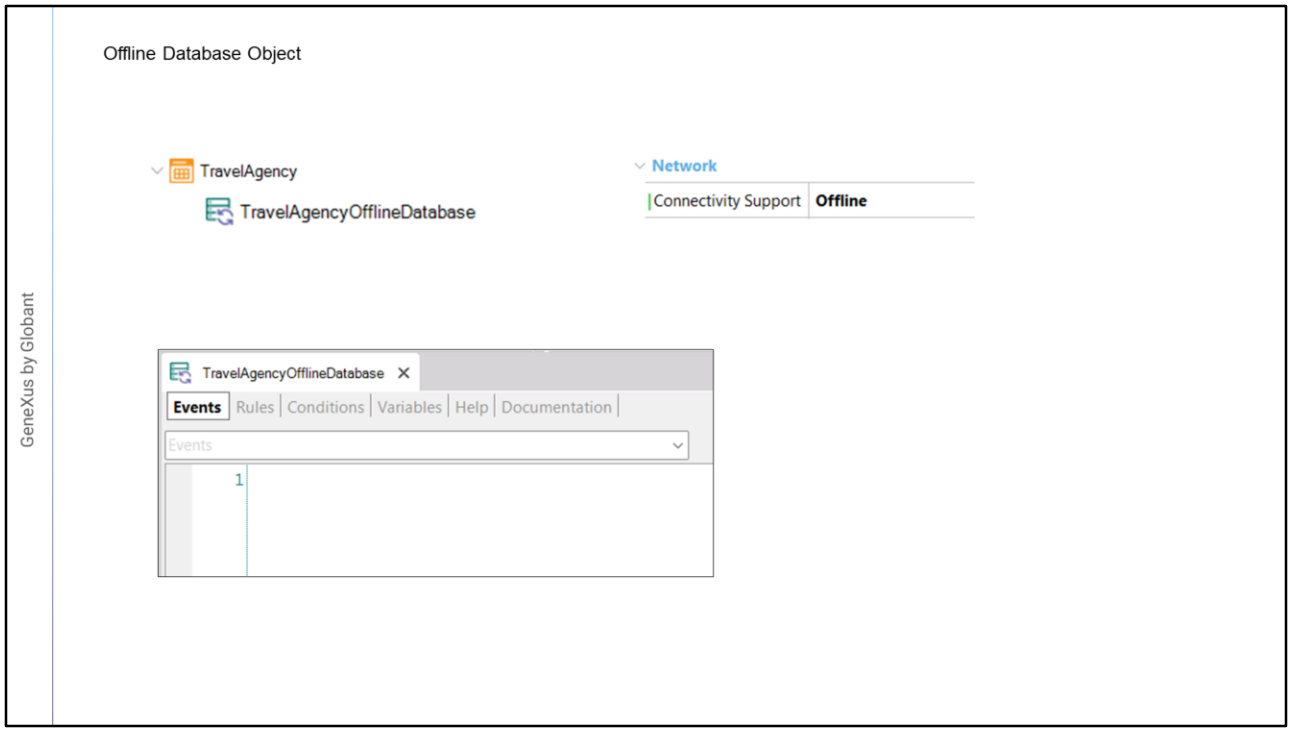

GeneXus by Globant
GeneXus™
by Globant
training.genexus.com

GeneXus by Globant

Offline Database Synchronization

Diego Marranghello

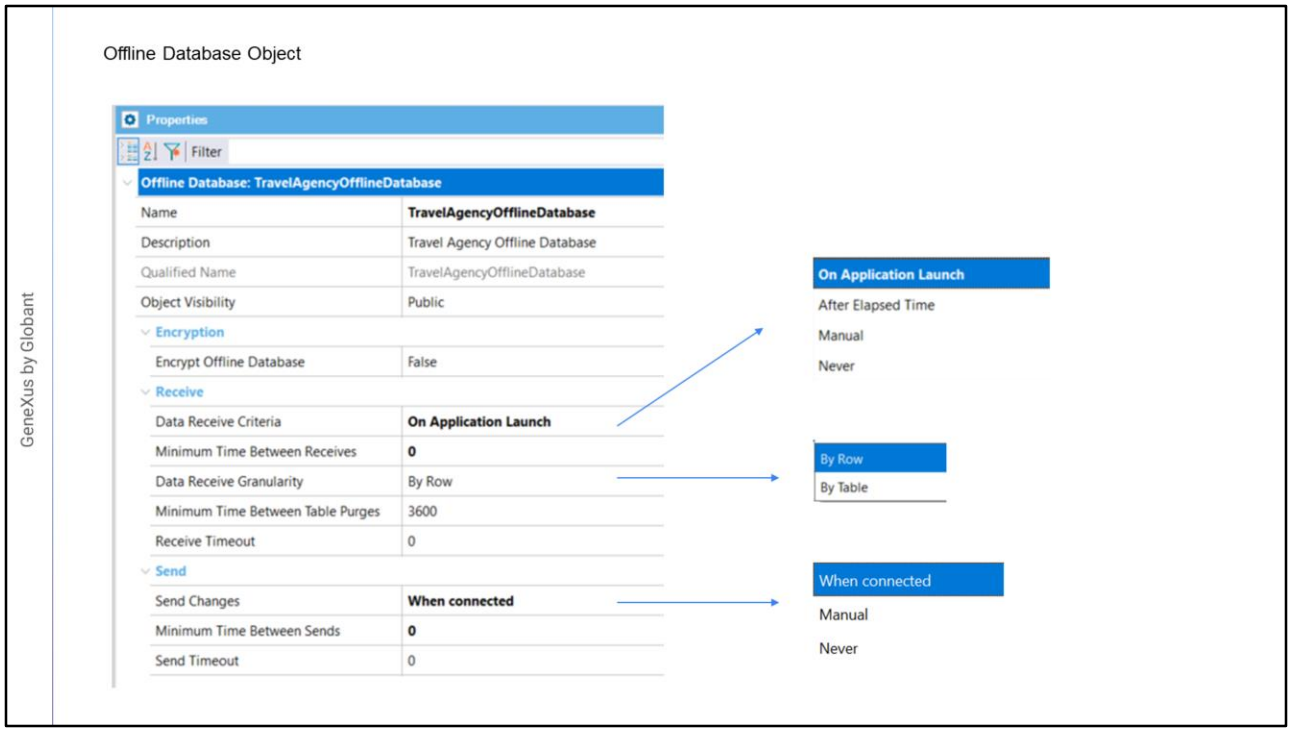


For every main object that has the Connectivity Support property set to Offline, an object called Offline Database is created.

As it is created, we'll be able to see the tables that will be added to the local database. The programs to create the local database are also created in the device's native language.

This object is responsible for determining when the synchronization is run, and which data is of interest when synchronizing with the server tables.

The Offline Database object also has events and conditions to determine its behavior.



Let's see some of the most important properties of the Offline Database object.

In the Receive group we have the “Data Receive Criteria” property, which will be used to receive data. The available values are as follows:

- When the application is started. This is the default value and indicates that the app will start receiving data after startup.

- With the value After Elapsed Time, data will be received when the time indicated in the Minimum Time Between Receives property –expressed in seconds– has elapsed. Also, it is made when the application is started.

- The Manual value indicates that data will only be received manually; here, an action must be performed to start receiving it.

- Lastly, the Never value indicates that data will never be received by default. GeneXus will not generate the necessary synchronization programs, which must be manually implemented if necessary.

As we had already seen, the “Data Receive Granularity” property allows us to establish the data reception mechanism, which can be by Row –the default value– or by Table.

Then, the Send Changes property will allow us to establish when we want to send the data from the device to the server. The available values are as follows:

- When Connected –this is the default value– indicates that the data will be sent immediately after detecting a connection.

- The Manual value indicates that data will only be sent manually; here, an action must be performed to start receiving it.

