2019-12-05 李宇桐

# ATL参考资料

官网：<https://www.eclipse.org/atl/>

来自Github模型转换的项目代码

<https://github.com/eclipse/atl/tree/master/tests/org.eclipse.m2m.atl.tests/data/inputs>

# 在eclipse上安装ATL

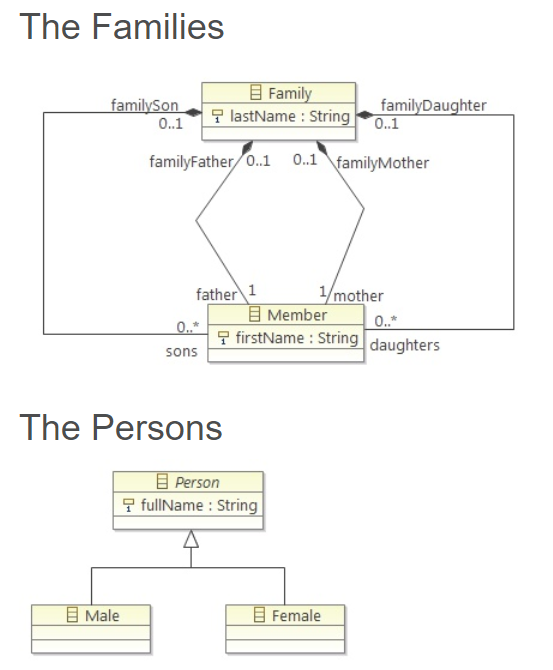
可参考：<https://wiki.eclipse.org/ATL/Tutorials_-_Create_a_simple_ATL_transformation>

步骤：Help – Install New Software – 选择更新站点 – 搜索ATL – 下载 – 验证

# 入门示例：Families2Person

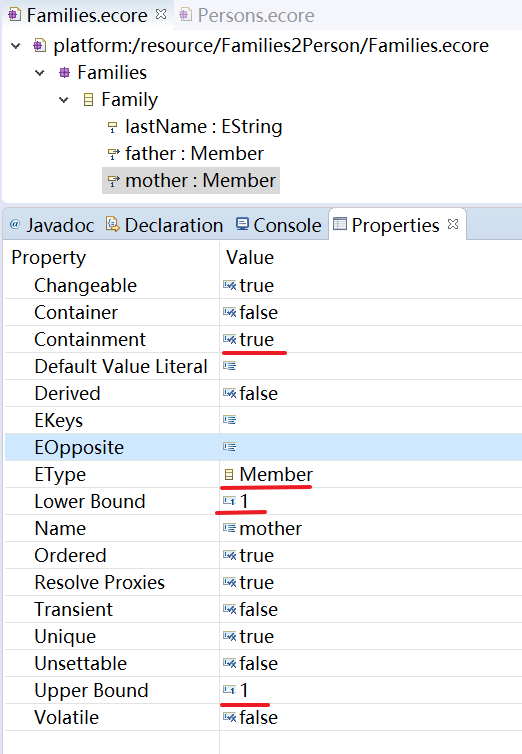
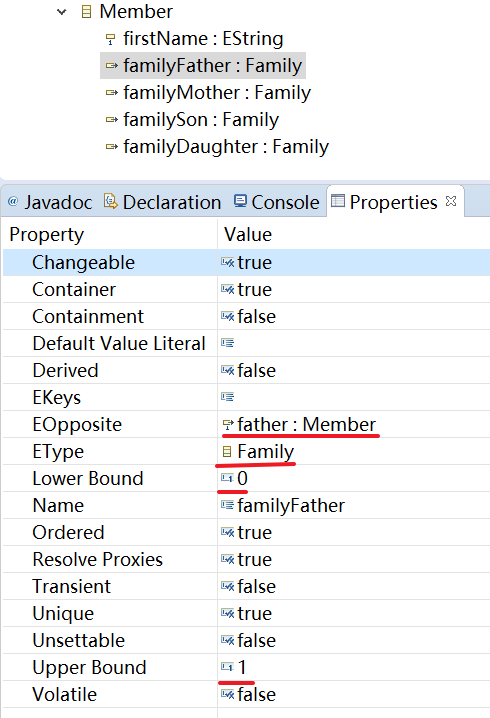
Family2Person：<https://wiki.eclipse.org/ATL/Tutorials_-_Create_a_simple_ATL_transformation>

## 3.1 源元模型和目标元模型的创建

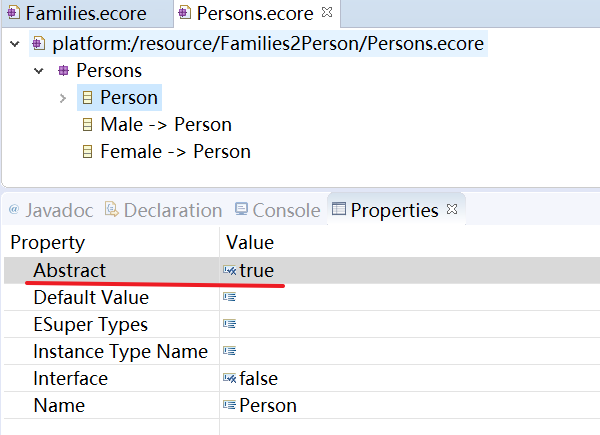


1. On one side (the source), we have a list of families.
2. Each family has a last name, and contains a father, a mother and a number of sons and daughters (0, 1 or more) all with a first name.
3. We want to transform this list into a new list of persons (the target), this means that each member of the family will be a person, without differentiating parents from children, and with no link between the members of a same family (except a part of their name).
4. In the end, we must only have a person with its full name (first name & last name), male or female.

