

# Travail de Bachelor

## Rendu Intermédiaire

### Slyum Relationnel

Non confidentiel



<b>Étudiant :</b>	<b>Yoann Rohrbasser</b>
<b>Enseignant responsable :</b>	<b>Pier Donini</b>
<b>Année académique</b>	<b>2019-2020</b>

Yverdon-les-Bains, le 19 juin 2020

# Authentication

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 19 juin 2020

Yoann Rohrbasser

# Table des matières

1	SLYUM .....	6
2	Cahier des Charges.....	6
2.1	Résumé du projet Donné par l'enseignant.....	6
2.2	Etapas du projet .....	6
2.3	Création et visualisation du schéma relationnel .....	6
2.3.1	Fonctionnalités .....	6
2.3.2	Etapas .....	6
2.4	Conversion du schéma UML en relationnel.....	7
2.4.1	Fonctionnalités .....	7
2.4.2	Etapas .....	7
2.5	Conversion du schéma relationnel en SQL.....	7
2.5.1	Fonctionnalités .....	7
2.5.2	SGBD visées.....	7
2.5.3	Etapas .....	7
2.6	Liaison entre relationnel et UML .....	7
2.6.1	Fonctionnalités .....	7
3	Modélisation .....	8
3.1	Meta-Schéma Relationnel .....	8
3.1.1	Schéma .....	8
3.1.2	Table .....	8
3.1.3	Clés Primaires/Etrangères/Alternatives .....	8
3.1.4	Attributs/PrimitiveType .....	8
3.1.5	Procédures Stockées/Vues/Triggers.....	8
3.1.6	Procédures.....	8
3.2	Diagramme de classe – entité .....	9
3.2.1	RelationalEntity.....	9
3.2.2	RelationalAttribute .....	9
3.2.3	Key .....	9
3.2.4	Trigger, TriggerTypes et ActivationTime.....	9
3.2.5	Procédure .....	9
3.2.6	StoredProcedure.....	9
3.2.7	View .....	10
3.3	Diagramme de classe - relation .....	10
4	Règles de conversion .....	11
5	Validation Schéma .....	12

6	Interface.....	12
7	Ce qui a été fait.....	14
7.1	Les objets.....	14
7.2	Les éléments d'interface .....	15
8	Ce qu'il reste à faire .....	15
8.1	Les éléments d'interface .....	15
8.2	Conversion du schéma UML en schéma relationnel .....	15
8.3	Conversion du schéma relationnel en scripte SQL .....	15
9	Timetable .....	16
10	Annexes.....	17
10.1	Journal de travail .....	17
10.1.1	Semaine 1 – 17 au 23 Février .....	17
10.1.2	Semaine 2 – 24 Février au 1 Mars .....	17
10.1.3	Semaine 3 – 2 au 8 Mars.....	17
10.1.4	Semaine 6 – 23 au 29 mars.....	17
10.1.5	Semaine 7 – 30 Mars au 5 Avril .....	17
10.1.6	Semaine 8 – 6 au 12 Avril .....	17
10.1.7	Semaine 9 – 13 Au 19 Avril .....	17
10.1.8	Semaine 10 – 20 au 26 Avril .....	17
10.1.9	Semaine 11 – 27 Avril au 3 Mai .....	18
10.1.10	Semaine 12 - 4 au 10 Mai .....	18
10.1.11	Semaine 13 – 11 au 17 Mai.....	18
10.1.12	Semaine 14 – 18 au 24 Mai.....	18
10.1.13	Semaine 15 – 25 au 31 Mai.....	18
10.1.14	Semaine 16 – 1 au 7 Juin .....	18
10.1.15	Semaine 17 – 8 au 14 Juin .....	18
10.1.16	Semaine 18 – 15 au 21 Juin .....	18

# Bibliographie

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

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

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

<https://www.mysql.com/products/workbench/> (utilisé pour exemples d'interface)

# 1 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*

## 2 Cahier des Charges

### 2.1 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).

### 2.2 Etapes du projet

1. Création et visualisation du schéma relationnel
2. Conversion du schéma UML en relationnel
3. Conversion du schéma relationnel en SQL
4. Etablissement du lien entre UML et relationnel

### 2.3 Création et visualisation du schéma relationnel

#### 2.3.1 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 ou, par défaut, création d'un champ ID
- Création des relations entre les tables et choix 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