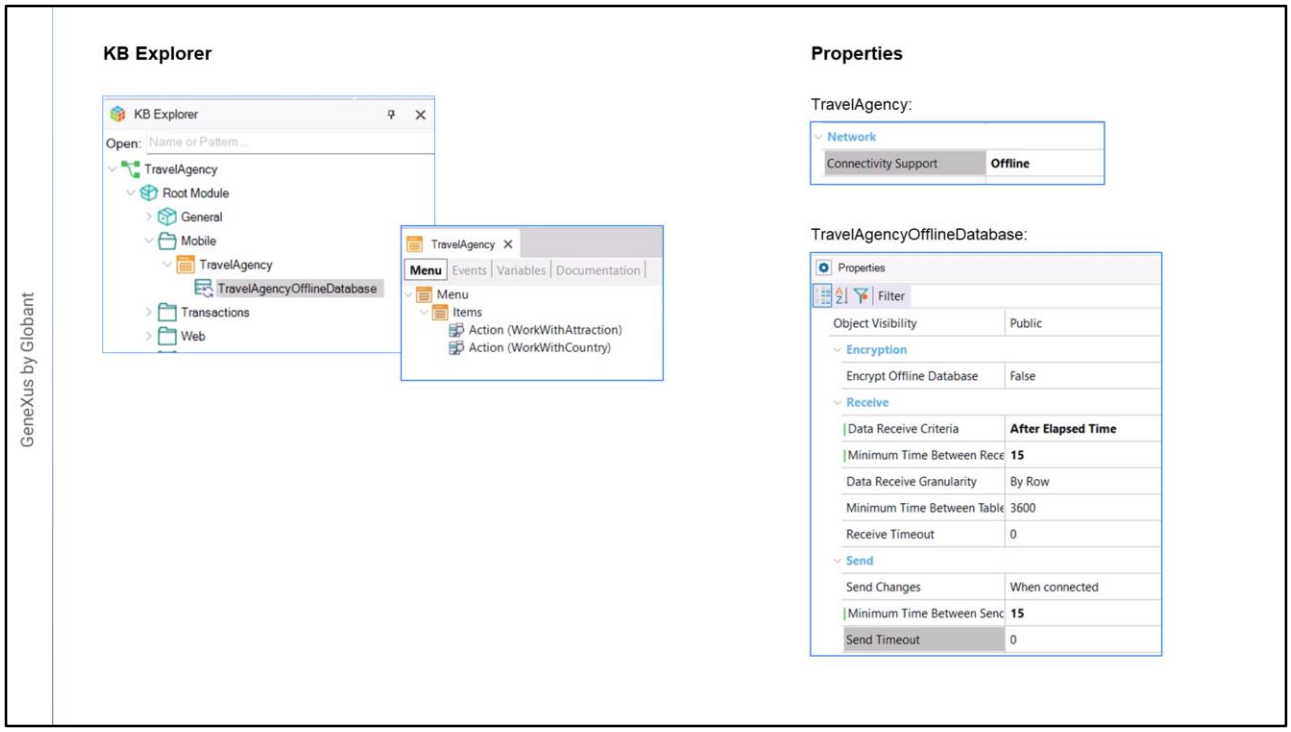
- The value Never indicates that data will not be automatically sent by GeneXus. The modified records will not be stored in the auxiliary table GXPendingEvent. So, if data needs to be sent, the developer is responsible for programming all the required logic.

If the Manual value is used to send or receive data or to provide more features to the user, we need to develop actions that perform the synchronization.
To this end, the Synchronization API is used.

This API is not found in the References like the rest of the APIs but is part of the grammar.
It has the Send and Receive methods for synchronization, ServerStatus to determine the server's status, and ResetOfflineDatabase to return the local database to its initial status, whether by running a Create Database operation to empty the tables or using a preloaded database.

With this API we could, for example, develop options in a panel for the users to trigger the synchronization when they want it.
Here we can see a very simple example where the two methods are used in the same event.

Let's see an example in GeneXus.



We have created part of an application for a travel agency, with a Menu object and WorkWith objects for Attraction and Country as items.

First, we'll configure the Offline Database object for the synchronization to be made every 15 seconds.