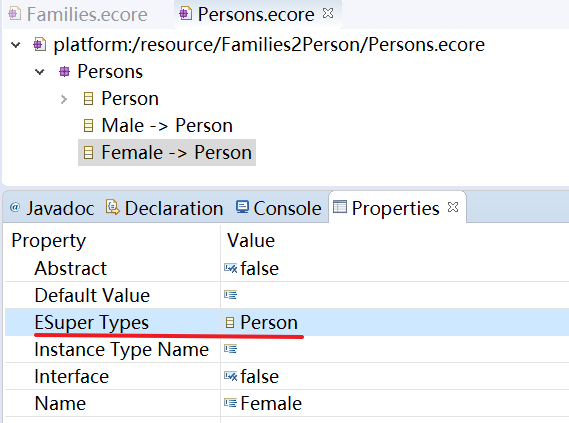
* Click on it, and in the Properties view, enter "Families" in the value of the "Name" property. This node is where we are going to put everything that makes a family.
* Now we have to make the links between the family and the members. For this purpose, you have to create children of the Family of the type EReference. Name these references "father", "mother", "sons" and "daughters". They will have the EType Member.
* we do the same for the Member class. It also needs references towards the Family class. Just add 4 EReferences to the Member class: "familyFather", "familyMother", "familySon" and "familyDaughter" with EType Family.
* Then, in order to tell which member refer to which family member, set their EOpposite field to their reference in the Family class (for example, familyFather refers to the father reference of the Family class).

接下来创建目标元模型。



1. name the root child node to "Persons".
2. Then add it a class "Person", with one attribute: "fullName" of EType EString and multiplicity 1..1.
3. Then set the Abstract attribute of the Person class to "true". We need to do this because we won't directly implement this class, but two other subclasses: "Male" and "Female", according to who was the person in the family, a man or a woman.



1. Create these two classes at the same level than Person. We make them subclasses of Person by setting their ESuper Types property to Person.

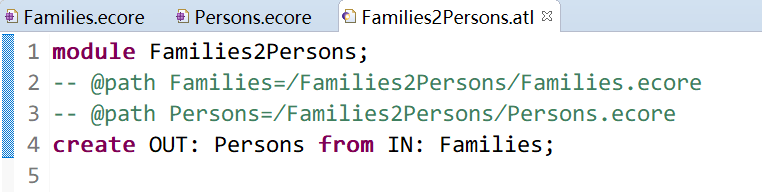
## 3.2 the ATL transformation code

目标：

1. melting his first and last name to have a full name
2. defining whether it's a man or a woman
3. copy these pieces of information into a Person object

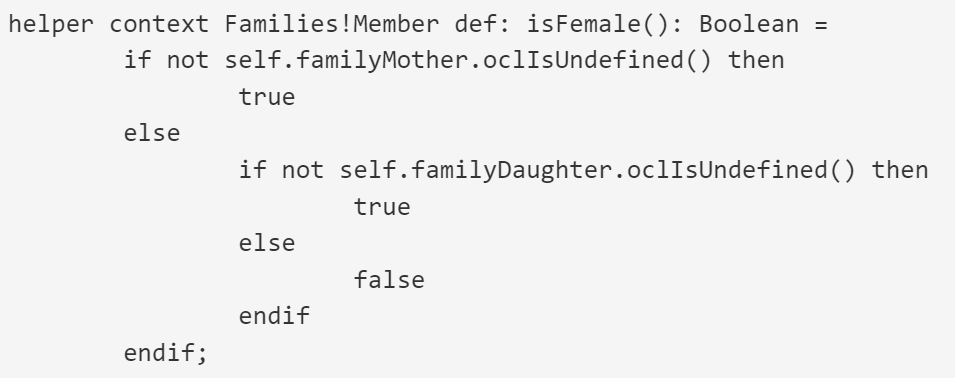
创建ATL File后：

1. 声明两个@path: the editor can use the auto-completion and documentation when we type in some code concerning the two metamodels
2. OUT和IN: tell ATL that we have families in and we want persons out

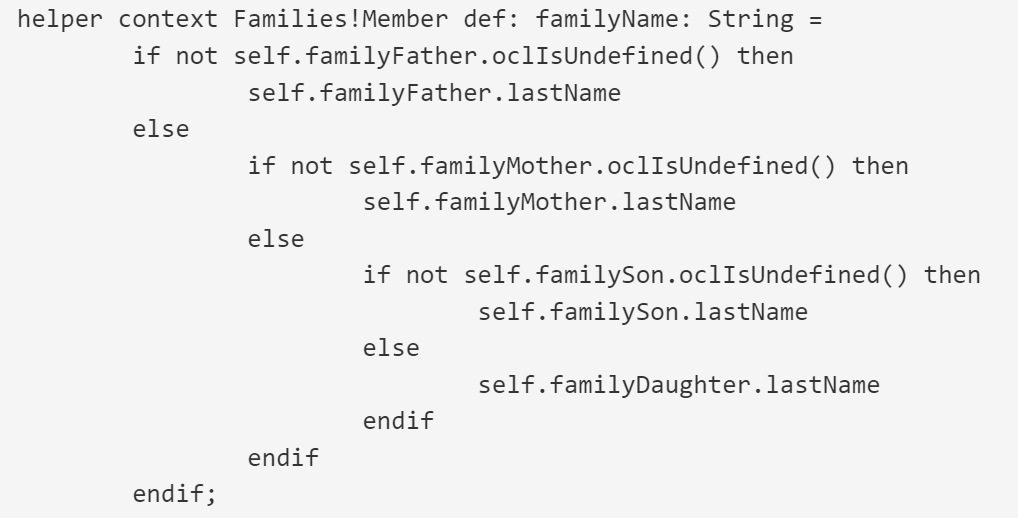


定义一些helpers：

* The first one is called(调用) on a member of a family (context Families!Member), gives us a boolean (: Boolean), and tells us whether the member is a female or not, by verifying if the familyDaughter or familyMother reference is defined or not.

