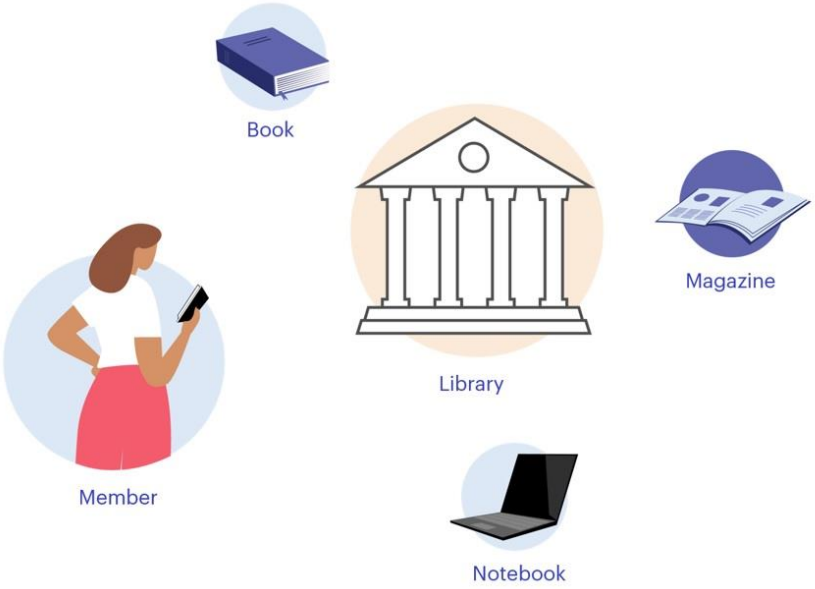
Analysis of the Transaction Design Model

Library

GeneXus™

Library



Throughout the previous course, we looked at everything we needed to correctly model a given reality in GeneXus. In this video, through the analysis of a limited reality, different options are analyzed to design transactions, using a set of essential resources and solving a series of real requirements.

Let's suppose that a GeneXus application needs to be designed for managing the tasks of a library related to the loans of books, magazines, and notebooks.
It works with Members who can access different reading material, such as books and magazines, as well as computer equipment loaned by the Library.

Entities

Country

- Id
- Name

Member

- Document
- Name
- Address
- Image
- Phone
- Over 20 years old

Author

- Id
- Name
- Image
- Country
- May have several books published

The information to be recorded is as follows:

Country
Each Country is registered with a unique identifier and its name.

Member
Each Library Member is registered with his/her ID card, name, address, photo and a contact phone number.

Library members must be 20 years of age or older.

Author
Each Author is registered with a unique identifier, name, photo and country of origin. An Author can have several books published.

Entities

Country

- Id
- Name

Book

- Id
- Title
- Edition Date
- Copies
- Literary Genre
- Author
- Publishing House in charge of its publication

Member

- Document
- Name
- Address
- Image
- Phone
- Over 20 years old

Magazine

- Id
- Title
- Publication Date
- Image
- Copies

Author

- Id
- Name
- Image
- Country
- May have several books published

Notebook

- Id
- Image
- Description
- Status

Book
Each book is registered with a unique identifier, title, date of publication and number of copies available in the Library. It can be a Novel, an Essay, a Poetry book, etc.; therefore, a book belongs to a literary genre.

In addition, a Book has an Author and a Publishing House in charge of its publication.

Magazine
Each magazine is registered with a unique identifier, title, date of publication, cover image and the number of copies available.

Notebook
The Library's services include loans of notebooks to its Members, since many of them are writers and researchers. Each notebook is registered with a unique identifier, its image, and short description.

In addition, its status (available or checked out) is recorded.

Entities

Country

- Id
- Name

Book

- Id
- Title
- Edition Date
- Copies
- Literary Genre
- Author
- Publishing House in charge of its publication

Loan

- Id
- Date
- Member
- Return Date
- 15-day deadline
- may include a maximum of 3 books, 4 magazines, and may or may not include borrowing a laptop
- Comment

Member

- Document
- Name
- Address
- Image
- Phone
- Over 20 years old

Magazine

- Id
- Title
- Publication Date
- Image
- Copies

Book Request

- Id
- Date
- Publishing House
- Book
- Number of copies

Author

- Id
- Name
- Image
- Country
- May have several books published

Notebook

- Id
- Image
- Description
- Status

Loan
The Library loans books, magazines, and notebooks to its Members.

Each loan is recorded with a unique identifier, the date of checkout, Member, and due date, which must be automatically determined.
All loans are for 15 days; they may include a maximum of 3 books, 4 magazines, and may or may not include the loan of a notebook.

Only one copy of each publication may be checked out, and it is possible to enter any comments considered necessary at the level of each copy's loan (e.g., damaged cover, missing pages, etc.).

In addition, the date on which a new loan is recorded is always the current date and it must not be possible to change it.

Request for copies (BookRequest)
It often happens that certain books are in high demand, and the Library decides to request more copies.

