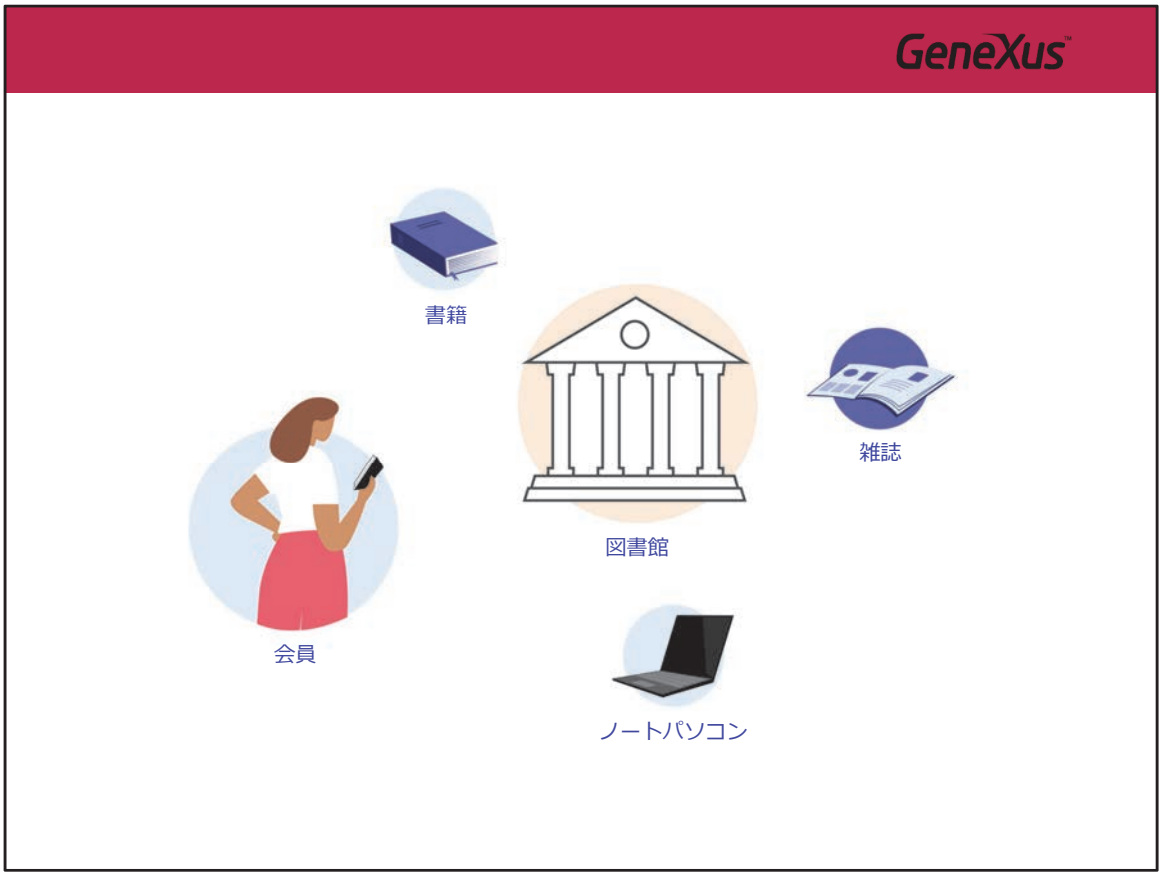
トランザクション設計モデルの分析

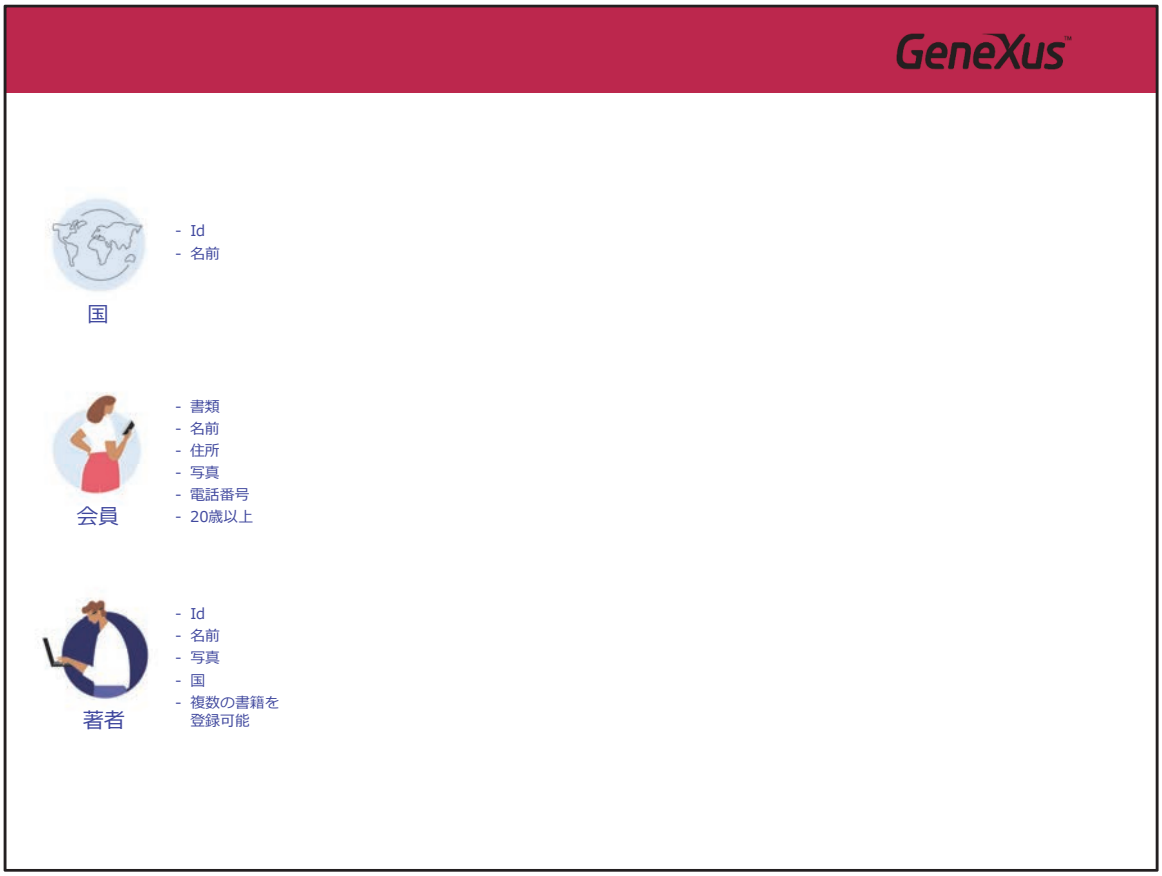
図書館向けアプリの場合

GeneXus™



Basic コースでは、特定の現実を GeneXus で適切にモデリングするために必要なことを確認しました。この章では、特定の現実を分析し、それに基づいてトランザクションを設計する方法を見ていきます。その際に使用する必要のある一連のリソースと、現実の要件への対応についても確認します。

図書館での書籍、雑誌、ノートパソコンの貸し出しに関するタスクを管理する GeneXus アプリケーションを設計するとします。
このアプリケーションの利用者は、図書館で貸し出される書籍や雑誌、コンピューター機器などを利用する会員です。