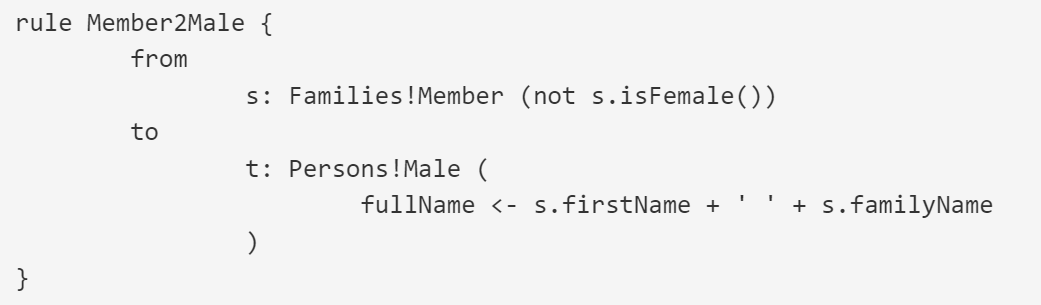


* The second one is also called(调用) on a member of a family, this time gives us a string (: String) and returns the last name of the member. It must look for it in every reference to the family, to see which one is defined (familyFather, familyMother, familySon or familyDaughter)

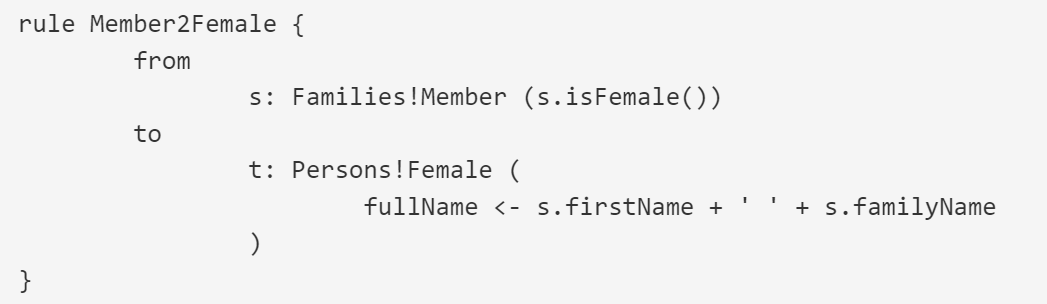


接下来，定义规则：creating male and female persons from members of families

* Each rule will be called(调用) on the object that respect the filter predicate in the from part.
* the first rule takes each member of each families (from s: Families!Member) that is not a female (using the helper we described above, not s.isFemale())
* And then it creates a male person (to t: Persons!Male) and set its fullName attribute to the first name of the member followed by its last name (using the helper familyName we saw above).

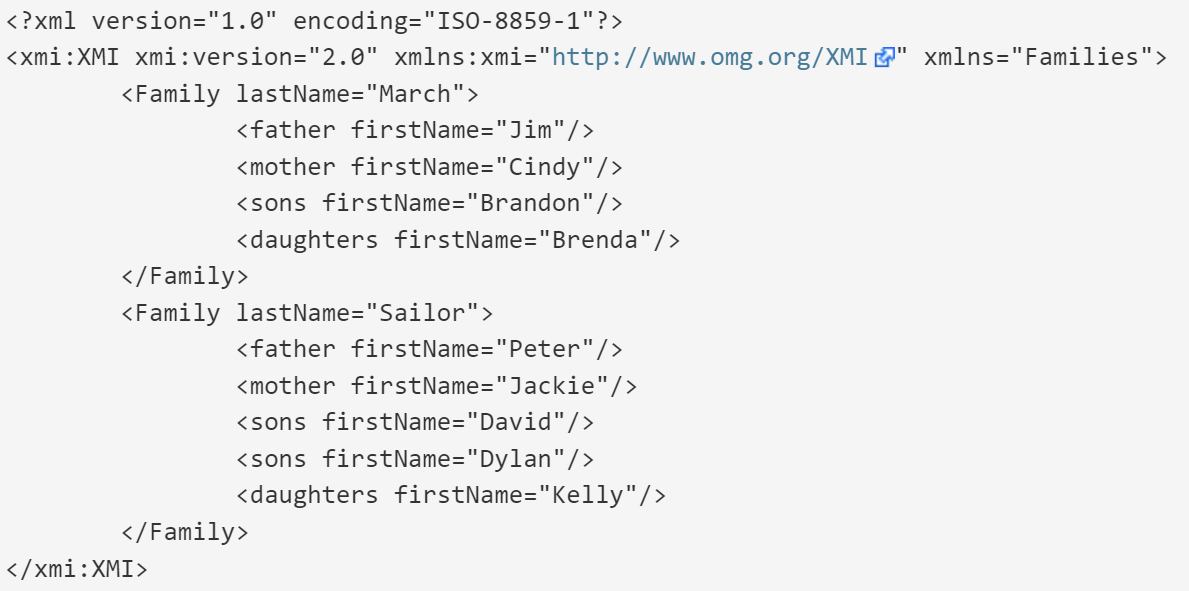


* The principle is the same for the second rule, whereas this time it takes only the female members.



## 3.3 The sample families model file

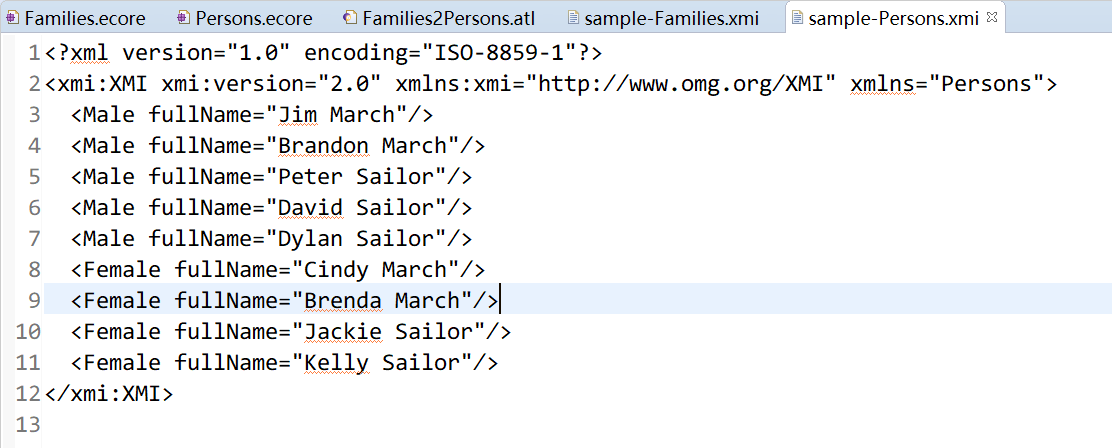
File – New – File，选择对应项目，命名为sample-Families.xmi



