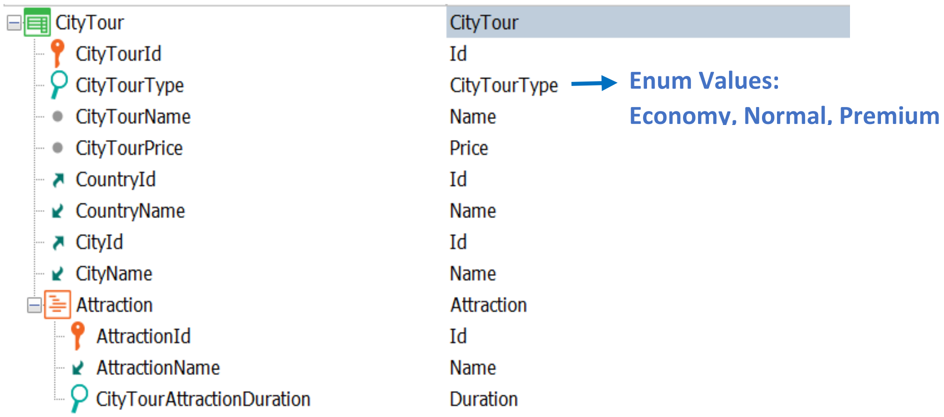
GeneXus 18 Analyst Exam

1. An application is being developed for a travel agency. The City tours available to visit different tourist attractions in the city need to be registered. Only the structure of the CityTour transaction is shown. The following behavior should be achieved:

   - To be warned as early as possible that the CityTourName field has been left empty (in any mode).
   - When inserting a city tour, and selecting the Economy type, entering a price that is above the price range of the economy type should be forbidden. This will be checked by a CheckPrice proc whose logic is not relevant for this exercise. However, you don't want the control to be performed on the client, but only on the server, right before the header is saved in the CityTour table.

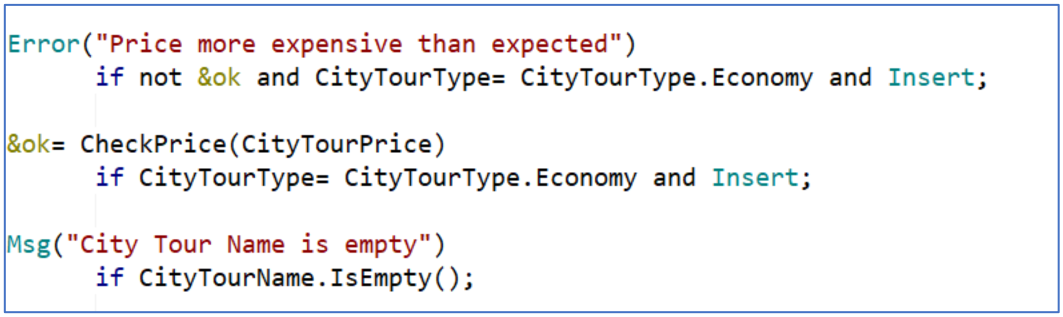
   Determine which of the following options correctly implements the rules according to these requirements.



CityTour	CityTour
CityTourId	Id
CityTourType	CityTourType
CityTourName	Name
CityTourPrice	Price
CountryId	Id
CountryName	Name
CityId	Id
CityName	Name
Attraction	Attraction
AttractionId	Id
AttractionName	Name
CityTourAttractionDuration	Duration

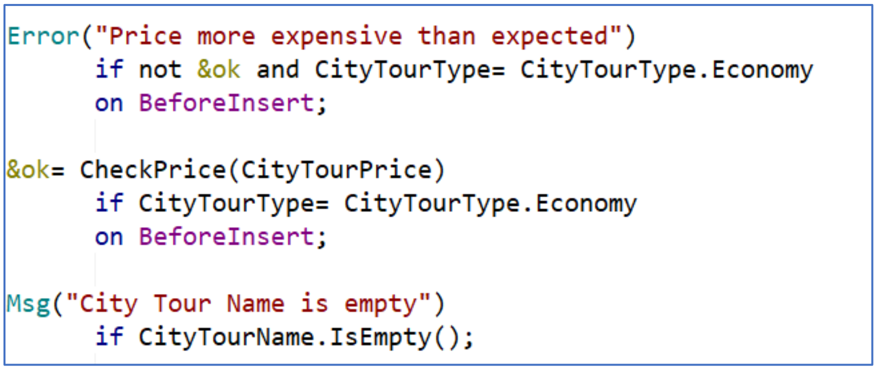
Enum Values:
Economy, Normal, Premium

a.



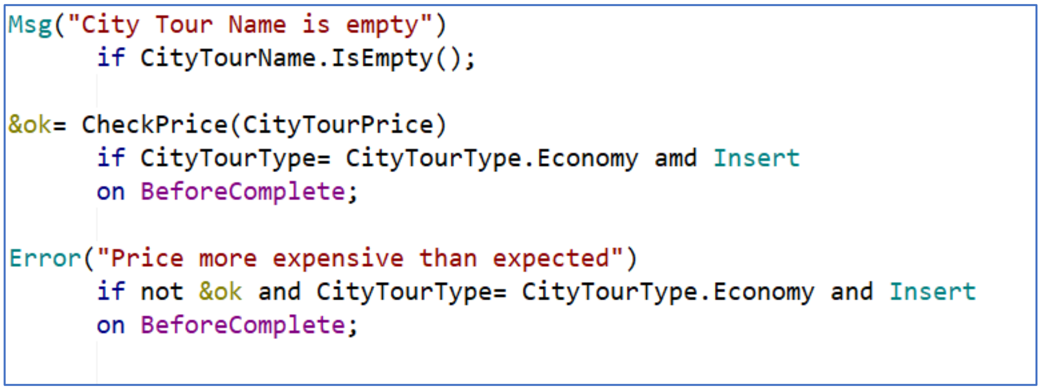
```
Error("Price more expensive than expected")
      if not &ok and CityTourType= CityTourType.Economy and Insert;

&ok= CheckPrice(CityTourPrice)
      if CityTourType= CityTourType.Economy and Insert;

Msg("City Tour Name is empty")
      if CityTourName.IsEmpty();
```

b.



```
Error("Price more expensive than expected")
      if not &ok and CityTourType= CityTourType.Economy
      on BeforeInsert;

&ok= CheckPrice(CityTourPrice)
      if CityTourType= CityTourType.Economy
      on BeforeInsert;

Msg("City Tour Name is empty")
      if CityTourName.IsEmpty();
```

c.



```
Msg("City Tour Name is empty")
      if CityTourName.IsEmpty();

&ok= CheckPrice(CityTourPrice)
      if CityTourType= CityTourType.Economy amd Insert
      on BeforeComplete;

Error("Price more expensive than expected")
      if not &ok and CityTourType= CityTourType.Economy and Insert
      on BeforeComplete;
```

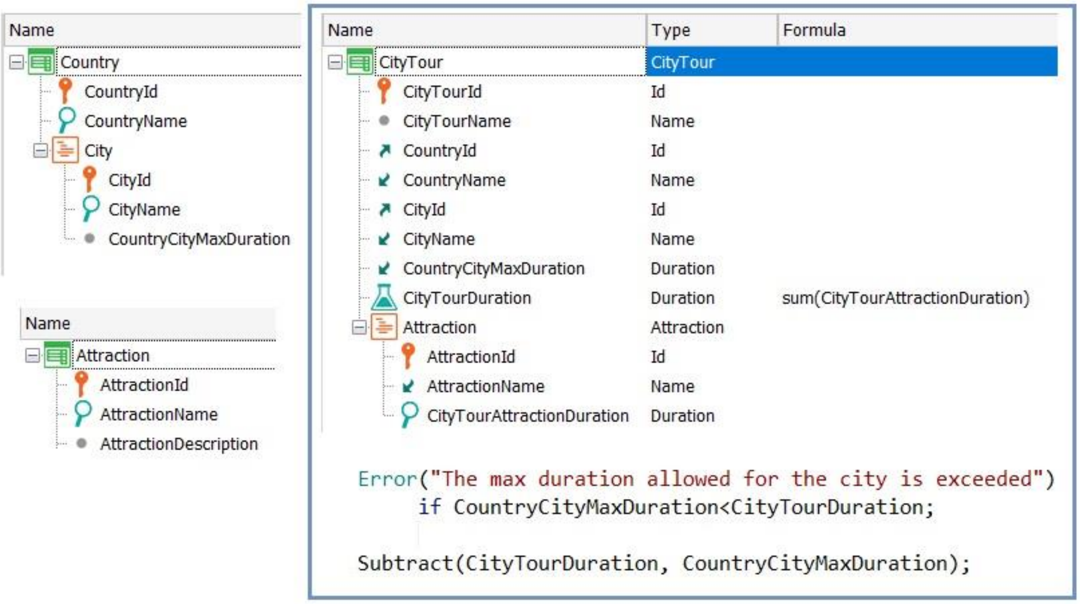
d. None of the above.

2. In the context of an application for a travel agency, there is a transaction to record the city tours offered by the agency to its customers. The tourist attractions to be visited are recorded, along with the time (duration) scheduled to visit each attraction.

   Assuming that each travel agency has a maximum amount of time allowed for all its city tours, the CountryCityMaxDuration attribute has been added to the Country transaction.

   Every time a city tour is entered for the city, that attribute will have to be updated according to the total duration of the city tour.

   For this purpose, the two rules shown in the image have been specified in CityTour. Determine whether or not they allow meeting the requirement. From the options given below, choose the correct one.



Name
Country
CountryId
CountryName
City
CityId
CityName
CountryCityMaxDuration

Name
Attraction
AttractionId
AttractionName
AttractionDescription

Name	Type	Formula
CityTour	CityTour	
CityTourId	Id	
CityTourName	Name	
CountryId	Id	
CountryName	Name	
CityId	Id	
CityName	Name	
CountryCityMaxDuration	Duration	
CityTourDuration	Duration	sum(CityTourAttractionDuration)
Attraction	Attraction	
AttractionId	Id	
AttractionName	Name	
CityTourAttractionDuration	Duration	

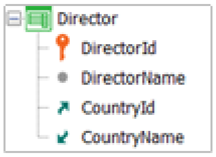
```
Error("The max duration allowed for the city is exceeded")
     if CountryCityMaxDuration<CityTourDuration;

Subtract(CityTourDuration, CountryCityMaxDuration);
```

   a. A formula attribute cannot be used to be subtracted, added, or updated from an attribute of the extended table of the same level.
   b. The error rule is incorrectly conditioned, because since it must use the CountryCityMaxDuration attribute, which is updated by the subtract rule, it will always be fired after it. Therefore, the condition should be “If CountryCityMaxDuration < 0”.
   c. The rules still need to be conditioned to the event “On BeforeInsert, BeforeUpdate,BeforeDelete” so that it is triggered only on the server and before updating the record. Otherwise, it will be triggered and updated twice.

d. None of the above.

