Record Tracking Satellites

Book Sections Reference: 5.3.5 & 12.1.8

Record Tracking Satellites provide the ability to track when business-keys were last seen provided by the source. This is required especially in the context when some system limitation is causing business keys to sometimes disappear, reappear. Record tracking satellites can be used to also highlight the combination of business keys, i.e. they can exist on Link tables as well. However, for Link tables we would propose you consider a Effectivity satellites, as it is probably more suited.

WhereScape 3D

Dependent objects

Sample Repo

- Look at the examples created in 3D repo **wsDVSamples.repo**

Model Conversion Rules

- 1 additional rule for the generation of Data Vault **(ws3d_grv - Create Record tracking satellites)**
- 1 additional rule for generation of Load and Stage **(ws3d_rvls - Record tracking satellite - staging)**

Copy from wsDVSamples.repo or import **RecordTracking_MCR.xml**

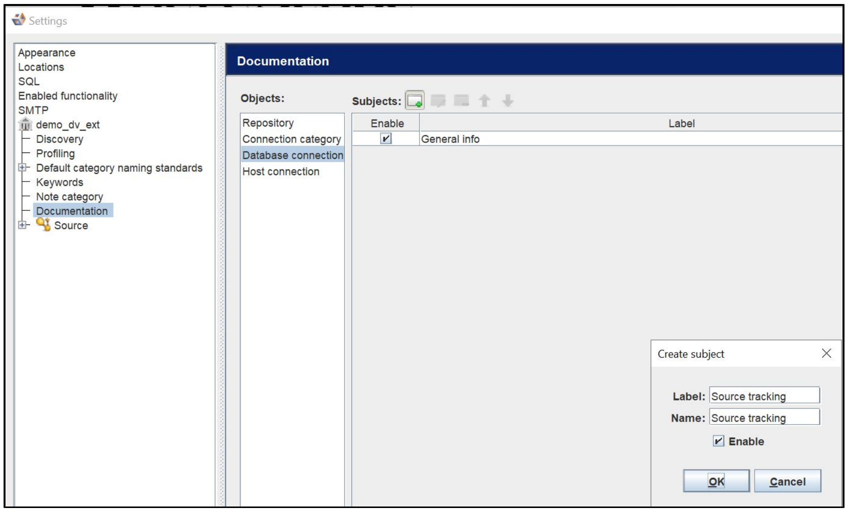
Template

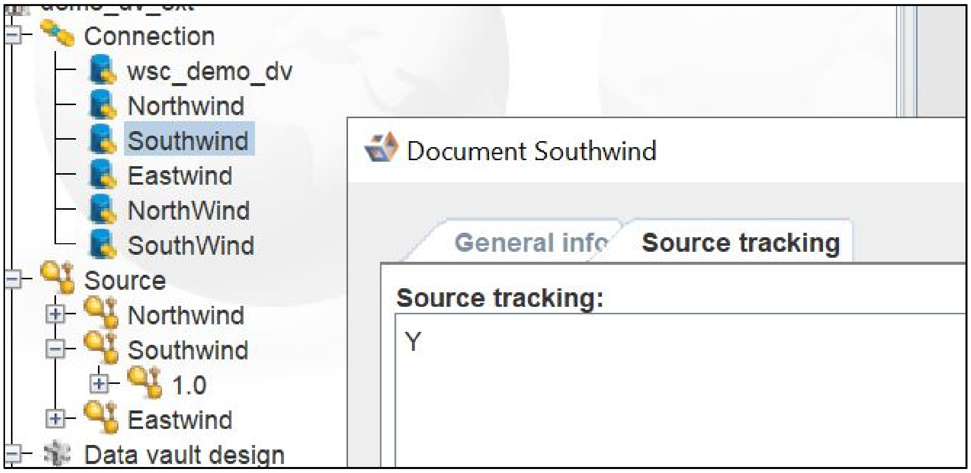
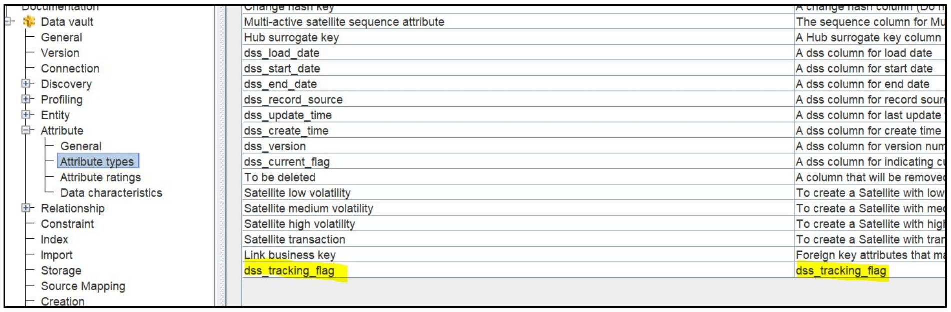
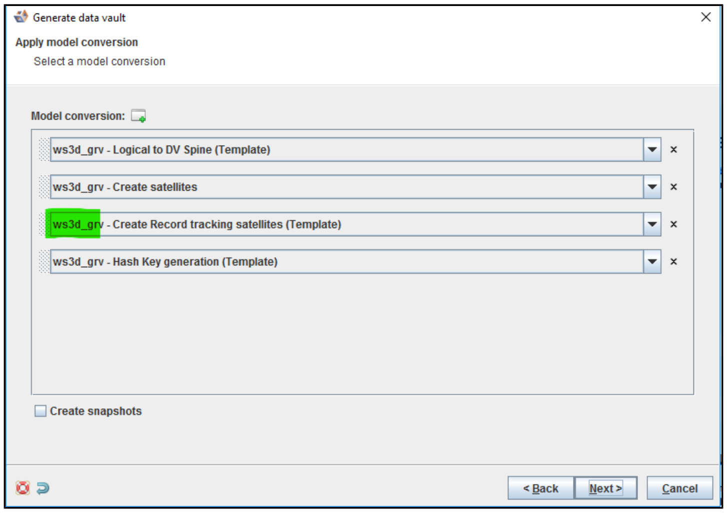
- 1 additional Query generation template used by the Load and Stage **(ws3d_dv_record_tracking)**