## 3.4 The launch configuration

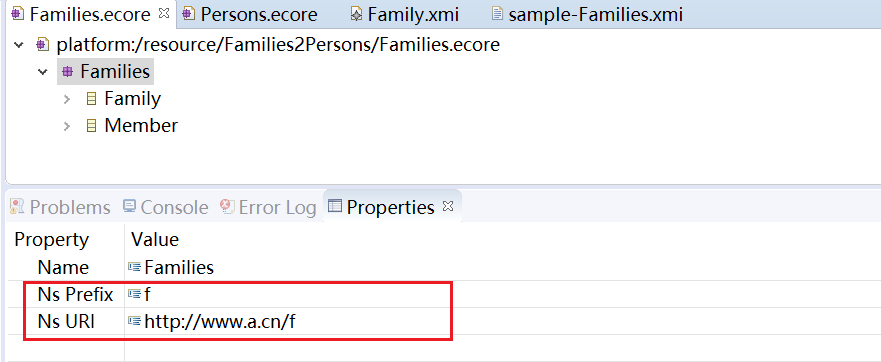
When you are in the ATL file (Families2Persons.atl), click on Run > Run

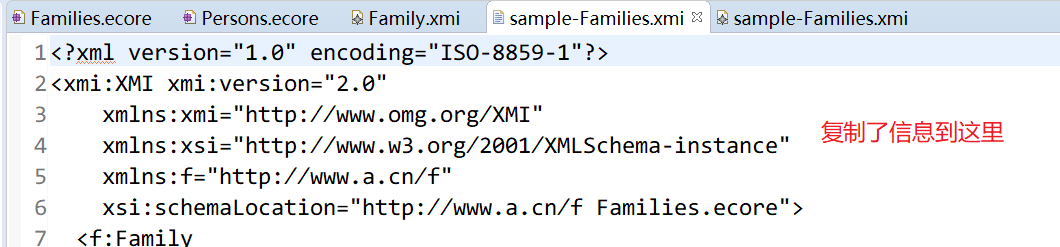
## 3.5 The result



## 3.6 xmi视图操作

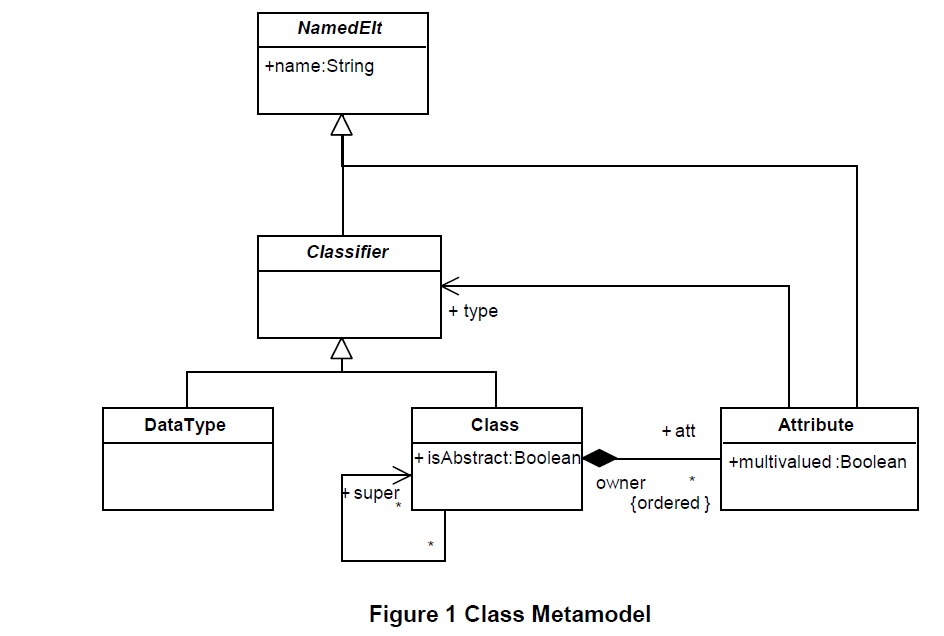
然而还是有问题…



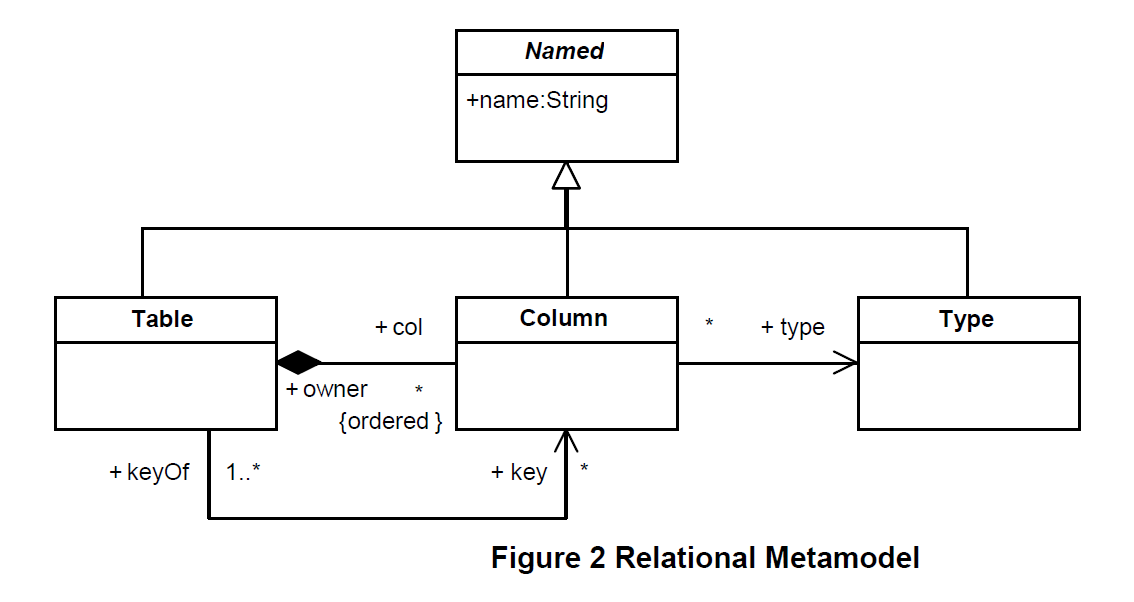


# Class2Relational

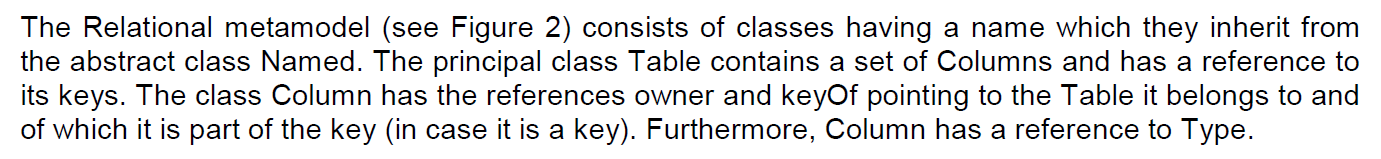
