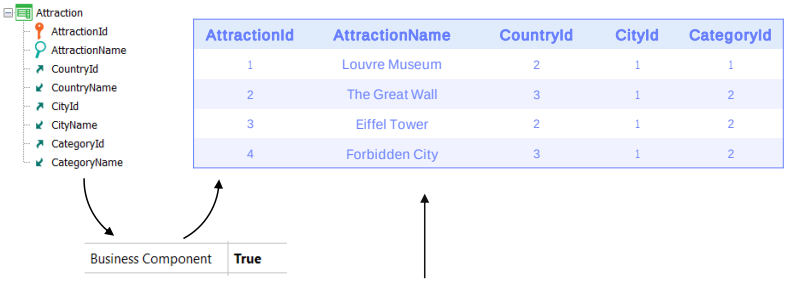
Database Update with Procedure-specific Commands

How to Insert Data

GeneXus™

1. **Business Component**: Insert(), Update(), Delete()

Insert, Update, Delete



2. **Procedure**: New, For each, Delete

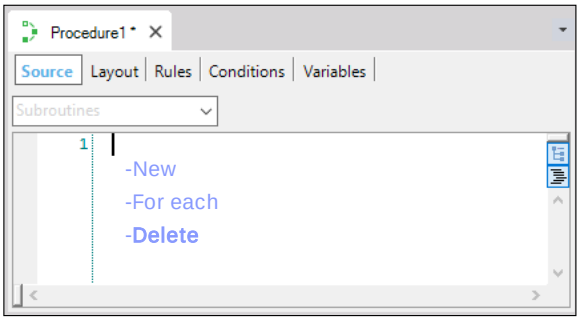
To update the database information using code, there are two possibilities:

Do it using the transaction's business component, through its Insert, Update or Delete methods (Save, and InsertOrUpdate), or do it exclusively within a procedure, through the New, For each, and Delete commands.

In other videos we study the first case in detail. Now we will focus on the second one.

PROCEDURE ONLY

Insert, Update, Delete

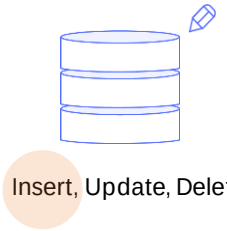


It is very important to keep in mind that this second type of update can only be performed in the Source of a procedure.
The commands we will study will not be valid anywhere else, such as Panels or Web Panels events, but only here, in procedures.

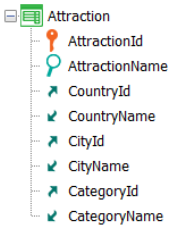
Insert

Let's start with the command that allows inserting a record into a table.

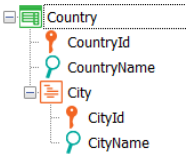
New Command

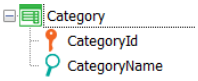


Insert, Update, Delete



AttractionId	AttractionName	CountryId	CityId	CategoryId
1	Louvre Museum	2	1	1
2	The Great Wall	3	1	2
3	Eiffel Tower	2	1	2
4	Forbidden City	3	1	2
5	Christ the Redeemer	1	2	2

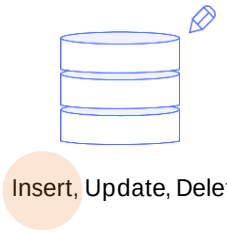


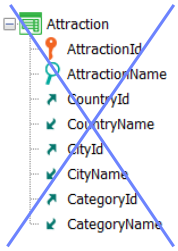


It is literally ONE record and in ONE table.

Suppose that we have the Attraction transaction, which records tourist attractions (where there will also be a Country transaction that records the information of each country and its cities and a Category, which records the categories in which the tourist attractions are classified), and that we want to insert Christ the Redeemer through a procedure in the table associated with Attraction.