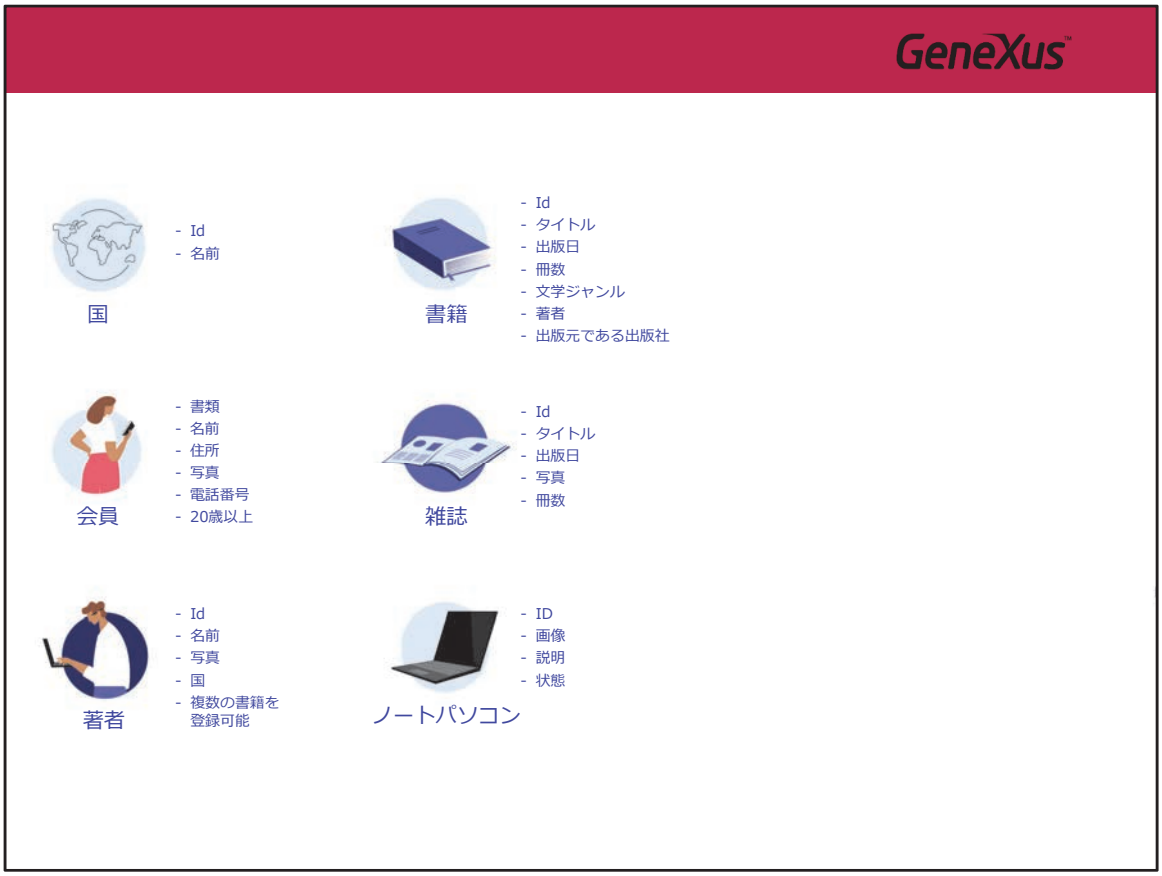
記録する情報には、次のものがあります。

国
国の情報として、一意の識別子と名前を登録します。

会員
図書館の会員の情報として、書類に記載された 8 桁の数値、名前、住所、写真、連絡先電話番号を登録します。

図書館の会員は 20 歳を超えている必要があります。

著者
著者の情報として、一意の識別子、名前、写真、出身国を登録します。1 人の著者に対して、複数の書籍を登録できます。



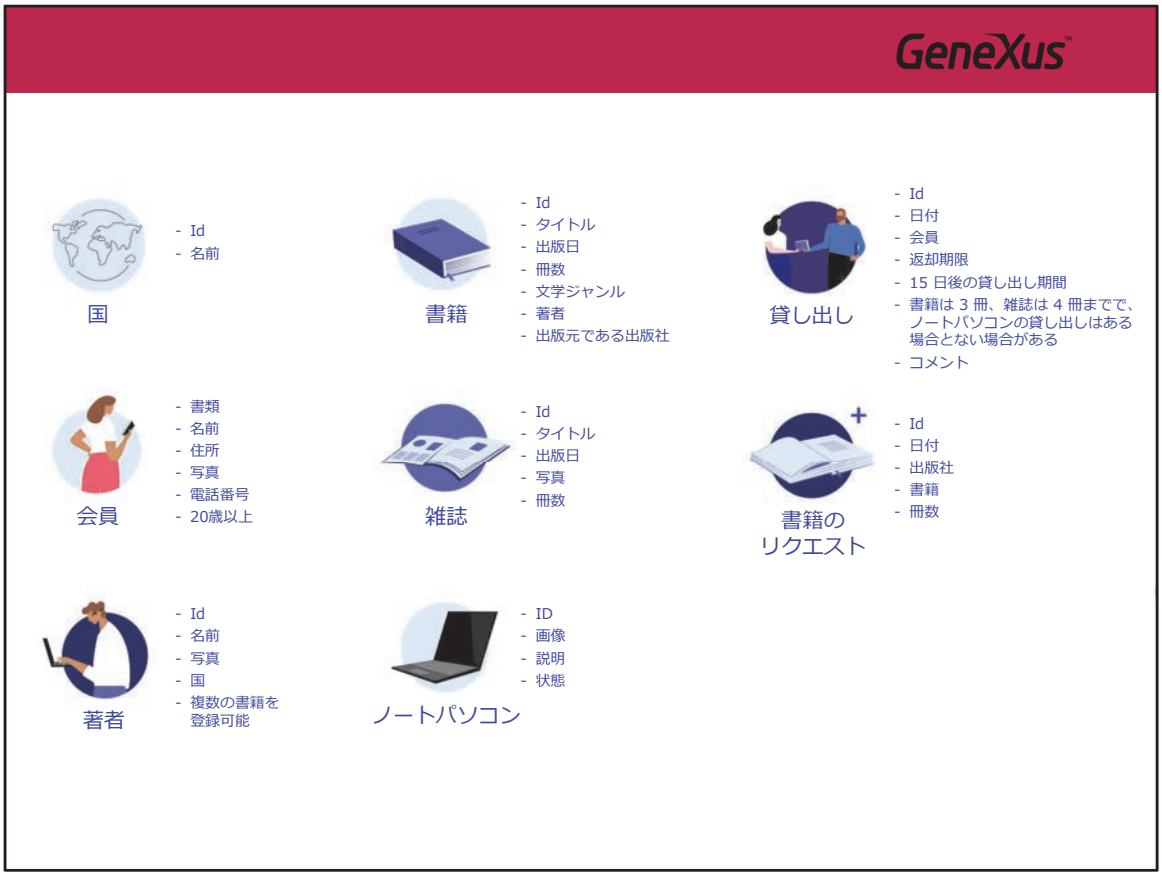
書籍
書籍の情報として、一意の識別子、タイトル、出版日、図書館で利用可能な冊数を登録します。各書籍は 1 つの文学ジャンルに属します (小説、エッセー、詩集など)。

さらに、書籍には、著者および出版元である出版社の情報を登録します。

雑誌
雑誌の情報として、一意の識別子、タイトル、出版日、表紙の画像、利用可能な冊数を登録します。

ノートパソコン
会員には著作家や研究者も多いため、ノートパソコンの貸し出しサービスも行っています。ノートパソコンの情報として、一意の識別子、画像、簡単な説明を登録します。

さらに、ステータス (利用可能または貸し出し中) も記録します。



貸し出し
図書館は、会員に書籍、雑誌、ノートパソコンを貸し出します。

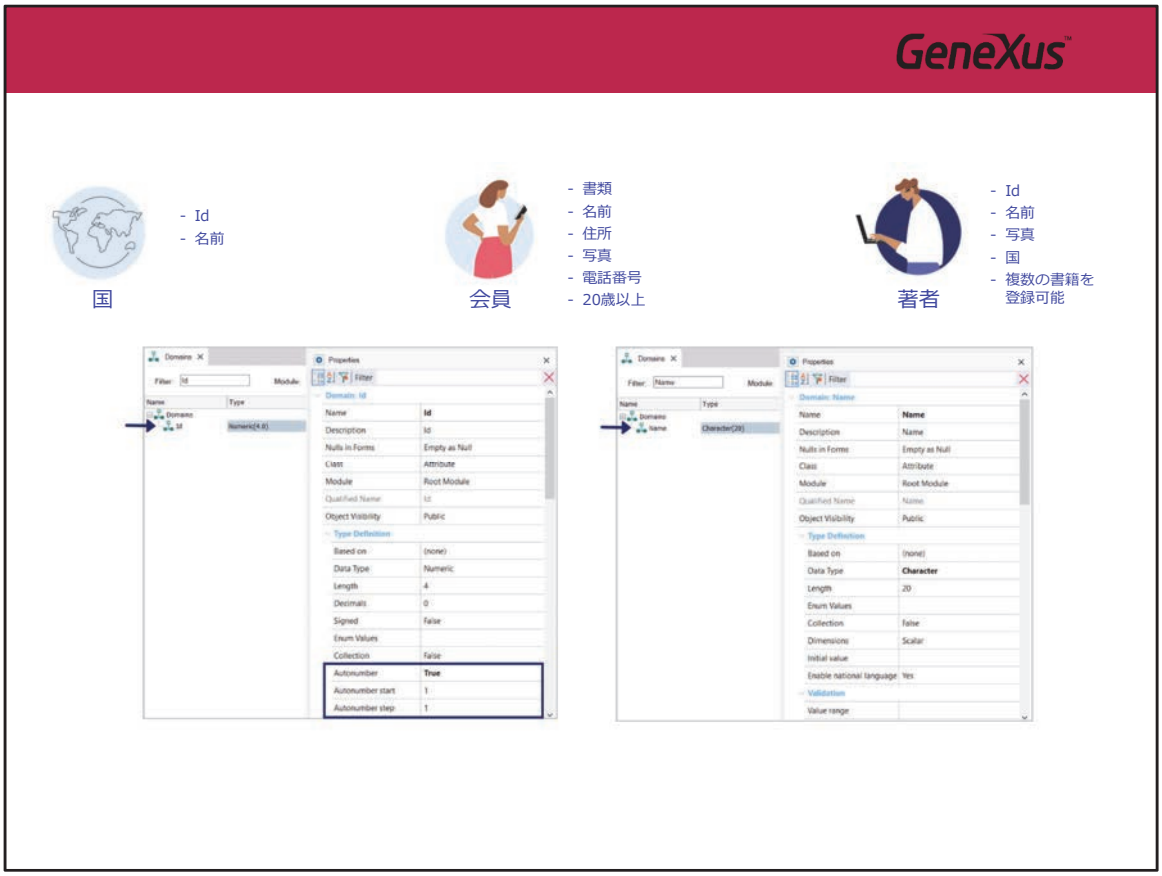
貸し出しの情報として、一意の識別子、貸し出し日、会員、返却期限 (自動的に決まる) を記録します。
貸し出し期間は一律 15 日間です。1 回の貸し出しにつき、書籍は 3 冊、雑誌は 4 冊までで、ノートパソコンの貸し出しはある場合とない場合があります。

同じ出版物は、一度に 1 冊のみ貸し出すことができます。貸し出す出版物ごとに、必要に応じて任意のコメントを入力できます (「表紙に破損あり」、「ページが欠けている」など)。

