Accessing External Databases

DBRET and DataViews

Many times we need to access **external databases** from our GeneXus applications.

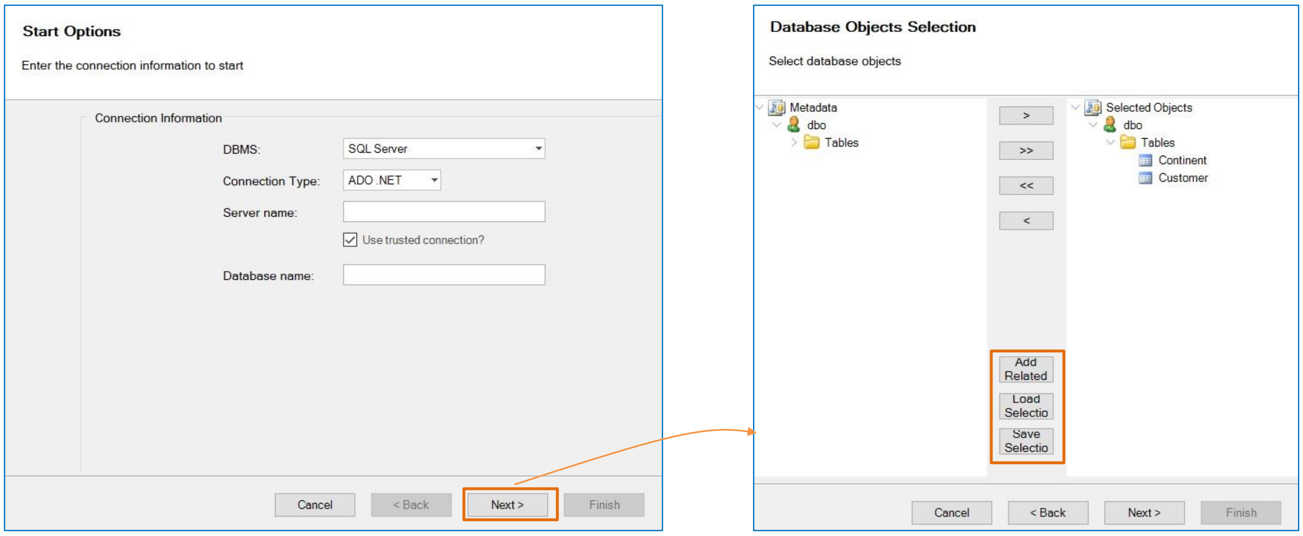
Accessing external databases

External database
accesses
GeneXus

For example, we may need to load data from an external database **to tables of our database associated with the knowledge base** to make an initial load… After that, we may no longer need to stay connected to that external database.

Or, we may also need to connect and stay **always** connected to a certain table or tables of one or more external databases… and not just to read them, but also to enter and change the data in them.
Let's see how all this can be done in GeneXus.

Database Reverse Engineering Tool (DBRET) - Wizard


Start Options

Enter the connection information to start

Connection Information

DBMS: SQL Server

Connection Type: ADO .NET

Server name:

☑ Use trusted connection?

Database name:

Database Objects Selection

Select database objects

Metadata
- dbo
  - Tables

Selected Objects
- dbo
  - Tables
    - Continent
    - Customer

In our Travel Agency knowledge base, let´s see in the menu: **Tools, Database Reverse Engineering option.**

Selecting this option opens a wizard that **allows us to apply reverse engineering to external databases. This means that for an existing database that contains tables, relationships, indexes, and so on, all the objects and configurations required for easy access will be created in GeneXus.**