New Command





AttractionId	AttractionName	CountryId	CityId	CategoryId
1	Louvre Museum	2	1	1
2	The Great Wall	3	1	2
3	Eiffel Tower	2	1	2
4	Forbidden City	3	1	2
5	Christ the Redeemer	1	2	2

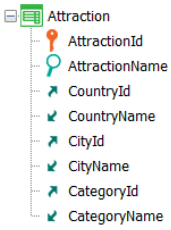
```
New
   AttractionId   = 5
   AttractionName = "Christ the Redeemer"
   CountryId = find( CountryId, CountryName = "Brazil" )
   CityId  = find( CityId, CityName = "Rio de Janeiro" )
   CategoryId  = find( CategoryId, CategoryName = "Monument" )

endnew
```

To this end, we have the New command executed in the Source of the procedure, assigning a value to each of the attributes of the table, for the record we want to insert. Here we will be working directly with the table, completely detached from the transaction that creates it. That is, the rules of the transaction don't matter, and neither do the events, nor anything else. The New command is completely indifferent to the transaction The only thing it pays attention to is the composition of the database table into which it will try to insert a record.

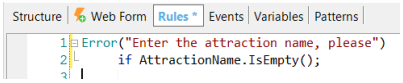
So, here the inferred attributes or transaction formulas are not involved at all. They don't exist for the New command. The only ones that matter are the table attributes. For this reason, in this case we have assigned a value to all of them.

New Command

Insert, Update, Delete



AttractionId	AttractionName	CountryId	CityId	CategoryId
1	Louvre Museum	2	1	1
2	The Great Wall	3	1	2
3	Eiffel Tower	2	1	2
4	Forbidden City	3	1	2
5		1	2	2



Not assigned attributes?
Secondary attribute

New

~~AttractionId = 5~~
AttractionName = "Christ the Redeemer"
CountryId = find(CountryId, CountryName = "Brazil")
CityId = find(CityId, CityName = "Rio de Janeiro")
CategoryId = find(CategoryId, CategoryName = "Monument")

endnew

Is this mandatory? Of course not. If we don't assign a value to an attribute, it will be considered empty.
Thus, if we don't assign a value for AttractionName, attraction 5 will be inserted without a name. Even if there is an error rule in the transaction that prevents it. As we said, the transaction here only serves to give existence to the table in the database. This is a first major difference with the insertion through a Business Component.

New Command

Not assigned attributes?

Primary key attribute

AttractionId	AttractionName	CountryId	CityId	CategoryId
1	Louvre Museum	2	1	1
2	The Great Wall	3	1	2
3	Eiffel Tower	2	1	2
4	Forbidden City	3	1	2

Uniqueness check

```
New
    AttractionId     = 5
    AttractionName  = "Christ the Redeemer"
    CountryId = find( CountryId, CountryName = "Brazil" )
    CityId  = find( CityId, CityName = "Rio de Janeiro" )
    CategoryId  = find( CategoryId, CategoryName = "Monument" )
endnew
```

AttractionId	AttractionName	CountryId	CityId	CategoryId
0	Louvre Museum	2	1	1
1	The Great Wall	3	1	2
2	Eiffel Tower	2	1	2
3	Forbidden City	3	1	2

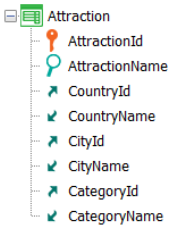
But, what if the primary key is not assigned?

An attempt will be made to insert an empty key record. As if we had explicitly assigned 0.
If the primary key is not auto numbered, then if a record with that ID 0 does not already exist in the table, it inserts it.
What if it already exists? It won't do anything. We will not see any difference between having executed the New command and not having done so.

That is, the New command controls the uniqueness of records. It will not insert a duplicate key record.

New Command

Insert, Update, Delete



AttractionId	AttractionName	CountryId	CityId	CategoryId
1	Louvre Museum	2	1	1
2	The Great Wall	3	1	2
3	Eiffel Tower	2	1	2
4	Forbidden City	3	1	2
5	Matisse Museum	2	2	1

New

Primary key attribute

Uniqueness check

```
AttractionId     = 5
AttractionName  = "Christ the Redeemer"
CountryId = find( CountryId, CountryName = "Brazil" )
CityId  = find( CityId, CityName = "Rio de Janeiro" )
CategoryId  = find( CategoryId, CategoryName = "Monument" )
```

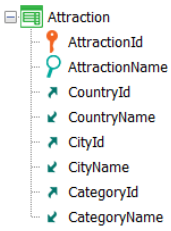
endnew

The same will happen if the attraction with ID 5 already exists in the table when we are going to execute the New command. Here it will not do anything at all, due to the uniqueness control.