#### 2.3.2 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

## 2.4 Conversion du schéma UML en relationnel

### 2.4.1 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

### 2.4.2 Etapes

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

## 2.5 Conversion du schéma relationnel en SQL

### 2.5.1 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

### 2.5.2 SGBD visées

- MySQL
- PostgreSQL
- SQLite

### 2.5.3 Etapes

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

## 2.6 Liaison entre relationnel et UML

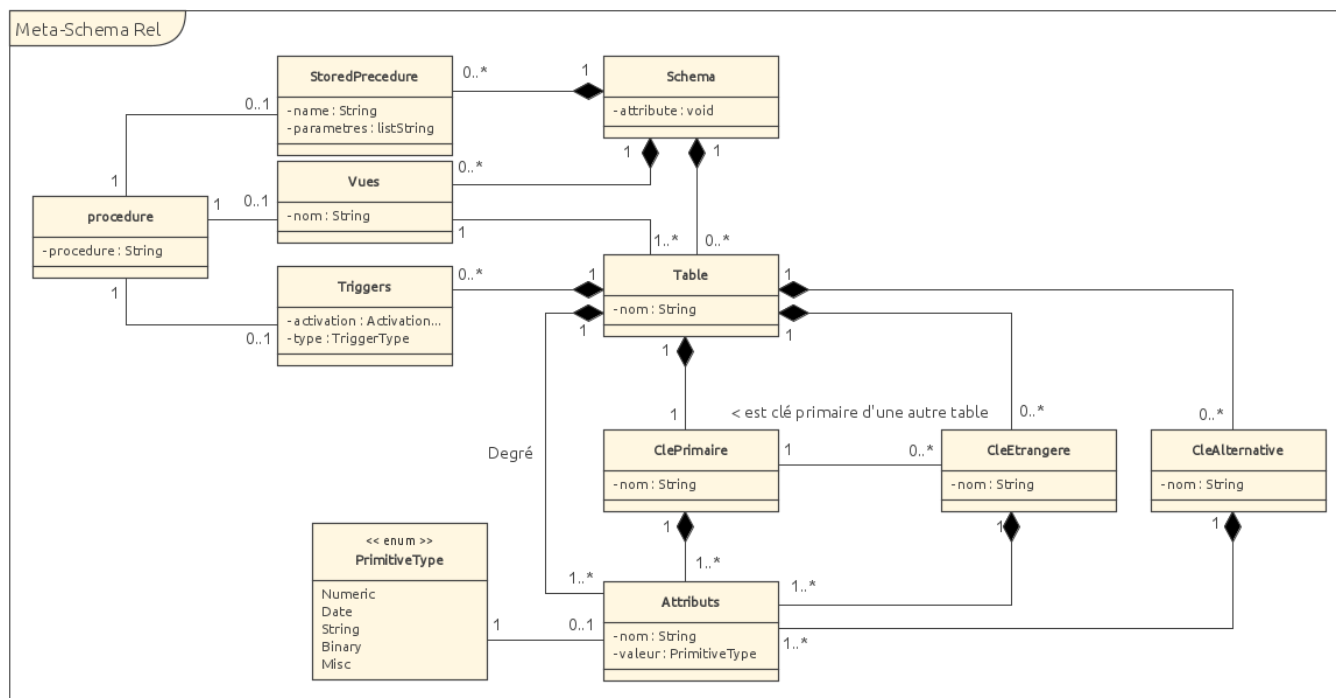
### 2.6.1 Fonctionnalités

- 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 doit confirmer d'effectuer le changement

## 3 Modélisation

### 3.1 Meta-Schéma Relationnel

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



#### 3.1.1 Schéma

Schéma est une classe englobante qui sert à différencier les schémas relationnels au sein du logiciel