さらに、新規貸し出しの記録日は、必ず貸し出し当日の日付になります。これは変更することはできません。

書籍のリクエスト (BookRequest)
特定の書籍に対する需要が高い場合、図書館はその書籍を追加でリクエストすることがあります。

その場合、その書籍の出版社にリクエストします。システムは、適切な出版社に書籍のリクエストが行われたことを確認する必要があります。



この現実を分析しましょう。

最初にできることは、国、会員、著者などのシンプルなエンティティを区別して定義することです。

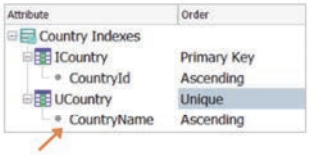
しかし、その前に、ほとんどのエンティティに自動採番可能な識別子が付与されることを考慮する必要があります。ただし、会員番号は例外です。会員番号は、先ほど言及したように、本人の身分証明書を使用して登録されるためです。
そのため、ID ドメインを定義し、[Autonumber] プロパティを True に設定します。

また、Name ドメインを 20 文字の文字列として定義します。

トランザクションの定義: Country

- Id
- 名前

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



まず Country トランザクションを定義します。**CountryId** を主キー、**CountryName** を従属項目属性にします。
CountryId は ID ドメインに基づいています。また、名前が重複していないことを確認するために、対応する一意のインデックスを定義します。名前が重複していないことを確認する必要があるすべてのエンティティに対して、これと同じ定義を使用します。

トランザクションの定義: Country

- Id
- 名前

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

著者が生まれた都市を確認するために各国の都市を記録する必要がある場合は、国との関係で弱いエンティティとしてモデリングします。これは、都市がそのコンテキスト以外に出現しないためです。

そのために Country トランザクションに第 2 レベルを追加します。ただし、都市は弱いエンティティなので、トランザクションとして生成されることはありません。

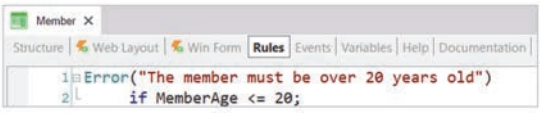
つまり、CityId はどのテーブルの主キーにもなりません。そのため、たとえば著者の生まれた都市を知りたい場合は、CountryId 項目属性と CityId 項目属性のペアを構成し、それを Country トランザクションの第 2 レベルに関連付けられた CountryCity Structure テーブルの主キーとする必要があります。

ただし、ここで分析する現実には都市の概念は含まれていないため、Country はシンプルなトランザクションとしてモデリングします。

トランザクションの定義: Member

- 書類
- 名前
- 住所
- 写真
- 電話番号
- 20歳以上

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())



次に、会員について見ていきます。次の項目属性を持つ Member トランザクションを定義します。

- **MemberDocument**。書類に対応するため、自動採番しない 8 桁の数値として定義します。
- **MemberName**。Name タイプにします。
- **MemberImage**。Image タイプにします。
- **MemberAddress**。Address セマンティックドメインに基づきます。
- **MemberPhone**。Phone タイプにします。

さらに、会員は 20 歳を超えている必要があるため、誕生日と年齢の情報が必要です。年齢は自動で計算されます。
そのため、トランザクション構造に次の項目属性を追加します。

- **MemberBirthDate**。Date タイプにします。
- **MemberAge**。3 桁の数値にします。

会員の年齢はどのように計算すればよいでしょうか。Age 関数を使用して、誕生日および現在の日付から年齢を計算します。
そこで、MemberAge は次のエクスプレッションで値を取得する、計算される項目属性として定義します。
Age(MemberBirthDate, Today())

&today 変数ではなく Today() 関数を使用するのはなぜでしょうか。
それは、式の宣言では変数を使用できないためです。

会員が 20 歳を超えていることを確認するには、次の Error ルールを宣言します。
Error(“The member must be over 20 years old”) if MemberAge <= 20;

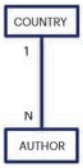
この分析では、基本的なデータ入力を制御するためのルールの宣言 (会員の名前が入力されていない場合の制御など) は考慮しません。

GeneXus

トランザクションの定義: Author

- Id
- 名前
- 写真
- 国
- 複数の書籍を登録可能

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



次に、著者について考えます。次の項目属性を持つ Author トランザクションを作成します。

- **AuthorId**
- **AuthorName**
- **AuthorImage**

著者の国情報も登録します。これは、どの著者にも生まれた国があるためです。
そのため、CountryId および CountryName を追加します。CountryId は外部キーで、CountryName はその外部キーから推論される項目属性です。
このようにして Country と Author の間の 1 対 N の関係を表します。

GeneXus

トランザクションの定義: Book

- Id
- タイトル
- 出版日
- 冊数
- 文学ジャンル
- 著者
- 出版元である出版社

次に、書籍の概念について考えます。これは、自動採番された値、タイトル、出版日、著者、図書館が購入した冊数で識別される、強いエンティティです。

また、書籍は、小説やエッセーなど、いずれかの文学ジャンルに属しています。

では、文学ジャンルはどのようにモデリングすればよいでしょうか。このようなジャンルが有限であるとすると、まず考えられる方法は、列挙型ドメインを作成することです。

トランザクションの定義: Book

- Id
- タイトル
- 出版日
- 冊数
- 文学ジャンル
- 著者
- 出版元である出版社

しかし、文学ジャンルの分け方は時間の経過とともに変化しており、今後も変わっていく可能性が高いものです。手動で変更する必要が出てくるため、拡張性のない列挙型ドメインは最適ではないと考えられます。

そこで、次の項目属性を持つ LiteraryGenre トランザクションを検討します。
- **LiteraryGenreId**
- **LiteraryGenreName**

文学ジャンルが重複して登録されないように、名前に一意のインデックスを定義することは良い判断です。

ただし、書籍については、著者および出版社も登録する必要があります。著者は既にエンティティとして定義されていますが、出版社はまだ定義されていません。明示的に宣言されてはいませんが、分析では、出版社のエンティティをモデリングすることも必要です。

そのため、次の項目属性を持つ PublishingHouse トランザクションを定義します。
- **PublishingHouseId**
- **PublishingHouseName**

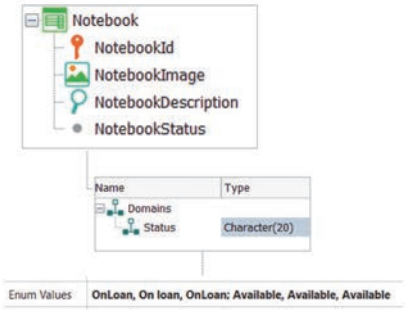
出版社の名前に対応する一意のインデックスを作成することもできます。

GeneXus

トランザクションの定義: Magazine および Notebook

- Id
- タイトル
- 出版日
- 写真
- 冊数

- ID
- 画像
- 説明
- 状態



図書館では、書籍のほかに雑誌やノートパソコンも貸し出しています。そのため、次の項目属性を持つ Magazine トランザクションを定義します。

- **MagazineId**
- **MagazineTitle**
- **MagazineImage**
- **MagazinePublicationDate**
- **MagazineCopies**。購入した冊数の記録用です。

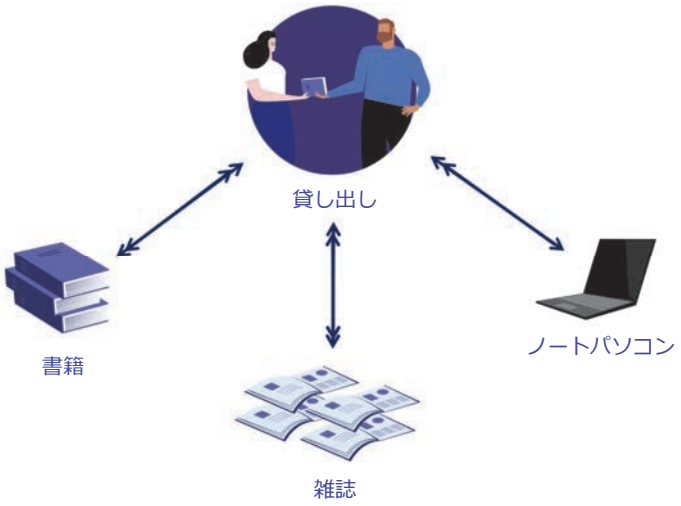
さらに、次の項目属性を持つ Notebook トランザクションを定義します。

- **NotebookId**
- **NotebookImage**
- **NotebookDescription**。LongVarChar タイプにします。
- **NotebookStatus**

ノートパソコンは「利用可能」か「貸し出し中」のどちらかであるため、これらの値を持つ Status 列挙型ドメインを定義します。

GeneXus

トランザクションの定義: Loan



では、この現実に不可欠な概念である貸し出しについて分析します。

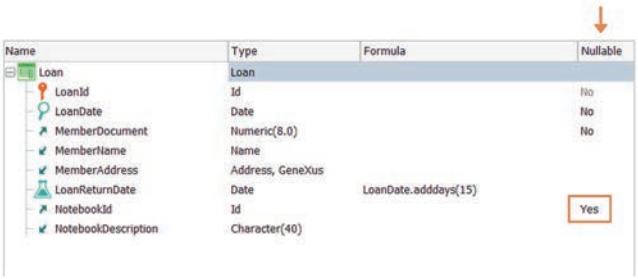
図書館の会員は、ノートパソコン 1 台、書籍 3 冊、雑誌 4 冊を 15 日間借りることができます。

エンティティである「貸し出し」と「ノートブック」の間には 1 対 N の関係があり、「貸し出し」と「書籍」、および「貸し出し」と「雑誌」の間には N 対 N の関係があります。

これをどのようにモデリングすればよいでしょうか。

GeneXus

トランザクションの定義: 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)		

1 つ目の方法を分析してみましょう。

Loan トランザクションを作成し、次の項目属性を定義します。
- **LoanId**
- **LoanDate**
- **MemberDocument**
- **MemberName**
- **MemberAddress**
- **LoanReturnDate**

返却日は自動計算されることが求められています。貸し出しは 15 日間であるため、LoanReturnDate 項目属性に関連する次の計算を宣言します。
LoanDate.AddDays(15)