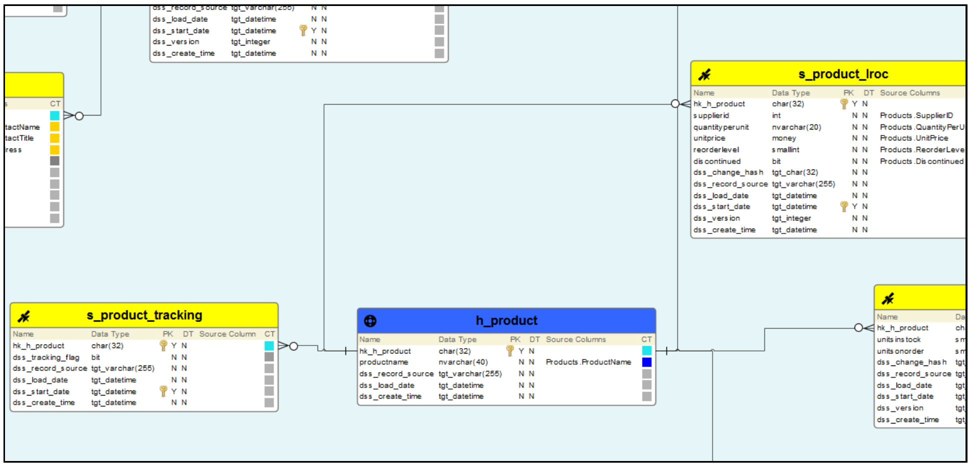
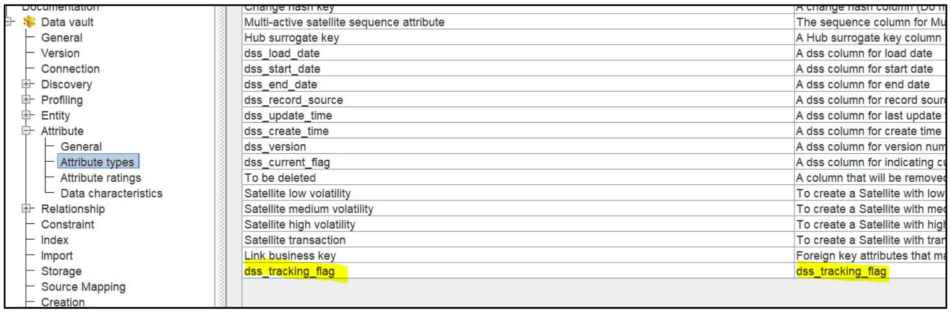
Copy from wsDVSamples.repo or import **RecordTracking_TEMPLATES.xml**

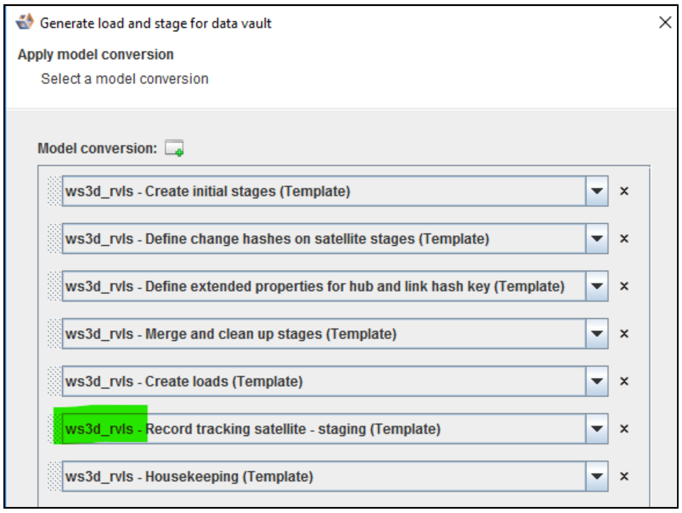
Steps to implement

If a particular stage is not mentioned, then you can assume it would follow the standard process of generating a Data Vault.

#	Category	Description
1	Global settings	Record-tracking is enabled on connection level You have to add a documentation subject on "Database connection" - level named "Source tracking"  Settings Appearance Locations SQL Enabled functionality SMTP demo_dv_ext Discovery Profiling Default category naming standards Keywords Note category Documentation Source Documentation Objects: Subjects: Repository Connection category Database connection Host connection Enable Label Create subject OK Cancel ☑ General info Label: Source tracking Name: Source tracking ☑ Enable

<table>
<tr><td>2</td><td>Connection level</td><td>Then set insert into the desired connection an "Y" for enabled - anything else for disabled

Connection
wsc_demo_dv
Northwind
Southwind
Eastwind
NorthWind
SouthWind
Source
Northwind
Southwind
1.0
Eastwind
Data vault design
Document Southwind
General info | Source tracking
Source tracking:
Y</td></tr>
<tr><td>3</td><td>Data Vault</td><td>Add Attribute-Type "dss_tracking_flag" to Data Vault

Data vault
General
Version
Connection
Discovery
Profiling
Entity
Attribute
General
Attribute types
Attribute ratings
Data characteristics
Relationship
Constraint
Index
Import
Storage
Source Mapping
Creation
<table>
<tr><td>Change hash key</td><td>A change hash column (Do n</td></tr>
<tr><td>Multi-active satellite sequence attribute</td><td>The sequence column for Mu</td></tr>
<tr><td>Hub surrogate key</td><td>A Hub surrogate key column</td></tr>
<tr><td>dss_load_date</td><td>A dss column for load date</td></tr>
<tr><td>dss_start_date</td><td>A dss column for start date</td></tr>
<tr><td>dss_end_date</td><td>A dss column for end date</td></tr>
<tr><td>dss_record_source</td><td>A dss column for record sour</td></tr>
<tr><td>dss_update_time</td><td>A dss column for last update</td></tr>
<tr><td>dss_create_time</td><td>A dss column for create time</td></tr>
<tr><td>dss_version</td><td>A dss column for version num</td></tr>
<tr><td>dss_current_flag</td><td>A dss column for indicating cu</td></tr>
<tr><td>To be deleted</td><td>A column that will be removed</td></tr>
<tr><td>Satellite low volatility</td><td>To create a Satellite with low</td></tr>
<tr><td>Satellite medium volatility</td><td>To create a Satellite with med</td></tr>
<tr><td>Satellite high volatility</td><td>To create a Satellite with high</td></tr>
<tr><td>Satellite transaction</td><td>To create a Satellite with tran</td></tr>
<tr><td>Link business key</td><td>Foreign key attributes that ma</td></tr>
<tr><td>dss_tracking_flag</td><td>dss_tracking_flag</td></tr>
</table></td></tr>
<tr><td></td><td></td><td>Generate from the Data Vault Design Model with the standard-process enhanced with the following Rules

Generate data vault
Apply model conversion
Select a model conversion
Model conversion:
ws3d_grv - Logical to DV Spine (Template)
ws3d_grv - Create satellites
ws3d_grv - Create Record tracking satellites (Template)
ws3d_grv - Hash Key generation (Template)
□ Create snapshots
&lt; Back | Next &gt; | Cancel</td></tr>
</table>

