><td>6</td><td>010100000000000...</td><td>POINT (10 10)</td></tr><tr><td>7</td><td>7</td><td>010400000000300...</td><td>MULTIPOINT ((10 10),(12 12),(14 10))</td></tr><tr><td>8</td><td>8</td><td>010300000000100...</td><td>POLYGON ((-5 -5,5 -5,0 5,-5 -5))</td></tr><tr><td>9</td><td>9</td><td>010600000000200...</td><td>MULTIPOLYGON (((-5 -5,5 -5,0 5,-5 -5),(-2 -2,-2 0,2 ...</td></tr><tr><td>10</td><td>10</td><td>010700000000200...</td><td>GEOMETRYCOLLECTION (LINESTRING (5 10,10 12,15...</td></tr></tbody></table>		ID	SHAPE	SHAPE	1	1	010800000000300...	CIRCULARSTRING (0 0,1 1,0 2)	2	2	010800000000500...	CIRCULARSTRING (0 0,1 1,0 2,-1 1,0 0)	3	3	010200000000200...	LINESTRING (0 0,5 10)	4	4	010200000000300...	LINESTRING (0 0,5 10,5 6)	5	5	010500000000200...	MULTILINESTRING ((10 10,12 12),(14 10,16 12))	6	6	010100000000000...	POINT (10 10)	7	7	010400000000300...	MULTIPOINT ((10 10),(12 12),(14 10))	8	8	010300000000100...	POLYGON ((-5 -5,5 -5,0 5,-5 -5))	9	9	010600000000200...	MULTIPOLYGON (((-5 -5,5 -5,0 5,-5 -5),(-2 -2,-2 0,2 ...	10	10	010700000000200...	GEOMETRYCOLLECTION (LINESTRING (5 10,10 12,15...	<p>We've created a new table that has an ST_Geometry type column. This column, named "SHAPE", must be this type as opposed to a point (ST_POINT) column as later we inserted a bunch of records using different geometry types.</p> <p>Once again we're using the ST_AsWKT() method to return a readable output for the SHAPE column.</p> <p>Go back to the SQL tab and look at the different constructors for these different geometry types.</p>
	ID	SHAPE	SHAPE																																												
1	1	010800000000300...	CIRCULARSTRING (0 0,1 1,0 2)																																												
2	2	010800000000500...	CIRCULARSTRING (0 0,1 1,0 2,-1 1,0 0)																																												
3	3	010200000000200...	LINESTRING (0 0,5 10)																																												
4	4	010200000000300...	LINESTRING (0 0,5 10,5 6)																																												
5	5	010500000000200...	MULTILINESTRING ((10 10,12 12),(14 10,16 12))																																												
6	6	010100000000000...	POINT (10 10)																																												
7	7	010400000000300...	MULTIPOINT ((10 10),(12 12),(14 10))																																												
8	8	010300000000100...	POLYGON ((-5 -5,5 -5,0 5,-5 -5))																																												
9	9	010600000000200...	MULTIPOLYGON (((-5 -5,5 -5,0 5,-5 -5),(-2 -2,-2 0,2 ...																																												
10	10	010700000000200...	GEOMETRYCOLLECTION (LINESTRING (5 10,10 12,15...																																												
16	Select and run lines 301 to 303.	<pre>300 -- polygon with issues 301 INSERT INTO GEO_SPATIAL.GEOTYPES VALUES (302 NEW ST_Polygon('Polygon ((-5 -5, 5 -5, 0 5, -6 -6))') 303); 304</pre> <p>Could not execute 'INSERT INTO GEO_SPATIAL.GEOTYPES VALUES (NEW ST_Polygon('Poly SAP DBTech JDBC: [669]: spatial error: exception 1600204: A ring must be closed, at function __st_polygon__()</p>	<p>We're trying to insert another polygon type but an error is returned. Given the error, what is the issue with this syntax?</p> <p>A polygon (ring) must be closed. To close a polygon the syntax must go back to the starting point of the ring. I.e. in this case the constructor must be as follows where the final point is the same as the starting point of "-5 -5".</p> <pre>NEW ST_Polygon('Polygon ((-5 -5, 5 -5, 0 5, -6 -6, -5 -5))')</pre>																																												

	What to do	What you should see	Notes																																																																						