To this end, it makes a Request to the corresponding Publishing House. The system must check that book copies are requested from the indicated Publishing House.

Domains

Country

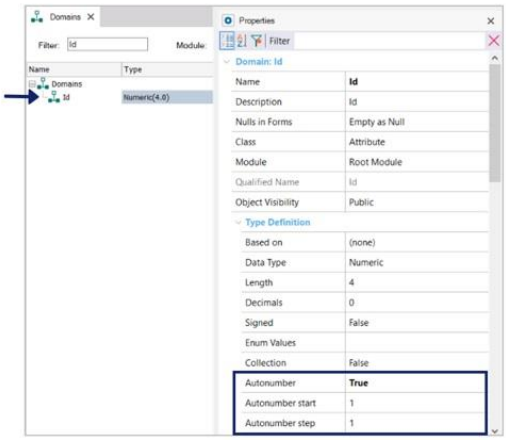
- Id
- Name

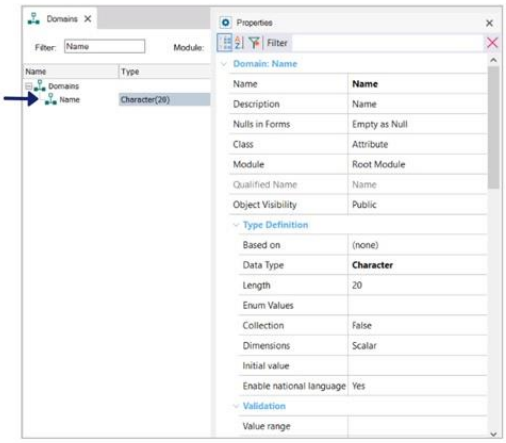
Member

- Document
- Name
- Address
- Image
- Phone
- Over 20 years old

Author

- Id
- Name
- Image
- Country
- May have several books published





Let´s start to analyze this reality.

Initially, we can clearly distinguish certain simple entities that we can begin to define, such as, for example, Country, Member and Author.

But first, we should take into account that most of the entities are registered with an identifier that can be autonumbered, except for the Member Number, who as we mentioned before, is registered with his or her identity document. Therefore, we define the Domain ID, with the Autonumber property set to True.

We also define the Name domain, as a character of 20.

Transaction definition: Country

- Id
- Name

Name	Type	Description
Country	Country	Country
CountryId	Id	Country Id
CountryName	Name	Country Name

Attribute	Order
Country Indexes	
ICountry	Primary Key
CountryId	Ascending
UCountry	Unique
CountryName	Ascending

Let's start by defining the Country transaction, with **CountryId** as primary key, and **CountryName** as secondary attribute.
CountryId is based on the ID domain, and to check that the name is not repeated, we define the corresponding unique index. This same definition is valid for all entities where it is necessary to check that the name is not repeated.

Transaction definition: Country

- Id
- Name

Weak 1-N

Name	Type	Description
Country	Country	Country
CountryId	Id	Country Id
CountryName	Name	Country Name
City	City	City
CityId	Id	City Id
CityName	Name	City Name

Attribute	Order
Country Indexes	
ICountry	Primary Key
CountryId	Ascending
UCountry	Unique
CountryName	Ascending

Name	Type
CountryCity Structure	
CountryId	Id
CityId	Id
CityName	Name

If it were necessary to record the cities of each country, because we need to know the city of birth of the author, we could model it as a weak entity in relation to the country, since a city does not exist outside that context.

So a second level would be added to the Country transaction, but since City is considered a weak entity it will not exist as a transaction in itself.

This means that CityId will not exist as a primary key in any table, and therefore, in order to know, for example, the city of birth of an author, the pair made up of the attributes CountryId, CityId, which will be the primary key of the COUNTRYCITY table associated with the second level of the Country transaction, will be needed.

However, since the description of the reality we are analyzing does not include the city concept, we will model the Country as a simple transaction.

Transaction definition: Member

- Document
- Name
- Address
- Image
- Phone
- Over 20 years old

Name	Type	Description	Formula
Member	Member	Member	
MemberDocument	Numeric(8.0)	Member Document	
MemberName	Name	Member Name	
MemberImage	Image	Member Image	
MemberAddress	Address, GeneXus	Member Address	
MemberPhone	Phone, GeneXus	Member Phone	
MemberBirthDate	Date	Member Birth Date	
MemberAge	Numeric(3.0)	Member Age	age(MemberBirthDate, today())

Member

Structure | Web Layout | Win Form | **Rules** | Events | Variables | Help | Documentation

```
1 Error("The member must be over 20 years old")
2     if MemberAge <= 20;
```

Let's move on to Member. For this we define the Member transaction, with the following attributes:

- MemberDocument, which corresponds to the ID card and therefore is not autonumbered, so we define it as an 8-digit numeric value.
- MemberName, of Name type.
- MemberImage, of Image type.
- MemberAddress, based on the Address semantic domain.
- and MemberPhone, of Phone type.

