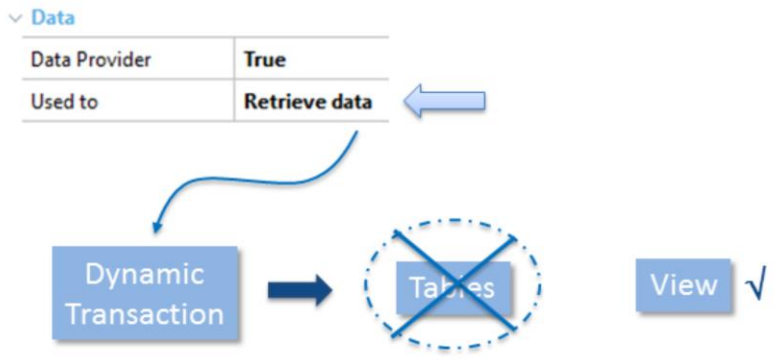
Dynamic Transactions

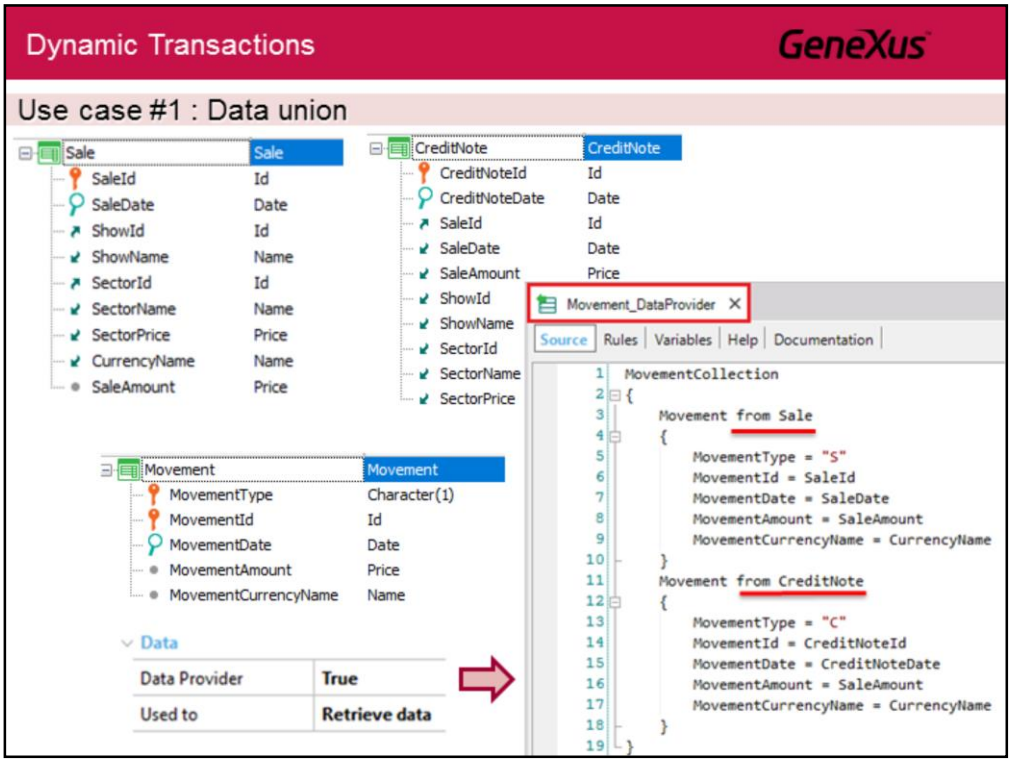
GeneXus 15

Dynamic Transactions

GeneXus



- In the DP, we must specify which data we want to retrieve…:
  - when the Transaction form is executed.
  - when using the attributes.
- A Dynamic Transaction can be referenced as Base Transaction


