Database Update

Using Business Components

GeneXus



In the previous video, we were given a detailed overview of familiar concepts to pave the way for understanding and using business components.

From the transaction structure with its logic (and by logic we mean controls for duplicates –not only primary key, but also candidate keys–, referential integrity, most of its rules and some of its events), a kind of data type similar to an SDT, but much more powerful, is obtained.

Then, it will be enough to define in almost any program a variable of that data type and manipulate it, which is what we will see next.

This variable, in its structure, will be handled similarly to an SDT. But it will also offer methods for doing something specific to a transaction: loading from the database, inserting, modifying, deleting; all this by executing the logic of the transaction, and then obtaining the generated messages and the results of success or failure.

The way to obtain that kind of data in the knowledge base is simply by turning on the Business Component property of the transaction.



We will start by creating a simple procedure that we will modify to implement a more complete requirement later. Let's not pay attention to the name right now.

We'll start with something very simple: suppose we want to insert a new tourist attraction.



The first thing we must do is create the Business Component of the Attraction transaction, so that it can be used in any object in the knowledge base. Therefore, we go to the transaction and among its properties find the one called Business Component. It is set to False by default. We change it to True. At first, we don't see any effect. However, if we save... we see this in the navigation listing.



If now we go to the procedure and define in it a variable named Attraction, we see that because we selected a name matching that of the transaction, a data type of that name has been automatically used. What data type is that?

If we select the variable properties and open the combo... there is a Business Component group that for now only offers one value: not coincidentally, it is Attraction. Here's where we see the effect of having enabled the transaction property. The data type Business Component Attraction was created in the KB; it is available and can now be assigned to any variable.



Having defined this variable in the procedure, memory space is automatically reserved to load all its elements as an SDT.

So, if we insert the variable and type a period, in this window we see the names of all the transaction's attributes, among other things, to use them in the structure, for example, by assigning them a value. In this way, we could...

TravelAgency - GeneXus Trial													— i	3 ×
File Edit View Layout Insert Build Knowledg	ge Manager W	/indow Tools Test Help												
) I III III X III () への, 器 III	😹 🕨 .Net En	vironment - Release	* =											
🞯 KB Explorer 🛛 🕫	× 🔯 Start i	Page 🗙 🔄 Attraction 🗙	InsertCateg	orvUpdateAttractions	• × 🖪 Navig	ation View X				\sim \circ	Properties			4
Open: Name or Pattern	Source *	Lavout Rules Conditio	ns Variables *							2.55	Filter			
✓ TavelAgency	A		ina i vanabica i i		_					2 2 2 4	a Politica			
- Root Module	Subroutine	es			~									
> CeneXus		1												
> El Airline		2 &attraction.Attra	ctionName = '	CountryId Co	untryName = "	(hina")				2				
> 🔄 Airport		4 &attraction.CityI	d⊇= find(Cit	yId, CityName	= "Beijing")				^				
> T Attraction	1	5 &attraction.Categ	oryId = find(CategoryId,	CategoryName	= "Monument")								
AttractionList														
AttractionsByName		attraction.1	aul a			-								
AttractionsPerCategory		From X	mIFile			&attr	raction.Insert	0	AttractionId	AttractionNar	e Countryld	CityId	Categoryld	
AttractionsReport		- GetMe	ssages						1	Louvre Museur	n z	1	1	
> In AttractionWithoutParameters		🕀 GetMe	tadata						2	The Great Wal	(a.	1.1	1	-
CategoriesAndTheirAttractionsList		- GetVa	ue						3	Eilfel Tower	2	1	2	-
> 🥅 Category		- Insert	vi la data	Insert(): Boolean					4	Christ the Rodem	ner t	1	2	-
CountriesRanking		- Inserto	ropoate						s	Smithsonian Insti	ute 4	2	1	-
CountriesWithMoreThan2Attractions		- Mode							6	Matase Museu				-
> 🧾 Country		- Save	~							Forbidden city				
> 🌉 Customer														
🔚 DataProviderCountries		&attraction												
🖧 Diagram 1		AttractionId												
💾 Diagram2		AttractionName	Forbidden city		AttractionId	AttractionName	Countryld	CityId	Categoryld .			+		
Lingram3		Countryld	3		1	Louvre Museum	2	1	1 .		AttractionId	7		
💑 Diagram4		CountryName			2	The Great Wall	3	1	2 .		AttractionName	Forbic	dden city	-
Liagram5		Cityld	1		3	Eiffel Tower	2	1	2 .		Countrold	2	addition	-
EnterAttractionsFilter		CityName			4	Christ the Redemme	ar 1	1	2 .		CountryId	Ohren		-
> 📺 Flight		Categoryld	2		5	Smithsonian Institut	ie 4	2	1 .		CountryName	- Contina		-
A FlightArrivalAirport		AttractionPhoto	-	_	6	Matisse Museum	2	2	1	-	Cityid	1		_
A FlightDepartureAirport		AttractionAddres	5								CityName	Beljin	g	
Gx00A0			-								Categoryld	2		
Gx00B1											CategoryName	Monu	ment	
Gx0010											AttractionPhoto			
Gx0020											AttractionAddres	s		
Gx0030														
Gx0040														
Gx0051	12									× 1		T	Contains 15	Tasta Dasa
Gx0060	↓ □ <										ropentes 100000	CQ Tests	Explorer 3=	Tests Hesu
C\Models\TrialTrunk\Travel.Agency											Ln 7 Col 1	4 Ch 14	1	INS