But, in addition, reality tells us that members must be over 20 years of age, so we need their date of birth and age, which should be calculated automatically.
So, to the transaction structure we add these attributes:

- MemberBirthDate, of Date type
- and MemberAge, a numeric value of 3 digits.

How do we calculate the Member's age? With the Age function, which calculates the age from the date of birth and the current date.
So, we define MemberAge as a calculated attribute that obtains its value from the following expression:
Age(MemberBirthDate, Today())

Why do we use the Today() function instead of the &today variable?
Because it's not possible to use variables in the formula declaration.

To check that every member is older than 20, it is enough to declare the following Error rule:

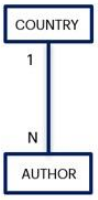
Error("The member must be over 20 years old") if MemberAge <= 20;

It should be noted that we are not considering in this analysis the declaration of rules for controlling basic data entry, such as, for example, controlling that a member is not entered without a name, etc.

Transaction definition: Author

- Id
- Name
- Image
- Country
- May have several books published

Name	Type	Description
Author	Author	Author
AuthorId	Id	Author Id
AuthorName	Name	Author Name
AuthorImage	Image	Author Image
CountryId	Id	Country Id
CountryName	Name	Country Name



Now let's consider the Authors. We need to create the Author transaction, with the following attributes:

- AuthorId
- AuthorName
- AuthorImage

Also, we know that the country of the Author must be registered, since every Author has a country of birth; therefore, we add CountryId and CountryName, where CountryId is a foreign key and CountryName is an attribute that is inferred from that foreign key.
In this way, we represent a 1-N relationship between Country and Author.

Transaction definition: Book

- Id
- Title
- Edition Date
- Copies
- Literary Genre
- Author
- Publishing House in charge of its publication

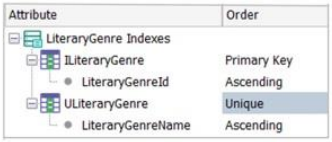
Now let's think of the concept of Book. It is a strong entity that is also identified with an autonumbered value, a title, a date of publication, an author, and the number of copies purchased by the Library.

But, in addition, reality tells us that a book belongs to a literary genre, since it can be a Novel, an Essay, etc.

So, how do we model the concept of literary genre? If we think that these genres are finite, a first option we can consider is the creation of an enumerated domain.

Transaction definition: Book

- Id
- Title
- Edition Date
- Copies
- Literary Genre
- Author
- Publishing House in charge of its publication



But the categorization of literary genres has been changing over time and is likely to continue to do so, so the enumerated domain doesn't seem to be the best option, as changes would have to be made manually and it would not be scalable.

We can consider a LiteraryGenre transaction with these attributes:
- LiteraryGenreId
- and LiteraryGenreName.

It is a good decision to define a unique index on the name of the genre in order to avoid the registration of literary genres with the same name.

But the Book also requires registering the Author and the Publishing House responsible for its publication. The Author is already defined as an entity, but the Publishing House is not. And although it is not explicitly declared, in the analysis the need to model the Publishing House entity arises.

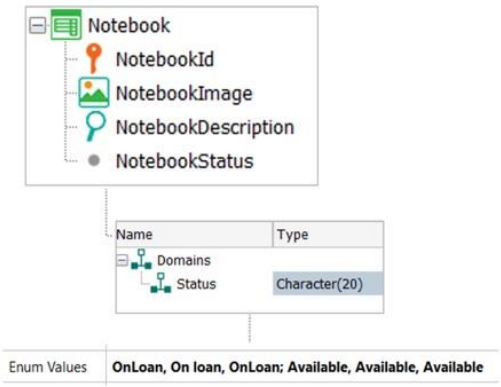
So, we define the PublishingHouse transaction with these attributes:
- PublishingHouseId
- and PublishingHouseName

We can also create the corresponding unique index on the name of the publisher.

Transaction definition: Magazine and Notebook

- Id
- Title
- Publication Date
- Image
- Copies

- Id
- Image
- Description
- Status

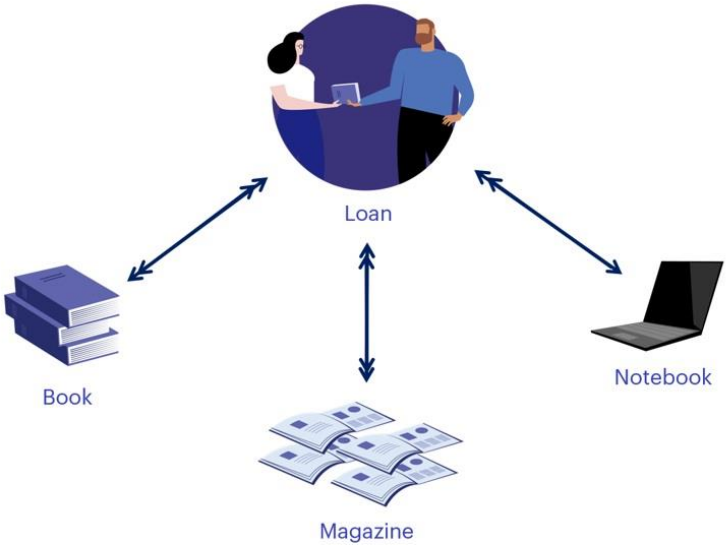


The Library also offers Magazines and Notebooks to its members, so we define the Magazine transaction, with the attributes:
- MagazineId
- MagazineTitle
- MagazineImage
- MagazinePublicationDate
- and MagazineCopies to record the number of copies purchased.