Dynamic Transactions
GeneXus
Use case #1 : Data union
Sale
Sale
SaleId Id
SaleDate Date
ShowId Id
ShowName Name
SectorId Id
SectorName Name
SectorPrice Price
CurrencyName Name
SaleAmount Price
CreditNote
CreditNote
CreditNoteId Id
CreditNoteDate Date
SaleId Id
SaleDate Date
SaleAmount Price
ShowId
ShowName
SectorId
SectorName
SectorPrice
Movement
Movement
MovementType Character(1)
MovementId Id
MovementDate Date
MovementAmount Price
MovementCurrencyName Name
Data
Data Provider True
Used to Retrieve data
Movement_DataProvider
Source Rules Variables Help Documentation
MovementCollection
{
Movement from Sale
{
MovementType = "S"
MovementId = SaleId
MovementDate = SaleDate
MovementAmount = SaleAmount
MovementCurrencyName = CurrencyName
}
Movement from CreditNote
{
MovementType = "C"
MovementId = CreditNoteId
MovementDate = CreditNoteDate
MovementAmount = SaleAmount
MovementCurrencyName = CurrencyName
}
}

Dynamic Transactions

GeneXus

Use case #1 : Data union

Movement	Movement
MovementType	Character(1)
MovementId	Id
MovementDate	Date
MovementAmount	Price
MovementCurrencyName	Name

Movements 09/27/17

Movement type	Movement Id	Movement Amount	Currency
S	1	200	US Dolar
S	2	3500	Uruguayan Peso
S	3	150	US Dolar
C	1	200	US Dolar

PrintTodayMovements

Source | Layout | Rules | Conditions | Variables | Help | Documentation

Subroutines

```
1  Print Title
2  For each Movement order (MovementType), MovementDate
3      where MovementDate=&Today
4      Print Movement
5  Endfor
```

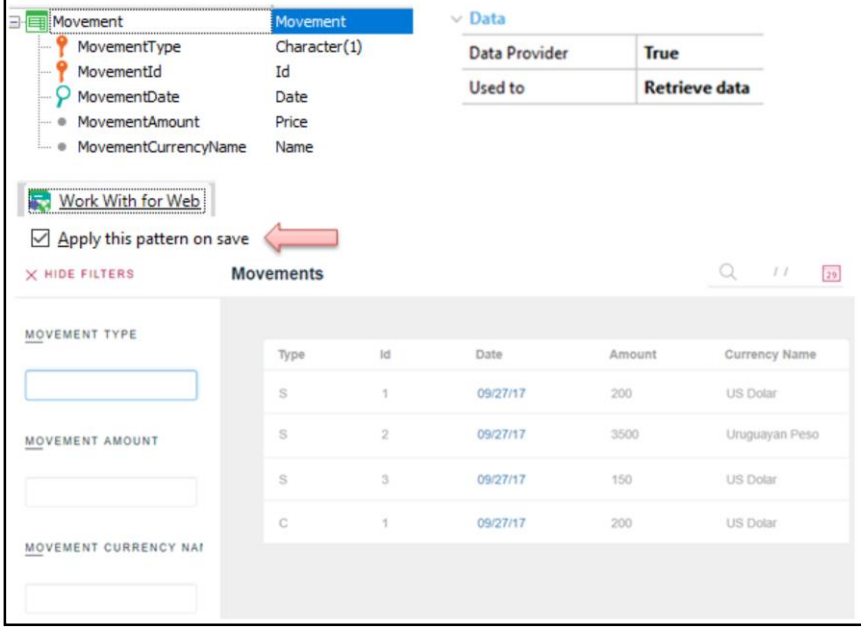
Movement

MovementType | MovementId | MovementAmount | MovementCurrencyName

Dynamic Transactions

GeneXus

Use case #1 : Data union



Dynamic Transactions

GeneXus

Use case #1 : Data union

Movement	Movement
MovementType	Character(1)
MovementId	Id
MovementDate	Date
MovementAmount	Price
MovementCurrencyName	Name

Data

Data Provider	**True**
Used to	**Retrieve data**

Work With for Smart Devices

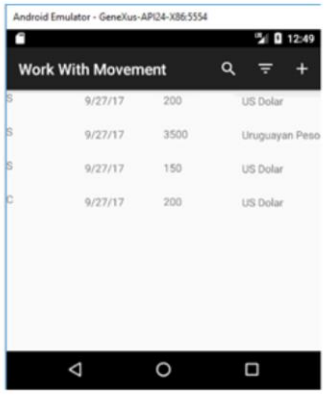
Apply this pattern on save

Level (Movement)

List

Detail

Section (General)