3. A Movie can be directed by one or more directors (Director). It is necessary to record the country where the movies were made and indicate the country of each director.

The development team has defined the Director transaction as follows:

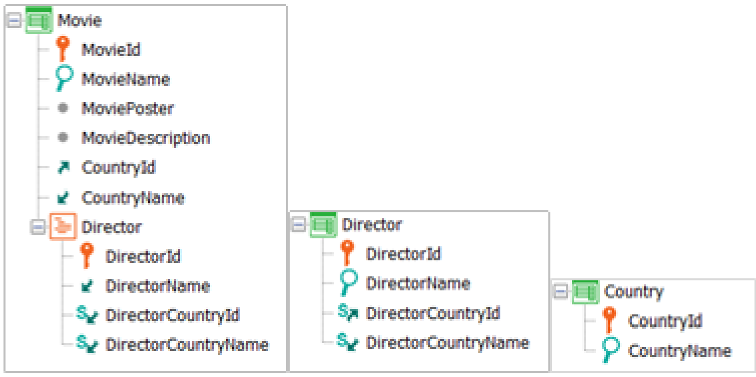


- Director
  - DirectorId
  - DirectorName
  - CountryId
  - CountryName

Before continuing with the other transactions of the model, the team has also created all the programs to meet the requirements of the company that will manage the Cinema regarding the directors (for example: lists of directors in PDF, viewing directors filtered by their country from a Web Panel, procedures that perform different checks with their data, etc.).

Consider options 1 and 2 and indicate what you consider correct.

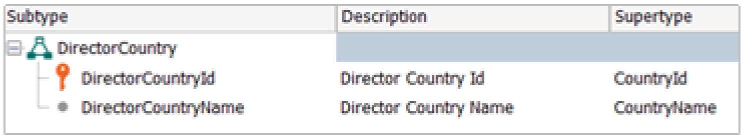
1)



- Movie
  - MovieId
  - MovieName
  - MoviePoster
  - MovieDescription
  - CountryId
  - CountryName
  - Director
    - DirectorId
    - DirectorName
    - DirectorCountryId
    - DirectorCountryName

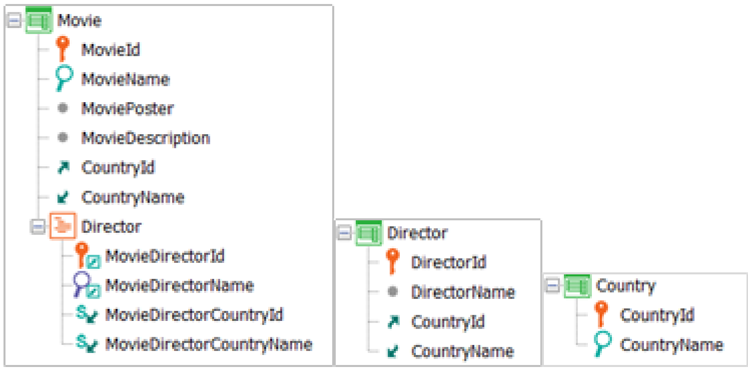
- Director
  - DirectorId
  - DirectorName
  - DirectorCountryId
  - DirectorCountryName

- Country
  - CountryId
  - CountryName



Subtype	Description	Supertype
DirectorCountry		
DirectorCountryId	Director Country Id	CountryId
DirectorCountryName	Director Country Name	CountryName

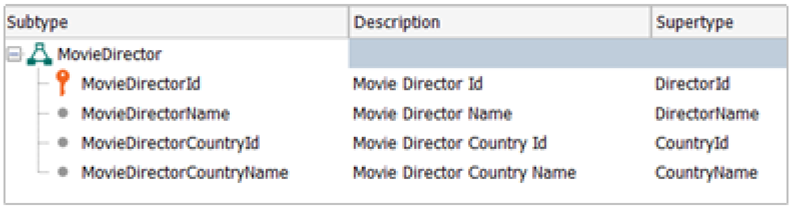
2)



- Movie
  - MovieId
  - MovieName
  - MoviePoster
  - MovieDescription
  - CountryId
  - CountryName
  - Director
    - MovieDirectorId
    - MovieDirectorName
    - MovieDirectorCountryId
    - MovieDirectorCountryName

- Director
  - DirectorId
  - DirectorName
  - CountryId
  - CountryName

- Country
  - CountryId
  - CountryName



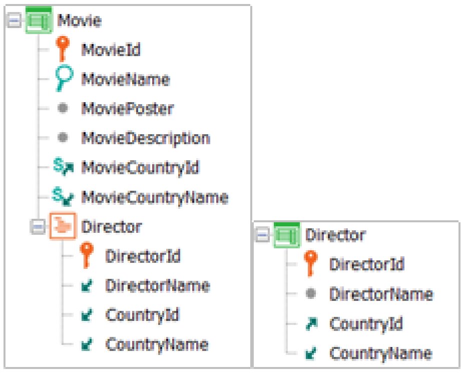
Subtype	Description	Supertype
MovieDirector		
MovieDirectorId	Movie Director Id	DirectorId
MovieDirectorName	Movie Director Name	DirectorName
MovieDirectorCountryId	Movie Director Country Id	CountryId
MovieDirectorCountryName	Movie Director Country Name	CountryName

a) Both options are equivalent and meet the requirement with no advantages or disadvantages over each other.

b) Although both options meet the requirement, option 1 is more appropriate in this case. Since the subtype group has fewer elements, fewer references are created, and when changing the names of the attributes CountryId and CountryName in the Director transaction, these attributes are automatically updated in the objects where they are used.

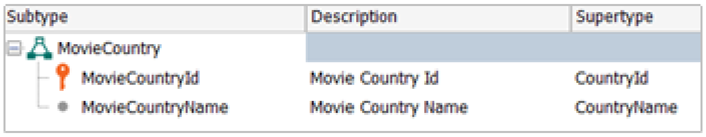
c) Although both options meet the requirement, option 2 is more appropriate in this case. With it, there is no need to go to each program already created to modify the attributes CountryId and CountryName because they don't change.

d) Neither option meets the requirement. The only solution would be as follows:



- Movie
  - MovieId
  - MovieName
  - MoviePoster
  - MovieDescription
  - MovieCountryId
  - MovieCountryName
  - Director
    - DirectorId
    - DirectorName
    - CountryId
    - CountryName

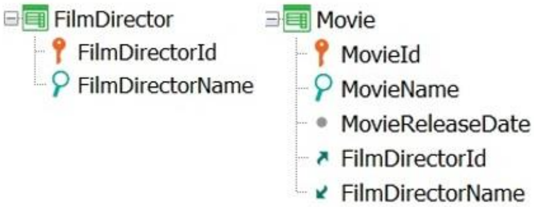
- Director
  - DirectorId
  - DirectorName
  - CountryId
  - CountryName



Subtype	Description	Supertype
MovieCountry		
MovieCountryId	Movie Country Id	CountryId
MovieCountryName	Movie Country Name	CountryName

4. Consider the transaction design shown below. A list is needed that receives a date by parameter and shows the movies (Movie) sorted by release date (MovieReleaseDate), if that parameter is a date later than the current one. Otherwise, the list will be sorted in descending order by the director's name (FilmDirectorName).

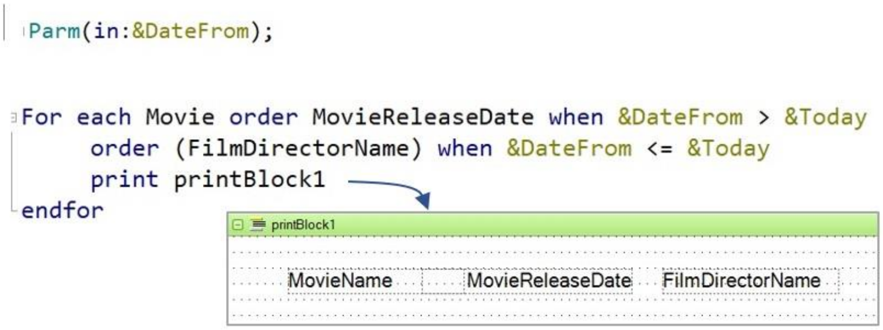
Determine what you consider correct:



FilmDirector
- FilmDirectorId
- FilmDirectorName

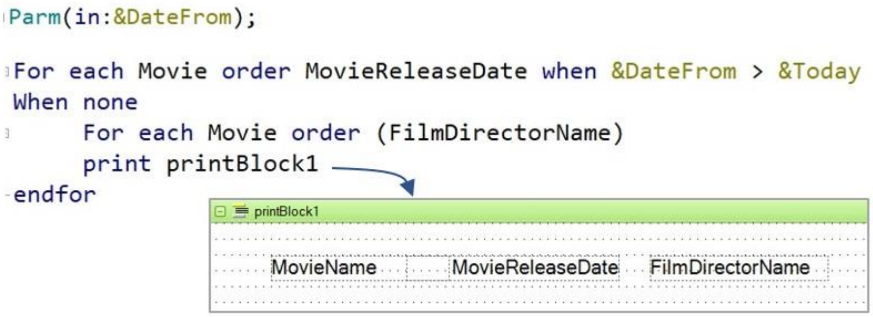
Movie
- MovieId
- MovieName
- MovieReleaseDate
- FilmDirectorId
- FilmDirectorName

a) The requirement is met by indicating the desired orders conditioned with the corresponding When clauses. The order that verifies the conditions will always be applied:



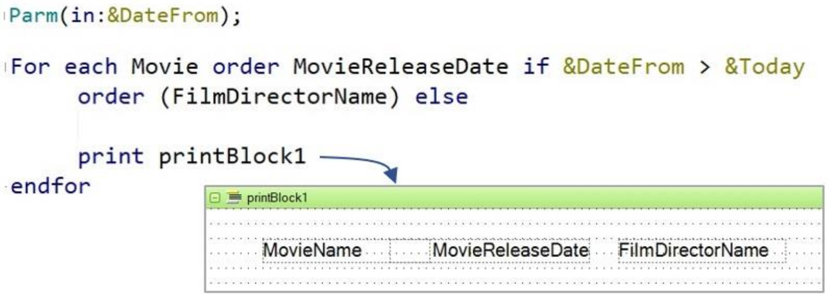
```
Parm(in:&DateFrom);

For each Movie order MovieReleaseDate when &DateFrom > &Today
     order (FilmDirectorName) when &DateFrom <= &Today
     print printBlock1
endfor
```

b) One of the conditional orders must be indicated, and if the condition is not met, the other one is indicated using the When none clause of the For each:



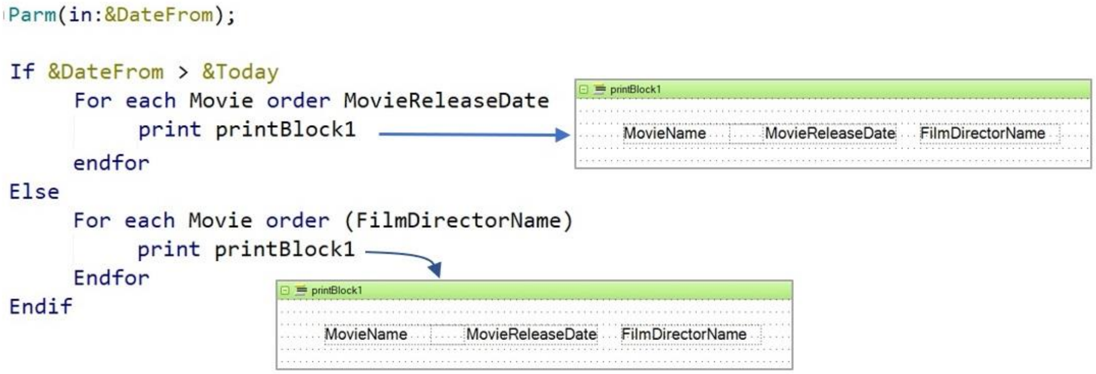
```
Parm(in:&DateFrom);

For each Movie order MovieReleaseDate when &DateFrom > &Today
When none
      For each Movie order (FilmDirectorName)
      print printBlock1
endfor
```

c) The requirement is met by indicating the desired orders conditioned with the If - Else clause as shown:



```
Parm(in:&DateFrom);

For each Movie order MovieReleaseDate if &DateFrom > &Today
      order (FilmDirectorName) else

      print printBlock1
endfor
```

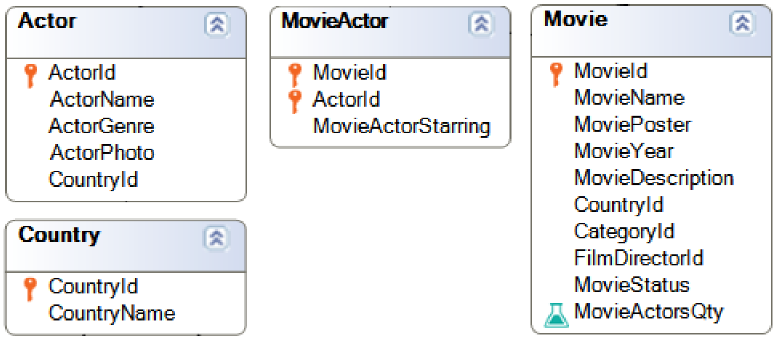
d) It is not possible to meet the requirement by declaring a single For each and indicating several conditioned orders. To solve it, every For each must be implemented according to the value of the parameter received:



```
Parm(in:&DateFrom);

If &DateFrom > &Today
     For each Movie order MovieReleaseDate
          print printBlock1
     endfor
Else
     For each Movie order (FilmDirectorName)
          print printBlock1
     Endfor
Endif
```

5. Consider the tables generated and the procedure shown below.

Based on this implementation, select the correct statement from the following ones.

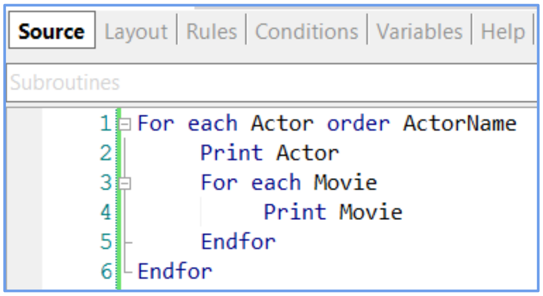


Actor
- ActorId
- ActorName
- ActorGenre
- ActorPhoto
- CountryId

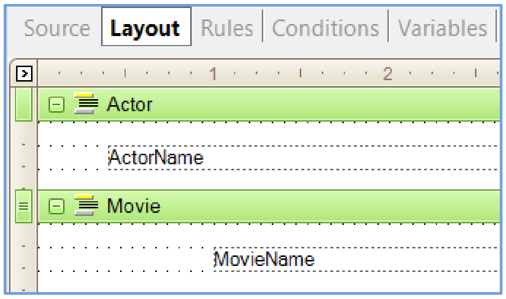
Country
- CountryId
- CountryName

MovieActor
- MovieId
- ActorId
- MovieActorStarring

Movie
- MovieId
- MovieName
- MoviePoster
- MovieYear
- MovieDescription
- CountryId
- CategoryId
- FilmDirectorId
- MovieStatus
- MovieActorsQty



```
For each Actor order ActorName
    Print Actor
    For each Movie
        Print Movie
    Endfor
Endfor
```



a. All actors will be printed. In addition, for every actor who is registered in at least one movie, all movies that have that actor entered will be printed as well.

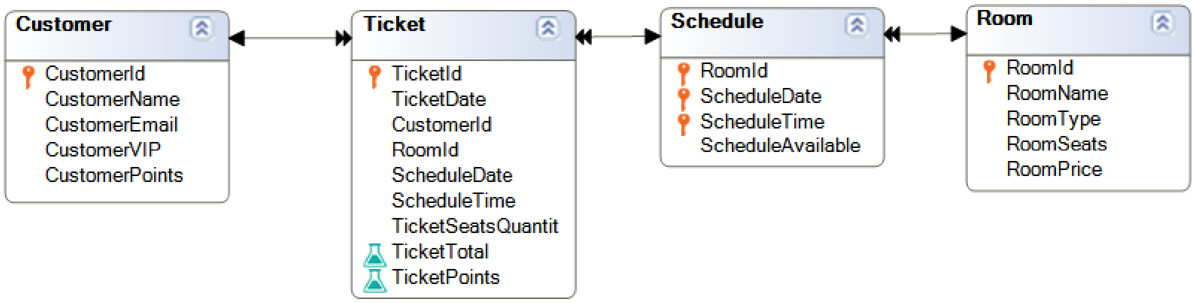
b. All actors will be printed. In addition, for every actor all the movies registered will also be printed.

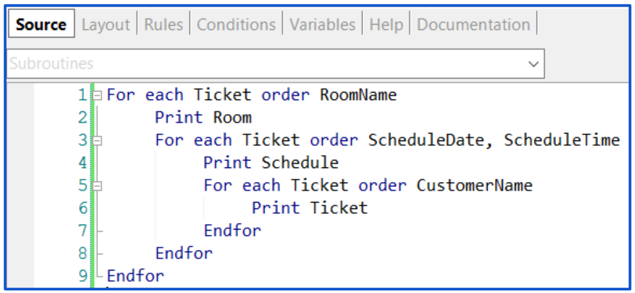
c. All actors will be printed. In addition, for every actor who is registered in at least one movie, all the movies registered will be printed as well.

d. All actors will be printed. In addition, for every actor who is registered in at least one movie, the movies whose country is the same as that of the actor will be printed as well.

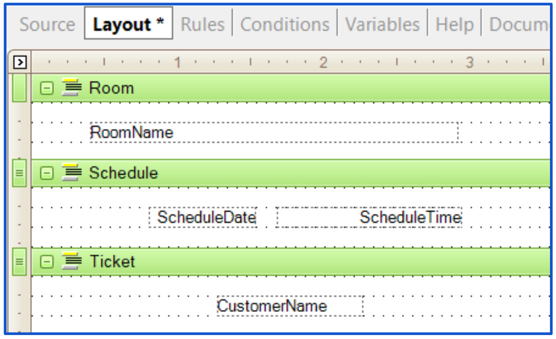
6. Consider the table diagram and the procedure shown below.

From this implementation, select the correct option.





```
For each Ticket order RoomName
    Print Room
    For each Ticket order ScheduleDate, ScheduleTime
        Print Schedule
        For each Ticket order CustomerName
            Print Ticket
        Endfor
    Endfor
Endfor
```



a. All the venues that appear in at least one ticket will be displayed on screen.
For each one of them, all the screenings entered for that venue, which are included in a ticket, grouped by date (ScheduleDate) and time (ScheduleTime) of the screening.

And for each of those screenings, all the customers who have purchased a ticket for that screening, grouped by name.

b. All the venues that appear in at least one ticket will be displayed on screen.
For each one of them, all the screenings entered for that venue, which are included in a ticket, grouped by date (ScheduleDate) and time (ScheduleTime) of the screening.

And for each of those screenings, all the customers who have purchased a ticket for that screening, sorted by name.

c. All the venues that appear in at least one ticket will be displayed on screen.
For each venue, all the screenings that have been entered into the system will be displayed, sorted by date (ScheduleDate) and time (ScheduleTime) of the screening (they will be repeated if there are screenings with the same date and time). And for each of those screenings, all customers sorted by name.

d. None of the above.

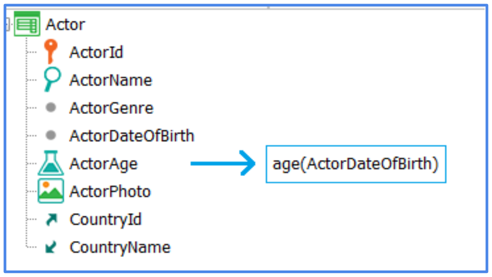
7. The application for a movie theater complex has the following implementation.

Look at the transactions and the procedure shown.

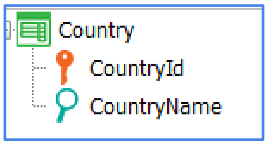
A list is needed of all countries, **without repetitions, that have registered actors**, each one with the average age of its actors.

Determine the option you consider correct.

Note: One of the functionalities of the Age() function is to calculate age from a date of birth.