In addition, there is the Notebook transaction with these attributes:
- NotebookId
- NotebookImage
- NotebookDescription of LongVarChar type
- and NotebookStatus.

A notebook is “Available” or “Checked out”, so we define the Status enumerated domain with these values.

Transaction definition: Loan



Now let's analyze the concept of Loan, which is essential in this reality.

The Library allows a member to borrow a notebook, 3 books and 4 magazines for 15 days.

We then have a 1-N relationship between the entities Loan and Notebook, and an N-N relationship between Loan and Book, and between Loan and Magazine.

How can we model it?

Transaction definition: Loan

Name	Type	Formula	Nullable
Loan	Loan		
LoanId	Id		No
LoanDate	Date		No
MemberDocument	Numeric(8.0)		No
MemberName	Name		
MemberAddress	Address, GeneXus		
LoanReturnDate	Date	LoanDate.adddays(15)	
NotebookId	Id		Yes
NotebookDescription	Character(40)		

Let's analyze a first option:

We create the Loan transaction and define the following attributes:
- LoanId
- LoanDate
- MemberDocument
- MemberName
- MemberAddress
- LoanReturnDate

The return date is requested to be calculated automatically. A loan is made for 15 days, so we declare the following calculation associated with the attribute LoanReturnDate:
LoanDate.AddDays(15)

In addition, a loan may or may not include a notebook, so we add:
- NotebookId
- and NotebookDescription.

But in this Loan transaction, NotebookId is a non-mandatory foreign key; therefore, we set its Nullable property to Yes.

Transaction definition: Loan

Name	Type	Formula	Nullable
Loan	Loan		
LoanId	Id		No
LoanDate	Date		No
MemberDocument	Numeric(8.0)		No
MemberName	Name		
MemberAddress	Address, GeneXus		
LoanReturnDate	Date	LoanDate.adddays(15)	
NotebookId	Id		Yes
NotebookDescription	Character(40)		
LoanBooksQty	Numeric(4.0)	count(LoanBookComment)	
Book	Book		
BookId	Id		
BookTitle	Character(20)		
LoanBookComment	Character(40)		

```
Error("Only 3 books can be checked out")
     if LoanBooksQty > 3;
```

A loan includes several books, with a maximum of 3. So we define a second level to record them, including a LoanBookComment attribute, to record any comments needed at the time of the loan.

How do we control that no more than 3 books are entered?

We can define a new LoanBooksQty attribute that counts that amount and then condition that value in an Error rule:
Error(“Only 3 books can be checked out”) if LoanBooksQty > 3;

Transaction definition: Loan

Name	Type	Formula	Nullable
Loan	Loan		
LoanId	Id		No
LoanDate	Date		No
MemberDocument	Numeric(8.0)		No
MemberName	Name		
MemberAddress	Address, GeneXus		
LoanReturnDate	Date	LoanDate.adddays(15)	
NotebookId	Id		Yes
NotebookDescription	Character(40)		
LoanBooksQty	Numeric(4.0)	count(LoanBookComment)	
LoanMagazinesQty	Numeric(4.0)	count(LoanMagazineComment)	
Book	Book		
BookId	Id		
BookTitle	Character(20)		
LoanBookComment	Character(40)		
Magazine	Magazine		
MagazineId	Id		
MagazineTitle	Character(20)		
LoanMagazineComment	Character(40)		

```
Error("Only 3 books can be checked out")
     if LoanBooksQty > 3;

Error("Only 4 magazines can be checked out")
     if LoanMagazinesQty > 4;
```

But the same loan can include up to 4 magazines, so we can define another second level, parallel to the book, to register the magazines. It is also possible to control that no more than 4 are checked out.

