

# Modélisation conceptuelle

Mini projet BDR

## Système de gestion des écoles

Groupe 6

Professeur : René Rentsch

Assistant : Sébastien Rosat



## **Table des matières**

1. Introduction .....	4
2. Modélisation conceptuelle .....	4
2.1. Schéma .....	4
2.2. Choix de conception .....	5
2.2.1. Modélisation des personnes .....	5
2.2.2. Modélisation du type de départ .....	5
2.2.3. Modélisation d'un bâtiment .....	5
2.2.4. Modélisation d'une salle de cours .....	5
2.2.5. Modélisation d'un test / note .....	6
2.2.6. Modélisation d'une leçon .....	6
2.2.7. Modélisation d'un cours .....	7
2.2.8. Modélisation d'un horaire .....	7
2.2.9. Modélisation des attributions de macarons .....	7
2.2.10. Modélisation des statistiques .....	7
2.3. Contraintes d'intégrité .....	7
3. Conclusion .....	8

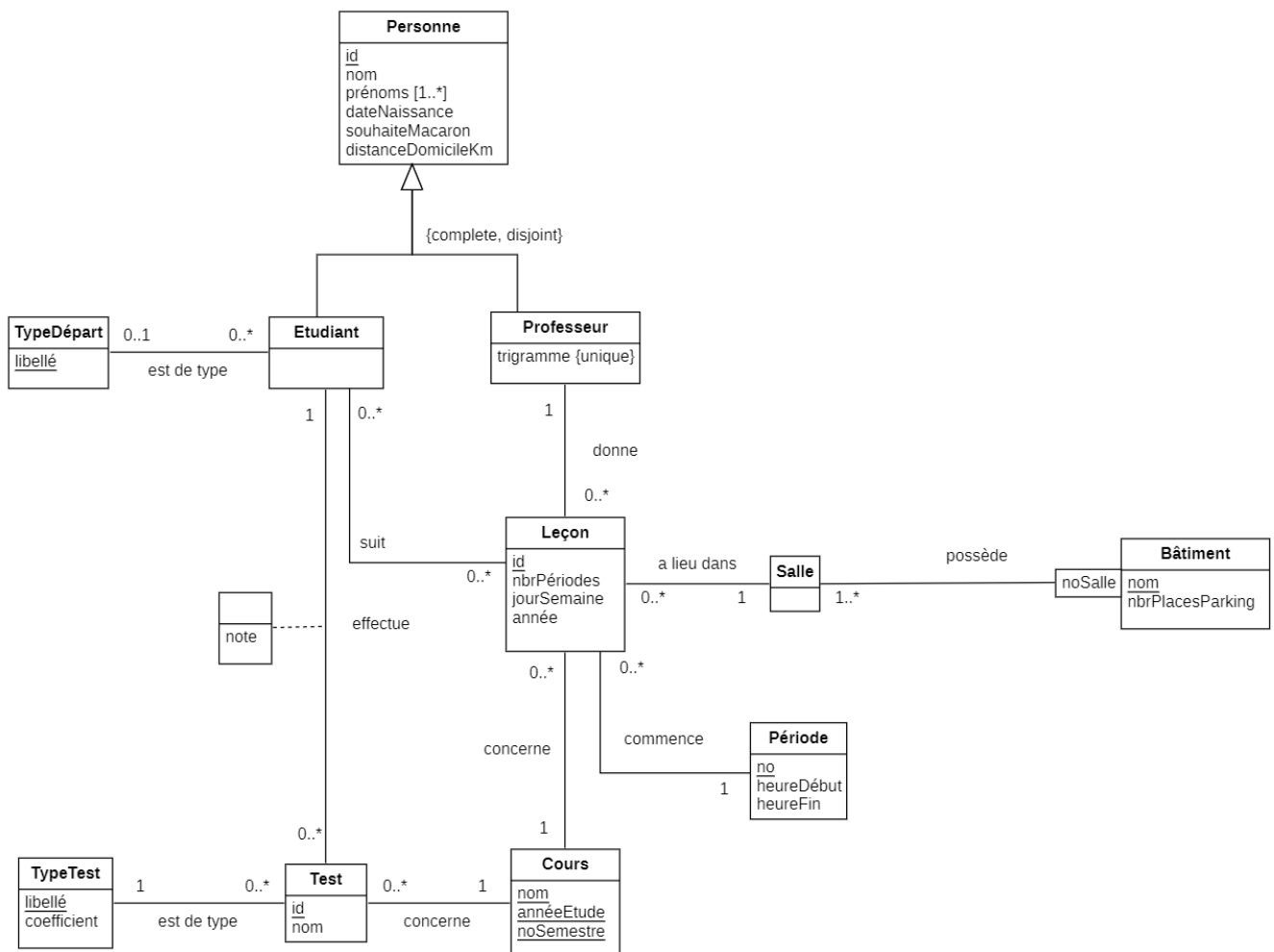
## 1. Introduction

Le but de ce document est de modéliser la base de données de notre projet du cours BDR au travers d'un schéma « entité-association ». Pour rappel, le but principal de cette application est de mettre en place un système de gestion des école (SMS – School Management System). Cette application s'intéresse uniquement à la partie « Formation et éducation » d'une école qui permet de gérer les entités suivantes :