17	Select and run lines 314 to 346.	<pre>313 314 DROP TABLE GEO_SPATIAL.GEOMULTIDIM; 315 CREATE COLUMN TABLE GEO_SPATIAL.GEOMULTIDIM (316 ID INTEGER PRIMARY KEY GENERATED ALWAYS AS IDENTITY, 317 GEO ST_GEOMETRY 318); 319 320 INSERT INTO GEO_SPATIAL.GEOMULTIDIM VALUES(321 NEW ST_POINT('POINT (5.0 6.0)') 322); 323 INSERT INTO GEO_SPATIAL.GEOMULTIDIM VALUES(324 NEW ST_POINT('POINT Z(5.0 6.0 8.0)') 325); 326 INSERT INTO GEO_SPATIAL.GEOMULTIDIM VALUES(327 NEW ST_POINT('POINT M(5.0 6.0 1400)') 328); 329 INSERT INTO GEO_SPATIAL.GEOMULTIDIM VALUES(330 NEW ST_POINT('POINT ZM(2 3 4 1000)') 331); 332 INSERT INTO GEO_SPATIAL.GEOMULTIDIM VALUES(333 NEW ST_POINT() 334); 335 INSERT INTO GEO_SPATIAL.GEOMULTIDIM VALUES(336 NEW ST_LINESTRING('LINESTRING ZM(3 3 4 2500)') 337); 338 INSERT INTO GEO_SPATIAL.GEOMULTIDIM VALUES(339 NEW ST_LINESTRING() 340); 341 INSERT INTO GEO_SPATIAL.GEOMULTIDIM VALUES(342 NEW ST_POLYGON('POLYGON ZM((6 7 4 1800, 10 0 0 0, 0 0 0 0, 6 7 4 1800))') 343); 344 INSERT INTO GEO_SPATIAL.GEOMULTIDIM VALUES(345 NEW ST_POLYGON() 346);</pre>	<p>New for SPS10 of HANA is support for multi-dimensional data. Multi-dimensional data allows one to store an additional Z dimension in geometries. A measure, M, can also be stored.</p> <p>The Z dimension does not necessarily have to be a spatial dimension as it can be used for storing other measures such as time.</p> <p>Note that the spatial column is created as ST_Geometry which is a requirement for multi-dimensional data.</p> <p>In the constructors for the new geometries:</p> <ul style="list-style-type: none">- lines 320 to 322 is the 2 dimensional method- lines 323 to 325 is a constructor for adding a third dimension- lines 326 to 328 is a constructor for adding a measure- lines 329 to 331 adds both a third dimension and a measure <p>This shows us that multi-dimensional data can be mixed with two-dimensional data.</p> <p>We've looked at spatial types point, line, string, and polygon. Other Supported Spatial Types for Multi-Dimensional Data are: ST_MultiPoint, ST_MultiLineString, ST_MultiPolygon, and ST_GeometryCollection</p> <p>The only geometry type that is not supported (as of SPS10 of HANA) is circular string.</p>																																																																						
18	Select and run lines 350 to 358.	<pre>SELECT ID, GEO.ST_ASWKT(), --GEO.ST_Z(), GEO.ST_ZMAX(), GEO.ST_ZMIN(), --GEO.ST_M(),</pre> <table><thead><tr><th></th><th>ID</th><th>GEO.ST_ASWKT()</th><th>GEO.ST_ZMAX()</th><th>GEO.ST_ZMIN()</th><th>GEO.ST_MMV()</th><th>GEO.ST_MMV()</th></tr></thead><tbody><tr><td>1</td><td>POINT (5 6)</td><td></td><td>?</td><td>?</td><td>?</td><td>?</td></tr><tr><td>2</td><td>POINT Z (5 6 8)</td><td></td><td>8</td><td>8</td><td>?</td><td>?</td></tr><tr><td>3</td><td>POINT M (5 6 1400)</td><td></td><td>?</td><td>?</td><td>1,400</td><td>1,400</td></tr><tr><td>4</td><td>POINT ZM (2 3 4 1000)</td><td></td><td>4</td><td>4</td><td>1,000</td><td>1,000</td></tr><tr><td>5</td><td>POINT EMPTY</td><td></td><td>?</td><td>?</td><td>?</td><td>?</td></tr><tr><td>6</td><td>LINESTRING Z...</td><td></td><td>4</td><td>?</td><td>2,600</td><td>2,200</td></tr><tr><td>7</td><td>LINESTRING E...</td><td></td><td>?</td><td>?</td><td>?</td><td>?</td></tr><tr><td>8</td><td>POLYGON ZM (...)</td><td></td><td>4</td><td>4</td><td>1,900</td><td>1,800</td></tr><tr><td>9</td><td>POLYGON EMPTY</td><td></td><td>?</td><td>?</td><td>?</td><td>?</td></tr></tbody></table>		ID	GEO.ST_ASWKT()	GEO.ST_ZMAX()	GEO.ST_ZMIN()	GEO.ST_MMV()	GEO.ST_MMV()	1	POINT (5 6)		?	?	?	?	2	POINT Z (5 6 8)		8	8	?	?	3	POINT M (5 6 1400)		?	?	1,400	1,400	4	POINT ZM (2 3 4 1000)		4	4	1,000	1,000	5	POINT EMPTY		?	?	?	?	6	LINESTRING Z...		4	?	2,600	2,200	7	LINESTRING E...		?	?	?	?	8	POLYGON ZM (...)		4	4	1,900	1,800	9	POLYGON EMPTY		?	?	?	?	<p>There are several new methods that accompany multi-dimensional data. ST_ZMax as an example will return the maximum Z value of a geometry...note that this is not an aggregate function as it's a row-cell level function.</p> <p>When ZMax is used on a 3-dimensional point it returns the single Z value...the same as ZMin. When these functions are used on 2-dimensional geometries, a NULL / ? is returned.</p>