Transaction definition: Loan

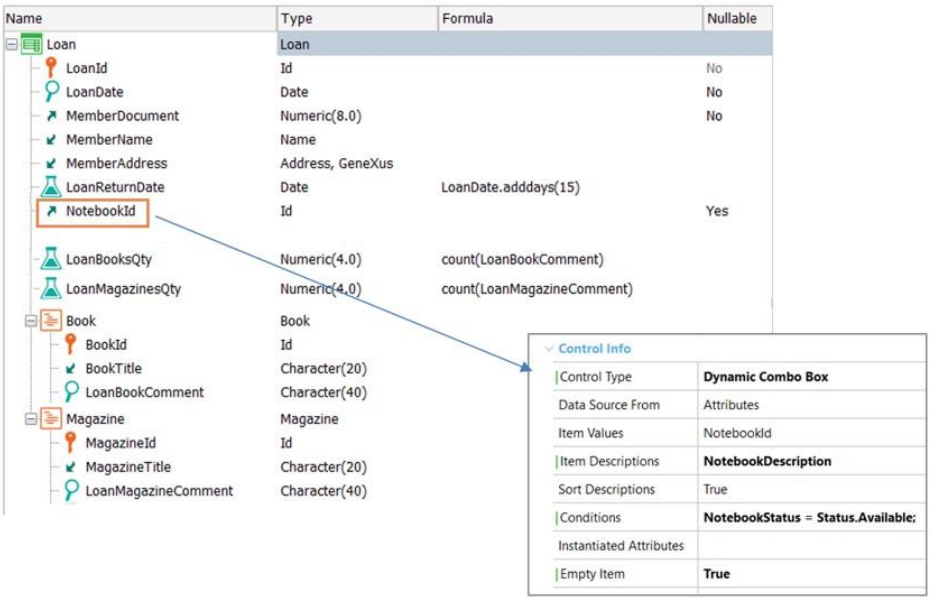
Name	Type	Formula	Nullable
Loan	Loan		
LoanId	Id		No
LoanDate	Date		No
MemberDocument	Numeric(8.0)		No
MemberName	Name		
MemberAddress	Address, GeneXus		
LoanReturnDate	Date	LoanDate.adddays(15)	
NotebookId	Id		Yes
NotebookDescription	Character(40)		
LoanBooksQty	Numeric(4.0)	count(LoanBookComment)	
LoanMagazinesQty	Numeric(4.0)	count(LoanMagazineComment)	
Book	Book		
BookId	Id		
BookTitle	Character(20)		
LoanBookComment	Character(40)		
Magazine	Magazine		
MagazineId	Id		
MagazineTitle	Character(20)		
LoanMagazineComment	Character(40)		

```
Error("Only 3 books can be checked out")
     if LoanBooksQty > 3;

Error("Only 4 magazines can be checked out")
     if LoanMagazinesQty > 4;

Default(LoanDate, today());

noaccept(LoanDate);
```

The reality clearly describes that the registration date of a loan must always be the current date, with no possibility of being modified.

To this end, we declare the following rules:
Default(LoanDate, today());
noaccept(LoanDate);

Transaction definition: Loan



Although the objective of this video is to focus on analyzing the transaction design, we are going to suggest an implementation option so that, when registering a loan, only notebooks that have Available status are offered. This implementation can be done from the definition of the transaction itself and that is why we include it.

We are going to remove the NotebookDescription attribute, because we will "disguise" the identifier of a notebook with its description. We select NotebookId and define it as a Dynamic Combo Box, with NotebookId in the property Item Values, and NotebookDescription in the property Item Descriptions.

This will load all the notebooks registered. So that only those with available status are offered, we declare the following condition:
NotebookStatus = Status.Available;

Also, we must configure the Empty Item property to True, since NotebookId may not be chosen because it can be null.

Although this is not the only way to control that a notebook is available when registering a loan, it is an implementation that can be solved from the definition of the transaction itself.

Available quantities of Books and Magazines

Name	Type	Formula
Book	Book	
BookId	Id	
BookTitle	Character(20)	
BookEditionDate	Date	
AuthorId	Id	
AuthorName	Name	
BookCopies	Numeric(4.0)	
LiteraryGenreId	Id	
LiteraryGenreName	Name	
PublishingHouseId	Id	
PublishingHouseName	Name	
BookOnLoanQty	Numeric(4.0)	count(LoanBookComment)

Name	Type	Formula
Magazine	Magazine	
MagazineId	Id	
MagazineTitle	Character(20)	
MagazineImage	Image	
MagazinePubicationDate	Date	
MagazineCopies	Numeric(4.0)	

Another necessary requirement is to know the availability of books or magazines in high demand.

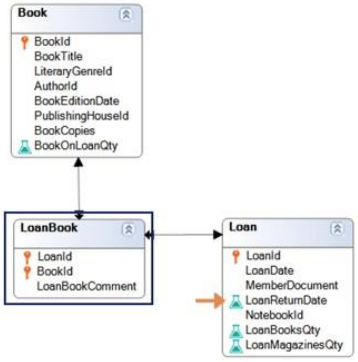
We know the number of copies purchased by the Library for each book and magazine, but we don't know how many are still available.

Can we obtain these values in a simple way and based on the transaction design?
If in the Book transaction we define a new attribute, BookOnLoanQty, we can calculate the number of copies borrowed and therefore know the number of copies available.

What happens if we associate the Count(LoanBookComment) calculation with this new BookOnLoanQty attribute? Will it effectively sum the number of copies of that book that have been checked out?
The answer is YES, because the table associated with the transaction where we define the calculation is BOOK. And the table where the calculation is resolved is LOANBOOK, since it is determined by the LoanBookComment attribute.
Between both tables there is an attribute in common which is BookId, and therefore it is an implicit filter that GeneXus will apply; that is to say, that this calculation will return the total number of copies of that book that has been loaned.