To insert the Forbidden City tourist attraction, as we saw before when we used the transaction, we will have to complete the data.

We don't need to assign a value to the identifier, because it will be autonumbered.

The name of the attraction will be... we know it's in China, it has the identifier 3, but what if we're wrong? We'd better look for that identifier with the Find formula, because one thing we are sure of is the name of the country, China.

That name is an inferred attribute in the transaction, so we don't have to assign a value to it here to insert the attraction.

We know the city is Beijing, so we find its identifier and assign it to the business component element and also remove the inferred element. We assign a value for the category ID, looking for the one named "Monument."

And we will not assign any value to the photo and address.

So, the variable is loaded with the data of the attraction we want to insert in the table. All we have to do is insert it. To do so, we have the Insert method of the Business component variable, which, as we can see, will return a Boolean value: True if it could insert it; False otherwise. We can just invoke it and it will try to insert a new record exactly as the transaction does, that is, executing all the rules and validations. If it succeeds, then the variable will be loaded with the ID given by the database and all of its own and inferred attributes.



However, we should be aware that although the record is already in the table, it is in an unstable state. If there is a power outage or system crash for any other reason, that record will be gone when the database is reestablished. This is because databases allow us to insert, edit and delete records as if at a logical level; whenever it seems convenient, we must indicate that all those operations we have been doing can be accepted. This is done with the Commit command. Therefore, if the insertion was successful, we commit. This is the action of making a commit on the database; that is, indicating that these operations should be fixed –made permanent– in the database. When a Commit is executed, all operations performed between the previous and the current commit will be accepted.

Just as we have the Commit operation to accept all those actions, the opposite one, called Rollback, allows us to undo everything that may have been done after the last Commit. Here we could make a Rollback if the operation was not successful... but it will not make much sense, because if the operation was not successful, we can assume that the record was not even inserted, so there will be nothing to undo.

To quickly run this procedure from the Developer Menu, we set it as main object and with HTTP invocation protocol. We press F5.

			Application	Name				97	
Attractions				INSERT	Q Name)			
	Id B	Name	Country Name	Category Name	Photo	City Name	Address		
	1	Louvre Museum	France	Museum		Paris		UPDATE	DELET
	2	The Great Wall	China	Monument	Stree-	Beijing		UPDATE	DELET
	3	Eiffel Tower	France	Monument	A	Paris		UPDATE	DELET
	4	Christ the Redeemer	Brazil	Monument	and a	Rio de Janeiro		UPDATE	DELET
	5	Smithsonian Institute	United State	Museum		Washington		UPDATE	DELET
	6	Matisse Museum	France	Museum		Nice		UPDATE	DELET
	8	Forbidden city	China	Monument		Beijing		UPDATE	DELET

If we open the Work With Attractions element, we will see that we already had attraction 7 for the Forbidden City, which we had inserted earlier through the transaction. Let's delete it (the identifier with value 7 will be lost; it will not be given again to the next attraction entered, which will be 8, as we will see now if we run the procedure).

Since it has no output, we don't see anything in the browser, but if we go back to Work With Attractions... it has actually inserted the Forbidden City.

Let's delete it again, to run the procedure again, but this time we'll add another attraction besides that one. For example, Notre Dame Cathedral.



So, we should insert it here. We no longer need the information that the variable had. Actually, we need to at least clean it up, to make sure **nothing is left behind when we load the Cathedral's data. One way is to** ask for new memory for the variable, and it is done in this way...