さらに、貸し出しにはノートパソコンが含まれる場合と含まれない場合があるため、次の項目属性を追加します。
- **NotebookId**
- **NotebookDescription**

ただし、この Loan トランザクションでは、NotebookId は必須ではない外部キーであるため、[Nullable] プロパティを Yes に設定します。

GeneXus

トランザクションの定義: 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;
```

1 回の貸し出しには最大 3 冊の書籍が含まれます。これを記録するための第 2 レベルを定義します。貸し出し時に任意のコメントを記録するための LoanBookComment 項目属性も含めます。

書籍が 4 冊以上入力されないように制御するにはどうすればよいでしょうか。

冊数をカウントし、その値を Error ルールの条件に設定する新しい LoanBooksQty 項目属性を定義します。

Error(“Only 3 books can be checked out”) if LoanBooksQty > 3;

トランザクションの定義: 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;
```

一方、雑誌は 1 回の貸し出しに最大 4 冊含めることができます。そのため、書籍と並行して雑誌を登録するための別の第 2 レベルを定義します。これも、貸し出し数を最大 4 に制限するように制御できます。

トランザクションの定義: 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)		

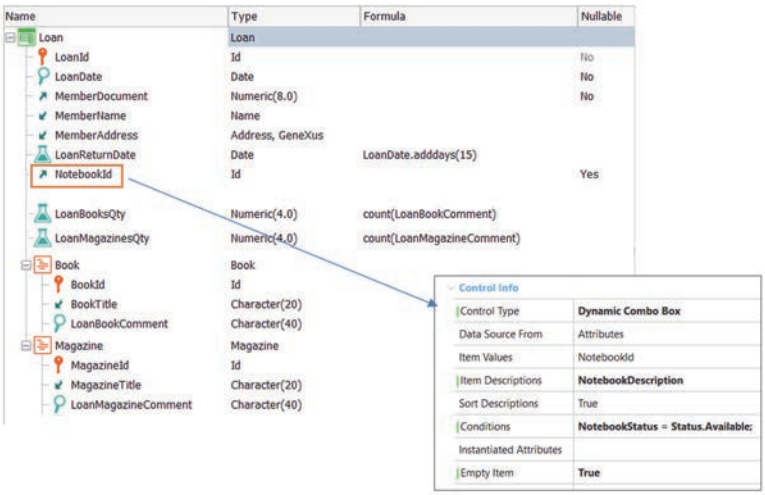
```
Error("Only 3 books can be checked out")
     if LoanBooksQty > 3;

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

Default(LoanDate, today());

noaccept(LoanDate);
```

現実では、貸し出しの登録日は常に現在の日付になり、変更されることはありません。

そのため、次のルールを宣言します。

Default(LoanDate, today());
noaccept(LoanDate);

トランザクションの定義: Loan



この章の目的は、トランザクションの設計の分析にフォーカスすることですが、貸し出しを登録する際にステータスが「利用可能」のノートパソコンのみを表示する実装も提案します。この実装はトランザクション自体の定義で行うことができます。この説明をここで行うのは、そのためです。

ノートパソコンの識別子をデスクリプションとして「偽装」するため、NotebookDescription 項目属性を削除します。NotebookId を選択し、**ダイナミック コンボ ボックス**として定義します。[**Item Values**] プロパティを NotebookId にして、[**Item Descriptions**] プロパティを NotebookDescription に設定します。

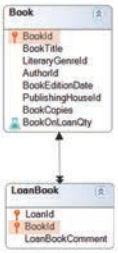
こうすると、登録されているすべてのノートパソコンがロードされます。ステータスが「利用可能」であるものだけが表示されるように、次の条件を宣言します。
NotebookStatus = Status.Available;

また、[**Empty Item**] プロパティを True に設定する必要があります。これは、NotebookId が選択されない場合、null として登録するためです。

貸し出しを登録する際に利用可能なノートパソコンの情報を制御する方法はこれ以外にもありますが、この方法はトランザクション自体の定義で解決できる実装です。

書籍と雑誌の現在利用可能な冊数

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)	

需要が高い書籍や雑誌が利用可能かどうかを確認するという要件もあります。

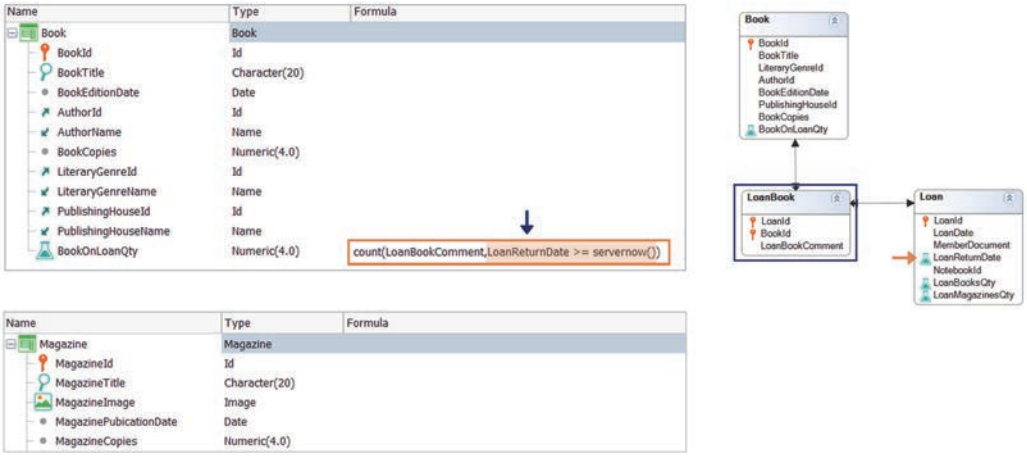
図書館が購入した書籍や雑誌の冊数は分かりますが、現在利用可能な冊数は分かりません。

この値を、トランザクションの設計に基づいてシンプルな方法で取得することはできるでしょうか。
Book トランザクションに新規項目属性 BookOnLoanQty を宣言することで、貸し出されている冊数を計算できるため、利用可能な冊数を知ることができます。

Count(LoanBookComment) の計算をこの新しい BookOnLoanQty 項目属性と関連付けるとどうなるでしょうか。貸し出されている本の冊数を効果的に集計できるでしょうか。
答えは「はい」です。計算を定義したトランザクションに関連付けられているテーブルは Book であり、この計算は LoanBook テーブルで行われます。これは、LoanBookComment 項目属性で決定されているためです。
これらのテーブルには共通の項目属性 BookId があり、GeneXus はこれを暗示的なフィルタとして利用します。そのため、この計算では、その書籍の貸し出し数の合計が返されます。

GeneXus

書籍と雑誌の現在利用可能な冊数



しかし、知りたいのは現在貸し出し中の冊数であるため、それを数えるための条件を計算に追加する必要があります。そのためには、返却日が現在の日付以降である貸し出しについて考えます。

次のように記述できます。

Count(LoanBookComment, LoanReturnDate >= Servernow())

Servernow() 関数を使用すると、サーバーの現在の日時を取得できます。Today() 関数を使用することもできます。

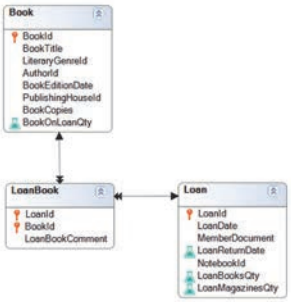
この計算条件を宣言することはできるでしょうか。はい、できます。これは、含まれている項目属性 (LoanReturnDate) が、宣言された計算が解決されるテーブルの拡張テーブル、つまり LoanBook に属しているためです。

GeneXus

書籍と雑誌の現在利用可能な冊数

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)	



そのため、合計冊数が分かっていて、貸し出し中の冊数が分かれば、現在利用可能な冊数が分かります。

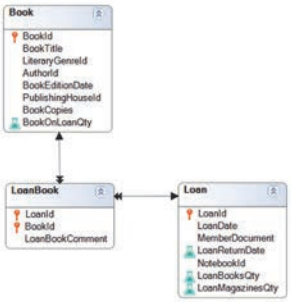
その場合、Book トランザクションの構造はこの図のようになります。

GeneXus

書籍と雑誌の現在利用可能な冊数

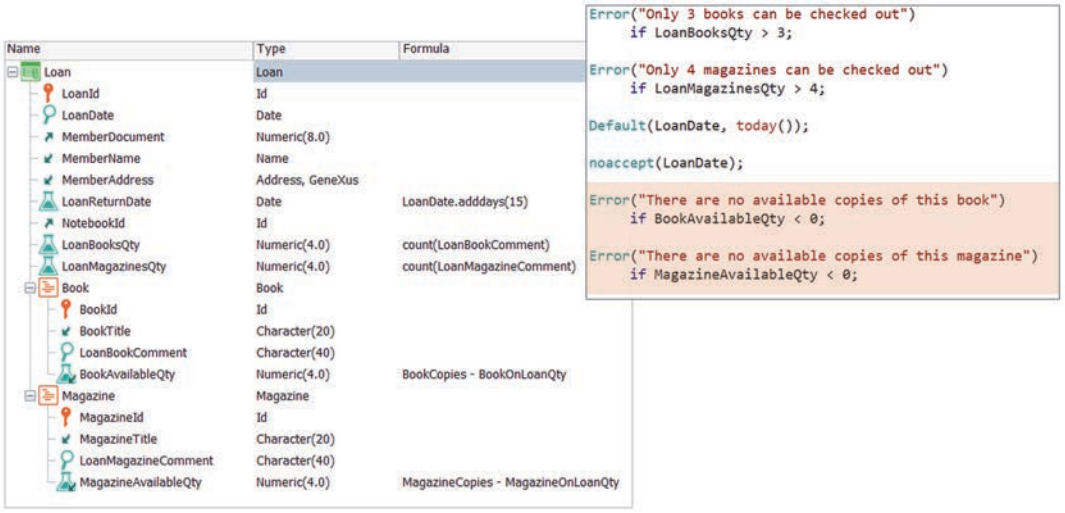
Name	Type	Formula
Book	Book	
BookId	Id	
BookTitle	Character(20)	
BookEditionDate	Date	
AuthorId	Id	
AuthorName	Name	
BookCopies	Numeric(4.0)	
LiteraryGenreId	Id	
LiteraryGenreName	Name	
PublishingHouseId	Id	
PublishingHouseName	Name	
BookOnLoanQty	Numeric(4.0)	count(LoanBookComment,LoanReturnDate >=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