#### 3.1.2 Table

Élément basique d'une base de données relationnel

#### 3.1.3 Clés Primaires/Etrangères/Alternatives

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

#### 3.1.4 Attributs/PrimitiveType

Les différentes colonnes de la table et leur type de donnée

#### 3.1.5 Procédures Stockées/Vues/Triggers

Éléments du script qui 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

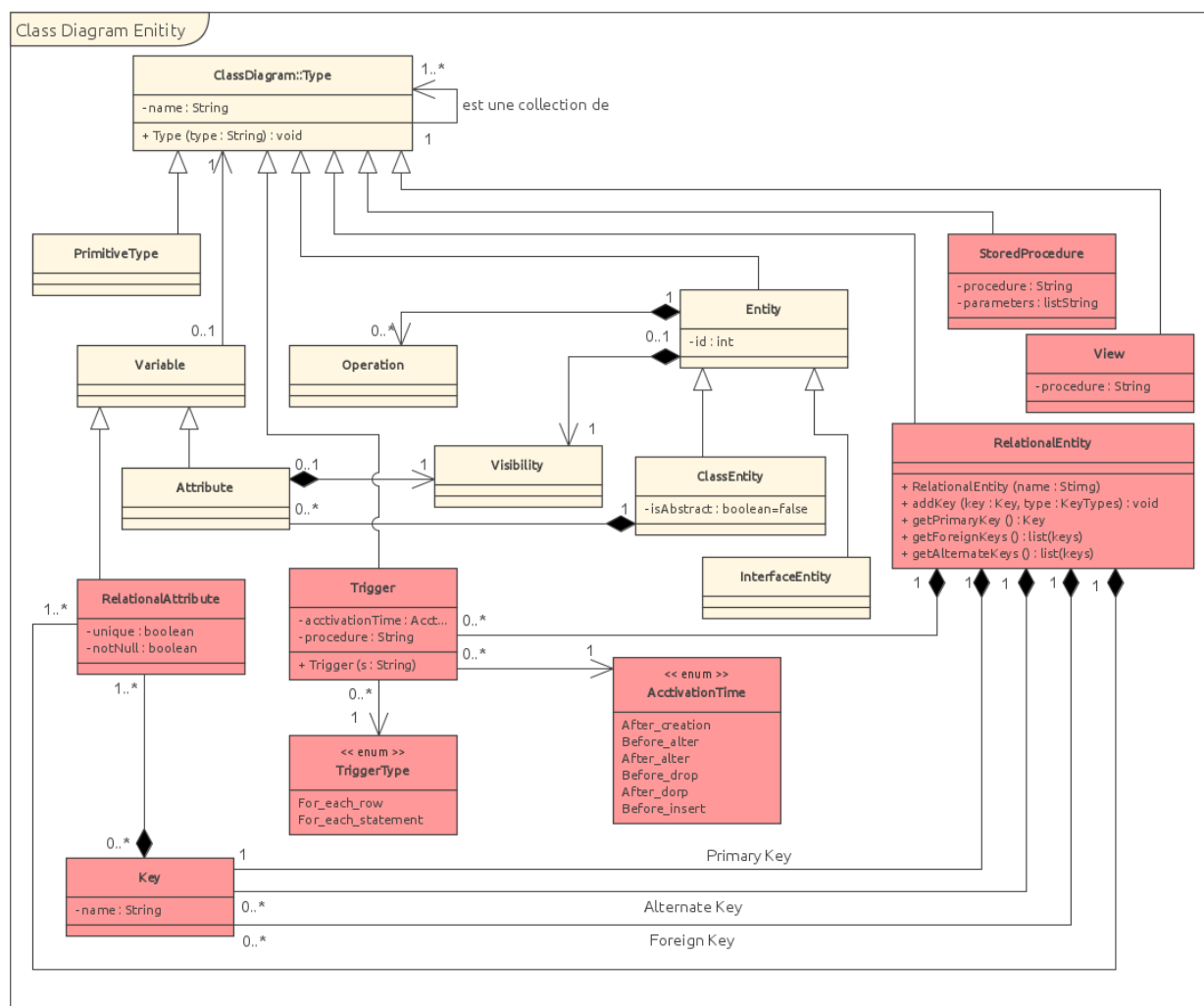
#### 3.1.6 Procédures

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.



## 3.2 Diagramme de classe – entité

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



### 3.2.1 RelationalEntity

Correspond à la table du meta-schéma

### 3.2.2 RelationalAttribute

Correspond à l'attribut du meta-schéma

### 3.2.3 Key

Correspond au 3 types de clés du meta-schéma qui seront stockées dans 3 attributs de RelationalEntity