- Les étudiants
- Les professeurs
- Les salles de cours et les bâtiments
- Les horaires
- Les cours
- Les notes

## 2. Modélisation conceptuelle

### 2.1. Schéma



## 2.2. Choix de conception

### 2.2.1. Modélisation des personnes

Comme cité plus haut, notre base de données contient des étudiants ainsi que des professeurs. Pour factoriser un maximum les informations, nous avons décidé de créer une table **Personne**. Les étudiants et les professeurs héritent donc de cette table ainsi que les attributs qui leurs sont communs. Il s'agit d'un héritage de type complet disjoint car toute instance de **Personne** est au moins instance de **Professeur** ou **Etudiant**, mais il est aussi disjoint car une personne ne peut pas être en même temps étudiant et professeur. Pour cette table, nous avons décidé d'utiliser une clé artificielle « id » pour identifier les différentes personnes car cela semble la solution la plus adaptée dans ce cas. Il est important de noter qu'une personne peut avoir plusieurs prénoms. Il s'agit donc d'un champ multivalué. Les étudiants ne possèdent pas d'attribut supplémentaire que ceux des **Personnes**. Cependant, les professeurs possèdent en plus un trigramme qui doit être unique.

### 2.2.2. Modélisation du type de départ

En plus des attributs hérités de la table **Personne**, les étudiants possèdent ou non une raison de départ. Cette raison permet de savoir la raison pour laquelle l'étudiant n'est plus aux études. Nous avons donc décidé de créer une table référence afin de factoriser au maximum ces « Types de départ » et ne pas répéter d'information.

Pour cette table, nous avons décidé d'utiliser le libellé comme clé primaire car on peut énumérer de manière univoque les différents types de départ qui seront :

- Réussite
- Echec
- Abandon

L'utilisation du libellé permet d'éviter l'utilisation d'une clé artificielle. Si un étudiant est toujours aux études, il ne possède pas de type de départ.

### 2.2.3. Modélisation d'un bâtiment

Les bâtiments sont identifiés par une clé naturelle qui est le nom du bâtiment. Cette solution paraît être la meilleure car il ne doit pas y avoir deux bâtiments possédant le même nom au sein de la même école.

### 2.2.4. Modélisation d'une salle de cours

Nous avons choisi d'utiliser un type d'entité faible pour représenter des salles de cours par rapport au bâtiment auquel elle appartient. Le numéro de chaque salle de cours constitue donc la clé partielle d'un type d'entité **Salle** qui est complétée par le numéro du bâtiment.

### 2.2.5. Modélisation d'un test / note

Comme décrit dans le cahier des charges, les étudiants peuvent effectuer des tests pour chaque cours suivis. Pour chaque test, un étudiant obtient une note. Nous avons décidé de modéliser la note obtenue comme un attribut de la relation entre **Etudiant** et **Test**. En d'autres termes, cela signifie que la note est modélisée comme la fonction entre un étudiant et un test.

La table **Test** possède une clé artificielle pour identifier de manière univoque les différents tests effectués. Nous n'avons pas utilisé le nom du test comme clé primaire car plusieurs tests ont potentiellement le même nom. Dans le cahier des charges, il est également indiqué que les tests possèdent un type. Pour ce faire, nous avons décidé de créer une table référence afin de factoriser au maximum ces « Types de test » et ne pas répéter d'information. Pour cette table, nous avons décidé d'utiliser le libellé comme clé primaire car on peut énumérer de manière univoque les différents types de test qui seront par exemple :

- TE (Travaux écrits),
- TP (Travaux pratiques)

De plus, chaque type de test possède son propre coefficient qui sera ensuite utilisé pour le calcul des moyennes.

### 2.2.6. Modélisation d'une leçon

Comme le mentionne le cahier des charges, la leçon est le point central de la base de données. Chaque leçon est liée à un ou plusieurs étudiants ainsi qu'à un seul professeur. Les leçons ont lieu dans une salle et concernent un cours spécifique. Nous avons décidé d'utiliser une clé artificielle pour identifier de manière univoque chaque leçon car nous ne possédons pas d'autre attribut permettant de faire ceci.

