GeneXus 18 Junior Analyst Exam

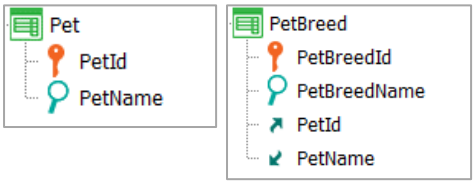
Reality: Pet shop.
About multiple choice questions:

- There is only one correct option.
- In this exam, NO points are deducted for incorrect answers.

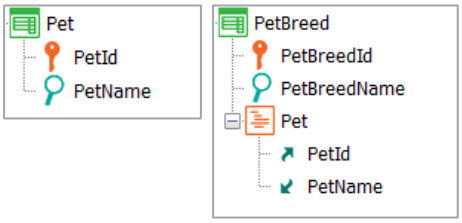
1) There is a GeneXus application for managing a pet shop.

Knowing that a pet (Pet) belongs to a breed (PetBreed), and many pets can belong to the same breed, determine the transaction design you consider correct.

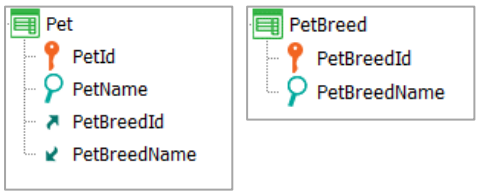
1.1 -



1.2 -



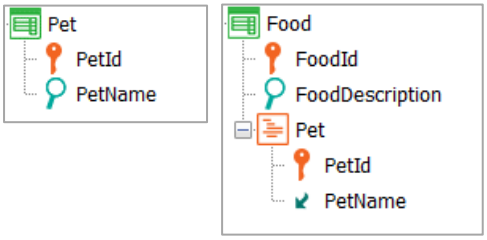
1.3 –



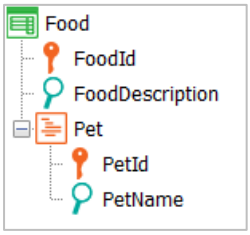
1.4 – None of the above options is correct.

2) Knowing that a pet (Pet) can eat several types of food (Food), and that one type of food can be eaten by several pets, determine the transaction design that you consider correct.

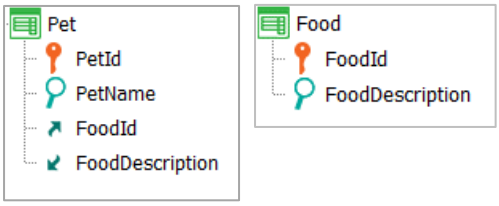
2.1 –



2.2 –



2.3 –



2.4 – None of the above options is correct.

3) Consider the transaction design shown below and determine the option that you consider correct.

3.1 – Every pet (Pet) has a set of specific care guidelines (SpecialCare) that are associated with it and are identified as unique to that pet.

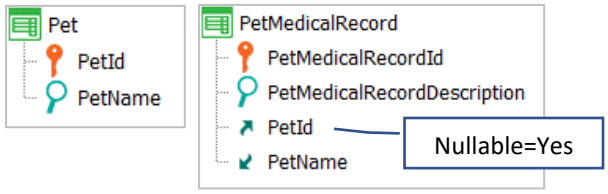
3.2 – Every pet (Pet) has a set of specific care guidelines (SpecialCare) that are associated with it, and the same care is not specific to a single pet, but can be applied to other pets.

3.3 – The design is not valid. It is not possible to create a two-level transaction without defining the second-level entity as a transaction by itself.

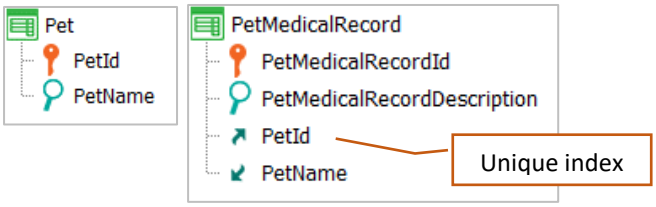
3.4 – None of the above options is correct.

4) Knowing that a pet (Pet) has only one medical record (PetMedicalRecord), and that this medical record is only for that pet, determine the option that you consider correct:

4.1 –



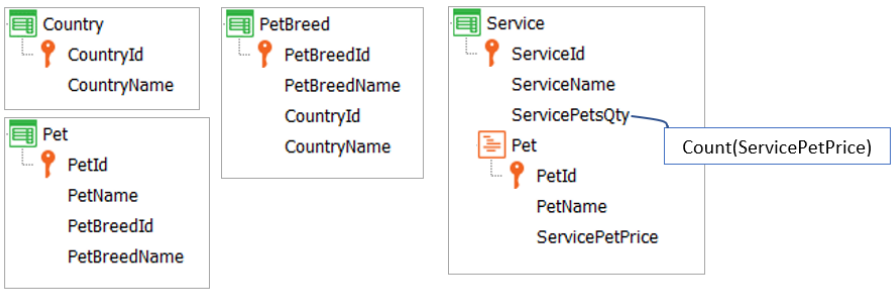
4.2 –



4.3 –

4.4 – None of the above options is correct.

5) Consider the transaction design shown below and determine the physical table structure that will be created by GeneXus.



5.1 -

COUNTRY	**PETBREED**	**PET**	**SERVICE**
CountryId*	PetBreedId*	PetId*	ServiceId*
CountryName	PetBreedName	PetName	PetId*
	CountryId	PetBreedId	ServiceName
			ServicePetsQty

5.2 –

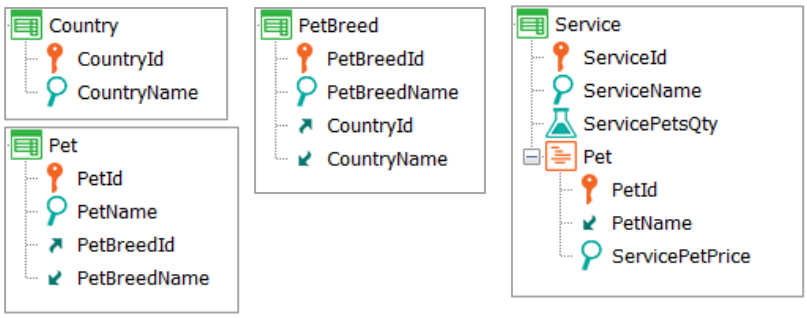
COUNTRY	**PETBREED**	**PET**	**SERVICE**	**SERVICEPET**
CountryId*	PetBreedId*	PetId*	ServiceId*	ServiceId*
CountryName	PetBreedName	PetName	ServiceName	PetId*
	CountryId	PetBreedId	ServicePetsQty	ServicePetPrice

5.3 –

COUNTRY	**PETBREED**	**PET**	**SERVICE**	**SERVICEPET**
CountryId*	PetBreedId*	PetId*	ServiceId*	ServiceId*
CountryName	PetBreedName	PetName	ServiceName	PetId*
	CountryId	PetBreedId		ServicePetPrice

5.4 – None of the above options is correct.

6) Based on the transaction design displayed, determine the extended table of the SERVICEPET table.



6.1 – SERVICEPET, SERVICE, PET

6.2 – SERVICEPET, SERVICE, PET, PETBREED, COUNTRY

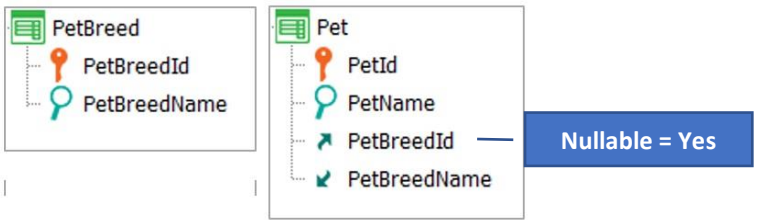
6.3 – SERVICEPET, SERVICE, PET, PETBREED

6.4 – SERVICEPET

7) Although every pet has a breed (PetBreed), sometimes it is not possible to determine that breed, and therefore this information is not considered mandatory when registering a pet.

If a breed (PetBreedId) is specified, this value must be valid.