And this is true for both primary and candidate keys.

New Command

Insert, Update, Delete



Unique index

AttractionId	AttractionName	CountryId	CityId	CategoryId
1	Louvre Museum	2	1	1
2	The Great Wall	3	1	2
3	Eiffel Tower	2	1	2
4	Forbidden City	3	1	2

New

AttractionId = 5
AttractionName = “Eiffel Tower"
CountryId = find(CountryId, CountryName = "Brazil“)
CityId = find(CityId, CityName = “Rio de Janeiro”)
CategoryId = find(CategoryId, CategoryName = "Monument“)

endnew

Candidate key attribute

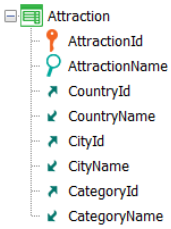
Uniqueness check

Let’s imagine, for example, that AttractionName is a candidate key; i.e., we have a unique index defined over this attribute.

And let's suppose that the record of ID 3 had as AttractionName value the same one we are now trying to insert as record 5.
When the New command is executed, it will do nothing. This is because it will find that there already exists a record with the same AttractionName as the record we want to insert.

New Command

Insert, Update, Delete



AttractionId	AttractionName	CountryId	CityId	CategoryId
1	Louvre Museum	2	1	1
2	The Great Wall	3	1	2
3	Eiffel Tower	2	1	2
4	Forbidden City	3	1	2
5	Christ the Redeemer	1	2	2

Not assigned attributes?

Primary key attribute

auto numbered

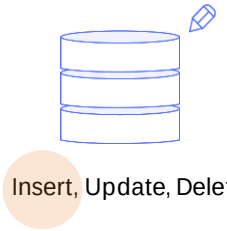
New

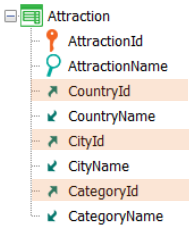
AttractionName = "Christ the Redeemer"
CountryId = find(CountryId, CountryName = "Brazil")
CityId = find(CityId, CityName = "Rio de Janeiro")
CategoryId = find(CategoryId, CategoryName = "Monument

&AttractionId = AttractionId

Now, going back to what we were wondering about what happens when we don't specify a value for the primary key... if it is auto numbered, then it will never fail due to a duplicate primary key: it will always insert a record with the next number. How do we know what number was assigned to it? The value in the attribute is stored in memory immediately after the New. Thus, we can, for example, assign it to a variable so as not to lose it the next time the attribute is used.

New Command





AttractionId	AttractionName	CountryId	CityId	CategoryId
1	Louvre Museum	2	1	1
2	The Great Wall	3	1	2
3	Eiffel Tower	2	1	2
4	Forbidden City	3	1	2

Referential integrity checks?

NO

Not assigned attributes?

Foreign key attribute

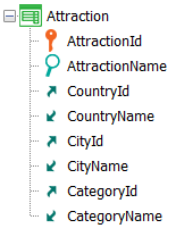
```
New
    AttractionId     = 5
    AttractionName  = "Christ the Redeemer"
    CountryId = find( CountryId, CountryName = "Brazil" )
    CityId  = find( CityId, CityName = "Rio de Janeiro" )
    CategoryId  = find( CategoryId, CategoryName = "Monument" )
endnew
```

What about attributes that are foreign keys? Such as CountryId, CityId, CategoryId.

Does the new command perform referential integrity checks?

The answer is no. This is because the command update was created to have a fast update method, with the best possible performance. Performing these checks always slows down the operation. When dealing with a single record this doesn't matter, but let's think about what happens if we have to insert millions of records.

New Command



AttractionId	AttractionName	CountryId	CityId	CategoryId
1	Louvre Museum	2	1	1
2	The Great Wall	3	1	2
3	Eiffel Tower	2	1	2
4	Forbidden City	3	1	2

Insert, Update, Delete

Referential integrity checks?

NO

Not assigned attributes?