同様に、特定の雑誌の利用可能冊数を知ることができます。

トランザクションの定義: 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

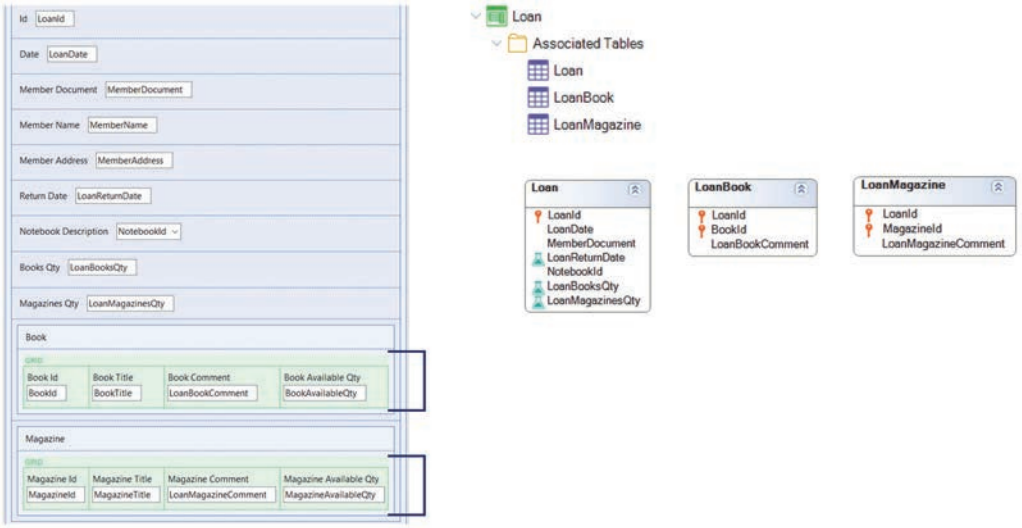
```
Error("Only 3 books can be checked out")
    if LoanBooksQty > 3;

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;
```

これで貸し出しを検証するために必要なすべての情報が分かったので、貸し出し可能な冊数を検証するために必要な項目属性を追加していきます。

- **BookAvailableQty**。Book レベルに追加します。
- **MagazineAvailableQty**。Magazine レベルに追加します。

これも Error ルールを使って制御します。

Loan トランザクション: フォームとテーブルの構造



これで、貸し出し登録の要件への対応ができました。この設計で生成されたフォームを見てみましょう。

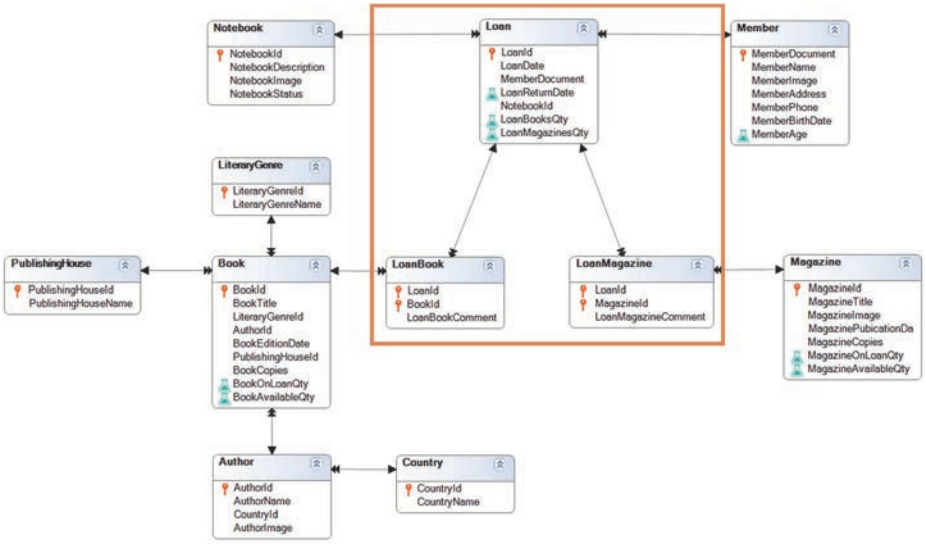
この Loan トランザクションでは、どのような関連テーブルが GeneXus によって作成されるでしょうか。

次の 3 つのテーブルが作成されます。
- **Loan**。LoanId を主キーとして、第 1 レベルに関連付けられます。
- **LoanBook**。LoanId および BookId を複合主キーとして、Book レベルに関連付けられます。
- **LoanMagazine**。LoanId および MagazineId を複合主キーとして、Magazine レベルに関連付けられます。

2 つのレベルを並列に定義することで、フォーム内にグリッドが並列して表示されます。エンドユーザーが、シンプルな設計にして、シンプルな画面で情報を管理することを希望した場合、このフォームは使いにくいかもしれません。

GeneXus

テーブルダイアグラム



現在の設計では、GeneXus によってこれらのテーブルが作成されました。Loan、LoanBook、LoanMagazine の各テーブルに注目してください。これらのテーブルは、定義した Loan トランザクションの 3 つのレベルに関連するテーブルに対応しています。

Loan トランザクションの別の設計を提案することはできるでしょうか。できます。

貸し出しのエンティティ: オプション 2: 3 つのトランザクション

- 貸し出しの全般的な情報を登録する画面

- 貸し出し対象の書籍を登録する画面

- 貸し出し対象の雑誌を登録する画面

それでは、**貸し出しの 2 つ目のモデリング方法**を確認しましょう。

エンドユーザーが、画面の読み込みを「効率化」または「軽く」して、**次の 3 つの画面**に情報を表示することを求めているとします。

- **貸し出しの全般的な情報を登録する画面**。日付、会員のデータ、ノートパソコンが含まれているかどうかの情報とあわせて貸し出しを登録する画面です。
- **貸し出し対象の書籍を登録する画面**。書籍を登録するための画面で、定義済みの必要なコントロールが適用されます。
- **貸し出し対象の雑誌を登録する画面**。雑誌を登録するための画面で、定義済みの必要なコントロールが適用されます。

これらはどうすれば実現できるでしょうか。

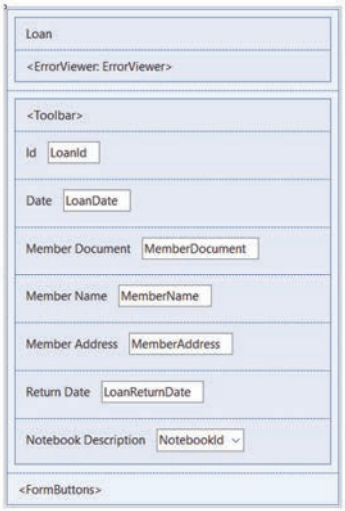
貸し出しのエンティティ: オプション 2: 3 つのトランザクション

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

```
Default(LoanDate, today());

noaccept(LoanDate);
```

関連するテーブル:

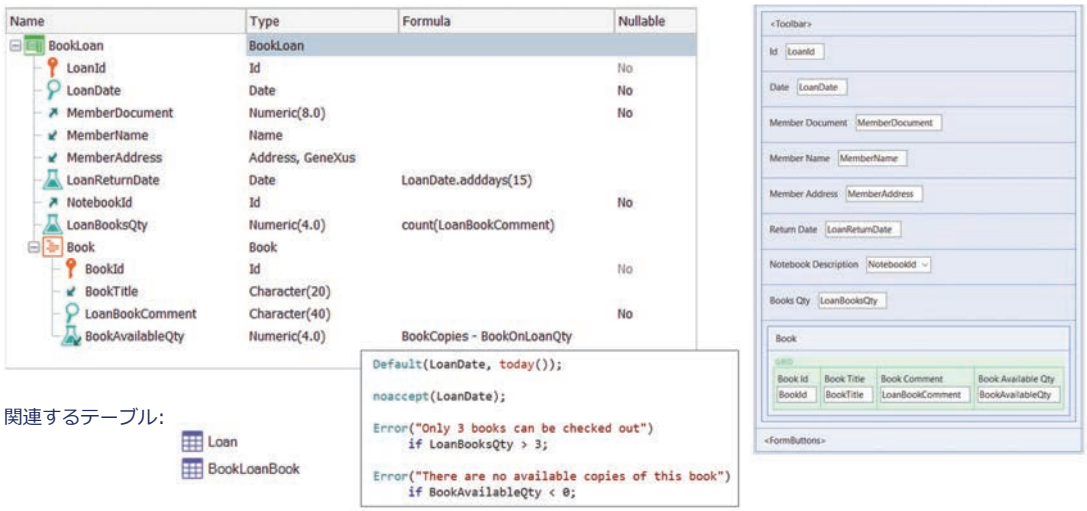
Loan



1 つ目の画面は、貸し出しの全般的なデータに対応します。そのため、Loan トランザクションのレベルを削除し、全般的な情報はそのままにします。その際、図に示すルールを適用します。