Based on the transaction design shown below, determine what you consider correct:



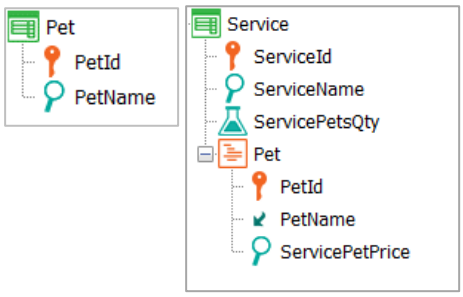
7.1) This implementation doesn't meet the requirement. By indicating that PetBreedId accepts nulls, GeneXus will not perform referential integrity checks. It will be possible to register a pet without a breed, but if a breed is indicated then it will not be checked if it exists as a record in PETBREED.

7.2) The implementation is not correct. A unique index must be defined on the PetBreedId attribute in PET.

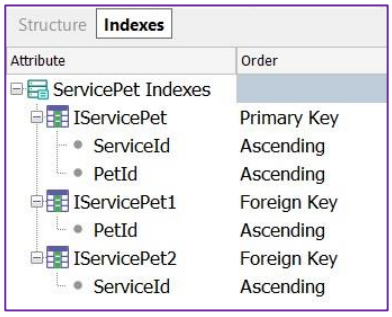
7.3) This implementation is correct and meets the requirement.

7.4) None of the above options is correct.

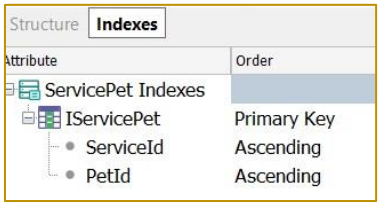
8) Based on the transaction design displayed, determine the indexes automatically created by GeneXus in the SERVICEPET table.



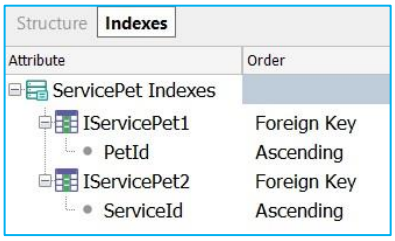
8.1)



8.2)

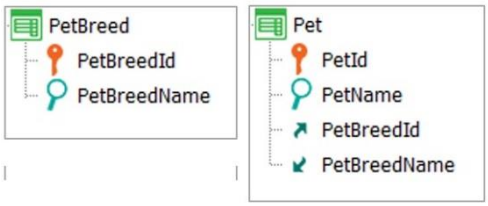


8.3)



8.4) None of the above options is correct.

9) Consider the transaction design shown below, and determine what will happen when trying to delete a breed (PetBreed) using the PetBreed transaction form.



9.1) GeneXus will delete it without making any controls.

9.2) GeneXus will automatically delete all the records in Pet that have PetBreedId as FK first, and then delete the corresponding PetBreed record.

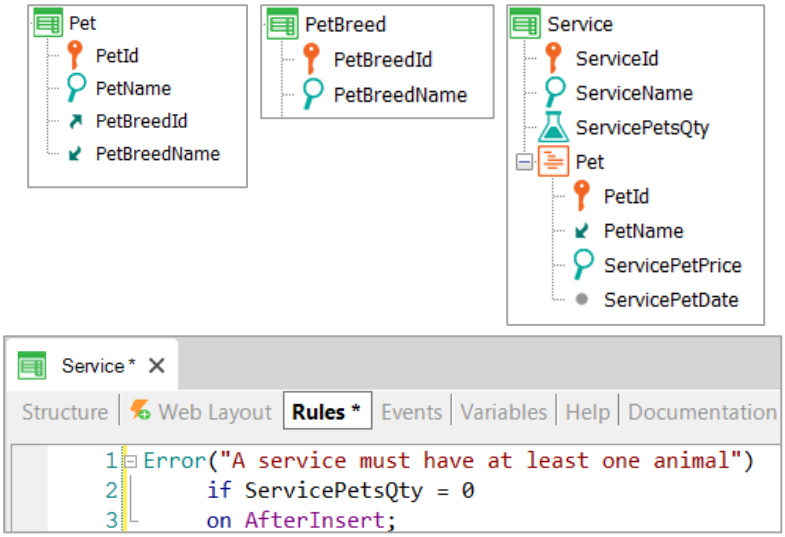
9.3) GeneXus will check that there are no records in Pet that have PetBreedId as FK. If they exist, it will issue a message indicating that related records exist and will not take any action.

9.4) None of the above options is correct.

10) In the following transaction design, the Service transaction has a formula attribute, ServicePetsQty, which counts the number of animals registered for a given service on a given date.

It is necessary to control that a service is never registered without an associated animal. The Error rule shown below is used to perform this control.

Determine what you consider correct:



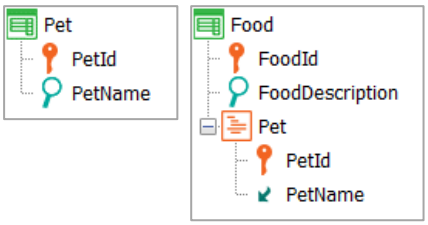
10.1 – The rule doesn't meet the requirement because it will be triggered on the server after the header data (Service) has been saved in the database and before the animals (Pet) start to be saved.

10.2 – The rule doesn't meet the requirement because it will be triggered on the server after the header data (Service) has been saved in the database and immediately after the last animal (Pet) has been saved.

10.3 – The rule doesn't meet the requirement because it will be triggered on the server right before the header data (Service) starts to be saved.

10.4 – The rule meets the requirement because it will be triggered on the client before pressing Confirm.

11) Consider the transactions displayed and determine the order in which the rules declared in the Food transaction will be triggered.



Rules:

a) FoodDetail(FoodId) on AfterComplete;
b) Reservation(FoodId) on AfterInsert;
c) StockControl(FoodId) on AfterLevel level PetId;

11.1 – b), c), a)

11.2 – c), b), a)

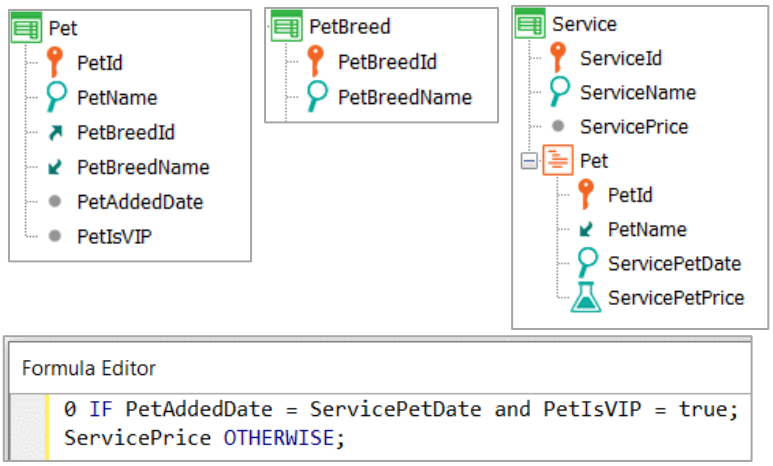
11.3 – c), a), b)

11.4 – The rules are triggered in the order in which they are declared.

12) In the pet store there are VIP pets, i.e. pets that have certain benefits.

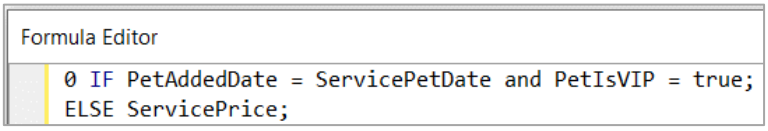
When associating a pet with a certain service, if the pet is a VIP and the day to perform the service matches the day on which the pet was registered in the store, the service will be free of charge. Otherwise, the base price of the service will be applied.

Determine what you consider correct from the calculation associated with the ServicePetPrice attribute.



12.1 – The implementation of the formula is incorrect because it is not possible to use the attributes PetAddedDate and PetIsVIP in it, since they are not found in the structure of the Service transaction (neither in the header nor in the Pet sublevel).

12.2 – The syntax of the formula is incorrect because when the IF structure is used, ELSE should be used to refer to the others. The valid implementation would be as follows:

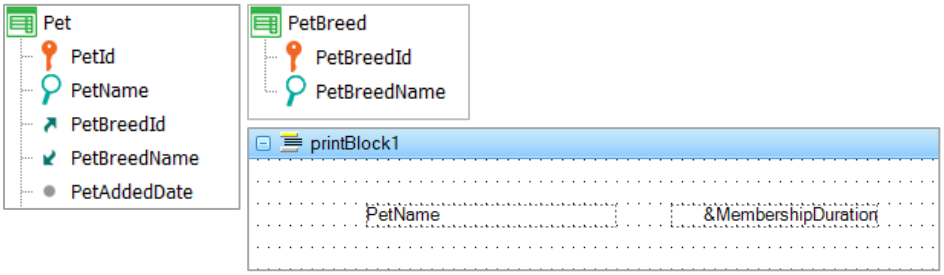


12.3 – The implementation of the formula meets the requirement.

