

---

## MODÉLISATION ASSISTÉE PAR L'IA POUR L'ENSEIGNEMENT DES BASES DE DONNÉES

---

**Emmanuel DESMONTILS<sup>1</sup>**

IREM Pays de la Loire

**Pascal ANDRÉ<sup>2</sup>**

**Loïc KODO<sup>3</sup>**

**Résumé.** Il y a 5 ans, la spécialité NSI est apparue au lycée en France. Pour le lancement, le DIU EIL (Enseigner l'Informatique au Lycée) a permis de former en urgence une grande partie des enseignants de cette spécialité. Parmi les thématiques abordées, l'enseignement des bases de données a trouvé sa place en terminale. Alors que la manipulation de la base, via le langage SQL, langage basé sur l'algèbre relationnelle, apparaît assez facile aux enseignants, la structuration de la base leur pose beaucoup de difficultés. Dans ce travail, nous rappelons les aspects méthodologiques de la structuration des données et les bonnes pratiques associées sur lesquels nous nous appuyons pour proposer une démarche pragmatique, au regard de notre expérience dans le domaine, pour aider les enseignants et les étudiants. Cette démarche s'appuie sur un travail expérimental à la fois d'analyse des modèles existants (sujets de bac ou proposition d'élève par exemple) et d'analyse des outils d'aide, notamment l'utilisation d'IA génératives (dans le cadre posé par le MEN) grand public ou open source, mais aussi locales ou distantes. Pour décrire les modèles de données, nous nous appuyons sur le modèle EAP de Merise, très accessible et au cœur de méthodologies éprouvées.

**Mots-clés.** Modélisation, conception de systèmes d'information, bases de données, méthodologie, intelligence artificielle, didactique des bases de données.

### Introduction

Ces dernières années, l'informatique a fait son retour au lycée avec la spécialité NSI (MENJS, 2019a, 2019b) ; auparavant, il y a eu, en bac général série S, l'option informatique en terminale (1988-92, 1995-98) et l'ISN en terminale (2012-16). La majorité des enseignants de cette spécialité sont issus des disciplines des mathématiques, de physique ou de technologie. Ils ont suivi une formation accélérée, le DIU EIL, qui leur a permis d'acquérir les connaissances fondamentales nécessaires à

l'enseignement de cette spécialité. Dans le cadre du chapitre consacré aux bases de données (MENJS, 2019b), les enseignants ont principalement concentré leur développement de compétences sur la compréhension des systèmes de gestion de base de données et sur le langage SQL basé sur l'**algèbre relationnelle** de Codd. Une difficulté majeure qu'ils rencontrent concerne la conception de leurs bases de données, qui, au-delà de l'aspect mathématique, met en œuvre un processus structuré basé sur l'abstraction puis le raffinement progressif des informations et de leurs intercon-

---

<sup>1</sup> emmanuel.desmontils@univ-nantes.fr

<sup>2</sup> pascal.andre@univ-nantes.fr

<sup>3</sup> kossi.kodo@etu.univ-nantes.fr

nexions en données structurées manipulables par des algorithmes. En effet, la modélisation des systèmes d'information est peu abordée dans leur formation. La manipulation des modèles de données (production, vérification, compréhension) est une difficulté fréquente qui va bien au-delà de l'algèbre relationnelle.

Depuis fin 2022, les systèmes artificiels intelligents génératifs (SAIg) (Andler, 2023), souvent appelés abusivement « intelligence artificielle », sont arrivés dans le paysage médiatique grand public. Ils ont bouleversé les pratiques enseignantes et le travail des élèves comme des étudiants (Bruillard, 2025). En constante évolution, ces outils sont parfois délicats à utiliser et proposent des résultats souvent intéressants, parfois compliqués, coûteux et risqués à exploiter. Cependant, ils apparaissent comme des outils potentiellement utiles pour l'éducation sous réserve de maîtriser leur usage et de comprendre leur impact (Higuera et Iyer, 2024). Dans ce cadre, MENESR (s.d.) propose un cadre pour leur usage dans ce contexte.

Nous nous intéressons à la complexité du travail de modélisation pour l'enseignement, en particulier pour la modélisation de données en bases de données. Nous cherchons à comprendre comment les SAIg peuvent être des outils utiles. Cette démarche intervient dans différents cas d'usage :

1. créer des énoncés de sujets à partir d'un modèle explicite (conceptuel ou relationnel) ou implicite, pour faire des sujets de TD, faciles à ajuster en séance, et des sujets d'évaluation pour lesquels il n'est pas possible de donner beaucoup d'explication ;
2. tester un sujet écrit en langage naturel : produire un premier modèle conceptuel ou relationnel, et vérifier que le texte est complet et clair ;

3. comparer les travaux (étudiants) à un modèle de référence (classes d'équivalence) ;
4. détecter des erreurs (correction ou vérification d'un modèle) : assistance à la correction ;
5. produire un schéma de Base de Données (BD) à partir d'un modèle ;
6. produire le modèle à partir d'une BD (rétro-ingénierie) : vérification ou évolution de BD.

Pour certains cas, comme le passage d'un modèle conceptuel à un modèle relationnel (cas 5), l'utilisation d'un SAIg n'est pas nécessaire, car la méthode est simple et il existe des outils dédiés qui le font bien (Mocodo, outils de génie logiciel...).

Dans ce travail, nous allons nous intéresser tout particulièrement à identifier la « distance qualitative » entre un modèle issu d'une méthodologie classique et celui produit par un SAIg. Il est à noter que les méthodologies classiques sont souvent mal maîtrisées par les étudiants, mais aussi par les enseignants (notamment en fonction du niveau du public : bac, licence, CAPES, agrégation). Ce questionnaire sur la qualité des modèles produits permet de traiter les cas 2 et 3. Nous avons mené diverses expérimentations pour comparer les modèles conçus par des analystes, experts ou pas, et par des outils de génération automatisés pour étayer notre argumentation.

À l'issue de ce constat, nous avons mené une autre série d'expérimentations pour tester différentes stratégies pour améliorer les résultats obtenus par les SAIg. Les paramètres sont nombreux pour estimer la compréhension du contexte (explicitation, biais culturels...) en lien avec les éléments de langages utilisés pour les cas d'étude.

Il en résulte une proposition d'outil méthodologique pour gérer cette distance et produire un modèle pour répondre aux cas d'usage.

Les contributions de cet article sont :

- (i) un état des lieux de la modélisation des données dans un système d'information pour une mise en œuvre dans les bases de données relationnelles ;
- (ii) une proposition structurée de conception de modèles de données assistée par les technologies récentes d'interactions homme/machine pour une application efficace et une pédagogie en phase avec son contexte actuel des pratiques des apprenants ;
- (iii) une validation des deux contributions précédentes par des expérimentations concrètes et suffisamment représentatives.

Le présent article est organisé comme suit. Les deux premières sections positionnent le domaine et son contexte : modélisation de données et de son enseignement dans la section 1 et synthèse rapide des SAIG dans la section 2. En section 3 nous présentons et analysons les résultats d'expérimentations sur l'analyse des modèles existants, notamment ceux des sujets d'examens de bac, CAPES et agrégation, et les comparons avec ceux que nous avons obtenus en utilisant des SAIG sur les sujets. Les résultats n'étant pas convaincants, notamment en termes de variabilité sur la qualité, nous avons mené une autre série d'expérimentations, cette fois sur l'utilisation pratique des outils SAIG, que nous présentons dans la section 4. Il en résulte une méthodologie que nous synthétisons dans la section 5 avant de conclure en présentant des perspectives ouvertes de travaux futurs.

## 1. – Aspects méthodologiques de la structuration des données et bonnes pratiques

Il nous semble important de positionner le périmètre du sujet et la terminologie, notamment pour les non-spécialistes, avant d'aborder les aspects méthodologiques.

### 1.1. - Contexte, périmètre et terminologie

La conception des bases de données, notre **domaine d'étude**, se place dans un contexte plus large de conception des systèmes d'information, dont les « données » constituent un des axes majeurs, car il fixe la structure du **système d'information (SI)**. Pour simplifier, on distingue (au moins) deux aspects dans le SI : un aspect dit *statique* qui structure les données du SI et un aspect dit *comportemental* qui comprend le traitement de l'information et la dynamique de ce traitement (ordre dans lequel on modifie/accède aux informations). Dans cet article, on se focalise uniquement sur l'aspect statique. Pour cet aspect, on a des propriétés attendues, qui matérialisent la qualité du SI. Par exemple, pour la structure des données, on réduit au maximum la redondance d'information pour éviter des incohérences lors des mises à jour de la base de données.

L'enseignement des méthodes de conception de SI figure aux programmes de DUT, de BUT et de MIAE depuis les années 80 ; il est également souvent présent aux programmes de licence informatique. Il est donc naturel qu'il apparaisse en spécialité NSI. Nous conseillons la lecture d'ouvrages synthétiques tels que André et Vailly (2001), Nanci et Espinasse (2001), Morley et al. (2011) pour bien comprendre et enseigner ce sujet. « *Enseigner des méthodes d'analyse et de conception de systèmes d'information (ACSI) est un des exercices les plus difficiles de l'enseignement de l'informatique* ». Cette remarque, extraite de

l'avant-propos de André et Vailly (2001), montre bien l'ampleur de la tâche à laquelle on se confronte. La difficulté vient à la fois de la variété des domaines d'application, dits métiers, de la variété des systèmes, mais aussi de la variété des profils des acteurs concernés par ces systèmes d'information, de la friction entre les domaines métiers (gestion d'organisations — pour simplifier) et les domaines techniques (informatique).

Sans entrer dans le détail de la conception des données, il nous semble important d'introduire la notion d'abstraction/raffinement qui établit la distance entre le système d'information et sa mise en œuvre.

Tout d'abord, on distingue la *donnée* élémentaire (75), de l'*information* qui structure et donne un sens (75 ans) de la *connaissance* qui en donne une interprétation et un usage dans un contexte donné (âge du capitaine, qui doit être en retraite). Une explication plus approfondie est donnée dans Boubaker et al. (2010).

Ensuite, on distingue les niveaux d'abstraction vis-à-vis de la mise en œuvre sur un support réel (e.g. une base de données) : le niveau *conceptuel* (informations et corrélations), le niveau *logique* ou organisationnel (type de base de données) et le niveau *physique* ou opérationnel (un système particulier de gestion de bases de données — SGBD).

La méthodologie fait habituellement appel à la notion de modèle et de modélisation, nous n'y échappons pas ici. Un **modèle** est une interprétation explicite par son utilisateur de la compréhension d'une situation, ou plus exactement de l'idée qu'il se fait d'une situation. Il peut être exprimé par des mathématiques, des symboles, des mots, mais essentiellement, c'est une description d'entités et de relations entre entités (André et Vailly, 2001). Le terme « représentation » est aussi utilisé. Ainsi, mo-

déliser revient à élaborer une vue partielle, plus ou moins abstraite de l'existant. Un modèle est correct s'il permet de répondre aux questions qu'on se pose. Un modèle est opérationnel s'il peut être exécuté par une machine. Par abus de langage, dans ce qui suit, le terme modèle désignera à la fois le résultat de la modélisation, la théorie sous-jacente et la notation utilisée. On citera ainsi le modèle relationnel, le modèle EAP, le modèle conceptuel des données... La modélisation, en informatique, est le passage du domaine du problème à celui de sa solution informatique. Ses difficultés sont d'appréhender la sémantique des concepts du monde réel qui sont liés au contexte du système d'information étudié et de la transformer en signes manipulables par les ordinateurs. La méthode de conception consistera donc à construire des modèles et, dans notre contexte, ce sont des modèles de données.

## 1.2. - Exemple « fil rouge »

Pour illustrer notre propos, nous utiliserons le même exemple dans cette section, mais aussi pour une partie des expérimentations dans la suite de l'article. L'énoncé est le suivant :

*Une mairie recense les associations de la commune. On y trouve une description des associations (nom, objectifs, etc.), les activités proposées par les associations et les administrés de la commune. Une activité peut être proposée par plusieurs associations et une association peut proposer plusieurs activités. Pour chaque activité, il y a un seul membre de l'association qui est nommé responsable. Les personnes de la commune peuvent être adhérentes de zéro, une ou plusieurs associations. On identifiera aussi le président et le trésorier pour chacune des associations. On note qu'il n'y a qu'un seul président et qu'un seul trésorier dans une association. Par contre, une personne peut être prési-*

dente ou trésorière de plusieurs associations.

Pour des raisons pédagogiques, cet exemple est concis et s'inscrit dans la vie courante pour être compris par tous.

### 1.3. - Conception de base de données

Il existe deux grandes approches de conception de bases de données, et des variantes :

- l'approche par dépendances fonctionnelles et normalisation qui s'applique aux bases de données relationnelles et à l'algèbre associée ;
- l'approche systémique par raffinement de modèles conceptuels, qui est applicable plus largement, car c'est au niveau logique qu'on peut choisir une implantation par modèle relationnel.

Les deux approches sont décrites en détail dans André et Vailly (2001) et Hainaut (2022). Nous les résumons dans la suite.