		In the Data Vault Model you should notice that for every hub and link that receives data from the rts-active connection a satellite "_tracking" has been created:  s_product_lroc Name Data Type PK DT Source Columns hk_h_product char(32) Y N supplierid int N N Products.SupplierID quantityperunit nvarchar(20) N N Products.QuantityPerU unitprice money N N Products.UnitPrice reorderlevel smallint N N Products.ReorderLeve discontinued bit N N Products.Discontinued dss_change_hash tgt_char(32) N N dss_record_source tgt_varchar(255) N N dss_load_date tgt_datetime N N dss_start_date tgt_datetime Y N dss_version tgt_integer N N dss_create_time tgt_datetime N N s_product_tracking Name Data Type PK DT Source Column CT hk_h_product char(32) Y N dss_tracking_flag bit N N dss_record_source tgt_varchar(255) N N dss_load_date tgt_datetime N N dss_start_date tgt_datetime Y N dss_create_time tgt_datetime N N h_product Name Data Type PK DT Source Columns CT hk_h_product char(32) Y N productname nvarchar(40) N N Products.ProductName dss_record_source tgt_varchar(255) N N dss_load_date tgt_datetime N N dss_create_time tgt_datetime N N Things to note: • The dss_tracking_flage indicates with 0 (not seen) or 1 (seen) that with the last loading this key has been seen or not. • There should't be any other columns (except for the standard-dates, recordsource and the key itself)
4	Load And Stage	Add Attribute-Type "dss_tracking_flag" to Load and Stage  Documentation Data vault General Version Connection Discovery Profiling Entity Attribute General Attribute types Attribute ratings Data characteristics Relationship Constraint Index Import Storage Source Mapping Creation Change hash key | A change hash column (Do n Multi-active satellite sequence attribute | The sequence column for Mu Hub surrogate key | A Hub surrogate key column dss_load_date | A dss column for load date dss_start_date | A dss column for start date dss_end_date | A dss column for end date dss_record_source | A dss column for record sourc dss_update_time | A dss column for last update dss_create_time | A dss column for create time dss_version | A dss column for version num dss_current_flag | A dss column for indicating cu To be deleted | A column that will be removed Satellite low volatility | To create a Satellite with low Satellite medium volatility | To create a Satellite with med Satellite high volatility | To create a Satellite with high Satellite transaction | To create a Satellite with tran Link business key | Foreign key attributes that ma dss_tracking_flag | dss_tracking_flag

5		Generate from the Data Vault Design Model with the standard-process enhanced with the following Rules:  Generate load and stage for data vault **Apply model conversion** Select a model conversion **Model conversion:** ws3d_rvls - Create initial stages (Template) ws3d_rvls - Define change hashes on satellite stages (Template) ws3d_rvls - Define extended properties for hub and link hash key (Template) ws3d_rvls - Merge and clean up stages (Template) ws3d_rvls - Create loads (Template) ws3d_rvls - Record tracking satellite - staging (Template) ws3d_rvls - Housekeeping (Template) Things to note: • There will be no additional stage for the tracking-satellites as they use the staging from the hub or link. • The DW-Query has already been created and will be exported to RED - that is why there should be 3D-templates for different target-DWH. • The DW-Query and the column-transformation will determine the query that will be used in the final script.

WhereScape RED

Dependent objects

Templates

- dv_perm templates need additional logic to remove the Current Satellite version join in the case of a tracking satellite

Steps to implement

Look at the sample template **cust_sqlserver_proc_dv_perm.peb** for an example of what changes are required.

#	Description of change	Snippet Code

1	The dv_perm template for your target platform needs to be adjusted in order to use record tracking satellites	{%- set isTrackingSatellite = false -%}
		{%- from table.columns as col where col.name == "dss_tracking_flag" -%}
		{%- set isTrackingSatellite = true -%}
		{%- endfrom -%}

2	This variable should then be used to NOT **addSatCurrentVersion**	{%- elseif table.objectType == Types.ObjectType.Satellite and not (isTrackingSatellite or isEffectivitySatellite) %}
		-->addSatCurrentVersion<--
		{{addSatCurrentVersion()}}

3	This variable should then be used to NOT **addSatWhereNotExists**	{%- elseif table.objectType == Types.ObjectType.Satellite and not (isTrackingSatellite or isEffectivitySatellite) %}
		-->addSatWhereNotExists<--
		{{addSatWhereNotExists()}}

Effectivity Satellites

Book Sections Reference: 5.3.4: "Its purpose is to track when the **link** is active according to the business and provides begin and end dates for this purposes"

See also: "The Data Vault Guru: A pragmatic guide on building a data vault" Chapter 6.3 Effectivity Satellite (Page 301)

As part of the enablement pack WhereScape ships the functionality to use the "Applied Date" to determine the effectivity of the link. You can choose whether you want a default system generated date to populate "Applied Date" or whether you want to choose no more than 1 date from the source.

In cases where multiple dates or fields from the source describes the temporality of the data we provide an option for a different approach, and not to resolve it during load of Raw Vault, but rather further downstream. See details under our Multi-Temporal sources enablement pack.

WhereScape 3D

Dependent objects

Sample Repo

- Look at the examples created in 3D repo **wsDVSamples**

Model Conversion Rules

- 1 of 3 generation of Data Vault rule **(ws3d_grv - Create effectivity satellites)**
- 1 of 3 generation of Data Vault rule **(ws3d_grv - Prepare effectivity satellites)**
- 1 of 3 generation of Data Vault rule **(ws3d_grv - Housekeeping effectivity satellites)**
- 1 additional generation of load and stage **(ws3d_rvls - Create effectivity satellites)**

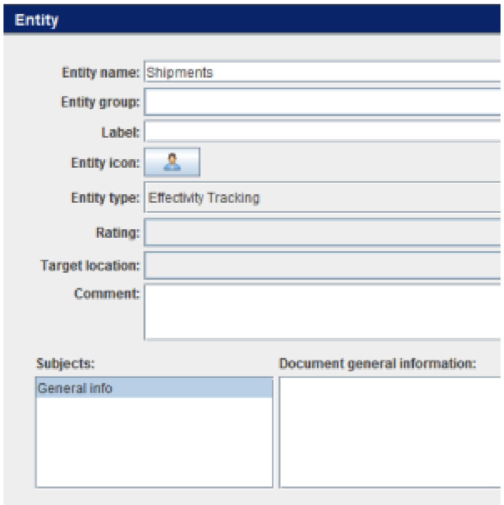
Copy from wsDVSamples or import **EffectivityTracking_MCR.xml**