12.4 – None of the above options is correct.

13) A list of all the animals (Pet) in the pet store needs to be displayed, with their name (PetName) and the time since registration (&MembershipDuration).

Look at the following transaction and the procedure Layout. What should be the implementation of the source?



13.1 –

```
For each Pet
     &MembershipDuration = &Today.Year() - PetAddedDate.Year()
endfor
Print printBlock1
```

13.2 –

```
&MembershipDuration = &Today.Year() - PetAddedDate.Year()
For each Pet
     Print printBlock1
endfor
```

13.3 –

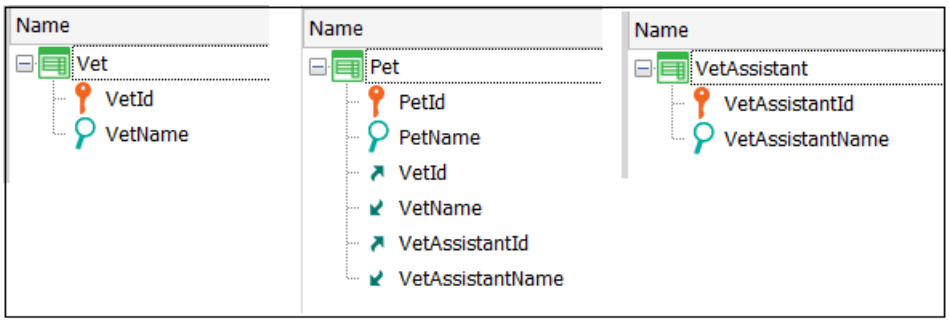
```
For each Pet
     &MembershipDuration = &Today.Year() - PetAddedDate.Year()
     Print printBlock1
endfor
```

13.4 – None of the above options is correct.

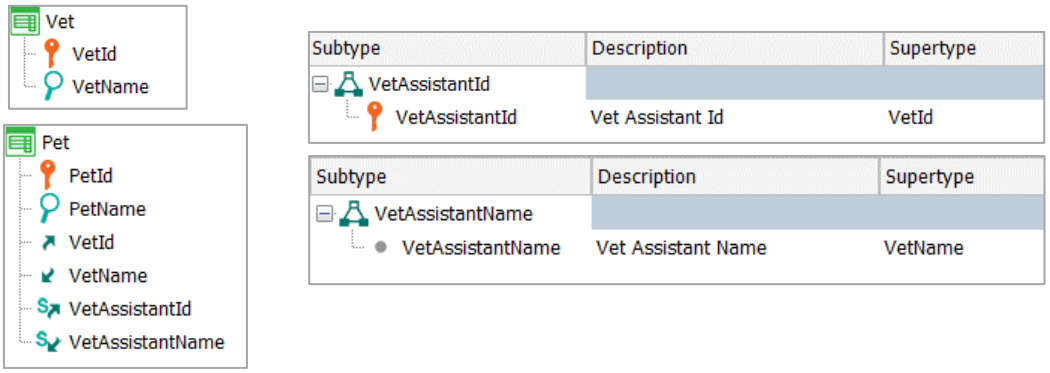
14) Even though every pet (Pet) has a primary veterinarian, a substitute veterinarian needs to be recorded for cases when the primary veterinarian is not available.

Indicate which of the following transaction designs (and of subtype groups if they are included) is adequate to model the above reality.

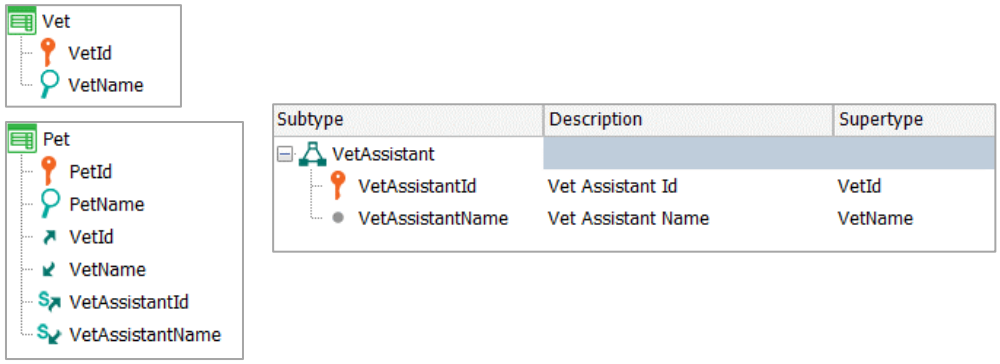
14.1 –



14.2 –



14.3 –

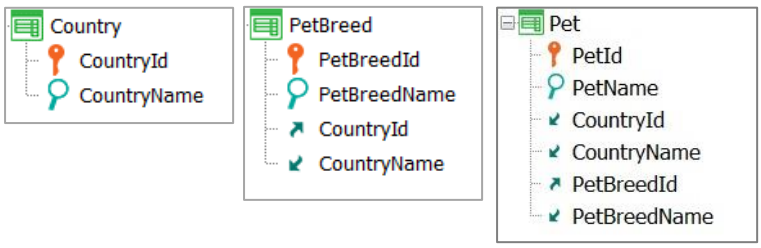


14.4 – None of the above options is correct.

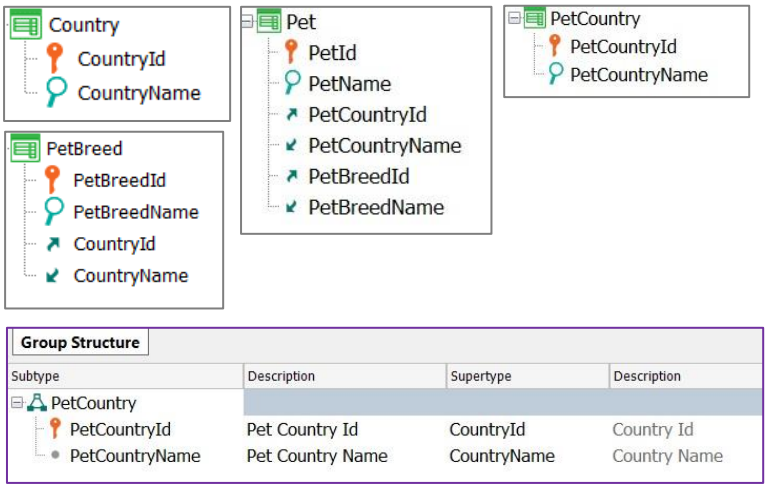
15) Although every breed (PetBreed) is associated with a certain country of origin (Country), the country where the pet (Pet) was obtained (adopted) by its owners should also be recorded.

Determine the implementation option you consider correct:

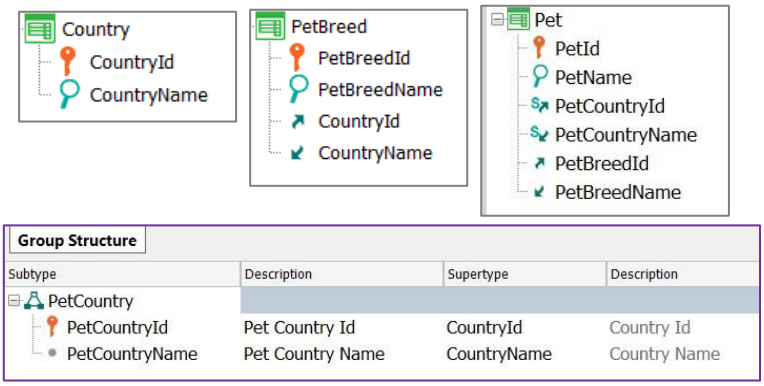
15.1) Simply due to the order in which the attributes are placed in the Pet transaction structure, it will be possible to record the country where the pet was adopted.



15.2) The PetCountry transaction must be defined, and the corresponding group of subtypes must be declared.