#### *Conception par normalisation*

Le modèle relationnel, conçu par E. F. Codd (Codd, 1970), est fondé sur la notion mathématique de *relation*, ensemble de  $n$ -uplets d'attributs ayant chacun un nom et un domaine de valeur. Un schéma de base de données relationnel (BDR) est défini par un ensemble de relations et un ensemble de dépendances fonctionnelles (DF)<sup>4</sup>.

Une dépendance fonctionnelle «  $X \rightarrow Y$  » entre deux ensembles d'attributs  $X$  et  $Y$  signifie que la connaissance de  $X$  implique au plus une valeur de  $Y$  ( $Y$  dépend de  $X$ ). Ces dépendances sont réflexives, non-symétriques, transitives et

munies d'autres propriétés (pseudo-transitivité, union, décomposition...). Une dépendance est élémentaire si elle n'est pas réductible à un sous ensemble d'attributs de  $X$ . L'ensemble d'attributs  $X$  est une clé de la relation  $R(X, Y, Z)$  si «  $X \rightarrow Y, Z$  » est une DF élémentaire. Une DF est directe si elle n'est pas transitive.

On classe les schémas de bases de données selon la structure des relations et la nature des dépendances en formes normales<sup>5</sup> d'inclusion croissante (1NF, 2NF, 3NF...), par rapport à deux critères essentiels : minimiser la redondance de l'information et minimiser les anomalies liées aux opérations de mise à jour. Deux formes sont d'un intérêt particulier, la 3NF (pas de redondance) et la BCKNF (Boyce-Codd-Kent) qui élimine des anomalies de clés candidates notamment de dépendances indirectes.

L'objectif de la normalisation est de définir un schéma de relations permettant d'éviter la redondance de données, les incohérences lors des mises à jour et les anomalies lors d'insertion et/ou suppression. La méthode de conception d'un schéma relationnel consiste à définir les attributs et les dépendances fonctionnelles puis à procéder par itération pour les diviser en relations par décomposition selon des règles définies. Des algorithmes sont proposés et illustrés par des exemples dans de multiples sources, dont André et Vailly (2001), Hainaut (2022), Gardarin (2003).

#### *Illustration sur le cas « fil rouge »*

De l'analyse du texte et son extrapolation sur les personnes résultent :

---

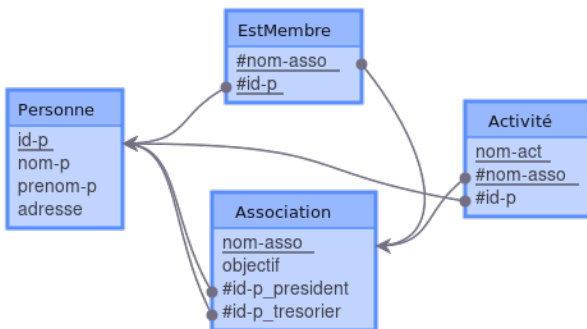
<sup>4</sup> Nous simplifions les choses ici, pour des raisons pédagogiques, car il y a aussi des prédicats et d'autres types de dépendances.

---

<sup>5</sup> Les formes normales sont des décompositions modulaires en relations ayant de « bonnes propriétés » (non redondance, préservation des dépendances).

- (i) la liste des attributs suivants : *nom-p*, *prenom-p*, *adresse* d'une personne, *nom-asso*, *objectif*, *nom-act* complétée par les identifiants mentionnés dans l'énoncé et les attributs. Il y a des conventions d'écriture (pas d'accent, usage de séparateurs, majuscules-minuscules...); ces attributs sont normalisés au sens où, pour le nom, il est indispensable de suffixer pour ne pas confondre les noms de personnes et ceux d'associations ;
- (ii) un ensemble de dépendances fonctionnelles entre ces attributs, par exemple « *id-p* → *nom-p* » indique qu'à un identifiant de personne correspond un seul nom ; cette dépendance deviendra une clé de la relation *Personne*. La dépendance « *id-p\_president* → *id-p* » indique que le président (d'une association) est une personne.

Dans la relation association, la double présence de *id-p* correspondant à deux situations, nous avons suffixé pour distinguer les deux rôles. L'approche par normalisation avec les dépendances fonctionnelles produit le schéma relationnel suivant avec 4 relations :



**Figure 1** : Modèle relationnel du cas Asso.

- Personne (id-p, nom-p, prenom-p, adresse)
- Association (nom-asso, objectif, président#, trésorier#)
- Activité (nom-act, nom-asso#, id-p#)
- EstMembre (id-p#, nom-asso#)

Les attributs soulignés dans la figure 1 sont les clés. Les flèches représentent les contraintes d'intégrité référentielles (CIR) issues des dépendances. Par exemple, le pré-

sident d'une association est une personne. D'autres contraintes sont possibles telles que la totalité (*toute personne est membre d'au moins une association*), l'exclusion (*le trésorier et le président sont deux personnes différentes*) exprimables ou pas par des dépendances fonctionnelles.

### Conception par raffinement

Cette approche, proposée par les méthodes ACSI, consiste à proposer un modèle conceptuel de la structuration des données (MCD) puis à *raffiner* (inverse de l'abstraction) vers un modèle logique MLD (ici celui des bases de données relationnelles) puis vers un SGBD spécifique du marché pour la mise en œuvre opérationnelle (MPD). Cette approche présente de nombreux avantages :

- elle est plus facile à construire et à valider, car le formalisme (quel que soit le langage) est plus simple et plus proche du langage naturel — la normalisation est plus technique. De fait, les utilisateurs finaux et décideurs peuvent participer au processus ;
- elle donne de meilleurs résultats, car le raffinement progressif conduit automatiquement à un schéma en 3NF par construction. Il garantit la qualité ;
- elle est plus évolutive, car indépendante du modèle de représentation logique des données (base de données hiérarchiques ou réseau par exemple).

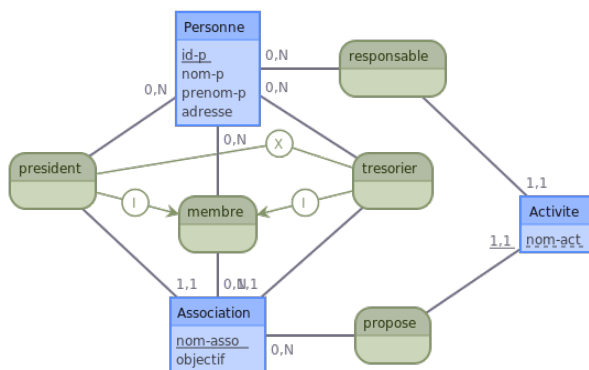
Nous détaillons cette approche dans les sections 1.5 et 1.6. Mais auparavant, introduisons le formalisme du modèle MCD.

#### 1.4. - Le formalisme MCD

Le modèle MCD est exprimé dans le formalisme Entité-Association-Propriété (E-A-P) qui est à la fois concis, abstrait et compréhensible par tous. Une **propriété** est une information élémentaire prenant ses valeurs dans un

domaine. Une **entité** est un objet matériel ou immatériel ayant une existence propre et conforme aux choix de gestion du système d'information étudié (e.g. *personne*, *association*). Un type d'entité (TE) est un ensemble d'entités ; il est défini par un ensemble de propriétés et l'identifiant du TE est un sous-ensemble des propriétés qui permettent de distinguer deux occurrences (deux entités). Une association est un lien entre des entités (unaire, binaire, ternaire, *n*-aire). Un type d'association (TA) définit l'ensemble des associations avec différentes contraintes telles que les cardinalités. Par exemple, *EstMembre* est un lien entre des *personnes* et des *associations*.

Dans la figure 2, les TE sont rectangulaires, les TA ont des coins arrondis. Les cardinalités (1,1) de *EstPrésident* signifient qu'une association a forcément un (et un seul) membre président. Les cardinalités (0,N) de *EstPrésident* signifient qu'une personne peut être présidente de plusieurs associations.



**Figure 2** : MCD du cas Asso.

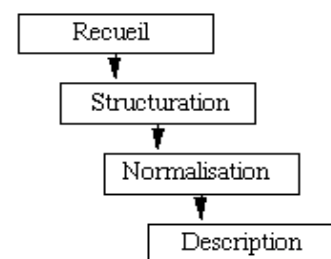
Le modèle MCD est défini par des ensembles d'entités et d'associations, respectivement TE et TA, et des contraintes (cardinalités, intégrité fonctionnelle, contraintes logiques inclusion, exclusion, totalité, unicité...). À noter, on ne représente que des contraintes sur des aspects statiques simples (*un membre a plus de 18 ans*) ou ensemblistes (*une activité n'a pas plus de 20 membres*). Les

aspects dynamiques simples (*la cotisation augmente d'un coefficient donné d'une année à l'autre*) ou ensembliste (*l'augmentation de l'ensemble des cotisations pour une année ne dépasse pas 3 %*) ne seront pas abordés. Dans la figure 2, outre les cardinalités, nous avons deux contraintes d'inclusion (I) qui indiquent que le président et le trésorier sont des membres de l'association. Il existe aussi une contrainte d'exclusion, qui indique que le président d'une association ne peut être trésorier de cette même association. Le lecteur notera que le formalisme E-A-P est plus riche que la présentation que nous en avons faite pour donner au lecteur uniquement l'information nécessaire pour comprendre cet article.

### 1.5. - Construction d'un MCD

Plusieurs approches de construction du MCD sont possibles :

- **systematique** : cette technique consiste d'abord à recueillir ces données, ensuite à les structurer, puis à les normaliser et enfin à les décrire (cf. figure 3).



**Figure 3** : Les quatre étapes de construction du MCD.

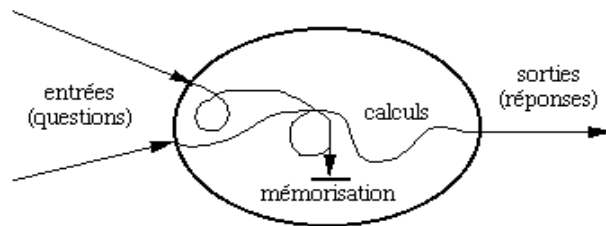
Cette démarche, qui ressemble à celle proposée pour une BDR dans la section 1.3. « Conception par normalisation », n'est pas toujours rectiligne. La normalisation, par exemple, peut contraindre l'analyste à revenir sur la structuration, voire sur la phase de recueil. Il est possible également de « sauter » quelques étapes.

- **analyse de textes** : cette technique repose sur l'analyse linguistique de textes décrivant les informations à mémoriser. Quelques règles d'analyse syntaxique permettent d'obtenir la structure sous-jacente. Cela nécessite une normalisation des phrases telle que : sujet-verbe-complément.
- **génération spontanée** : l'expertise d'un domaine et l'expérience du modèle permettent de monter rapidement en abstraction pour identifier entités et associations. On précise ensuite les propriétés et leur domaine ainsi que les contraintes diverses qui réduisent l'espace des solutions.

Nous conseillons la méthode systématique aux débutants et la génération spontanée aux étudiants confirmés. La méthode par analyse de textes est la plus adaptée pour les IA génératives, mais cela implique un formatage des énoncés textuels qui frise la paraphrase d'un modèle existant. Nous la déconseillons aux étudiants comme approche unique, mais elle est utile comme heuristique dans l'approche systématique.

Revenons un peu plus en détail sur cette méthode, car elle se distinguera de l'approche par IA que nous explorons dans la section 4. Le lecteur trouvera un développement plus précis dans André et Vailly (2001), Nanci et Espinasse (2001), Hainaut (2022).

Le premier travail à faire consiste à **recueillir**, auprès des utilisateurs, les informations utiles. L'utilité d'une information se mesure en examinant les objectifs assignés au système d'information pour ses sorties (*cf.* figure 4). Les informations inutiles ne seront pas recueillies.



**Figure 4** : Comportement d'un système d'information.

Un travail de normalisation terminologique (homonymes, synonymes...) est fait pour produire un dictionnaire des données. Le choix des informations est soumis à l'utilisateur qui les acceptera ou les refusera. On ajoute aussi des règles de gestion sur l'usage de ces informations dans l'organisation. À la fin de cette étape, l'analyste se trouve avec une masse d'informations qu'il lui faut trier et structurer.

La **structuration** comprend la mise en évidence des entités, en regroupant toutes les informations par affinité. On élimine toutes les informations calculées pour ne garder que des informations brutes qu'on appellera dorénavant propriétés. On détermine les identifiants. Puis on met en évidence les associations, porteuses ou pas d'informations, par corrélation entre entités et sur la base des informations non rattachées à des entités. On étudie les cardinalités à partir des occurrences d'entités et des règles de gestion. La dernière étape consiste à définir les contraintes entre concepts du MCD (appelées *invariant du modèle de données*).

La **normalisation** consiste à vérifier si le modèle est conforme aux règles de l'art et le cas échéant à le modifier. Cela va du renommage de concepts à la restructuration de modèle en passant par un assouplissement des contraintes. On normalise les propriétés, les TE et TA, les cardinalités... De nombreuses règles sont à vérifier.

La **description** inclut le modèle correct, les dictionnaires de données et la documentation à la fois du modèle et des hypothèses sur le sujet et des choix de modélisation.

### 1.6. - Raffinement d'un MCD en MR

Lorsque les données du système d'information sont mises en œuvre, de manière logique, dans une base de données relationnelle on utilise une approche assez systématique de traduction « EAP → MR ». Elle se fait en deux étapes :

- (i) génération du modèle,
- (ii) optimisation du modèle.