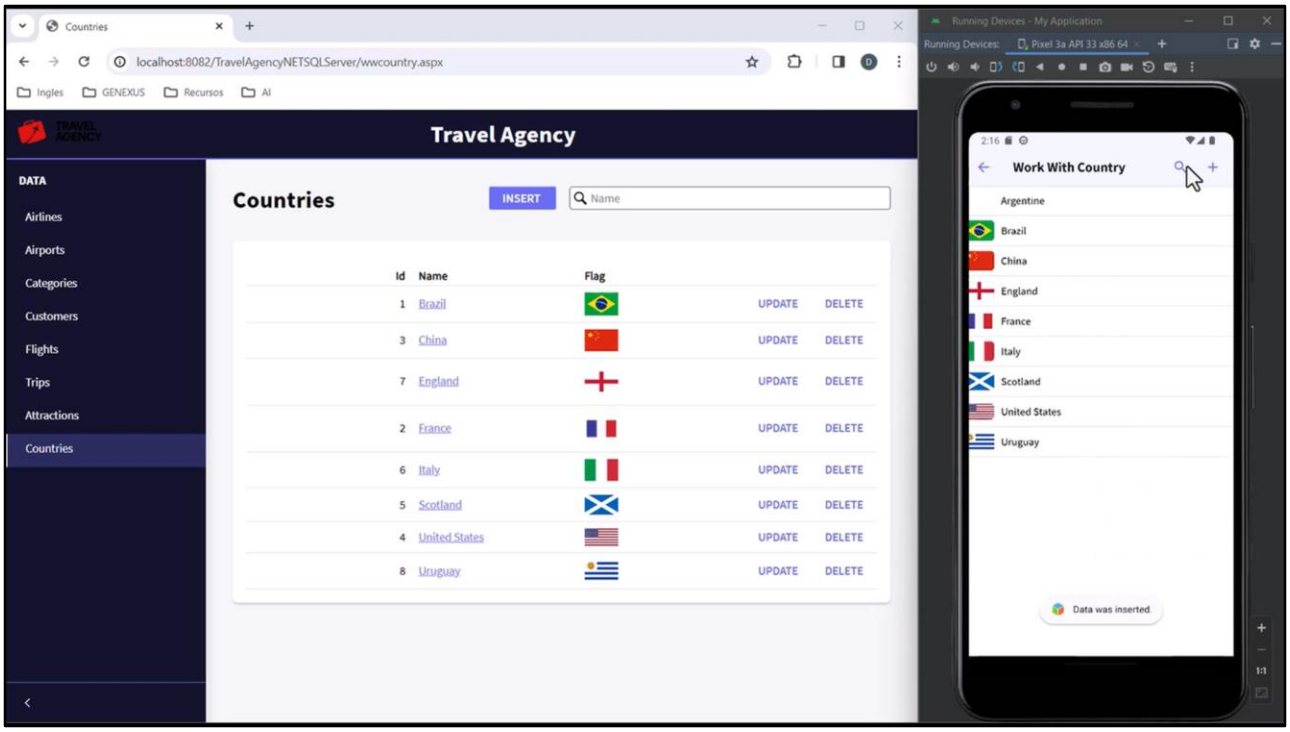
We open the TravelAgencyOfflineDatabase object, which was created by setting the Connectivity Support property to Offline.

We go to the properties and in the Data Receive Criteria property we enter the value After Elapsed Time; in Minimum Time Between Receives we enter the value 15.

Now in the Send group, in the Minimum Time Between Sends property we'll also enter the value 15.

With this configuration, the minimum time between sending/receiving data in our application will be 15 seconds.

Let's try this. We save and do a rebuild all of the application.