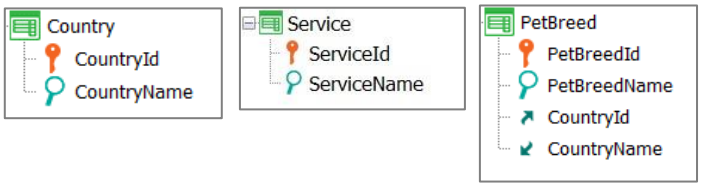
15.3) The attributes and subtype group must be defined as follows:

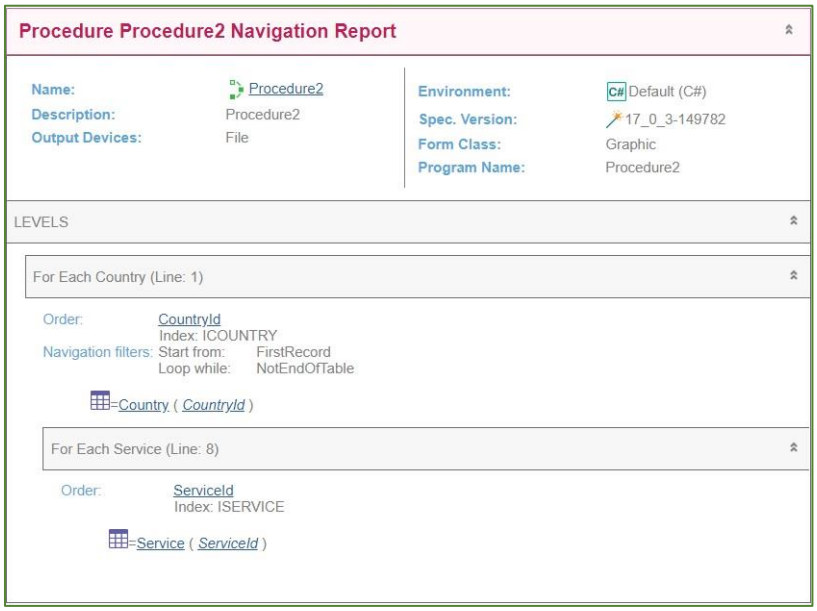


15.4) None of the above options is correct.

16) Consider the transaction design and navigation list shown below.

What does it implement?





16.1) Cartesian product

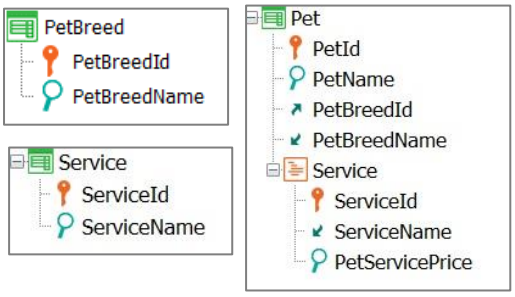
16.2) Control break

16.3) Join

16.4) None of the above options is correct.

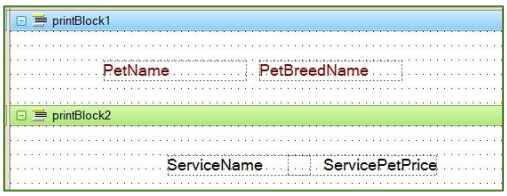
17) Consider the transaction design shown below. Complete information (header and lines) about a certain Pet received in a parameter needs to be listed.

Determine the implementation that you consider correct:



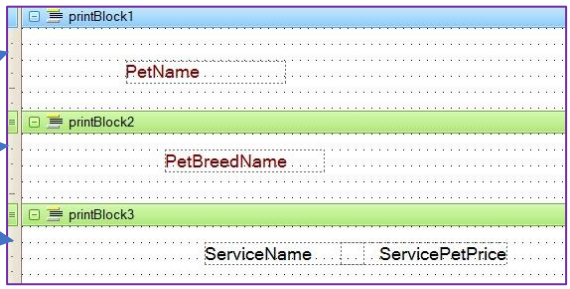
17.1) **Parm**(**in**: PetId);

```
For each Pet.Service
    Print printBlock1
    For each Pet.Service
      Print printBlock2
    Endfor
Endfor
```



17.2) **Parm**(**in**: &PetId);

```
For each Pet
   Where PetId = &PetId
     Print printBlock1
     For each PetBreed
       Print printBlock2
       For each Pet.Service
         Print printBlock3
       Endfor
     Endfor
Endfor
```

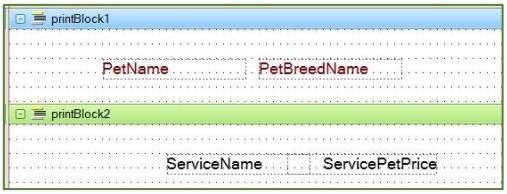


17.3) **Parm**(**in**: &PetId);

```
For each Pet
   Where PetId = &PetId
     Print printBlock1
     Print printBlock2
   Endfor
Endfor
```

17.4) **Parm**(**in**: PetId);

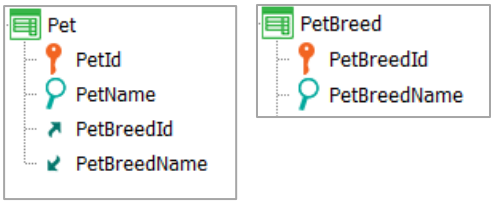
```
For each Pet
    Print printBlock1
    For each Pet.Service
      Print printBlock2
    Endfor
Endfor
```



18) Consider the transaction design shown below. A list has to be defined to show all the breeds (PetBreed) and, for each one of them, the list of pets (Pet) that belong to it.

All the breeds should be listed, regardless of whether they have registered pets of that breed or not.

Select the implementation option you consider correct to meet the request described.



18.1 –

For each PetBreed
Print printblock1
Endfor
For each Pet
Print printblock2
Endfor

18.2 –

For each Pet
Print printblock1
For each Pet
Print printblock2
Endfor
Endfor

18.3 –

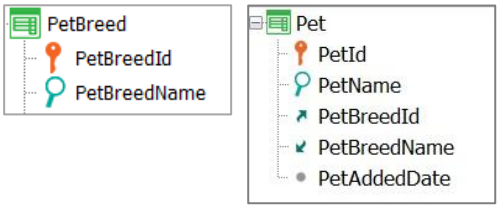
For each Pet
Print printblock1
Print printblock2
Endfor

18.4 –

For each PetBreed
Print printblock1
For each Pet
Print printblock2
Endfor
Endfor

19) Consider the transaction design shown below. The pets (Pet) of breed "Beagle" (PetBreedId = 4) and "Cocker" (PetBreedId = 7) that were registered (PetAddedDate) in the year 2020 need to be listed.

Determine if the proposed implementation is correct or not.

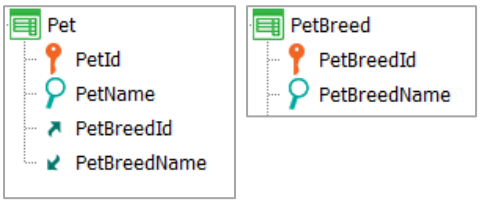


For each Pet
Where PetBreedId = 4 **or** PetBreedId = 7
Where PetAddedDate.Year() = 2020
Print printBlock1
Endfor

☐ True ☐ False

20) Consider the transaction design shown below. A list has to be defined to show all the pets (Pet) grouped by breed (PetBreed).

Only those breeds that have pets registered should be included in the list.



20.1 –

For each PetBreed order PetBreedId
Print printblock1
Print printblock2
Endfor

20.2 –

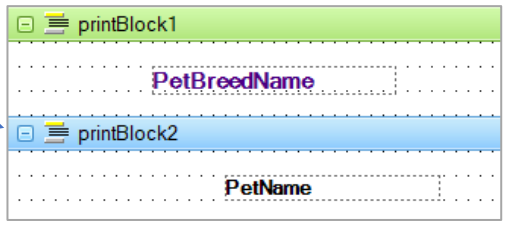
For each PetBreed order PetBreedId
Print printblock1
For each Pet
Print printblock2
Endfor
Endfor

20.3 –

For each Pet order PetBreedId
Print printblock1
For each Pet
Print printblock2
Endfor
Endfor

20.4 –

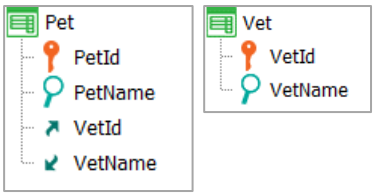
For each Pet
Print printblock1
For each Pet
Print printblock2
Endfor
Endfor



21) Consider the transaction design shown below.