### 3.2.4 Trigger, TriggerTypes et ActivationTime

Correspond au Trigger du meta-schéma

### 3.2.5 Procédure

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

### 3.2.6 StoredProcedure

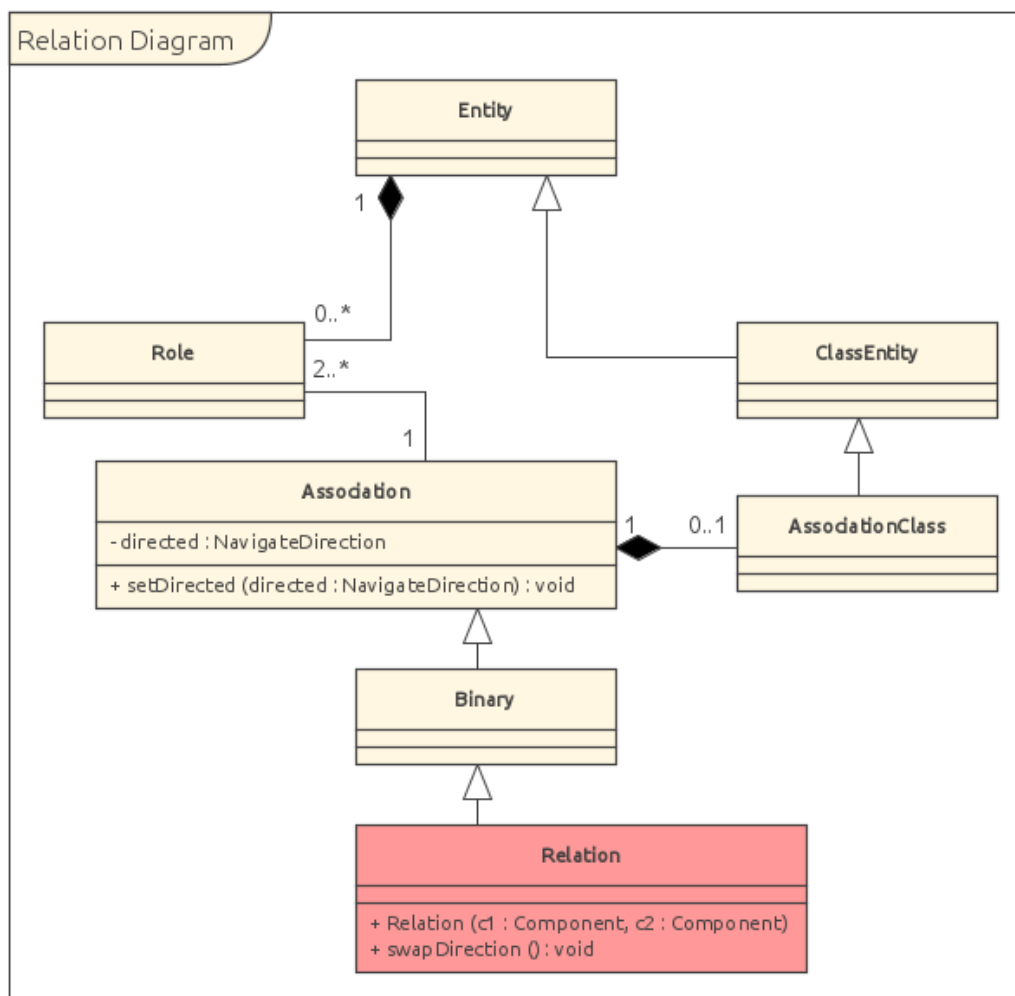
Correspondent respectivement à procédure stockée du meta-schéma.

### 3.2.7 View

Correspond aux vues du méta-schéma

## 3.3 Diagramme de classe - relation

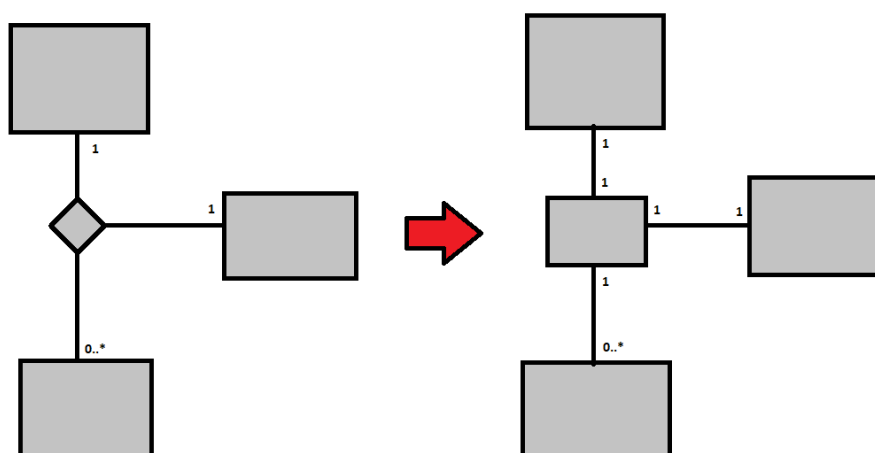
Ce diagramme montre comment la relation entre les tables est défini



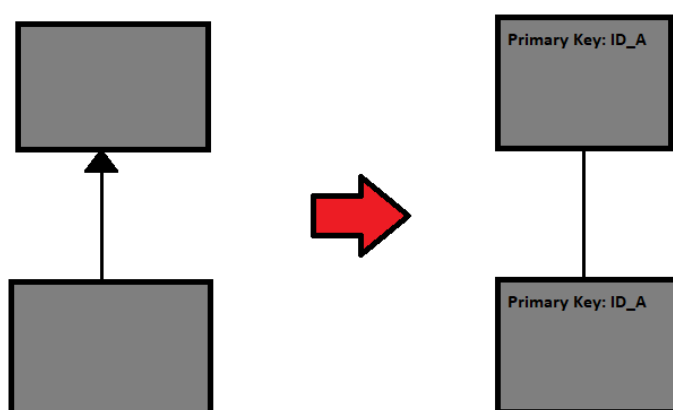
La classe relation est le seul moyen de créer des clés étrangères dans les tables. Quand une relation est créée, la table d'origine du lien est définie comme clé primaire et la table de destination comme clé étrangère. La clé primaire de la table d'origine est ajoutée à la table de destination comme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

