Examen Analista Junior GeneXus 18

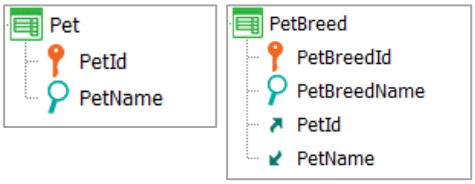
Realidad: Tienda de Mascotas.
Sobre las preguntas de múltiple opción:

- Hay una sola opción correcta.
- Este examen NO resta puntos por respuestas incorrectas.

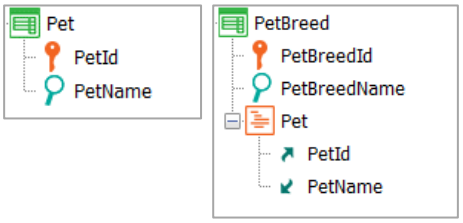
1) Se cuenta con una aplicación GeneXus para la gestión de una tienda de mascotas.

Sabiendo que una mascota (Pet) pertenece a una raza (PetBreed), y que muchas mascotas pueden ser de la misma raza, determina el diseño de transacciones que consideres correcto.

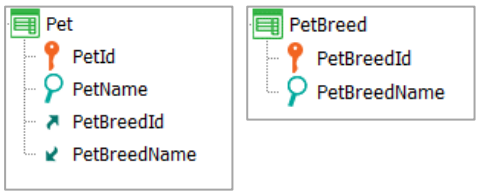
1.1 -



1.2 -



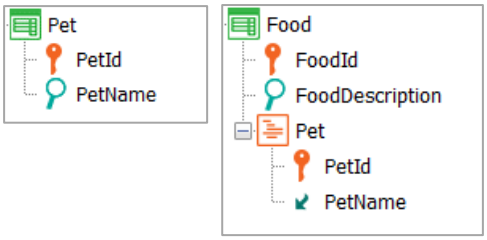
1.3 –



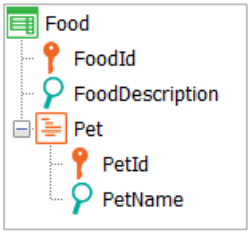
1.4 – Ninguna de las opciones anteriores es correcta.

2) Sabiendo que una mascota (Pet) puede consumir varios alimentos (Food), y que un mismo alimento puede ser consumido por varias mascotas, determina el diseño de transacciones que consideres correcto.

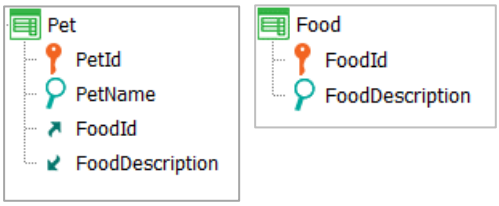
2.1 –



2.2 –



2.3 –



2.4 – Ninguna de las opciones anteriores es correcta.

3) Considera el diseño de la transacción que se muestra y determina lo que consideres correcto.

3.1 – Toda mascota (Pet) tiene asociado un conjunto de cuidados especiales (SpecialCare) que se identifican como únicos de esa mascota.

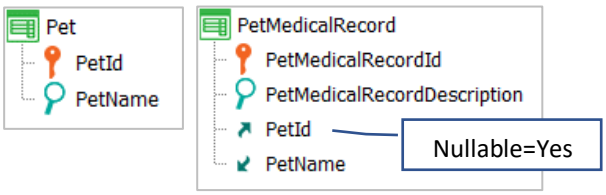
3.2 **–** Toda mascota (Pet) tiene asociado un conjunto de cuidados especiales (SpecialCare), y esos mismos cuidados no son propios de una única mascota, sino que pueden aplicarse a otras mascotas.

3.3 **–** El diseño no es válido. No es posible definir una transacción de dos niveles sin que la entidad del segundo nivel deba definirse además como una transacción en sí misma.

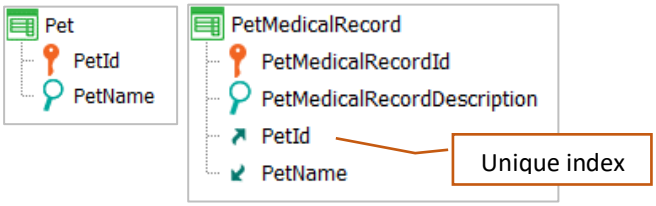
3.4 – Ninguna de las opciones anteriores es correcta.

4) Sabiendo que una mascota (Pet) tiene una única ficha médica (PetMedicalRecord), y que a su vez esa ficha médica es solamente de esa mascota, determina la opción que consideres correcta:

4.1 **–**



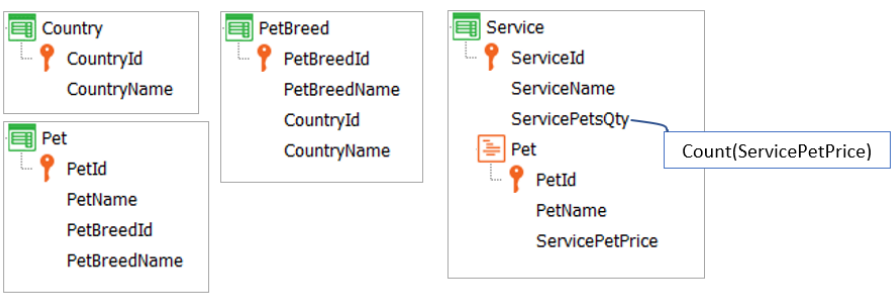
4.2 **–**



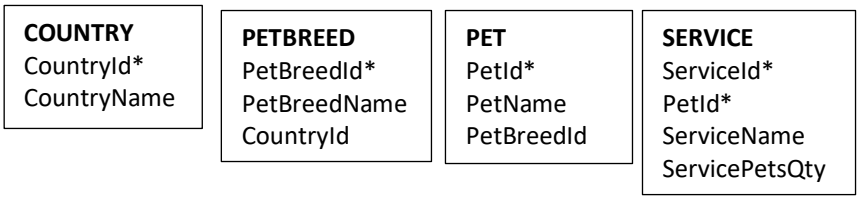
4.3 **–**

4.4 **–** Ninguna de las opciones anteriores es correcta.

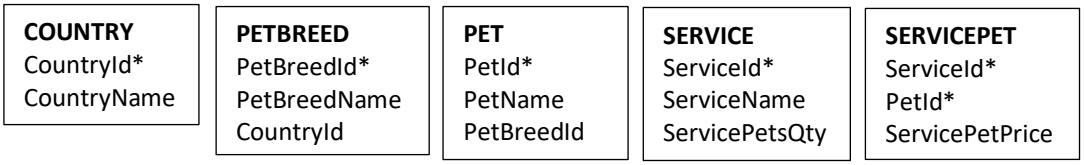
5) Considera el diseño de transacciones que se muestra y determina la estructura física de las tablas que GeneXus creará.



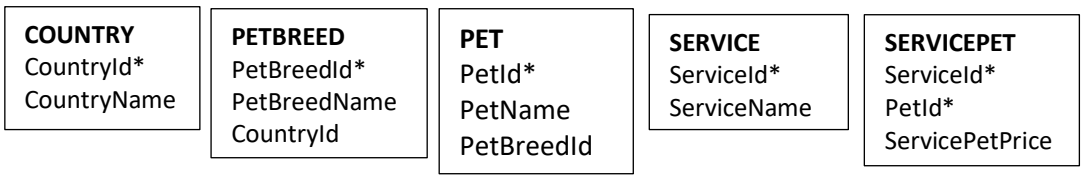
5.1 -



5.2 –

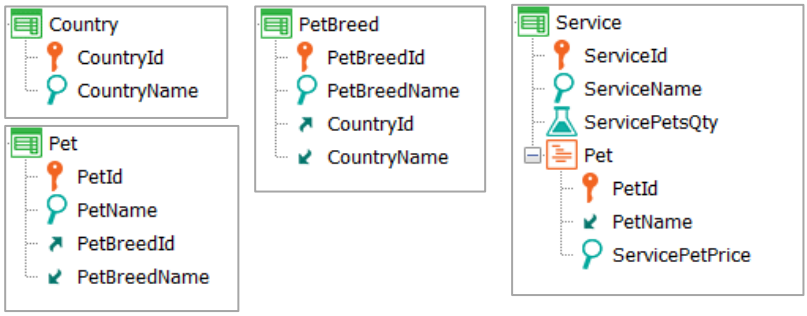


5.3 –



5.4 – Ninguna de las opciones anteriores es correcta.

6) A partir del diseño de transacciones que se muestra, determina la tabla extendida de la tabla SERVICEPET.



6.1 – SERVICEPET, SERVICE, PET

6.2 – SERVICEPET, SERVICE, PET, PETBREED, COUNTRY

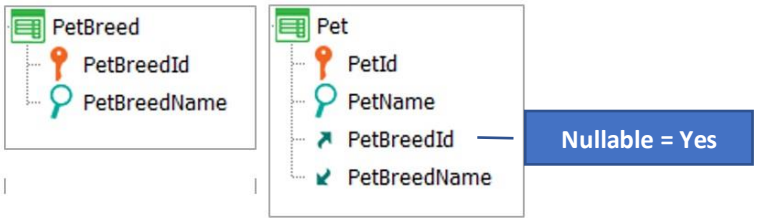
6.3 – SERVICEPET, SERVICE, PET, PETBREED

6.4 – SERVICEPET

7) Si bien toda mascota (Pet) tiene una raza (PetBreed), a veces no es posible determinar esa raza, y por lo tanto no se considera un dato obligatorio al registrar una mascota.