Available quantities of Books and Magazines

Name	Type	Formula
Book	Book	
BookId	Id	
BookTitle	Character(20)	
BookEditionDate	Date	
AuthorId	Id	
AuthorName	Name	
BookCopies	Numeric(4.0)	
LiteraryGenreId	Id	
LiteraryGenreName	Name	
PublishingHouseId	Id	
PublishingHouseName	Name	
BookOnLoanQty	Numeric(4.0)	count(LoanBookComment,LoanReturnDate >= servernow())



Name	Type	Formula
Magazine	Magazine	
MagazineId	Id	
MagazineTitle	Character(20)	
MagazineImage	Image	
MagazinePubicationDate	Date	
MagazineCopies	Numeric(4.0)	

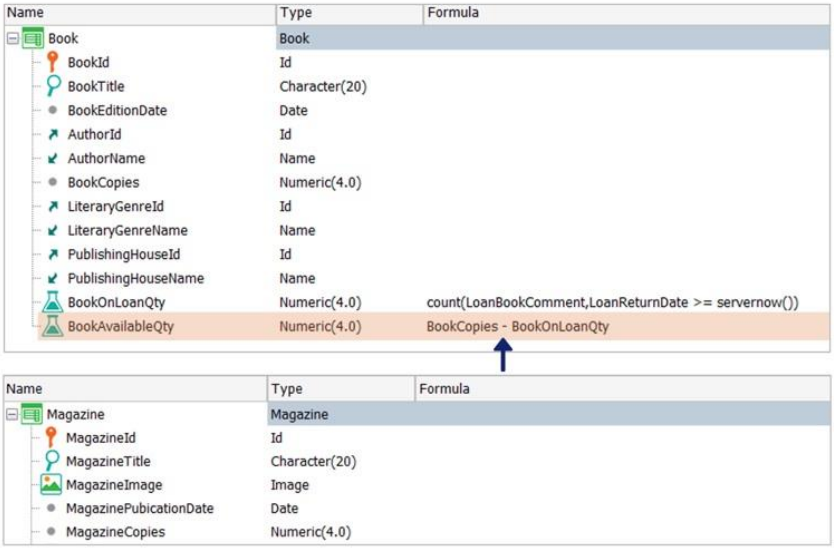
But we need to know the number of copies currently checked out, so we have to add a condition to the calculation that allows counting the number of copies currently checked out. For this purpose, we consider loans whose return date is greater than or equal to the current date.

We can state the following:
Count(LoanBookComment, LoanReturnDate >= servernow())

The servernow() function allows obtaining the server's current date and time. The Today() function could also be used.

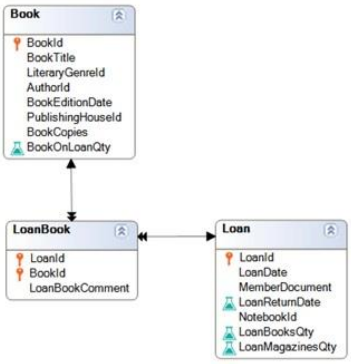
Can we declare this calculation condition? Yes, because the attribute involved (LoanReturnDate) belongs to the extended table of the table where the declared calculation is resolved, which is LOANBOOK.

Available quantities of Books and Magazines



Name	Type	Formula
Book	Book	
BookId	Id	
BookTitle	Character(20)	
BookEditionDate	Date	
AuthorId	Id	
AuthorName	Name	
BookCopies	Numeric(4.0)	
LiteraryGenreId	Id	
LiteraryGenreName	Name	
PublishingHouseId	Id	
PublishingHouseName	Name	
BookOnLoanQty	Numeric(4.0)	count(LoanBookComment,LoanReturnDate >= servernow())
BookAvailableQty	Numeric(4.0)	BookCopies - BookOnLoanQty

Name	Type	Formula
Magazine	Magazine	
MagazineId	Id	
MagazineTitle	Character(20)	
MagazineImage	Image	
MagazinePubicationDate	Date	
MagazineCopies	Numeric(4.0)	



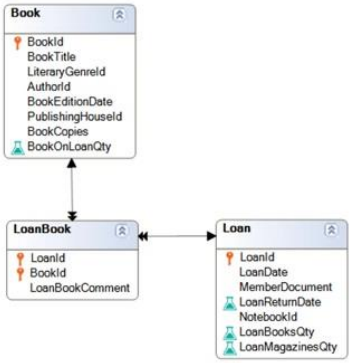
Therefore, if we know the total number of copies, and the number of copies currently checked out, then we can know the number of copies currently available.