Les règles de base de l'étape (i) sont :

1. Chaque TE devient une relation. Attention, en fin de processus, cette table peut disparaître lors de l'optimisation. Chaque propriété d'un TE devient un attribut de la relation. L'identifiant devient la clé primaire de la relation.
2. Un TA unaire ou binaire avec au moins une cardinalité maximale à 1 du côté TE1 (appelé fonction) donne lieu à une copie de la clé de la relation issue de TE2 dans la relation issue de TE1. Pour conserver la traçabilité avec le MCD, cette copie est suffixée par le nom de l'association. Les éventuelles propriétés du TA deviennent des attributs dans la relation issue de TE2. Noter que si la cardinalité est (0,1) l'attribut est optionnel, et l'accès à l'attribut devra prendre en compte l'absence de valeur.
3. Tout autre TA se transforme en une relation dont la clé est l'union des clés issues des identifiants des TE impliqués dans le TA. Chaque propriété du TA devient un attribut de la relation.
4. Les contraintes sont à étudier. Certaines contraintes peuvent faire l'objet d'instructions dans la clause « Constraint » de SQL ou dans des procédures stockées de SQL.

Pour l'étape (ii) d'optimisation, mentionnons qu'une relation mono-attribut n'est pas toujours conservée. Il y a d'autres règles.

L'application à notre exemple fil rouge produit le modèle relationnel de la figure 5. Les trois TE (*Personne*, *Association*, *Activité*) sont transformés en relations. Quatre TA (*president*, *tresorier*, *responsable*, *propose*) sont transformés selon la règle 2 relative aux fonctions. Le TA membre est transformé selon la règle 3 et donne donc lieu à la relation *Membre*. Le résultat est équivalent à celui de la figure 1 obtenu par la méthode de conception par normalisation, au renommage près des relations et attributs.

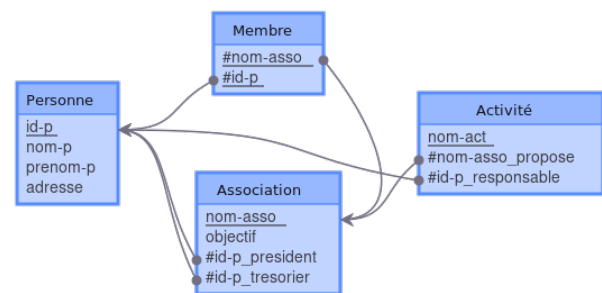


Figure 5 : Transformation MCD → MR du cas Asso.

Le passage inverse « MR → EAP », qualifié de **rétro-ingénierie**, est moins fréquent et plus difficile à pratiquer, car l'abstraction est plus ouverte que le raffinement. Il s'applique quand on reprend un système d'information informatisé (SII) existant.

### 1.7. - Outillage de génie logiciel

Si, dans l'apprentissage, les étudiants décrivent leurs modèles de manière manuscrite, l'outillage d'édition des modèles s'avère indispensable à la fois pour la lisibilité, la correction, mais surtout l'évolution des modèles. Il existe de nombreux outils gratuits ou payants plus ou moins sophistiqués (édition, vérification, validation, transformations, travail colla-

---

**MODÉLISATION ASSISTÉE PAR L'IA  
POUR L'ENSEIGNEMENT DES  
BASES DE DONNÉES**


---

boratif...). Un outil comme *WinDesign*<sup>6</sup> permet par exemple de construire le modèle relationnel à partir du MCD. Nous suggérons ici des éditeurs gratuits simples et accessibles comme *Looping-MCD*<sup>7</sup>, mais aussi *Mocodo*<sup>8</sup> — que nous avons utilisé pour cette section —, *DBConcept*<sup>9</sup> ou *PlantUML*<sup>10</sup> qui présentent l'avantage de travailler sur du texte et donc de ne pas perdre de temps sur la disposition visuelle du modèle (*layout*).

Par exemple, le modèle de la figure 2 est décrit par le texte suivant en *Mocodo/MCD* :

```

::
Personne : id-p, nom-p, prenom-p, adresse
:
responsable, 0N Personne, 1I Activite
:
president, 1I Association, 0N Personne
:
membre, 0N Association, 0N Personne
:
tresorier, 1I Association, 0N Personne
:
Activite : nom-act
::
Association : nom-asso, objectif
:
propose, 0N Association, _1I Activite
:
(I) --tresorier, ->membre
(I) --president, ->membre
(X) --president, --tresorier, responsable

```

Pour le modèle relationnel, il est possible d'utiliser le langage *PlantUML/EID*. Ce qui donne, sur notre exemple :

```

@startuml
hide circle

entity "Personne" {

```

<sup>6</sup> <https://www.win-design.com/>

<sup>7</sup> <https://www.looping-mcd.fr/>

<sup>8</sup> <https://www.mocodo.net/>

<sup>9</sup> <https://dbconcept.forge.apps.education.fr/dbconcept/>

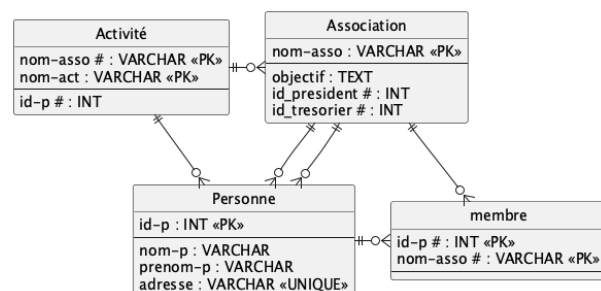
<sup>10</sup> <https://plantuml.com/fr/>

```

id-p : INT <<PK>>
--
nom-p : VARCHAR
prenom-p : VARCHAR
adresse : VARCHAR <<UNIQUE>>
}
entity "Association" {
nom-asso : VARCHAR <<PK>>
--
objectif : TEXT
id_president # : INT
id_tresorier # : INT
}
entity "Activité" {
nom-asso # : VARCHAR <<PK>>
nom-act : VARCHAR <<PK>>
--
id-p # : INT
}
entity "membre" {
id-p # : INT <<PK>>
nom-asso # : VARCHAR <<PK>>
--
}
Personne ||--right-o{ membre
Association ||--o{ membre
Activité ||--right-o{ Association
Activité ||--o{ Personne
Association ||--o{ Personne
Association ||--o{ Personne
@enduml

```

Cet exemple donnera le modèle produit par *PlantUML* en figure 6. De plus, par leur nature, ils sont bien plus « interopérables » avec d'autres outils de traitement automatisés (pas d'import/export sophistiqués et légers, car le texte n'occupe que quelques octets contrairement aux modèles XML ou, pire, aux images).



**Figure 6 :** Modèle relationnel en PlantUML.

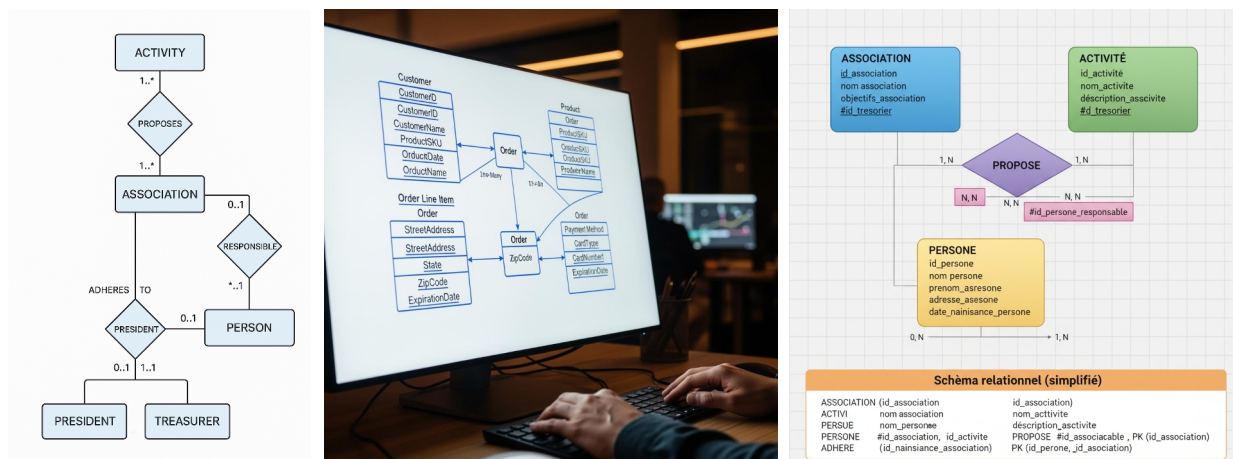
## 2. – Utilisation d'IA génératives

### 2.1. - Les SAIg

Le terme *SAIg* recouvre un ensemble d'outils basés sur des modèles de langage génératif dont le principe est de produire un résultat (texte, image, vidéo...) à partir d'une description (texte, image...). Cette génération s'effectue en utilisant des systèmes à apprentissage profond, donc probabilistes.

Dans la suite, nous nous intéresserons uniquement aux systèmes générant du texte à partir de textes. En effet, les cas d'usage qui

nous intéressent ici sont de produire un modèle à partir d'un texte. Les outils de modélisation habituellement utilisés sont souvent des diagrammes (Merise ou UML par exemple). Il serait donc tentant d'utiliser des systèmes de génération d'images à partir de textes. Cependant, la génération de textes et la génération d'images fonctionnent de manière différente. La génération d'image, qu'elle soit par diffusion ou antagoniste, va chercher à produire une image du système, à un coût écologique important, souvent de très mauvaise qualité et quasiment impossible à modifier. La figure 7 propose trois exemples.



GPT-image-1 - nov 2025.

Imagen-4-Ultra - nov 2025.

Nano-Banana - nov 2025.

Figure 7 : Exemples de modèles générés sous forme d'images.

Inversement, il existe des outils numériques de transformation automatique de modèles textes en modèles diagrammes (image). En conséquence, une utilisation rationnelle de l'IA pour construire le modèle, dans un langage dédié permettra d'utiliser les outils pour transformer ces modèles textuels en diagrammes sous forme d'images. On trouvera par exemple *PlantUML* pour UML ou *Mocodo* pour les modèles de données Merise. Nous avons donc fait le choix de générer les modèles conceptuels en *Mocodo/MCD* (Merise)

ou *PlantUML* (UML). Les modèles relationnels seront en *Mocodo/MLD*, *PlantUML* ou simplement en SQL/DDL<sup>11</sup>.

### 2.2. - Les SAIg textuels

Le principe de base de ce type de système est assez simple. Compte tenu d'un texte existant, quelle va être la suite à proposer ? Les

<sup>11</sup> SQL Data Description Language [https://fr.wikipedia.org/wiki/Langage\\_de\\_d%C3%A9finition\\_de\\_donn%C3%A9es](https://fr.wikipedia.org/wiki/Langage_de_d%C3%A9finition_de_donn%C3%A9es)

systèmes ne sont pas, la plupart du temps, déterministes. Ils génèrent un ensemble de suites possibles (avec un degré de probabilité pour chacune) et en choisissent une parmi les plus probables. Les suites sont déterminées après apprentissage sur de très gros corpus d'informations généralistes ou spécialisées.

Ce choix est paramétré par une valeur appelée la température. Plus cette valeur est haute (entre 0 et 1.5) plus le choix de suites possibles est grand. Donc pour une valeur à 0, le système prendra systématiquement la valeur la plus probable et sera donc quasiment déterministe. Pour des valeurs plus hautes, le système ne sera plus déterministe. Pour plusieurs demandes identiques, le résultat pourra être très différent. Pour des valeurs très hautes, le résultat deviendra incompréhensible.

Ainsi, le texte en entrée est construit à partir d'un ensemble de sources :

- le **prompt d'instruction**, produit par l'organisation qui propose le système et qui donne les grandes règles de fonctionnement ;
- le **prompt système**, proposé par l'utilisateur, souvent dans un menu de configuration de l'outil, permet de spécifier le style de la réponse (langage, détail, rôle...);
- le **prompt utilisateur**, proposé par l'utilisateur, c'est l'élément de dialogue courant ;
- les **échanges précédents** (prompts utilisateurs précédents et réponses du système).

Le texte en entrée est contraint par une « fenêtre contextuelle » dont la taille peut varier de 2048 *token* à plus de 10 millions de *token*. Un *token* est une unité élémentaire de texte pour le système. Les grandes fenêtres permettent de mieux gérer les longs documents et les conversations étendues, mais augmentent

les besoins en calcul et parfois les hallucinations<sup>12</sup>, ainsi que le temps de réponse.

Les SAIG sont souvent décrits par plusieurs informations importantes pour le choix :

- la taille de la fenêtre de contexte ;
- le nombre de paramètres, c'est-à-dire le nombre de valeurs gérées par le modèle (poids de neurones...);
- la température...