Foreign key attribute

New

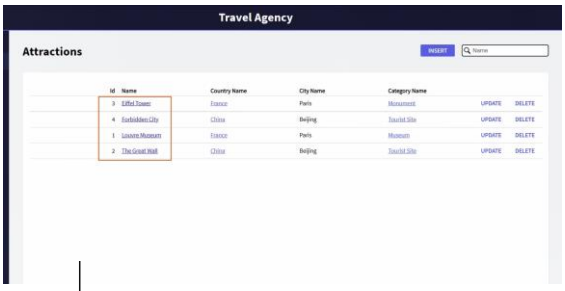
AttractionId = 5
AttractionName = "Christ the Redeemer"
CountryId = 15
CityId = find(CityId, CityName = "Rio de Janeiro")
CategoryId = find(CategoryId, CategoryName = "Monument")

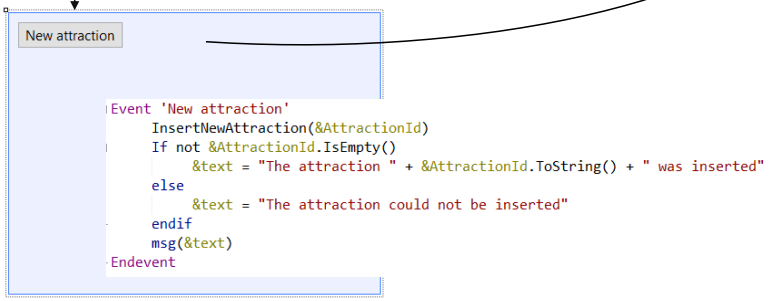
endnew

So, for example, if for the new record we want to assign a country with ID 15, the New command will not check that there is a country with that value in the table that stores the countries. It will insert the record without any problem. And the database will be left in an inconsistent state.

Let's go to GeneXus to try it.

New Command

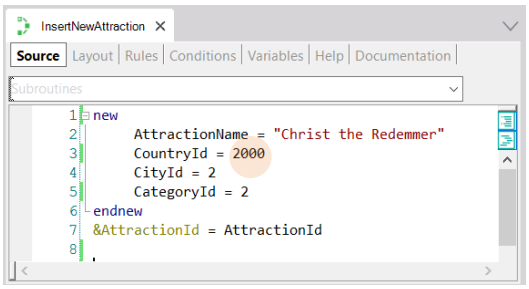


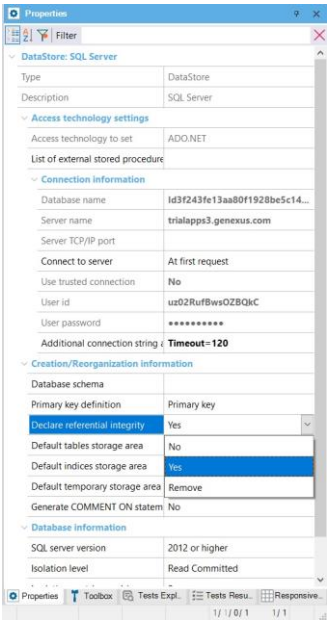


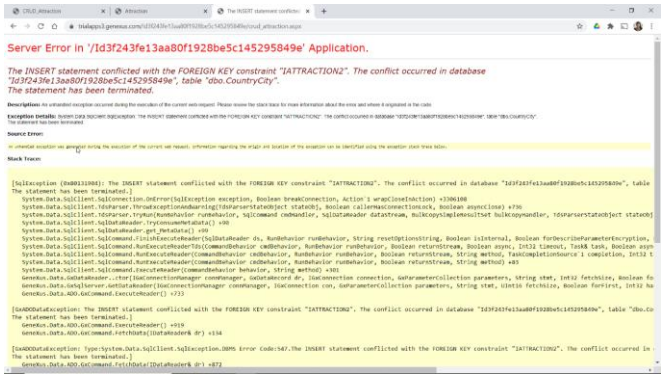
We have the three transactions with data. In particular, Attraction has these four records.
AttractionId is auto-numbered.

We have created this web panel where the user will press the “New attraction” button, which will call a procedure that will try to insert a new record in the Attraction table for Christ the Redeemer.
The procedure will return the AttractionId that the database assigned to the inserted record. With this we create the message that the user will see in the panel.
Let’s try it.
Attraction 5 was inserted. Let's see... and here is, indeed, attraction 5 inserted in the table.

