**Travail de Bachelor**

Slyum Relationnel

Non confidentiel



|  |  |
| --- | --- |
| **Étudiant :** | **Yoann Rohrbasser** |
| **Enseignant responsable :** | **Pier Donini** |
| **Année académique :** | 2019-2020 |
|  |  |

Yverdon-les-Bains, le 31 juillet 2018

Département TIC

Filière Informatique

Orientation Choisissez un élément.

Étudiant Yoann Rohrbasser

Enseignant responsable Pier Donini

Travail de Bachelor 2019-2020

Slyum Relationnel

Nom de l’entreprise/institution

**Résumé publiable**

Cliquez ou appuyez ici pour entrer du texte.

|  |  |  |
| --- | --- | --- |
| Étudiant :  Nom prénom | Date et lieu :  …………………………………… | Signature :  …………………………………… |
| Enseignant responsable :  Nom prénom | Date et lieu :  …………………………………… | Signature :  …………………………………… |
| Nom de l’entreprise/institution :  Nom prénom de la personne confiant l’étude | Date et lieu :  …………………………………… | Signature :  …………………………………… |

**Préambule**

Ce travail de Bachelor (ci-après TB) est réalisé en fin de cursus d’études, en vue de l’obtention du titre de Bachelor of Science HES-SO en Ingénierie / Economie d’entreprise.

En tant que travail académique, son contenu, sans préjuger de sa valeur, n'engage ni la responsabilité de l'auteur, ni celles du jury du travail de Bachelor et de l'Ecole.

Toute utilisation, même partielle, de ce TB doit être faite dans le respect du droit d’auteur.

HEIG-VD

Le Chef du Département

Yverdon-les-Bains, le 31 juillet 2020

**Authentification**

Le soussigné, Yoann Rohrbasser, atteste par la présente avoir réalisé seul ce travail et n’avoir utilisé aucune autre source que celles expressément mentionnées.

Aigle, le 31 juillet 2020

Yoann Rohrbasser

**Table des matières**

Contents

[1 SLYUM 7](#_Toc46755149)

[1.1 Unified Modeling Language (UML) 7](#_Toc46755150)

[1.2 Modèle relationnel 7](#_Toc46755151)

[1.3 Schéma relationnel 7](#_Toc46755152)

[2 Cahier des Charges 8](#_Toc46755153)

[2.1 Résumé du projet Donné par l’enseignant 8](#_Toc46755154)

[2.2 Etapes du projet 8](#_Toc46755155)

[2.3 Création et visualisation du schéma relationnel 8](#_Toc46755156)

[2.3.1 Fonctionnalités 8](#_Toc46755157)

[2.3.2 Etapes 8](#_Toc46755158)

[2.4 Conversion du schéma UML en relationnel 8](#_Toc46755159)

[2.4.1 Fonctionnalités 8](#_Toc46755160)

[2.4.2 Etapes 9](#_Toc46755161)

[2.5 Conversion du schéma relationnel en SQL 9](#_Toc46755162)

[2.5.1 Fonctionnalités 9](#_Toc46755163)

[2.5.2 SGBD visées 9](#_Toc46755164)

[2.5.3 Etapes 9](#_Toc46755165)

[2.6 Liaison entre relationnel et UML 9](#_Toc46755166)

[2.6.1 Fonctionnalités 9](#_Toc46755167)

[3 Modélisation 10](#_Toc46755168)

[3.1 Meta-Schéma Relationnel 10](#_Toc46755169)

[3.1.1 Schéma 10](#_Toc46755170)

[3.1.2 Table 10](#_Toc46755171)

[3.1.3 Clés Primaires/Etrangères/Alternatives 10](#_Toc46755172)

[3.1.4 Attributs/PrimitiveType 10](#_Toc46755173)

[3.1.5 Procédures Stockées/Vues/Triggers 10](#_Toc46755174)

[3.1.6 Procédures 10](#_Toc46755175)

[3.2 Diagramme de classe – entité 11](#_Toc46755176)

[3.2.1 RelationalEntity 11](#_Toc46755177)

[3.2.2 RelationalAttribute 11](#_Toc46755178)

[3.2.3 Key 11](#_Toc46755179)

[3.2.4 Trigger, TriggerTypes et ActivationTime 11](#_Toc46755180)

[3.2.5 Procédure 11](#_Toc46755181)

[3.2.6 StoredProcedure 12](#_Toc46755182)

[3.2.7 View 12](#_Toc46755183)

[3.3 Diagramme de classe - relation 12](#_Toc46755184)

[4 Règles de conversion 13](#_Toc46755185)

[4.1 Conversions basiques 13](#_Toc46755186)

[4.2 Conversions des cardinalités 13](#_Toc46755187)

[4.2.1 Conversion des associations binaires 1 : 1 13](#_Toc46755188)

[4.2.2 Conversion des associations binaires 1 : 0..1 13](#_Toc46755189)

[4.2.3 Conversion des associations binaires 0..1 : 0..1 13](#_Toc46755190)

[4.2.4 Conversion des associations binaires 1 : N 13](#_Toc46755191)

[4.2.5 Conversion des associations binaires 0..1 : N 13](#_Toc46755192)

[4.2.6 Conversion des associations binaires N : M 13](#_Toc46755193)

[4.3 Conversion des associations multiples 14](#_Toc46755194)

[4.4 Conversion des liens d’héritage 14](#_Toc46755195)

[5 Validation Schéma 15](#_Toc46755196)

[6 Mockup Interface 15](#_Toc46755197)

[7 Structure générale 17](#_Toc46755198)

[8 Création et visualisation du schéma relationnel 18](#_Toc46755199)

[8.1 Nouveaux éléments 19](#_Toc46755200)

[8.1.1 Tables 19](#_Toc46755201)

[8.1.2 Vues 19](#_Toc46755202)

[8.1.3 Relations 19](#_Toc46755203)

[8.2 Procédures des vues et des triggers 19](#_Toc46755204)

[8.3 Vues graphiques 19](#_Toc46755205)

[8.3.1 Vue du diagramme 19](#_Toc46755206)

[8.3.2 Vue hiérarchique 20](#_Toc46755207)

[8.3.3 Vue des propriétés 21](#_Toc46755208)

[8.3.4 Propriétés clés alternatives 22](#_Toc46755209)

[8.3.5 Procédures stockées 22](#_Toc46755210)

[8.3.6 Validateur 23](#_Toc46755211)

[8.3.7 Règles de validation 24](#_Toc46755212)

[8.4 Améliorations possibles 24](#_Toc46755213)

[9 Conversion du diagramme UML en relationnel 25](#_Toc46755214)

**Bibliographie**

<https://fr.wikipedia.org/wiki/Mod%C3%A8le_relationnel>

<https://www.javatpoint.com/java-swing>

<https://www.postgresql.org/docs/>

<https://github.com/HEIG-GAPS/slyum>

<https://www.mysql.com/products/workbench/>

<https://www.mysqltutorial.org>

<https://www.postgresqltutorial.com/>

<https://stackoverflow.com/>

<https://www.geeksforgeeks.org/detect-cycle-in-a-graph/>

# SLYUM

“ Slyum est un logiciel de construction de diagrammes de classe UML. Slyum rend la création de ces diagrammes simple et intuitif avec une interface intuitive, propre et facile à utiliser.