A list needs to be defined showing all registered veterinarians (Vet) who have at least one pet in their care. If there is no veterinarian with at least one pet, a text should inform it.

Determine the implementation option you consider correct to meet the requirement described.



21.1 –

```
For each Vet
     where Count(PetName) >= 1
     Print printBlock1
     else
     Print printBlock2
Endfor
```

21.2 –

```
For each Vet
     where Count(PetName) >= 1
     Print printBlock1
unique
     Print printBlock2
Endfor
```

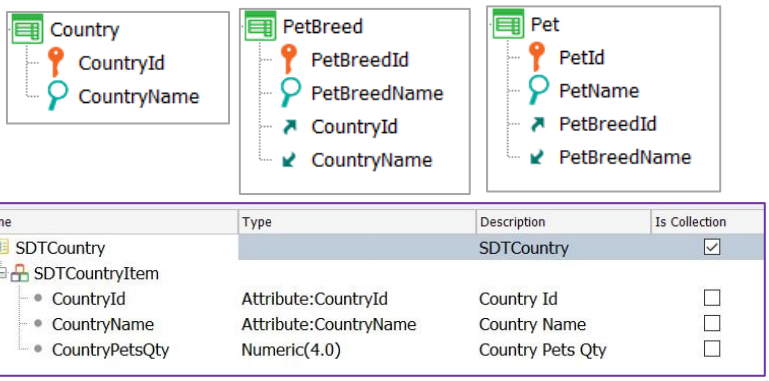
21.3 –

```
For each Vet
     where Count(PetName) >= 1
     Print printBlock1
when none
     Print printBlock2
Endfor
```

21.4 – None of the above options is correct.

22) Consider the transaction design and the definition of the SDTCountry structured data type that are shown below. A Data Provider needs to be designed to load a collection of countries (Country), each with the number of pets (Pet) of breeds (PetBreed) of that country.

Select the implementation option you consider correct.



22.1)

```
SDTCountry from Pet
{
     SDTCountryItem
     {
          CountryId
          CountryName
          CountryPetQty = count(PetName)
     }
}
```

Output	
Infer Structure	No
Output	**SDTCountry**
Collection	False

22.2)

```
SDTCountry from Country
{
     SDTCountryItem
     {
          CountryId
          CountryName
          CountryPetsQty = count(PetName)
     }
}
```

Output	
Infer Structure	No
Output	**SDTCountry**
Collection	False

22.3)

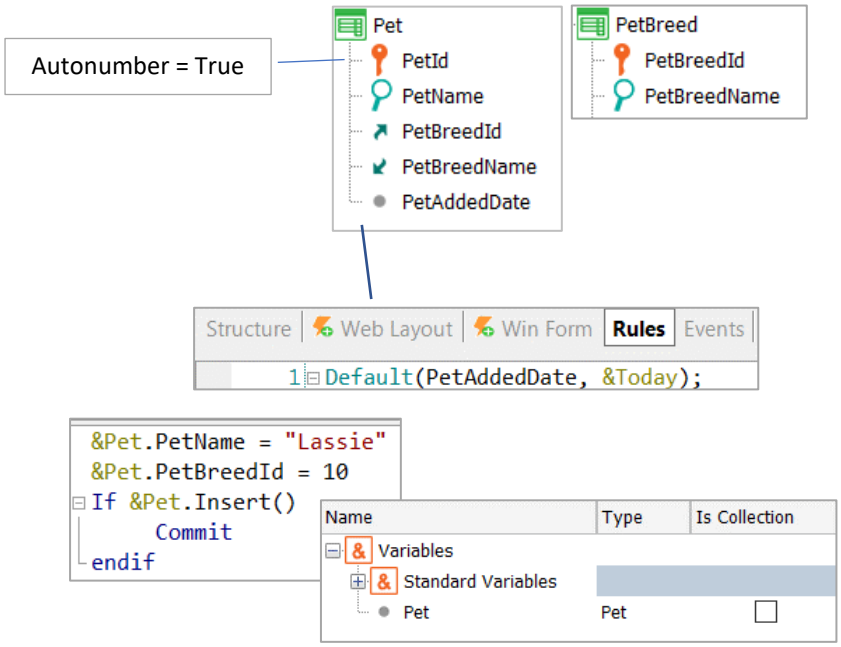
```
SDTCountry from PetBreed
{
     SDTCountryItem
     {
          CountryId
          CountryName
          CountryPetsQty = count(PetName)
     }
}
```

Output	
Infer Structure	No
Output	**SDTCountry**
Collection	**True**
Collection Name	**Countries**

22.4) None of the above options is correct.

23) Consider the transaction design shown below. The Pet transaction has been configured as a Business Component and the PetId attribute is autonumbered.
A new pet (Pet) called "Lassie" has to be added using a Business Component of Pet.

A procedure has been programmed with the following code. Select the correct option.



23.1 – The pet will only be inserted if breed 10 exists in the PetBreed table. Otherwise, the referential integrity will fail and it will not be inserted. If it is inserted, it will have an empty entry date, since no entry date was specified in the code.

23.2 – The pet will only be inserted if breed 10 exists in the PetBreed table. Otherwise, the referential integrity will fail and it will not be inserted. If it is inserted, its entry date will be today's date.

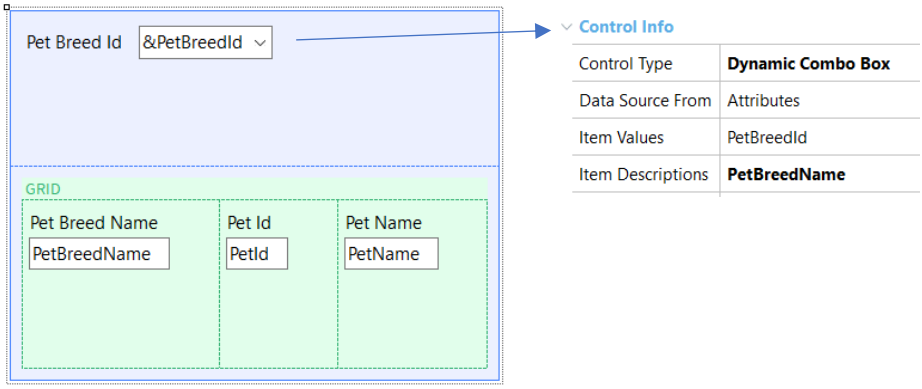
23.3 – The pet will always be inserted, even if there is no breed with identifier 10 in the PetBreed table, because Business Components do not control the referential integrity. It will have an empty entry date, since no entry date was specified in the code.

23.4 – The pet will always be inserted, even if there is no breed with identifier 10 in the PetBreed table, because Business Components do not control the referential integrity. Its entry date will be today's date.

24)

Consider the transaction design and the Web Panel layout shown below. A Web Panel has to be designed to show the identifiers and names of the pets (PetName) of a certain breed, selected by the user, as well as the name of the selected breed.

Determine the option that you consider correct.



24.1 – The Load event must be coded as shown:

```
Event Load
     For each Pet
          where PetBreedId = &PetBreedId
          &PetBreedName = PetBreedName
          &PetId = PetID
          &PetName = PetName
          Load
     Endfor
Endevent
```

24.2 – The Load event must be coded as shown:

```
Event Load
     For each Pet
          where PetBreedId = &PetBreedId
          Load
     Endfor
Endevent
```

24.3 – The Start event must be modified as shown:

```
Event Start
     PetBreedId = &PetBreedId
Endevent
```

24.4 – The following condition must be stated in the Grid:

```
Grid1's Conditions
    PetBreedId = &PetBreedId;
```

25) Consider the transaction design and the Web Panel layout shown below. A Web Panel has to be designed to show all the breeds (PetBreed), each one with their corresponding number of pets registered. If more than 10 pets are registered, the comment "Many pets" will be displayed. Otherwise, "Few pets" will be displayed.

Select the implementation option you consider correct.