OK. Now we have it insert the data.

Asking for new memory is important. If, for example, we had not assigned a value to the category identifier in order to leave it empty for the Notre Dame Cathedral, it would have been loaded with the value we assigned before. Therefore, it is always good practice to ask for memory before completing the data of a business component that we are going to insert. Even up here.

Finally, before running it, note that in this case we inserted the first attraction and we will commit if that insertion was successful. Let's do the same with the second one, and commit.



But we might want to commit only once, after we've inserted both attractions. Or we could even try to insert the second attraction only if the first one was successful. For example, in that case, it would be like this.

Here we will only commit when the first and the second insertions were successful. It could happen that the first one was successful and the second one was not. Then the record of the first one will be saved in the database, but it will not be committed. So, maybe in that case, when the second one is not successful, we don't want the first record to remain in the database; and then we would use the Rollback command.

Let's run it. F5.

			Application	Name					
Attractions				INSERT	Q Name]			
	Idy	Name	Country Name	Category Name	Photo	City Name	Address		
	1	Louvre Museum	France	Museum		Paris		UPDATE	DELETE
	2	Great Wall	China	Monument	(Server)	Beijing		UPDATE	DELETE
	3	Eiffel Tower	France	Monument		Paris		UPDATE	DELETE
	4	Christ the Redeemer	Brazil	Monument	and the second se	Rio de Janeiro		UPDATE	DELETE
	5	Smithsonian Institute	United State	Museum	and the second	Washington		UPDATE	DELETE
	6	Matisse Museum	France	Museum		Nice		UPDATE	DELETE
	9	Forbidden city	China	Monument		Beijing		UPDATE	DELETE
	10	Notre Dame Cathedral	France	Monument		Paris		UPDATE	DELETE

Let's see the attractions, run the procedure, and go back to Work With Attractions. Now we have 9 and 10.

What if now we wanted to change the category of the Chinese Wall so that it is no longer a "Monument" and becomes a "Famous Landmark," as well as change its name by removing "The"?

xpiorer 4 X 🔯	Start Page X 🛐 Attraction	× 🔰 InsertCategoryUpdate	eAttractions * X						×	Properties	à
ame or Pattern	urce * Layout Rules Condi	ions Variables								Eilter	
ivelAgency	in the										
Root Module	•		-						-		
GeneXus	0-14-0-1								1		
Et Airline	Gattraction								-		
Airport	AttractionId	2						-			
Attraction	AttractionNa	ne The Oreat Wall	AttractionId	AttractionName	Countryld	CityId	Categoryld	000			
AttractionList	Countryld	3	- 1	Louvre Museum	2	1	1				
AttractionsByName	CountryNam	China	2	The Great Wall	3		2	-			
AttractionsPerCategory	CityId	1	3	Eiffel Tower	2	1	.2				
AttractionsReport	CityName	Beeing	4	Christ the Redemmer	1	1	2				
AttractionWithoutParameters	Categoryld	2	5	Smithsonian Institute	4	2	1				
CategoriesAndTheirAttractionsList	CategoryNar	Monument	6	Matisse Museum	2	2	1				
Category	AttractionPh	to	9	Forbidden city	3	1	2				
* CountriesRanking	AttractionAd	iness	10	Notre Dame Cathedral	2	1	2				
Customer DataProviderCountries	21										
Customer Customer Datagram1 Datagram1 Datagram2 Diagram3 Diagram4 Diagram4 Plight Parther Flight A FlightNervalArport Flight Custome Cus	21 22 &attraction.Loa	1(2)									

Let's leave the previous code with comments so that these two attractions are not inserted again (if their identifiers weren't auto-numbered, the insertions would fail because there is a duplicate key, but this is not the case).

We need to make an Update, as we would do with the transaction. First of all, we need to load the variable with the Attraction values we want to change. We know its ID is 2. So we invoke the Load method of the variable by passing it the key, 2, as a parameter.

When running this, it will go to the table to find the record 2, and if so, it will load the variable BC with all values: its own, inferred ones and formulas, as it happens when executing the transaction.