これにより、図の右側のフォームが生成されます。

貸し出しのエンティティ: オプション 2: 3 つのトランザクション



2 つ目の書籍を登録するための画面を作成するには、新しいトランザクションが必要です。これを BookLoan とします。

貸し出しの識別子を入力し、全般的な情報を表示し、書籍情報を入力します。そのため、この主キーも LoanId になり、新しいテーブルは作成されません。これは、LoanId が主キーであるテーブルが既に存在するためです。このように、同じ Loan エンティティを参照し、第 2 レベルに関連するテーブルのみが作成されます。

BookLoan トランザクションは図の左上に示す構造になり、中央下に示すルールが適用されています。

図の右側に示すフォームが生成されます。

貸し出しのエンティティ: オプション 2: 3 つのトランザクション

Name	Type	Formula	Nullable
MagazineLoan	MagazineLoan		
LoanId	Id		No
LoanDate	Date		No
MemberDocument	Numeric(8.0)		No
MemberName	Name		
MemberAddress	Address, GeneXus		
LoanReturnDate	Date	LoanDate.adddays(15)	
NotebookId	Id		Yes
LoanMagazinesQty	Numeric(4.0)	count(LoanMagazineComment)	
Magazine	Magazine		
MagazineId	Id		No
MagazineTitle	Character(20)		
LoanMagazineComment	Character(40)		No
MagazineAvailableQty	Numeric(4.0)	MagazineCopies - MagazineOnLoanQty	

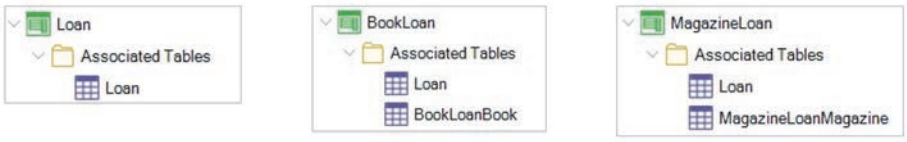
関連するテーブル:

- Loan
- MagazineLoanMagazine

```
Default(LoanDate, today());

noaccept(LoanDate);

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

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

雑誌の貸し出しの記録についても、同様のことを行います。図に示す構造およびルールを持つ MagazineLoan トランザクションを作成します。

貸し出しのエンティティ: オプション 2: 3 つのトランザクション



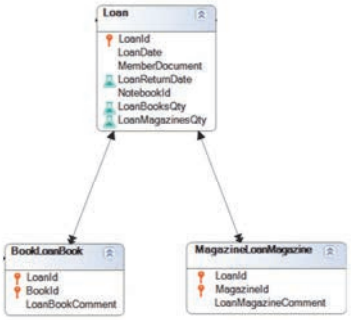
生成されたテーブルダイアグラムを分析する前に、この提案で作成されたトランザクションに関連付けられたテーブルを確認してみましょう。GeneXus によって次の 3 つのテーブルが作成されています。

- **Loan**
- **BookLoan Book**
- **MagazineLoanMagazine**

GeneXus

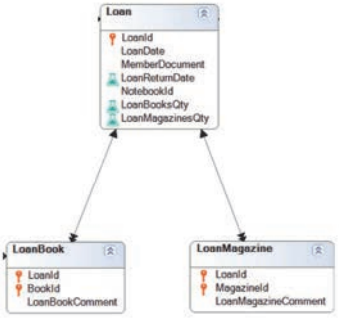
テーブルダイアグラム

現在のモデル:



=

以前のモデル:



テーブルダイアグラムを確認します。

Loan、 BookLoanBook、 MagazineLoanMagazine の各テーブルを見てみると、
先ほど提案した設計で生成されたテーブルとまったく同じ構造をしていることが分かります。

GeneXus

テーブルダイアグラム

現在のモデル:

以前のモデル:

つまり、これらの設計の違いは、生成される構造にあるのではなく、情報を管理するためにエンドユーザーに表示される画面にあります。どのような設計が最適かは、常にエンドユーザーによって決まります。

データベース構造の面では、まったく同じです。

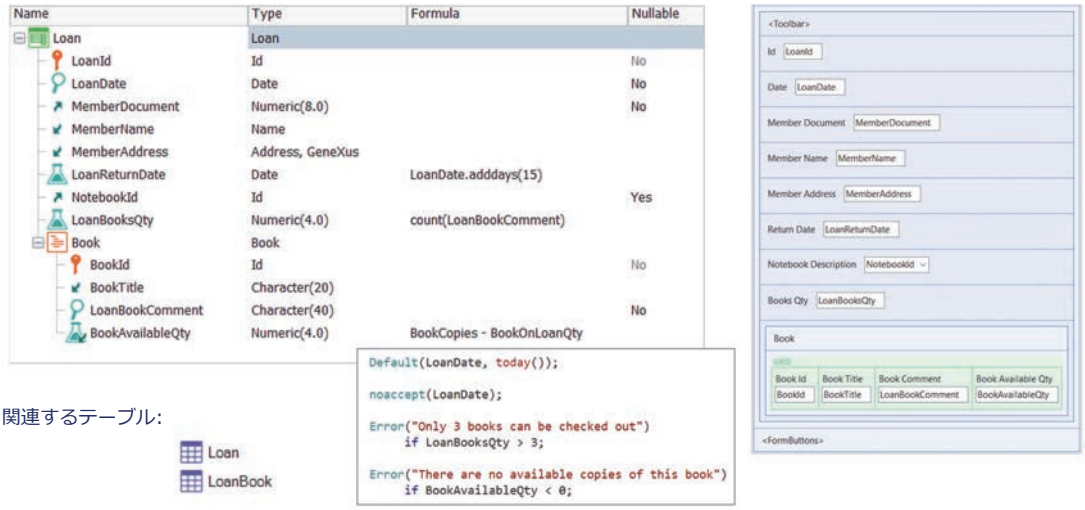
GeneXus

貸し出しのエンティティ: オプション 3: 2 つのトランザクション

貸し出しの 3 つ目のモデリング方法を提案することもできます。

図書館は書籍の貸し出しを基本としており、エンドユーザーは貸し出しの全般的なデータと貸し出される書籍のデータを入力する単一の画面を求めていると考えることができます。そして、別の画面から、その貸し出しに含まれる雑誌を登録します。

貸し出しのエンティティ: オプション 3: 2 つのトランザクション



関連するテーブル:

これを実現するために、書籍の情報を記録するための第 2 レベルを持つ Loan トランザクションを検討します。この図に示す構造になり、その下に表示されているルールが適用されます。

GeneXus

貸し出しのエンティティ: オプション 3: 2 つのトランザクション

Name	Type	Formula	Nullable
MagazineLoan	MagazineLoan		
LoanId	Id		No
LoanDate	Date		No
MemberDocument	Numeric(8.0)		No
MemberName	Name		
MemberAddress	Address, GeneXus		
LoanReturnDate	Date	LoanDate.adddays(15)	
NotebookId	Id		Yes
LoanMagazinesQty	Numeric(4.0)	count(LoanMagazineComment)	
Magazine	Magazine		
MagazineId	Id		No
MagazineTitle	Character(20)		
LoanMagazineComment	Character(40)		No
MagazineAvailableQty	Numeric(4.0)	MagazineCopies - MagazineOnLoanQty	

```
Default(LoanDate, today());

noaccept(LoanDate);

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

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

関連するテーブル:

- Loan
- MagazineLoanMagazine

貸し出しに含まれる雑誌の登録用に、図に示す構造を持つ MagazineLoan トランザクションがあります。これには、その下に表示されているルールが適用されます。

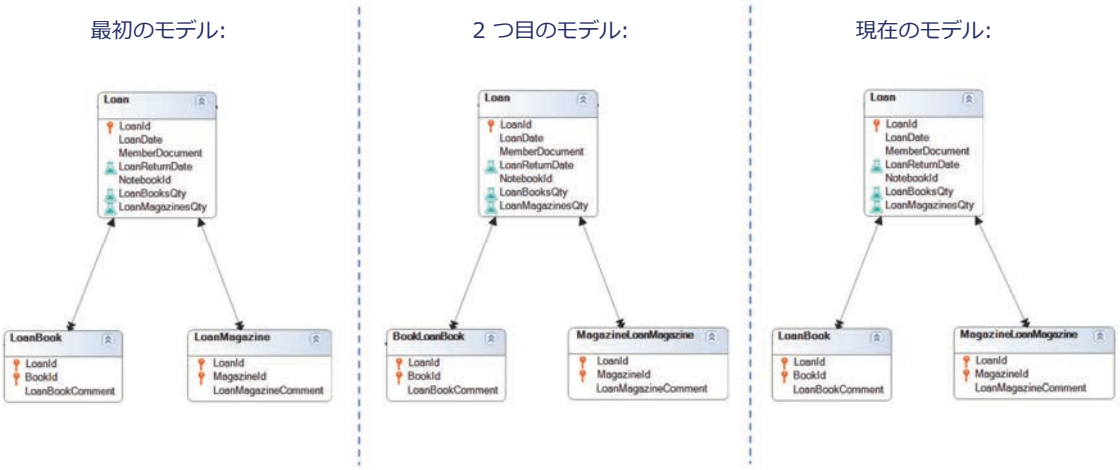
GeneXus

貸し出しのエンティティ: オプション 3: 2 つのトランザクション

これらのトランザクションに関連して、GeneXus はどのようなテーブルを作成するでしょうか。3 つのテーブル、Loan、LoanBook、MagazineLoanMagazine が作成されます。

- Loan と LoanBook は Loan トランザクションで関連付けられています。
- Loan と MagazineLoanMagazine は MagazineLoan トランザクションに関連付けられています。