### 2.3. - Choix des SAIG

Depuis l'automne 2022, un très grand nombre de SAIG ont été proposés. À notre connaissance, il n'existe pas de système spécialisé sur la modélisation de données. Nous avons donc été contraints d'utiliser des systèmes généralistes. Ces derniers peuvent être scindés en deux catégories :

- les **modèles larges** ou *Large Language Models* (LLM) qui ont un très grand nombre de paramètres (taille du modèle : 70 à plus de 1760 milliards), une fenêtre contextuelle importante, un corpus d'apprentissage très important et qui fonctionnent sur des machines très puissantes ;
- les **modèles petits** ou *Small Language Models* (SLM) qui ont une taille modeste (200 millions à 20 milliards de paramètres), une fenêtre contextuelle petite et qui peuvent fonctionner sur de plus petites machines, voire sur des ordinateurs de bureau ou des ordinateurs portables.

Il faut noter que, fréquemment, les SLM sont des versions « allégées » de LLM. Il arrive aussi que leur corpus d'apprentissage comprenne des données « artificielles », c'est-à-dire des jeux de données produites par le LLM associé, avec donc un risque d'amplification des biais et hallucinations.

<sup>12</sup> Une réponse fautive ou trompeuse qui est présentée comme un fait certain.

## 2.4. - Extensions de SAIG

Ces systèmes sont parfois étendus. On peut évoquer :

1. les **RAG** (Génération à Enrichissement Contextuel en français). Pour ces outils, on associe au système classique une base documentaire qui est indexée. Au moment du traitement du prompt, un pré-traitement ajoute des extraits significatifs pour améliorer le contexte de la demande ;
2. l'ajout de **systèmes externes** (mail, base de données...), via un protocole standard. Cela permet d'ajouter des sources de données qui seront prises en compte dans le traitement du prompt ;
3. le **raisonnement**. Par le passé technique de *prompting* (technique de chaîne de pensée ou COT), le raisonnement est maintenant intégré dans la phase de traitement du prompt, surtout pour les LLM. Cette phase est méthodologique. Le système ajoute une étape de la forme « expliciter la manière d'obtenir une réponse »...

Dans le cadre de ce travail, nous ne prendrons en considération que la technique de RAG, même si certains systèmes intègrent aussi le raisonnement (*Claude 4* ou *GPT5* par exemple).

## 2.5. - Outils logiciels

Tous les modèles de langage sont accessibles individuellement ou à travers des « agrégateurs de modèles ». On trouve par exemple :

- *POE*<sup>13</sup> (modèles variés, USA) ;
- *BlueGPT*<sup>14</sup> (modèles variés, FR) ;
- *OpenRouter*<sup>15</sup> (API pour différents modèles, USA) ;

<sup>13</sup> <https://poe.com/>

<sup>14</sup> <https://bluegpt.app/fr>

<sup>15</sup> <https://openrouter.ai/>

- *Hugging Face*<sup>16</sup> (modèles open source, USA/FR).

Les gros modèles testés dans le cadre de notre étude l'ont été à travers *POE* (pour des questions budgétaires et techniques).

Ces agrégateurs proposent aussi des SLM. Mais, afin d'être dans des conditions intéressantes (protection de données personnelles, coût écologique, charge de réseau et latence), nous les avons, le plus souvent, chargés sur des outils locaux comme :

- *Ollama*<sup>17</sup> : outil rapide et léger, il permet d'utiliser de nombreux modèles SLM Open Source soit en ligne de commande soit par interface (assez récemment). Assez léger, il peut être installé même sur des micro-PC, mais ne propose pas le mode RAG ;
- *GPT4ALL*<sup>18</sup> : application assez simple pour des non spécialistes, il propose un catalogue de modèles un peu restreint et propose le mode RAG ;
- *LM Studio*<sup>19</sup> : application plus technique (mais possède différents modes d'utilisation assez pratiques), mais qui propose un catalogue de modèles important (pris principalement sur Hugging Face) et le mode RAG.

Dans nos expérimentations, nous avons utilisé les trois outils. Ils ont aussi l'avantage de permettre un réglage fin des modèles, en particulier pour la température.

<sup>16</sup> <https://huggingface.co/>

<sup>17</sup> <https://ollama.com/>

<sup>18</sup> <https://www.nomic.ai/gpt4all>

<sup>19</sup> <https://lmstudio.ai/>

## 2.6. - Questionnements éthiques et déontologiques

L'usage des SAIG pose des questions éthiques et déontologiques importantes qu'il ne faut pas éluder (Higuera et Iyer, 2024). De nombreuses institutions nationales et internationales s'en préoccupent (Nantes Université, 2025 ; CNIL, s.d. ; UNESCO, 2023 ; Germain et al., 2022). Ces questionnements sont sur deux niveaux :

- le choix de l'outil ;
- l'usage que l'on en fait.

Ces préoccupations sont fondamentales, mais nous n'entrerons pas ici dans le détail. Nous aborderons seulement les questions importantes pour la suite de notre présentation.

Le premier niveau questionne **le corpus d'apprentissage utilisé et la méthode d'apprentissage choisie**. Le corpus d'apprentissage est important pour comprendre certains biais (par exemple culturels) et certaines hallucinations. En particulier, beaucoup d'outils ont des corpus à très grande majorité anglophones. Ainsi, les approches pédagogiques et didactiques sont naturellement plus dans la tradition anglo-saxonne que française. Ces biais sont possiblement renforcés pour les SLM qui, nous l'avons dit, ont une partie de leur apprentissage issue des LLM tout en ayant moins de paramètres (et donc une fiabilité plus faible). Il faut ajouter à cela le coût en ressources naturelles pour la phase d'apprentissage (parfois plusieurs semaines sur des gros calculateurs pour des téraoctets de données) et la validation de la qualité des résultats par des « experts » exploités durant les phases de paramétrage et d'ajustement de l'outil.

Le second niveau questionne **la manière d'utiliser l'outil et les informations que l'on met en jeu**. En effet, les outils proposés exploitent (avec ou sans notre consentement) les

informations mises en jeu dans les prompts. Il convient donc de faire extrêmement attention aux données personnelles (les nôtres ou celles de nos élèves ou étudiants), aux données sensibles et au droit d'auteur. Ce niveau interroge aussi notre capacité à vérifier le résultat proposé. En effet, souvent caricaturés comme étant des « perroquets stochastiques », il ne reste pas moins que le résultat produit par ces outils est issu d'un processus probabiliste compilant des sources diverses et pas toujours cohérentes. L'illustration habituelle est la difficulté qu'ont ces outils à proposer des bibliographies réelles et pas seulement réalistes. Nous en avons fait l'expérience avec nos doctorant·es, les citations paraissaient à première vue crédibles, mais étaient en réalité erronées. Des prénoms étaient parfois inventés (l'IA inférait un prénom à partir d'initiales de prénoms) ou les titres n'étaient pas les bons. En effet, ils combinent des prénoms, des noms et des conférences existants dans le domaine pour en faire des entrées bibliographiques bien écrites, mais fausses. De plus, ils sont conditionnés pour aller dans le sens de l'utilisateur et non dans le sens de la vérité scientifique.

Pour toutes ces raisons, les institutions proposent des cadres d'usage de ces systèmes. C'est le cas, en France, du cadre du ministère de l'éducation nationale (MENESR, 2025). Ce cadre pose un certain nombre de consignes dont les suivantes :

1. Il encourage une utilisation comme **assistant de manière responsable**. En particulier, l'IA ne doit être utilisée que si aucune autre solution moins coûteuse écologiquement ne répond de façon satisfaisante au besoin.

2. Il impose de **respecter le cadre légal sur la protection des données à caractère personnel** (RGPD) et de n'utiliser que des données qui peuvent être rendues publiques. Il n'est pas possible de demander aux élèves de créer des comptes à titre personnel.
3. Il demande un **usage transparent** (communication explicite sur son rôle et la façon dont elle a été utilisée). Toute décision s'appuyant sur l'IA et ayant un impact sur l'évaluation ou le parcours des élèves, doit faire l'objet d'une supervision humaine garantissant son usage équitable et son explicabilité. Il doit aussi faire l'objet d'une validation préalable par l'autorité compétente.
4. L'usage de l'IA doit être **en assistance** et non en substitution des apprentissages et de l'effort intellectuel. Il doit être encadré et accompagné par l'enseignant.
5. L'usage des IA « grand public » est possible (en connaissance de cause).

Nous notons que beaucoup de ces consignes relèvent du droit, du bon usage du numérique ou de la déontologie de l'enseignement. Elles ne sont pas spécifiques aux SAIg.

Ces différents éléments ont servi de support à notre travail d'évaluation. En attendant une éventuelle « IA souveraine » imposée par l'institution (en tout cas en France), nous allons proposer d'utiliser les LLM et les SLM actuels en tant qu'assistants pour la manipulation de modèles de données. En dehors du contexte proposé qui doit être original ou utilisé dans le respect des licences attachées, nos propositions respectent bien le cadre proposé. Seul le critère de « coût écologique » peut être à juste titre questionné. C'est pour cela que nous proposerons différents outils, dont des SLM en local (pour d'éventuelles données sensibles et pour leur usage *a priori* plus écologique), même s'ils sont plus délicats à utiliser (installation, paramétrage, *prompting*...).

## 2.7. - Modèles « open source »

Dans la technosphère des IA génératives, il est souvent question de modèles de langage open source. Il convient de préciser que cet adjectif recouvre plusieurs notions. La notion la plus courante est celle concernant les paramètres du modèle (les poids des neurones). Il est plus rare que les autres données soient accessibles. Certains modèles comme *Lucie (Open LLM, France)* ou *Apertus (EPFL, ETH Zurich et le CSCS, Suisse)* sont plus en avance puisqu'ils donnent accès aussi aux données d'apprentissage et aux processus de construction du modèle.

Il convient aussi d'être vigilant sur la licence d'utilisation du modèle qui est, parfois, loin de la philosophie de l'open source ou de l'open data.

## 3. – Analyse de modèles existants et expérimentation des outils d'aide

Cette section résume un travail expérimental d'analyse de l'existant mené lors d'un stage par Loïc Kodo en 2025 (Kodo, 2025). L'objectif principal était d'étudier la modélisation des bases de données dans un cadre didactique, afin d'accompagner les enseignants de NSI dans la conception de modèles adaptés à leurs besoins pédagogiques. Cette étude s'est organisée en deux grandes phases : d'une part, l'analyse des modèles proposés dans les sujets officiels (bac NSI, CAPES, agrégation) pour en évaluer la pertinence ; d'autre part, l'évaluation de la capacité de certains LLM à générer des modèles de qualité susceptibles d'être utilisés par les enseignants.

### 3.1. - Analyse de modèles existants

Cette phase consiste à analyser tous les modèles proposés dans les sujets d'exams du bac NSI (2021-2024), du CAPES NSI

(2020-2025) et de l'agrégation d'informatique (2022-2024). Cette analyse a été faite au regard des critères (André et Vailly, 2001 ; Gardarin, 2003) permettant la mise en place d'une grille d'analyse en tenant compte de leur structure, de leur complexité et de leur impact sur l'apprentissage (notamment pour les requêtes SQL). La grille a donc été remplie pour chaque modèle et une synthèse d'analyse faite.

On souhaite comprendre comment la modélisation des bases de données est abordée dans ces différents contextes d'évaluation, identifier les éventuelles limites ou incohérences des modèles proposés et synthétiser ces analyses dans des fiches structurées. Pour chaque sujet analysé, trois étapes étaient systématiquement menées. D'abord, l'identification des problèmes ou incohérences présents dans le modèle fourni ou induits par l'énoncé. Ensuite, la réalisation d'un modèle via rétro-ingénierie, fondée sur les données et exemples disponibles. Enfin, la proposition d'un modèle élaboré prenant en compte le contexte global du sujet, en formulant explicitement les hypothèses retenues lorsque certaines informations manquaient ou restaient ambiguës. Ces éléments étaient ensuite synthétisés dans une fiche d'analyse comprenant :

- une liste des problèmes identifiés ;
- les deux versions du modèle (rétro-construit et contextuel) ;
- une brève comparaison entre les deux ;
- une conclusion rapide sur les implications pédagogiques ou conceptuelles.

La réalisation des modèles a nécessité l'expérimentation de plusieurs outils avant d'aboutir à une solution satisfaisante avec *DBConcept* dont le langage reprend *Mocodo*, tant en termes de lisibilité que de souplesse.

### *Analyse de sujets du Baccalauréat NSI*

Les sujets du bac NSI constituent la majorité du corpus analysé, représentant environ 80 % des sujets étudiés. Cette proportion s'explique par la multiplicité des zones géographiques, des sessions (normales et de remplacement) et des jours de composition, qui génèrent chaque année un grand nombre de sujets distincts. Au total, 64 sujets de bac ont été collectés et analysés. Pour chaque sujet, une fiche d'analyse a été produite. Celles-ci ont conduit à la génération d'environ 130 modèles valides, en comptant les différentes versions rétro-construites et contextuelles. Si l'on ajoute les essais, les erreurs et les changements d'outil, c'est près de 200 modèles qui ont été conçus uniquement pour cette catégorie. L'analyse des sujets de bac a été particulièrement formatrice, notamment lors de la première série de traitements, en 2021. Certains sujets se sont distingués par leur complexité ou leur manque de clarté. Le cas le plus emblématique est celui du sujet Amérique du Nord 1 - 2021, dont l'ambiguïté a rendu la rétro-ingénierie impossible. Le sujet ne permettait pas d'extraire un modèle cohérent en raison de problèmes de normalisation.