Beaucoup d’autres éditeurs de diagrammes existent dans les mondes industriels et open source. Mais beaucoup d’entre eux sont compliqué et pas plaisent à regarder. De plus ajouter des nouveaux éléments est difficile pour l’utilisateur. C’est pour ces raisons que nous avons développé ce projet. Cette application sera simple d’utilisation et seulement les éléments utiles seront intégrés. Le but de ce projet est qu’il puisse être utilisé pour l’apprentissage de l’UML. ”

Traduction non-officiel du préambule de la page GitHub de Slyum

## Unified Modeling Language (UML)

UML permet, en génie logiciel, de concevoir la structure d’un projet orienté objet sous la forme d’un diagramme.

Vu que UML n’est pas le sujet principal de ce projet je ne vais pas élaborer plus dessus mais un définition plus complète est disponible dans le rapport de l’ancien travail de bachelor Slyum à l’adresse suivante : <https://github.com/HEIG-GAPS/slyum/blob/master/doc/Slyum_Rapport.pdf>

## Modèle relationnel

“ Le modèle relationnel est une manière de modéliser les relations existantes entre plusieurs informations, et de les ordonner entre elles. Cette modélisation est souvent retranscrite physiquement (« implémentée ») dans une base de données.

On appelle « relation » un ensemble d'attributs qui caractérisent une proposition ou une combinaison de propositions comme "un employé a un matricule, il a un nom, il a un employeur". Dans cet exemple, les attributs de l'employé sont : son matricule, son nom et son employeur. Chaque combinaison de propositions ainsi formée est appelée uplet ou collection ordonnée d'objets. Par exemple l'ensemble ("1245", "Jean Dupond", "Compagnie des belles lettres") constitue un uplet de relation "employé". Les relations sont d'ordinaire représentées sous la forme de tables. Dans l'exemple précédent, la table serait libellée "employé". Usuellement, les praticiens accordent la même signification aux concepts de "relation" et de "table". De même, ils assimilent d'une part la "ligne dans la table" et l'uplet, et d'autre part le "libellé de colonne dans la table" et l'attribut. Par définition, chaque uplet d'une relation est unique. Il est identifié par un attribut (un identifiant unique appelé "clef primaire") ou une combinaison de plusieurs attributs qui forme la clef. L'ordre des uplets n'est pas significatif.

Le modèle relationnel est aujourd’hui l'un des modèles les plus utilisés. Le modèle relationnel est basé sur deux instruments puissants : l’algèbre relationnelle (c'est-à-dire le concept mathématique de relation en théorie des ensembles) et la notion de produit cartésien. Ce modèle définit une façon de représenter les données, les opérations qui peuvent être effectuées ainsi que les mécanismes pour préserver la consistance des données. ”

Abréviation de la définition du modèle relationnel selon wikipedia.org

## Schéma relationnel

Un schéma relationnel est une représentation visuelle du modèle relationnel similaire au schéma UML qui est utilisé pour conceptualiser une base de données relationnel.

# Cahier des Charges

## Résumé du projet Donné par l’enseignant

Lors d'un précédent travail de bachelor, un éditeur de diagrammes de classes UML a été développé, SLYUM. Ce logiciel, écrit en Java, permet la définition graphique de diagrammes de classes UML 1.4.

**Objectifs**

Le travail consistera à ajouter les fonctionnalités suivantes à l'application Slyum :

Permettre la traduction d'un schéma UML en schéma relationnel.

Permettre l'édition graphique de schémas relationnels.

Permettre la traduction d'un schéma relationnel en UML.

Permettre la génération de code SQL pour différents SGBD (PostgreSQL, MySQL)

Permettre l'évolution du schéma exprimé dans un langage (UML ou relationnel) tout en maintenant la cohérence de sa traduction (relationnel ou UML) et sans perte de ses modifications (p.ex. placement graphique d'une table).

## Etapes du projet

1. La création et visualisation du schéma relationnel.
2. La conversion du schéma UML en relationnel.
3. La conversion du schéma relationnel en SQL.
4. L’etablissement du lien entre UML et relationnel.

## Création et visualisation du schéma relationnel

### Fonctionnalités

* Créer des tables et définir une clé primaire.
* Définir les attributs de la table.
* Création d’une clé primaire, choix d’une clé, composite ou non parmi les attributs existants.
* Création des relations entre les tables et ajout de la clé étrangère.
* Création de clés alternatives.
* Validation du schéma relationnel et indication les erreurs.
* L’éditeur de schéma relationnel gardera toutes les fonctionnalités pertinentes de l’éditeur de schéma UML.

### Etapes

1. Pouvoir sélectionner si on crée un UML ou un REL quand on crée un nouveau schéma
2. Désactiver la UI UML-only et activer la REL-only (même si elle ne fait rien pour le moment)
3. Vérifier que le placement des éléments visuels fonctionne toujours et les réparer sinon
4. Implémenter les classes des éléments du REL
5. Ajouter une UI pour définir/créer la clé primaire de la classe
6. Ajouter une UI pour afficher les clés étrangères
7. Implémenter le validateur REL et les éléments de UI associés

## Conversion du schéma UML en relationnel

### Fonctionnalités

* Générer un schéma relationnel par rapport au schéma UML affiché.
* Les informations nécessaires à la complétion du schéma relationnel sont demandées à l’utilisateur via un formulaire à compléter.

### Etapes

1. Créer l’algorithme de conversion
2. Ajouter la UI pour entrer les informations complémentaires

## Conversion du schéma relationnel en SQL

### Fonctionnalités

* Convertir le schéma relationnel en un script SQL parmi les options de SGBD implémentées.
* Si des options/informations supplémentaires sont nécessaires à la conversion elles seront demandées à la création.
* Le script SQL ne sera pas lié au schéma mais sera créé et exporté du coup tout changement du modèle demande la régénération du script.
* Avant la génération du script, vérifier que le schéma relationnel soit correct.