## 4 Règles de conversion

- Classes, interfaces et classes d'association deviennent des **Tables**
- 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**.
- La conversion des **cardinalités** des relations se fait selon les règles standard (celles du cours de BDR)
- Si une association a une **classe d'association**, les attributs de cette classe sont ajoutés à une des tables de la relation selon les règles standard (BDR)
- Une **association multiple** est convertie en une table liée aux autres tables de la relation via des relations 1-X.
- Un avertissement est donné à l'utilisateur pour lui permettre de le faire manuellement



- Les liens d'**héritage** sont convertis en **association simple** avec la sous-classe ayant la même clé que la classe parent.



- Les classes internes deviennent des classes normales et suivent les mêmes règles que celles-ci.

**La conversion a 2 modes**, rajouter un attribut id a toutes les classes qui sera la clé primaire ou, pour chaque classe un popup laissera à l'utilisateur le choix de définir sa propre clé (attribut simple, composition d'attributs ou id par défaut).

## 5 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).

## 6 Interface

Mockup interface tables, attributs et clés primaire

Table	Attribute	Type	Visibility	NOT NULL	UNIQUE
<b>Primary key</b> <input type="text" value="key name"/>	id	int	Private	<input type="checkbox"/>	<input checked="" type="checkbox"/>
<u>id</u>	attribute_1	int	Private	<input type="checkbox"/>	<input type="checkbox"/>
	attribute_2	bool	Private	<input type="checkbox"/>	<input type="checkbox"/>
	attribute_3	string	Private	<input type="checkbox"/>	<input type="checkbox"/>

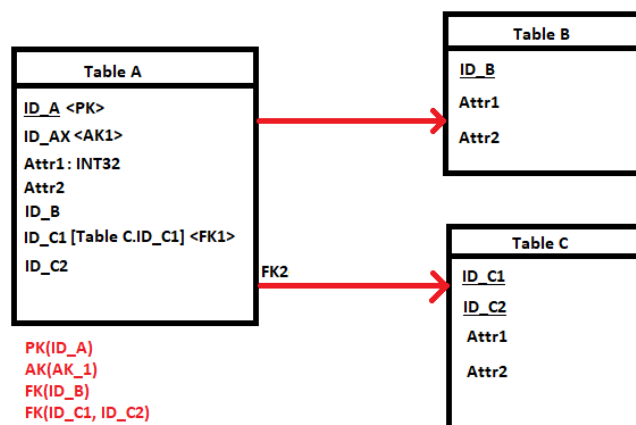
Mockup interface triggers

Trigger	Type	Activation	Abstract	Status	Procedure
trigger	for each row	after creation	<input type="checkbox"/>	<input type="checkbox"/>	//procedure text here

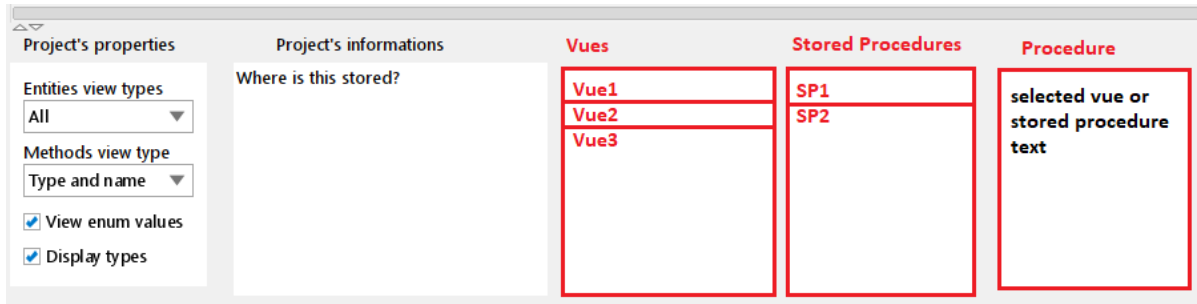
Mockup interface clés alternatives

Clé Alternative	Attributs
nom_clé_1	attribut_1
nom_clé_2	attribut_2

Mockup représentations des clés et attributs



Mockup interface de projet (vue et procédures stockées).