Parmi les problèmes fréquemment rencontrés dans les sujets de bac, on peut citer :

- l'absence de définition explicite des clés primaires, parfois volontaire, pour alimenter des questions de réflexion, mais souvent source d'ambiguïtés ;
- la confusion entre entités et associations, ajoutant des contraintes supplémentaires ;
- les adresses gardées comme attributs enfreignent ainsi la première forme normale car l'attribut adresse contient une liste d'informations ;
- des structures incohérentes ou incomplètes, ne permettant pas toujours une

modélisation fidèle sans ajout d'hypothèses.

Ces observations permettent d'avoir une vision sur la conception de ces épreuves et la manière dont la modélisation est évaluée à ce niveau d'enseignement.

### *Analyse des sujets d'écrit du CAPES*

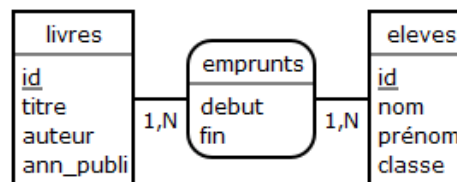
L'analyse des sujets du CAPES NSI a constitué une étape importante du travail, dans la mesure où ces épreuves ne s'adressent plus à des élèves, mais à des candidats à l'enseignement de l'informatique. Par conséquent, les exigences en matière de rigueur, de structuration et de justification des modèles sont, en théorie, significativement plus élevées.

Bien que le volume de sujets soit nettement inférieur à celui des épreuves du baccalauréat, leur profondeur et leur complexité justifient un traitement approfondi. Au total, sept sujets officiels ont été analysés, couvrant les sessions de 2021 à 2024 pour les épreuves 1 et 2, certaines années proposant plusieurs exercices de modélisation. Ainsi, environ 70 % des sujets contiennent au moins un exercice portant sur les bases de données. Pour chacun de ces sujets, une fiche d'analyse a été produite selon une méthodologie structurée : identification des problèmes posés, application d'une démarche de rétro-ingénierie, modélisation contextuelle incluant les hypothèses nécessaires, puis élaboration d'une synthèse.

Les sujets du CAPES se distinguent par une approche plus technique et plus exigeante. Les énoncés y sont souvent plus détaillés, et les exercices de modélisation s'inscrivent dans un cadre plus large, incluant la description de systèmes, l'analyse de besoins ou la conception d'algorithmes liés aux données. Toutefois, malgré ces attentes élevées, plusieurs limitations ont été relevées :

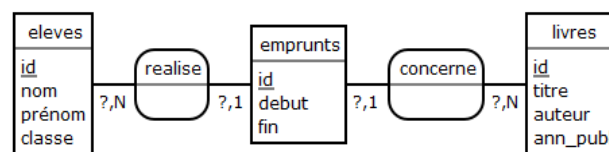
- des modèles partiels ou implicites, contraignant les candidats à formuler de nombreuses hypothèses ;
- des relations floues ou mal définies entre entités ;
- la présence de redondances ou d'ambiguïtés dans la structuration des données.

Un sujet en particulier s'est démarqué : l'épreuve 1 de la session 2021. Même dans sa version corrigée, le modèle relationnel proposé ne correspond pas au modèle conceptuel attendu à partir de l'énoncé. Le sujet propose le modèle de la figure 8.



**Figure 8** : MCD du sujet CAPES 2021.

La rétro-ingénierie aurait dû donner un modèle équivalent mais on obtient celui de la figure 9.



**Figure 9** : MCD rétro-conçu du sujet CAPES 2021.

Cela met en lumière un certain manque de rigueur de la part des concepteurs, du moins sur les exercices de bases de données.

Environ vingt modèles ont été générés à partir des sujets du CAPES, avec quelques ajustements nécessaires en raison d'erreurs initiales ou de reformulations successives. Ces

modèles ont permis d'explorer des cas plus complexes que ceux du baccalauréat, incluant des structures de données plus riches et des contraintes plus fines.

Ce travail a également contribué à mieux cerner les attentes implicites du concours en matière de modélisation, souvent absentes ou peu explicitées dans les rapports officiels du jury.

### *Analyse des sujets de l'agrégation*

L'analyse des sujets de l'agrégation d'informatique, bien que portant sur un nombre limité d'épreuves, s'est révélée particulièrement riche sur le plan conceptuel. Ce concours, destiné à un public déjà fortement qualifié, exige une maîtrise approfondie des concepts informatiques, notamment en matière de modélisation des bases de données.

Sur les quinze sujets disponibles, seuls cinq comportent au moins un exercice ou une section traitant des bases de données, répartis entre les sessions 2021 et 2025. Cela représente environ 33 % des sujets. Les sujets d'agrégation se distinguent par leur densité conceptuelle et leur ancrage dans des problématiques techniques complexes : réseaux, systèmes distribués, architectures logicielles, etc. Les cas de modélisation s'inscrivent dans des contextes plus exigeants, nécessitant une approche fine et rigoureuse.

Dans ces épreuves, la modélisation ne constitue pas un objectif isolé ; elle s'intègre dans une démarche globale, incluant l'analyse de systèmes, la conception de bases relationnelles ou la justification de choix d'implémentation. Il est attendu des candidats qu'ils mobilisent les règles classiques de modélisation tout en les adaptant aux contraintes spécifiques de chaque situation.

L'analyse de ces sujets a soulevé plusieurs difficultés :

- les sections portant sur les bases de données ne sont pas isolées, mais intégrées à un tout qu'il convient de comprendre dans sa globalité ;
- des problématiques similaires à celles rencontrées dans d'autres concours subsistent, telles que des erreurs sur les clés primaires ou des violations de la première forme normale (1FN).

Comme pour les autres examens, chaque sujet a fait l'objet d'une fiche d'analyse incluant une démarche de rétro-ingénierie (lorsqu'elle était applicable) ainsi qu'une modélisation contextuelle accompagnée d'hypothèses explicites. Au total, onze modèles ont été produits dans le cadre de cette analyse. Leur élaboration a exigé un effort d'interprétation accru, menant souvent à des choix de modélisation plus nuancés, notamment sur les questions de cardinalités, de dépendances fonctionnelles et de contraintes spécifiques au domaine d'application. Ce travail sur les sujets de l'agrégation a constitué un approfondissement significatif, en offrant l'occasion d'aborder la modélisation dans des contextes où les outils standards ne suffisent pas toujours à représenter fidèlement la complexité du réel.

### *Synthèse des analyses*

À l'issue de l'analyse détaillée des trois corpus (bac NSI, CAPES NSI et agrégation d'informatique), une synthèse comparative a été réalisée afin d'évaluer la manière dont la modélisation des bases de données est abordée selon le niveau de l'examen, les attentes pédagogiques et la qualité formelle des énoncés. Cette synthèse permet non seulement de dresser un état des lieux global, mais aussi de faire émerger des points d'attention en vue de l'amélioration des sujets futurs et de la conception d'outils pédagogiques adaptés.

### *Baccalauréat NSI*

Le volume important de sujets disponibles permet une analyse statistiquement représentative. Il en ressort que :

- les erreurs techniques sont fréquentes : attributs composites, absence de clé primaire, confusion entre entité et association ;
- les énoncés sont souvent trop ouverts ou imprécis, nécessitant une reformulation par le candidat ;
- la modélisation est généralement traitée comme un exercice ponctuel, parfois même déconnecté du reste du sujet.

Ce type de sujet met davantage l'accent sur l'intention pédagogique (identifier des erreurs, compléter un modèle) que sur la validité ou la cohérence des modèles proposés.

### *CAPES NSI*

Les modèles des sujets du CAPES ressemblent en apparence à ceux du baccalauréat, leurs objectifs diffèrent profondément : il ne s'agit plus simplement d'appliquer des connaissances, mais de démontrer une capacité à les transmettre et à en comprendre les enjeux didactiques... Cependant :

- les modèles sont parfois mal spécifiés, laissant planer des incertitudes sur l'intention du concepteur ;
- les candidats doivent souvent interpréter de manière autonome des situations mal formalisées ;
- certains sujets présentent des ambiguïtés problématiques, peu cohérentes avec le niveau d'exigence attendu.

On observe ici une transition entre une approche scolaire et une approche professionnelle, sans que la rigueur formelle soit pleinement assurée.

### *Agrégation d'informatique*

Les sujets d'agrégation se distinguent nettement :

- la modélisation est intégrée dans une problématique globale, rarement traitée isolément ;
- les cas sont cohérents mais techniques, nécessitant une forte capacité de contextualisation ;
- les candidats sont implicitement évalués sur leur aptitude à adapter les outils conceptuels à des situations complexes.

Il s'agit ici d'un niveau d'expertise avancé, où la maîtrise du formalisme est présumée, mais rarement guidée ou explicitement corrigée.

### *Synthèse*

Cette analyse comparative met en évidence les points suivants :

- plus le niveau de l'examen est élevé, plus la modélisation devient complexe, contextualisée et intégrée à une problématique métier ;
- les erreurs ou imprécisions ne disparaissent pas avec le niveau, mais changent de nature : elles évoluent de fautes formelles (bac) à des choix d'interprétation ambigus (CAPES, agrégation) ;
- il existe un manque global d'harmonisation dans la manière d'introduire et d'évaluer la modélisation, ce qui complique la comparaison des compétences attendues d'un niveau à l'autre.

Ces constats justifient pleinement l'intérêt de développer des outils d'aide à la modélisation ou de vérification de modèles, afin de mieux accompagner les concepteurs de sujets comme les apprenants.

### *Source d'apprentissage pour les IAG*

Ces expérimentations montrent que les sujets et corrigés peuvent servir de source à des modèles d'apprentissage, mais une qualification manuelle d'un niveau de qualité est un prétraitement nécessaire pour améliorer la qualité de l'apprentissage.

### **3.2. - Expérimentations des modèles de langage**

Cette phase vise à évaluer la pertinence des modèles de langage en tant qu'outils d'assistance pour les enseignants de NSI ainsi que pour les concours de recrutement. L'objectif est double. D'une part, identifier les modèles de langage les mieux adaptés à la génération de schémas relationnels. D'autre part, tester concrètement leurs capacités à traiter des sujets d'examen réels.

Les outils logiciels utilisés pour cette phase sont *Ollama* et *LM Studio*. Cette phase se décline en trois étapes principales :

1. le recensement des modèles de langage disponibles ;
2. l'évaluation de leurs capacités d'analyse et de production ;
3. une application pratique sur des sujets d'examen sélectionnés.

#### ***Recensement de modèles de langage***

L'étape initiale a consisté à identifier et sélectionner des modèles de langage pertinents pour la génération automatique de schémas relationnels. L'objectif était de constituer une base de modèles à tester, en s'appuyant sur des critères spécifiquement adaptés aux besoins pédagogiques. Les critères de sélection retenus sont les suivants :

- capacité à produire des modèles : le LLM doit être capable de générer correctement les tables, identifiants, relations,

tout en respectant les principes de normalisation ;

- pertinence pédagogique : les réponses doivent être adaptées à différents niveaux d'enseignement, allant du lycée à la préparation aux concours ;
- accessibilité : priorité aux modèles open source ou accessibles via une API gratuite, afin de garantir un usage facilité dans un contexte éducatif ;
- qualité des résultats : conformité aux formes normales (1NF, 2NF, 3NF) et aux standards reconnus de modélisation ;
- performance : faible latence et exigences limitées en ressources, pour permettre une intégration fluide dans un outil pédagogique ;
- éthique : respect des principes de transparence, d'équité et de non-biais dans les réponses fournies.

Après un premier filtrage, les modèles retenus pour cette l'évaluation sont les SLM suivants :

- *Gemma3* : modèle de langage polyvalent développé par *Google* ;
- *Mistral* : modèle open-source performant créé par *Mistral AI* ;
- *StarCoder2* : modèle spécialisé en génération de code développé par *BigCode* (open source) ;
- *CodeLlama* : modèle de code basé sur *LLaMA*, développé par *Meta AI*.

#### ***Étude des capacités d'analyse et de production des SLM***

Une fois les modèles recensés, nous avons évalué leur performance sur des tâches concrètes de modélisation à partir de descriptions textuelles. Pour cela, une série de tests a été menée sur un nombre restreint de sujets choisis, avec pour objectifs :

- déterminer la capacité des SLM à produire plusieurs formats de schéma

(UML, ER, EAP) pour un même énoncé ;

- comparer les résultats selon la nature des descriptions (non structurées d'abord, puis structurées) ;
- affiner la structuration des prompts et mesurer leur efficacité.

Dans un premier temps, différents types de prompts ont été expérimentés via *LM Studio*, dont l'interface graphique a simplifié les échanges. Ces tests préliminaires ont permis d'identifier les formulations les plus facilement interprétées par les modèles. À partir de ces observations, nous avons élaboré un prompt optimisé, puis automatisé les évaluations grâce à un script exécuté sur *Ollama*. Cette automatisation s'est révélée indispensable, car les essais manuels sur *LM Studio* se sont avérés trop gourmands en temps et répétitifs.