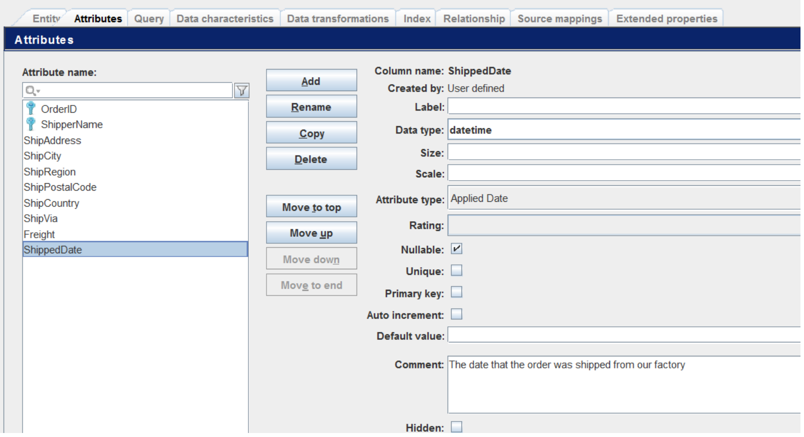
Template

No additional templates are required

Steps to implement

If a particular stage is not mentioned, then you can assume it would follow the standard process of generating a Data Vault.

#	Category	Description
1	Data Vault Design	For the purpose of this document, the Shipments-table has been chosen to create an Effectivity Satellite upon. In this scenario the OrderID shall be the DrivingKey (dss_driving_key) and the ShippedDate has been declared as Applied Date (dss_applied_date) - which is purely for the example purposes. The entity containing the link business keys has been set to the entity-type "Effectivity Tracking"  Entity Entity name: Shipments Entity group: Label: Entity icon: Entity type: Effectivity Tracking Rating: Target location: Comment: Subjects: General info Document general information:

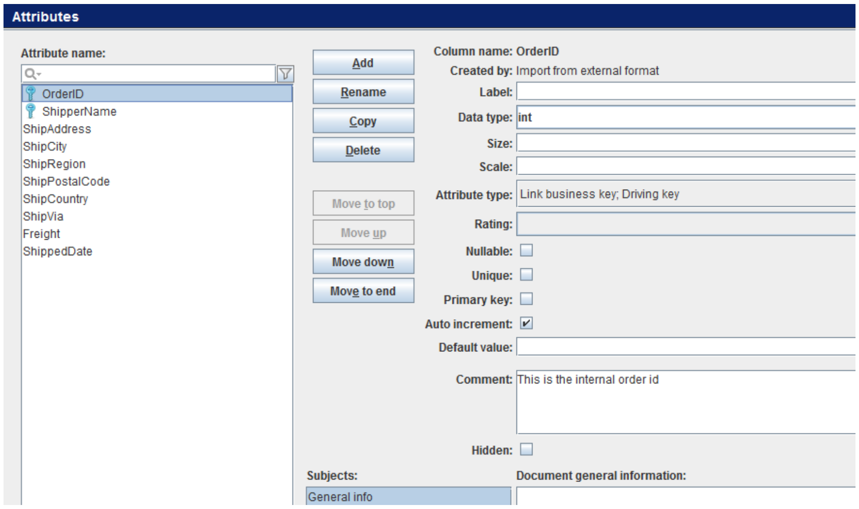


Attributes

Attribute name:

- OrderID
- ShipperName
- ShipAddress
- ShipCity
- ShipRegion
- ShipPostalCode
- ShipCountry
- ShipVia
- Freight
- ShippedDate

Column name: ShippedDate
Created by: User defined
Label:
Data type: datetime
Size:
Scale:
Attribute type: Applied Date
Rating:
Nullable: ☑
Unique: □
Primary key: □
Auto increment: □
Default value:
Comment: The date that the order was shipped from our factory
Hidden: □



Attributes

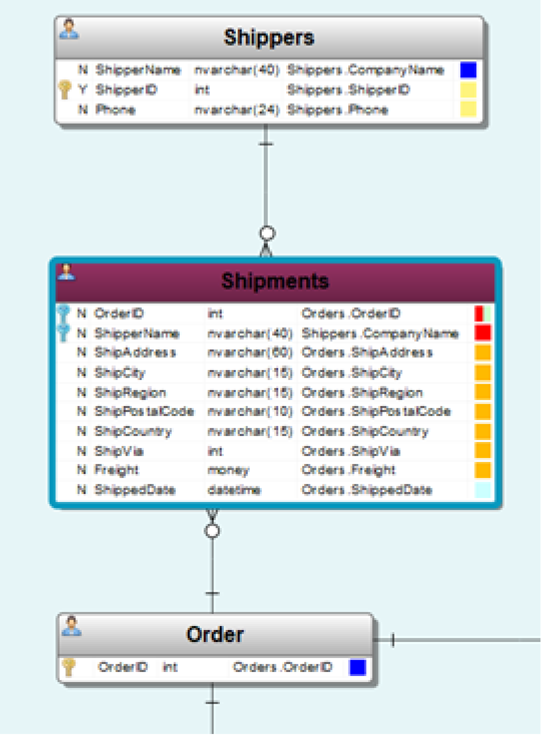
Attribute name:

- OrderID
- ShipperName
- ShipAddress
- ShipCity
- ShipRegion
- ShipPostalCode
- ShipCountry
- ShipVia
- Freight
- ShippedDate

Column name: OrderID
Created by: Import from external format
Label:
Data type: int
Size:
Scale:
Attribute type: Link business key; Driving key
Rating:
Nullable: □
Unique: □
Primary key: □
Auto increment: ☑
Default value:
Comment: This is the internal order id
Hidden: □

Subjects: General info

Document general information:



Shippers

N	ShipperName	nvarchar(40)	Shippers.CompanyName	🟦
Y	ShipperID	int	Shippers.ShipperID	🟨
N	Phone	nvarchar(24)	Shippers.Phone	🟨

Shipments

N	OrderID	int	Orders.OrderID	🟥
N	ShipperName	nvarchar(40)	Shippers.CompanyName	🟥
N	ShipAddress	nvarchar(60)	Orders.ShipAddress	🟧
N	ShipCity	nvarchar(15)	Orders.ShipCity	🟧
N	ShipRegion	nvarchar(15)	Orders.ShipRegion	🟧
N	ShipPostalCode	nvarchar(10)	Orders.ShipPostalCode	🟧
N	ShipCountry	nvarchar(15)	Orders.ShipCountry	🟧
N	ShipVia	int	Orders.ShipVia	🟧
N	Freight	money	Orders.Freight	🟧
N	ShippedDate	datetime	Orders.ShippedDate	⬜