En caso de indicar una raza (PetBreedId), dicho valor debe ser válido.

A partir del diseño de transacciones que se muestra, determina lo que consideres correcto:



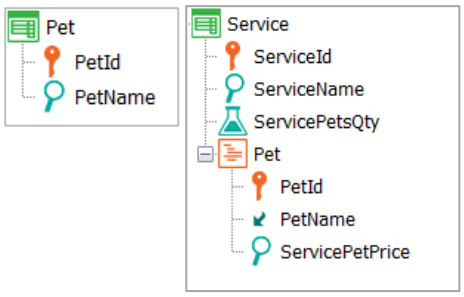
7.1) La implementación no resuelve el requerimiento. Al indicar que PetBreedId acepta nulos, GeneXus no realizará los controles de integridad referencial. Se podrá registrar una mascota sin raza, pero si se indica una raza entonces no se controlará si existe como registro en PETBREED.

7.2) La implementación no es correcta. Debe definirse un índice unique sobre el atributo PetBreedId en PET.

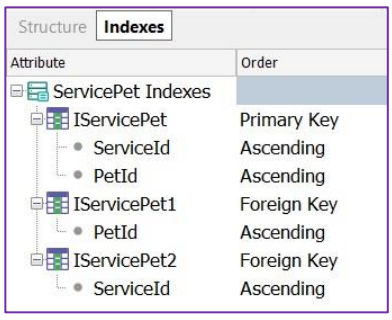
7.3) La implementación es correcta y resuelve el requerimiento.

7.4) Ninguna de las opciones anteriores es correcta.

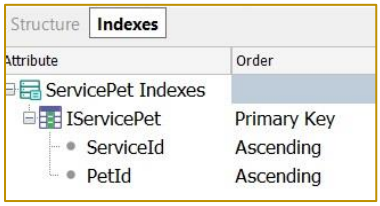
8) A partir del diseño de transacciones que se muestra, determina los índices automáticamente creados por GeneXus en la tabla SERVICEPET.



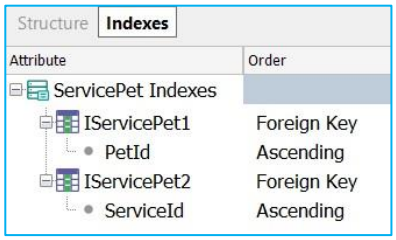
8.1)



8.2)

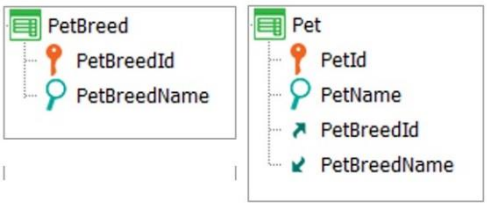


8.3)



8.4) Ninguna de las opciones anteriores es correcta.

9) Considera el diseño de transacciones que se muestra, y determina qué sucederá al momento de intentar eliminar una raza (PetBreed) utilizando el form de la transacción PetBreed.



9.1) GeneXus la eliminará sin realizar ningún tipo de control.

9.2) Automáticamente GeneXus eliminará primero todos los registros en Pet que tengan el PetBreedId como FK, y luego eliminará el correspondiente registro de PetBreed.

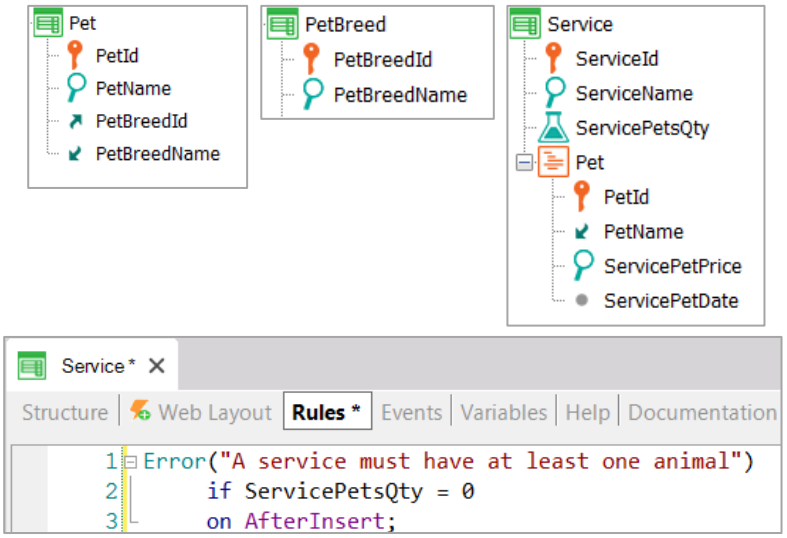
9.3) GeneXus controlará que no existan en Pet registros que tengan el PetBreedId como FK. En caso de existir, emitirá un mensaje indicando que existen registros relacionados y no realizará ninguna acción.

9.4) Ninguna de las opciones anteriores es correcta.

10) En el siguiente diseño de transacciones, la transacción Service tiene un atributo fórmula, ServicePetsQty, que cuenta la cantidad de animales registrados para determinado servicio en determinada fecha.

Se necesita controlar que un servicio nunca quede registrado sin ningún animal asociado. Para realizar este control se utiliza la regla Error mostrada a continuación.

Determina lo que consideres correcto:



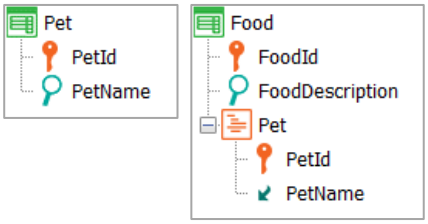
10.1 – La regla no cumple con el requisito solicitado, ya que será disparada en el servidor luego de que se hayan grabado en la base de datos los datos del cabezal (Service) y antes de que comiencen a grabarse los animales (Pet).

10.2 – La regla no cumple el requisito solicitado ya que será disparada en el servidor, luego de haber grabado en la base de datos los datos del cabezal (Service) e inmediatamente después de que se grabe el último animal (Pet).

10.3 – La regla no cumple el requisito solicitado, ya que será disparada en el servidor, exactamente antes de comenzar a grabar los datos del cabezal (Service).

10.4 – La regla cumple con el requisito solicitado, ya que será disparada en el cliente, antes de presionar Confirmar.

11) Considera las transacciones que se muestran y determina el orden en el cual se dispararán las reglas declaradas en la transacción Food.



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)

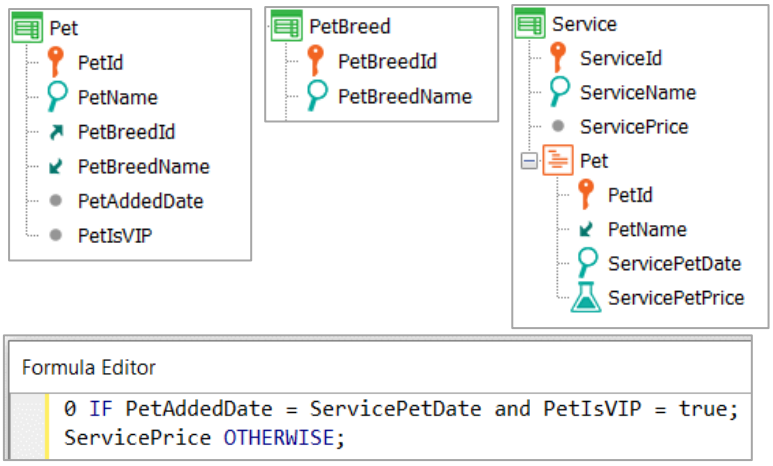
11.3 – c), a), b)

11.4 – Las reglas se disparan en el orden en el que son declaradas.

12) En la tienda de mascotas existen mascotas VIP, es decir, mascotas que cuentan con ciertos beneficios.

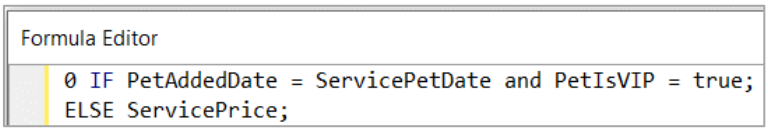
Al asociar una mascota a determinado servicio, si la mascota es VIP y el día para realizar el servicio coincide con el día en el que la mascota fue registrada en la tienda, el mismo no tendrá costo. De lo contrario, asumirá el precio base del servicio.

Determina lo que consideres correcto a partir del cálculo asociado al atributo ServicePetPrice.



12.1 – La implementación de la fórmula es incorrecta, ya que no es posible utilizar los atributos PetAddedDate y PetIsVIP en la misma, dado que estos no se encuentran en la estructura de la transacción Service (ni en el cabezal, ni en el subnivel Pet).