Une variante supplémentaire du prompt a été rédigée entièrement en langage naturel car cela correspond au mode d'entraînement des SLM. De nouveaux tests ont ainsi comparé les deux approches (structurée vs naturelle). Concernant les formats de sortie, le format EAP a rapidement été abandonné : même s'il est toujours utilisé, il s'agit d'un format ancien et complexe pour les LLM (et pour l'analyse automatisée de leurs réponses). L'étude s'est donc recentrée ici sur les formats UML et ER, plus accessibles et mieux adaptés au contexte pédagogique.

Enfin, nous avons inversé la chaîne de traitement pour tester l'inverse : à partir d'un schéma écrit (au format UML/ER), demander au SLM de générer l'énoncé descriptif correspondant. Deux scripts ont été conçus pour cette phase, l'un en langage naturel, l'autre en format structuré, afin de comparer la fidélité et la pertinence des textes produits...

Au terme de ces tests, on peut retenir que la qualité des résultats dépend fortement du prompt : les instructions structurées donnent des schémas plus précis, tandis que le langage naturel produit des réponses plus fluides. Les SLM, excellents pour générer des diagrammes UML, sont corrects en ER et ont beaucoup plus de difficultés en EAP. Ils peinent sur les énoncés ambigus qui ne contiennent pas assez de détails et commettent souvent des oublis ou des erreurs de cardinalités, même si un prompt bien conçu les aide à formuler des hypothèses sensées. Enfin, l'automatisation via un script *Ollama* a standardisé les tests, fait gagner du temps et assuré la traçabilité. Sur cette base, nous pouvons passer à l'application sur des sujets réels.

### *Application aux sujets sélectionnés*

Une application concrète des modèles de langage a été réalisée sur un ensemble de sujets sélectionnés parmi les corpus étudiés (bac, CAPES, agrégation). L'objectif était d'évaluer leur utilité dans un contexte réel d'enseignement. Pour ce faire, la démarche suivante a été adoptée : la sélection de sujets disposant d'une description textuelle exploitable, la soumission des énoncés aux SLM sélectionnés, la génération des modèles comprenant relations, tables, clés et contraintes, puis enfin la comparaison des modèles produits automatiquement avec ceux obtenus manuellement lors des phases précédentes. L'évaluation s'appuie sur plusieurs critères :

- conformité aux normes relationnelles : les modèles doivent respecter les règles théoriques fondamentales des bases de données, notamment la normalisation (1NF, 2NF, 3NF), la définition claire des clés primaires et étrangères, et la cohérence des relations entre entités, afin d'assurer une structure fiable et cohérente ;

- niveau de complexité adapté à l'épreuve : les modèles doivent refléter un degré de difficulté correspondant au niveau attendu des candidats, en évitant à la fois une trop grande simplicité qui ne stimule pas la réflexion, et une complexité excessive qui pourrait rendre l'exercice inabordable ;
- utilité pédagogique : il s'agit de produire des modèles clairs, compréhensibles et facilement interprétables, qui facilitent l'apprentissage des concepts de modélisation, et qui peuvent être adaptés aux différentes approches pédagogiques ou contextes d'exercice.

Pour l'autre cas d'usage, qui concerne la génération du texte descriptif d'un modèle, la méthodologie est la suivante :

- sélection de quelques modèles pertinents : choisir des modèles représentatifs et variés, issus des analyses précédentes, afin de tester la capacité des SLM à générer des descriptions précises et complètes à partir de cas concrets ;
- soumission du modèle sous forme textuelle (format *Mocodo*) aux SLM : fournir la représentation textuelle standardisée du modèle (au format *Mocodo*) pour garantir une interprétation uniforme et exploitable ;
- évaluation de la cohérence du texte généré : analyser la qualité descriptive obtenue, en vérifiant la fidélité du texte par rapport au modèle original, la clarté des explications, et l'absence d'erreurs ou d'ambiguïtés pouvant nuire à la compréhension.

Les premières évaluations révèlent que les SLM produisent parfois correctement les structures de base (entités, relations, clés et contraintes) dans les formats UML et ER, mais qu'ils échouent fréquemment à respecter strictement ces formats ou à gérer les cas ambigus.

Pour pallier ces écarts, il est indispensable :

- d'optimiser les scripts d'automatisation, en renforçant la gestion des erreurs, la relance automatique en cas d'échec et la traçabilité des logs ;
- d'ajuster finement la « température » (équilibre créativité/rigueur) des modèles, afin de trouver le meilleur compromis entre créativité et précision, ce qui devrait conduire à une amélioration sensible de la fiabilité et de la cohérence des générations futures.

### ***Comparaison des modèles manuels et des modèles générés par LLM***

Pour comparer les modèles de données générés par les SLM à ceux créés manuellement, nous avons évalué trois points clés. D'une part, la conformité aux normes relationnelles : si les modèles manuels garantissent dans la plupart des cas la normalisation (1NF, 2NF, 3NF), la définition claire des clés et la cohérence des relations, les SLM n'atteignent pas toujours ce niveau de rigueur. Ils produisent parfois des omissions de clés ou des erreurs de cardinalités, même lorsque des prompts structurés améliorent leur précision. D'autre part, l'adéquation de la complexité au niveau des épreuves : les modèles manuels sont calibrés pour être simples au bac NSI et plus riches pour le CAPES et l'agrégation, tandis que les SLM, sans directive explicite sur le public cible, génèrent parfois des schémas trop élémentaires ou au contraire trop denses, nécessitant un réglage fin des consignes.

D'un point de vue méthodologique, la construction manuelle d'un modèle amène l'étudiant ou l'enseignant à se poser bien plus de questions sur le cas étudié que la proposition d'un SAIG, à échanger avec les autres (quand c'est possible), à réfléchir aux hypothèses et à leurs conséquences. Ces questionnements favorisent les échanges, la prise de

conscience des choix et les compromis, consolidant ainsi une vision pédagogique de l'apprentissage. Par exemple, l'analogie avec des cas existants est plus naturelle qu'avec un SAIg dont le processus, souvent personnel, est très orienté vers le but.

Ces constats montrent que, si les SLM ne remplacent pas l'expertise humaine pour produire des modèles totalement fiables, ils constituent un atout pour accélérer la génération de proto-modèles et gagner en évolutivité. L'intégration d'un outil automatisé, combinant prompts optimisés, réglage de la « température » et validation systématique de la qualité des résultats selon la grille d'analyse<sup>20</sup>, apparaît comme une voie prometteuse pour accompagner les enseignants de NSI dans la modélisation de bases de données.

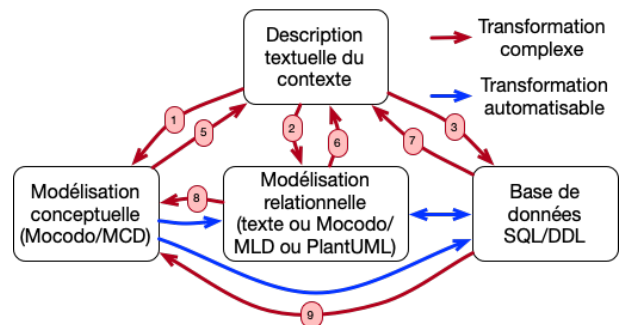
#### 4. – Construire un modèle avec les SAIg

Nous poursuivons dans cette section les expérimentations en portant l'accent sur les techniques de paramétrage et de prompt pour la construction des modèles de données et l'ingénierie à mettre en œuvre pour espérer un résultat de qualité suffisante. Même si la phase de modélisation n'est pas au programme du lycée, il est tout de même intéressant de l'étudier pour l'enseignant de NSI ou pour produire certains exercices de modélisation de bases de données en premier cycle universitaire.

##### 4.1. - Contexte

Nous avons travaillé sur l'impact du prompt et des spécificités des modèles de langages sur la production d'un modèle de données. Nos expérimentations présentées ici concernent les cadres d'usage visant à obtenir

un modèle (conceptuel ou relationnel) à partir d'un contexte textuel.



**Figure 10** : Passages entre modèles.

Pour la génération du modèle relationnel, comme l'illustre la figure 10, plusieurs stratégies sont possibles :

- le modèle en SQL/DDDL depuis le texte ;
- le modèle relationnel puis le modèle SQL/DDDL ;
- le modèle conceptuel, puis en SQL/DDDL ;
- le modèle conceptuel, puis le modèle relationnel (textuel), et enfin en SQL/DDDL.

Le modèle conceptuel peut être écrit en *Mocodo/MCD* et le modèle relationnel en *Mocodo/MLD* ou en *PlantUML*. Nous l'avons déjà évoqué, la littérature propose des algorithmes assez simples pour passer du modèle conceptuel au modèle relationnel et/ou au modèle SQL. Des outils comme *Mocodo*, *Windsign* ou *Looping-MCD* automatisent cette transformation. Finalement, la partie complexe, qui n'est pas automatisée et pour laquelle un SAIg peut apporter une aide (les flèches rouges sur la figure 10), est le passage du texte à un modèle (flèches 1 à 3) ou d'une modélisation relationnelle au modèle conceptuel du fait de la relative pauvreté descriptive du relationnel (flèches 8 et 9). Les opérations inverses vers le texte, appelée « reformulation » (flèches 5, 6 et 7), sont aussi des tâches complexes pour lesquelles un SAIg peut être utile. Dans la suite, nous nous intéresserons à trois transformations délicates : le MCD à par-

<sup>20</sup> Grille disponible dans Kodo (2025), p. 15.

tir du contexte (flèche 1), le MLD à partir du contexte (flèche 2) et le SQL/DDI à partir du contexte (flèche 3). Nous expérimenterons ces transformations sur l'exemple « fil-rouge ». Avant d'aborder la question du prompt, nous allons d'abord nous intéresser aux stratégies mises en œuvre pour gérer au mieux les SAIG.

## 4.2. - Choix des modèles

Nous avons eu à choisir les modèles de langage à tester. Pour cela, nous avons pris en compte les usages habituels des enseignants et le cadre proposé par l'institution.

Tout d'abord, nous avons sélectionné quelques LLM connus :

- *GPT-5* et *GPT-4o* (*Open AI*, USA), LLM les plus connus et, sans doute, les plus utilisés chez les enseignants ;
- *Claude 4 Sonnet* (*Anthropic*, USA), un des modèles les plus puissants à la date de la rédaction de cet article ;
- *Mistral Pro* (*Mistral AI*, FR), un modèle ouvert, français, avec un corpus plus francophone ;
- *LLama 4* (*Meta*, USA), autre moteur open source reconnu et faisant l'objet de multiples spécialisations.

Ces outils sont rapides (sauf si le mode raisonnement<sup>21</sup> est activé) et donnent des réponses développées avec un minimum d'erreurs (selon divers benchmarks). Ils sont dotés de capacités de raisonnement (*GPT5*, *Claude 4*, *Mistral Pro*) et, pour certains, peuvent être connectés à des sources externes (Web, courriel...). Par contre, ils demandent de grosses ressources de calcul et chargent le réseau. Ils ne nécessitent aucune installation en local (sauf à utiliser une application dédiée

comme *GPT4ALL*). Les données des prompts sont souvent utilisées pour l'apprentissage des futures versions. Il faut donc être très attentif aux informations présentées dans le prompt. Pour terminer, même si parfois il existe un usage « gratuit » (limité dans le temps, le volume ou la qualité), l'usage de ces outils peut être assez coûteux.

Nous avons aussi choisi de tester des SLM :

- *Gemma3n:4b* (*Google*, USA), petit modèle optimisé ;
- *Codellama:7b* (*Meta*, USA), spécialisé sur la programmation ;
- *Hermes-Llama-3.1-8b* (*Nous Research*, USA), aussi spécialisé sur le code ;
- *Mathstral-7b* (*Mistral AI / Mistral Small*, FR), spécialisé sur les maths ;
- *Mistral-NeMo* (*Mistral AI / Mistral Small*, FR).

Ces outils sont, le plus souvent, assez loin des LLM en termes de qualité. Le résultat diverge parfois vite pour les plus petits. Cependant, ils sont légers et donc rapides. Certains, les plus récents, ont même aussi des capacités de raisonnement. De plus, ils sont moins gourmands en ressources et peuvent être installés sur des PC portables (sans les fameuses cartes graphiques) ou des micro-PC (*Raspberry Pi 5* par exemple).

Nous appelons « divergence du résultat » le fait qu'un modèle construit une réponse sans cohérence, en changeant constamment de « langue » voire en proposant une suite de caractères qui ne sont plus interprétables. Les petits SLM ont une forte tendance à avoir ce comportement. Ce défaut peut être limité en jouant avec la température. L'idée est de la mettre à 0 (ou très proche de 0). Ainsi, parmi les suites choisies, les plus probables (donc *a priori* les plus cohérentes) seront sélectionnées. Quand c'est possible, on peut aussi le

---

<sup>21</sup> Beaucoup de xLM actuels intègrent une phase de raisonnement préalable à la réponse. Bien que très coûteuse en temps, cette phase améliore assez souvent la réponse.

faire avec les LLM. En effet, ici on ne cherche pas la « créativité » (la relative surprise dans la réponse générée), mais l'exactitude, la précision. L'autre avantage est le déterminisme du résultat. Ceci permet de faire plusieurs générations avec le même résultat. Ainsi, dans certains cas d'usage, cela permet d'ajuster le contexte pour le rendre plus précis par rapport au modèle attendu.