## 4.1 Class元模型和Relational元模型



Class元模型由具有名称的类组成，这些名称从抽象类NamedElts继承。主体类是Class，它包含Attribute类型的一组属性，并具有用于对继承树进行建模的super引用。DataType类对原始数据类型进行建模。Class和DataType继承自Classifier，Classifier用于声明Attributes的类型。属性可以是多值的，这对转换具有重要影响。

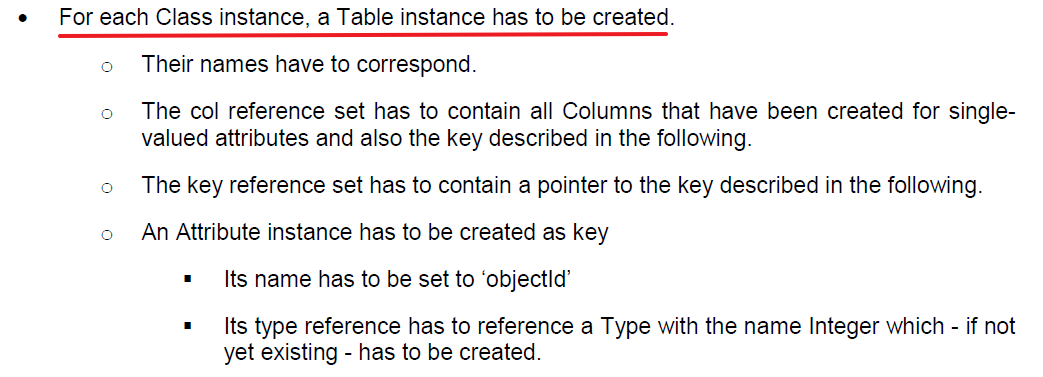


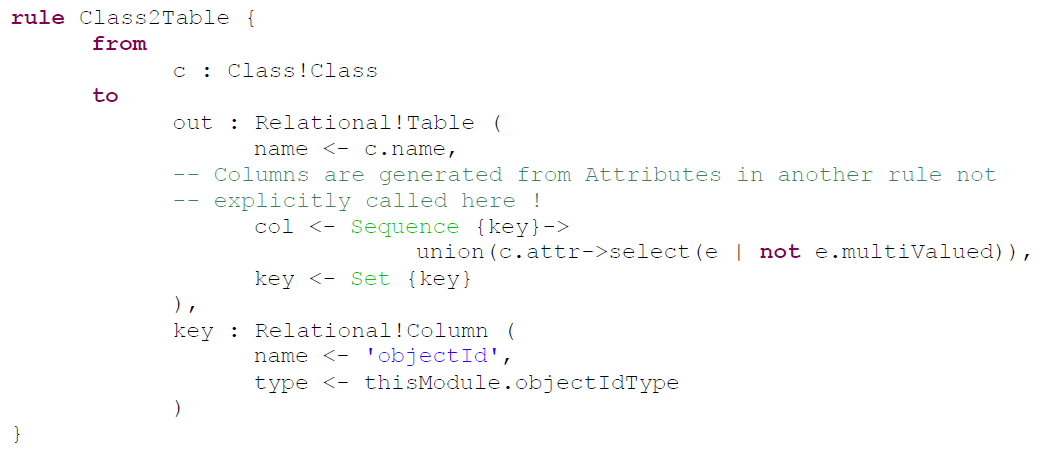
Relational元模型由具有名称的类组成，这些名称从抽象类Named继承。主体类Table包含一组Columns并用其keys进行引用。类Column的引用owner指向它所属的表，keyOf引用说明它是键的一部分（如果它是键的话）。此外，Column引用了Type。