12.2 – La sintaxis de la fórmula es incorrecta, ya que cuando se usa la estructura IF debe utilizarse ELSE para referirse a los demás. La implementación válida sería así:

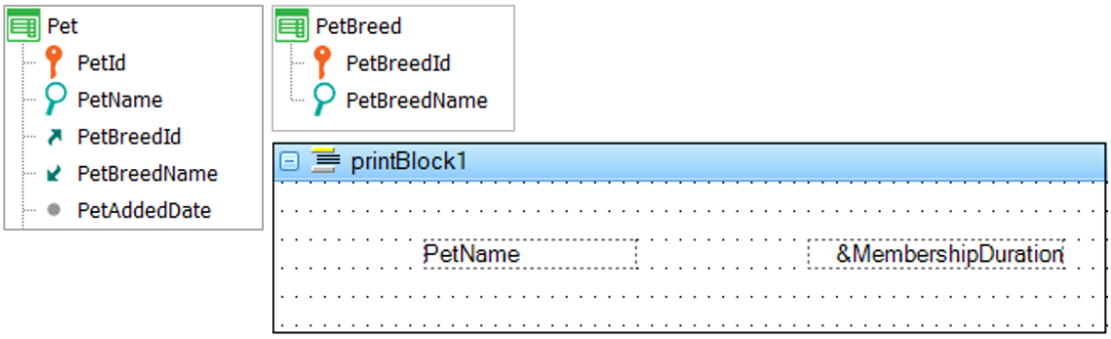


12.3 – La implementación de la fórmula resuelve el requerimiento solicitado.

12.4 – Ninguna de las anteriores es correcta.

13) Se necesita mostrar en un listado todos los animales (Pet) de la tienda de mascotas, con su nombre (PetName) y su antigüedad.

Observa la siguiente transacción y el Layout del procedimiento. ¿Cuál debe ser la implementación del 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 –

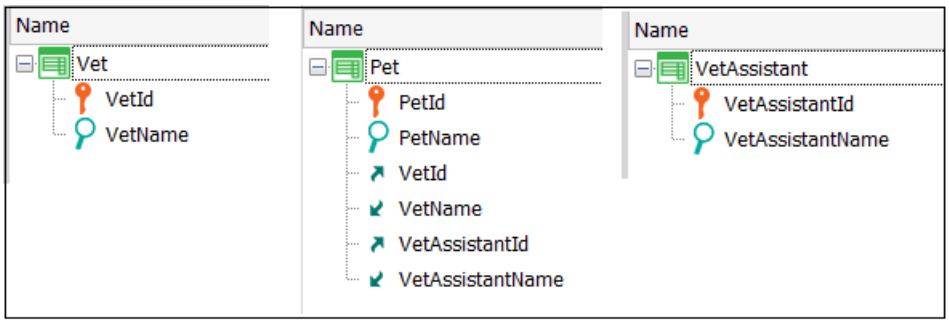
```
For each Pet
      &MembershipDuration = &Today.Year() - PetAddedDate.Year()
      print PrintBlock1
Endfor
```

13.4 – Ninguna de las opciones anteriores está correcta.

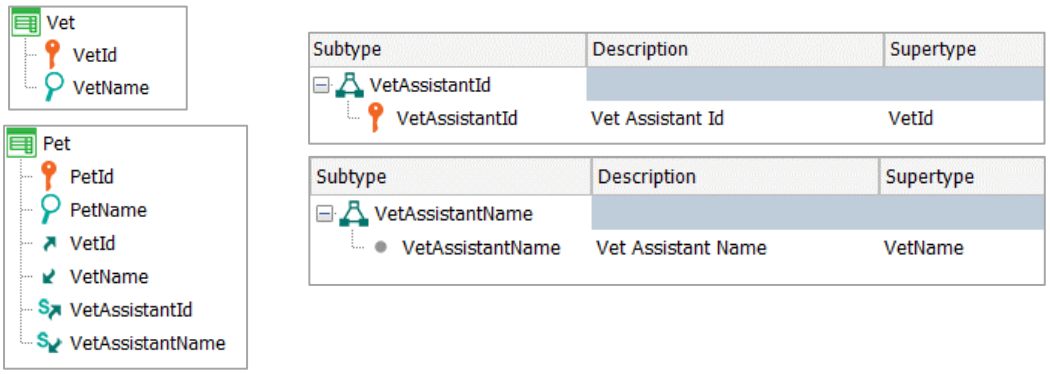
14) Si bien toda mascota (Pet) tiene un veterinario de cabecera, se necesita registrar otro veterinario asistente para los casos en los que el veterinario de cabecera no esté disponible.

Determina cuál de los siguientes diseños de transacciones (y de grupos de subtipos si se incluyen), es el adecuado para modelar la realidad descripta anteriormente.

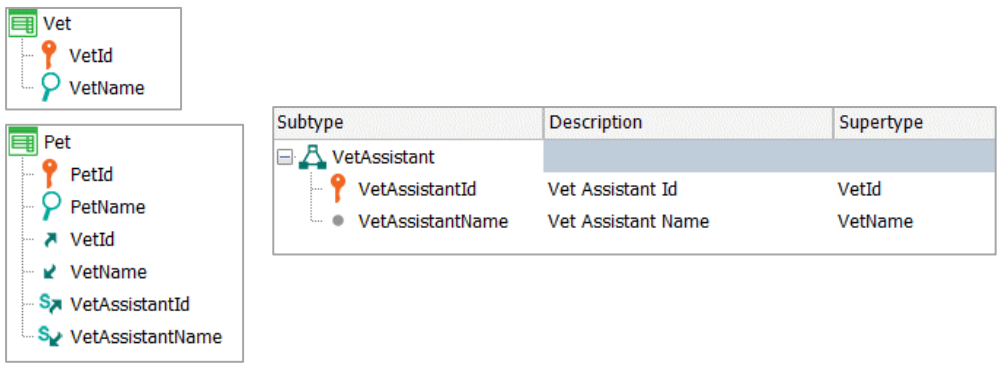
14.1 –



14.2 –



14.3 –

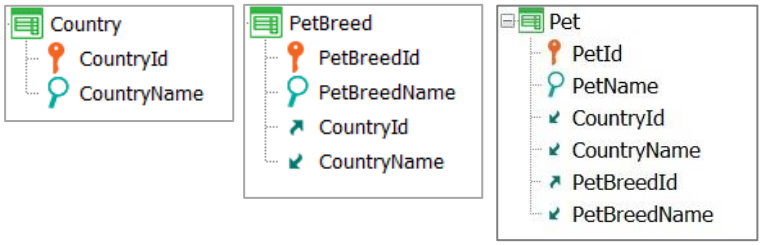


14.4 – Ninguna de las opciones anteriores es correcta.

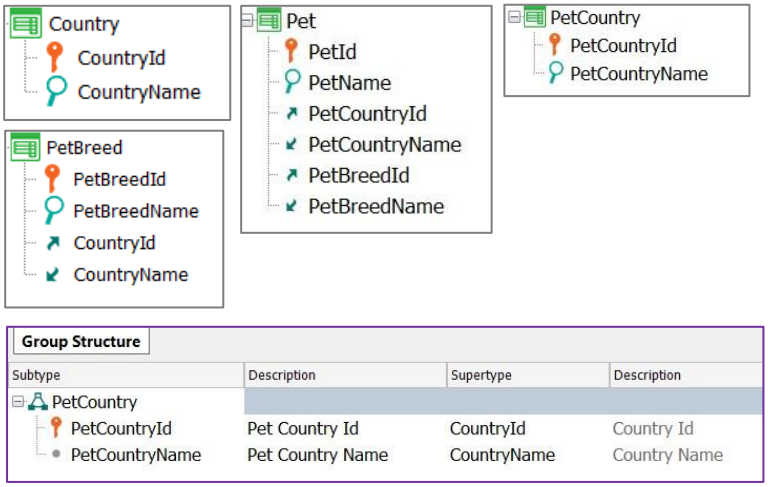
15) Si bien toda raza (PetBreed) se asocia con un determinado país de origen (Country), interesa registrar también el país donde la mascota (Pet) fue adquirida (adoptada) por sus dueños.

Determina la opción de implementación que consideres correcta:

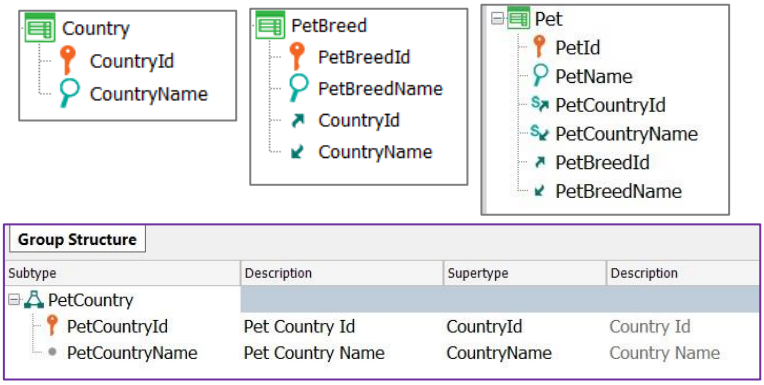
15.1) Simplemente por el orden en el cual se sitúan los atributos en la estructura de la transacción Pet, se podrá registrar el país donde fue adoptada la mascota.



15.2) Se debe definir la transacción PetCountry, y declarar el correspondiente grupo de subtipos.