Order

OrderID	int	Orders.OrderID	🟦

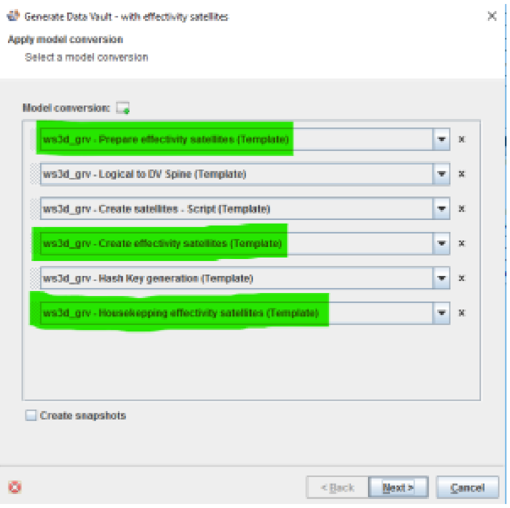
Things to note:

- Attribute types that are being used are:
  - Driving Key (light green)
  - Applied Date (light blue)
  - Entity-Type "Effectivity Tracking" (burgundy)
- The single source of each attribute is shown in the diagram
- Using the dss_applied_date is optional - if not used, the dss_create_time will be used as source

2

Data Vault Model

Generate from the Data Vault Design Model with the standard-process enhanced with the following Rules:



Generate Data Vault - with effectivity satellites

Apply model conversion

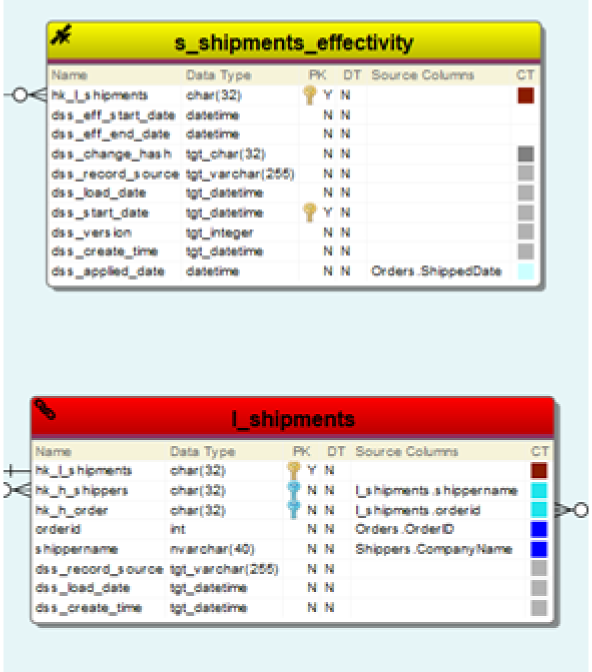
Select a model conversion

Model conversion:

- ws3d_grv - Prepare effectivity satellites (Template)
- ws3d_grv - Logical to DV Spine (Template)
- ws3d_grv - Create satellites - Script (Template)
- ws3d_grv - Create effectivity satellites (Template)
- ws3d_grv - Hash Key generation (Template)
- ws3d_grv - Housekeeping effectivity satellites (Template)

□ Create snapshots

The Data Vault Model should look like this



s_shipments_effectivity

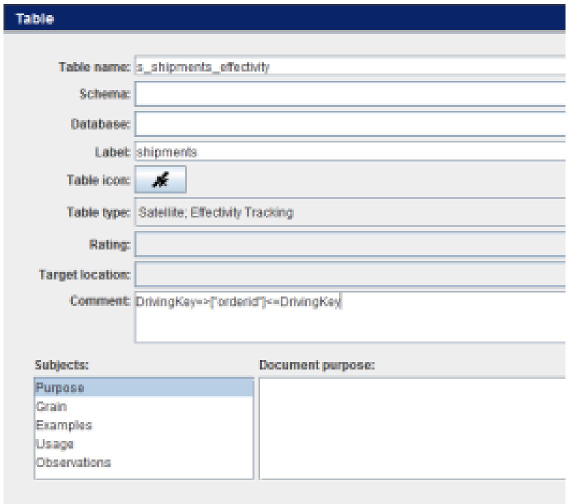
Name	Data Type	PK	DT	Source Columns	CT
hk_l_shipments	char(32)	Y	N		🟥
dss_eff_start_date	datetime	N	N		
dss_eff_end_date	datetime	N	N		
dss_change_hash	tgt_char(32)	N	N		▨
dss_record_source	tgt_varchar(255)	N	N		▨
dss_load_date	tgt_datetime	N	N		▨
dss_start_date	tgt_datetime	Y	N		▨
dss_version	tgt_integer	N	N		▨
dss_create_time	tgt_datetime	N	N		▨
dss_applied_date	datetime	N	N	Orders.ShippedDate	□

l_shipments

Name	Data Type	PK	DT	Source Columns	CT
hk_l_shipments	char(32)	Y	N		🟥
hk_h_shippers	char(32)	N	N	l_shipments.shippername	🟦
hk_h_order	char(32)	N	N	l_shipments.orderid	🟦
orderid	int	N	N	Orders.OrderID	🟦
shippername	nvarchar(40)	N	N	Shippers.CompanyName	🟦
dss_record_source	tgt_varchar(255)	N	N		▨
dss_load_date	tgt_datetime	N	N		▨
dss_create_time	tgt_datetime	N	N		▨

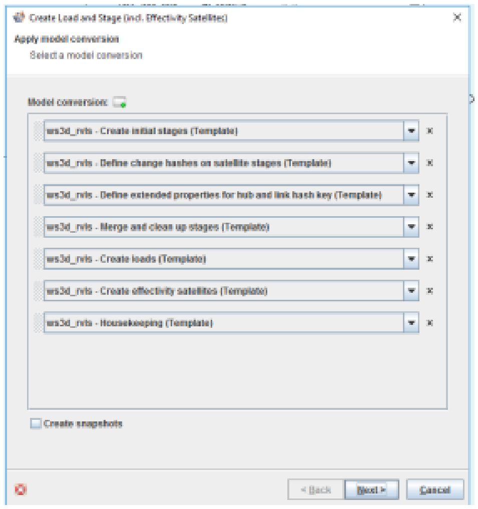
Things to note:

- All of the extra DV control columns have been added to the entities as per DV 2.0 rules
- Hub and Change hash keys have been added where needed.
- The setting of the attribute "Driving Key" has been copied into the comment of the newly created satellite.