### *Construction des prompts*

Dans notre approche, nous allons agir dans trois axes pour « piloter » le SAIG : le prompt système, le prompt utilisateur et le RAG.

Dans toutes les situations, nous avons utilisé le même prompt système en lien fort avec les objectifs de notre étude :

```
Tu es un assistant en ingénierie des
systèmes d'information, spécialiste des
Modèles Conceptuels de Données (MCD) et
des bases de données SQL. Tu t'appuies
sur la méthode MERISE qui utilise la
représentation Entité-Association-
Propriété (EAP) pour le MCD. Pour les
bases de données, tu utilises SQL et
PostgreSQL. Tu écris en français. Tu
connais Mocodo pour représenter les
modèles conceptuels et relationnels et
PlantUML pour les modèles relationnels.
```

Nous avons fait le choix, un peu réducteur, de proposer un prompt utilisateur unique et non un dialogue avec le SAIG. L'idée est d'analyser le « premier jet » du système. Un dialogue pouvant s'instaurer après pour affiner le résultat. Cependant, ces prompts comportent des instructions en deux ou trois phases qui guident la production de la réponse. La première est une phase de mise en évidence des concepts présents dans le texte. Ensuite, le modèle est demandé. L'idée est de solliciter une analyse du texte, avant la production de résultat cherché, afin de permettre au système de produire les éléments de base, et donc d'améliorer la génération.

Nous avons visé trois cas d'usages (présentés plus haut) à savoir obtenir un modèle conceptuel, un modèle logique et un modèle relationnel SQL à partir d'un texte.

Pour le premier cas, le prompt utilisateur de base est le suivant :

```
1- Sur le texte suivant, explique les
entités présentes et leurs relations :
```md
<ici le texte qui décrit le contexte>
```

2- Propose un modèle conceptuel écrit en
Mocodo/MCD permettant de modéliser les
données de ce texte.
```

Cette approche est la plus complexe puisque la notion de modèle conceptuel n'est pas extrêmement répandue dans les espaces informationnels utilisés pour l'apprentissage des modèles fondamentaux.

Pour le second, le prompt utilisateur de base pour SQL est :

```
1- Sur le texte suivant, explique les
entités présentes et leurs relations :
```md
<ici le texte qui décrit le contexte>
```

2- Propose un modèle relationnel écrit en
SQL (avec PostgreSQL) permettant de
construire une base de données de ce
texte.
```

Viser un SGBD particulier (ici PostgreSQL) essaie de faire en sorte que le SQL proposé soit cohérent sur une syntaxe SQL et d'éviter un mélange des variantes des différents SGBD. *A priori*, nous sommes sur la génération la plus favorable, puisqu'il existe un grand nombre de ressources pédagogiques qui présentent un contexte et la base de données (dans le standard SQL) qui correspond.

Le prompt pour obtenir le modèle logique de données est le suivant :

```
1- Sur le texte suivant, explique les
entités présentes et leurs relations :
```md
```

---

**MODÉLISATION ASSISTÉE PAR L'IA  
POUR L'ENSEIGNEMENT DES  
BASES DE DONNÉES**


---

<ici le texte qui décrit le contexte>  
...

- 2- Propose un modèle relationnel permettant de construire une base de données de ce texte.
- 3- Propose le modèle relationnel en **PlantUML**/EID. Ne mets pas les types. Ne pas utiliser de fonction.

Le souci principal porte sur la forme du résultat. L'explication textuelle ne pose pas de problème au système. Il en est de même pour le SQL, langage fréquent et donc ayant fait l'objet d'un apprentissage important. Par contre, les langages pour *Mocodo* (*Mocodo/MCD* et *Mocodo/MLD*) et *PlantUML* sont plus confidentiels. De plus, *PlantUML* n'est pas bien adapté pour décrire les modèles relationnels. Par conséquent, afin d'améliorer la production pour ces langages, nous avons choisi d'ajouter un « contexte de langage » avant les instructions. Ce contexte propose d'abord une description rapide du langage puis trois exemples où l'on propose un texte et la sortie attendue.

Nous avons aussi essayé de mettre en place un ensemble de documents pour alimenter un système RAG quand c'est possible. Nous avons produit les documents de description des langages (même description que dans le contexte de langage), et quelques documents de description de la modélisation conceptuelle et relationnelle issus d'un cours e-miage<sup>22</sup>.

Quand certains modèles le permettent, nous avons parfois activé ou renforcé le raisonnement (avec *Claude 4* et *GPT-5* par exemple) pour voir l'impact sur le résultat.

### 4.3. - Analyse des expérimentations

À l'aide de ces prompts, nous allons maintenant analyser les résultats sur notre exemple fil rouge. Nous avons effectué trois

<sup>22</sup> [https://www.desmontils.net/emiage/A354\\_v1/c2/a354\\_2.html](https://www.desmontils.net/emiage/A354_v1/c2/a354_2.html)

séries de tests (un par cas d'usage) en faisant varier la présence ou non du RAG, la présence ou non du contexte de langage dans le prompt et, parfois, en activant le raisonnement. Ces tests sont accessibles en ligne à l'adresse : <https://gitlab.univ-nantes.fr/ls2n-didactique/modelisation-bd-didactique> (les PDF dans Génération/tests).

### *Génération du SQL*

Le premier cas que nous présentons est celui permettant d'obtenir le modèle SQL/DDL. Dans ce cas, il n'est pas nécessairement utile d'ajouter le contexte de langage. Le tableau de synthèse est proposé en annexe 1.

Globalement, nous constatons tout de même un réel plus à ajouter le contexte de langage. Par contre, le RAG n'est absolument pas efficace. Il faudrait plutôt des documents en lien avec le contexte.

Bien sûr, les LLM sont plus efficaces (particulièrement *Claude*) que les SLM. Beaucoup de modèles ont des difficultés avec la gestion des relations (cardinalités 1N et surtout 11 en MCD). Il convient donc d'être attentifs à la bonne gestion des unicités et des obligations, surtout avec les SLM, mais pas seulement. Plusieurs modèles posent une table « Mairie » (ou équivalent) qui n'est pas nécessaire dans le cadre de cette modélisation.

Les modes de raisonnement « forts » n'apportent pas de vraie différence dans notre cas.

### *Génération du modèle logique*

Dans ce cas d'utilisation, nous ne cherchons pas à produire du SQL. Nous demandons un modèle relationnel textuel, puis en version *PlantUML*. Le tableau de synthèse est proposé en annexe 2.

Remarque : cette expérimentation ayant été faite en dernier, nous n'avons pas fait d'essai sans le contexte, vue l'efficacité constatée par les autres expérimentations.

Pour ce cas, les SLM ont beaucoup de difficultés. En particulier, les rôles sont assez mal gérés. Une hypothèse est la demande de *PlantUML*. Autant pour SQL, ils ont appris, autant pour *PlantUML* c'est beaucoup moins le cas. Ils ont clairement moins de facilité à « apprendre » un langage que les LLM. Pour ces derniers, les majeurs sont bien présents et efficaces pour cette modélisation. Ils ont tout de même des difficultés avec les liens entre les tables en *PlantUML*. Nous avons constaté que, à nouveau, les modes de raisonnement renforcés n'apportent pas grand-chose.

Par exemple, le modèle proposé ici par *GPT-5* (avec contexte, sans RAG) est plutôt riche, mais le modèle *PlantUML* n'est pas tout

à fait complet (*cf.* figure 11 ; manquent les liens pour président et trésorier) :

```

« - personne(id_personne PK, nom, prenom, ...)
- association(id_association PK, nom,
objectifs, id_president FK -> personne,
id_tresorier FK -> personne)
- Contrainte d'intégrité métier
recommandée: (id_association,
id_president) et (id_association,
id_tresorier) doivent référencer une
adhésion existante (garantit que
président et trésorier sont des membres).
- activite(id_activite PK, libelle,
description, ...)
- adhesion(id_association FK -> association,
id_personne FK -> personne,
date_adhesion, PK (id_association,
id_personne))
- propose(id_association FK -> association,
id_activite FK -> activite,
id_responsable FK -> personne, PK
(id_association, id_activite))
- Contrainte d'intégrité métier:
(id_association, id_responsable) FK ->
adhesion(id_association, id_personne)
pour garantir que le responsable est
adhérent de l'association. »

```

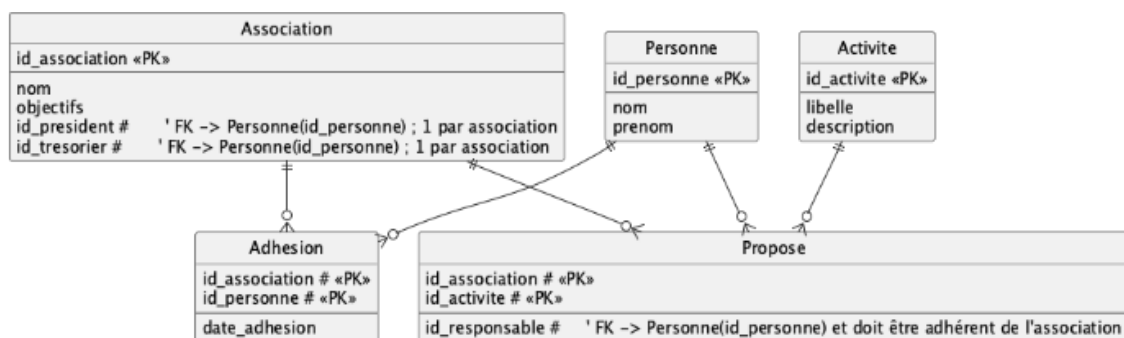


Figure 11 : Modèle relationnel en PlantUML pour GPT5.

### Génération du modèle conceptuel

Dans notre dernier cas, nous abordons la génération du modèle conceptuel. Cet objectif est *a priori* plus complexe, puisque la notion de modèle conceptuel de données est une notion liée à la méthodologie Merise, même si elle peut être décrite en diagramme de classes UML. Nous pouvons remarquer, cependant, que le sous-langage de description de modèle

conceptuel *Mocodo/MCD* (le sous-langage de description du modèle relationnel est apparu plus récemment) est assez bien géré. Le tableau de synthèse est proposé en annexe 3.

Comme pour le modèle relationnel, la gestion des cardinalités est compliquée. Nous constatons fréquemment que les cardinalités semblent « inversées ». Une explication possible est la présence beaucoup plus importante

de descriptions de type UML où, dans les diagrammes de classe, les cardinalités sont effectivement inversées. Le contexte de langage aide bien pour améliorer la syntaxe *Mocodo*. La notion d'entité faible est très rare dans les modèles générés. La cohérence entre cardinalités pose des problèmes, même aux meilleurs outils. Dans notre exemple, il y a quasiment systématiquement 0N de l'association vers l'adhérent et 11 pour le président et le trésorier. Il peut donc ne pas y avoir d'adhérents, mais il y a toujours un président et un trésorier.

Comme pour les autres cas, la plus-value des RAG n'est pas évidente sur le résultat. Les documents proposés étaient des documents sur la méthodologie de modélisation conceptuelle, difficiles à exploiter. Donc, le travail direct du prompt est bien plus efficace. Cependant, surtout avec les SLM, la longueur du prompt est importante, ce qui pose des soucis lorsque la fenêtre contextuelle est petite.

Les modes de raisonnement boostés n'apportent pas non plus de réelles améliorations sur la qualité de la production.

La figure 12 montre un exemple de bon modèle généré par un SLM (reste un problème de cohérence de cardinalité).

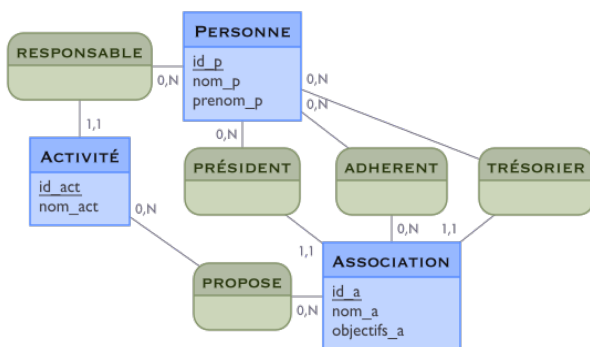


Figure 12 : MCD proposé par  
Hermes 3 Llama 3.1.

Dans la section suivante, nous synthétisons une partie des enseignements des expérimentations pour proposer une première version de méthodologie de construction de modèles conceptuels et logiques de bases de données avec assistance des SAIg.

## 5. – Méthodologie pratique

Lors de nos expérimentations, nous avons pu constater que modéliser une situation à partir d'un texte est aussi (voire plus) « difficile » pour les SAIg qu'« à la main ». Pour un même contexte relativement petit (en nombre de tables ou d'entités et d'associations), ils produisent des solutions assez variées, intéressantes, mais comportant souvent des lacunes, notamment pour la maîtrise des contraintes de présence d'une information, en particulier : « au moins présente une fois », « présente une seule fois »... Dans le cadre du modèle conceptuel, ces problèmes sont induits par l'architecture (entités multiples) ou, plus fréquemment, par des cardinalités totalement ou partiellement fausses. Ces problèmes sont naturellement plus fréquents pour des SLM que pour des LLM.

Cependant, même pour les SLM, l'architecture du modèle est parfois proche d'une solution possible. L'architecture du modèle est la structure des entités et les liens qu'elles ont à travers des associations. Les noms des propriétés ou des associations ne sont pas essentiels de ce point de vue.

Nous allons donc nous poser la question suivante : comment l'IA générative peut-elle améliorer le travail complexe de modélisation pour un enseignant en informatique (NSI ou 1<sup>er</sup> cycle) ou un étudiant ? En particulier, la phase la plus délicate (pour nos étudiants comme pour nous, enseignants) est l'amorce, c'est-à-dire la **construction d'un proto-modèle**.

L'autre difficulté majeure est de **vérifier le modèle**. Autrement dit, comment est-on certain(e) que le modèle répond bien aux besoins du contexte ?

### 5.1. - Aide à la construction du modèle

Comme nous l'avons dit, proposer un modèle (conceptuel ou relationnel) à partir d'une spécification de besoins (souvent sous forme de texte, de documents scannés...) est une tâche complexe. Les raisons sont variées, mais c'est essentiellement dû au fait que le langage naturel est fondamentalement imprécis et qu'il est compliqué de transposer ses besoins par écrit et même d'être exhaustif. Il est complexe de passer d'une description imprécise, floue, à un modèle formel. Dans la suite, nous supposons l'expression des besoins uniquement sous forme de texte. Par texte, il est possible d'imaginer aussi des informations issues d'un tableur. Ces informations sont bien exploitées par les SAIG.

Remarque : certains systèmes, dits multimodaux, sont capables d'analyser des images. Ils pourraient aussi être sollicités pour exploiter des documents non textuels. Cependant, ce n'est sans doute pas fondamental dans le cadre de l'enseignement des bases de données.

Pour une bonne modélisation, l'étape clé est la construction du proto-modèle. Celui-ci n'est pas complet, mais comporte les entités et les associations principales, ainsi qu'une partie

des propriétés (essentiellement les identifiants). Pour les cardinalités, elles peuvent être absentes (ou seulement les cardinalités très contraignantes). Décrit comme cela, beaucoup de LLM (et même les SLM) sont capables de produire ce qui peut être considéré comme un proto-modèle. Pour cette approche, un SLM moyen (autour de 12 milliards de paramètres) peut être un choix intéressant pour le respect des cadres éthiques et déontologiques (Germain et al., 2022, MENESR (s.d.), CNIL (s.d.)).

Pour cette construction, il faut bien évidemment le texte de contexte (les besoins). Ensuite, il est possible de prendre l'invite (prompt système) présenté dans la section précédente. Enfin, la construction du prompt utilisateur peut suivre le patron présenté aussi dans la section précédente en fonction du modèle cible choisi. Si, dans le cadre des besoins, des documents textuels (traitement de texte de type *Word*, Markdown, PDF, tableur de type *Excel*) sont disponibles, il est possible d'envisager d'utiliser un SAIG avec RAG.

Le processus de base est donc le suivant :

1. sélectionner le modèle de langage adapté (en accord avec les cadre éthiques et déontologiques) ;
2. construire les prompts (système et utilisateur) et, le cas échéant, regrouper les documents pour alimenter en RAG ;
3. quand c'est possible, régler le modèle (température, gestion du RAG...) ;
4. générer le proto-modèle à l'aide du SAIG ; en fonction du choix de température, générer plusieurs modèles pour choisir celui qui semble le plus adapté ;
5. modifier le proto-modèle proposé pour l'ajuster aux besoins et en faire un modèle satisfaisant ;
6. valider le modèle.

Nous proposons le projet *MoDIA*<sup>23</sup> pour automatiser partiellement ce processus. Il est important de partir sur une bonne base. Jouer sur des variantes issues du modèle (à travers une température à 0.25 au lieu de 0 par exemple) peut être aussi intéressant. Pour le reste, il convient d'être critique vis-à-vis de la solution proposée. Nous l'avons vu, les modèles de langage ont très souvent des difficultés avec l'obligation de présence d'une information, son unicité ou le statut d'identifiant. De même, les propriétés sont parfois insuffisantes, mais aussi parfois des propriétés complémentaires sont proposées qui font sens par rapport au sens commun du contexte. Là, c'est très intéressant.

Dans nos prompts, nous demandons, avant la production du modèle, une analyse du texte en termes d'entités et de relations entre les entités. Souvent, cette analyse est de qualité. Nous sommes alors vraiment dans le domaine de prédilection des SAIG : produire du texte à partir de texte. Aussi, cette phase peut être suffisante pour construire un bon modèle, sans pour autant perdre du temps à faire générer un modèle médiocre. Par exemple, voici l'analyse produite par *Llama 4 Scout-T* (avec contexte) où l'analyse est plutôt bonne :