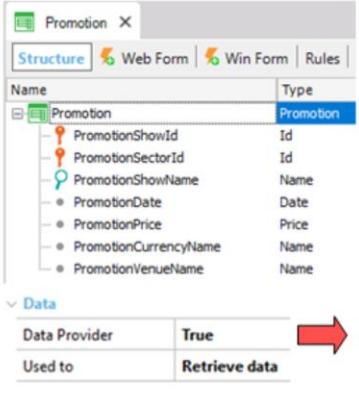
Dynamic Transactions

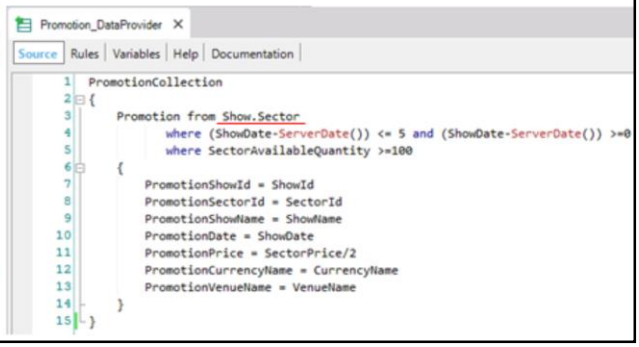
GeneXus

Use case #2 : Modeling a reality

50%

- ✓ Shows to be presented within 5 days or less
- ✓ SectorAvailableQuantity >= 100

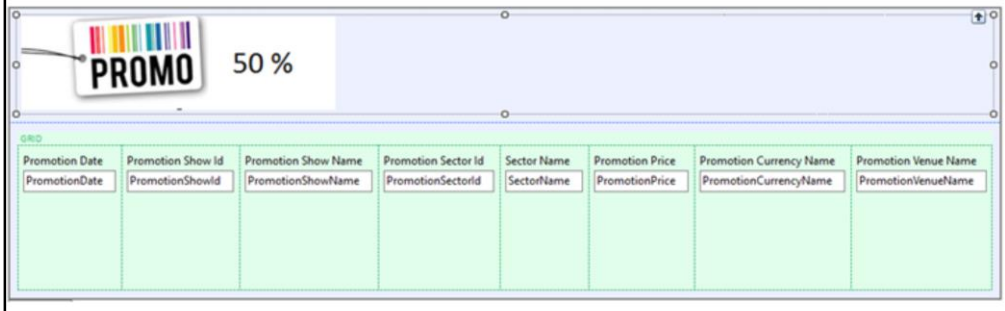




Dynamic Transactions

GeneXus

Use case #2 : Modeling a reality



Dynamic Transactions

GeneXus

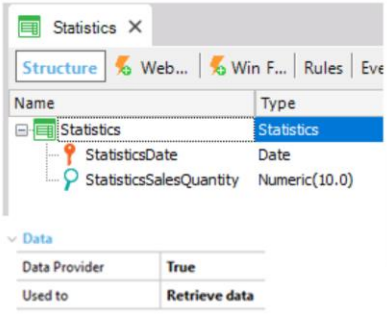
Use case #2 : Modeling a reality

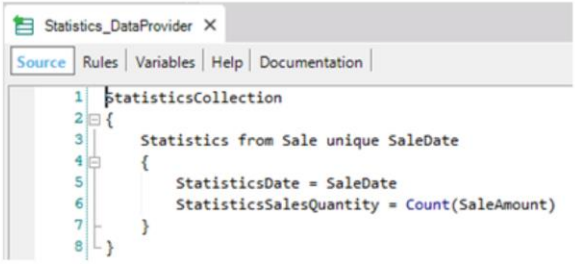
50 %

Promotion Date	Promotion Show Id	Promotion Show Name	Promotion Sector Id	Sector Name	Promotion Venue Name	Promotion Price	Promotion Currency Name
10/03/17	1	Madonna in Concert	1	Orchestra Platinum	Madison Square Garden	100	US Dollar
10/03/17	1	Madonna in Concert	2	Orchestra Gold	Madison Square Garden	75	US Dollar
10/06/17	2	David Bisbal - Tour 2017	1	America Rosrtrum	Estadio Centenario	3500	Uruguayan Peso

Use case #3 : Solving statistics

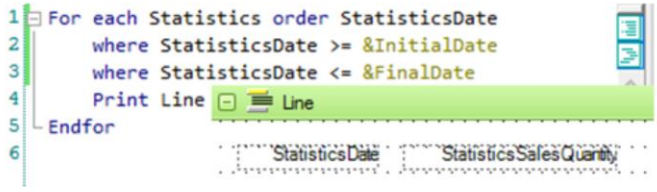
- For a given period, to know the number of sales per day.
- For a given date, to know the number of sales.
- Best day of the year.