The mockup interface is divided into several sections:

- Project's properties:** Includes dropdowns for "Entities view types" (set to "All") and "Methods view type" (set to "Type and name"). It also has checkboxes for "View enum values" and "Display types", both of which are checked.
- Project's informations:** A section titled "Where is this stored?".
- Vues:** A list containing "Vue1", "Vue2", and "Vue3".
- Stored Procedures:** A list containing "SP1" and "SP2".
- Procedure:** A text area with the placeholder text "selected vue or stored procedure text".

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

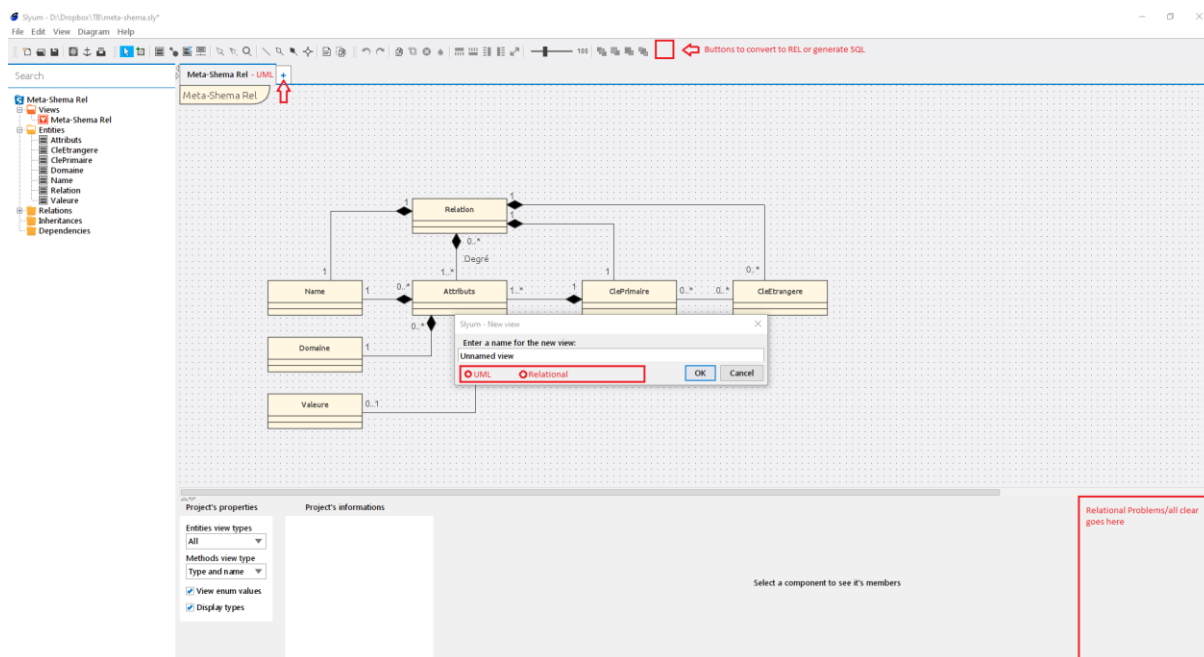
Une table pour définir une clé primaire est rajoutée. Cette table peut contenir un ou plusieurs attributs. A la création d'une classe, un attribut "id" est créé pour créer une clé par défaut.

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"