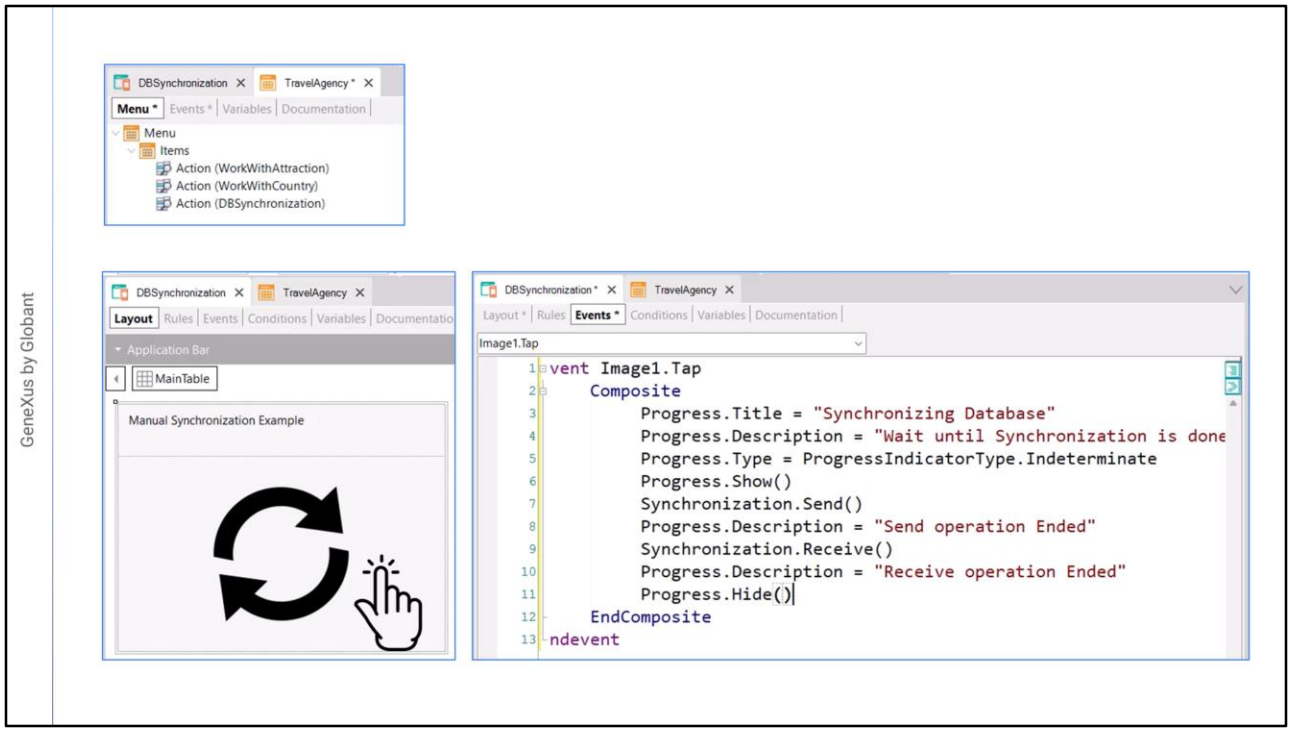
OK, the changes are made.

We will run it on mobile and web, accessing the Country Work With in both cases.
Let's create a new country from mobile.

When refreshing on the web, the information is immediately updated with the country that has just been entered, because the first send is automatic. But if before 15 seconds have elapsed we enter another record, when we refresh the web screen it still doesn't appear. This is because the 15 seconds that we configured have not passed between one entry and another. Since this period has not elapsed, we will have to wait this time for the synchronization.

We are going to refresh the panel and the registered country will appear.

Now we will enter the first country created and delete it. We confirm.
We go to web and refresh. Since less than 15 seconds have passed between the last data entry and the deletion, we will have to wait that time after the deletion for the synchronization to be made. That's it, the information is now updated in the server.



Now we are going to program a panel that allows us to perform the synchronization when the user wants it, with no need to wait for the established time.
I have already created a Panel, called DBSynchronization, which for now has only an image and a text.

We add it to the TravelAgency menu and save.

We want the synchronization to be performed when the user taps on the image, so we are going to program the Tap event.

In this event, since it is on the client side, we are going to use composite. Remember that synchronization will always be started on the client side.

So, we type Synchronization.Send(), which will send the data from the device to the server, and Syncrhronization.Receive() to do the receiving.

We are also going to give the user some feedback about the action they are performing; for this we are going to use “Progress”.
Progress is an external object that uses a panel to show what’s going on.
As a title, we enter “Synchronizing Database”.
Also, we'll enter a description in Progress.Description.