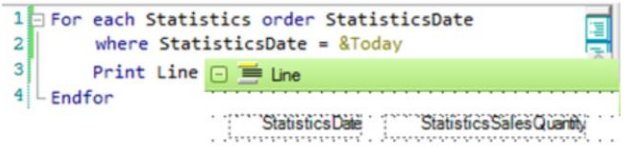


Use case #3 : Solving statistics

- For a given period, to know the number of sales per day



- For a given date, to know the number of sales.



Dynamic Transactions

GeneXus

Use case #3 : Solving statistics

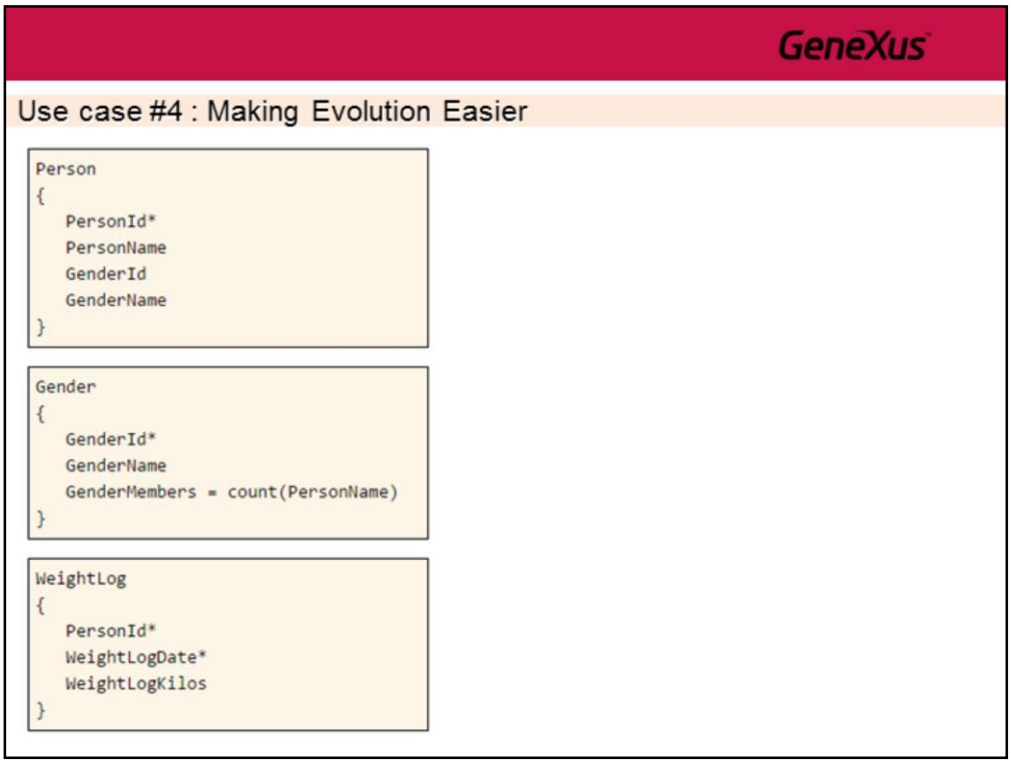
- Best day of the year.

```
1  For each Statistics order (StatisticsSalesQuantity)
2       where StatisticsDate.Year() = &Today.Year()
3       Print Line
4       exit
5  Endfor
```

Line

StatisticsDate | StatisticsSalesQuantity

```
&BestDate = Max(StatisticsSalesQuantity,StatisticsDate.Year()=&Today.Year(),,StatisticsDate)
```