### SGBD visées

* MySQL
* PostgreSQL
* SQLite

### Etapes

1. Créer l’algorithme de conversion
2. Ajouter une UI pour choisir la SGBD visé

## Liaison entre relationnel et UML

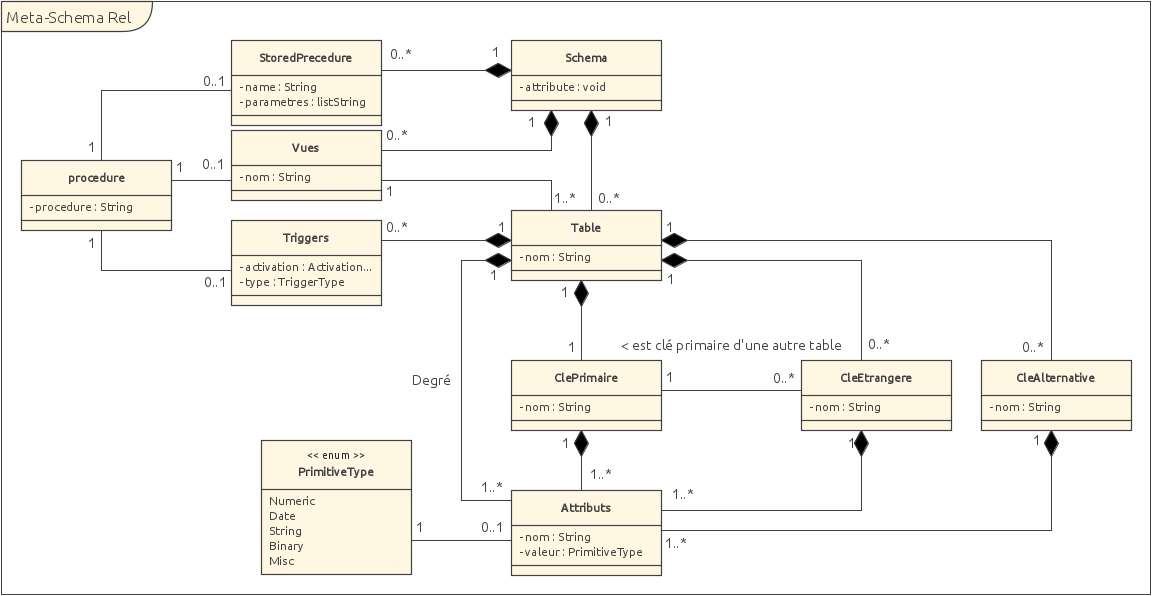
### Fonctionnalités

* Quand un schéma UML ou relationnel est généré par rapport à un autre schéma celui-ci est rajouté dans le fichier de sauvegarde.
* Tout changement dans un des diagrammes (ajout/suppression/modification de classes/liens) devra se répercuter dans l’autre schéma.
* Si un changement dans le schéma UML casse le schéma relationnel (ex : suppression de la clé primaire) l’utilisateur est averti avant et dois confirmer d’effectuer le changement.

# Modélisation

## Meta-Schéma Relationnel

Le meta-schéma sert à représenter les éléments du schéma relationnel qui seront présent dans Slyum Relationnel.



### Schéma

Le schéma est une classe englobante qui sert à différentier les schémas relationnels au sein du logiciel.

### Table

La table est l’élément basique d’une base de données relationnel.

### Clés Primaires/Etrangères/Alternatives

Les différentes clés des tables doivent être uniques et sont composées d’un ou plusieurs attributs.

### Attributs/PrimitiveType

Les attributs sont les différentes colonnes de la table et PrmitiveType leur type de donnée.

### Procédures Stockées/Vues/Triggers

Ces éléments ne font pas partie du modèle relationnel strict mais sont présent dans beaucoup de SGBD et sont intéressant à inclure pour la génération du script SQL.

### Procédures

Les procédures génériques sont, pour le moment, des strings qui correspondent aux procédures/requêtes associées aux triggers, vues et procédures stockées. Pour le moment ce sont de simples strings mais elles pourraient, si le temps le permet, être améliorés en un(des) objet(s) plus complexe(s) qui pourrai(ent) stocker la procédure indépendamment de la SGBD visé.

## Diagramme de classe – entité

Ce diagramme contient tous les éléments du meta-schéma sauf la relation entre clé primaire et clé étrangère.

Les classes en beige existent déjà et celles en rouge sont celles que je vais rajouter.

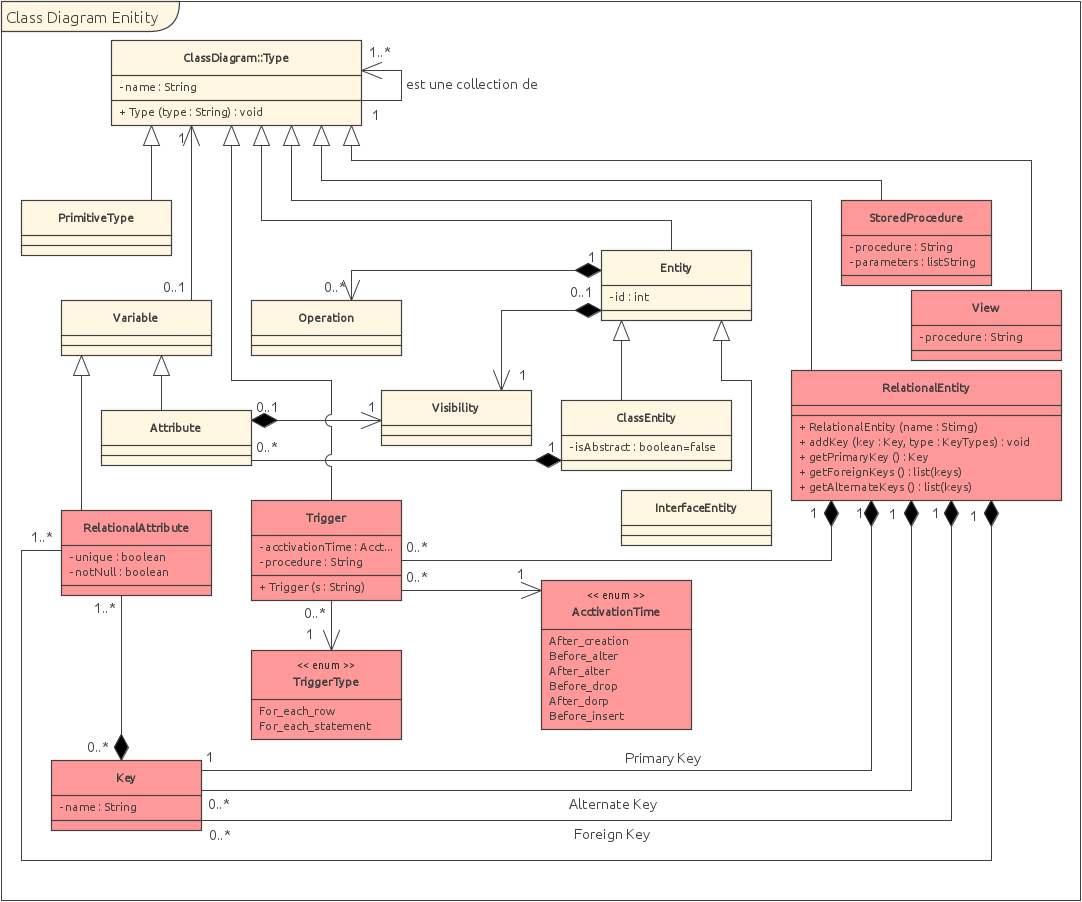


Diagramme de classe proposé au moment de la conception

### RelationalEntity

RelationalEntity est l’objet correspondant à la table du meta-schéma.

### RelationalAttribute

RelationalAttribute est l’objet correspondant à l’attribut du meta-schéma.

### Key

Key correspond au 3 types de clés du meta-schéma qui seront stockées dans RelationalEntity.

### Trigger, TriggerTypes et ActivationTime

Trigger correspond au Trigger du meta-schéma. TriggerTypes et ActivationTime sont des propriétés du trigger.

### Procédure

L’élément procédure du meta-schéma est représenté par l’attribut “procédure” dans les classes Trigger, StoredProcedure et View.