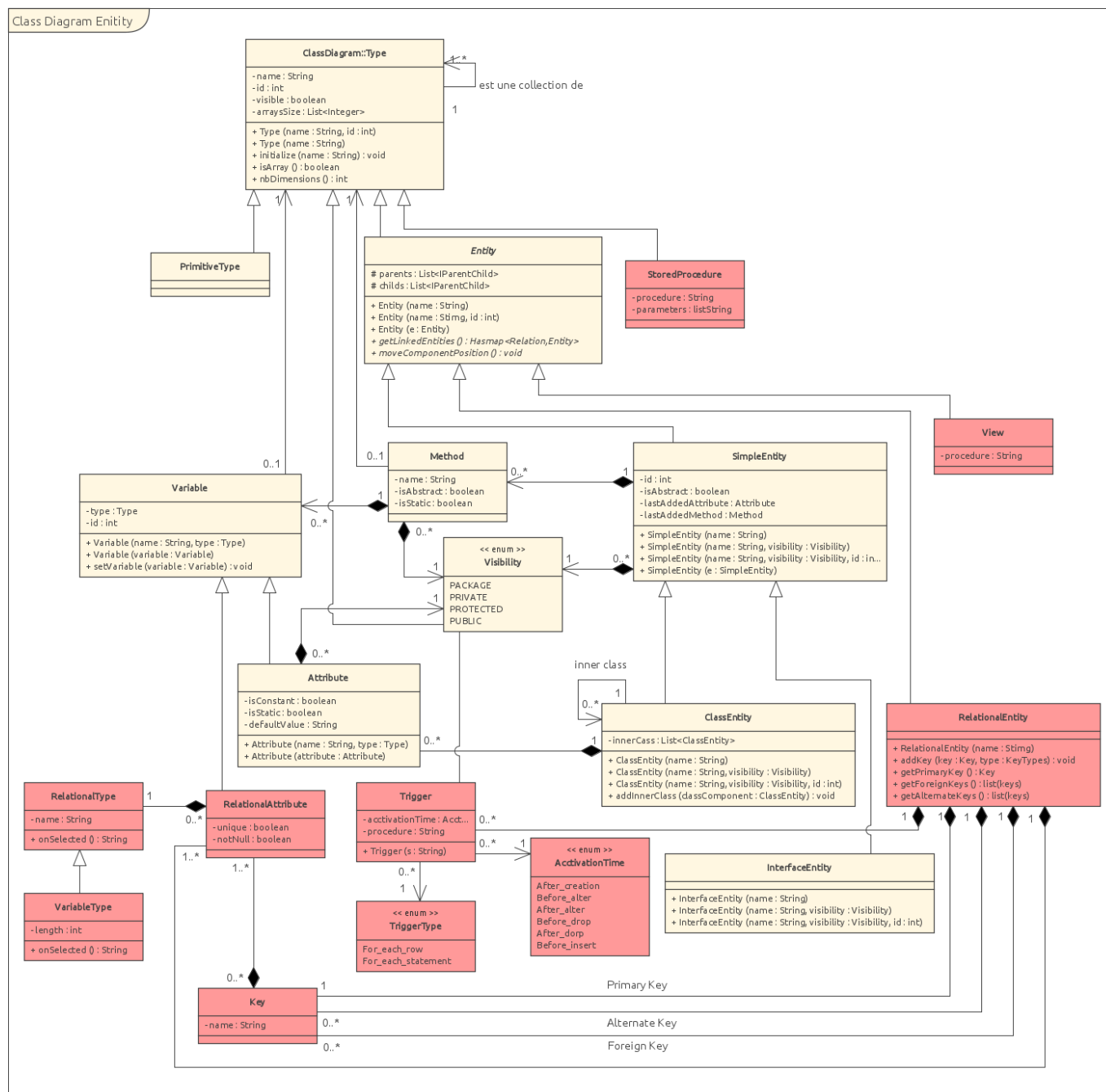
Le choix de uml ou relationnel quand on crée une nouvelle vue ne change pas les classes utilisées mais détermine les fonctions utilisables sur la GUI (ex : on peut créer des liens d'héritage en uml mais pas en relationnel)

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

## 7 Ce qui a été fait

### 7.1 Les objets

Les objets du diagramme de classe ont tous été implémenté avec succès. Certaines modifications ont pu être fait sur le diagramme original vu que le code de Slyum contenait une classe abstraite “simpleEntity” qui contiens les attributs du diagramme UML non-nécessaires au schéma relationnel. Je peux du coup faire que les éléments de ce dernier héritent d'Entity ce qui simplifie ma tâche énormément.

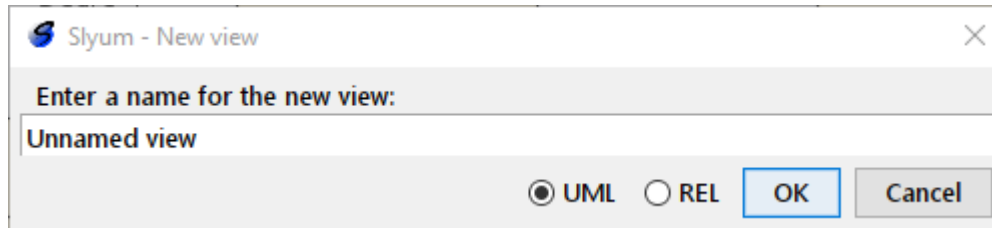


Pour rappel tous les objets héritent de Type et peuvent donc être représenté dans un même schéma. Ça sera le rôle de l'interface de faire en sorte que l'utilisateur ne mette pas des éléments uml et rel dans le même diagramme.

## 7.2 Les éléments d'interface

**Notice Importante :** La version courante de Slyum utilise le terme *vue(view)* pour définir les différents schémas d'un projet dû au fait qu'il utilise une représentation graphique "View" du Framework Swing pour les afficher. Ceci ne causait pas de problème pour un logiciel qui ne traite que de l'UML mais, maintenant que j'y rajoute du relationnel, il y a un conflit avec la *vue SQL*. Pour éviter toute confusion en attendant du renommage des onglet, j'utiliserai le terme "*vue graphique*" quand je parle des vues Swing et "*vue SQL*" ou "*view*" pour les vues SQL.