We have to indicate the type of progress we are going to use. We will use indeterminate, which is used when we don't know or can't indicate the degree of progress of the

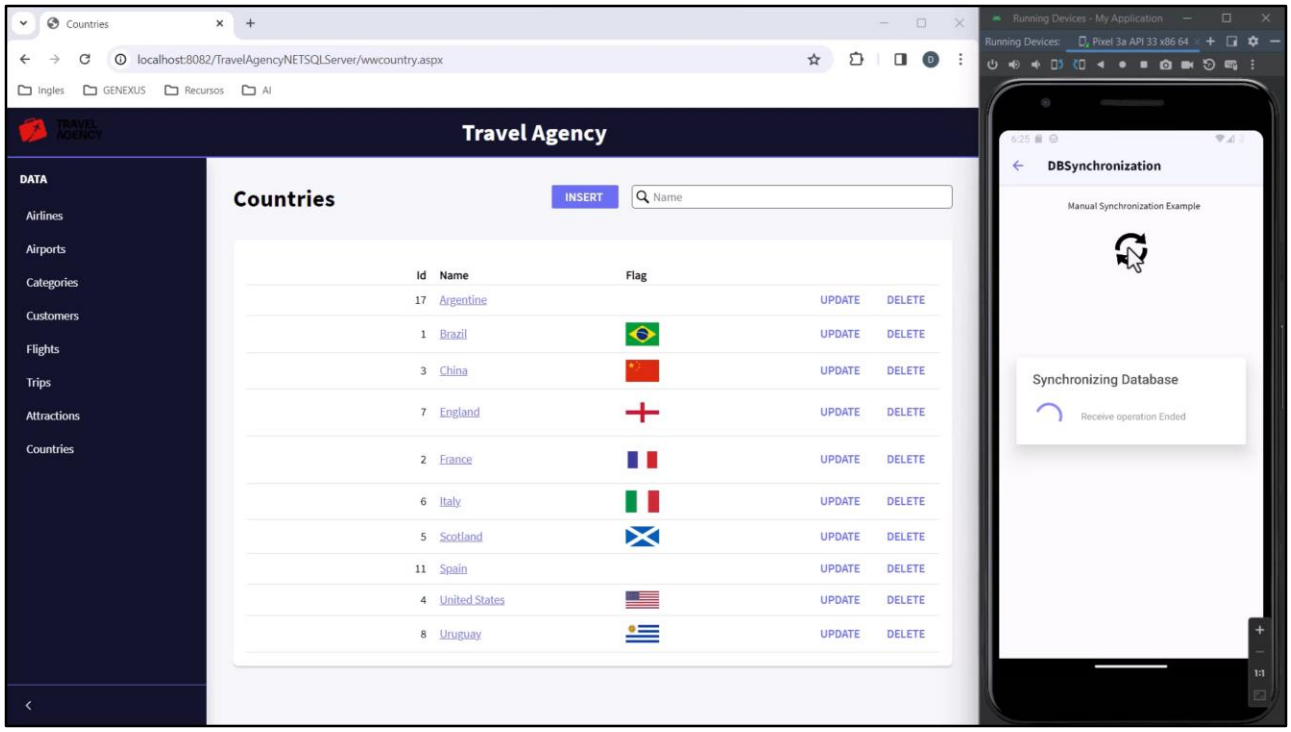
process.

We show the control with Progress.Show().

Next, we perform the Send. When it is finished we could change the description, so we will enter a text in Progress.Description.

Then we do the Receive, and again we change the description to indicate that it is complete.

Finally, we have to hide the control, so we write Progress.Hide(), which is the method that hides it.



We run the application.

Good. We already have the application in the emulator and for the web.

Let's add a country again from the mobile application and confirm.
If we update in the web application, we immediately see the synchronization because the first send is automatic, but if we add another country right away and go to the web application, it is not displayed.
We are not going to wait 15 seconds for it to synchronize, we are going to do it manually.
We tap on the icon, which indicates that the send is finished.
And now we can see the country entered.

Now we are going to delete it from the device and confirm. We are not going to wait; we go to synchronization, tap, and now we see the changes as soon as we refresh.

In this small demo we saw how to configure some properties of the Offline Database object. Also, we saw how to use the Synchronization API to run Send and Receive operations at the user's request.


GeneXus by Globant
GeneXus™
by Globant
training.genexus.com