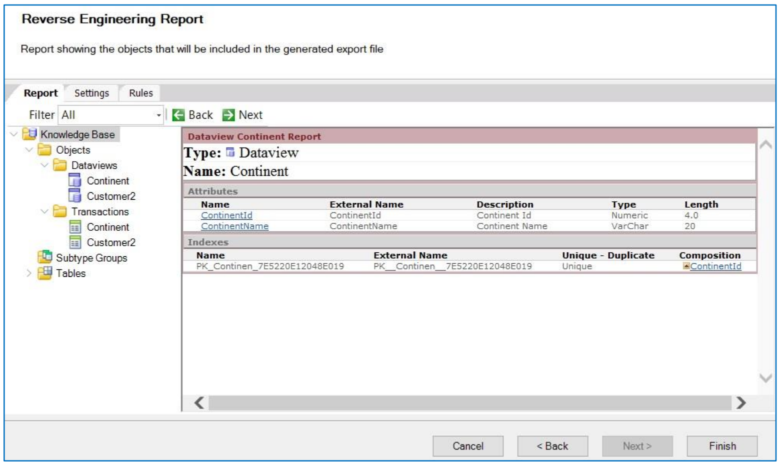
In this first page of the wizard, we need to enter the connection data to the database we want to access.
Because the database we want to access is precisely SQL Server, in the “DBMS” option we leave “SQL Server.”
The “Connection Type” option also remains as it is, and then we indicate the name of the server and the name of the database to which we want to connect. We click on the Next button and move on to the second page of the wizard...

The tables of the database to which we are connected are displayed here. We must make sure to leave in the right window only those tables to which we want to apply reverse engineering, in order to access them from our knowledge base.
We need to access the Customer and Continent tables, so they must be in the window on the right.
Before clicking on Next, let's take a look at these other buttons:

The “Add Related” button allows us to automatically add all the tables that have an N-1 relationship with the tables we selected.

In addition, the “Save Selection” button allows us to save the table selection we made, so that when running it again later on we can quickly load the same selection by clicking on the “Load Selection” button.

Database Reverse Engineering Tool (DBRET) - Wizard



Reverse Engineering Report

Report showing the objects that will be included in the generated export file

Dataview Continent Report

Type: Dataview

Name: Continent

Attributes

Name	External Name	Description	Type	Length
ContinentId	ContinentId	Continent Id	Numeric	4.0
ContinentName	ContinentName	Continent Name	VarChar	20

Indexes

Name	External Name	Unique - Duplicate	Composition
PK_Continen_7E5220E12048E019	PK__Continen__7E5220E12048E019	Unique	ContinentId

We click on Next:
The reverse engineering process begins in this moment… that is to say, the composition and everything related to the tables selected will be evaluated, and the GeneXus objects required will be created.

It has 3 tabs:
- Report
- Settings
- Rules

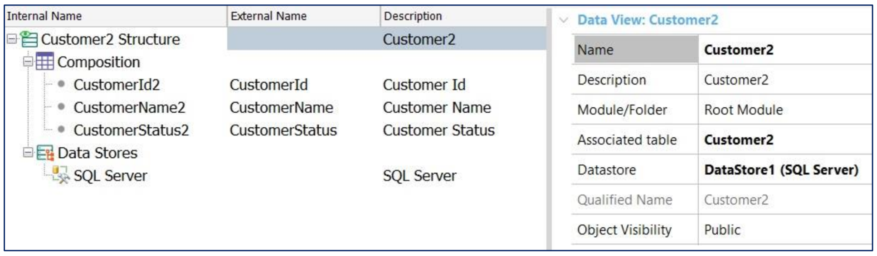
Let's begin by the main tab: Report. The report that we're viewing informs us about the objects created during the reverse engineering process.
Note that below the **Objects** folder there is a **Dataviews** folder and a **Transaction** folder. Below each one of them, there is an object called Continent and another one called Customer2.

The first question that comes to mind is why the name "Continent" was kept exactly the same as the name of the external table and why for "Customer," the name of the external table, the KB objects were named "Customer2"?
The reason is that our knowledge base already had a transaction called "Customer" and the corresponding table with the same name. It wasn't possible to create another transaction called "Customer," and that's why it was named "Customer2," just like the other objects created to access the Customer external table.

And… **what is a Data View object?** We haven't seen it so far.
A Data View object allows defining and configuring information from an external table. In other words, **for each external table that we need to access, there must be a Data View defined in the knowledge base.**

DataView Object

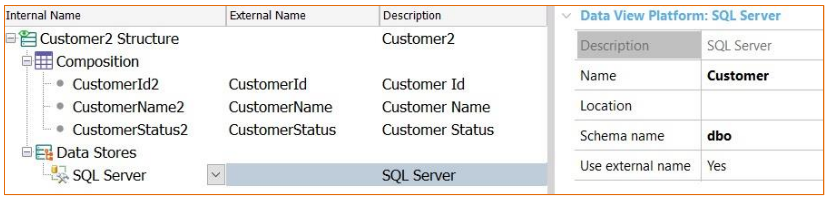
It allows defining and configuring information from an external table. For each external table that needs to be accessed, there must be a DataView in the knowledge base.