### StoredProcedure

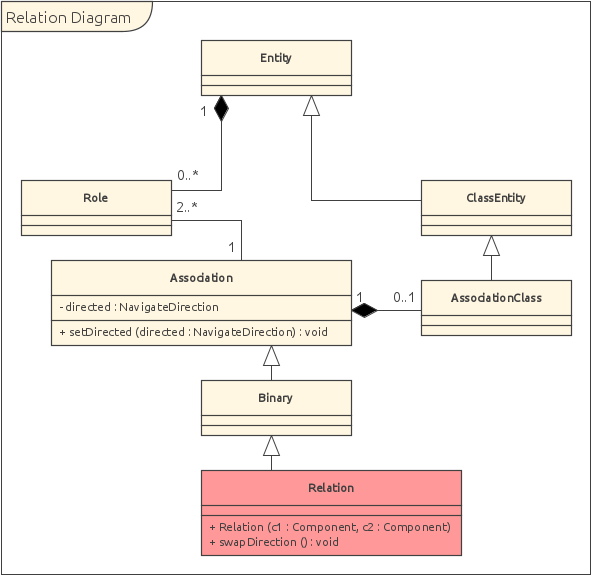
StoredProcedure correspondent à procédure stockée du meta-schéma.

### View

View correspond aux vues du méta-schéma.

## Diagramme de classe - relation

Ce diagramme montre comment la relation entre les tables est défini.



La classe relation est le seul moyen de créer des clés étrangères dans les tables. Quand une relation est créée, la table d’origine du lien est définie comme clé primaire et la table de destination comme clé étrangère. La clé primaire de la table de destination est ajoutée à la table d’origine comme clé étrangère. La méthode swapDirrection permet de changer la “direction” du lien et ainsi changer quelle table recevra la clé étrangère.

# Règles de conversion

## Conversions basiques

* Les classes, interfaces et classes d’association deviennent des **Tables.**
* Les attributs deviennent des **Attributs Relationnels**
* **Les opérations sont négligées à la conversion car elles n’ont pas d’équivalence directe en relationnel**
* Associations simples, agrégations et compositions deviennent des **Relations.**

Une clé par défaut appelé “ID” est ajoutée aux tables converties et dois être peuplée manuellement par l’utilisateur.

## Conversions des cardinalités

### Conversion des associations binaires 1 : 1

* On rajoute la clé étrangère dans la table du premier coté (cela n’a pas vraiment d’importance)
* La clé doit être **unique et non null.**

### Conversion des associations binaires 1 : 0..1

La conversion est similaire à la 1:1 à l’exception que la clé étrangère dois être dans la table qui est de cardinalité 1.

### Conversion des associations binaires 0..1 : 0..1

La conversion est similaire à la 1:1 à l’exception que la clé est unique est **nullable**.

### Conversion des associations binaires 1 : N

* On rajoute la clé étrangère dans la table qui a **1** comme cardinalité.
* La clé doit être **non null**.

### Conversion des associations binaires 0..1 : N

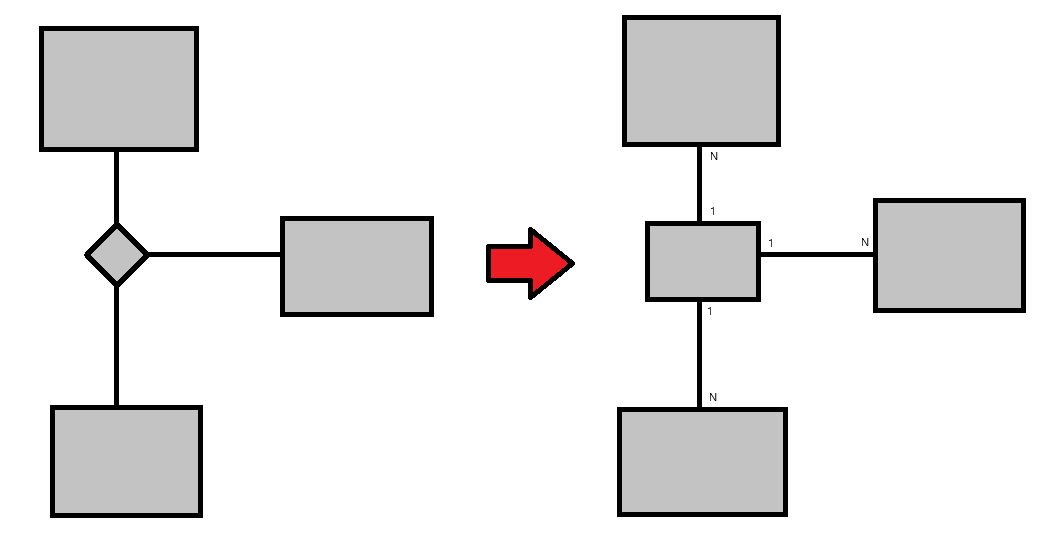
La conversion est similaire à 1:N à l’exception que la clé est **nullable**.

### Conversion des associations binaires N : M

La conversion de d’une association N:M ce fait en créant une table entre les 2 deux. Cette table prendra les clés étrangères des 2 tables de l’association et utilisera leurs composants comme clé primaire.

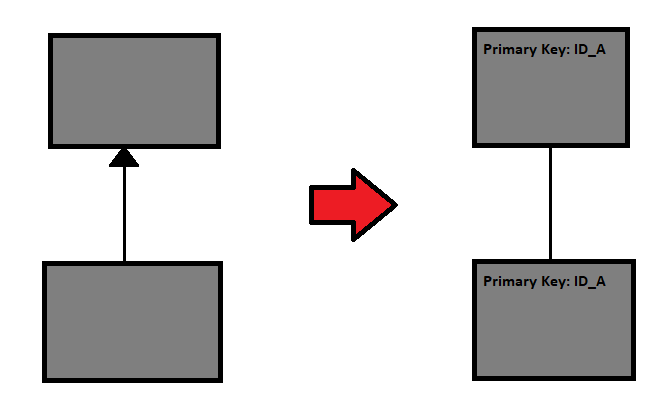
## Conversion des associations multiples

Une association multiple est convertie en une table liée aux autres tables de la relation via des relations 1-N avec le N du côté des tables.



## Conversion des liens d’héritage

Les liens d’**héritage** sont convertis en **relation** avec la sous-classe ayant la même clé que la classe parent.

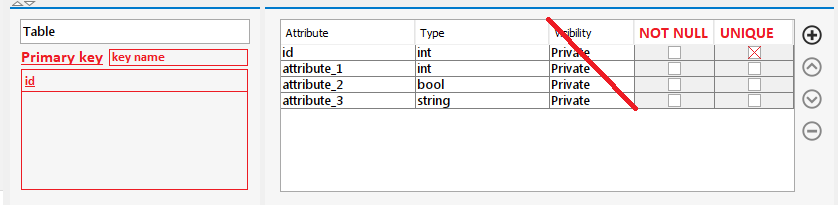


# Validation Schéma

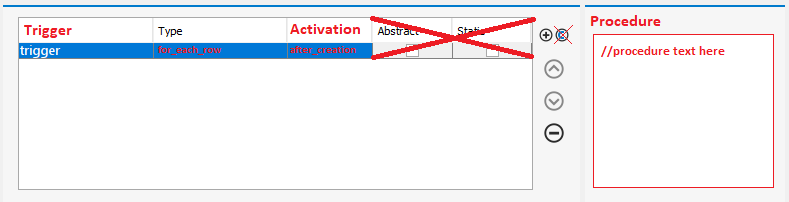
Un validateur du schéma relationnel sera lancé automatiquement à la conversion en relationnel et avant la conversion en SQL. La validation peut aussi être lancée manuellement via l’interface. Les résultats de cette analyse peuvent se trouver sur l’interface de projet (voir plus bas).

