ECSE 439 Project Report - P439 6
Bank DSL to SQL

Connor Fowlie : 260687955

Grégoire Moreau : 260874685

1.0 Breakdown of Work

1.1 Conception and MetaModel Definition

This was done by both members of the project team. A software tool called LucidChart was used to allow for easy sharing and collaborative work on the UML diagram.

1.2 Project Creation for XText

The team used Pair Programming for the initial project creation and setup. Both members spent time individually editing the project remotely, to ensure the .xtext file accurately reflects the UML MetaModel.

1.3 Project for Xpand

Due to difficulties getting xpand to install correctly, Connor could not run XPand on his machine. Therefore the group met to work together to define the .xpt file to setup the SQL tables and their properties. With some help from the Prof. to convert our plaintext models to xml to be correctly parsed, Xpand began generating code correctly.

1.4 Report and Slides

The group met to work together on the report and slides, working in google docs to allow for easiest cooperation on the documents.

2.0 Language Definition

2.1 Chosen DSL

For our project we chose to create a new DSL of a simple banking system. Similar to some of the examples we have seen in class, this DSL allows us to demonstrate the functions of the transformation with some relatively simple examples.

The DSL describes things such as the type of bank, branches, employees and people working at the branch and so on down to which accounts are held by whom at each bank. The DSL also includes Users which can be extended to allow for access roles, although this is not implemented in the project.

2.2 Language Editor

You can run the language editor by opening the project called “project” in Eclipse and running it as an Eclipse Application. This will open a new runtime Eclipse where you should create a new project of type “General”. In that project, create a new file and use “.bank” as its extension. Eclipse will ask you whether you want to convert your new project to an Xtext project. Click yes. This will open a new file in which Eclipse will check if what you type corresponds to the syntax defined in BankSystem.xtext. If you want, you can also open one of the provided sample models example1.bank or example2.bank in the runtime Eclipse Application. Those files can be found in the folder sample_models inside the “project” folder.

2.3 List of Model and Edited Files

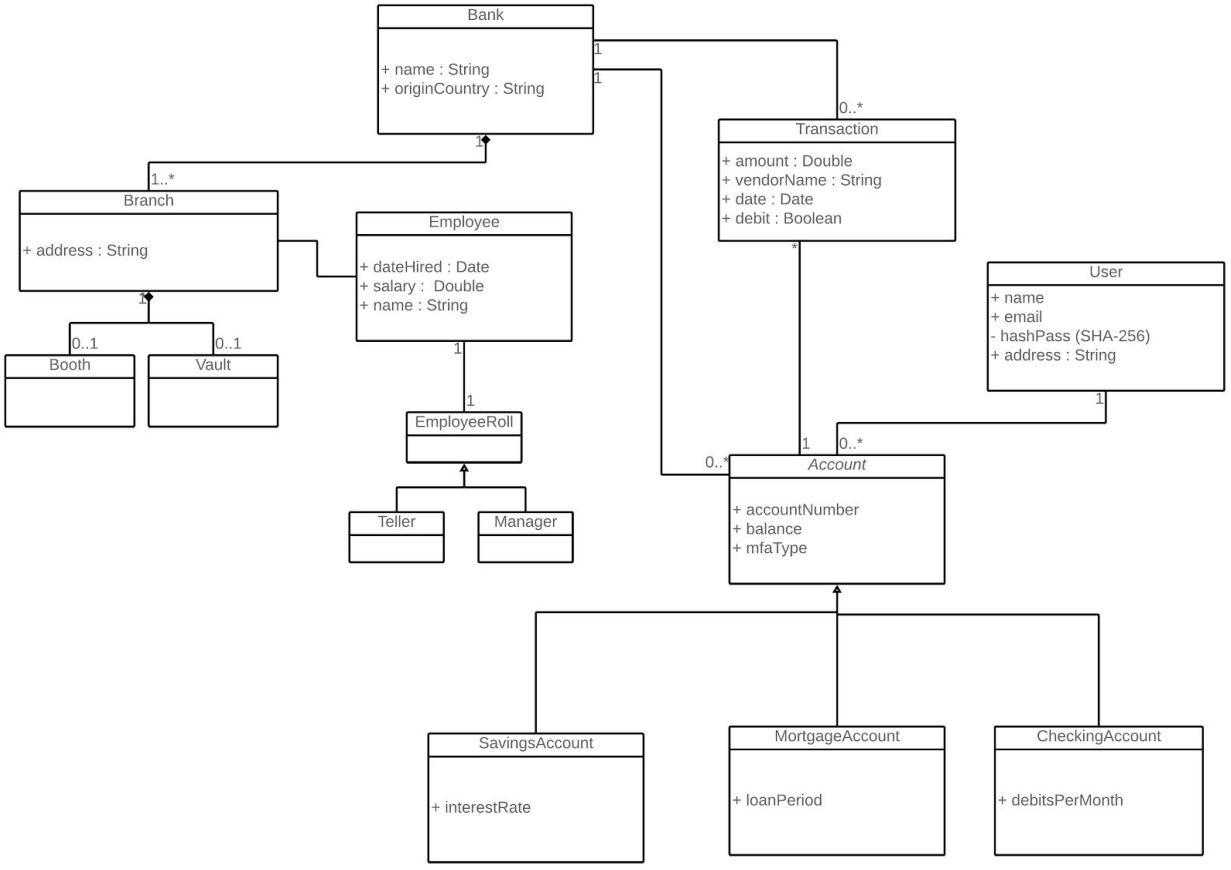
1. Xtext definition file : BankSystem.xtext
2. Xtext validation file : BankSystemValidator.xtend
3. Example Model 1 (Small) : example1.bank
4. Example Model 2 (Large) : example2.bank