XB Explorer 🛛 🗘 🗙	Start Page X	TAttraction ×	InsertCategoryUpdate/	Attractions * ×						$\mathbf{\vee}$	Properties	ą
Name or Pattern	Source * Layout	Rules Conditions	Variables								Filter	
TravelAgency	Subvestinas											
Root Module												
> CaneXus												
> 🛄 Airline	&att	raction								-		
> 🔣 Airport		AttractionId	2	1					-			
> 📷 Attraction		AttractionName	Crosset Marili	AttractionId	AttractionName	Countryld	CityId	Categoryld				
AttractionList		Auracuonname	OLERC WOR	1. J.	Louvre Museum	2	1	1				
AttractionsByName		Countryid	3	2	Great Wall	3		3				
AttractionsPerCategory		Countryiname	China	3	Eiffel Tower	2	1	2				
AttractionsReport		Cityid	1	4	Christ the Redemmer	01.1		2				
> 📰 AttractionWithoutParameters		CityName	Beijing	5	Smithsonian Institute	4	2	1				
CategoriesAndTheirAttractionsList		Categoryld	3	6	Matisso Museum	2	2	1				
> 🧾 Category		CategoryName	Famous Landmark	0	Forbidden oity	(3)	1	2				
Countries Ranking		AttractionPhoto				0						
Countrine MithMore Than 20th actions				10	Notre Dame Cathedral	1		2				
Country		AttractionAddress		10	Notre Dame Cathedral			2				
Country Country Country Country Country Country Country Depart	21 22 & att 23 & att 24 & att 25 26 & att	Attraction.Load(2) traction.Load(2) traction.Attracti fraction.Load(2) traction.Load(2)	ionName = "Great L Id = find(Catego 2 2 2 2	all" rryId, CategoryNa G if &attract 7 Commit endif	<pre>me = "Famous Landma cion.Update()</pre>	rk")		2				
Country C	21 22 & att 23 & att 24 & att 25 26 & att	Attraction.Load(2) traction.Load(2) traction.Attracti traction.Category traction.Update()	ionName = "Great L II = find(Catego 2 2 2 2	Aall" rryId, CategoryNa 6 if &attract 7 Commit 8 endif	<pre>me = "Famous Landma cion.Update()</pre>	rk")		2				

Of all these values, we need to change two: the name of the attraction... And its category...

All we have to do is have the variable update this data in the table. To do so, we have the Update method.

When invoking it, exactly as in the transaction, all the rules, formulas, candidate key uniqueness and referential integrity controls will be run; if everything went well, the record will be updated in the table. And, of course, the variable will be loaded with the current data. Again, we could ask for the result of the update to, for example, already perform a Commit.

Let's try it.

We run the procedure, and check the attractions. Note it's been modified.

								-	o >
			Applicatio	n Name					
Attractions				INSERT	Q Name				
	Id	Name	Country Name	Category Name	Photo	City Name	Address		
	4	Christ the Redeemer	Brazil	Monument		Rio de Janeiro		UPDATE	DELETE
	3	Eiffel Tower	France	Monument	A	Paris		UPDATE	DELETE
	9	Forbidden city	China	Monument		Beijing		UPDATE	DELETE
	-2	Great Wall	China	Famous Landmark		Beijing		UPDATE	DELETE
	1	Louvre Museum	Erance	Museum		Paris		UPDATE	DELETE
	6	Matisse Museum	France	Museum	and the second se	Nice		UPDATE	DELETE
	10	Notre Dame Cathedral	France	Monument		Paris		UPDATE	DELETE
	5	Smithsonian Institute	United State	Museum	And Designed	Washington		UPDATE	DELETE

To complete the operations, what if now we wanted to delete an attraction by code?

For example, delete the Great Wall.