# Mockup Interface

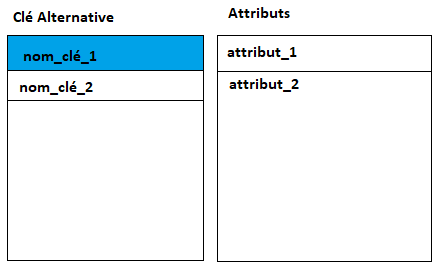
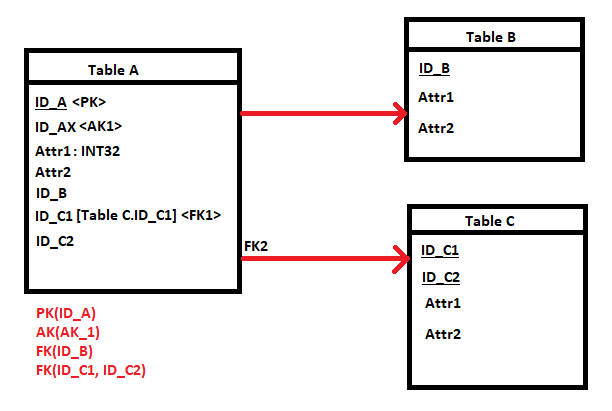
Mockup interface tables, attributs et clés primaire



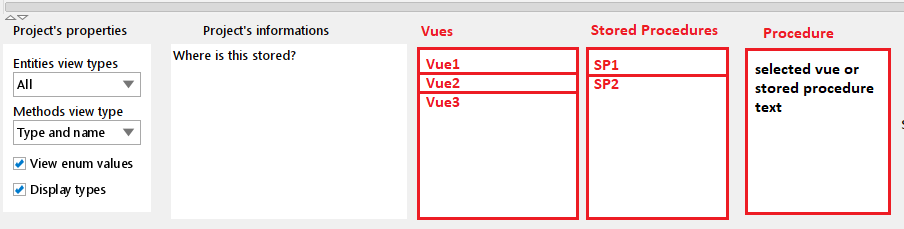
Mockup interface triggers



Mockup interface clés alternatives Mockup représentations des clés et attributs



Mockup interface de projet (vue et procédures stockées).



La partie de l’interface dédiée aux éléments de la classe est adapté pour fonctionner avec les tables relationnelles.

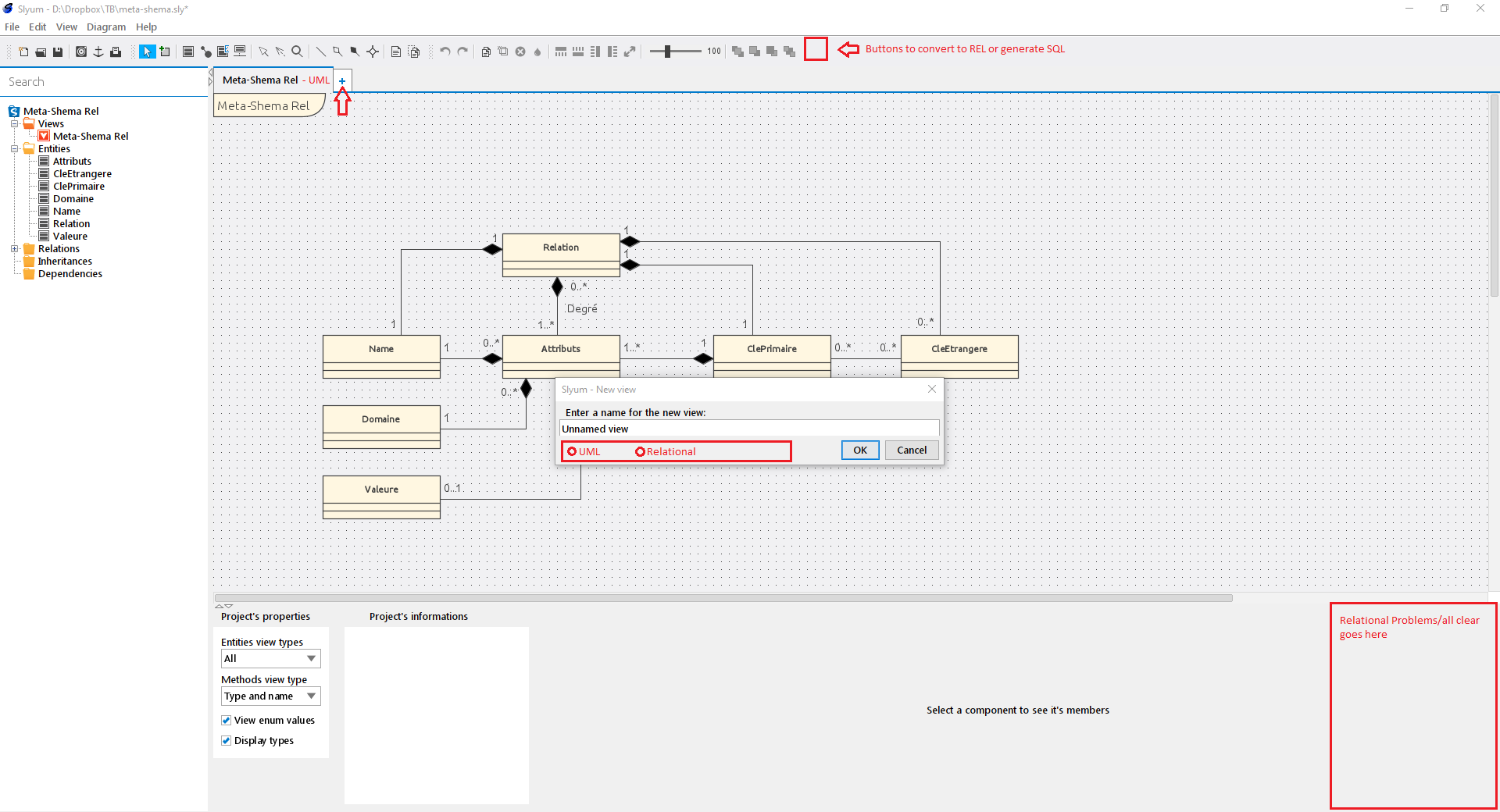
Une table pour définir une clé primaire est rajoutée. Cette table peut contenir un ou plusieurs attributs. A la création d’une classe, un attribut “id” est créé pour créer une clé par défaut.

La partie contenant les méthodes de classes est changée pour pouvoir stocker des triggers. La partie montrant les paramètres de la méthode montre maintenant une zone de texte pour la procédure.

Une troisième partie est rajoutée pour définir les clés alternatives et ses attributs.

Si aucune table n’est sélectionnée, on voit l’interface du projet avec les vues et les procédures stockées à côté.

Mockup interface création de vue “uml” ou “relationnel”



Le choix de uml ou relationnel quand on crée une nouvelle vue ne change pas les classes utilisées mais détermines les fonctions utilisables sur la GUI (ex : on peut créer des liens d’héritage en uml mais pas en relationnel)

Je profite aussi de cette vue globale pour montrer où se situent les résultats de la validation relationnel.