L'interface de création de vues graphiques a été modifiée pour pouvoir sélectionner si celle-ci représente un schéma UML ou REL.



De plus la création et la sélection d'une vue graphique rend actif/inactif les éléments graphiques nécessaires pour s'assurer que l'utilisateur ne puisse pas créer des éléments qui ne sont pas du schéma associé.



Les éléments des tables et des liens peuvent être créés mais pour le moment, l'affichage des clés n'est pas encore implémenté.

Juste avant l'écriture de ce rapport, l'affichage des vues SQL a changé d'un élément unique représentant une liste de celles-ci à des éléments distincts chacun représentant une vue SQL. De ce fait, le code des vues existe mais n'est pas encore 100% fonctionnel et du coup ne peut pas encore être affiché.

## 8 Ce qu'il reste à faire

### 8.1 Les éléments d'interface

- Finir les Vues SQL
- Implémenter les procédures stockées
- Ajouter la visualisation des clés dans les tables
- Ajouter l'interface de validation REL

Une fois ces éléments accomplis on devra pouvoir créer un schéma relationnel

### 8.2 Conversion du schéma UML en schéma relationnel

- Algorithme de conversion entre UML et REL (Règles de conversion → partie 3)
- Interface de choix de conversion (automatique/manuel)
- Interface de conversion manuel

### 8.3 Conversion du schéma relationnel en scripte SQL

- Algorithme de conversion REL à Scripte
- Interface du choix de SGBD

## 9 Timetable

SEMAINE	TACHE
22 AU 28 JUIN	Présentation et continuation interface vues
29 JUIN AU 5 JUILLET	Interfaces vues et clés
6 AU 12 JUILLET	Interfaces procédures stockées et validateur
13 AU 19 JUILLET	Conversion UML → REL et REL → SQL
20 AU 26 JUILLET	Semaine catchup ou Lien UML et REL et rapport
27 AU 31 JUILLET	Rapport Final



## 10 Annexes

### 10.1 Journal de travail

#### 10.1.1 Semaine 1 – 17 au 23 Février

- Lecture des directives de TB de l'HEIG
- Lecture du rapport Slyum
- Rapport Slyum

#### 10.1.2 Semaine 2 – 24 Février au 1 Mars

- Rédaction du cahier des charges

#### 10.1.3 Semaine 3 – 2 au 8 Mars

- Meta-schéma relationnel
- Meta-schéma rel. et modification du diagramme de classe UML pour implémenter REL

#### 10.1.4 Semaine 6 – 23 au 29 mars

- Continuation modification du diagramme de classe UML pour implémenter REL

#### 10.1.5 Semaine 7 – 30 Mars au 5 Avril

- Modélisation rapport
- Séparation des classes UML et REL

#### 10.1.6 Semaine 8 – 6 au 12 Avril

- RDV assistants
- Reformulation de la modélisation

#### 10.1.7 Semaine 9 – 13 Au 19 Avril

- Introduction des triggers dans le schéma
- Introduction des vues et des procédures stockées
- Changement de l'héritage de RelationalEntity et RelationalAttribute
- Mise à jour du meta-schéma

#### 10.1.8 Semaine 10 – 20 au 26 Avril

- Finalisation de la modélisation

### 10.1.9 Semaine 11 – 27 Avril au 3 Mai

- Implémentation de classes
  - RelationalAttribute
  - BufferRelationalAttribute
  - Key
  - RelationalEntity
- Changement de l'héritage de RelationalEntity grâce à la découverte de SimpleEntity

### 10.1.10 Semaine 12 - 4 au 10 Mai

- Lecture et compréhension du code graphique Slyum

### 10.1.11 Semaine 13 – 11 au 17 Mai

- Révision du Framework swing
- Suite révision swing
- Essai et expérimentation avec le code graphique de Slyum

### 10.1.12 Semaine 14 – 18 au 24 Mai

- Implémentation de la sélection de schéma “uml ou rel”
- Début de UI “rel-only” et implémentation du swap de UI quand on change de schéma

### 10.1.13 Semaine 15 – 25 au 31 Mai

- Début de l'implémentation des éléments graphiques du schéma relationnel
- Investigation de comment créer la UI pour les “vues” et “procédures stockées”

### 10.1.14 Semaine 16 – 1 au 7 Juin

- Remodélisation de l'affichage des clés
- Remodélisation des Views et StoredProcedures

### 10.1.15 Semaine 17 – 8 au 14 Juin

- Début reimplémentation de la UI des views
- Début rédaction rapport intermédiaire

### 10.1.16 Semaine 18 – 15 au 21 Juin

- Rédaction rapport intermédiaire
- Préparation présentation intermédiaire