Control Production Control Production Support Production Variable (Control Production) Variable (Variable (Variab	KB Explorer	a X 🐼 Start Pr	age X 🔽 Attraction	X linsertCategor	UpdateAttractions* X						Properties	ą
Image: Second	Name or Pattern _	Source *	Lavout Rules Condit	ions Variables							TH A Filter	
Nutration Nutration Attraction Attraction Attraction Attraction Control Attraction Attraction Attraction Control Attraction Control Attraction Control Attraction Control Attraction Control Control Attraction Control Attraction Control Attraction Control Attraction Control Attraction Attraction Control Attraction	TravelAgency	^	entres l'estim	ingents (Antoningens)							1 22 44 F 1 1 1 1 1 1 1 1	
○ Control Attaction Saturcoll	Root Module	Subroutines			~							
Image: Second Secon	GeneXus									3		
Image: Second Seco	> El Airline									2		
Imachanical Amachanical	> 📑 Airport											
AttractionsPlace AttractionsPlace AttractionsPlace AttractionsPlace AttractionsPlace Contry AttractionsPlace Co	> 📰 Attraction											
Attraction/Ame 2 Attraction/Baper Attraction/Baper Attraction/Baper Control Control C	AttractionList	&a1	ttraction									
Attraction/Barrel Cologony Attraction/Barrel Country/Mame Co	AttractionsByName											
AttractionName AttractionName Contryide	AttractionsPerCategory		AttractionId	2						_		
 American Without/Pearmeters Country Mare Matericon Load (2) Mattraction .Load (2) Sattract	AttractionsReport		AttractionName	Great Wall	Attractionic	AttractionName	Countryld	Cityld	Categoryld	***		
Concernment 0 <t< td=""><td>> 🛐 AttractionWithoutParameters</td><td></td><td>Countryld</td><td>3</td><td></td><td>Louvre Museum</td><td>2</td><td>1</td><td>1</td><td></td><td></td><td></td></t<>	> 🛐 AttractionWithoutParameters		Countryld	3		Louvre Museum	2	1	1			
Image: Second	CategoriesAndTheirAttractionsList		CountryName	China	- 2	Great Wall	3	-	3	_		
Contribus Weit/More TrancAttractions Controls Relationment 1 1 2 - Controls Weit/More TrancAttractions Controls Relationment 4 Control Relationment 4 2 - Controls Relations Controls Relationment 4 2 1 - Controls Relations Attraction/Address 0 Testaden Macuru 2 2 1 Data Provide/Controls Attraction/Address 0 Testaden Macuru 2 2 1 - Data Provide/Controls Attraction/Address 0 Testaden Macuru 2 1 2 - Disgrand Attraction/Address Attraction.Load(2) Noten Dame Caludelit 2 1 2 - Segrand Segrand Segrand Segrand 2 1 2 - Pipothymathymatic 30 Rattraction.Load(2) 30 Rattraction.Load(2) 31 Rattraction.Load(2) 33 Genoval Sector 31 Rattraction.Delete() 33 - - - - Seconol Sector <td< td=""><td>> 🧾 Category</td><td></td><td>Cityld</td><td>1</td><td>3</td><td>Eiffel Tower</td><td>2</td><td>1</td><td>2</td><td></td><td></td><td></td></td<>	> 🧾 Category		Cityld	1	3	Eiffel Tower	2	1	2			
Categoryid 3 Categoryid 3 <td>Countries Ranking</td> <td></td> <td>CityName</td> <td>Beijing</td> <td>4</td> <td>Christ the Redemmer</td> <td>1</td> <td>1</td> <td>2</td> <td></td> <td></td> <td></td>	Countries Ranking		CityName	Beijing	4	Christ the Redemmer	1	1	2			
Image: Second	CountriesWithMoreThan2Attractions		Categoryld	3	5	Smithsonian Institute	4	2	1			
Image: Sector of the sector	> 🧱 Country		CategoryName	Famous Landmark	6.	Matisso Museum	2	2	1			
AttractionAddress AttractionAddress AttractionAddress NonetDameCalhedad 2 1 2 AttractionAddress NonetDameCalhedad 30 8attraction.Load(2) 3 AttractionAddress NonetDameCalhedad 31 8attraction.Success() 33 Commit NonetDameCalhedad 34 endif 34 Commit NonetDameCalhedad NonetDameCalhedad NonetDameCalhedad <	> 🧾 Customer		AttractionPhoto		9	Forbickien city	3.	1	2			
A Dagami A	🔚 DataProviderCountries		AttractionAddress		10	Notre Dame Cathedral	2	4	2			
♪ Dagam3 ♪ Dagam4 ♪ Dagam5 ☆ Dagam6 ♪ Dagam6 ♪ Dagam6 > Fight 29 A FightMinakapot 30 & Sattraction.Load(2) A FightMinakapot 31 & Sattraction.Delete() 32 > G 6000 G 6000 G 6000 G 6000 G 6000 G 6000	2 Diagram 1											
Diagram3 Diagram4 Diagram5 Diagram5 Diagram5 Diagram5 Diagram6	Diagram2											
Comparises a construction Flar Fight 29 Fight 29 Fight 29 Settraction.Load(2) 30 Sattraction.Load(2) 31 Sattraction.Delete() 31 Sattraction.Delete() 32 Sattraction.Delete() 32 Sattraction.Success() 33 Commit 34 Commit 34	P Diagram3											
Commit	Diagram4											
Image: Structure State Image: Structure State Image: Structure State Image: Structure Struct	Lagram5											
Fight 29 A FightMemotion 300 A FightDependention to all & attraction.Load(2) 30 & & traction.Load(2) 31 & & & traction.Delete() 32 aif & & & & & & & & & & & & & & & & & & &	EnterAttractionsFilter											
A Fighthemakapot 30 &attraction.Load(2) 30 &attraction.Load(2) A Fighthemakapot 31 &attraction.Delete() 31 &attraction.Delete() C Guo00 32 34 &attraction.Success() C Guo00 34 commit C Guo00 34 endif	> 🥅 Flight	29										
A FightDepartureAirport 31 & attraction.Delete() C Guodol C Guo	A FlightArrivalAirport	30	&attraction.Loa	1(2)	30 &attract	ion.Load(2)						
Common 32 if & attraction .Success() Commit 33 Commit Commit 34 endif		31	&attraction.Del	ete()	31 &attract	ion.Delete()						
C GA0081 33 Commit C GA0010 34 endif C GA0000 34 endif	A FlightDepartureAirport				32 ∃if &attr	action.Success()						
ব ৫০০০০ 34 endif ব ৫০০০০ ব ৫০০০০	A FlightDepartureAirport				33 Con	mit						
	FlightDepartureAirport				34 endif							
	FlightDepartureAirport Gx00A0 Gx00B1 Gx0010				×							
G Gx0040	FlightDepartureAirport Gx00A0 Gx00B1 Gx0010 Gx0020											
	A FlightDepartureAirport G: 0x00A0 G: 0x00B1 G: 0x00010 G: 0x00010 G: 0x00020 G: 0x0030	N										