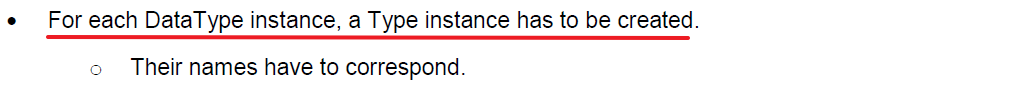
## 4.2 转换规则

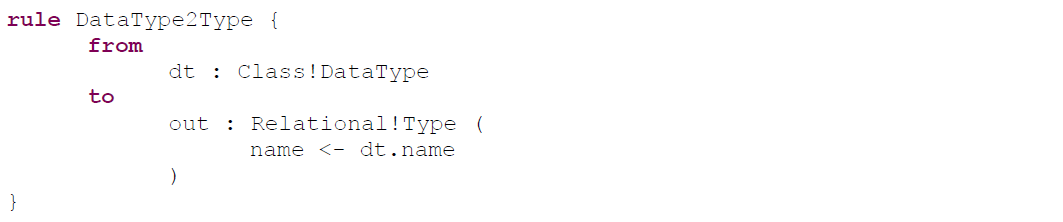
1. 对于每个类实例，创建一个Talbe实例



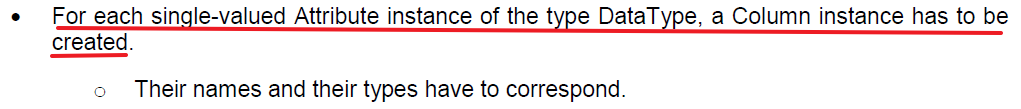


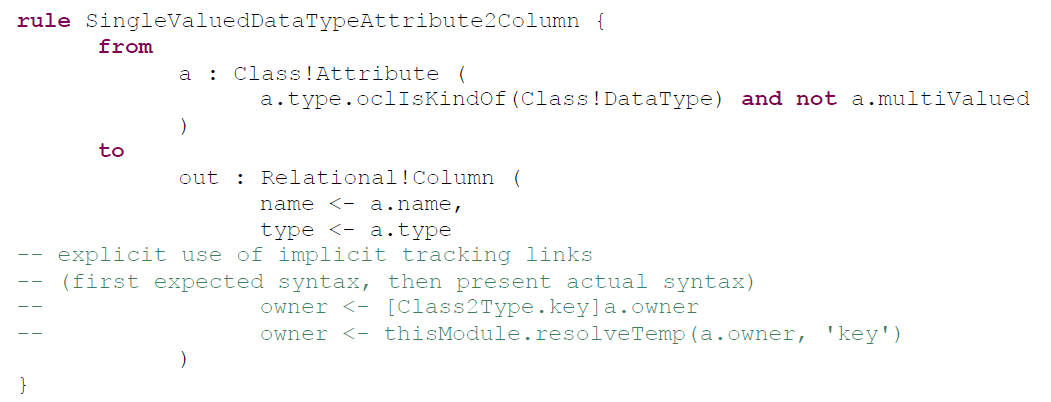
1. 对于每个DataType实例，创建一个Type实例



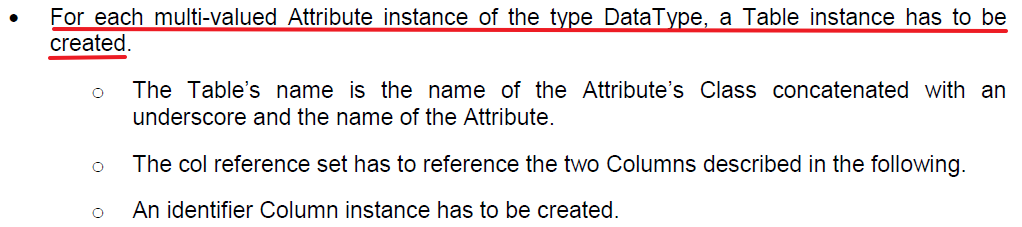


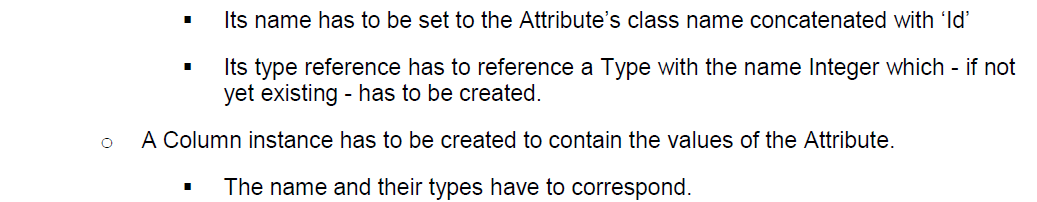
1. 对于每个DataType类型的单值属性实例，创建一个Column实例