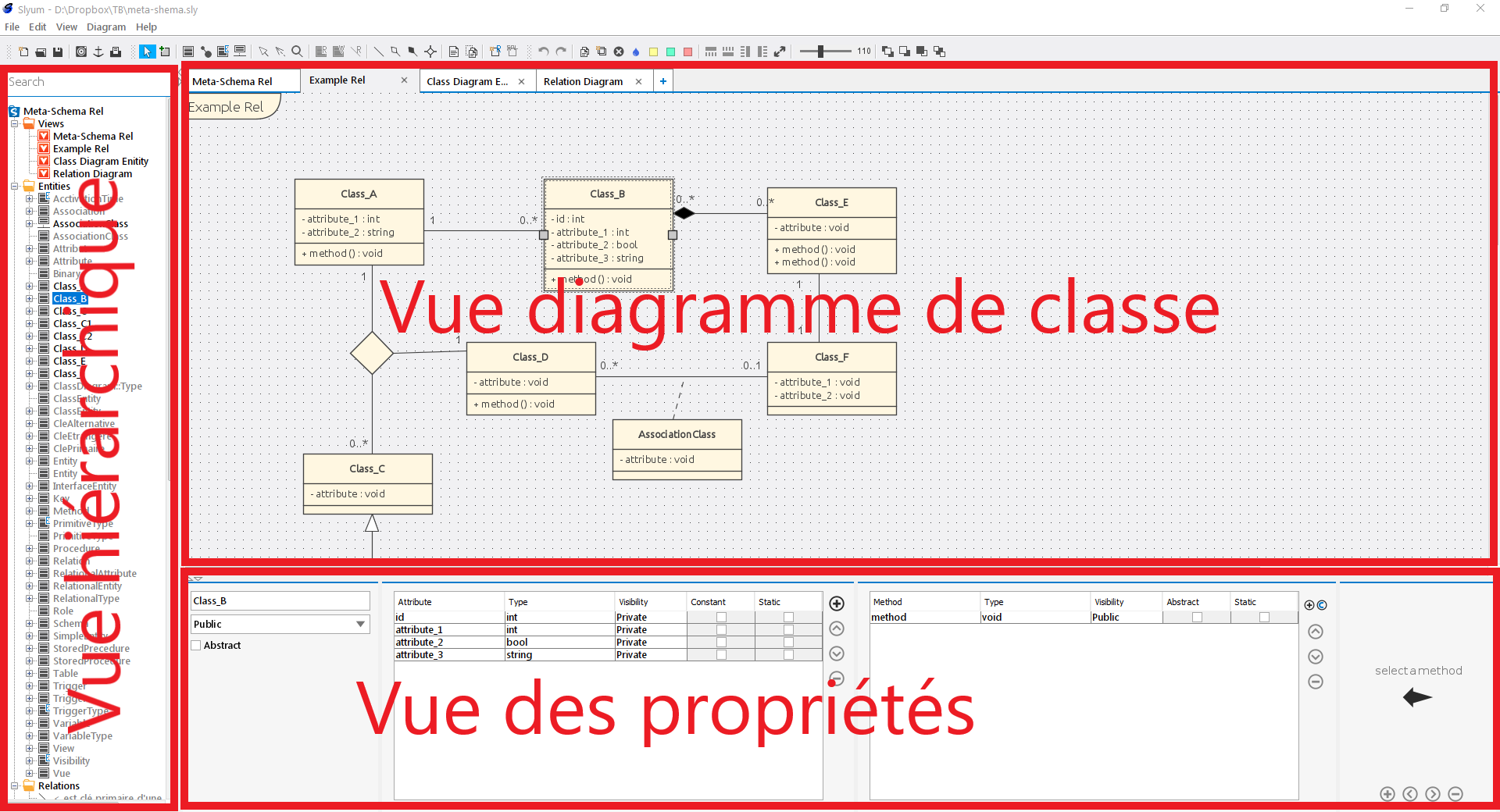
# Structure générale

Le projet original de Slyum est séparé en 4 parties.

La partie principale est la structure des classes qui contient les toutes les entités objet des schémas (classes, enums, attributs, associations, etc.) auquel j’ai rajouté les entités relationnelles. Cette partie permet de stocker tous les composants d’un diagramme de classe (et un schéma relationnel maintenant).

Les trois autres parties de Slyum sont des composants de l’interface graphique de Slyum.

* La “Vue diagramme de classe” dans laquelle est affichée la représentation graphique du schéma.
* La “Vue hiérarchique” dans laquelle est stockée une représentation en arbre de tous les éléments du schéma.
* La “Vue des propriétés” dans laquelle est affichée les propriétés de l’élément graphique sélectionné.



Ces trois parties sont reliées à la structure principale avec patron de conception “observer” qui est implémenté dans Slyum avec l’api Observer-Observable.

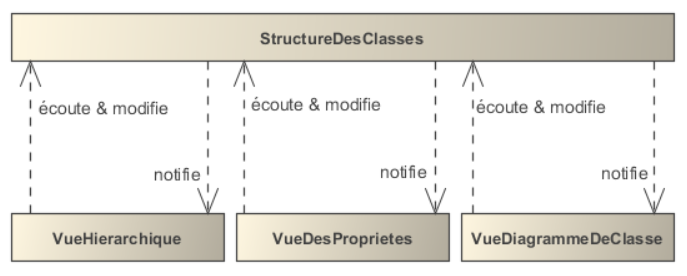
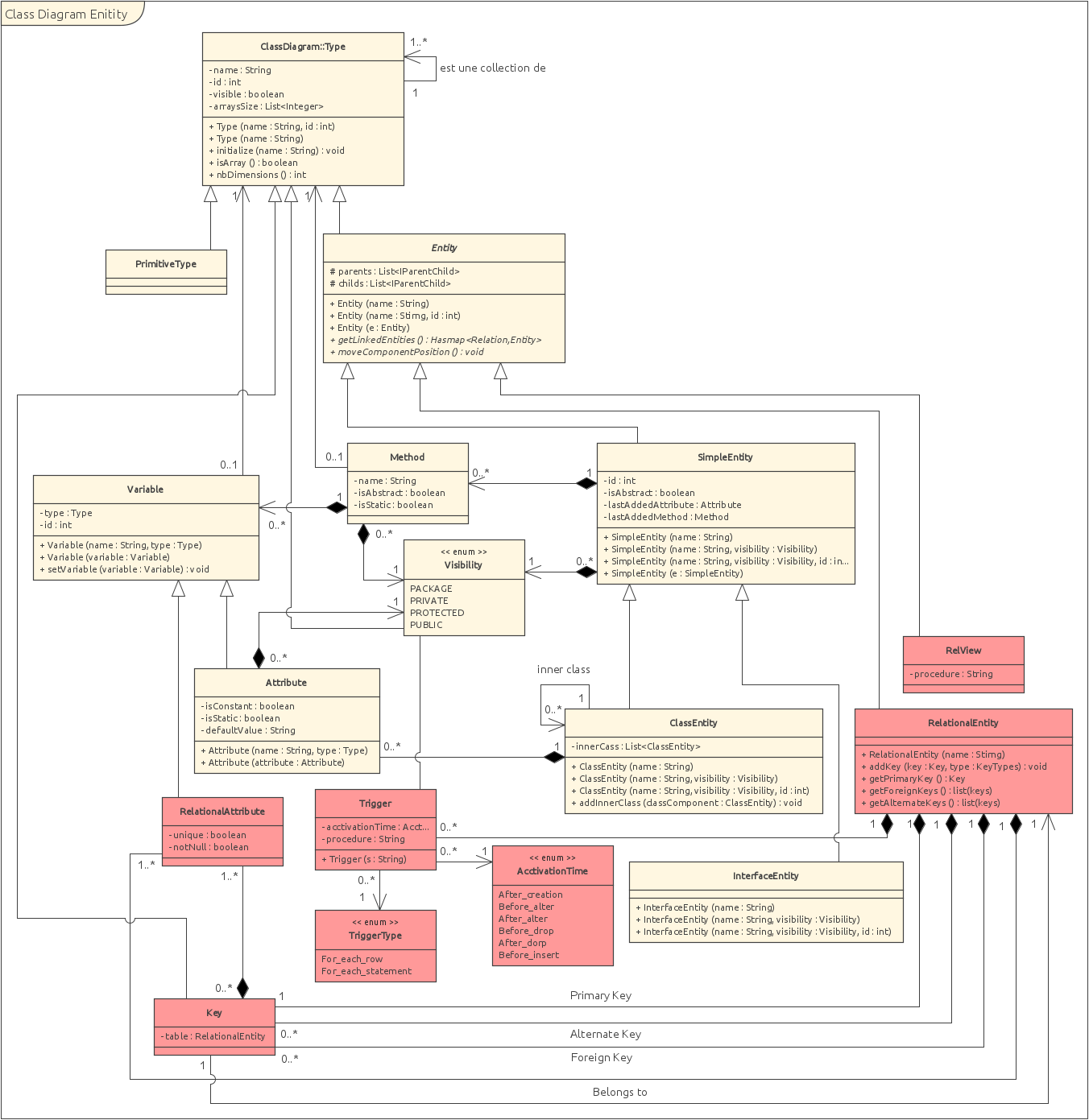