Pour modéliser de manière efficace les horaires d'une leçon (heure de début, durée), nous avons décidé d'utiliser en lieu et place de valeurs sous la forme « d'heures » des entiers permettant de modéliser le numéro de chaque période de la journée. Ainsi, une leçon commence à une certaine période de la journée et dure N périodes pour un jour donné. Nous avons également créé une table **Période** permettant de faire le lien entre les numéros de chaque période avec une heure précise. Grâce à cette technique, il est tout à fait possible d'ajouter des périodes à une journée ou de changer les heures de chaque période sans que cela ait des effets négatifs sur les données déjà insérées. Ce système est assez avantageux car il permet une plus grande souplesse et permettra par la suite de simplifier les requêtes afin de traiter des entiers plutôt que des dates. L'identifiant unique de chaque période est tout simplement le numéro de période de la journée qui est unique.

Dans la table leçon nous stockons également l'année durant laquelle le cours est donné car un cours n'est valable qu'une seule année.

### 2.2.7. Modélisation d'un cours

L'identifiant unique pour ce type d'entité est composé de tous les attributs qui le compose. En effet, le nom du cours associé à son année d'étude et son numéro de semestre durant lequel le cours est donné est unique. Nous avons donc utilisé ces 3 attributs comme clés primaire plutôt que d'utiliser une clé artificielle. Il est également important de noter que le numéro de semestre peut être soit 1 ou 2. Aucune autre valeur n'est acceptée et l'année d'étude correspond au numéro de l'année durant laquelle le cours est donné. Par exemple année d'étude = 2 pour le cours BDR.

### 2.2.8. Modélisation d'un horaire

Dans le cahier des charges, il est indiqué qu'il doit être possible d'afficher l'horaire de chaque étudiant, professeur ou salle de cours. Nous avons décidé de ne pas créer de table **Horaire** pour modéliser ceci car l'horaire est en fait une liste de différentes leçons. Pour factoriser les informations, nous n'avons donc pas besoin de créer un type d'entité à cet effet.

### 2.2.9. Modélisation des attributions de macarons

Dans le cahier des charges, il est indiqué qu'il est possible de gérer l'attribution des macarons aux étudiants et professeurs. Cette fonctionnalité ne fait pas intervenir de type d'entité supplémentaire. En utilisant les tables existantes et l'attribut « souhaiteMacaron » de la table **Personne** il sera possible de récupérer les personnes éligibles. Il n'y a donc rien à faire à ce niveau pour la modélisation conceptuelle.

### 2.2.10. Modélisation des statistiques

Tout comme pour les macarons, les statistiques ne font intervenir aucun nouveau type d'entité. En effet telles que présentées dans le cahier des charges, les statistiques se basent sur les données existantes et génèrent des résultats associés. Il n'y a donc rien à faire à ce niveau pour la modélisation conceptuelle.

## 2.3. Contraintes d'intégrité

- Un étudiant ou un professeur ne peuvent pas participer à plusieurs leçons qui ont lieu en même temps
- Un étudiant ne peut pas avoir un test d'un cours qu'il ne suit pas
- Un étudiant ne peut pas suivre 2 cours de semestre différents mais d'une même année
- Les heures des différentes périodes ne peuvent pas se chevaucher
- Un cours ne peut pas avoir un numéro de semestre différent de 1 ou 2
- Une leçon ne peut pas avoir lieu dans la même salle à la même heure
- Une leçon ne peut pas avoir lieu à une période de début n'existant pas dans la table Période
- Il n'est pas possible de créer une leçon avec une année déjà passée
- Une leçon ne peut pas durer plus longtemps qu'il n'y a de périodes dans la journée

- La date de naissance d'une personne ne peut pas être plus grande que la date du jour
- La distance domicile-école en km doit être une valeur supérieure ou égale à 0

### 3. Conclusion

En conclusion, nous avons essayé de respecter au maximum notre cahier des charges. Malheureusement nous avons rencontré quelques difficultés. C'est pourquoi nous avons dû modifier ou ajouter des tables que nous n'avions pas prévues dans le cahier des charges. Par exemple pour pouvoir compter le nombre de tests qu'un élève a eus pour un cours, nous étions obligés de rajouter la table **Test**.

Nous nous sommes rendu compte qu'il est difficile de tout prévoir et sommes conscients que tout oubli posera beaucoup de problèmes durant la conception. L'étape de passer par le schéma entité association est donc une étape nécessaire car elle nous permet de nous rendre compte des problèmes que posent les idées décrites dans le cahier des charges et de les corriger.

Cette étape nous a donc beaucoup aidé à déceler nos erreurs et nous sommes donc content d'être passé par là.