		 Table Table name: s_shipments_effectivity Schema: Database: Label: shipments Table icon: Table type: Satellite; Effectivity Tracking Rating: Target location: Comment: DrivingKey=>["orderid"]<=DrivingKey Subjects: Purpose Grain Examples Usage Observations Document purpose:

3 **Load and Staging Model**

Generate from the Data Vault Design Model with the standard-process enhanced with the following Rules:



Create Load and Stage (incl. Effectivity Satellites)

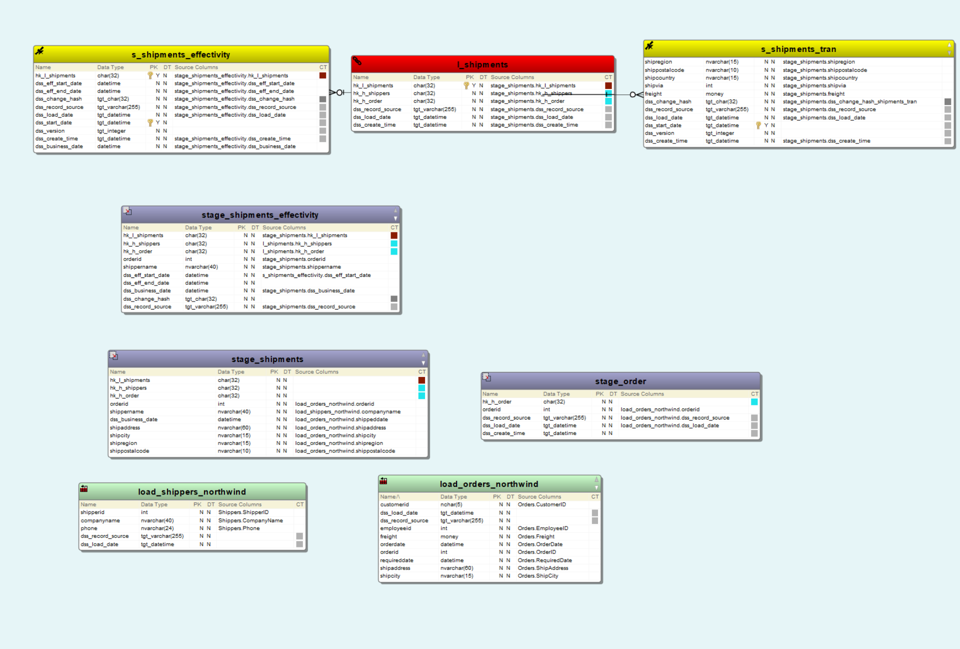
Apply model conversion

Select a model conversion

Model conversion:

- ws3d_rvis - Create initial stages (Template)
- ws3d_rvis - Define change hashes on satellite stages (Template)
- ws3d_rvis - Define extended properties for hub and link hash key (Template)
- ws3d_rvis - Merge and clean up stages (Template)
- ws3d_rvis - Create loads (Template)
- ws3d_rvis - Create effectivity satellites (Template)
- ws3d_rvis - Housekeeping (Template)

☐ Create snapshots



s_shipments_effectivity

Name	Data Type	PK	DT	Source Columns	CT
hk_l_shipments	char(32)	Y	N	stage_shipments_effectivity.hk_l_shipments	🟥
dss_eff_start_date	datetime	N	N	stage_shipments_effectivity.dss_eff_start_date	
dss_eff_end_date	datetime	N	N	stage_shipments_effectivity.dss_eff_end_date	
dss_change_hash	tgt_char(32)	N	N	stage_shipments_effectivity.dss_change_hash	■
dss_record_source	tgt_varchar(255)	N	N	stage_shipments_effectivity.dss_record_source	□
dss_load_date	tgt_datetime	N	N	stage_shipments_effectivity.dss_load_date	□
dss_start_date	tgt_datetime	Y	N		□
dss_version	tgt_integer	N	N		□
dss_create_time	tgt_datetime	N	N	stage_shipments_effectivity.dss_create_time	□
dss_business_date	datetime	N	N	stage_shipments_effectivity.dss_business_date	

l_shipments

Name	Data Type	PK	DT	Source Columns	CT
hk_l_shipments	char(32)	Y	N	stage_shipments.hk_l_shipments	🟥
hk_h_shippers	char(32)	N	N	stage_shipments.hk_h_shippers	🟦
hk_h_order	char(32)	N	N	stage_shipments.hk_h_order	🟦
dss_record_source	tgt_varchar(255)	N	N	stage_shipments.dss_record_source	□
dss_load_date	tgt_datetime	N	N	stage_shipments.dss_load_date	□
dss_create_time	tgt_datetime	N	N	stage_shipments.dss_create_time	□

s_shipments_tran

shipregion	nvarchar(15)	N	N	stage_shipments.shipregion	
shippostalcode	nvarchar(10)	N	N	stage_shipments.shippostalcode	
shipcountry	nvarchar(15)	N	N	stage_shipments.shipcountry	
shipvia	int	N	N	stage_shipments.shipvia	
freight	money	N	N	stage_shipments.freight	
dss_change_hash	tgt_char(32)	N	N	stage_shipments.dss_change_hash_shipments_tran	■
dss_record_source	tgt_varchar(255)	N	N	stage_shipments.dss_record_source	□
dss_load_date	tgt_datetime	N	N	stage_shipments.dss_load_date	□
dss_start_date	tgt_datetime	Y	N		□
dss_version	tgt_integer	N	N		□
dss_create_time	tgt_datetime	N	N	stage_shipments.dss_create_time	□

stage_shipments_effectivity

Name	Data Type	PK	DT	Source Columns	CT
hk_l_shipments	char(32)	N	N	stage_shipments.hk_l_shipments	🟥
hk_h_shippers	char(32)	N	N	l_shipments.hk_h_shippers	🟦
hk_h_order	char(32)	N	N	l_shipments.hk_h_order	🟦
orderid	int	N	N	stage_shipments.orderid	
shippername	nvarchar(40)	N	N	stage_shipments.shippername	
dss_eff_start_date	datetime	N	N	s_shipments_effectivity.dss_eff_start_date	
dss_eff_end_date	datetime	N	N		
dss_business_date	datetime	N	N	stage_shipments.dss_business_date	
dss_change_hash	tgt_char(32)	N	N		■
dss_record_source	tgt_varchar(255)	N	N	stage_shipments.dss_record_source	□

stage_shipments

Name	Data Type	PK	DT	Source Columns	CT
hk_l_shipments	char(32)	N	N		🟥
hk_h_shippers	char(32)	N	N		🟦
hk_h_order	char(32)	N	N		🟦
orderid	int	N	N	load_orders_northwind.orderid	
shippername	nvarchar(40)	N	N	load_shippers_northwind.companyname	
dss_business_date	datetime	N	N	load_orders_northwind.shippeddate	
shipaddress	nvarchar(60)	N	N	load_orders_northwind.shipaddress	
shipcity	nvarchar(15)	N	N	load_orders_northwind.shipcity	
shipregion	nvarchar(15)	N	N	load_orders_northwind.shipregion	
shippostalcode	nvarchar(10)	N	N	load_orders_northwind.shippostalcode	