Illustration de la structure générale du projet du rapport Slyum de 2011 par David Miserez

# Création et visualisation du schéma relationnel

Certaines modifications ont pu être faites sur le diagramme original vu que le code de Slyum contenait une classe abstraite “simpleEntity” qui s’interpose entre la classe d’entité générique et les spécialisations UML (Classe, Enum, etc.) et qui contiens des éléments du diagramme UML non-nécessaires au schéma relationnel. J’ai pu du coup faire hériter mes composants relationnels à l’entité générique (Entity) et du coup réutiliser une plus grande partie du code.



Pour rappel tous les objets héritent de Type et peuvent donc être représenté dans un même schéma. Ça sera le rôle de l’interface de faire en sorte que l’utilisateur ne mette pas des éléments uml et rel dans le même diagramme.

## Nouveaux éléments

### Tables

Une table a un nom, une clé primaire, une liste d’attributs, une liste de clé alternatives, une liste de clés étrangères et une liste de triggers.

### Vues

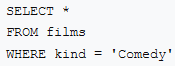
Une vue a un nom et une procédure stockée sous la forme d’un string.

### Relations

Une relation a les mêmes attributs qu’une association binaire, c’est à dire un nom, une entité source, une entité cible et une direction.

## Procédures des vues et des triggers

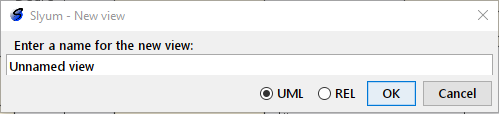
Dans cette version de Slyum, les vues relationnels et les triggers stockent leur procédure respective sous la forme d’une chaine de charactères. Cela veut dire que si l’utilisateur veut utiliser la fonction de génération de scripte SQL il doit connaitre à l’avance quel SGBD il vise et écrire la procédure manuellement.

Exemple de procédure Postgres de https://www.postgresql.org/docs/9.2/sql-createview.html

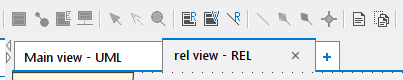
## Vues graphiques

### Vue du diagramme

A la création d’un nouveau diagramme, le dialogue de création propose de choisir entre créer une vue relationnelle et une vue UML.



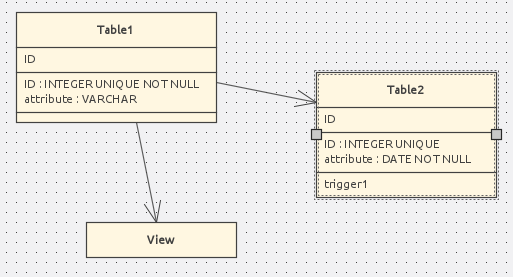
Quand une vue relationnelle est créée, les éléments d’interface pour ajouter des entités de diagramme UML sont désactivé et ceux des entités relationnelles sont activées.



De plus, quand une nouvelle vue graphique (un onglet) est créée, un identifiant (-UML ou -REL) est rajouté à l’onglet pour identifier son type. Celui-ci peut fait partie du nom de la vue et peut du coup être enlevé si l’utilisateur désire.

Les entités instanciable sont dans l’ordre : les tables, les vues et les liens entre ces entités.

Les tables montrent dans l’ordre : les clés, les attributs et les triggers. Vu que le seul composant de la vue relationnel est ça procédure seul son nom est affiché dans la vue du diagramme. Les flèches indiquent le sens de la relation. La clé étrangère est placée du côté de la source. La liaison entre table et vue est purement visuelle.



### Vue hiérarchique

La vue hiérarchique montre tous les éléments de toutes les vues graphiques. Ceux qui ne sont pas dans la vue active sont grisé.

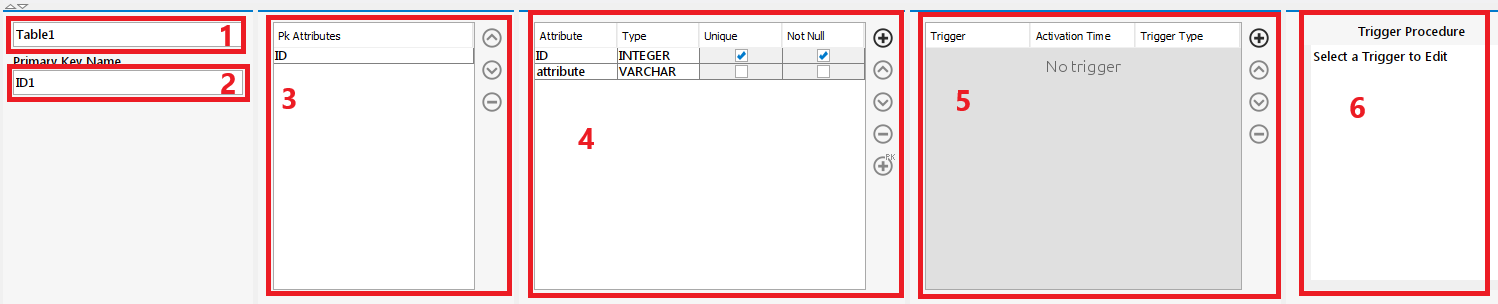
On peut aussi voir les attributs, triggers et clés des tables.