Internal Name	External Name	Description
Customer2 Structure		Customer2
Composition		
CustomerId2	CustomerId	Customer Id
CustomerName2	CustomerName	Customer Name
CustomerStatus2	CustomerStatus	Customer Status
Data Stores		
SQL Server		SQL Server

Data View: Customer2

Name	**Customer2**
Description	Customer2
Module/Folder	Root Module
Associated table	**Customer2**
Datastore	**DataStore1 (SQL Server)**
Qualified Name	Customer2
Object Visibility	Public



Internal Name	External Name	Description
Customer2 Structure		Customer2
Composition		
CustomerId2	CustomerId	Customer Id
CustomerName2	CustomerName	Customer Name
CustomerStatus2	CustomerStatus	Customer Status
Data Stores		
SQL Server		SQL Server

Data View Platform: SQL Server

Description	SQL Server
Name	**Customer**
Location	
Schema name	**dbo**
Use external name	Yes

Since we are in the middle of the reverse engineering process, so as not to cancel it we will look at this image and comment how the "Customer2" Data View will be defined.

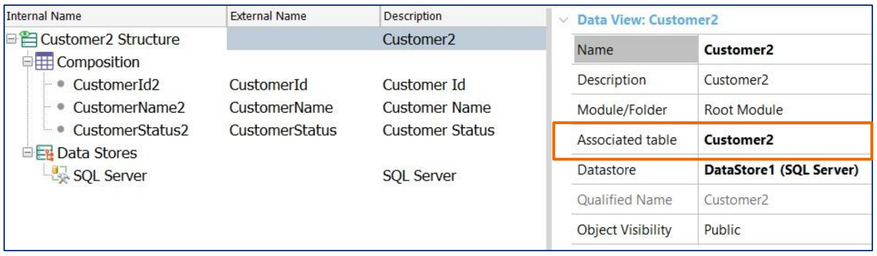
The first thing we see below Composition is a list of attribute names. Note that for each attribute there is an External Name **(the name of the physical field in the external table)** and an Internal Name **(the name given at the knowledge base level)**; that is to say, **the Internal Name is the name of the attribute included in the transaction automatically created to access the external table in an interactive manner**, and it is the name that will be included in all the GeneXus objects, such as grids, events, and so on.

Below Composition is the **mapping** of names of the table's external fields and their corresponding internal attribute names using GeneXus' naming convention.
In our knowledge base, we will always make reference to internal names, and when generating the code in the corresponding language, GeneXus will make reference to the physical field names, taking into account the name correspondence defined in the Data View.
Where is the external table name, for GeneXus to make reference to the table when generating the code?

Here we see that the external platform is SQL Server and looking at its related properties, we see the **Name** property that has the name of the external table.

DataView Object

Associated table: Its value is the name of the transaction associated with the DataView.



Internal Name	External Name	Description
Customer2 Structure		Customer2
Composition		
CustomerId2	CustomerId	Customer Id
CustomerName2	CustomerName	Customer Name
CustomerStatus2	CustomerStatus	Customer Status
Data Stores		
SQL Server		SQL Server

Data View: Customer2

Name	Customer2
Description	Customer2
Module/Folder	Root Module
Associated table	Customer2
Datastore	DataStore1 (SQL Server)
Qualified Name	Customer2
Object Visibility	Public

Where is the information of the **external database in which this table is located**?

If we look again at the properties of the main node of the Data View, we can see the **Data Store property, which points to DataStore1, automatically created in the reverse engineering process.** Even though at the end of this video we will see where the Data Stores are defined in a knowledge base, for now the concepts that we have to learn are as follows:

Each Data View has the information of an external table and each Data Store has the access information to a Database.
Next, for each Data View we need to make reference to the Data Store it belongs to.