テーブルダイアグラム



生成されたテーブルダイアグラムをもう一度見てみましょう。

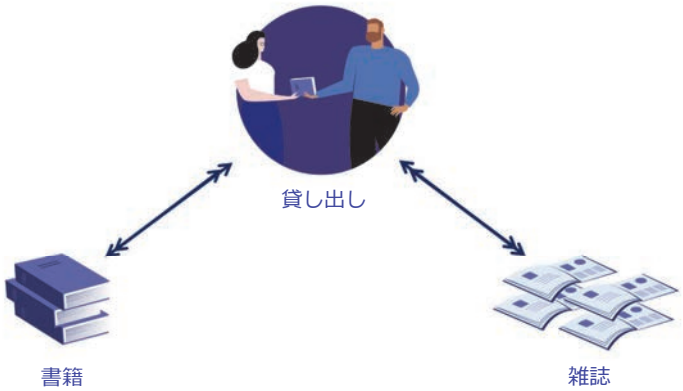
Loan、LoanBook、MagazineLoanMagazine の各テーブルの構造を改めて見てみると、前の提案で生成されたテーブルと同じであることが分かります。

これらの提案では、エンドユーザーに表示される画面は異なりますが、データベースの構造は同じです。

書籍よりも雑誌の貸し出しを優先した場合でも、生成されるテーブルは同じ構造になります。

GeneXus

現実



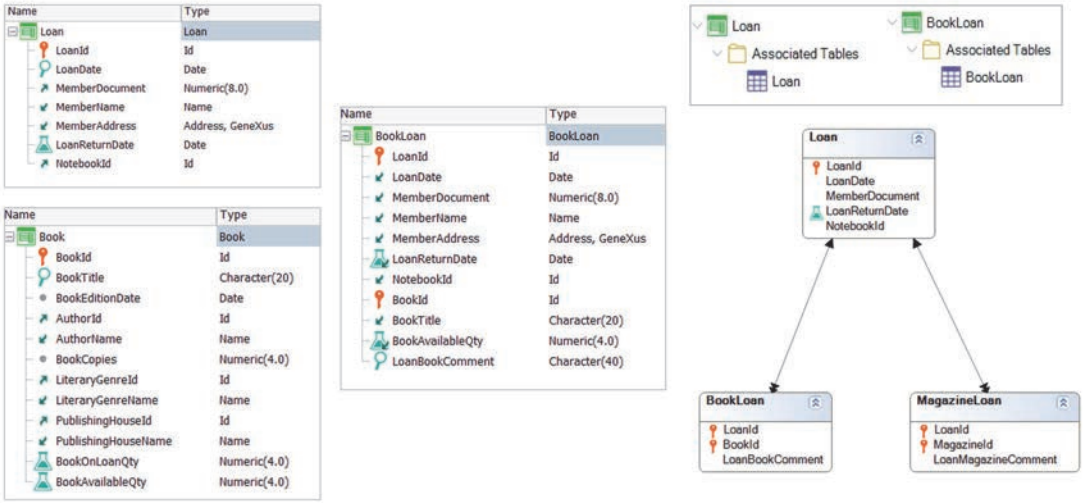
では、第 2 レベルを含まない完全にフラットな方法はあるでしょうか。エンドユーザーが、グリッドのないシンプルな画面を求めている場合、どのようなものを提案できるでしょうか。

エンティティである「貸し出し」と「書籍」の間、また、「貸し出し」と「雑誌」の間には、N 対 N の関係があることを思い出してください。

「貸し出し」と「書籍」について分析し、それをもって「貸し出し」と「雑誌」の場合を類推することにします。

GeneXus

貸し出しのエンティティ: オプション 4:
3 つのシンプルなトランザクション (1 レベルのトランザクション)



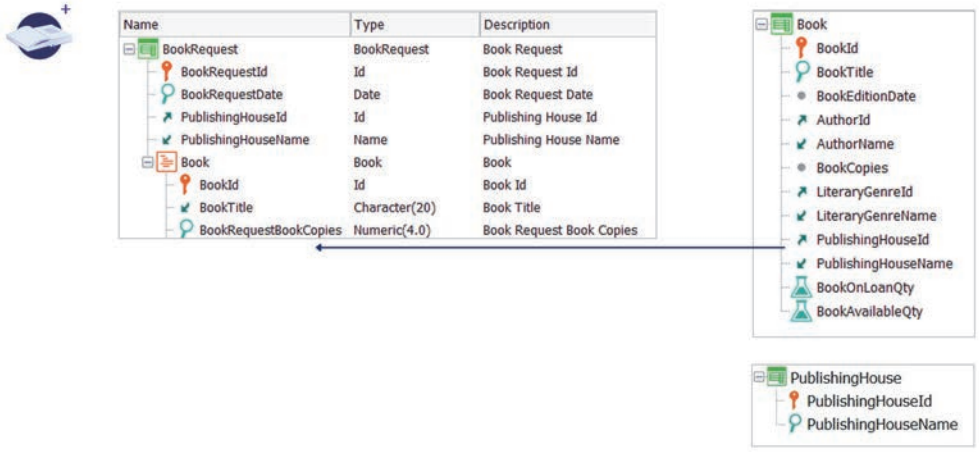
構造を考えてみましょう。

これらの間に N 対 N の関係が存在するためには、LoanId と BookId の複合キーを持つ関係テーブルが必要です。

LoanId および BookId で構成される主キーを持ち、かつ、図の中央に示す構造を持つ BookLoan トランザクションを定義します。

どのようなテーブルが生成されるでしょうか。先ほど見たものと同じテーブルが生成されます。

BookRequest トランザクション: 構造



この時点では、特定の出版社に対する書籍の追加リクエストのモデリングがまだ解決されていません。

これに対応するために、BookRequest トランザクションを定義します。出版社に対するリクエストを表し、次の項目属性を持ちます。
- **BookRequestId**。ID ドメインに基づきます。
- **BookRequestDate**。既定では当日になります。
- **PublishingHouseId**。リクエストは出版社に対して行うためです。
- **PublishingHouseName**。外部キー PublishingHouseId から推論される項目属性です。

この新しいエンティティと書籍との間には、N 対 N の関係があります。リクエストに複数の書籍が含まれ、1 冊の書籍が複数のリクエストに含まれる場合があるためです。ここまで、複数の設計方法があることを見てきました。
では、リクエストの第 2 レベルとして書籍を追加します。指定した書籍が該当する出版社から発行されていることを確認するのは重要な要件です。
Book トランザクションの構造を思い出してください。PublishingHouseId は外部キーとなっています。これは、1 冊の書籍は 1 つの出版社から発行されるためです。

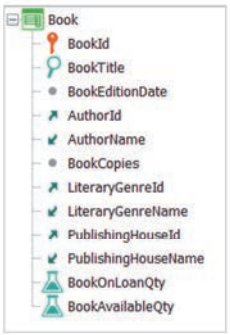
この項目属性を新しいトランザクションに追加して、BookId から推論できるようにすること、また、対象の書籍が実際にその出版社から出版されているか確認できるようにすることは可能でしょうか。

BookRequest トランザクション: 構造

Name	Type	Description
BookRequest	BookRequest	Book Request
BookRequestId	Id	Book Request Id
BookRequestDate	Date	Book Request Date
PublishingHouseId	Id	Publishing House Id
PublishingHouseName	Name	Publishing House Name
Book	Book	Book
BookId	Id	Book Id
BookTitle	Character(20)	Book Title
BookRequestBookCopies	Numeric(4.0)	Book Request Book Copies
PublishingHouseId		

Duplicate Attribute Name: 'PublishingHouseId'

複数参照の競合:
どこでサブタイプグループを定義するか



追加しようとするとどうなるでしょうか。GeneXus はエラーを返します。これは、PublishingHouseId 項目属性が既にトランザクション構造内で宣言されており、再度追加することができないためです。

複数参照の問題が発生します。出版社の概念を複数回参照する必要があります。では、このような競合を解決するにはどうすればよいでしょうか。これにはサブタイプを使用します。つまり、同じ出版社の概念を参照する新しい項目属性を、別の名前で定義します。

しかし、サブタイプはどこで定義すればよいでしょうか。ヘッダーで参照される出版社の概念内でしょうか、それとも第 2 レベルで参照されるものでしょうか。これらは同じことでしょうか。

BookRequest トランザクション: 構造

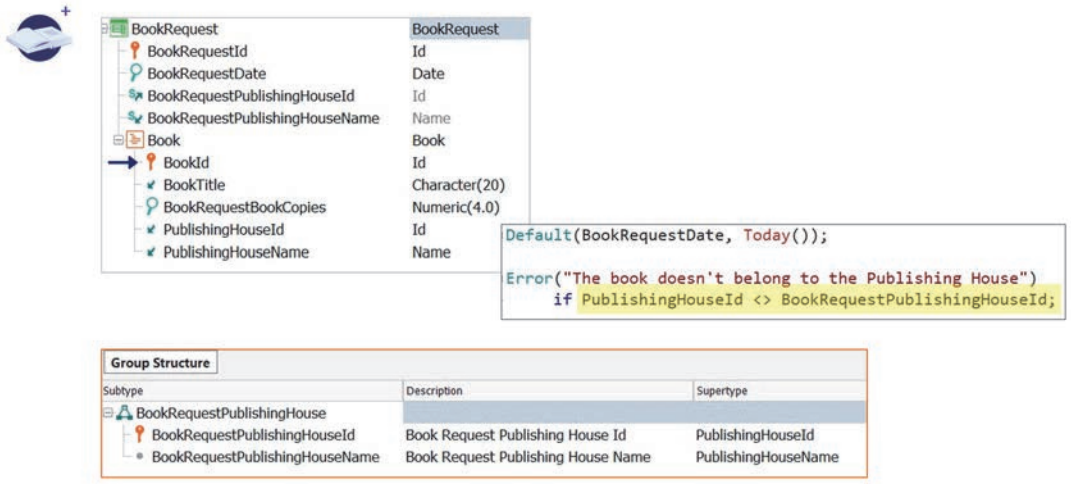
Name	Type	Description
BookRequest	BookRequest	Book Request
BookRequestId	Id	Book Request Id
BookRequestDate	Date	Book Request Date
PublishingHouseId	Id	Publishing House Id
PublishingHouseName	Name	Publishing House Name
Book	Book	Book
BookId	Id	Book Id
BookTitle	Character(20)	Book Title
BookRequestBookCopies	Numeric(4.0)	Book Request Book Copies
BookRequestPublishingHouseId		