As we do in the transaction, we first have to load attraction 2 in the variable structure, and then simply give the order to delete.

This is done with the Delete command. We cannot repeat it enough: when this method is executed, all the transaction logic will be executed in Delete mode, including the referential integrity controls. Therefore, if there were related information, such as city tours featuring this attraction that we want to delete, the referential integrity control performed by the transaction – and by the Business Component– will prevent it. An error will occur and the record will not be deleted from the table. However, if there are no errors the record will be deleted. The variable will be loaded with the data anyway, in case we want to do something with them.

The Delete method does not return the result of the operation, so to find out whether it was successful, we will have to query it explicitly using the Success method (the opposite is the Fail method). And, for example, Commit the deletion there, i.e., permanently delete the record. This method can be used after any operation, for example, after Load, to know if it found the requested record in the table.

Let's try what we've seen: We run the procedure and note that record 2

has indeed been deleted.

TravelAge	ency - GeneYus Trial										- 11	×
File Edit V	/iew Lavout Insert Build Kno	wledge I	Janager Wind	ow Tools	Test Help						D	~
	IX D BIOGO W	. *** V	Net Enviro	nment	- Release -							
🎯 KB Explor	ier	7 X	Attraction	× 🐉 II	nsertCategoryUpdateAttractions * >	c			V Properties		ą	ф Х
Open: Name of			Source * L	-		_						×
) c	CategoriesAndTheirAttractionsList	^	Subrautinas				Applicat	tion Name	2			
> 🔳 C	Category		subroutines	-								
) c	CountriesRanking		8									
) c	CountriesWithMoreThan2Attractions		9									
> 🔳 C	Country		10		Attraction		, lhn					
> 🥅 C	Customer		11	1			\bigcirc					
1 D	DataProviderCountries		13									
Å, D	Diagram 1		14									
Å,	Diagram2		15		Id		0					
ř.	Diagram3		16	1								
÷,0	Diagram4		18									
	Diagram5		19		Name				The attraction name must not be empty	- I		
	EnterAttractionsFilter		20									
	light		21		Country Id							
	lightArrivalAirport		23		country tu		<u> </u>					
	iightDepartureAirport		24									
	3X00A0		25	1	Country Name							
	2-0010		27									
	2+0020		28	//endif								
	20020		29									
	20040		30	//&attra //&attra	ction.Load(2)							
	3x0051		32	//if &at	traction.Success()							
	ax0060		33	// Com	mit							
G G	ax0070		34	//endif								
G G	ax0081		36	т		& a	attraction AttractionNa	ame?				
G G	3x0090		37	&attract:	ion = new()	_ uu		inno.				
🔑 le	nsertCategoryUpdateAttractions		38	&attract	ion.CountryId = find(Co	untryId,	CountryName = "China")					
) N	lumberOfAttractionsByCountry		40	&attract: &attract:	ion.CityId = find(CityI	ategoryId	ame = Beljing) d. CategoryName = "Monument")					
) P	PrintRanking		41				,,					
1 R	RankingCountriesWithAttractionsQty		42 🗉	If &attra	action.Insert()							
🔒 s	DTCountries		43	Com	mit							
🔒 s	DTCountry		44	ena r					~			
🔒 S	DTCustomer	~	<						Properties Toolbox	C Tests Exp	plorer 🗄 Tests R	Results
Cutput												
C:\Models\Trial	ITrunk\Travel.Agency								Ln 44 Col 6	Ch 6	1/0/0/11/1	NS

And what happens if we run the procedure again? Nothing happens. Why?