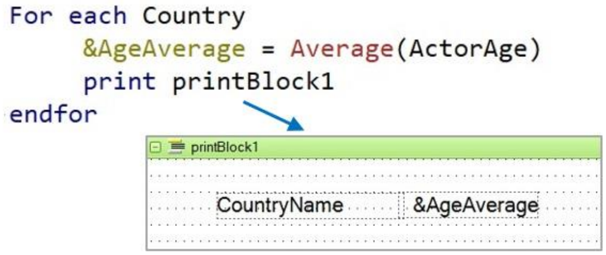


Actor
ActorId
ActorName
ActorGenre
ActorDateOfBirth
ActorAge → age(ActorDateOfBirth)
ActorPhoto
CountryId
CountryName



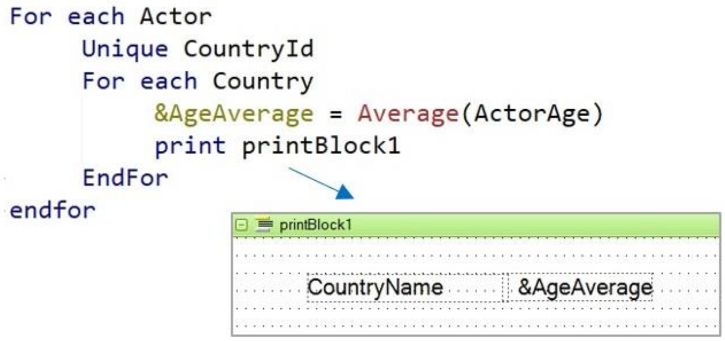
Country
CountryId
CountryName

a)



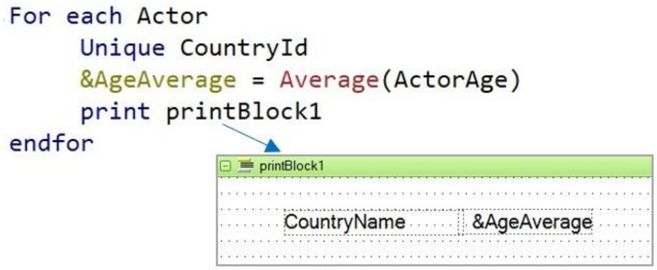
```
For each Country
     &AgeAverage = Average(ActorAge)
     print printBlock1
endfor
```

b)



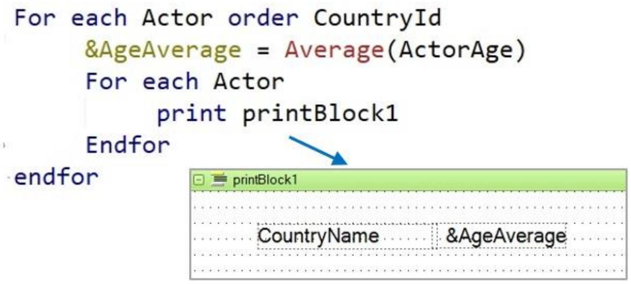
```
For each Actor
     Unique CountryId
     For each Country
          &AgeAverage = Average(ActorAge)
          print printBlock1
     EndFor
endfor
```

c)



```
For each Actor
     Unique CountryId
     &AgeAverage = Average(ActorAge)
     print printBlock1
endfor
```

d)

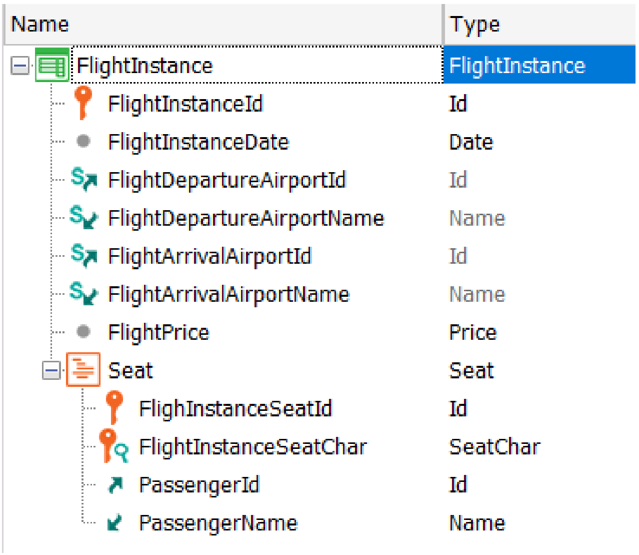


```
For each Actor order CountryId
    &AgeAverage = Average(ActorAge)
    For each Actor
        print printBlock1
    Endfor
endfor
```

printBlock1

CountryName &AgeAverage

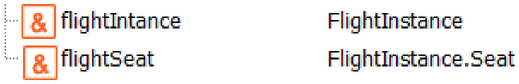
8. A simplified application is being developed for a travel agency. There is a FlightInstance transaction that records flights. For each one, in addition to the date of the flight, its departure airport and arrival airport (subtype groups), its price and the registered passengers, with their seats, are recorded.



Name	Type
FlightInstance	FlightInstance
FlightInstanceId	Id
FlightInstanceDate	Date
FlightDepartureAirportId	Id
FlightDepartureAirportName	Name
FlightArrivalAirportId	Id
FlightArrivalAirportName	Name
FlightPrice	Price
Seat	Seat
FlighInstanceSeatId	Id
FlightInstanceSeatChar	SeatChar
PassengerId	Id
PassengerName	Name

We want to substitute one passenger for another on all flights after today's date. To do so, a procedure will be programmed that will receive by parameters the passenger to be changed, &OldPassengerId, and the passenger that will replace it, &NewPassengerId.

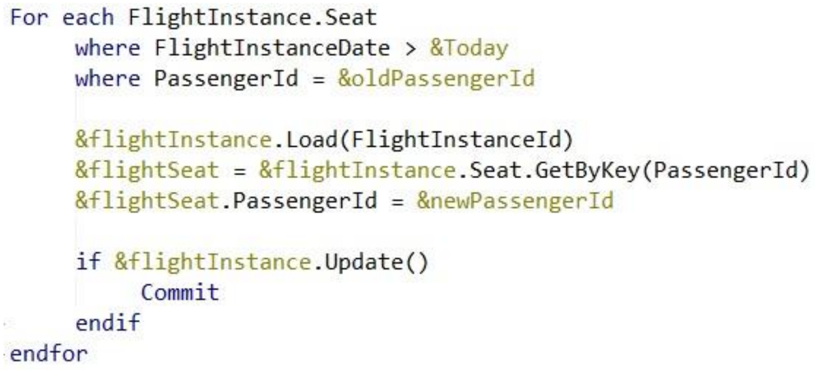
In the procedure, there are two variables of business component type:



& flightIntance	FlightInstance
& flightSeat	FlightInstance.Seat

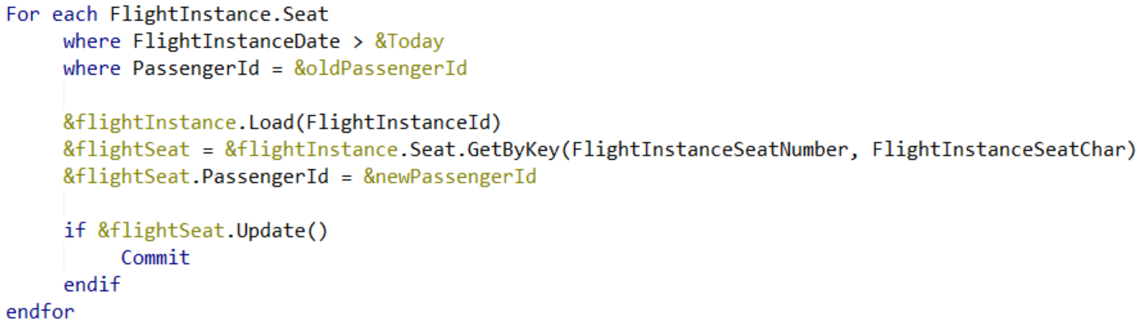
The following options are possible implementations of the requirement. Indicate the correct one.

a.



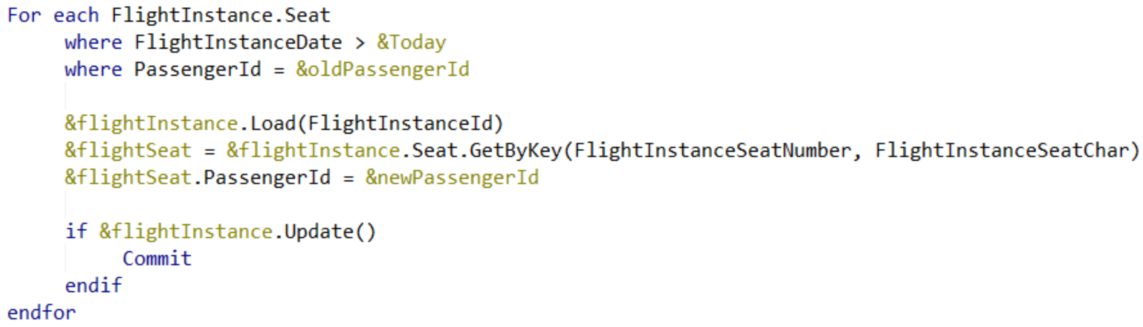
```
For each FlightInstance.Seat
     where FlightInstanceDate > &Today
     where PassengerId = &oldPassengerId

     &flightInstance.Load(FlightInstanceId)
     &flightSeat = &flightInstance.Seat.GetByKey(PassengerId)
     &flightSeat.PassengerId = &newPassengerId

     if &flightInstance.Update()
          Commit
     endif
endfor
```

b.



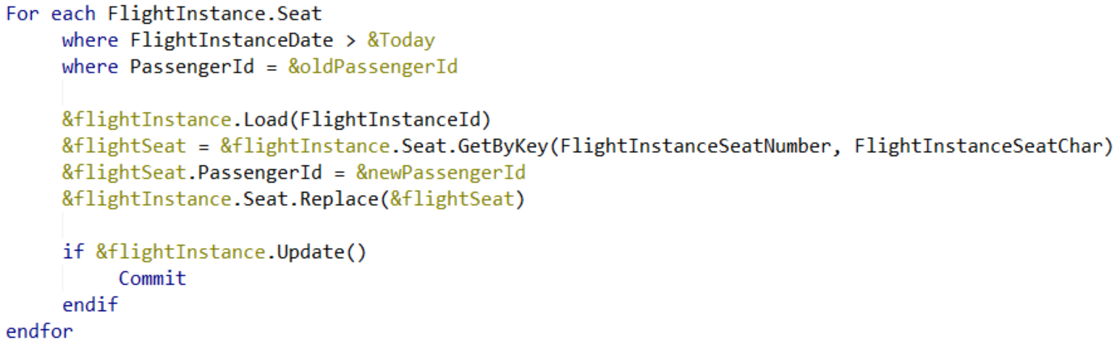
```
For each FlightInstance.Seat
     where FlightInstanceDate > &Today
     where PassengerId = &oldPassengerId

     &flightInstance.Load(FlightInstanceId)
     &flightSeat = &flightInstance.Seat.GetByKey(FlightInstanceSeatNumber, FlightInstanceSeatChar)
     &flightSeat.PassengerId = &newPassengerId

     if &flightSeat.Update()
          Commit
     endif
endfor
```

c.