25.1 –

```
Event Load
    For each PetBreed
        &Quantity = Count(PetName)
        If &Quantilty >10
            &Comment = "Many pets"
        Else
            &Comment = "Few pets"
        Endif
        Load
    Endfor
EndEvent
```

25.2 –

```
Event Load
      &Quantity = Count(PetName)
      For each PetBreed
         Where &Quantity > 10
             &Comment = “Many pets”
      When none
             &Comment = “Few pets”
     Endfor
EndEvent
```

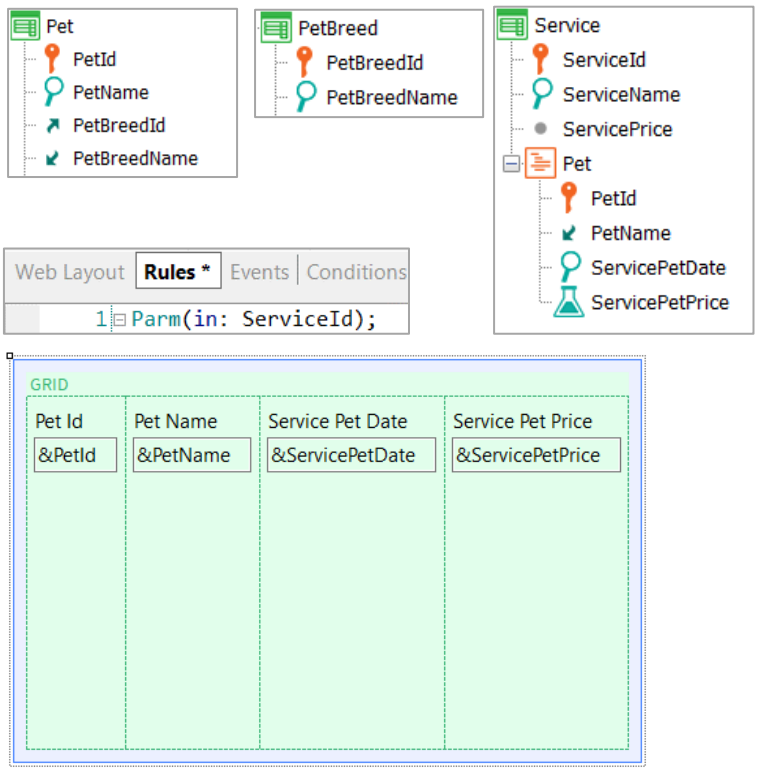
25.3 –

```
Event Load
      &Quantity = Count(PetName)
      If &Quantilty >10
          &Comment = “Many pets”
      Else
           &Comment = “Few pets”
      Endif
EndEvent
```

25.4 – None of the above options is correct.

26) In a Web Panel, the pets associated with a certain service received in a parameter are to be displayed.

Based on the detailed transaction and Web Panel design (what is not shown, such as properties, has not been modified except for the variables, which are all ReadOnly), determine the option you consider correct.



26.1 –

```
Event Load
     For each Service
          &PetId = PetId
          &PetName = PetName
          &ServicePetDate = ServicePetDate
          &ServicePetPrice = ServicePetPrice
          Load
     Endfor
Endevent
```

26.2 –

```
Event Load
     For each Service.Pet
          &PetId = PetId
          &PetName = PetName
          &ServicePetDate = ServicePetDate
          &ServicePetPrice = ServicePetPrice
          Load
     Endfor
Endevent
```

26.3 –

```
Event Load
     For each Service
          &PetId = PetId
          &PetName = PetName
          For each Service.Pet
               &ServicePetDate = ServicePetDate
               &ServicePetPrice = ServicePetPrice
               Load
          Endfor
     Endfor
Endevent
```

26.4 – None of the above options is correct.

ANSWERS

1) 3
Justification – *When an attribute that is a primary key of a transaction is added to another transaction, it becomes a foreign key and a strong 1-N relationship is established, where the entity containing the foreign key is the N side of the relationship. Since the question states that 1 breed has N pets (since many pets can be of the same breed), the correct answer is option 3, because the Pet transaction has PetBreedId as a foreign key. Option 1 cannot be the correct answer because the relationship goes the other way (one pet has many breeds) and option 2 represents the model of an N to N relationship.*

2) 1
Justification – *In the question, many-to-many relationship between foods and pets is established. To model an N-N relationship in GeneXus, we use a two-level transaction, where the second-level entity is also added as a separate transaction. The model that represents what is requested in the question is option 1. Option 2 is not correct because it represents a weak 1-to-N relationship (where Pet is the weak entity; note that Pet is not included as a separate transaction) and option 3 represents a strong 1-N relationship, where 1 food is consumed by N pets, but the other part of the relationship is not fulfilled.*

3) 1
Justification – *When we use a two-level transaction, we are modeling a 1 to N relationship where the subordinate entity (the one on the second level) is a weak entity. This means that to correctly identify this entity, we need the strong entity (the one on side 1 of the relationship), since without that association the weak entity loses meaning. In the example, special care makes sense for a certain pet, because that care is unique to it and another pet may require other care. Note that with the proposed transaction model, GeneXus creates 2 tables, one with the attributes of the first level (Pet) and another PetSpecialCare whose primary key is made up of PetId and SpecialCareId, demonstrating that to identify special care the pet identifier is needed.*

Therefore, the correct answer is option 1, since it states precisely that and the evidence is the compound primary key that will identify the special care. Option 2 states that care is not unique to a single pet and that would happen if we had modeled a strong 1-N relationship between Pet and SpecialCare using two separate transactions and with PetId as the foreign key. Option 3 is incorrect, as we have seen examples of weak 1-N modeled in this way, such as the case of the flight and its seats (weak entity).

4) 2
Justification – *To model a 1-to-1 relationship as described in the question, in GeneXus we use 2 transactions modeling a strong 1-N relationship (using a foreign key) and then creating a unique index by the attribute that is a foreign key. The unique index makes that attribute a candidate key and therefore it cannot be repeated. The option that represents this correctly is 2, where PetId is a foreign key in PetMedicalRecord, and in addition a unique index is created by that attribute in the PetMedicalRecord table. This means that there cannot be two medical records for the same pet. Option 1 is not correct because it models the strong 1-N relationship, but instead of the unique index by PetId, this attribute is made nullable, which does not prevent it*

from being repeatable if entered. Option 3 has a compound key, but this does not prevent that the same PetId can be used for different PetMedicalRecordId values.

5) 3
Justification – *This is a normalization exercise and we know that GeneXus normalizes tables in Third Normal Form. Option 1 has the COUNTRY, PETBREED and PET tables correct, but the SERVICE table is wrong because it includes the ServicePetsQty attribute, which is a formula and is not created in the table, and the PetId attribute, key of the second level. In addition, the SERVICEPET table is missing. In option 2 the COUNTRY, PETBREED, PET and SERVICEPET tables are correct, but the SERVICE table still includes the formula attribute ServicePetsQty. Option 3 is the only correct one.*

6) 2
Justification – *To answer this question correctly it is useful to create the Bachmann diagram for the tables. The extended table of SERVICEPET is the set of the attributes of the base table and all those that can be reached by N to 1 relationships directly or indirectly. Therefore, the extended table of SERVICEPET is: SERVICEPET (base table), SERVICE, PET, PETBREED and COUNTRY, which corresponds to option 2.*

7) 3
Justification – *The Nullable property applies mainly to foreign keys and if the value is Yes, it allows the record to be entered without giving a value to that foreign key and the referential integrity controls are not performed. If a value is entered, it is checked that the value exists as a primary key in the corresponding table. Therefore, the correct option is number 3, since when setting the PetBreedId attribute with Nullable to Yes, it is not mandatory to enter its value, but if it is entered, it is checked that it is valid, as stated in the question. Option 1 is not correct because if a value is entered, it is checked that it exists as a record in PETBREED. Option 2 is not correct because if a unique index is defined on the PetBreedId attribute in PET it prevents its value from being repeated, but it must always be entered.*

8) 1
Justification – *When GeneXus creates a table in the database, it automatically creates indexes by primary key and foreign keys. This is done to ensure the efficiency of the referential integrity checks that are automatically performed. The SERVICEPET table has a primary key made up of the ServiceId and PetId attributes, and also contains the ServicePetPrice attribute. In addition, the ServiceId attribute is a foreign key to the SERVICE table and the PetId attribute is a foreign key to the PET table.*

Therefore, GeneXus will create one index by primary key for the attributes ServiceId and PetId, one index by foreign key for ServiceId and another index by foreign key for PetId. The correct option is number 1, because it is the only one that contains the above mentioned indexes. Option 2 is incorrect because it is missing the two indexes by foreign key and option 3 is incorrect because it is missing the index by primary key.