	ID	GEO.ST_ASWKT()	GEO.ST_ZMAX()	GEO.ST_ZMIN()	GEO.ST_MMV()	GEO.ST_MMV()																																																																			
1	POINT (5 6)		?	?	?	?																																																																			
2	POINT Z (5 6 8)		8	8	?	?																																																																			
3	POINT M (5 6 1400)		?	?	1,400	1,400																																																																			
4	POINT ZM (2 3 4 1000)		4	4	1,000	1,000																																																																			
5	POINT EMPTY		?	?	?	?																																																																			
6	LINESTRING Z...		4	?	2,600	2,200																																																																			
7	LINESTRING E...		?	?	?	?																																																																			
8	POLYGON ZM (...)		4	4	1,900	1,800																																																																			
9	POLYGON EMPTY		?	?	?	?																																																																			

	What to do	What you should see	Notes
19	Uncomment line 352 and run lines 350 to 358 again.		<p>What does this error mean? The ST_Z method can only be used with geometries that have a Z dimension or value.</p> <p>If you were to put the comment back in line 352, uncomment line 355, and rerun the query you would get a similar error. The ST_M method can only be used with geometries that have a measure value.</p> <p>In the next step we will see how to avoid these errors.</p>
20	Select and run lines 362 to 373.		<p>Other new functions for multi-dimensional data are the ST_IsMeasured function and the ST_Is3D function. These are Boolean functions which return a 1 or a 0.</p> <p>In this query we want to bring back only records which have a measure or are 3d. Here we use case statements that utilize the aforementioned functions to return a measure or 3rd dimension when applicable or a NULL / ? value when those values are not applicable.</p> <p>Another method to avoid the error in the previous step is to use only the max or min functions (e.g. ST_ZMax) as opposed to using the ST_Z or ST_M functions.</p>
21	In your GEO_SPATIAL schema Tables, right click on the GEOTYPES table and choose Open Data Preview.		 <p>New in SPS10 was a spatial preview in HANA Studio. In previous versions when one previewed (using Open Data Preview) a spatial column, the output was in a binary format. Now the output is in XML with an SVG document type.</p>

	What to do	What you should see	Notes
	Double click on any of the SHAPE column records.	 <p>The screenshot shows a table with columns 'ID' and 'SHAPE'. The 'SHAPE' column contains XML data. Below the table, a window titled 'SHAPE - Row:4' is open, showing a spatial preview of a line segment. The window has tabs for 'Map' and 'XML', and buttons for 'Previous', 'Next', 'Copy', 'Save as...', and 'Close'.</p>	The SHAPE file is now shown as an SVG via the spatial preview in HANA Studio.

	What to do	What you should see	Notes
22	Relax as you're done.		<p>For lots of videos on lots of topics on SAP HANA, please visit the SAP HANA Academy's site here.</p> <p>As mentioned earlier, be sure to check out the Live 4 / ERP Agility course created by the HANA Academy where you'll use spatial queries as well as utilize the Here Maps API in a SAPUI5 application to create a mapping application. You'll also see how to install some of the additional spatial content, such as zip-code tables, available for licensed HANA users.</p>