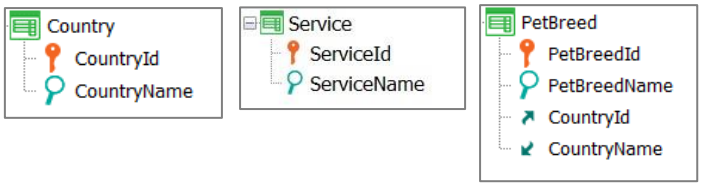
15.3) Se debe definir los atributos y grupo de subtipos como se muestra:

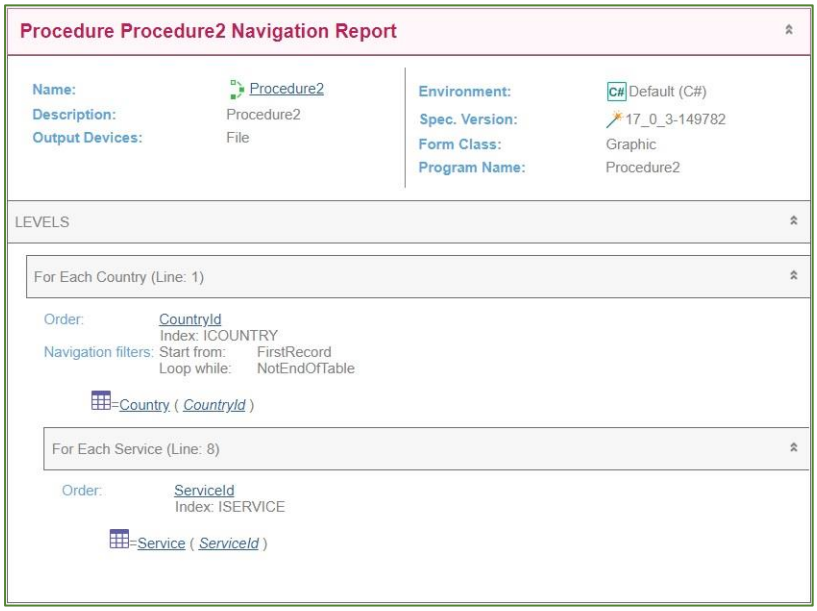


15.4) Ninguna de las opciones anteriores es correcta.

16) Observa el diseño de transacciones y el listado de navegación que se muestra.

¿Qué implementa?





16.1) Producto Cartesiano

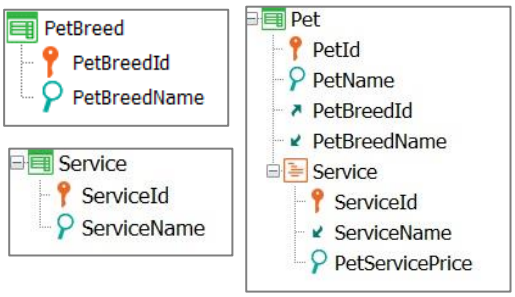
16.2) Corte de Control

16.3) Join

16.4) Ninguna de las opciones anteriores es correcta.

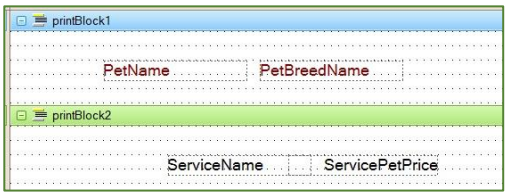
17) Considera el diseño de transacciones que se muestra. Se necesita listar la información completa (cabezal y líneas) de una determinada mascota (Pet) recibida por parámetro.

Determina la implementación que consideres correcta:



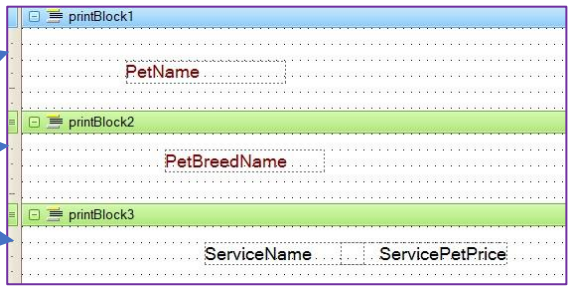
17.1) **Parm**(**in**: PetId);

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



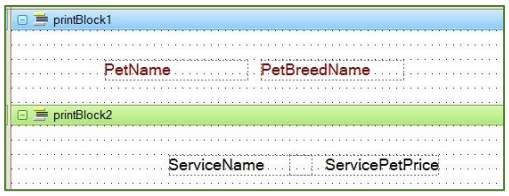
17.2) **Parm**(**in**: &PetId);

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



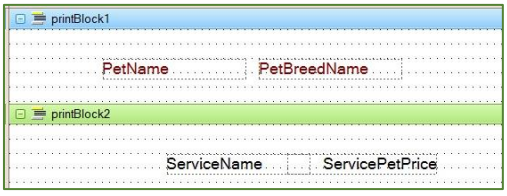
17.3) **Parm**(**in**: &PetId);

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



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

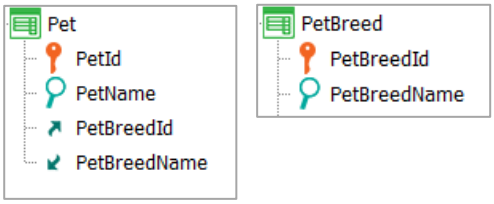
```
For each Pet
    Print printBlock1
    For each Pet.Service
      Print printBlock2
    Endfor
Endfor
```



18) Considere el diseño de transacciones que se muestra. Se necesita definir un listado que muestre todas las razas (PetBreed) y para cada una, la lista de mascotas (Pet) pertenecientes a la misma.

Interesa ver en el listado todas las razas, independientemente de que tengan mascotas registradas de esa raza o no.

Determina la opción de implementación que consideres correcta para resolver adecuadamenete el requisito descripto.



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 –

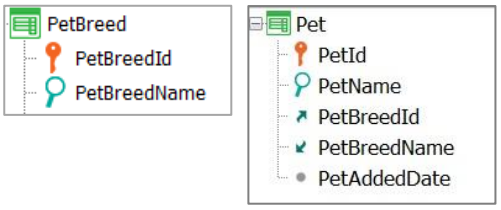
For each Pet
Print printblock1
Print printblock2
Endfor

18.4 –

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

19) Considera el diseño de transacciones que se muestra. Se necesita listar las mascotas (Pet) de raza "Beagle" (PetBreedId = 4) y "Cocker" (PetBreedId = 7) que fueron registradas (PetAddedDate) en el año 2020.

Determina si la implementación propuesta es correcta o no.

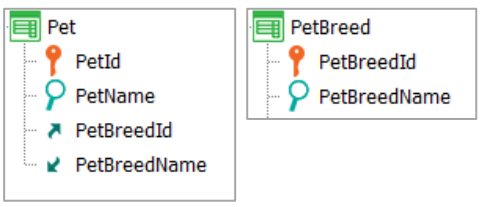


For each Pet
Where PetBreedId = 4 **or** PetBreedId = 7
Where PetAddedDate.Year() = 2020
Print printBlock1
Endfor

☐ Verdadero ☐ Falso

20) Considera el diseño de transacciones que se muestra. Se necesita definir un listado que muestre todas las mascotas (Pet) agrupadas por raza (PetBreed).

Interesa que salgan en el listado solamente aquellas razas que tengan mascotas registradas.



20.1 –

For each PetBreed order PetBreadId
Print printblock1
Print printblock2
Endfor

20.2 –

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

20.3 –

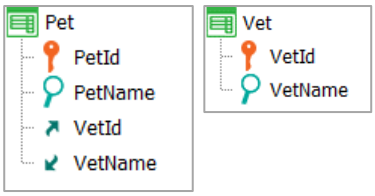
For each Pet order PetBreedId
Print printblock1
For each Pet
Print printblock2
Endfor
Endfor

20.4 –

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

21) Considera el diseño de transacciones que se muestra.

Se necesita definir un listado que muestre todos los veterinarios (Vet) registrados que tengan al menos una mascota a su cargo. Si no hay ningún veterinario que tenga al menos una mascota, un texto debe informarlo.
Determina la opción de implementación que consideres correcta para resolver el requisito descripto.



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
```

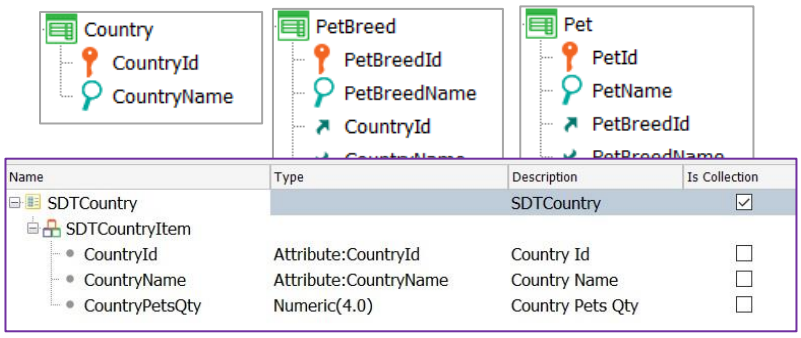
21.3 –

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

21.4 – Ninguna de las opciones anteriores es correcta.

22) Considera el diseño de transacciones y la definición del tipo de datos estructurado SDTCountry, que se muestran. Se necesita diseñar un Data Provider que cargue una colección de países (Country), cada uno con la cantidad de mascotas (Pet) de razas (PetBreed) de dicho país.

Determina la opción de implementación que consideres correcta.



22.1)