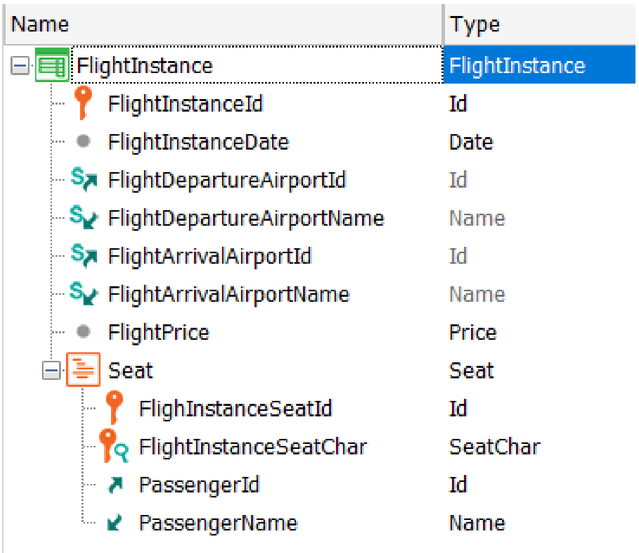
```
For each FlightInstance.Seat
     where FlightInstanceDate > &Today
     where PassengerId = &oldPassengerId

     &flightInstance.Load(FlightInstanceId)
     &flightSeat = &flightInstance.Seat.GetByKey(FlightInstanceSeatNumber, FlightInstanceSeatChar)
     &flightSeat.PassengerId = &newPassengerId

     if &flightInstance.Update()
          Commit
     endif
endfor
```

d.



```
For each FlightInstance.Seat
     where FlightInstanceDate > &Today
     where PassengerId = &oldPassengerId

     &flightInstance.Load(FlightInstanceId)
     &flightSeat = &flightInstance.Seat.GetByKey(FlightInstanceSeatNumber, FlightInstanceSeatChar)
     &flightSeat.PassengerId = &newPassengerId
     &flightInstance.Seat.Replace(&flightSeat)

     if &flightInstance.Update()
          Commit
     endif
endfor
```

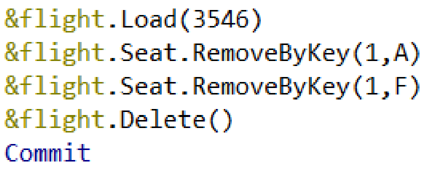
9. There is a FlightInstance transaction that records flights. For each one, in addition to the date of the flight, its departure airport and arrival airport (subtype groups), its price and the registered passengers, with their seats, are recorded.



Name	Type
FlightInstance	FlightInstance
FlightInstanceId	Id
FlightInstanceDate	Date
FlightDepartureAirportId	Id
FlightDepartureAirportName	Name
FlightArrivalAirportId	Id
FlightArrivalAirportName	Name
FlightPrice	Price
Seat	Seat
FlighInstanceSeatId	Id
FlightInstanceSeatChar	SeatChar
PassengerId	Id
PassengerName	Name

Seats 1A and 1F are to be deleted from a particular flight (3546) through a variable &flight Business Component of FlightInstance.

For this purpose, the code shown below has been written:



```
&flight.Load(3546)
&flight.Seat.RemoveByKey(1,A)
&flight.Seat.RemoveByKey(1,F)
&flight.Delete()
Commit
```

Select the correct option from the following:

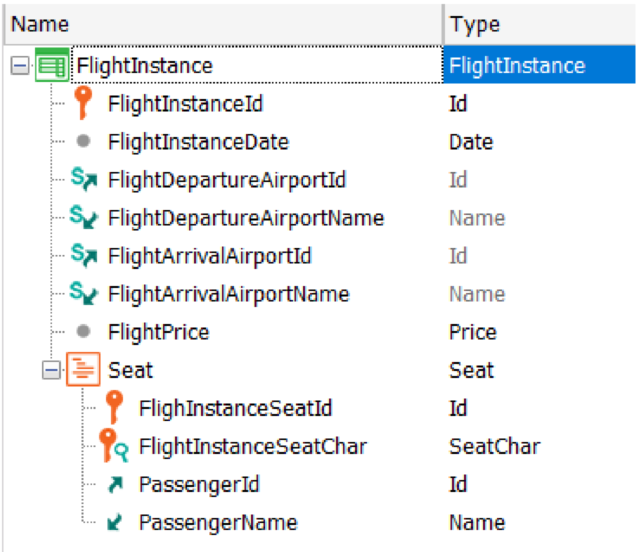
a. The above code is correct and does what we want.

b. The above code is incorrect. Even though both lines will be deleted from the Seat collection in the BC, since the executed method is Delete, everything will be deleted, including the header and lines. To be correct, &flight.Update() should have been written in the 4th line.

c. The above code is incorrect. It would be correct if we deleted the 4th line—&flight.Delete()—leaving the rest unchanged.

d. The above code is incorrect for other reasons.

10. There is a FlightInstance transaction that records flights. For each one, in addition to the date of the flight, its departure airport and arrival airport (subtype groups), its price and the registered passengers, with their seats, are recorded.



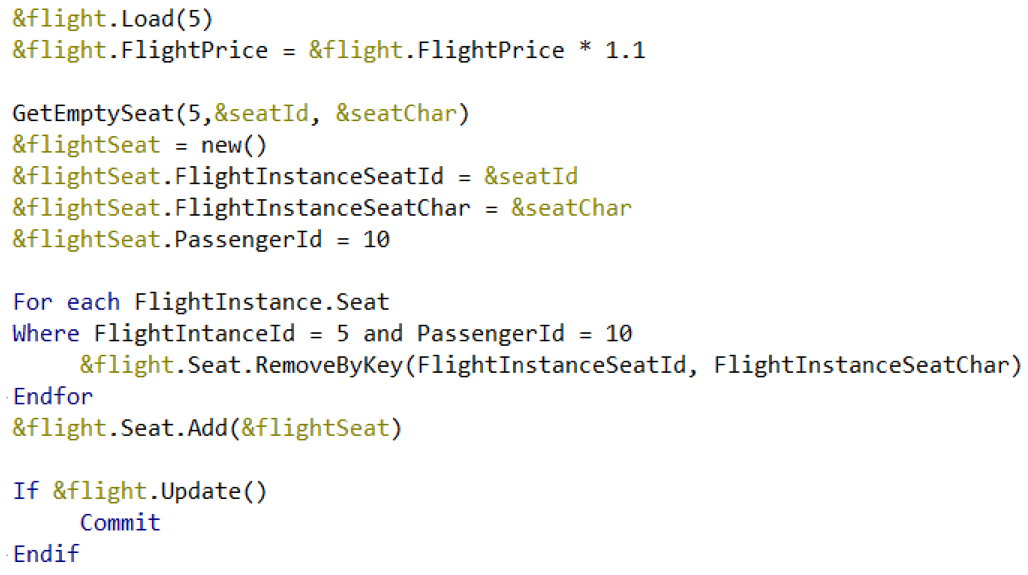
Name	Type
FlightInstance	FlightInstance
FlightInstanceId	Id
FlightInstanceDate	Date
FlightDepartureAirportId	Id
FlightDepartureAirportName	Name
FlightArrivalAirportId	Id
FlightArrivalAirportName	Name
FlightPrice	Price
Seat	Seat
FlighInstanceSeatId	Id
FlightInstanceSeatChar	SeatChar
PassengerId	Id
PassengerName	Name

We want to change the seat of a given passenger, 10, on a given flight, 5, and modify the price of that flight, increasing its value by 10%.

Suppose that {FlightInstanceId, PassengerId} is a candidate key of FlightInstanceSeat, so that one passenger can only be in one seat on the same flight. To change the passenger's seat we will delete the passenger's seat **record** for that flight and insert it again, with only the seat changed.

Suppose we have the GetEmptySeat procedure, which by passing it the flight ID returns a SeatId and SeatChar not occupied on the flight, so as to reassign them to the passenger to be moved.

To implement the requirement, the following code has been programmed in a Web Panel event, where &flight is of BC FlightInstance data type and &flightSeat is of BC FlightInstance.Seat data type.

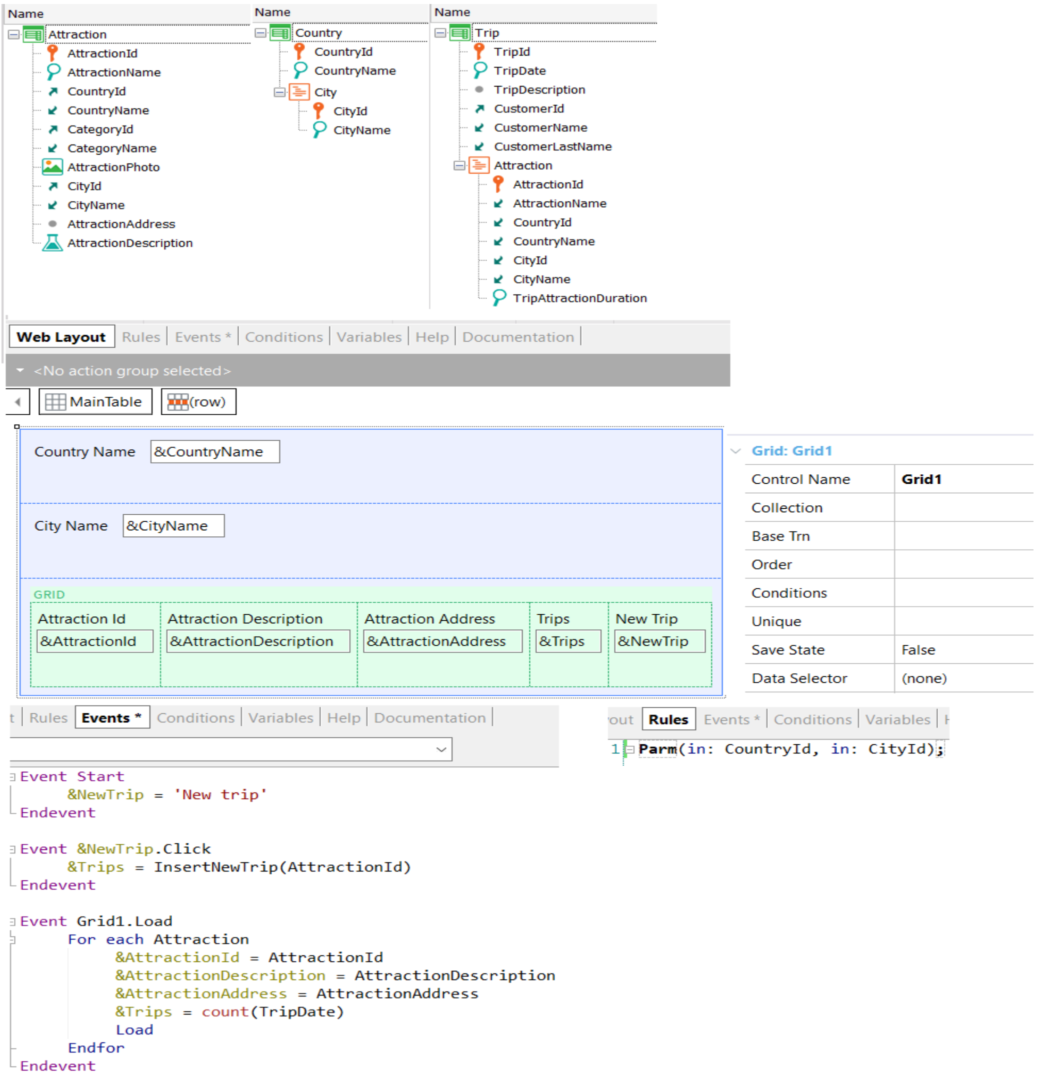


```
&flight.Load(5)
&flight.FlightPrice = &flight.FlightPrice * 1.1

GetEmptySeat(5,&seatId, &seatChar)
&flightSeat = new()
&flightSeat.FlightInstanceSeatId = &seatId
&flightSeat.FlightInstanceSeatChar = &seatChar
&flightSeat.PassengerId = 10

For each FlightInstance.Seat
Where FlightIntanceId = 5 and PassengerId = 10
     &flight.Seat.RemoveByKey(FlightInstanceSeatId, FlightInstanceSeatChar)
Endfor
&flight.Seat.Add(&flightSeat)

If &flight.Update()
     Commit
Endif
```

Indicate which one of the following options is correct.

a. The implementation is correct.
b. It is poorly implemented because the line insertion is wrong.
c. It is poorly implemented because the line deletion is wrong.
d. None of the above.