New Command





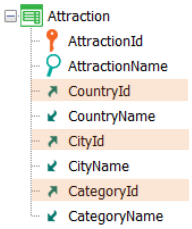


Now, let's see what happens if we want to assign a non-existent country for the new record. Such as this one. We run it.
The program crashed, when in fact we expected it to have inserted the record without any problem, since we said that the New command doesn't check referential integrity. What happened then?

It is true that the program is not checking referential integrity, but the database is. So the New command tried to perform the insertion, but the database would not let it, and threw an exception.

By default, the database checks referential integrity. We can turn off this control, by means of a property.
But what we clearly see is that the New command is not doing so. That's why we have to be very careful when using this command, because this type of program crash is unacceptable to the end user.

New Command



AttractionId	AttractionName	CountryId	CityId	CategoryId
1	Louvre Museum	2	1	1
2	The Great Wall	3	1	2
3	Eiffel Tower	2	1	2
4	Forbidden City	3	1	2

Insert, Update, Delete

Referential integrity checks?

NO

Not assigned attributes?

Foreign key attribute

New

AttractionId = 5
~~Attraction~~Name = "Christ the Redeemer"
CountryId = 15
CityId = find(CityId, CityName = “Rio de Janeiro”)
CategoryId = find(CategoryId, CategoryName = "Monument“)

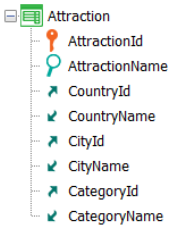
endnew

What if we leave a foreign key unassigned?

Again, the New command will not perform any check and will try to insert the record. If the database makes them, then it will throw an exception like the previous one that will stop the program if it is not caught and handled by it.
We might think that if the attribute accepts nulls, it will never fail because the database will allow that null for the foreign key. However, we must be careful about how the database interprets that it is a null and not an empty value. We will discuss this in another video.

New Command

Insert, Update, Delete



AttractionId	AttractionName	CountryId	CityId	CategoryId
1	Louvre Museum	2	1	1
2	The Great Wall	3	1	2
3	Eiffel Tower	2	1	2
4	Forbidden City	3	1	2
5	Christ the Redeemer	1	2	2

COMMIT?

New

AttractionId = 5
AttractionName = "Christ the Redeemer"
CountryId = find(CountryId, CountryName = “Brazil”)
CityId = find(CityId, CityName = “Rio de Janeiro”)
CategoryId = find(CategoryId, CategoryName = "Monument“

endnew
....
Commit

If everything is OK, then the New command inserts a record in the table. And what happens with the Commit?

If we look at the navigation of the procedure that we executed a while ago, it is showing a warning that says that the program could be called by another program and that the Commit on Exit property is set to Yes. We can see it here.
This property is also found in the other object that operates on the database: the transaction object. Here we see it, below the Transaction integrity group (in another class we will study this important topic, which is how to define and achieve transactional integrity).

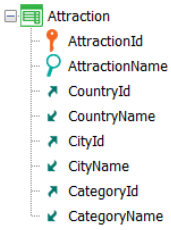
The important thing is that if this property is set to Yes, it means that an automatic Commit will be added in the object's source code (as long as the object performs some operation on the database).

New Command

In the transaction, at the end of the operation on the header and its lines.

New Command

Insert, Update, Delete



AttractionId	AttractionName	CountryId	CityId	CategoryId
1	Louvre Museum	2	1	1
2	The Great Wall	3	1	2
3	Eiffel Tower	2	1	2
4	Forbidden City	3	1	2
5	Christ the Redeemer	1	2	2

COMMIT?

New

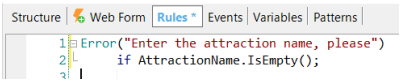
AttractionId = 5
AttractionName = "Christ the Redeemer"
CountryId = find(CountryId, CountryName = “Brazil”)
CityId = find(CityId, CityName = “Rio de Janeiro”)
CategoryId = find(CategoryId, CategoryName = "Monument“

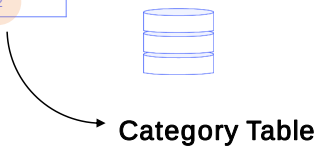
endnew
....
Commit

In a procedure object, at the end of the Source. That's why we didn't have to specify it. If the property were turned off, then we would have to explicitly write the Commit command, just as we did with Business Components.