9) 3

Justification – *When using the transaction form or the transaction as a business component, GeneXus performs the referential integrity checks automatically. This implies that if you want to delete a PetBreed record, it will first check that there are no records in Pet that contain that PetBreedId value as a foreign key. If found, a message will be shown and the deletion will not be allowed. This behavior is explained in option 3, which is the correct option. Option 1 is not correct because referential integrity checks are performed when the transaction form is being used as stated in the question. Option 2 is not correct because this behavior corresponds to cascading delete, which is not implemented in GeneXus.*

10) 1

Justification – *To establish the triggering order of rules and formulas, GeneXus creates an evaluation tree. However, since sometimes the order chosen by GeneXus does not suit certain requirements, we have available certain triggering moments that are always performed on the server, after Confirm is pressed on the transaction form. These triggering moments start with the "on" clause and in a two-level transaction we have moments before and after the header validation and save instances, before and after the validation and saving of each line, and before and after the Commit. The rule specified in the question has the on AfterInsert triggering moment and only includes attributes of the header (in this case, ServicePetsQty), so the rule will be triggered only when a header record is being inserted and immediately after the record is saved, but before starting to process the lines, i.e. the second level records.*

Since the initial requirement of the question is that the rule must prevent the recording of a service if it does not have registered animals, the correct option is number 1; with this implementation, the service record (header) will be saved first and only after that the rule will be triggered, so the rule does not perform the control preventing the saving of the record. Option 2 is not correct because it states that the rule will be triggered after validating and recording all the lines (animals), which as we saw is not the case. Option 3 is not correct because it states that the rule will be triggered before inserting the header (when the moment used is on AfterInsert). And option 4 is not correct, because when the rule is conditioned with a triggering moment, it will always be triggered on the server after pressing Confirm and never on the client.

11) 1

Justification – *Rule a) has only one header attribute (FoodId) and the AfterComplete triggering moment is the last one available, after the Commit is made. Rule b) has only one header attribute (FoodId) and the AfterInsert triggering moment causes it to be triggered immediately after the header record is inserted. Rule c) has a header attribute (FoodId) and uses the triggering moment on AfterLevel Level PetId, with PetId being an attribute of the lines (subordinate level). Therefore, as the server first performs the validation and saving of the header, followed by the validation and saving of each line, then finishes processing the lines and finally performs the Commit, the order of the rules will be b), c) , a), which corresponds to option 1 which is the correct option. The other options are incorrect and in particular option 4; due to the evaluation tree that GeneXus builds for the triggering (in which it takes into account the dependency between rules and formulas), we cannot assume that the rules are triggered in the order in which they are declared.*

12) 3

Justification – *The formula presented in the question is a horizontal formula, which is expressed in a declarative way, may contain one or more conditional assignments and will execute only the assignment that has the condition that is met. This type of formula navigates a single record and the extended table, so the attributes mentioned must all belong to the extended table of the table associated with the formula attribute.*

The formula is assigning to the ServicePetPrice attribute the value 0 if the day to perform the service matches the day when the pet was registered in the store and also the pet is VIP, and will assign the price ServicePrice if the previous conditions are not met (since this is conditioned with OTHERWISE). Therefore, the formula is correct and meets the requirement, so the correct answer is 3. Option 1 is not correct because it does not take into account that the attributes PetAddedDate and PetIsVIP belong to the extended table of SERVICEPET which is what is required. Option 2 is not correct either because it establishes that the IF must be used with ELSE when this type of clause corresponds to procedural programming and not to declarative programming, which is the one used in the definition of formulas.

13) 3

Justification – *The length of the pet's membership in years is calculated as the difference between the current year and the year of the pet's joining date. The joining date is given by the PetAddedDate attribute and the pet's name by the PetName attribute. To list all the animals in the pet store with their name and membership duration, a layout has been designed where in PrintBlock1 the PetName attribute and the &MembershipDuration variable are included and to run through the pet table, a For each Pet command is used. Option 1 is not correct because the instruction to print PrintBlock1 is outside the For each and although the variable &MembershipDuration will have the value of the last PET record, the printBlock cannot be instantiated because it is outside the For each, so it cannot be printed and the list does not even show the page. Option 2 is not correct either because now the PetAddedDate attribute is outside the For each and cannot be instantiated, so the &MembershipDuration variable will be set to the value of the current year (e.g.: 2024). The For each will print all the names of the pets, with the value of the current year in the membership duration. The correct option is 3, whose For each includes the calculation of the &MembershipDuration variable and the printing of PrintBlock1 with the required data in the list. This will list all the pets in the store with their name and membership duration.*

14) 3

Justification – *Both the regular and the substitute veterinarian should be able to be chosen from those stored in the VET table, and if the identifier is entered, check that this value exists as an identifier of a record in the table. Option 1 is not correct because although the value of VetId and VetAssistantId can be chosen independently, it is logical that the table with the veterinarians should be a single table and not have two separate tables with the names of the veterinarians, as is the case of the VETASSISTANT table where there may even be veterinarians in VET that are not in VETASSISTANT and vice versa. Option 2 is not correct either, because the VetAssistantName attribute is in a different subtype group than the VetAssistantId, so the name of the veterinarian will not be inferred when the identifier is chosen. Option 3 is the correct one, because VetAssistantId is a subtype of VetId, so it will behave as such; it will be a foreign key in Pet and the referential integrity checks will be performed on Vet. Moreover, as the subtype group includes VetAssistantName as a subtype of VetName, when choosing the value of VetAssistantId*

the value of VetAssistantName will be inferred, as can be seen in the image of the Pet transaction where the VetAssistantName attribute is shown as inferred.

15) 3

Justification – *The question requires that we can enter the country of the pet, regardless of the country of the breed. If we look at the PetBreed transaction, the CountryId and CountryName attributes are included. When PetBreedId is included in Pet, if we add the CountryId attribute, it will not be a foreign key to Country, but it will be an inferred attribute of PetBreedId. In other words, it will take the value of the country of the breed, it will be read only and we will not be able to choose a country for the pet. Option 1 is incorrect because even if the country attributes appear first before the breed attributes, the country will still be inferred from the breed as can be seen in the image, where CountryId has a down arrow indicating that it is an inferred attribute. Option 2 adds one more PetCountry transaction, with the attributes PetCountryId and PetCountryName, which it then adds to the Pet transaction. It also adds a group of subtypes where PetCountryId is a subtype of CountryId and PetCountryName is a subtype of CountryName, thus establishing a specialization relationship regarding CountryId and CountryName, respectively. Although this solution works, it is not adequate because it adds unnecessary objects to the solution, such as the PetCountry transaction or the specialization relationship. The correct solution is 3, which includes in the Pet transaction the attributes PetCountryId defined as a subtype of CountrId and PetCountryName defined as a subtype of CountryName, in the respective group of subtypes. This solution makes it possible to enter a country for the pet independent of the country of the breed, using PetCountryId as a foreign key and PetCountryName as an inferred attribute, without the need to add unnecessary objects.*

16) 1

Justification – *The navigation list of the procedure shows two nested For each commands, the external one with base table COUNTRY (the For each above) and the nested one (shown indented) with base table SERVICE. If we analyze the transactions, we see that there is no relationship between the COUNTRY and SERVICE tables, so we know that if we have two nested For each commands, with different base table (in this case COUNTRY and SERVICE), the resulting navigation will be a Cartesian product; that is, for each record in the COUNTRY table, all the existing records in the SERVICE table will be listed. We can see this in the navigation list because in the nested level that navigates the SERVICE table there is no indication of filtering by context. If GeneXus had found any relationship between COUNTRY and SERVICE, to the right of the table there would be text indicating an implicit filter such as: CountryId = @CountryId. Since this is not the case, we can be sure that GeneXus did not find any relationship between the base tables of the For each commands and therefore the resulting navigation will be a Cartesian product. The correct answer is 1.*

17) 4

Justification – *The question asks to run through the PET table, filtered by the pet received by parameter and then run through the PETSERVICE table to bring the services associated with that pet. If we run through the PET table, the PETSERVICE attributes do not belong to the extended PET table, so we need two nested For each commands, one running through PET and the nested one running through PETSERVICE. In option 1 the PetId attribute is received by parameter, so the navigations will be filtered automatically and only records whose PetId is equal to the value*

received by parameter will be shown. Then we have 2 nested For each commands, both with base table PETSERVICE (indicated by the use of the base transaction Pet.Service in both), which will cause a control break, but as the order by PetId is missing in the external For each that causes the grouping, the pet names repeated by each service will be listed, so option 1 is not correct. In option 2, three nested For each commands are used, which is totally unnecessary since we only need two and, besides, a For each by PETBREED is not needed at all, since that attribute belongs to the extended table of PET, so option 2 is not correct either. Option 3 is not correct because it implements a single For each with PET base table and as we explained above the attributes ServiceName and SevicePetPrice cannot be reached because they do not belong to the extended table of PET. Option 4 has the two required nested For each commands, the external one by PET and the nested one by PETSERVICE. The external For each does not include the Where clause to filter by the PetId value because the parameter is received in the PetId attribute, which as we have already seen acts as an equality filter by the value received. Therefore, in PET only the corresponding pet record will be run through and since the For each commands are nested and the PET and PETSERVICE tables are related, only the services corresponding to the pet received by parameter will be listed, so option 4 is the correct one.