11. In an application for a travel agency, with the transactions shown in the image, you want to see the attractions of a city in a certain country, with the trips that have been made to each attraction. In addition, it should allow creating a new trip to visit a particular attraction. The following Web Panel has been implemented. Determine its base table.



```
Name
Attraction
    AttractionId
    AttractionName
    CountryId
    CountryName
    CategoryId
    CategoryName
    AttractionPhoto
    CityId
    CityName
    AttractionAddress
    AttractionDescription
```

```
Name
Country
    CountryId
    CountryName
    City
        CityId
        CityName
```

```
Name
Trip
    TripId
    TripDate
    TripDescription
    CustomerId
    CustomerName
    CustomerLastName
    Attraction
        AttractionId
        AttractionName
        CountryId
        CountryName
        CityId
        CityName
        TripAttractionDuration
```

Country Name &CountryName

City Name &CityName

Attraction Id	Attraction Description	Attraction Address	Trips	New Trip
&AttractionId	&AttractionDescription	&AttractionAddress	&Trips	&NewTrip

Grid: Grid1

Control Name	Grid1
Collection	
Base Trn	
Order	
Conditions	
Unique	
Save State	False
Data Selector	(none)

```
Event Start
      &NewTrip = 'New trip'
Endevent

Event &NewTrip.Click
      &Trips = InsertNewTrip(AttractionId)
Endevent

Event Grid1.Load
      For each Attraction
            &AttractionId = AttractionId
            &AttractionDescription = AttractionDescription
            &AttractionAddress = AttractionAddress
            &Trips = count(TripDate)
            Load
      Endfor
Endevent
```

```
Parm(in: CountryId, in: CityId);
```

a) ATTRACTION

b) COUNTRYCITY

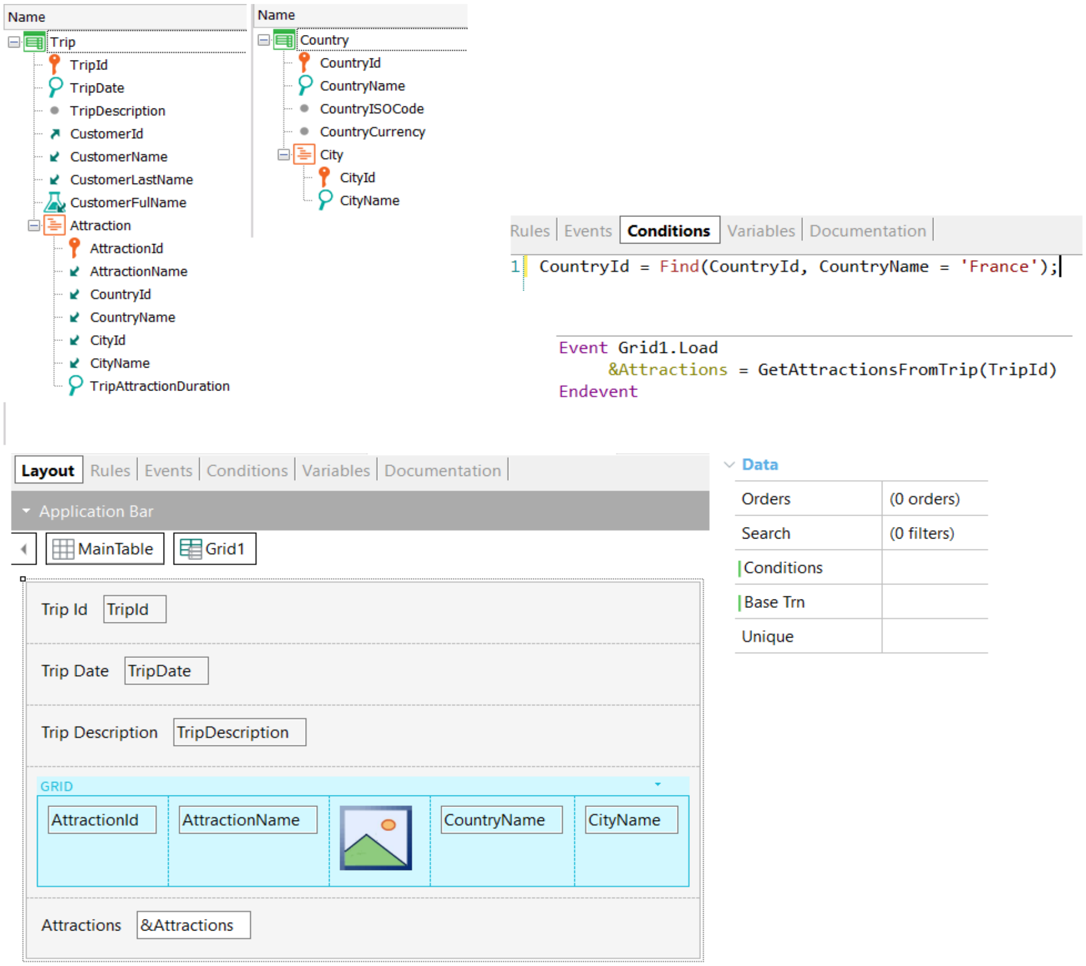
c) COUNTRY

d) TRIPATTRACTION

e) The Web Panel doesn’t have a base table

12. In the context of an application for a travel agency, the panel object shown below is implemented. The transactions involved are added too. The default values left unchanged are not mentioned here.

Select the option you consider correct.



```
Name
Trip
    TripId
    TripDate
    TripDescription
    CustomerId
    CustomerName
    CustomerLastName
    CustomerFulName
    Attraction
        AttractionId
        AttractionName
        CountryId
        CountryName
        CityId
        CityName
        TripAttractionDuration
```

```
Name
Country
    CountryId
    CountryName
    CountryISOCode
    CountryCurrency
    City
        CityId
        CityName
```

```
CountryId = Find(CountryId, CountryName = 'France');
```

```
Event Grid1.Load
	&Attractions = GetAttractionsFromTrip(TripId)
Endevent
```

Layout | Rules | Events | Conditions | Variables | Documentation
Application Bar
MainTable
Grid1
GRID

Trip Id	TripId
Trip Date	TripDate
Trip Description	TripDescription
Attractions	&Attractions

Grid: AttractionId | AttractionName | CountryName | CityName

Data

Orders	(0 orders)
Search	(0 filters)
Conditions	
Base Trn	
Unique	

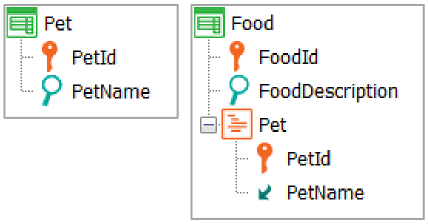
a) The base table of the panel object is TRIPATTRACTION.

b) The base table of the fixed part is TRIP and that of the grid is ATTRACTION.

c) The base table of the fixed part is TRIP and that of the grid is TRIPATTRACTION.

d) The base table of the fixed part is TRIPATTRACTION and that of the grid is TRIPATTRACTION.

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



- Pet
  - PetId
  - PetName

- Food
  - FoodId
  - FoodDescription
  - Pet
    - PetId
    - PetName

Rules:

1. FoodDetail(FoodId) on AfterComplete;
2. Reservation(FoodId) on AfterInsert;
3. StockControl(FoodId) on AfterLevel level PetId;

***a)* –** 2), 3), 1)

***b)* –** 3), 2), 1)

***c)* –** 3), 1), 2)

***d)* –** The rules are triggered in the order in which they are declared.

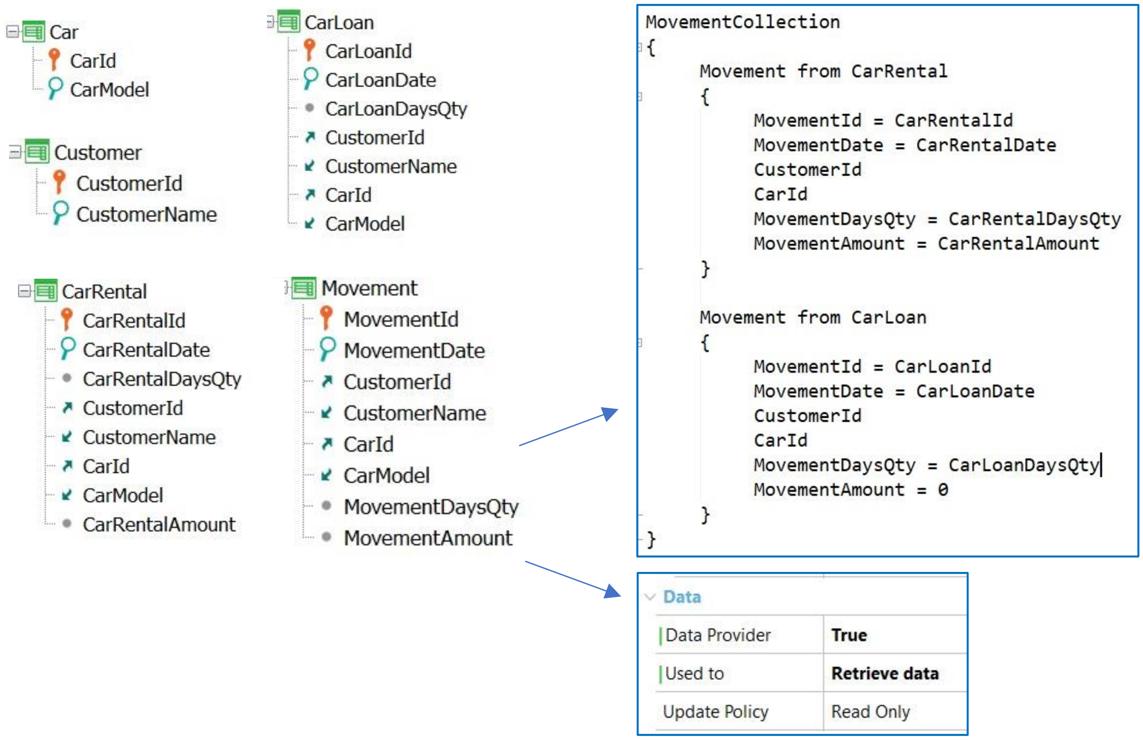
14. An application is being designed for a Car rental company.