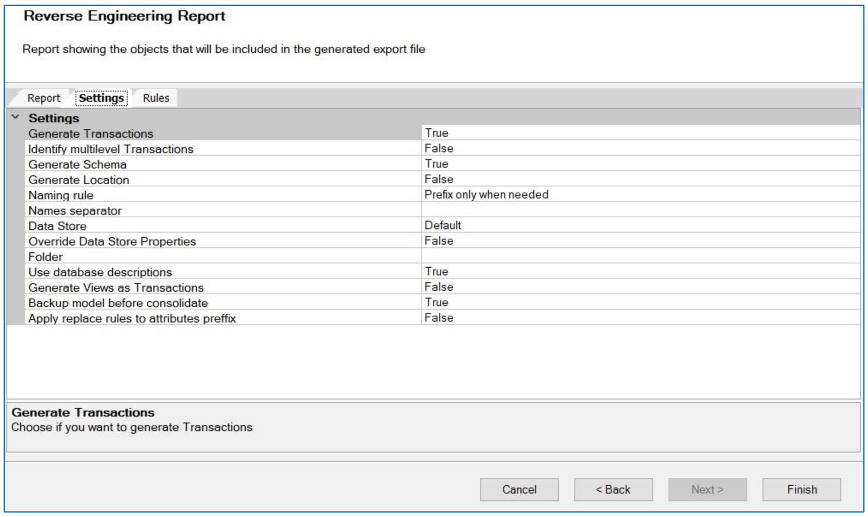
Now we will take a look at the Data View property called "Associated **table.**"

Its value is the name of the **transaction associated with the Data View**; that is to say, the transaction that was also automatically created with the same name as the Data View, so as to allow making additions, changes and deletions in the external table in an interactive manner.

Transactions associated with Data Views don't cause the creation of tables or reorganizations. They are not different from the other transactions, but when they are referenced in Data Views as Associated tables, GeneXus understands that it will only have to generate the form and its programming. The purpose is to allow interaction with the external table whose information is stored in the Data View in question.
It should be pointed out that transactions associated with Data Views can have, at most, the same number of attributes as the external table. They can have fewer attributes if some fields are not to be accessed… or, at most, the same number as the external table. But they can't have more because GeneXus doesn't reorganize external tables.

Note that if any errors had occurred during this reverse engineering process, they would have been notified.

Going back to the Wizard...



Reverse Engineering Report

Report showing the objects that will be included in the generated export file

Settings

Generate Transactions	True
Identify multilevel Transactions	False
Generate Schema	True
Generate Location	False
Naming rule	Prefix only when needed
Names separator	
Data Store	Default
Override Data Store Properties	False
Folder	
Use database descriptions	True
Generate Views as Transactions	False
Backup model before consolidate	True
Apply replace rules to attributes preffix	False

Generate Transactions
Choose if you want to generate Transactions

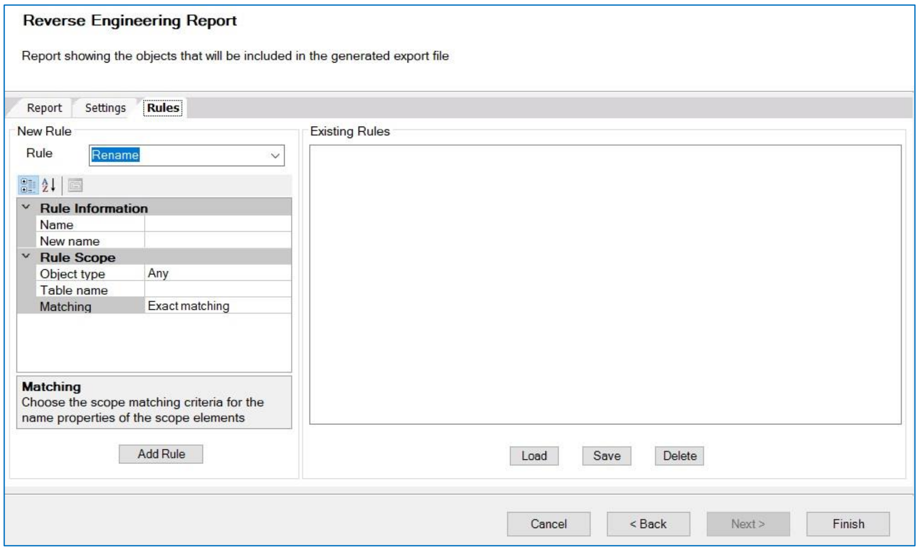
OK. Now let's continue with the other tabs offered in this page of the wizard.
The **Settings** tab, as its name suggests, allows making certain configurations.