```
SDTCountry from Pet
{
     SDTCountryItem
     {
          CountryId
          CountryName
          CountryPetsQty = 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)
     }
}
```

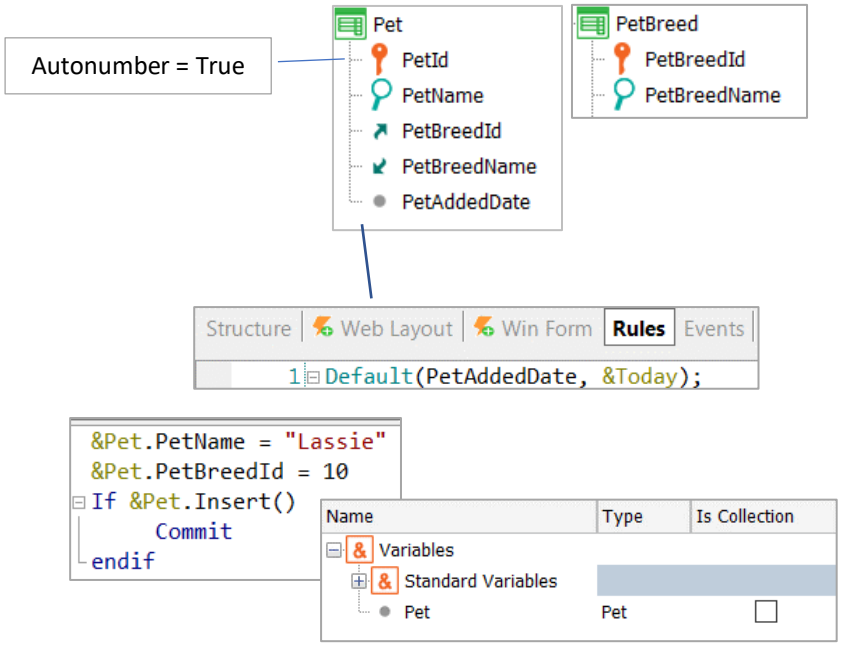
Output	
Infer Structure	No
Output	**SDTCountry**
Collection	**True**
Collection Name	**Countries**

22.4) Ninguna de las opciones anteriores es correcta.

23) Considere el diseño de la transacción que se muestra. La transacción Pet fue configurada como Business Component y el atributo PetId es autonumerado.

Se desea insertar una nueva mascota (Pet) de nombre "Lassie", utilizando un Business Component de Pet.

Se ha programado un procedimiento con el siguiente código. Indique la opción que considera correcta.



23.1 **–** La mascota solo será insertada si existe la raza 10 en la tabla PetBreed. De lo contrario, fallará la integridad referencial y la misma no se insertará. Si es insertada, quedará con fecha de ingreso vacía, pues no se especificó ninguna en el código.

23.2 **-** La mascota solo será insertada si existe la raza 10 en la tabla PetBreed. De lo contrario, fallará la integridad referencial y la misma no se insertará. Si es insertada, quedará con fecha la fecha de ingreso del día de hoy.

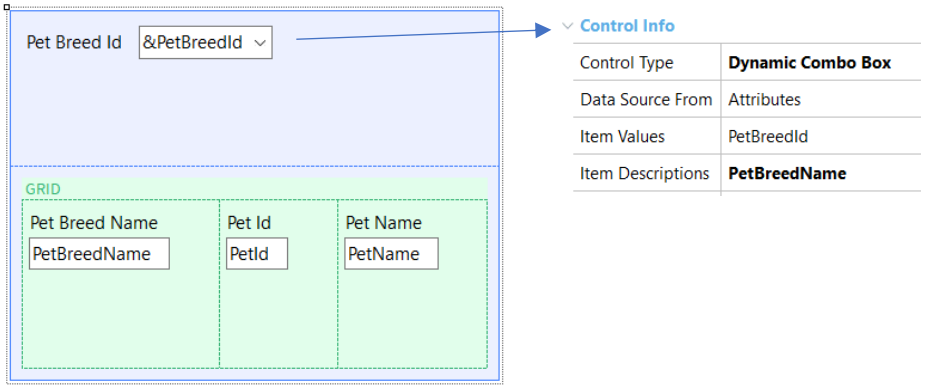
23.3 **–** La mascota será insertada siempre, aunque no exista una raza con identificador 10 en la tabla PetBreed, pues los Business Components no controlan la integridad referencial. Quedará con fecha de ingreso vacía, pues no se especificó ninguna en el código.

23.4 **–** La mascota será insertada siempre, aunque no exista una raza con identificador 10 en la tabla PetBreed, pues los Business Components no controlan la integridad referencial. Quedará con fecha de ingreso del día de hoy.

24) Considera el diseño de transacciones y el layout del Web Panel que se muestra. Se desea diseñar un Web Panel que los identificadores y los nombres de las mascotas

(PetName) de determinada raza, seleccionada por el usuario, así como el nombre de la raza seleccionada.

Determina la opción que consideres correcta.



24.1 – Debe codificar el evento Load, como se muestra:

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

24.2 – Debe codificar el evento Load, como se muestra:

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

24.3 – Debe modificar el evento Start, como se muestra:

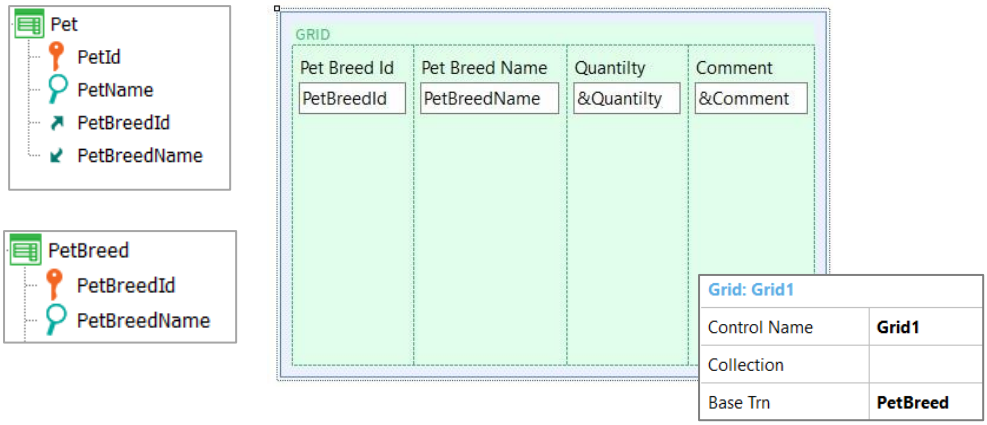
```
Event Start
     PetBreedId = &PetBreedId
Endevent
```

24.4 – Debe establecer la siguiente condición en el Grid:

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

25) Considera el diseño de transacciones y el layout del Web Panel que se muestra. Se desea diseñar un Web Panel que muestre todas las razas (PetBreed), cada una con su respectiva cantidad de mascotas registradas. En caso de registrar más de 10 mascotas se visualizará el comentario “Many pets”. De lo contrario se visualizará “Few pets”.

Determina la opción de implementación que consideres correcta.



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
```

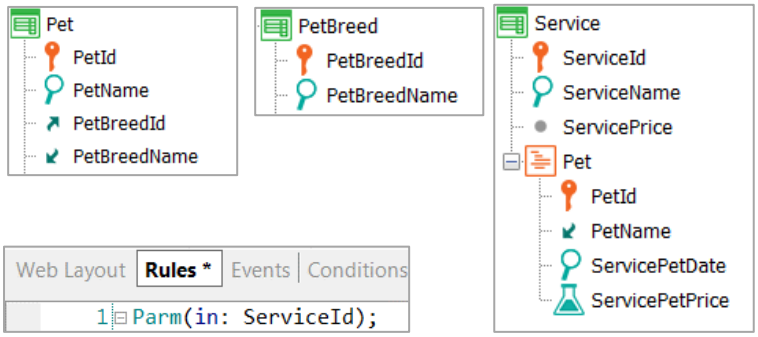
25.3 –

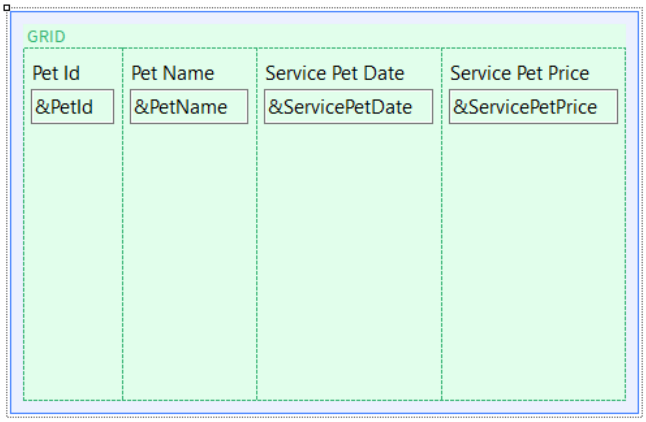
```
Event Load
    &Quantity = Count(PetName)
    If &Quantilty >10
        &Comment = “Many pets”
    Else
        &Comment = “Few pets”
    Endif
EndEvent
```

25.4 – Ninguna de las opciones anteriores es correcta.

26) En un Web Panel, se desea mostrar las mascotas asociadas a determinado servicio recibido por parámetro.

A partir del diseño de transacciones y del Web Panel que se detallan (lo que no se muestra, como las propiedades, es porque no se ha modificado salvo las de las variables que son todas ReadOnly), determina la opción que consideres correcta.





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 – Ninguna de las opciones anteriores es correcta.