stage_order

Name	Data Type	PK	DT	Source Columns	CT
hk_h_order	char(32)	N	N		🟦
orderid	int	N	N	load_orders_northwind.orderid	
dss_record_source	tgt_varchar(255)	N	N	load_orders_northwind.dss_record_source	□
dss_load_date	tgt_datetime	N	N	load_orders_northwind.dss_load_date	□
dss_create_time	tgt_datetime	N	N		□

load_shippers_northwind

Name	Data Type	PK	DT	Source Columns	CT
shipperid	int	N	N	Shippers.ShipperID	
companyname	nvarchar(40)	N	N	Shippers.CompanyName	
phone	nvarchar(24)	N	N	Shippers.Phone	
dss_record_source	tgt_varchar(255)	N	N		□
dss_load_date	tgt_datetime	N	N		□

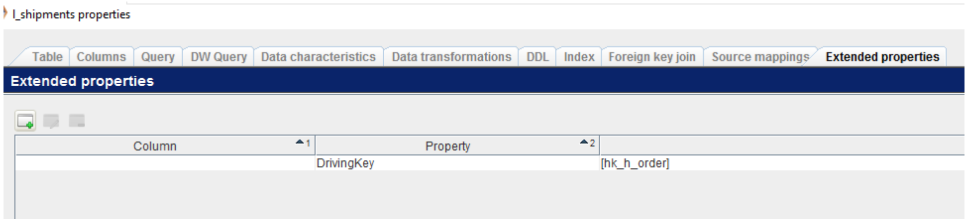
load_orders_northwind

Name/\	Data Type	PK	DT	Source Columns	CT
customerid	nchar(5)	N	N	Orders.CustomerID	
dss_load_date	tgt_datetime	N	N		□
dss_record_source	tgt_varchar(255)	N	N		□
employeeid	int	N	N	Orders.EmployeeID	
freight	money	N	N	Orders.Freight	
orderdate	datetime	N	N	Orders.OrderDate	
orderid	int	N	N	Orders.OrderID	
requireddate	datetime	N	N	Orders.RequiredDate	
shipaddress	nvarchar(60)	N	N	Orders.ShipAddress	
shipcity	nvarchar(15)	N	N	Orders.ShipCity	

Things to note:

- There should be an additional stage for every effective satellite which is sourced mainly from the original stage for the link - but also from the link and the effectivity-satellite itself

The setting "Driving Key" from the comment has now moved to the extended Properties:



l_shipments properties

Extended properties

Column	Property	
	DrivingKey	[hk_h_order]

4 **Export to RED**

Export the RED Export model version to RED using the Basic Process described previously.
The model should deploy to RED without any issues and all Load, Stage and Data Vault objects can be loaded successfully.

WhereScape RED

*Currently only target SQL Server is supported

Dependent objects

Templates

3 new templates need to be used or the logic incorporated into existing templates

- cust_sqlserver_proc_dv_stage (Procedure template for Data Vault Stage)
- cust_sqlserver_proc_dv_stage_effectivity_utility
- cust_sqlserver_utility_dv

Steps to implement

Via application installation

#	Description of change
1	Unzip file app_tem_SQLDV_EFSAT_202210040854
2	Install application DV_SQL_EFSAT from wsl files in the zipped folder

Via copying the content from template files

The 3 templates have also been attached separately to look at.

Multi-Temporal Sources

Source systems often do not store only a single version of the data. There are multiple fields that describe the temporality of the source data. This could be technical dates, business effective dates and even CDC dates (or a combination of all). This leaves a question of how to adequately load the data into the Raw Vault. An option is simply to add all of the fields describing the temporality as additional metadata on the satellites. Then the correct effectivity can be determined downstream as this would consist of some business logic.

WhereScape 3D

Dependent objects

Sample Repo

- Look at the examples created in 3D repo **wsDVSamples**

Model Conversion Rules

- Group Temporal Metadata from Source
    - ws3d_grv - Add Temporal Metadata to all Satellites
    - ws3d_rvls - Remove Custom Attribute types

Copy from wsDVSamples or import **Temporal_Source_additional_fields.xml**

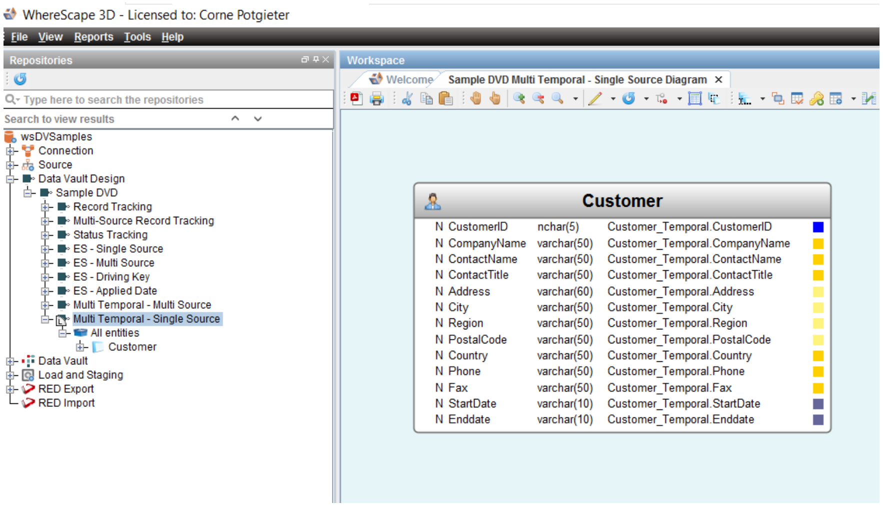
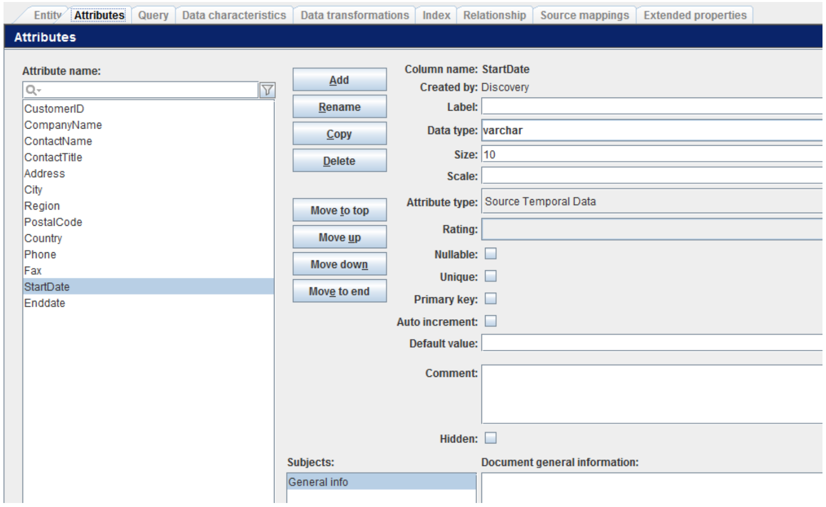
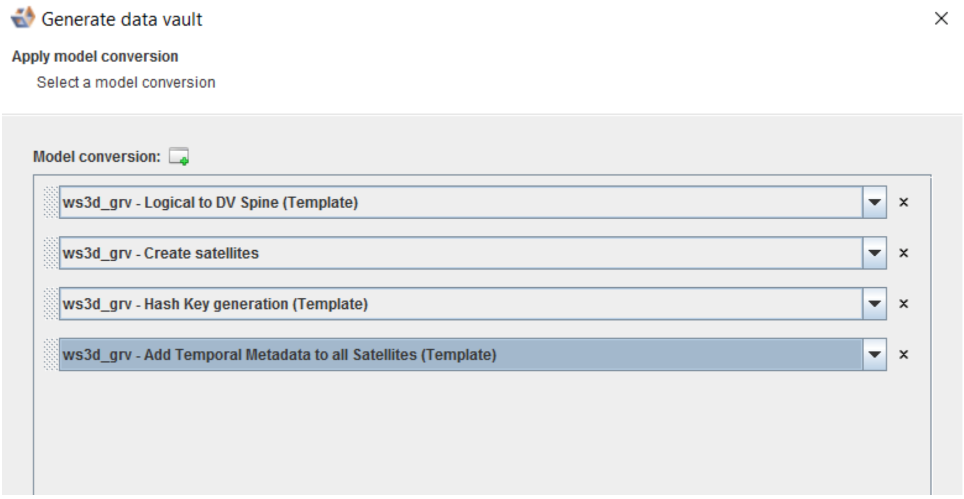
Template