It provides car rentals and loans to its customers. It is necessary to unify the information on car rentals and loans in order to list this information sorted by date.

The company also needs to record promotions (Promotion) for these car rentals and loans.

Note: Take into account that all IDs are autonumbered.

Consider this reality, and from the implementation shown, determine what you consider correct:



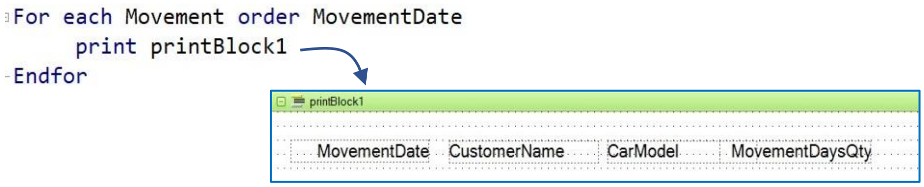
```
Car
    CarId
    CarModel

Customer
    CustomerId
    CustomerName

CarRental
    CarRentalId
    CarRentalDate
    CarRentalDaysQty
    CustomerId
    CustomerName
    CarId
    CarModel
    CarRentalAmount

CarLoan
    CarLoanId
    CarLoanDate
    CarLoanDaysQty
    CustomerId
    CustomerName
    CarId
    CarModel

Movement
    MovementId
    MovementDate
    CustomerId
    CustomerName
    CarId
    CarModel
    MovementDaysQty
    MovementAmount
```

```
MovementCollection
{
        Movement from CarRental
        {
                MovementId = CarRentalId
                MovementDate = CarRentalDate
                CustomerId
                CarId
                MovementDaysQty = CarRentalDaysQty
                MovementAmount = CarRentalAmount
        }

        Movement from CarLoan
        {
                MovementId = CarLoanId
                MovementDate = CarLoanDate
                CustomerId
                CarId
                MovementDaysQty = CarLoanDaysQty
                MovementAmount = 0
        }
}
```

Data

Data Provider	**True**
Used to	**Retrieve data**
Update Policy	Read Only

a) The implementation displayed is correct but incomplete. To fully meet the requirement, the only thing left is to list the complete information.

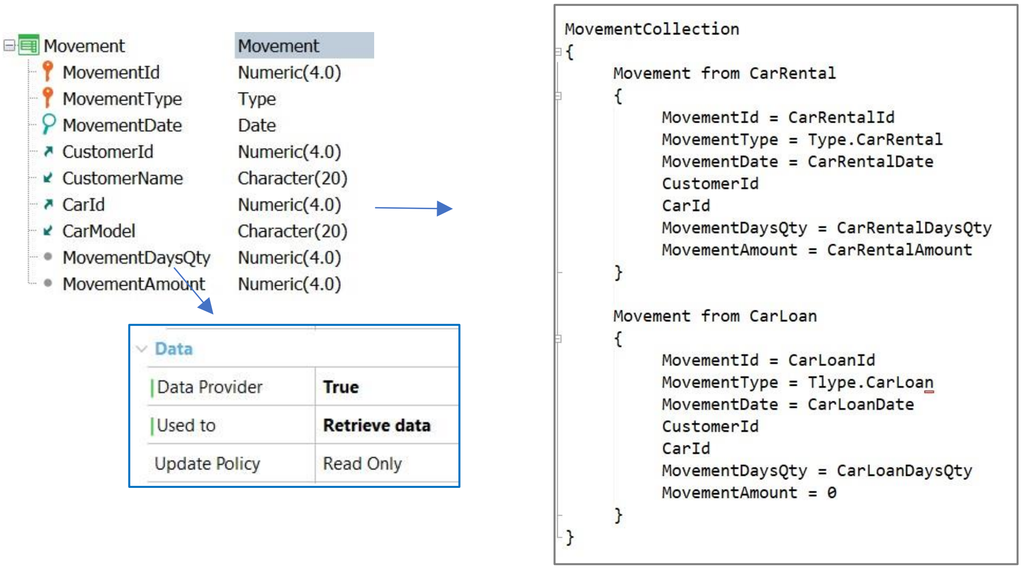


```
For each Movement order MovementDate
      print printBlock1
Endfor
```

printBlock1

MovementDate CustomerName CarModel MovementDaysQty

b) The definition of the dynamic transaction (Movement) is not correct, since it is not possible to know whether it is a rental or a loan. The correct definition is shown below, and the requirement is met with this implementation together with option a).

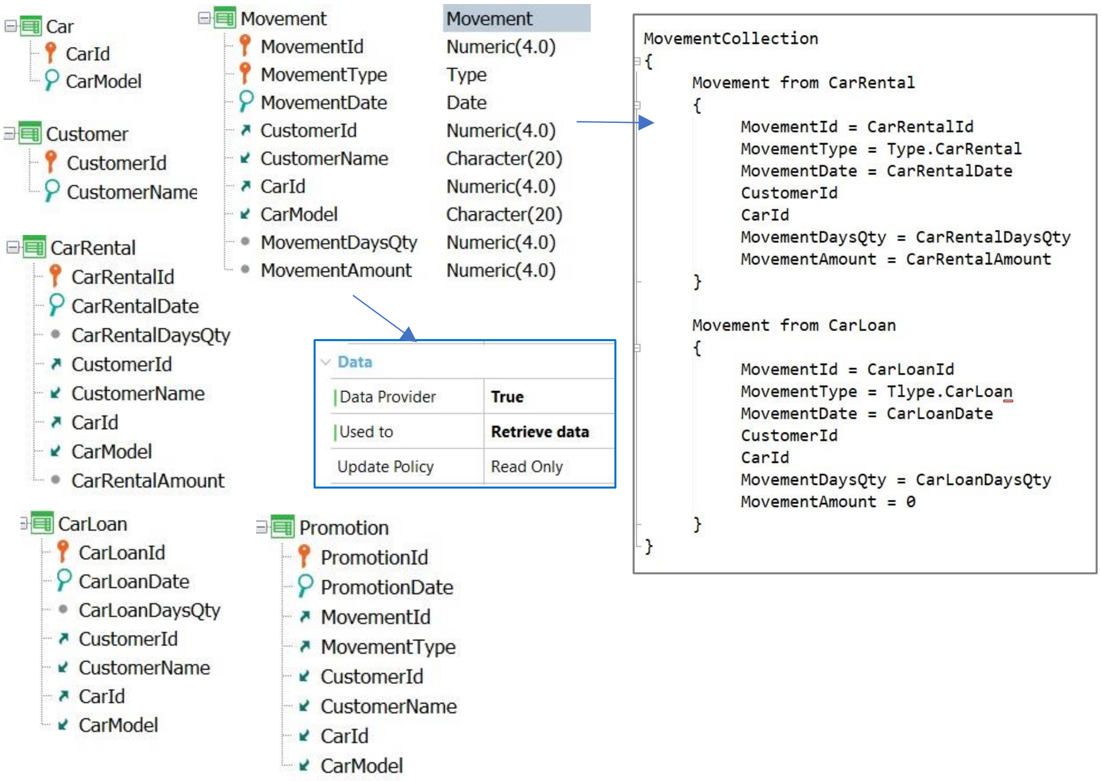


Movement	Movement
MovementId	Numeric(4.0)
MovementType	Type
MovementDate	Date
CustomerId	Numeric(4.0)
CustomerName	Character(20)
CarId	Numeric(4.0)
CarModel	Character(20)
MovementDaysQty	Numeric(4.0)
MovementAmount	Numeric(4.0)

Data	
Data Provider	**True**
Used to	**Retrieve data**
Update Policy	Read Only

```
MovementCollection
{
     Movement from CarRental
     {
          MovementId = CarRentalId
          MovementType = Type.CarRental
          MovementDate = CarRentalDate
          CustomerId
          CarId
          MovementDaysQty = CarRentalDaysQty
          MovementAmount = CarRentalAmount
     }

     Movement from CarLoan
     {
          MovementId = CarLoanId
          MovementType = Tlype.CarLoan
          MovementDate = CarLoanDate
          CustomerId
          CarId
          MovementDaysQty = CarLoanDaysQty
          MovementAmount = 0
     }
}
```

c) What is implemented in b) is correct. To also record rental and loan promotions (Movement) the transaction shown must be defined. The solution that fully meets the requirement is as follows:



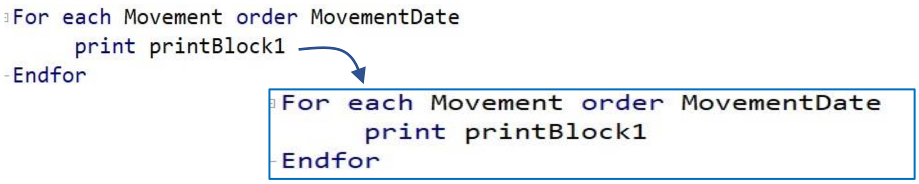
```
Car
   CarId
   CarModel

Customer
   CustomerId
   CustomerName

CarRental
   CarRentalId
   CarRentalDate
   CarRentalDaysQty
   CustomerId
   CustomerName
   CarId
   CarModel
   CarRentalAmount

CarLoan
   CarLoanId
   CarLoanDate
   CarLoanDaysQty
   CustomerId
   CustomerName
   CarId
   CarModel
```

Movement	Movement
MovementId	Numeric(4.0)
MovementType	Type
MovementDate	Date
CustomerId	Numeric(4.0)
CustomerName	Character(20)
CarId	Numeric(4.0)
CarModel	Character(20)
MovementDaysQty	Numeric(4.0)
MovementAmount	Numeric(4.0)

Data	
Data Provider	**True**
Used to	**Retrieve data**
Update Policy	Read Only

```
Promotion
   PromotionId
   PromotionDate
   MovementId
   MovementType
   CustomerId
   CustomerName
   CarId
   CarModel
```