RESPUESTAS

1) 3

Justificación – *Cuando un atributo que es clave primaria de una transacción lo agregamos a otra transacción, pasa a ser una clave foránea y se establece una relación 1-N fuerte, en la que la entidad donde está la clave foránea es el lado N de la relación. Como la letra dice que 1 raza tiene N mascotas (ya que muchas mascotas pueden ser de la misma raza), la respuesta correcta es la opción 3, dado que la transacción Pet tiene a PetBreedId como clave foránea. La opción 1 no puede ser porque la relación queda para el otro lado (una mascota tiene muchas razas) y la opción 2 representa el modelo de una relación N a N.*

2) 1

Justificación – *La letra establece una relación muchos a muchos entre alimentos y mascotas. Para modelar una relación N-N en GeneXus, utilizamos una transacción de dos niveles, donde la entidad del segundo nivel se agrega también como transacción aparte. El modelo que representa lo pedido en la letra es el de la opción 1. La opción 2 no es correcta porque representa a una relación 1 a N débil (donde Pet es la entidad débil, notemos que no se incluye a la entidad Pet como transacción independiente) y la opción 3 representa una relación 1-N fuerte, donde 1 alimento es consumido por N mascotas, pero no se cumple la otra parte de la relación.*

3) 1

Justificación – *Cuando usamos una transacción de dos niveles, estamos modelando una relación 1 a N donde la entidad subordinada (la que está en el segundo nivel) es una entidad débil. Esto significa que para identificar correctamente a esta entidad, necesitamos de la entidad fuerte (la que está del lado 1 de la relación), ya que sin esa asociación la entidad débil pierde sentido. En el ejemplo, los cuidados especiales tienen sentido para una determinada mascota, porque esos cuidados son únicos para ella y otra mascota puede requerir de otros cuidados. Notemos que con el modelo de transacciones propuesto, GeneXus crea 2 tablas, una con los atributos del primer nivel (Pet) y otra PetSpecialCare cuya clave primaria es compuesta por PetId y SpecialCareId, demostrando que efectivamente para identificar un cuidado especial se necesita al identificador de la mascota.*

Por lo tanto, la respuesta correcta es la opción 1, ya que establece justamente eso y la prueba es la clave primaria compuesta que identificará al cuidado especial. La opción 2 establece que los cuidados no son propios de una única mascota y eso pasaría si hubiéramos modelado una relación 1-N fuerte entre Pet y SpecialCare utilizando dos transacciones independientes y con PetId como clave foránea. La opción 3 es incorrecta, ya que hemos visto ejemplos de 1-N débil modelados de esta manera, como el caso del vuelo y sus asientos (entidad débil).

4) 2

Justificación – *Para modelar una relación 1 a 1 como la descripta en la letra, en GeneXus utilizamos 2 transacciones modelando una relación 1-N fuerte (usando una clave foránea) y luego creando un índice único por el atributo que es clave foránea. El índice único convierte a ese atributo en una clave candidata y por lo tanto no puede repetirse. La opción que representa esto correctamente es la 2, donde PetId es clave foránea en PetMedicalRecord y además se crea un índice único por ese atributo en la tabla PetMedicalRecord. Eso hace que no pueda haber dos fichas médicas para la misma mascota. La opción 1 no es correcta porque modela la relación 1-N fuerte, pero en lugar del índice único por PetId, se hace que este atributo sea nullable, lo que no impide que pueda repetirse en caso de ingresarse. La opción 3 tiene una clave compuesta, pero esto no impide que pueda utilizarse el mismo PetId para distintos valores de PetMedicalRecordId.*

5) 3

Justificación – *Este es un ejercicio de normalización y sabemos que GeneXus normaliza las tablas en Tercera Forma Normal. La opción 1 tiene correctas a las tablas COUNTRY, PETBREED y PET, pero la tabla SERVICE está mal ya que incluye al atributo ServicePetsQty que es fórmula y no se crea en la tabla y al atributo PetId, clave del segundo nivel. Además falta la tabla SERVICEPET. En la opción 2 las tablas COUNTRY, PETBREED, PET y SERVICEPET están correctas, pero la tabla SERVICE sigue incluyendo al atributo fórmula ServicePetsQty, La opción 3 es la única correcta.*

6) 2

Justificación – *Para responder correctamente esta pregunta es útil construir el diagrama de Bachmann de las tablas. La tabla extendida de SERVICEPET es el conjunto de los atributos de la tabla base y todos aquellos que puedan alcanzarse mediante relaciones N a 1 en forma directa o indirecta. Por lo tanto, la tabla extendida de SERVICEPET es: SERVICEPET (table base), SERVICE, PET, PETBREED y COUNTRY, lo que corresponde con la opción 2.*

7) 3

Justificación – *La propiedad Nullable aplica fundamentalmente a claves foráneas y si el valor es Yes, permite que pueda ingresarse el registro sin darle valor a esa clave foránea y no se realizan los controles de integridad referencial. En caso de que se le ingrese un valor, entonces sí se controla que el valor exista como clave primaria en la tabla correspondiente. Por lo tanto, la opción correcta es la 3, ya que al poner al atributo PetBreedId con Nullable en Yes, no es obligatorio ingresar su valor, pero en caso de ingresarse, se controla que sea válido, como establece la letra de la pregunta. La opción 1 no es correcta porque si se ingresa un valor, sí se controla que exista como registro en PETBREED. La opción 2 no es correcta porque si se define un índice único sobre el atributo PetBreedId en PET evita que el valor del mismo se repita pero siempre deberá ingresarse.*

8) 1

Justificación – *Cuando GeneXus crea una tabla en la base de datos, automáticamente crea índices por clave primaria y claves foráneas. Esto lo hace para que sean eficientes los controles de integridad referencial que se realizan en forma automática. La tabla SERVICEPET tiene un clave primaria compuesta por los atributos ServiceId y PetId, y además contiene al atributo ServicePetPrice. Además, el atributo*

ServiceId es clave foránea a la tabla SERVICE y el atributo PetId es clave foránea a la tabla PET.

Por lo tanto, GeneXus creará un índice por clave primaria por los atributos ServiceId y PetId, un índice por clave foránea por ServiceId y otro índice por clave foránea por PetId. La opción correcta es la 1, porque es la única que contiene a los índices antes mencionados. La opción 2 es incorrecta porque le faltan los dos índices por clave foránea y la opción 3 es incorrecta porque le falta el índice por clave primaria.

9) 3

Justificación – *Cuando se utiliza el form de la transacción o la transacción como business component, GeneXus realiza los controles de integridad referencial en forma automática. Esto implica que si se desea eliminar un registro de PetBreed, primero verificará que no existan registros en Pet que contengan ese valor de PetBreedId como clave foránea. En caso de encontrarlo, se dará un mensaje y no se permitirá la eliminación. Este comportamiento es el explicado en la opción 3 que es la opción correcta. La opción 1 no es correcta porque sí se realizan controles de integridad referencial cuando se está utilizando el form de la transacción como establece la letra. La opción 2 no es correcta porque este comportamiento corresponde a la eliminación en cascada, que no se implementa en GeneXus.*

10) 1

Justificación – *Para establecer el orden de disparo de las reglas y las fórmulas, GeneXus arma un árbol de evaluación. Sin embargo, como a veces el orden elegido por GeneXus no se adapta a ciertos requerimientos, tenemos disponibles ciertos momentos de disparo que se llevan a cabo siempre en el servidor, luego de que se presiona Confirm en el form de la transacción. Estos momentos de disparo empiezan con la cláusula "on" y en una transacción de dos niveles tenemos momentos antes y después de las instancias de validación y grabación del cabezal, antes y después de la validación y grabación de cada línea, y antes y después del Commit. La regla especificada en la letra tiene el momento de disparo on AfterInsert y solamente incluye atributos del cabezal (en este caso ServicePetsQty), por lo que la regla se disparará únicamente cuando se está insertando un registro del cabezal e inmediatamente después de que el registro sea grabado, pero antes de empezar a procesar las líneas, es decir, los registros del segundo nivel.*
Como el requerimiento inicial de la letra es que la regla debe evitar que se grabe un servicio si no tiene animales registrados, la opción correcta es la 1 ya que con esta implementación se grabará primero el registro del servicio (cabrezal) y recién después se disparará la regla, por lo que la regla no realiza el control evitando que se grabe el registro. La opción 2 no es correcta porque establece que la regla se disparará luego de validar y grabar todas las líneas (animales), que como vimos antes no es así. La opción 3 no es correcta porque indica que la regla se disparará antes de insertar el cabezal (cuando el momento usado es on AfterInsert). Y la opción 4 no es correcta, porque al condicionar la regla con un momento de disparo, siempre se disparará en el servidor luego de presionar Confirm y nunca en el cliente.