Le type de la clé est indiqué dans les crochets. PK pour clé primaire, AK pour clé alternative et FK pour clé étrangère. Si la clé est étrangère, ça table d’origine est aussi noté.

### Vue des propriétés

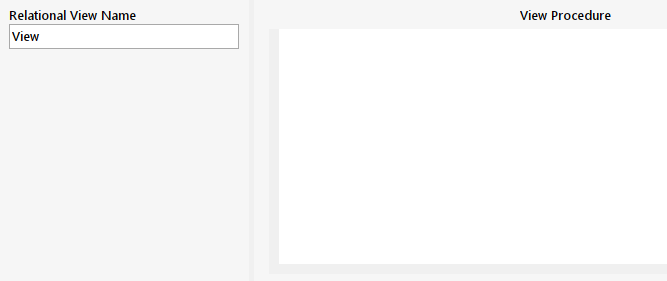
Quand un élément de la vue du diagramme est sélectionné la vue des propriétés affiche ces propriétés.

#### Propriétés des tables



1. Le nom de l’entité
2. Le nom de la clé primaire
3. Les attributs de la clé primaire
   * Les flèches peuvent réordonner les attributs.
   * Le moins supprime l’attribut sélectionné de la clé primaire.
4. Tous les attributs de la table
   * On peut rajouter un nouvel attribut avec le plus.
   * Les flèches peuvent réordonner les attributs.
   * Le moins supprime l’attribut sélectionné de la table.
   * Le plus du bas rajoute l’attribut sélectionné a la clé primaire
5. Les triggers de la table
   * On peut rajouter un nouveau trigger avec le plus.
   * Les flèches peuvent réordonner les triggers.
   * Le moins supprime le trigger sélectionné de la table.
6. La procédure du trigger sélectionné

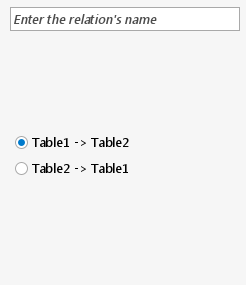
#### Propriétés des vues



Les propriétés des vues sont relativement simples. Le petit champ sur la gauche est le nom de la vue

Et le gros champ sur la droite est la procédure de cette vue.

#### Propriétés des relations

Les propriétés des relations sont simples aussi. Il y a le nom de la relation qui est optionnel et deux boutons radiaux qui indiquent le sens de la relation.

### Propriétés clés alternatives

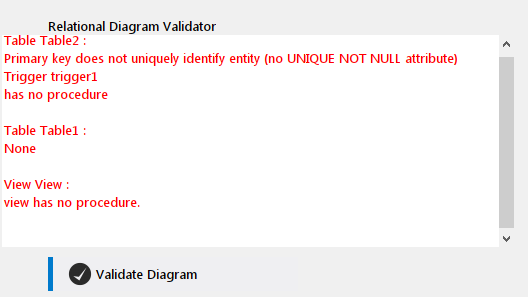
Les propriétés des tables étaient sensées contenir deux tableaux pour gérer les clés alternatives. Malheureusement, j’ai mal calculé la place que chaque élément prendra et je n’avais plus la place de les rajouter à l’interface et, au lieu de perdre du temps à refaire ma modélisation et trouver une façon ergonomique de rajouter cet élément d’interface, j’ai préféré mettre ça de côté et me concentrer sur les éléments du projet que j’ai jugé plus important.

Cela veut dire qu’un utilisateur ne peut pas rajouter des clés alternatives via l’interface. La seule façon de les rajouter est de les rajouter manuellement dans un fichier de sauvegarde et de charger ce fichier.

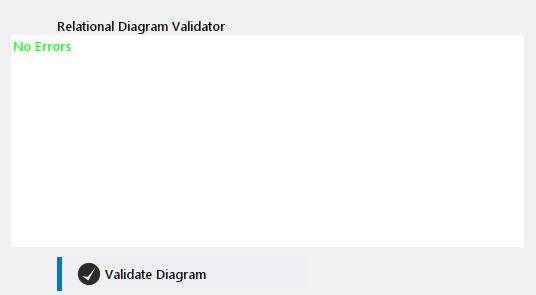
### Procédures stockées

### Validateur

Quand aucun élément n’est sélectionné l’utilisateur peut voir les options globales. J’y ai rajouté une section qui permet de trouver les erreurs dans un schéma relationnel. Quand l’utilisateur appuie sur le bouton un rapport des erreurs est généré et est affiché.



Example de rapport d’erreurs du validateur relationnel



Example de rapport d’erreur vide

Le validateur est aussi lacé quand l’utilisateur essaye de générer un script SQL pour ne pas avoir des erreurs vérifiables dans le scripte.

### Règles de validation

**Tables :**

* Une table doit avoir un nom unique (inclut les vues).
* Une table doit avoir une clé primaire et cette clé dois être valide.
* Une table doit avoir au moins 1 attribut.
* Chaque attribut doit avoir un nom unique.
* Chaque trigger doit être valide
* Chaque clé alternative doit être valide

**Clé :**

* Une clé doit avoir un nom.
* Une clé doit avoir au moins 1 attribut.
* Une clé doit pouvoir identifier sa table (elle doit contenir au moins un attribut unique)

**Triggers :**

* Un trigger doit avoir un nom unique.
* Un trigger doit avoir une procédure.

**Vues :**

* Une vue doit avoir un nom unique (inclut les tables).
* Une vue doit avoir une procédure.

**Cycles :**

Un schéma relationnel ne doit pas contenir de cycles. Je détecte les cycles en transformant le schéma en un graphe orienté puis j’utilise un algorithme de recherche de cycles DFS repris du site [www.geeksforgeeks.org](http://www.geeksforgeeks.org) pour détecter un cycle.

Ces règles permettent de garantir une justesse relative du schéma et du script SQL généré. En effet, quelques éléments ne sont pas vérifiés strictement et peuvent du coup encore contenir des erreurs (ex : le contenu de la procédure n’est pas vérifié ou les types des attributs ne sont pas garanti d’exister dans SQL.

## Améliorations possibles

Voici une liste non-exhaustive de potentielles améliorations qui pourraient être apporté au créateur de schéma relationnels avec plus de temps de développement.

* Rendre les procédures des vues et les triggers plus génériques de façon à ce qu’elles puissent être transformé en un scripte SQL adapté à la SGBD visé.
* Inclure les procédures stockées qui ont été mise de côté pour le développement de cette version de Slyum.
* Trouver une façon ergonomique d’inclure l’interface concernant les clés alternatives.
* Changer les types des attributs de façon à ne que pouvoir choisir entre des types d’attributs défini ou alors changer la validation pour détecter si le type est valide.
* Améliorer le validateur relationnel de façon à pouvoir valider les modifications proposées ci-dessus mais aussi rajouter des éléments de validation liés à des SGBD spécifiques qui seraient lancés au moment de la génération du script SQL.

# Conversion du diagramme UML en relationnel

Quand une vue(onglet) UML est selectionné l’utilisateur peut convertir le diagramme UML en schéma relationnel en utilisant les règles définie plus haut dans ce rapport (section 4).

Voici un example de conversion :