So the structure of the Book transaction looks as follows:

Available quantities of Books and Magazines

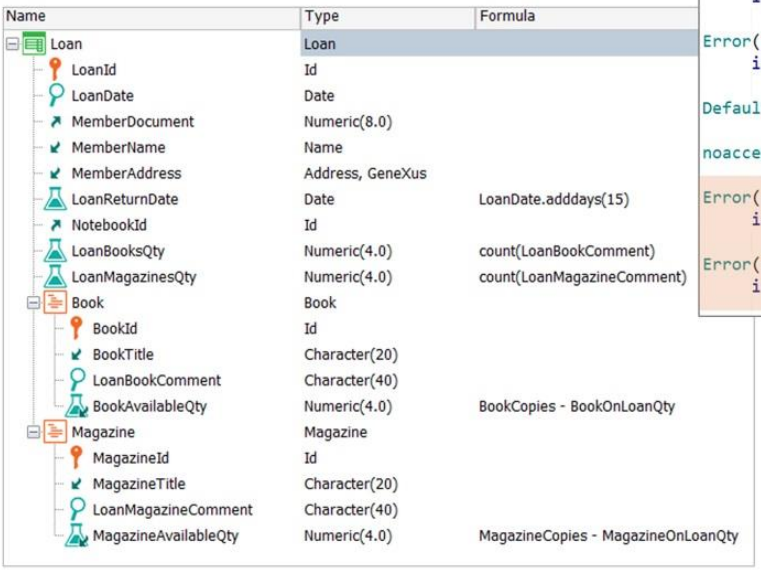
Name	Type	Formula
Book	Book	
BookId	Id	
BookTitle	Character(20)	
BookEditionDate	Date	
AuthorId	Id	
AuthorName	Name	
BookCopies	Numeric(4.0)	
LiteraryGenreId	Id	
LiteraryGenreName	Name	
PublishingHouseId	Id	
PublishingHouseName	Name	
BookOnLoanQty	Numeric(4.0)	count(LoanBookComment,LoanReturnDate >= servernow())
BookAvailableQty	Numeric(4.0)	BookCopies - BookOnLoanQty

Name	Type	Formula
Magazine	Magazine	
MagazineId	Id	
MagazineTitle	Character(20)	
MagazineImage	Image	
MagazinePubicationDate	Date	
MagazineCopies	Numeric(4.0)	
MagazineOnLoanQty	Numeric(4.0)	count(LoanMagazineComment, LoanReturnDate >= servernow())
MagazineAvailableQty	Numeric(4.0)	MagazineCopies - MagazineOnLoanQty



In the same way, we can know the number of available copies of a given magazine:

Transaction definition: Loan



Name	Type	Formula
Loan	Loan	
LoanId	Id	
LoanDate	Date	
MemberDocument	Numeric(8.0)	
MemberName	Name	
MemberAddress	Address, GeneXus	
LoanReturnDate	Date	LoanDate.adddays(15)
NotebookId	Id	
LoanBooksQty	Numeric(4.0)	count(LoanBookComment)
LoanMagazinesQty	Numeric(4.0)	count(LoanMagazineComment)
Book	Book	
BookId	Id	
BookTitle	Character(20)	
LoanBookComment	Character(40)	
BookAvailableQty	Numeric(4.0)	BookCopies - BookOnLoanQty
Magazine	Magazine	
MagazineId	Id	
MagazineTitle	Character(20)	
LoanMagazineComment	Character(40)	
MagazineAvailableQty	Numeric(4.0)	MagazineCopies - MagazineOnLoanQty

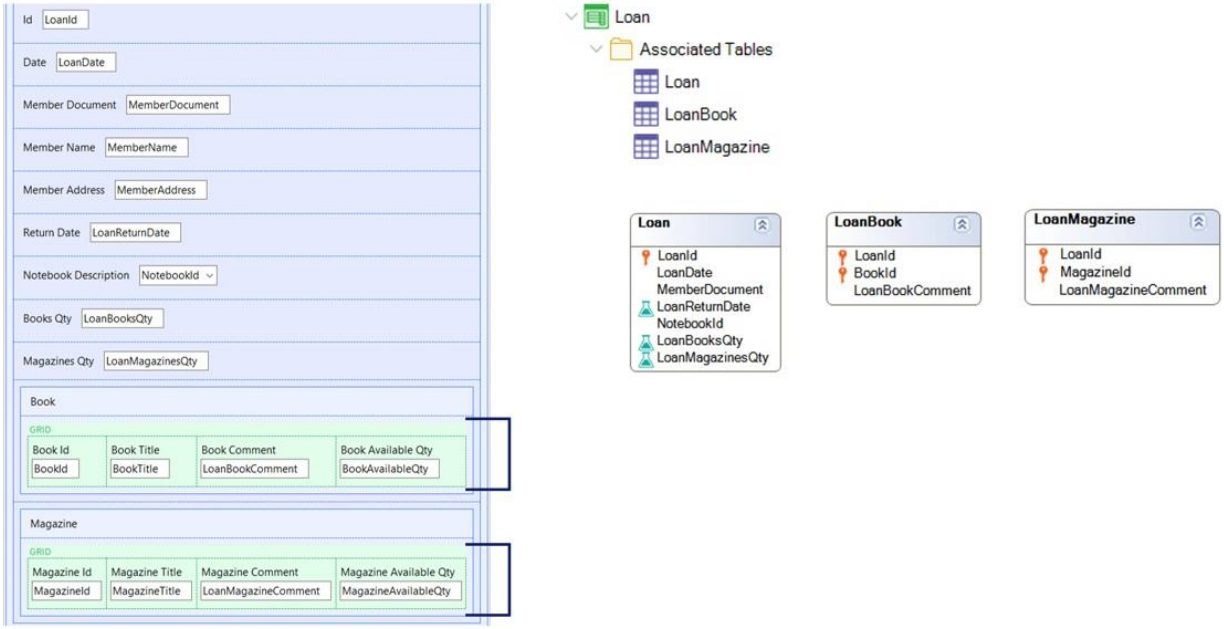
```
Error("Only 3 books can be checked out")
     if LoanBooksQty > 3;

Error("Only 4 magazines can be checked out")
     if LoanMagazinesQty > 4;

Default(LoanDate, today());

noaccept(LoanDate);

Error("There are no available copies of this book")
     if BookAvailableQty < 0;

Error("There are no available copies of this magazine")
     if MagazineAvailableQty < 0;
```