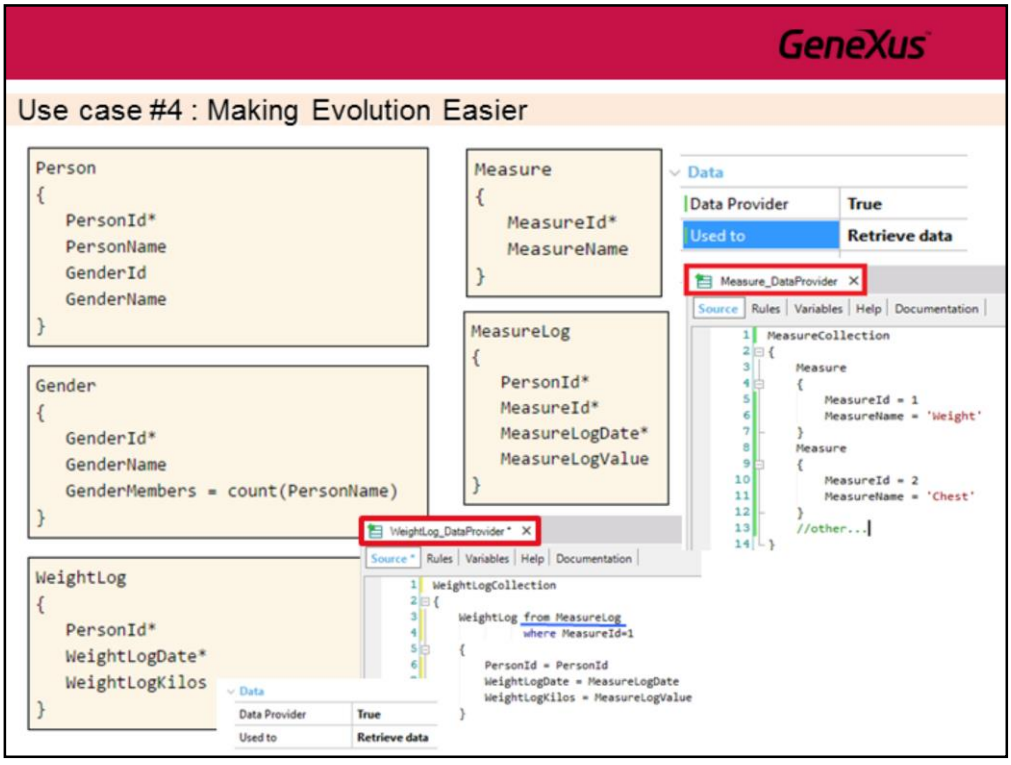
The objective of use case #4 is to show how defining dynamic transactions helps us to easily evolve.

We have a GeneXus Knowledge Base for tracking body weight, with the transactions shown on the slide.

- Person: Allows registering people and for each of them, their gender.
- Gender: Allows registering genders and has a defined formula to know how many people are registered in each gender.
- WeightLog: Allows registering, for each person, in each data, his/her weight.

Now suppose that, with the system already up and running, the developer is asked to track not only the weights but also other body measurements (like chest or waist circumference).

The database model needs to be redesigned in order to store this new data. Of course, it's possible to create a new Transaction object for each new measurement to be tracked, but a better (and more extensible) design is to have just one Transaction for any kind of measurements, as the following slide shows.