In this second run, attraction 2 no longer exists. Therefore, it has not been loaded or removed.

What if we now try to re-insert the Chinese Wall, but forget to enter its name? Let's try it.

We run the procedure. We open Work With Attraction, and don't see the new attraction inserted. Of course, we forgot to enter its name and the error rule we had declared is being triggered, preventing that record from being added to the table!

If we had done this through the transaction, the user would have received the message we specified in the error rule.

Where was that message when we tried to insert it through the business component?

TravelAgency - GeneXus Trial							- 11	×
File Edit View Lavout Insert Build Knowledge	Manager Window Tools Test H	ielp						
	Net Environment Rele	ase •						
KB Explorer # X	Attraction X 🛟 InsertCate	aqoryUpdateAttractions X 💽 Navigation View	v X 🕂 Messages [Read-only] X		\sim	Properties		₽ ×
Open: Name or Pattern	Structure					📳 👌 🌾 Filter		×
SDTCustomer ^	Name	Type	Description	Is Collection		× Member: Type		^
A SDTCustomer2	🖃 📒 Messages		Messages		_	Name	Turne	
> Ell Supplier	🖹 🔒 Message				- 1	Name	туре	
", Domains	- 0 Id	VarChar(128)	Id		_	Description	Туре	
✓ p⊞ References	Pescription	VarChar(256)	Description			Class	Attribute	
GeneXus					- 1	✓ Type Definition		
Client					- 1	Based on	MessageTypes, GeneXus	
> 🕎 Configuration					- 1	Data Type	Numeric	
> 😚 Notifications					- 1	Length	2	
> 🌍 UI					- 1	Decimals	0	
> Deprecated					- 1	Signed	Falzo	
> Directions Services				***	- 1	signed	raise	
X Analytics					- 1	Collection	False	
AnalyticsPurchase					- 1	Initial value		
X Cipboard					- 1	Validation		
GridState					- 1	Value range	0123	
🔒 LocationInfo					- 1	Validation Failed M	GXSPC_OutOfRange	
LocationPickerParameters					- 1	Control Info		
LocationProximityAlert					- 1	Control Type	Combo Box	
X Log					- 1	Values	Warning 0 Error 1 Infor 2 Dah	
A Massaca					- 1	values	warning.o,error. 1,inro.z,deb	Jug.5
X Buntime					- 1	Empty Item	False	
X Server					- 1	Notify Context Char	False	
TrackingParameters					- 1	Appearance		
L Domains					- 1	Auto Resize	True	
> 🎯 SD					- 1	Width	2chr	
> 🚱 Server						Fill	True	
> 🛞 Social					- 1	Bash Calas	Window	
La Domains						Back Color	Window	
Customization					- 1	Fore Color	WindowText	ts Resulte
Output								0.000000
C:\Models\TrialTrunk\Travel.Agency							1/0/0/11/	1

The transaction has a screen to show errors to the user, but the business component does not. Consequently, error handling is also left to the developer.

How do we obtain them?

The KB Explorer has a group of References, which we will talk about another time, but it will always contain a GeneXus module, with functionalities already implemented to be used in our KB. In particular, there we will find an SDT called Messages. It is a collection of items consisting of an ID, a type, and a description. The type is also predefined in the module. It is numbered and indicates the type of message in each case: warning, error, information, debug.