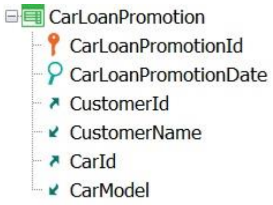
```
MovementCollection
{
     Movement from CarRental
     {
          MovementId = CarRentalId
          MovementType = Type.CarRental
          MovementDate = CarRentalDate
          CustomerId
          CarId
          MovementDaysQty = CarRentalDaysQty
          MovementAmount = CarRentalAmount
     }

     Movement from CarLoan
     {
          MovementId = CarLoanId
          MovementType = Tlype.CarLoan
          MovementDate = CarLoanDate
          CustomerId
          CarId
          MovementDaysQty = CarLoanDaysQty
          MovementAmount = 0
     }
}
```



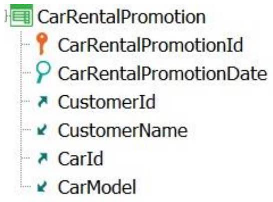
```
For each Movement order MovementDate
      print printBlock1
Endfor
```

```
For each Movement order MovementDate
      print printBlock1
Endfor
```

d) The solution proposed in option c) is correct except for the definition of the Promotion transaction. The pair consisting of MovementId, MovementType does not make up a foreign key because they do not exist as attributes stored in a table. Rental promotions and loan promotions must be recorded separately, as shown:



```
CarLoanPromotion
  CarLoanPromotionId
  CarLoanPromotionDate
  CustomerId
  CustomerName
  CarId
  CarModel
```



```
CarRentalPromotion
  CarRentalPromotionId
  CarRentalPromotionDate
  CustomerId
  CustomerName
  CarId
  CarModel
```

e) None of the above options are correct.

Answers

1. b

Justification by subparagraphs:

a. Incorrect. The first 2 rules do not meet the requirement that they are evaluated on the server and not on the client side. As a result, they do not meet what is requested.

b. Correct. The Msg rule meets what is requested in the first item as it will show the message that the city name is empty when leaving the field. To meet the second requirement, a triggering event must be assigned to both rules to be executed after confirming but before inserting the header data, and this is met with the BeforeInsert event.

c. Incorrect. The Msg rule is properly defined, but the rules in the second requirement do not meet what is requested. Although they are associated with a triggering event, it will be executed after the insertion of the header and detail data, a moment before committing and not before inserting the first level as requested.

d. Incorrect. Option (b) meets the requirement.

2. b

Justification by subparagraphs:

a. Incorrect. It is possible to use a formula attribute to perform the operations mentioned in the statement.

b. Correct. When there are 2 rules that use the same attribute, the rule that modifies the database will be executed first and then the one that queries it; therefore, the Subtract rule will be triggered first and the Error rule second. For this reason, it should evaluate CountryCityMaxDuration < 0 and not against CityTourDuration.

c. Incorrect. What the statement says is incorrect. The Subtract rule has the intelligence to determine the calculation to be performed depending on the mode in which it is being executed; therefore, it does not require conditioning to a triggering event that also will not allow the validation to be on the client side.

d. Incorrect. Option (b) meets the requirement.

3. c

Justification by subparagraphs:

a. Incorrect. Although both options meet the requirement, they are not equivalent. In the case of option 1, when editing the design of the Director transaction, it will affect the rest *of the objects already defined that make reference to the director, which will imply making modifications to them.*

b. Incorrect, since the objects that make reference to a transaction that has had attributes modified are not automatically updated; these changes must be made manually.

c. Correct. In option 2 it was identified that the Director structure should not be modified so as not to affect the previously defined objects, so the best alternative was to make a group of subtypes of the whole Director structure in order to invoke it in Movie and meet the requirement.

d. Incorrect. The proposed solution is also viable and meets the requirement; what makes it incorrect is the statement indicating that the previous options do not meet the requirement.

4. a

Justification by subparagraphs:

a. Correct. The when clause evaluates that the condition preceding it is fulfilled and if so, it executes the specified order.

b. Incorrect. The first order is properly defined but it doesn't do anything because since it has no Where there are no records that don't fulfill the filter and it never enters the When none.

c. Incorrect. The conditioning of the orders is done with the When clause applied to each one. The syntax shown is incorrect.

d. Incorrect. The requirement is met with a single For each, declaring the possible orders conditioned with the When clause.

5. d

Justification by subparagraphs:

a. Incorrect. It will not print all the movies that have that actor, only those whose country is the same as the actor's, since an indirect Join is being made by CountryId, which is the attribute in common between both tables.

b. Incorrect. The movies whose country is the same as the actor's will be printed and not all the movies registered. It must be identified that an indirect Join by CountryId, which is the attribute in common between both tables, is being made.

c. Incorrect. Not all the movies registered by Actor will be printed, since an indirect Join is being made by CountryId, which is the attribute in common between both tables. Therefore, it will print the movies whose country is the same as the actor's country.

d. Correct. The relationship between the Actor and Movie table is indirect 1-N, through the Country table. The attribute both tables have in common is CountryId, and a Join will be made by it, giving as a result what the option indicates.

6. b

Justification by subparagraphs:

a. Incorrect. It will not be grouped by name in the last For each, but it will be ordered by it.

b. Correct. What is shown in the source is a double control break where the first 2 For each commands indicate the criteria by which the information will be grouped; the innermost For each has the ordered data of the customers who bought tickets for that room in that performance.

c. Incorrect. The performances will not be repeated, but grouped.

d. Incorrect. Option (b) meets the requirement.

7. c

Justification by subparagraphs:

a. Incorrect. Although the implementation shows the average ages per country, since the base table of the For each is COUNTRY, the list may show countries that do not have registered actors.

b. Incorrect. The unique clause does not apply to nested For each commands.

c. Correct. The unique clause will allow us to have a list without repeated elements and also acts as a condition in the Avarage formula to average only those customer ages that belong to that country.

d. Incorrect. An attempt is made to solve this problem by means of a control break, but it is not correct.

8. c

Justification by subparagraphs:

a. Incorrect. The error is found when trying to be positioned in the passenger's seat record since the correct thing is to be positioned in the PK of the sublevel to identify the passenger to be updated, as follows: &FlightSeat= &FlightInstance.Seat.GetbyKey(FlightInstanceSeatId,FlightInstanceSeatChar).

b. Incorrect. The error is found at the moment of updating because the Update method is not associated with the &FlightSeat variable of the sublevel, but it must be done on &FlightInstace.Update().

c. Correct. The table FlightInstance.Seat is accessed and the filters are applied correctly to identify the flights after the current date; the Load method is correct to be positioned in the flights and the same goes for the use of the GetByKey method that is positioned in the PK of the sublevel where the passenger is located; the change is made and then the Update method associated with the &FlightInstance variable, ending with the commit.

d. Incorrect. There is no Replace method to update an attribute.

9. b

Justification by subparagraphs:

a. Incorrect. The correct answer is option 2. The Delete method will delete the entire Flight and will not update it, which is what is needed.

b. Correct. The request involves deleting 2 seats, from a particular flight, which implies an update of the flight. After indicating the seats to be removed in line 2 and 3 we need to use the Update method as indicated in the statement.

c. Incorrect. Working with BC is analogous to working with the transaction. It is not enough to select the lines to be removed; it is necessary to end with an operation to the DB and it is obtained using the Update method.

d. Incorrect. Option (b) meets the requirement.

10. a

General Justification:

The correct answer is option (a). The implementation is correct. The Load method positions us in the flight we want to work with and line 2 updates the price of the flight. The Procedure is invoked to return the values of the available seat, where we will make the insertion as shown from line 4 to 7. In order to delete the previous seat a For each is done to identify the record to be deleted; once deleted, the line &Flight.Seat.Add(&FlightSeat) adds the new seat defined from line 4 to 7 to assign the passenger to their new seat. The Update Method updates all the changes.

11. a

General Justification:

When there is a Web Panel with a grid the following parts must be considered to determine if it has a base table: loose attributes in the form (layout), visible and non-visible attributes of a grid, Base Transaction, Order, Conditions, Unique and Data Selector property of the grid as well as the attributes in events without context, i.e. outside of a For each or formula. In this case, the base table is Attraction and is determined by the AttractionId attribute found in the &NewTrip.Click event. The attributes of the Parm rule are not involved in the determination of the base table.

It should be mentioned that the Web Panel was poorly implemented, since the For each of the Load event will make up a control break by primary key with the grid's base table.

12. d

Justification by subparagraphs:

a. Incorrect. Unlike the Web Panel, in a Panel object the determination of the base table of the fixed part is independent of the determination of the grid's base table. For this reason, it is not correct because it is not possible to speak of a "base table of the panel object", even if there is no grid because in that case it would be the base table of the fixed part of the panel.

b. Incorrect: We find TripId, TripDate and TripDescription in the fixed part and there are no buttons or controls that can contain attributes, nor attributes in the Application Bar. We do find CountryId in the Conditions Tab of the Panel (CountryName does not participate because it is in a formula) and there is no Refresh event, so the minimum extended table that contains those attributes is TRIPATTRACTION, and not TRIP.

c. Incorrect. As stated in option (b), the base table of the fixed part is TRIPATTRACTION.

d. Correct. As stated in option (b), the base table of the fixed part is TRIPATTRACTION. To determine the base table of the grid we have the attributes AttractionId, AttractionName, AttractionPhoto, CountryName and CityName. In the general conditions of the Panel we have CountryId and in the Load event we have the TripId attribute that is not in a context (neither in a For Each, nor in a formula) so it is involved in determining the base table of the grid. The minimum extended table that contains these attributes is TRIPATTRACTION, so the base table of the grid is TRIPATTRACTION

13. a

General Justification:

The correct option is (a) resulting in the order: 2,3,1, because the first rule to be executed is that of item 2, which will be triggered after the insertion of the first level of the Food *transaction; then item 3, which will be executed after leaving the Food.Pet level, leaving item 1 in last place as the last one to be executed after commit.*

14. c

Justification by subparagraphs:

a. Incorrect. The definition of Movement is incomplete; the primary key should be modified to identify whether it is a loan or a rental and also an entity to enter promotions is missing.

b. Incorrect. While the definition of the Movement transaction is corrected to be able to identify a rental from a loan, a transaction to define promotions is missing.

c. Correct. A correct definition of the Movement transaction is shown and the Loans transaction is added, as well as the PDF Listing, thus meeting the requirement.

d. Incorrect. The PK of a dynamic transaction of Retrieve type can be used as FK in another transaction. GeneXus will create a pseudo key to be able to execute the integrity controls, so what this option says is not correct.

e. Incorrect. Option (c) meets the requirement.