11) 1

Justificación – *La regla a) tiene únicamente un atributo del cabezal (FoodId) y el momento de disparo AfterComplete es el último disponible, luego de que se efectúa el Commit. La regla b) tiene únicamente un atributo del cabezal (FoodId) y el momento de disparo AfterInsert hace que se dispara inmediatamente después de que el registro del cabezal es insertado. La regla c) tiene un atributo del cabezal (FoodId) y usa el momento de disparo on AfterLevel Level PetId, siendo PetId un atributo de las líneas (nivel subordinado). Por lo tanto, como el servidor primero realiza la validación y grabación del cabezal, luego la validación y grabación de cada línea, después termina de procesar las líneas y por último realiza el Commit, el orden de las reglas será b), c) , a), que corresponde a la opción 1 que es la opción correcta. Las opciones restantes son incorrectas y en particular a la opción 4, debido al árbol de evaluación que GeneXus arma para el disparo (en el que toma en cuenta dependencia entre reglas y fórmulas), no podemos asumir que las reglas se disparan en el orden en el que son declaradas.*

12) 3

Justificación – *La fórmula presentada en la letra es una fórmula horizontal, que se expresa en forma declarativa, puede contener una o varias asignaciones condicionadas y ejecutará únicamente la asignación que tiene la condición que se cumple. Este tipo de fórmula navega un único registro y la tabla extendida, por lo que los atributos mencionados deberán pertenecer todos a la tabla extendida de la tabla asociada al atributo fórmula.*
La fórmula está asignando al atributo ServicePetPrice el valor 0 si el día para realizar el servicio coincide con el día en el que la mascota fue registrada en la tienda y además la mascota es VIP, y asignará el precio ServicePrice si no se cumplen las condiciones anteriores (ya que esto se condiciona con OTHERWISE). Por lo tanto, la fórmula está correcta y resuelve el requerimiento solicitado, por lo que la respuesta correcta es la 3. La opción 1 no es correcta porque no toma en cuenta que los atributos PetAddedDate y PetIsVIP pertenecen a la tabla extendida de SERVICEPET que es lo requerido. La opción 2 tampoco es correcta porque establece que el IF debe utilizarse con ELSE siendo que este tipo de cláusula corresponde a una programación procedural y no a la programación declarativa que es la usada en la definición de las fórmulas.

13) 3

Justificación – *La duración de la membresía de la mascota en años, se calcula como la diferencia entre el año actual y el año de la fecha de ingreso a la tienda de mascotas. La fecha de ingreso está dada por el atributo PetAddedDate y el nombre de la mascota por el atributo PetName. Para listar todos los animales de la tienda de mascotas con su nombre y duración de la membresía, se ha diseñado un layout donde en el PrintBlock1 se incluye al atributo PetName y a la variable &MembershipDuration y para recorrer la tabla de mascotas, se utiliza un comando For each Pet. La opción 1 no es correcta porque la instrucción de imprimir el PrintBlock1 está fuera del For each y si bien la variable &MembershipDuration va a quedar con el valor del último registro de PET, el printBlock no va a poder instanciarse por estar fuera del For Each, por lo que no puede imprimirse y el listado ni siquiera muestra la página. La opción 2 tampoco es correcta porque ahora es el atributo PetAddedDate el que está fuera del For each y no podrá instanciarse, con lo cual la variable &MemershipDuration quedará en el valor del año actual (ej:2024). El For each*

imprimirá todos los nombres de las mascotas, con el valor del año actual en la duración de la membresía. La opción correcta es la 3, en la que se incluye dentro del For each el cálculo de la variable &MembershipDuration y la impresión del PrintBlock1 con los datos requeridos en el listado. De esta manera se listará a todas las mascotas de la tienda con su nombre y duración de la membresía.

14) 3

Justificación – *Tanto el veterinario titular como el suplente deben poder elegirse de los almacenados en la tabla VET, y si se ingresa el identificador, controlarse que ese valor exista como identificador de un registro en la tabla. La opción 1 no es correcta porque si bien el valor de VetId y el de VetAssistantId pueden elegirse en forma independiente, lo lógico es que la tabla con los veterinarios sea una sola y no tener dos tablas separadas con los nombres de los veterinarios, como es el caso de la tabla VETASSITANT donde incluso puede darse que haya veterinarios en VET que no estén en VETASSISTANT y viceversa. La opción 2 tampoco es correcta, porque el atributo VetAssistantName está en un grupo de subtipos diferente que el VetAssistantId, por lo que no se inferirá el nombre del veterinario cuando se elija el identificador. La opción 3 es la correcta, ya que VetAssistantId es subtipo de VetId con lo que se comportará como tal, será clave foránea en Pet y se harán los controles de integridad referencial sobre Vet. Además, como el grupo de subtipos incluye a VetAssistantName como subtipo de VetName, al elegir el valor de VetAssistantId se inferirá el valor de VetAssistantName, como puede verse en la imagen de la transacción Pet donde se muestra al atributo VetAssistantName como inferido.*

15) 3

Justificación – *Lo que requiere la letra es que podamos ingresar el país de la mascota, independientemente del país de la raza. Si observamos la transacción PetBreed, se incluyen los atributos CountryId y CountryName. Al incluirse PetBreedId en Pet, si agregamos al atributo CountryId, el mismo no será una clave foránea a Country, sino que será un atributo inferido de PetBreedId. En otras palabras tomará el valor del país de la raza, será read only y no podremos elegir un país para la mascota. La opción 1 es incorrecta porque por más que aparezcan primero los atributos del país antes que los de la raza, igual el país será inferido de la raza como puede observarse en la imagen, donde CountryId tiene una flecha hacia abajo indicando que es un atributo inferido. La opción 2 agrega una transacción más PetCountry, con los atributos PetCountryId y PetCountryName, que luego agrega a la transacción Pet. Además agrega un grupo de subtipos donde PetCountryId es subtipo de CountryId y PetCountryName es subtipo de CountryName, con lo que se establece una relación de especialización respecto a CountryId y CountryName respectivamente. Esta solución si bien funciona, no es adecuada porque agrega objetos innecesarios para la solución, como la transacción PetCountry o la relación de especialización. La solución correcta es la 3, que incluye en la transacción Pet a los atributos PetCountryId definido como subtipo de CountrId y PetCountryName definido como subtipo de CountryName, en el respectivo grupo de subtipos. Esta solución hace que se pueda ingresar un país para la mascota independiente del país de la raza, utilizando PetCountryId como clave foránea y PetCountryName como atributo inferido, sin necesidad de agregar objetos innecesarios.*

16) 1

Justificación – *El listado de navegación del procedimiento nos muestra dos For Eachs anidados, el externo con tabla base COUNTRY (el For Each de más arriba) y el anidado (que se muestra indentado) con tabla base SERVICE. Si analizamos las transacciones, vemos que no existe relación alguna entre las tablas COUNTRY y SERVICE, de modo que sabemos que si tenemos dos For Eachs anidados, con tabla base diferente (en este caso COUNTRY y SERVICE), la navegación resultante será un producto cartesiano, es decir, por cada registro de la tabla COUNTRY, se listarán todos los registros existentes en la tabla SERVICE. Esto lo podemos observar en el listado de navegación porque en el nivel anidado que navega la tabla de SERVICE no se encuentra ninguna indicación de filtro por contexto. Si GeneXus hubiera encontrado alguna relación entre COUNTRY y SERVICE, aparecería a la derecha de la tabla texto indicando un filtro implícito de la forma: CountryId = @CountryId. Como no es el caso, podemos asegurar que GeneXus no encontró ninguna relación entre las tablas bases de los For Eachs y por lo tanto la navegación resultante será un producto cartesiano. La respuesta correcta es la 1.*

17) 4

Justificación – *La letra pide recorrer la tabla PET, filtrada por la mascota recibida por parámetro y luego recorrer la tabla PETSERVICE para traer los servicios asociados a esa mascota. Si recorremos la tabla PET, los atributos de PETSERVICE no pertenecen a la tabla extendida de PET, por lo tanto necesitamos dos For Eachs anidados, uno que recorra PET y el anidado que recorra PETSERVICE. En la opción 1 se recibe por parámetro el atributo PetId, por lo que automáticamente se filtrarán las navegaciones y solo se mostrarán registros cuyo PetId sea igual al valor recibido por parámetro. Luego tenemos 2 For Eachs anidados, ambos con tabla base PETSERVICE (indicado por el uso de la transacción base Pet.Service en ambos), lo que provocará un corte de control, pero como falta el order por PetId en el for each externo que provoque la agrupación, se listarán los nombres de mascota repetidos por cada servicio por lo que la opción 1 no es correcta. En la opción 2 se utilizan tres For Eachs anidados, lo que es totalmente innecesario ya que únicamente necesitamos dos y además no se necesita para nada un For Each por PETBREED, ya que ese atributo pertenece a la extendida de PET, por lo que la opción 2 tampoco es correcta. La opción 3 no es correcta ya que implementa un único For Each con tabla base PET y como explicamos anteriormente no se podrán alcanzar los atributos ServiceName y SevicePetPrice porque no pertenecen a la extendida de PET. La opción 4 tiene los dos For Eachs anidados requeridos, el externo por PET y el anidado por PETSERVICE. En el For Each externo no se incluye la cláusula Where para filtrar por el valor de PetId porque el parámetro se recibe en el atributo PetId, que como ya vimos actúa como filtro por igualdad por el valor recibido, por lo que en PET únicamente se recorrerá el registro de la mascota correspondiente y al estar los For Eachs anidados y las tablas PET y PETSERVICE estar relacionadas, se listarán únicamente los servicios que correspondan a la mascota recibida por parámetro, por lo cual la opción 4 es la correcta.*

18) 4

Justificación – *Como la letra solicita listar todas las razas registradas independientemente de que haya mascotas de esa raza y para cada raza que tenga mascotas, las mascotas asociadas, nos están solicitando un JOIN. Esto implica dos For Eachs anidados, el externo con tabla base PETBREED (para que recorra todas las razas registradas) y el anidado con tabla base PET. Para que haya un JOIN además se requiere que las tablas bases estén relacionadas, que es el caso entre PETBREED y PET. Si vemos las opciones, la opción 1 es incorrecta porque*

presenta dos For Each paralelos en lugar de estar anidados. La opción 2 sí muestra dos For Eachs anidados pero sobre la misma tabla base PET, que no es lo que precisamos, por lo cual tampoco es correcta. La opción 3 tiene un único For Each con tabla base PET con lo cual no se formarán grupos por raza, sino que se repetirá el nombre de la raza por cada registro de PET, que no es lo que se pretende de que para cada raza se muestren las mascotas de esa raza, por lo que la opción 3 tampoco es correcta. Si analizamos la opción 4, vemos que ahora tenemos dos For Eachs anidados, el externo con tabla base PETBREED y el anidado por PET, es decir distintas tablas base pero relacionadas. Se listará todas las razas registradas en PETBREED y para cada una solamente las mascotas de esa raza, ya que debido a esta relación se efectuará un JOIN. Esta opción 4 es la correcta.