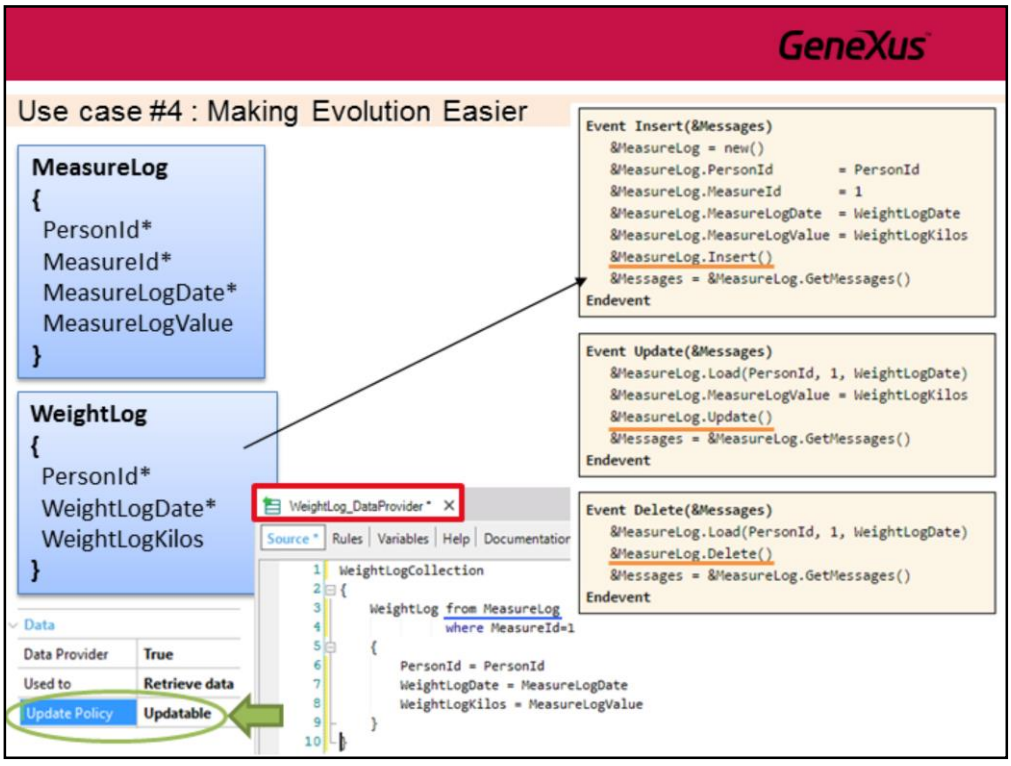
As you can see, we have defined two new transactions.

- Measure: Allows registering measurements (for example, “Weight”, “Chest”, etc.) appropriately.
- MeasureLog: Allows registering, for each person, on each data, for a specific measurement, the value measured.

Note that the Measure transaction has been defined as dynamic (take a look at its properties). You should also note that in the associated Data Provider, the data assigned to the Measure attributes is not obtained from tables, but rather it’s fixed values. This is because the developer does not want users to edit, delete, or enter new measurements. The developer has assigned specific values. A physical table associated with the Measure transaction will not be created, and note that the MeasureId attribute is used as a foreign key in the MeasureLog transaction.

The WeightLog transaction is not needed anymore since all measurements will be stored in the physical table associated with the new MeasureLog transaction. However, the application code still references it as base transaction in many places, such as For Each statements. So, instead of removing the WeightLog transaction and having to modify wherever it is referenced, it's a good idea to change it into a dynamic Transaction.

The developer must not forget that if he defines a transaction as Dynamic, the associated physical tables will no longer exist. So, before proceeding with this proposal, he/she has to move the data (in this case, weights from WeightLog to MeassureLog table).



By default, the form of dynamic Transactions shows data with read-only behavior.

Now, let's suppose that users are used to executing the WeightLog Transaction, and they ask us to still be able to edit it through its form. They also use the MeasureLog transaction, but they want to use both.

It's possible to complete these request, since transactions offer another property under the "Data" group. Its name is "Update Policy" and its possible values are: Read Only / Updatable.

So, by setting the WeightLog Transaction "Update Policy" property = Updatable, its form will allow the users to edit the data; **but in which physical table will the updates be stored?**

The developer has to codify the Insert, Update and Delete events in the WeightLog Transaction Events section, in order to specify his intention. In this example, the logical solution is to store the data in the MeasureLog physical table, using the Business Component concept as shown.

Note that after applying the Insert(), Update() and Delete() methods respectively to the &MeasureLog business component variable, the developer obtains the messages and/or errors triggered (in the &Messages collection variable). By declaring the &Messages variable as a parameter in each event (as shown), those messages are displayed in the WeightLog Dynamic Transaction in a transparent way, like its own messages.

In this way, the WeightLog Dynamic Transaction can be used in **exactly the same way as**

before and no changes are necessary to dependent programs. This also applies if the transaction is used as a Business Component, because it is a Dynamic Transaction that allows updates and the corresponding events to store the data are codified.

ADVANTAGES

Describe realities and intentions flexibly

Systems evolve more easily

Simplify developing

Easier for new developers