<pre>&attraction = new() &attraction.CountryId = find(CountryId, CountryName = "China") &attraction.CountryId = find(CityId, CityName = "Beijing") &attraction.CategoryId = find(CategoryId, CategoryName = "Monument") >If extension Incent()</pre>			
<pre>&attraction = new() &attraction.CountryId = find(CountryId, CountryName = "China") &attraction.CityId = find(CityId, CityName = "Beijing") &attraction.CategoryId = find(CategoryId, CategoryName = "Monument") >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>></pre>			
TIL Pottmantian Incont()			
Commit else &messages = &attraction.GetMessages() for &message in &messages print MessagePB			
endfor endif		Values	Warning:0,Error:1,Info:2,Debug:3
Source * Layout * Rules Conditions Variables * [s]		Empty Item	False
MessageP8 Knessage10 Knessage1	 AttractionNan 1 The attraction 	nelsEmpty n name must not be #	empty
2 2 2 3 4 5 5 7 7 7 7 7 8 7 7 7 7 8 7 7 7 7 8 7 7 7 7 8 7	traction name tionName.IsEm	: must not be en npty();	mpty", AttractionNameIsEmpt

Every time an operation is executed on a Business Component variable, you can obtain the collection of messages generated by that operation with the GetMessages method. We will have to define a variable of the data type returned by that method in order to manipulate its result.

For example, just to be practical, let's show the result in a PDF file. We will run through the collection of messages generated with the for in command that allows running through collections, among other things. For this we define a variable of the data type of the message collection items. And we print every message of that collection in the output.

Let's enter the outputfile rule, and try it.

We run the procedure, and here is the output...

Only one item was obtained in the message collection, with an empty ID, of type 1 –which is an error. Its description is exactly what we entered in the error rule of the transaction.

If we look at the syntax of the error rule, we see that it allows a second parameter, which is optional. This parameter will allow defining the ID of the error for the business component's message. For example, we will assign it this value. If we try now... there we see it.



Let's do one last test: let's modify the rule so that it is no longer an error, but a message.

Also, let's enter a non-existent country identifier as Countryld to see how the insertion fails due to the referential integrity control. Let's try it.

When trying to insert, three messages were displayed: the first one is a Warning message, and it would not have caused the insertion failure by itself. However, the second and third ones could have caused the failure because they are of the Error type. The internal ID is this, and the message displayed to the transaction user when the country integrity fails is, NOT COINCIDENTALLY, No matching 'Country'.

Also, a referential integrity error will be thrown when trying to insert the city, which depends on the country.



Having said that, we have seen the most relevant aspects of updating the database with Business Components.

In this way, we saw that from the transaction it is possible to create a data type that is like an SDT, but that allows operations to be performed on the database from methods.

These operations preserve the logic of the transaction. Although we didn't go deeper into it, clearly not all the rules and events of the transaction will be incorporated into the business component. The ones that have to do with the interface obviously will not. The Parm rule is not taken into account, either.

In addition to allowing these operations on the database, we can check the result of the last operation performed, as well as obtain the messages generated, in a collection.

There are more methods to study. For example, the Mode method returns the mode of the business component: if it is Insert, Update, Delete. The Save method will try Insert or Update according to the variable's mode.

Also, the InsertOrUpdate method will always try to insert, and if it fails because of a duplicate key, then it will try to update.

The Insert, Update, Delete and InsertOrUpdate methods can also be applied to Business Components collections.



So, if the &attractions variable were a collection of the Attraction Business Component –in this case, a collection of three Attraction items–, applying the InsertOrUpdate method would be the equivalent of running through the collection and applying the method individually to each item.

The result will be True if the individual results were all True.



The last important thing to mention is that a variable of Business Component type can be used in any object that has some section of code, not only procedures. This means that we can update the database; for example, from a panel event that requests or displays data to the user, as we'll see.

It can also be done from events in transactions, although in that case there are some limitations that we will not see here.

		GeneXus
More		
 InsertCategoryUpdateAttractions proc 	Insert "Tourist site" category Update Attractions of Beijing	
CategoriesAndAttractions web panel	Do: Insert "Tourist site" category Update Attractions of Beijing	Undo: Delete "Tourist site" category Update Attractions of Beijing
MassiveInsertRemove panel	Remove Data: From Category From Attraction	

In the following video, we will apply everything seen here in an example. You may skip it, except for the final part.

There:

We will implement again the procedure created here, so that it inserts a new tourist attraction category: "Tourist Site." We'll change all the Beijing attractions that had the "monument" category and assign it the new one.

We will see how to do this through an interactive web panel, which offers to do the above, but also to undo what was done, leaving everything as it was.

Lastly, we will create another web panel that will allow us to remove the information from both tables, Category and Attraction. This will be taken up again later, so we recommend watching at least this final part.

GeneXus"

To be continued...

Of course there's a lot more to see, but we'll leave that for the next level. If you're interested, you can search for videos related to this topic in the GeneXus Analyst course.



training.genexus.com wiki.genexus.com