	Uniqueness check	Referential Integrity check	Rule/Event Execution
New command	✓	✗	✗

AttractionId	AttractionName	CountryId	CityId	CategoryId
1	Louvre Museum	2	1	1
2	The Great Wall	3	1	2
3	Eiffel Tower	2	1	2
4	Forbidden City	3	1	2
5	Christ the Redeemer	1	2	2





To sum up what we've seen so far:

When trying to insert a record into a table using the New command in a procedure, the command performs uniqueness controls by primary key and candidate keys, thus ensuring that a record is not added to the table that duplicates the key. If it finds a duplicate key then it does nothing.

If any attribute of the record to be inserted is a foreign key, the New command will not perform a referential integrity check. That is to say, it will not search the table that is referenced for the existence of a record with the value that we want to use, in order to insert the record. BUT if the database has referential integrity declared, then it will perform the check and cancel the program if integrity is being violated. If it doesn't have referential integrity declared, then the insertion of the record will be allowed regardless of whether it violates integrity.

And regarding the rules and events found in the transaction associated with the table, these clearly have nothing to do here. Let's remember that for the New command only the table matters, not where it comes from.

	Uniqueness check
New command	✓

AttractionId	AttractionName	CountryId	CityId	CategoryId
1	Louvre Museum	2	1	1
2	The Great Wall	3	1	2
3	Eiffel Tower	2	1	3
4	Forbidden City	3	1	2

New

```
AttractionId   = 3
AttractionName = "Eiffel Tower"
CountryId = 2
CityId  = 1
CategoryId  = 3

endnew
```

New

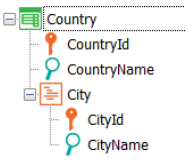
```
AttractionId   = 3
AttractionName = "Eiffel Tower"
CountryId = 2
CityId  = 1
CategoryId  = 3

When duplicate
   for each Attraction
      CategoryId =
   3
   endfor
```

However, we might want to perform some action if the record is duplicated by a key. For example, instead of inserting it, update it. In the example, we may want to change the value of CategoryId.

To this end, the when duplicate clause is used. The code it contains will only be executed when the primary key or a candidate key is found to be duplicated. If what we want in that case is to update the record, then we have to do it inside a For each [C], as we see here. Values are not assigned directly to the attributes to be modified, as you might think; instead, this has to be done by writing a For each, without having to filter by AttractionId, because it is already implied. This is the way the New command understands that we want to update those attributes of the record it found duplicated. In this case, update only the CategoryId attribute. Here it would be possible update attributes of the extended table.

Not assigned attributes → Depending on the context. are they instantiated?

```
For each Country.City
    where CountryName = "France"

    print cityInfo

    new

        AttractionName  = CityName + " attra
        CategoryId      = 2

    endnew

endfor
```



CountryId	CityId	CityName
1	1	Sao Paulo
1	2	Rio de Janeiro
2	1	Paris
2	2	Nice
3	1	Beijing

AttractionId	AttractionName	CountryId	CityId	CategoryId
1	Mattisse Museum	2	2	1
2	The Great Wall	3	1	2
3	Eiffel Tower	2	1	2
4	Forbidden City	3	1	2
5	Paris attraction	2	1	2
6	Nice attraction	2	2	2

Returning to the issue of what happens to the attributes of the New command table that are not assigned a value, we had said that they remain empty, but it actually depends on the context. They are empty if they are not instantiated in the context in which the New command is found; i.e. if they have no value.

For example, suppose we are running through the cities of France with a For each command and print each one, for example. In addition, we immediately execute the New command that we are looking at.

First of all: How does GeneXus determine the base table of the New command? By paying attention only to the attributes to which a value is being assigned. It has to find a table in the database that contains them. It will clearly be Attraction.

Does this CityName take part? No, it doesn't. If it is there, it is because in the context in which we have written the New command, we know that it is instantiated. This CityName will be that of the For each. In sum, for each city, we want to insert a new tourist attraction with the name of the city and category 2.