For example "Generate Transactions" which is set to True by default, and as described here, allows us to indicate that transactions are created to handle data from external tables interactively. If we change the value to False, only Data Views are created, not transactions.

The option "Identify **Multilevel** Transactions" also as its name suggests, allows us to set whether to identify subordination relationships, and to create transactions with more than one level as long as patterns that make it possible are identified. Its default value is False. As a result, every table to be imported defines a single transaction with a single level and the related Data View.

The option "Generate Schema" is set to True by default, and this means that the table schema data is stored in the Schema property of the corresponding Data View.

Going back to the Wizard…


Reverse Engineering Report

Report showing the objects that will be included in the generated export file

New Rule

Rule: Rename

Rule Information	
Name	
New name	
Rule Scope	
Object type	Any
Table name	
Matching	Exact matching

Matching
Choose the scope matching criteria for the name properties of the scope elements

We are not going to continue looking at each of the possible configurations, since by clicking on them the necessary help is provided, so let's continue with the Rules tab.

Since objects and attributes are automatically generated from an external database, some name conflicts may occur.

For this reason, the Rules tab makes it possible to define name mapping rules to "solve" these ambiguities.
Therefore, this tab allows us to state rules to rename attributes, objects, etc., as well as change data types and make any necessary adjustments before generating the objects.

In our example we don't need to state rules of this type, but it is very intuitive to use, as in general all editors are.

This is the last step of the wizard. We click on Finish.

Impact analysis report

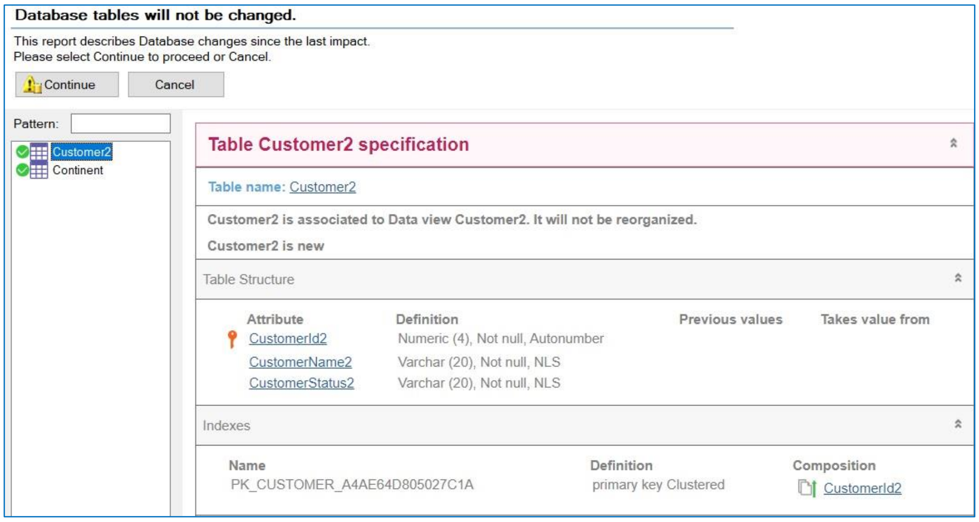

Database tables will not be changed.
This report describes Database changes since the last impact.
Please select Continue to proceed or Cancel.

Table Customer2 specification

Table name: Customer2

Customer2 is associated to Data view Customer2. It will not be reorganized.