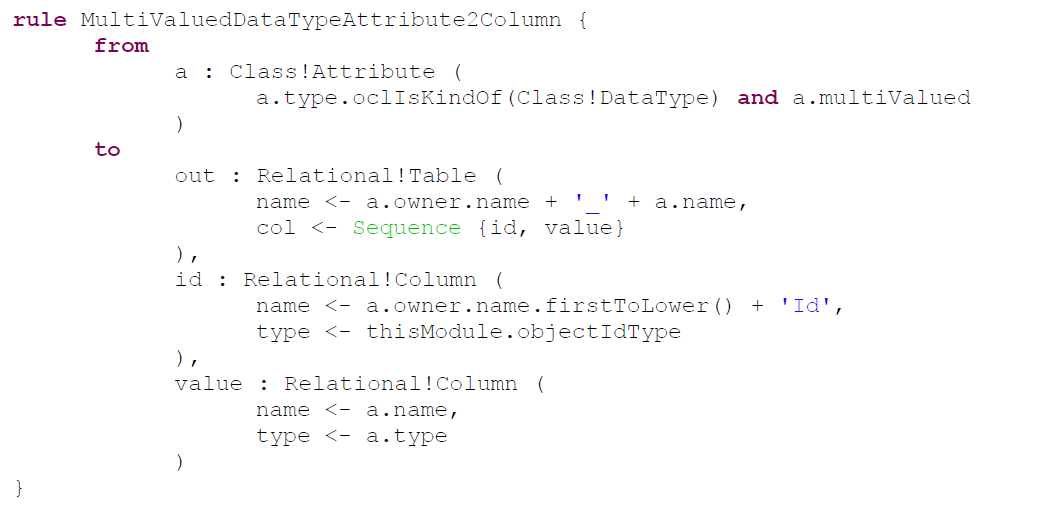




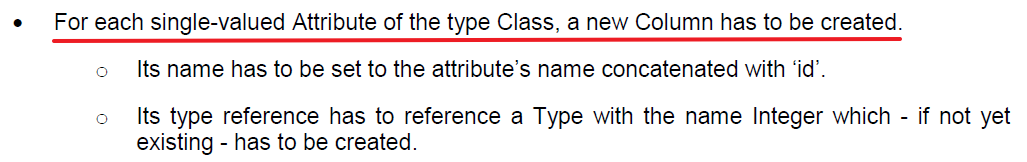
1. 对于每个DataType类型的多值属性实例，创建一个Table实例

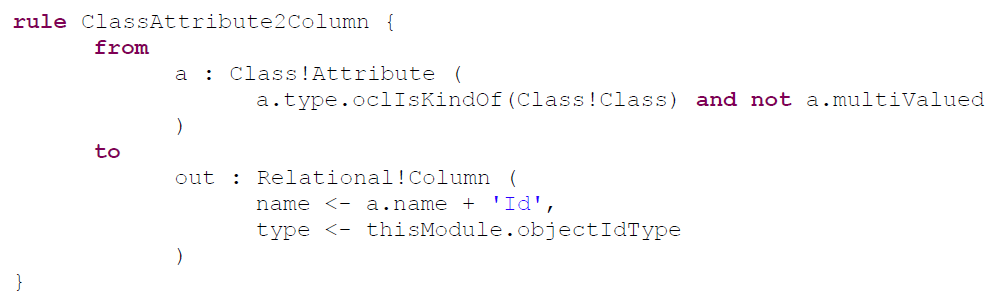




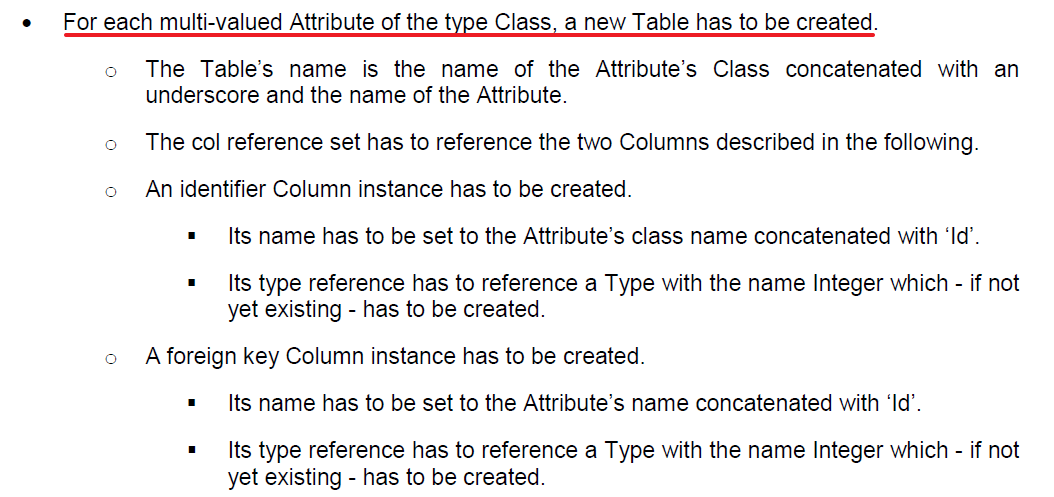


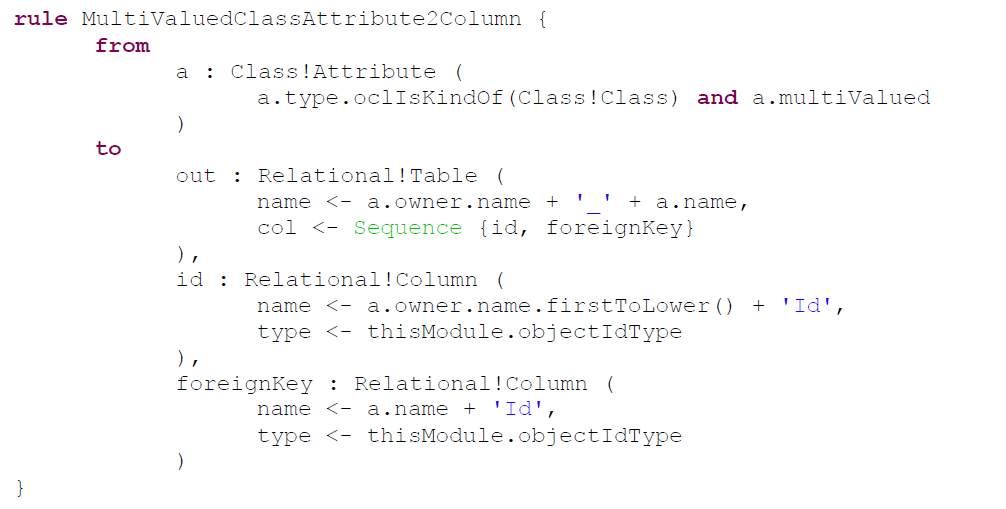
1. 对于每个Class类型的单值属性，创建一个Column





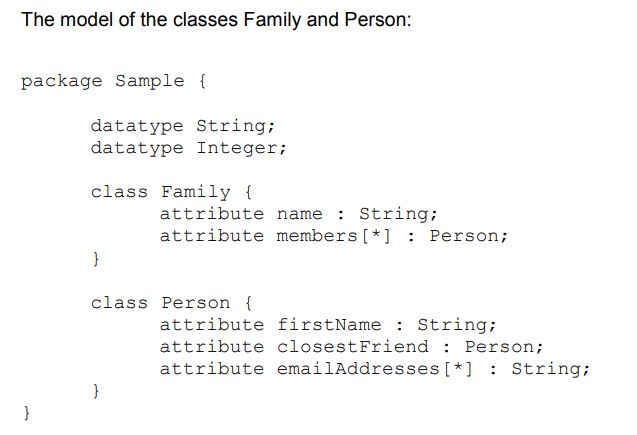
1. 对于每个Class类型的多值属性，创建一个Table