OK, but what attraction, country and city identifier will be assigned to the record to be inserted? In this case, the only empty one will be AttractionId, which is not instantiated in the context, because it isn't in the extended table of the For each. And it is not

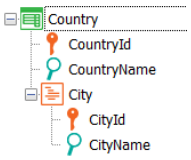
received by parameter in the attribute either, which is the other instantiation method we know.
So if AttractionId is auto numbered, then when the record is inserted the database will give the next one to the last number.
However, CountryId and CityId are instantiated in the context. Therefore, they will be assigned this context value. In this case, the country and city in which we are positioned in the For each.

And then the same will happen for the next city in the For each...

Not assigned attributes → Depending on the context.. are they instantiated?

```
For each Country.City
    where CountryName = "France"

    print cityInfo

    new
        AttractionName =
        CategoryId     = 2
    endnew

endfor
```

WARNING
Base table: CATEGORY!!!!



CountryId	CityId	CityName
1	1	Sao Paulo
1	2	Rio de Janeiro
2	1	Paris
2	2	Nice
3	1	Beijing

AttractionId	AttractionName	CountryId	CityId	CategoryId
1	Mattisse Museum	2	2	1
2	The Great Wall	3	1	2
3	Eiffel Tower	2	1	2
4	Forbidden City	3	1	2
5		2	1	2
6		2	2	2

What would happen if we wanted the same two records to be inserted in Attraction, but with an empty AttractionName?

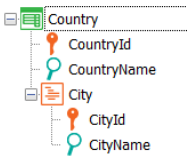
It would seem that by simply not assigning a value to AttractionName we would be implementing it. However, if we look at the attributes that GeneXus will use to determine its base table, that of the New, there is only CategoryId. Therefore, it will not choose Attraction as the base table, but Category.

So, how do we have it choose the Attraction table? One possibility is to explicitly assign the empty value for AttractionName. Because in this way, by naming this attribute, it will participate in determining the base table and it will make the base table to be Attraction instead of Category.

New Command

Not assigned attributes → Depending on the context ... are they instantiated?

```
For each Country.City
    where CountryName = "France"

    print cityInfo

    new Defined by AttractionName

        CategoryId          = 2

    endnew

endfor
```



CountryId	CityId	CityName
1	1	Sao Paulo
1	2	Rio de Janeiro
2	1	Paris
2	2	Nice
3	1	Beijing

AttractionId	AttractionName	CountryId	CityId	CategoryId
1	Mattisse Museum	2	2	1
2	The Great Wall	3	1	2
3	Eiffel Tower	2	1	2
4	Forbidden City	3	1	2
5		2	1	2
6		2	2	2

The other option is to use the Defined by clause that allows adding more attributes to be considered together with the others in determining the base table.

Summary

new
Defined by *Attribute*$_1$, *Attribute*$_2$, ..., *Attribute*$_N$
Blocking *NumericExpression*

Attribute$_1$ = *expression*$_1$
Attribute$_2$ = *expression*$_2$
...
Attribute$_N$ = *expression*$_N$

When duplicate
...
for each *BaseTransaction*
Attribute$_I$ = *expression*$_I$
Attribute$_K$ = *expression*$_K$
...
endfor
...

	Uniqueness check	Referential Integrity check
New command	✓	✗

COMMIT

In short, the New command is used to insert a record into a table. The table is determined by the attributes assigned to it. If a Defined By clause is added, then the attributes listed there also participate. Here the concept of an extended table does not make any sense.

On the other hand, we had seen that the only programmatic control performed by the New command is the uniqueness control. And that if a record was found with the key we are trying to insert, then the New command did nothing.... Unless we had programmed the When Duplicate clause.

We had said that there, among other commands, we can include a For each to update attributes of the record that was found duplicated. Can all attributes be updated? For example, can the primary key be updated there? The answer is no, but it will be possible update attributes of the extended table.

Finally: for the record to be committed in the database we must make sure that the Commit command is executed. In a procedure, by default, an implicit Commit is placed at the end. But we can explicitly write Commits in the Source, where it is convenient for us.

We will not see it here, but optionally a Blocking clause can be specified, which allows making insertions in the database in

blocks, instead of record by record, as long as the New command is inside a repetitive structure. This will clearly be used for efficiency reasons, when batch insertions are very large.

GeneXus™

training.genexus.com
wiki.genexus.com