複数参照の競合:
どこでサブタイプグループを定義するか

トランザクションの設計を確認しましょう。PublishingHouseId 項目属性が Book トランザクション構造内に外部キーとして存在しています。そのため、GeneXus はその値を BookId から推論します。
BookId および PublishingHouseId がほかのトランザクションに存在する場合は、BookRequest の第 2 レベルの状況と同様に、GeneXus はその関係を適用し、PublishingHouseId が推論された外部キーとなります。

ここで、項目属性の名前を変更し、必要なサブタイプグループを定義するのは、適切といえるでしょうか。適切ではありません。なぜなら、この関係を削除することになり、Book で以前定義されたものとは異なる PublishingHouseId の値をユーザーが追加することを許可してしまうからです。

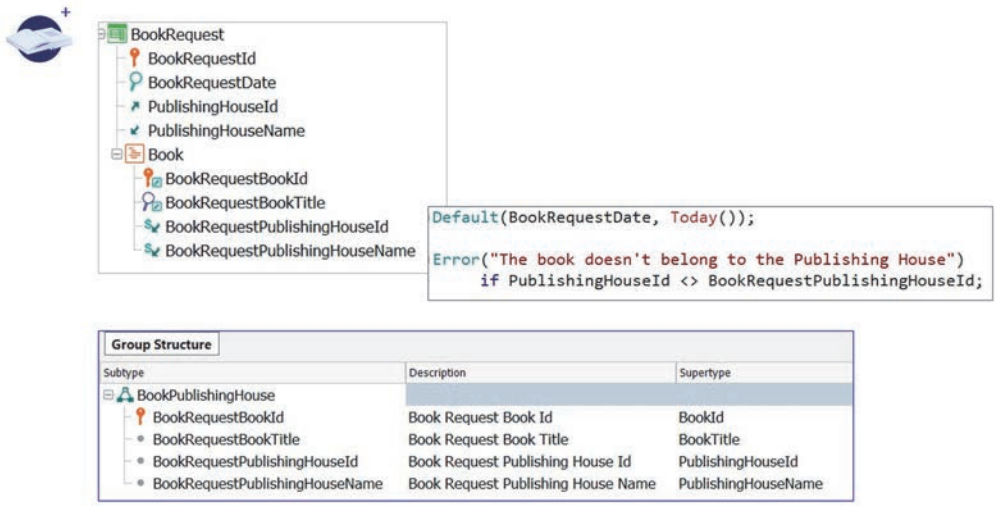
BookRequest トランザクション:
第 1 レベル内のサブタイプグループ



次に、BookRequest ヘッダーを見てみましょう。PublishingHouseId 項目属性が共通の外部キーとなっています。これは何かから推論されたものではありません。そのため、これを完全に削除し、新しい項目属性を定義して、PublishingHouseId と PublishingHouseName のサブタイプとして宣言します。これらを Error ルールで比較して、リクエストされた書籍が、指定された出版社から出版されたものであることを検証できるようにします。

要件が満たされるように、図に示すルールを宣言します。
この Error ルールには両方のレベルの項目属性が含まれているため、GeneXus によって第 2 レベルに関連付けられたルールになります。そのため、各行でトリガーされます。つまり、新しい BookId を入力すると、条件が評価され、true の場合はエラーがトリガーされます。

BookRequest トランザクション:
第 2 レベル内のサブタイプグループ

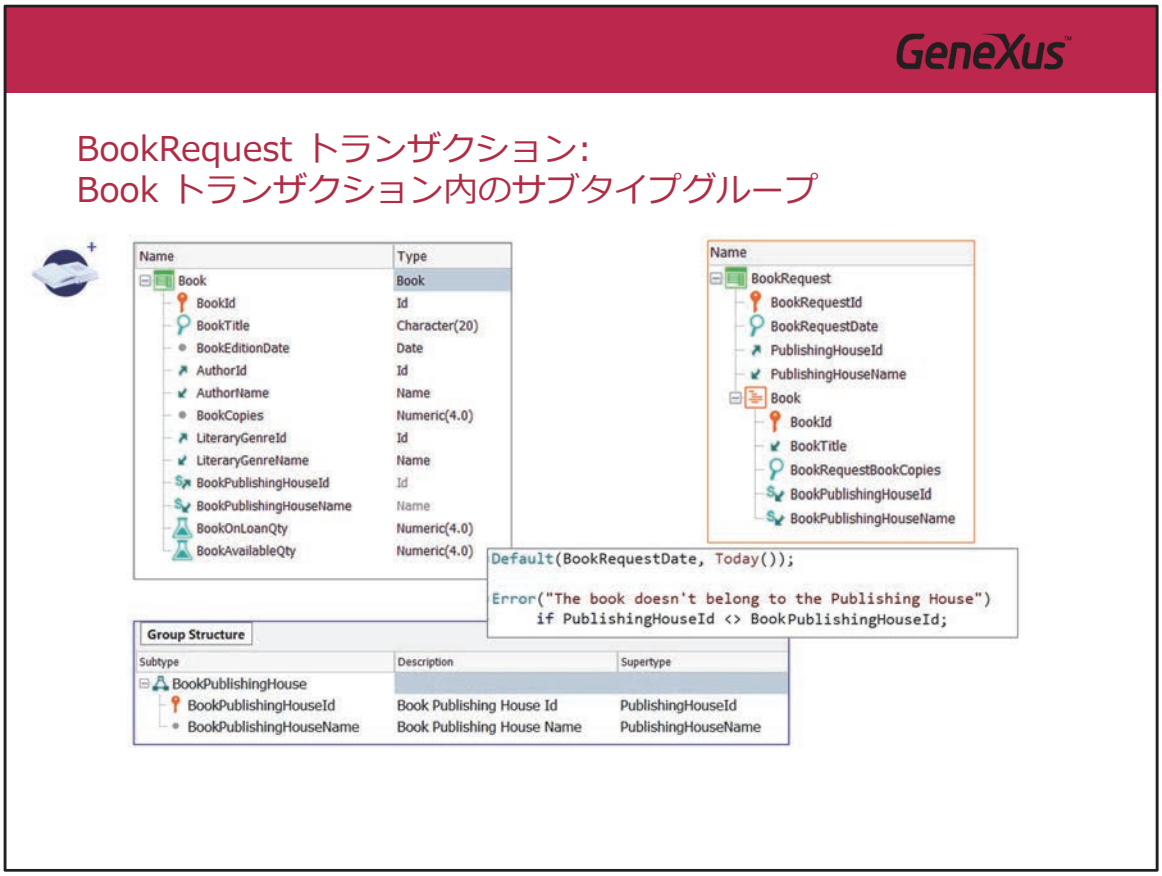


Group Structure

Subtype	Description	Supertype
BookPublishingHouse		
BookRequestBookId	Book Request Book Id	BookId
BookRequestBookTitle	Book Request Book Title	BookTitle
BookRequestPublishingHouseId	Book Request Publishing House Id	PublishingHouseId
BookRequestPublishingHouseName	Book Request Publishing House Name	PublishingHouseName

これで要件を満たすことができました。サブタイプのグループとして第 2 レベル全体を定義した場合はどうでしょうか。

その場合も解決することができます。ただし、次の点が異なります。(サブタイプの章で詳しく分析したように) ソリューションが明確ではなく、必要なところで推論するために、より多くのサブタイプを作成することが必要になる可能性があります。ただし、あいまいさが発生するテーブルと同じテーブル内で解決することができるというメリットもあります。



レベルが 2 つあるトランザクションで、サブタイプの定義を回避するために、ほかの方法を提案できないでしょうか。

できます。ただし、基準として、参照の競合については、それが発生したトランザクション内で対処することを推奨します。先ほど BookRequest トランザクション内で解決したのはそのためです。

では、Book トランザクションを確認し、そこにサブタイプグループを定義します。

これにはどのようなメリットがあるでしょうか。BookRequest トランザクション内に参照の競合がなくなります。
ただし、Book に書籍カタログなどのクエリが既に定義されている場合は、一部の項目属性の名前が変更されているため、それらのクエリを更新する必要があります。

最後に、サブタイプを使用しないようにすることはできるでしょうか。はい、できます。たとえば、各行で、記録する前に、指定した出版社から出版されたものかどうかを返すプロセスを呼び出すことで実現できます。
サブタイプグループの定義は回避できますが、トランザクション自体の定義で解決できる条件を評価するためのプロセスが各行で呼び出されます。

まとめ

常に現実を分析し、評価してモデリングし、適切と思われる
方法を実装する。常にエンドユーザーと協力する。

まとめると、常に現実を分析し、評価してモデリングし、適切と思われる方法を実装する必要があります。そのためには、設計を選択する際に指針となるエンドユーザーの協力が不可欠です。

アプリケーションは、エンドユーザーがビジネスをより適切に管理し、展開するためのサポートツールであることを忘れないでください。