19) True

Justificación – *La implementación muestra un For Each con tabla base PET, que recorrerá esta tabla únicamente para las mascotas con raza con PetBreedId = 4 o PetBreedId = 7 y que además el año de la fecha de ingreso es 2020. Esto cumple con los requerimientos establecidos en la letra de la pregunta. La duda que nos puede quedar es si todos los atributos incluidos en las cláusulas Where y en el printblock pertenecen o no a la tabla extendida de PET, que es la tabla base del For Each. Chequeamos eso y vemos que sí se cumple, por lo que la implementación es correcta y la respuesta es VERDADERO.*

20) 3

Justificación – *Como la letra solicita listar solamente las razas que tengan mascotas registradas, cada una con sus correspondientes mascotas, nos están solicitando un CORTE DE CONTROL. No vamos a poder listar las razas desde la tabla PETBREED, porque allí están todas las razas, tengan o no tengan mascotas registradas, por lo que debemos recuperar las razas de la tabla PET. Como en este caso recuperamos la raza de cada mascota, va a ser una raza que tiene mascotas registradas. Para implementar un corte de control entre razas y mascotas, necesitamos dos For Eachs anidados, ambos con la misma tabla base y con la cláusula ORDER por el atributo que queremos hacer el corte (agrupando por PetBreedId) en el For Each externo. La opción 1 no es correcta porque hay un solo For Each y las mascotas no salen agrupadas por raza. La opción 2 tiene al For Each externo con tabla base PETBREED que ya vimos que no es correcta. La opción 3 cumple con todo lo que necesitamos para implementar correctamente el corte de control, por lo que la opción 3 es la correcta. A la opción 4 le falta el order que permite los agrupamientos, por lo cual es incorrecta.*

21) 3

Justificación – *La letra nos pide listar todos los veterinarios que tengan al menos una mascota a cargo y en caso de que no haya ningún veterinario que tenga mascotas asignadas, debe mostrarse un mensaje. Para eso, debemos recorrer la tabla VET filtrando aquellos registros que la cuenta de mascotas sea mayor o igual a 1. Todas las opciones hacen esto correctamente, el problema es cómo implementamos la segunda parte, que es mostrar un mensaje si no hay ningíun veterinario con mascotas. La opción 1 no es correcta porque usa un ELSE que no forma parte de la sintaxis del For Each. La opción 2 luego de imprimir el printBlock1 usa una cláusula UNIQUE, pero esta cláusula se usa para evitar repetidos y no aplica a este caso por lo que tampoco es correcta. La opción 3 sí es correcta porque utiliza WHEN NONE, que es justamente la cláusula adecuada en la sintaxis del For Each para usar*

en estos casos, es decir, cuando no hay ningún registro que cumpla las condiciones de los filtros.

22) 2
Justificación – *El tipo de datos estructurado SDTCountry es colección y para cargar esa colección, el Data Provider deberá recorrer la tabla COUNTRY para obtener el identificador del país, el nombre y la cantidad de mascotas. Para eso, el Data Provider debe incluir la cláusula from Country para indicar la transacción base, cuya tabla base será la tabla recorrida por el Data Provider. Además, el Data Provider debe tener la propiedad Collection en False, ya que el output será la estructura SDTCountry que ya es colección. Las tres opciones tienen la misma implementación para el ítem SDTCountryItem y esta implementación es correcta. La opción 1 no es correcta porque la transacción base es Pet y no Country. La opción 2 es la correcta, ya que cumple que se recorre la tabla base COUNTRY y que la propiedad Collection está en False. La opción 3 no es correcta ya que intenta recorrer la tabla PETBREED.*

23) 2
Justificación – *Para insertar un registro en la tabla PET usando la variable &Pet definida como business component de Pet, el procedimiento debería asignar valores al PetName, PetBreedId y PetAddedDate, ya que al PetId no es necesario por ser autonumerado. Sin embargo, debido a que la transacción Pet tiene definida la regla Default que asigna la fecha del día a PetAddedDate, no es necesario asignar este valor en la implementación del procedimiento. Y como sabemos que GeneXus controla la integridad referencial cuando se usan business component, el valor de PetBreedId debe ser válido, es decir que debe existir como clave primaria en la tabla PETBREED. Luego de asignar estos valores, el source del procedimiento invoca al método Insert() y si el resultado es verdadero se realiza el Commit, lo que está correctamente implementado. La opción 1 tiene la primera parte correcta cuando habla del valor de PetBreedId, pero no es cierto que si se inserta el registro quede vacía la fecha PetAddedDate, ya que la regla Default se dispara y se asigna este valor. La opción 2 es la correcta ya que explica exactamente lo que sucederá al ejecutarse el código implementado. La opción 3 no es correcta porque no es cierto que si usamos BC no se implementen los controles de integridad referencial y además por la regla la fecha no quedará vacía. La opción 4 tampoco es correcta, porque si bien es correcto que se asigna la fecha de hoy, no es cierta la afirmación de que al usar BC no se realizan los controles de integridad referencial.*

24) 4
Justificación – *El Grid tiene tabla base ya que tiene a los atributos PetBreedName, PetId y PetName. Para filtrar el Grid con tabla base, utilizamos su propiedad Conditions con la condición de que el PetBreedId sea igual al valor seleccionado en la variable &PetBreedId. La opción 1 es incorrecta ya que al tener tabla base el Grid, GeneXus crea un For Each implícito sobre la tabla PET que permite obtener los valores de PetId y PetName, así como PetBreedName por la tabla extendida y por lo tanto no es necesario implementar un For each en el evento Load para recuperar esos datos. La opción 2 tampoco es correcta porque para filtrar por el valor de PetBreedId seleccionado en la variable se deben utilizar las Conditions del Grid y no un For Each en el evento Load, ya que este For Each quedaría anidado al For*

Each implícito y este where no filtrará a dicho For Each implícito. La opción 3 tampoco es correcta porque el evento Start se dispara una única vez y con esa asignación no quedará filtrado el For Each implícito por el valor de la raza seleccionada. La opción 4 es la opción correcta, ya que con esa condición agregada a las propiedades del Grid GeneXus logrará filtrar los registros de la grilla por el valor elegido de &PetBreedId.

25) 3

Justificación – *Como el Grid tiene tabla base PETBREED, debido al valor de la propiedad Base Trn, sabemos que el evento Load se disparará N veces, una vez por cada registro de la tabla PETBREED. Es en el evento Load donde debemos obtener la cantidad de mascotas -mediante la fórmula count(PetName)- y además dependiendo de esta cantidad asignar el valor de la variable &Comment. La opción 1 no es correcta porque al ser un Grid con tabla base se creará un For Each implícito y no es necesario agregar un For Each en el evento Load. La opción 2 si bien intenta una implementación diferente de la 1, tampoco es correcta porque también agrega un For Each innecesario, que además quedará anidado al implícito. La opción 3 es la opción correcta, ya que para cada registro de PETBREED se disparará el evento Load y se calculará la cantidad de mascotas para la raza que se esté recorriendo y luego si la cantidad es mayor a 10 se asignará el mensaje de "muchas mascotas" y para el caso contrario "pocas mascotas". La opción 4 por lo tanto tampoco es correcta.*

26) 2

Justificación – Como en el Grid no hay atributos y se aclara que el resto de las propiedades del Grid y secciones del Web Panel tienen valores por defecto, podemos afirmar que el Web Panel no tiene tabla base. En este caso, sabemos que el evento Load se disparará una única vez y por lo tanto en dicho evento debemos implementar un For Each que recorra la tabla SERVICEPET para que a través de su extendida podamos recuperar los valores de los atributos PetId, PetName, ServicePetDate y ServicePetPrice y asignar los valores de las variables del Grid, utilizando el comando Load para copiar los sucesivos valores a cada línea del Grid. La opción 1 no es correcta porque la transacción base del For Each es Service, y desde la tabla base SERVICE no es posible alcanzar los atributos que necesitamos. La opción 2 es la correcta ya que asignando la transacción base Service.Pet, la tabla base del For each será SERVICEPET y nos permitirá obtener los valores necesarios para cargar el Grid. La opción 3 no es correcta, porque no es necesario el For Each Service, ya que con un único For Each con tabla base SERVICEPET ya podemos recuperar todos los atributos a través de su tabla extendida. La opción 4 tampoco es correcta.