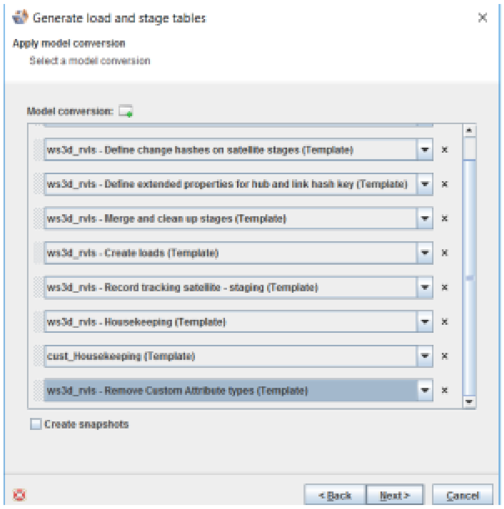
No additional templates required

Steps to implement

If a particular stage is not mentioned, then you can assume it would follow the standard process of generating a Data Vault.

#	Category	Description

1	Data Vault Design	See these multi temporal examples in the Samples repo  WhereScape 3D - Licensed to: Corne Potgieter File View Reports Tools Help Repositories Type here to search the repositories Search to view results wsDVSamples Connection Source Data Vault Design Sample DVD Record Tracking Multi-Source Record Tracking Status Tracking ES - Single Source ES - Multi Source ES - Driving Key ES - Applied Date Multi Temporal - Multi Source Multi Temporal - Single Source All entities Customer Data Vault Load and Staging RED Export RED Import Workspace Welcome Sample DVD Multi Temporal - Single Source Diagram Customer N CustomerID nchar(5) Customer_Temporal.CustomerID N CompanyName varchar(50) Customer_Temporal.CompanyName N ContactName varchar(50) Customer_Temporal.ContactName N ContactTitle varchar(50) Customer_Temporal.ContactTitle N Address varchar(60) Customer_Temporal.Address N City varchar(50) Customer_Temporal.City N Region varchar(50) Customer_Temporal.Region N PostalCode varchar(50) Customer_Temporal.PostalCode N Country varchar(50) Customer_Temporal.Country N Phone varchar(50) Customer_Temporal.Phone N Fax varchar(50) Customer_Temporal.Fax N StartDate varchar(10) Customer_Temporal.StartDate N Enddate varchar(10) Customer_Temporal.Enddate Tag the source fields that describe the temporality or effectivity as "Source Temporal Data"  Entity Attributes Query Data characteristics Data transformations Index Relationship Source mappings Extended properties Attributes Add Rename Copy Delete Move to top Move up Move down Move to end Attribute name: CustomerID CompanyName ContactName ContactTitle Address City Region PostalCode Country Phone Fax StartDate Enddate Column name: StartDate Created by: Discovery Label: Data type: varchar Size: 10 Scale: Attribute type: Source Temporal Data Rating: Nullable: □ Unique: □ Primary key: □ Auto increment: □ Default value: Comment: Hidden: □ Subjects: General info Document general information:
2	Generating Data Vault	When generating the Data Vault, this additional MCR (ws3d_grv - Add Temporal Metadata to all Satellites) should be used to copy the fields to all relevant satellites  Generate data vault Apply model conversion Select a model conversion Model conversion: ws3d_grv - Logical to DV Spine (Template) ws3d_grv - Create satellites ws3d_grv - Hash Key generation (Template) ws3d_grv - Add Temporal Metadata to all Satellites (Template)

3	Generating Load and Stage	During the generation of Load and Stage the model conversion rule "ws3d_rvls - Remove Custom Attribute types" should be applied at the end  Generate load and stage tables Apply model conversion Select a model conversion Model conversion: ws3d_rvls - Define change hashes on satellite stages (Template) ws3d_rvls - Define extended properties for hub and link hash key (Template) ws3d_rvls - Merge and clean up stages (Template) ws3d_rvls - Create loads (Template) ws3d_rvls - Record tracking satellite - staging (Template) ws3d_rvls - Housekeeping (Template) cust_Housekeeping (Template) ws3d_rvls - Remove Custom Attribute types (Template) ☐ Create snapshots Prerequisite: Multi-Active Satellite Natural Key Attribute type needs to be an attribute type for Load and Stage. We are reusing this field for deploying to RED to allow for multiple versions of the data to be active based on the source temporality

WhereScape RED

No prerequisites are required for RED. The templates should handle the inserts into the Satellites if there is a Multi-active Natural key present. The loads will be aware of the additional keys and will allow multiple versions of the same business key.