Customer2 is new

Table Structure

Attribute	Definition	Previous values	Takes value from
CustomerId2	Numeric (4), Not null, Autonumber		
CustomerName2	Varchar (20), Not null, NLS		
CustomerStatus2	Varchar (20), Not null, NLS		

Indexes

Name	Definition	Composition
PK_CUSTOMER_A4AE64D805027C1A	primary key Clustered	CustomerId2

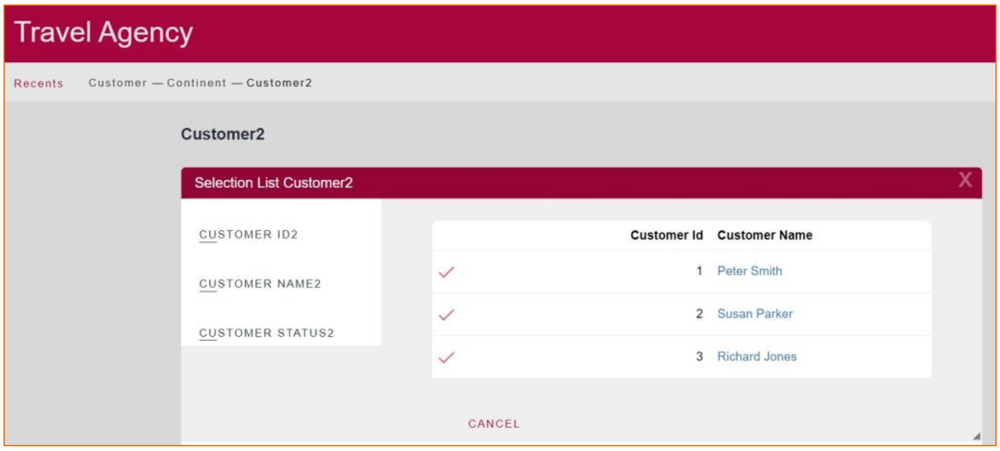
Let's see what it has generated:
The Data View Continent and the Data View Customer2 , as well as the corresponding Continent and Customer2 transactions.

OK. We press F5.

In the impact analysis report, we can verify that the tables associated with the Data Views will not be reorganized.

Let's continue…

At runtime...


Travel Agency

Recents Customer — Continent — Customer2

Customer2

Selection List Customer2

CUSTOMER ID2

CUSTOMER NAME2

CUSTOMER STATUS2

	Customer Id	Customer Name
✓	1	Peter Smith
✓	2	Susan Parker
✓	3	Richard Jones

CANCEL

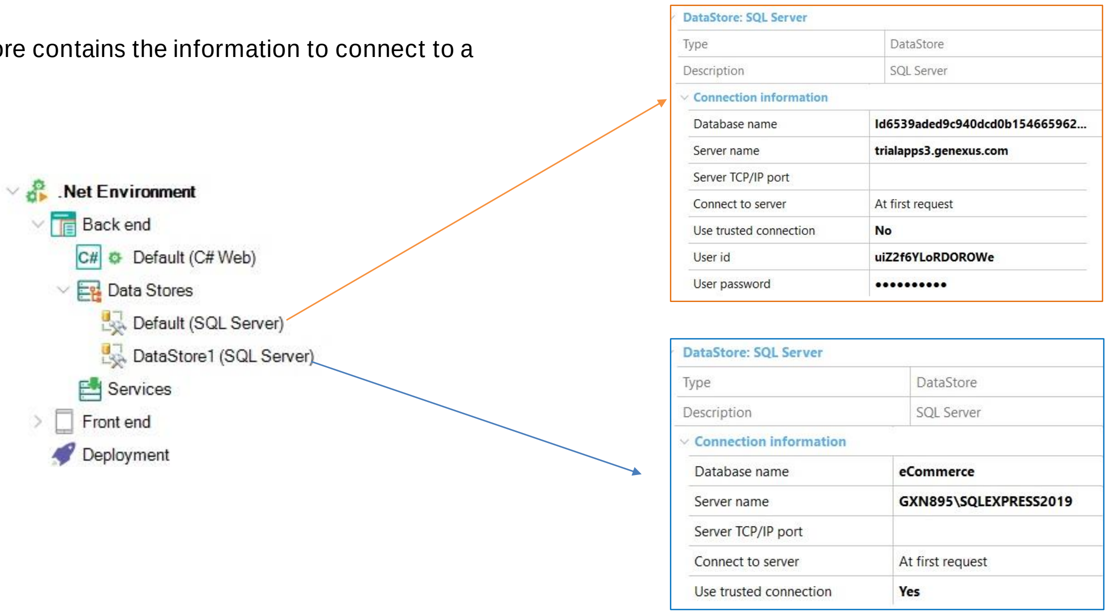
Now we will open the Customer2 transaction. Remember that through the Customer2 transaction we access the CUSTOMER table of the external database.