2.4 Description of MetaModel

2.4.1 Design Choices

When contemplating this design, a few choices were made with our target language in mind. Specifically since SQL is relational tables, we purposefully excluded any many-to-many relationships to ensure we did not have to generate join tables.

2.4.2 UML Diagram



3.0 Transformation

3.1 Target Language

SQL is our target language. More specifically, a SQL Query (.sql) file that can be executed to set up and create the database, then import the corresponding data from the model.

As this is a model to text transformation, we tested the output SQL files to ensure the database was set up correctly. This was tested using MySQL with XAMPP v3.2.2

3.2 Source to Target Mapping

3.2.1 Database Setup

We first create the database and corresponding tables to allow a transformation into a MySQL database. The system is described by the Database itself, where each table

corresponds to a class in the metamodel. Relationships are mapped via table ids using foreign keys (See section 3.2.3). Due to the design choices of the metal model excluding many-to-many relationships, no joint key tables had to be generated.

3.2.2 Data Import

A number of insert calls are then generated to allow for the data from the model to be imported to the newly created tables. This is done for each piece of data within the model.

3.2.3 Foreign Keys

Additional selections are done to during creation of some inserts to allow for previously inserted rows ids to be used as foreign keys as relationships. For example, selecting the id from the user table to create a relationship to accounts.

3.3 Running with Xpand

To convert your .bank files into SQL instructions, you will first need to convert them to XML files so that Xpand can properly parse them. To do that, change the value of the String “inputURI” in the file project/src/org/xtext/ExportXMI.java and change “outputURI” to your desired output file path, then run that Java file. Next, open xpandtranslation/src/workflow/generator.mwe, and change the value of the property called “model” to the path of your XML file. In the file Template.xpt, on line 7, change “database.sql” to whatever filename you want for your SQL output. After that, run generator.mwe as a MWE workflow. This will generate a new SQL file in the src-gen folder.

3.4 List of Model and Edited Files

5. Xpand file : Template.xpt
6. Example Model 1 (Small) : example1.xmi
7. Example Model 2 (Large : example2.xmi

4.0 Reflection

4.1 What went well

With some forward thinking when designing our metamodel with our target language in mind, our SQL database structure is fairly simple and allows for easily model to text translations without the need for multiple relationship / join tables.

The language used to define the template that Xpand uses for the model-to-text transformation is easily understandable and very similar to the one used for Acceleo. This made our template file easy to write.

4.2 What did not go well

Throughout the project, the group encountered a number of difficulties working with Eclipse plugin Xpand. One member of the group was completely unable to get Xpand to correctly install, despite Xtext and other plugins functioning correctly. This led to group meet ups and working in person on a single device.

The format of our model file was also a pain point with this project. Originally described in plain text to conform to the metamodel, Xpand could not parse the model file. This was resolved after a discussion with the Prof who provided a code snippet to convert our plain text to XML. This allowed Xpand to correctly parse the model files and generate SQL Queries accordingly.