Now we have all the information we need to validate a loan, so let's add the necessary attributes to validate the availability of the copies to be borrowed:

- BookAvailableQty, at the book level, and
- MagazineAvailableQty, at the magazine level.

And we control with the following Error rules:

Loan transaction: Form and tables structure



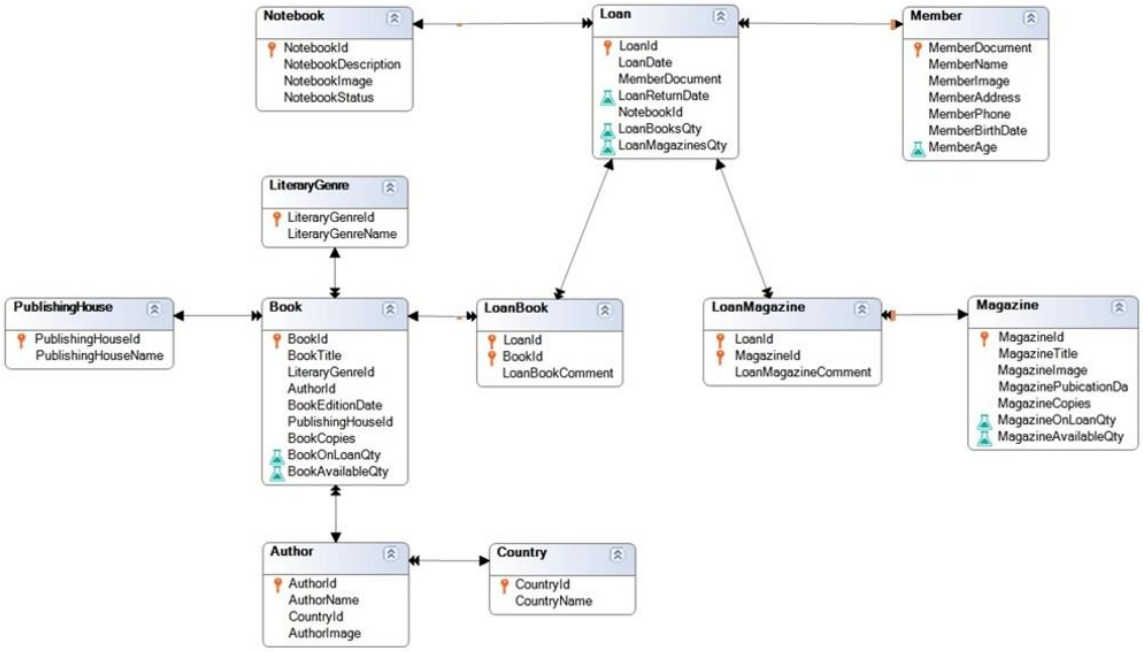
So far we have met the requirements to register loans. But let's look at the form generated for the design we have made.

What associated tables are created by GeneXus for this Loan transaction?

It creates 3 tables:
- LOAN, associated with the first level, with LoanId as PK.
- **LOANBOOK**, associated with the book level, with LoanId and BookId as compound PK.
- **LOANMAGAZINE**, associated with the magazine level, with LoanId and MagazineId as compound PK.

By defining two parallel levels, we see parallel grids in the form. This may be cumbersome for the end user, who may request to simplify the design, and manage the information in different, simpler screens.

Table diagram



For the current design, GeneXus has created the following tables.

Summary

Analyze, evaluate the reality to be modeled and implement the option that we consider correct, always working together with the end user

Remember that we must always analyze, evaluate the reality to be modeled and implement the option we consider correct, and for this it is essential to always work together with the end user, who will guide us in selecting the design.

Let's not forget that the application must be a support tool for end users to better manage and develop their business.

GeneXus™

training.genexus.com
wiki.genexus.com