We can see the records that have already been entered. For example, we enter a new customer, and see that the record was entered successfully.

We can also modify the status of another customer, and we can even delete the customer we have just entered. We see, then, that we are actually working directly on the CUSTOMER table through the Customer2 transaction created by the reverse engineering process.

Data Stores

A Data Store contains the information to connect to a database.



DataStore: SQL Server

Type	DataStore
Description	SQL Server
Connection information	
Database name	Id6539aded9c940dcd0b154665962...
Server name	trialapps3.genexus.com
Server TCP/IP port	
Connect to server	At first request
Use trusted connection	No
User id	uiZ2f6YLoRDOROWe
User password	••••••••••

DataStore: SQL Server

Type	DataStore
Description	SQL Server
Connection information	
Database name	eCommerce
Server name	GXN895\SQLEXPRESS2019
Server TCP/IP port	
Connect to server	At first request
Use trusted connection	Yes

Lastly, let's talk about Data Stores. In the Preferences view, below the .Net Environment node, which is our execution environment, we can see the DataStores below the Backend node.

What do Default Data Store and DataStore1 correspond to?

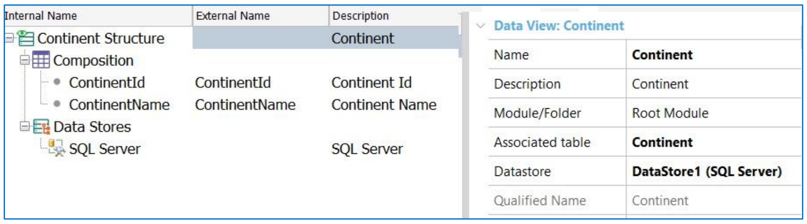
As we've said before, **a Data Store contains the information to connect to a database**.
When we press F5 for the first time in a knowledge base and prototype locally, the information related to the database is requested, and a **Default Data Store** is created with this information.

If we look at its properties, we can see that it connects to the database created at the time. It corresponds to **the database related to our knowledge base** and within it the tables that GeneXus determines are created, analyzing our defined transactions.

Which database will DataStore1 connect to? With the external eCommerce database to which we have connected.

Accessing external information

Continent DataView



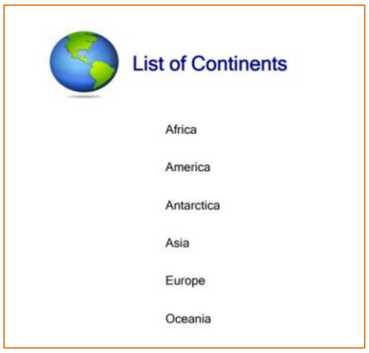
Internal Name	External Name	Description
Continent Structure		Continent
Composition		
ContinentId	ContinentId	Continent Id
ContinentName	ContinentName	Continent Name
Data Stores		
SQL Server		SQL Server

Data View: Continent

Name	**Continent**
Description	Continent
Module/Folder	Root Module
Associated table	**Continent**
Datastore	**DataStore1 (SQL Server)**
Qualified Name	Continent

```
Print Title

For each Continent order ContinentName
     Print ContinentDetail
Endfor
```



List of Continents

Africa

America

Antarctica

Asia

Europe

Oceania

OK. Finally, if we look at the properties of the DataViews Continent and Customer2 we see that they point to the DataStore1 created automatically by GeneXus in this reverse engineering process.

To finish and to see one more example running, we have created a very simple list with the list of continents.
Note in the source of this procedure that we are making reference to the Continent base transaction, sorting by the ContinentName attribute.

To run it, we right-click and select Run.

training.genexus.com
wiki.genexus.com
training.genexus.com/certifications