18) 4

Justification – *Since the question asks to list all the registered breeds regardless of whether there are pets of that breed, and the associated pets for each breed that has pets, we are being asked for a JOIN. This involves two nested For each commands, the external one with base table PETBREED (so that it runs through all the registered breeds) and the nested one with base table PET. For there to be a JOIN it is also required that the base tables are related, which is the case between PETBREED and PET. If we look at the options, option 1 is incorrect because it shows two parallel For each commands instead of being nested. Option 2 does show two nested For each commands but on the same PET base table, which is not what we need, so it is not correct either. Option 3 has only one For each with a PET base table, which means that no groups will be formed by breed, but the name of the breed will be repeated for each PET record, which is not what we want to show the pets of each breed, so option 3 is not correct either. If we analyze option 4, we see that we now have two nested For each commands, the external one with base table PETBREED and the one nested by PET, i.e. different but related base tables. All the breeds registered in PETBREED will be listed and for each one only the pets of that breed, since due to this relationship a JOIN will be performed. Option 4 is the correct one.*

19) True

Justification – *The implementation shows a For each with PET base table, which will run through this table only for pets with breed PetBreedId = 4 or PetBreedId = 7 and also the year of the entry date is 2020. This meets the requirements stated in the question. The doubt we may be left with is whether or not all the attributes included in the Where clauses and in the printblock belong to the PET extended table, which is the base table of the For each. We check that and see that it is fulfilled, so the implementation is correct and the answer is TRUE.*

20) 3

Justification – *Since the question asks to list only breeds that have registered pets, each with their corresponding pets, we are being asked for a CONTROL BREAK. We will not be able to list*

the breeds from the PETBREED table, because all the breeds are there, whether they have registered pets or not, so we must retrieve the breeds from the PET table. As in this case we retrieve the breed of each pet, it will be a breed that has registered pets. To implement a control break between breeds and pets, we need two nested For each commands, both with the same base table and with the ORDER clause by the attribute we want to do the break (grouping by PetBreedId) in the outer For each. Option 1 is not correct because there is only one For each and the pets are not grouped by breed. Option 2 has the external For each with PETBREED base table which we have already seen is not correct. Option 3 complies with everything we need to correctly implement the control break, so option 3 is the correct one. Option 4 lacks the order that allows the groupings, so it is incorrect.

21) 3
Justification – *The question asks to list all the veterinarians who are in charge of at least one pet, and in case there is no veterinarian that has pets assigned, a message should be displayed. For that, we must run through the VET table filtering those records where the pet count is greater than or equal to 1. All the options do this correctly, the problem is how we implement the second part, which is to display a message if there is no vet with pets. Option 1 is not correct because it uses an ELSE which is not part of the For each syntax. Option 2 after printing the printBlock1 uses a UNIQUE clause, but this clause is used to avoid repetitions and does not apply to this case so it is not correct either. Option 3 is correct because it uses WHEN NONE, which is precisely the appropriate clause in the syntax of the For each to use in these cases; that is, when there is no record that meets the conditions of the filters.*

22) 2
Justification – *The SDTCountry structured data type is a collection and to load this collection, the Data Provider must run through the COUNTRY table to obtain the country identifier, the name and the number of pets. For that, the Data Provider must include the clause from Country to indicate the base transaction, whose base table will be the table run through by the Data Provider. In addition, the Data Provider must have the Collection property set to False, since the output will be the SDTCountry structure, which is already a collection. The three options have the same implementation for the SDTCountryItem and this implementation is correct. Option 1 is not correct because the base transaction is Pet and not Country. Option 2 is correct because it meets the requirement that the COUNTRY base table is run through and the Collection property is set to False. Option 3 is not correct because it tries to run through the PETBREED table.*

23) 2
Justification – *To insert a record in the PET table using the &Pet variable defined as Pet business component, the procedure should assign values to PetName, PetBreedId and PetAddedDate, since PetId is not necessary because it is autonumbered. However, since the Pet transaction has defined the Default rule that assigns the current date to PetAddedDate, it is not necessary to assign this value in the implementation of the procedure. And since we know that GeneXus controls the referential integrity when using a business component, the value of PetBreedId must be valid; that is, it must exist as a primary key in the PETBREED table. After assigning these values, the procedure source invokes the Insert() method and if the result is true, the Commit is performed, which is correctly implemented. The first part of option 1 is correct when it talks about the PetBreedId value, but it is not true that if the record is inserted the PetAddedDate is*

empty, since the Default rule is triggered and this value is assigned. Option 2 is the correct option because it explains what will happen when the implemented code is executed. Option 3 is not correct because it is not true that if we use BC the referential integrity checks are not implemented and also because of the rule the date will not be empty. Option 4 is not correct either, because although it is correct that today's date is assigned, the statement that when using BC the referential integrity controls are not implemented is not true.

24) 4

Justification – *The Grid has a base table because it has the attributes PetBreedName, PetId and PetName. To filter the Grid with base table, we use its Conditions property with the condition that the PetBreedId is equal to the value selected in the &PetBreedId variable. Option 1 is incorrect because when the Grid has a base table, GeneXus creates an implicit For each on the PET table that allows getting the values of PetId and PetName, as well as PetBreedName by the extended table, and therefore it is not necessary to implement a For each in the Load event to retrieve that data. Option 2 is not correct either because to filter by the value of PetBreedId selected in the variable, the Conditions of the Grid must be used and not a For each in the Load event, because this For each would be nested to the implicit For each and this where will not filter this implicit For each. Option 3 is not correct either because the Start event is triggered only once and with this assignment the implicit For each will not be filtered by the value of the selected breed. Option 4 is the correct option, because with this condition added to the properties of the Grid, GeneXus will filter the records of the grid by the selected value of &PetBreedId.*

25) 3

Justification – *Since the Grid has a PETBREED base table, due to the value of the Base Trn property, we know that the Load event will be triggered N times, once for each record of the PETBREED table. It is in the Load event where we must obtain the number of pets –through the formula count(PetName)– and also depending on this number assign the value of the variable &Comment. Option 1 is not correct because being a Grid with a base table an implicit For each will be created and it is not necessary to add a For each in the Load event. Option 2, although it tries a different implementation from option 1, is not correct either because it also adds an unnecessary For each, which will also be nested to the implicit one. Option 3 is the correct option, since for each PETBREED record the Load event will be triggered and the number of pets for the breed being run through will be calculated; then, if the number is greater than 10 the message "many pets" will be assigned and for the opposite case "few pets". Option 4 is therefore not correct either.*

26) 2

Justification – *Since there are no attributes in the Grid and it is made clear that the rest of the Grid properties and Web Panel sections have default values, we can state that the Web Panel does not have a base table. In this case, we know that the Load event will be triggered only once; therefore, in this event we must implement a For each that runs through the SERVICEPET table so that through its extended table we can retrieve the values of the attributes PetId, PetName, ServicePetDate and ServicePetPrice, as well as assign the values of the Grid variables, using the Load command to copy the successive values to each line of the Grid. Option 1 is not correct because the base transaction of the For each is Service, and from the SERVICE base table it is not possible to reach the attributes we need. Option 2 is the correct option because by assigning the*

base transaction Service.Pet, the base table of the For each will be SERVICEPET and will allow us to obtain the necessary values to load the Grid. Option 3 is not correct because For each Service is not necessary, since with a single For each with SERVICEPET base table we can retrieve all the attributes through its extended table. Option 4 is not correct either.