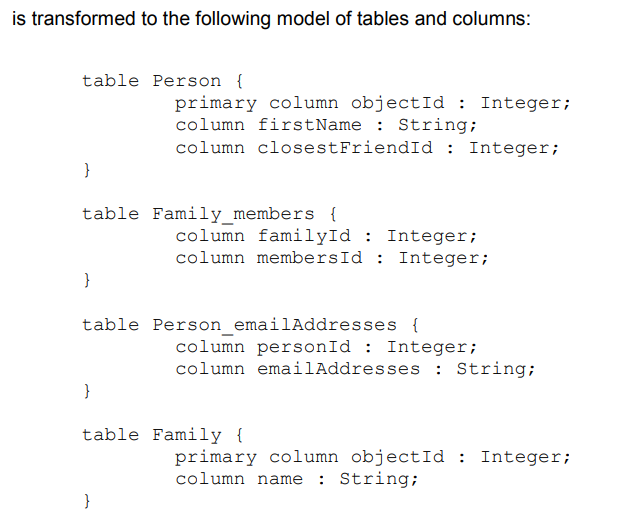




## 4.3 测试样例

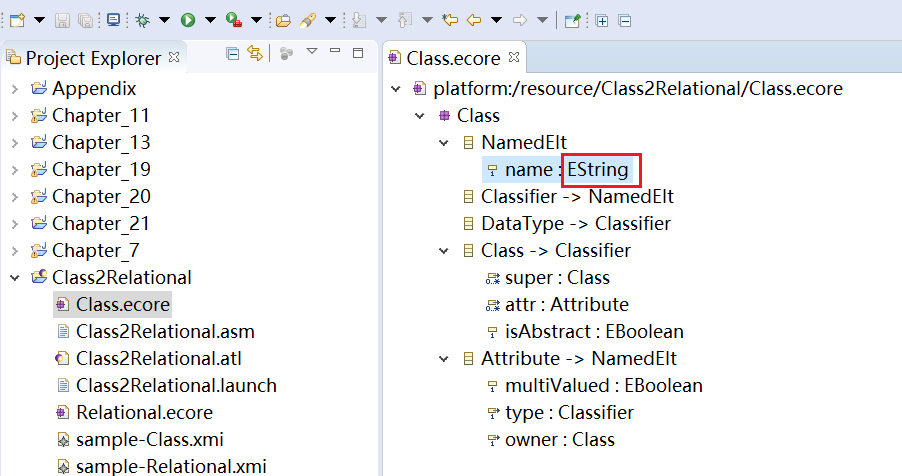
下面将显示的测试示例已使用KM3实现。它涉及带有类Family和类Person的package Sample，这些样本将转换为相应的relational模型。



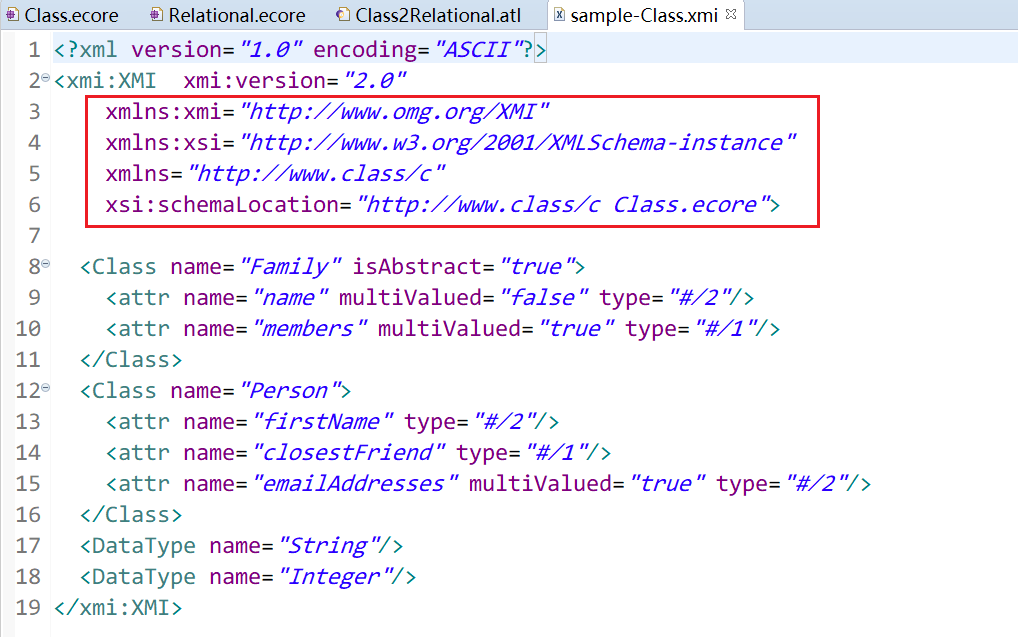


## 4.4 操作

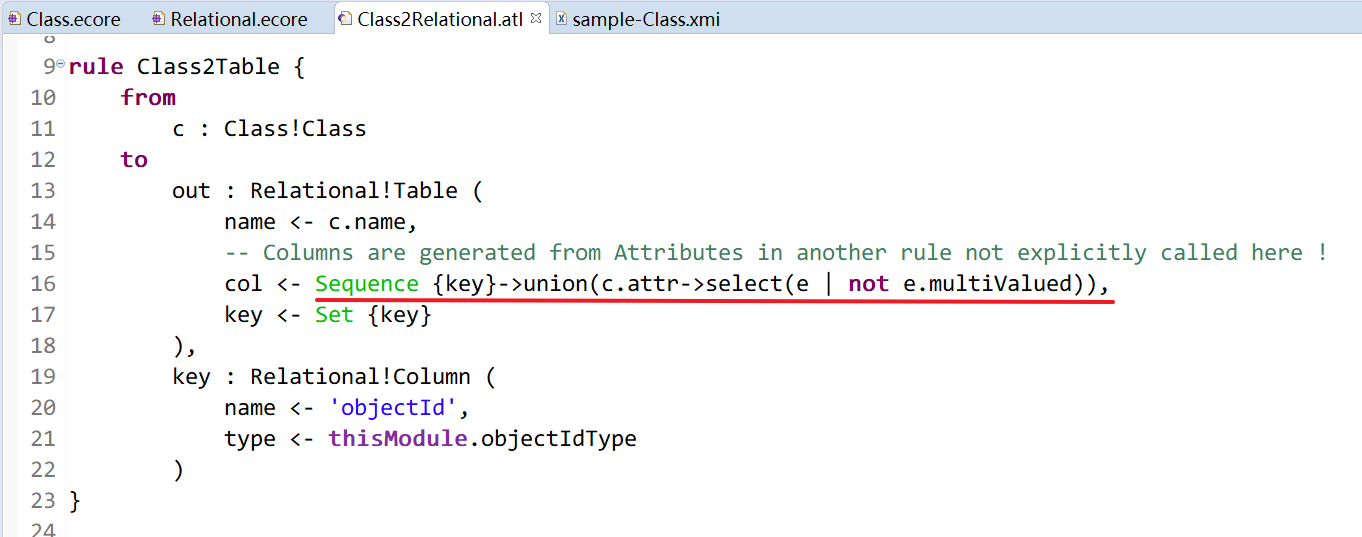
1. 数据类型选自带的



2. sample-Class.xmi头部更改后，可以用图形编辑器打开



1. Class变成Table时，col由key+单值属性组成



1. 笔记