```
## 1. Analyse des entités et de leurs relations
D'après le texte fourni, les entités principales sont :
- **Association** : Une association de la commune, caractérisée par des informations telles que le nom et les objectifs.
- **Activité** : Une activité proposée par une ou plusieurs associations.
- **Personne** : Une personne de la commune qui peut être adhérente à une ou plusieurs associations, et qui peut également occuper des rôles spécifiques comme président ou trésorier dans une association.
Les relations entre ces entités sont :
- Une association peut proposer plusieurs activités, et une activité peut être proposée par plusieurs associations.
```

- Une personne peut être adhérente à plusieurs associations, et une association est composée de plusieurs personnes adhérentes.
- Chaque activité a un seul responsable qui est membre de l'association.
- Chaque association a un seul président et un seul trésorier, mais une personne peut être présidente ou trésorière de plusieurs associations.

Remarque : une stratégie possible serait d'inverser les rôles. L'enseignant produit le proto-modèle et le SAIG le corrige et l'enrichit. L'enseignant termine en validant et en corrigeant. Nous n'avons pas évalué cette approche.

## 5.2. - Vérifier un modèle

Dans l'approche précédente, une phase délicate est la vérification du modèle. C'est une interrogation fréquente chez nos étudiants et chez les enseignants que nous rencontrons. Le processus de validation se fait essentiellement à partir de l'expression de besoins. À partir de ceux-ci, un ensemble de scénarios est produit et testés sur le modèle. Si un scénario ne peut pas être joué, le modèle n'est pas satisfaisant. Un travail sur la normalisation du modèle permet souvent de mettre en évidence des erreurs de modélisation. Ces techniques sont toujours applicables sur un modèle généré par un SAIG.

Les SAIG peuvent être des outils utiles pour accompagner cette phase. Pour cela, deux approches sont possibles : par comparaison de modèles ou par comparaison de besoins.

Dans le premier cas, l'enseignant produit un modèle M1 par lui-même. Il veut le vérifier. À partir de l'expression des besoins, un autre modèle M2 est généré par un SAIG. L'enseignant compare les deux modèles produits.

<sup>23</sup> <https://gitlab.univ-nantes.fr/ls2n-didactique/modia>

Remarque : on pourrait aussi imaginer un SAIG chargé de comparer les modèles. Nous n'avons pas encore évalué cette approche.

Dans le second cas, l'idée est liée à la vérification de la conformité aux besoins. À partir du modèle à vérifier, un SAIG paraphrase le modèle en langage naturel. Le texte produit est alors comparé au texte d'expression des besoins. L'éventuel décalage sémantique entre les deux textes est un indicateur de mauvaise conception. Cette approche est beaucoup plus intéressante que la précédente puisque les SAIG sont plus à l'aise dans la production en langage naturel.

### 5.3. - Vérifier l'expression des besoins

Pour terminer, nous pouvons aussi reprendre les processus précédents dans un autre objectif. Dans le cadre de la production d'un exercice, une difficulté fréquente pour un enseignant en bases de données est la production d'un texte d'expression de besoin :

- pour illustrer une base de données (en lycée et en 1<sup>er</sup> cycle universitaire) ;
- pour demander un modèle associé à ce texte (en 1<sup>er</sup> cycle universitaire).

Le texte d'expression des besoins est complexe à produire, car il est susceptible de multiples interprétations et, parfois, a comme fondement des éléments culturels implicites. Aussi, un SAIG peut assister à la conception de ces textes. Là encore, deux approches (déjà présentées) peuvent être évoquées : par comparaison de modèles ou par comparaison de textes.

Dans le premier cas, l'enseignant fait produire le modèle à partir des besoins par un SAIG. Il compare le modèle produit au modèle qu'il a imaginé. S'ils sont différents, le texte des besoins nécessite peut-être de faire l'objet d'une reformulation. Dans ce cas, il faut avoir

une certaine confiance en sa capacité à modéliser ou concevoir l'exercice par rapport à un modèle cible et non un besoin. C'est le cas lorsque l'on veut par exemple travailler des types de requêtes SQL spécifiques sur une structure de base de données particulière. L'expression des besoins sert alors à « habiller » l'exercice.

Dans le second cas, l'enseignant produit un modèle à partir de ce qu'il imagine de l'expression des besoins. Il utilise ensuite SAIG pour reformuler le modèle en langage naturel. Si les textes sont sémantiquement différents, une cause peut être une mauvaise formulation du besoin (et non du modèle comme dans la section précédente).

### Conclusion

La construction de modèles de données pour les bases de données relationnelles est une thématique abondamment étudiée dans les années 1970-1980. Il en résulte des méthodes de conception largement validées, que nous enseignons depuis des années dans les formations de génie logiciel en informatique et que nous avons présentées dans la section 1. Néanmoins, ces méthodes ne sont pas toujours maîtrisées et la qualité du résultat est liée à l'expérience de l'analyste de systèmes d'information dans l'application de ces méthodes. Si le schéma de données résultat est de mauvaise qualité alors la performance du système d'information risque d'être très dégradée et surtout sa maintenance va devenir très onéreuse.

Il convient donc, et nous l'avons montré dans cet article, de maîtriser le processus et la qualité des résultats. Ce point devient critique avec l'usage de l'intelligence artificielle et notamment les systèmes artificiels intelligents génératifs (SAIG), qui n'appliquent pas les mêmes méthodes de conception que celles

mentionnées ci-dessus. Nous avons montré à travers les expérimentations en sections 3 et 4, que les résultats sont de qualité aléatoire et nous pouvons donc affirmer que les approches SAIG ne sont pas matures sur le sujet. Un autre fait vient étayer notre constat : la thématique de recherche « modélisation conceptuelle » qui déclinait dans les conférences sur les systèmes d'information, à cause de la maturité du sujet, devient bien plus active en intégrant le SAIG (Keserovic et al., 2025). Il convient donc d'être vigilant à les utiliser et encore plus à les encadrer pour l'enseignement. Le constat est sans appel : les étudiants qui prennent « pour argent comptant » le résultat d'une SAIG risquent d'être bien déçus par leur note. Car si le résultat paraît plausible, ou probable, car c'est une affaire de probabilité pour les SAIG, il n'est pas forcément correct, cohérent, complet ou valide. Ce sont une partie des qualités attendues pour des modèles.

En réponse, il nous paraît crucial :

- d'une part que les étudiants et les enseignants maîtrisent les méthodologies de conception, par apprentissage classique (cf. section 1) pour être capables d'évaluer puis d'améliorer la qualité des modèles obtenus avec les SAIG, dont l'intérêt ici est d'accélérer (assister) le processus de construction de modèles de données ;
- d'autre part qu'ils maîtrisent le processus d'interactions avec les SAIG pour limiter les errements (essais successifs divergents) à construire un modèle de données acceptable. Nous avons proposé un premier processus en ce sens dans la section 5.

Le travail de synthèse et d'expérimentation exposé dans cet article établit les fondations d'une nouvelle méthodologie de construction de modèles conceptuels et logiques de données. Les travaux doivent se poursuivre selon plusieurs perspectives.

- Dans un premier temps, la méthodologie doit être affinée pour définir un cadre d'application cohérent et complet, préliminaire à la construction pédagogique de son enseignement.
- La méthodologie doit être expérimentée et validée par un panel large et hétérogène de praticiens pour valider l'approche et les modèles produits, mais aussi pour détecter d'éventuelles difficultés dans sa pratique.
- Les technologies SAIG évoluant très rapidement, il est nécessaire de périodiquement « challenger » les outils pour déterminer leur performance à résoudre un « benchmark » de cas tel que nous l'avons présenté dans la section 3. Développer une communauté pédagogique sur cet aspect serait une réponse efficace.

En conclusion, les SAIG sont une assistance formidable dans la conception de modèles de données, à condition de se donner les moyens intellectuels de les utiliser.

**Emmanuel DESMONTILS**

LS2N & CREN, Nantes Université  
IREM Pays de la Loire, Groupe EIEM

**Pascal ANDRÉ**

LS2N, Nantes Université

**Loïc KODO**

Stagiaire au LS2N  
3<sup>e</sup> année de Licence Informatique 2024-2025  
Nantes Université

## Références bibliographiques

Andler, D. (2023). *Intelligence artificielle, intelligence humaine : la double énigme*. Gallimard.

- André, P. & Vailly, A. (2001). *Volume 1 : Conception des Systèmes d'Information, Panorama des méthodes et des techniques*. Éditions Ellipses.
- André, P. & Vailly, A. (2003). *Volume 3 : Exercices corrigés de conception logicielle, Modélisation de Systèmes d'Information par la pratique*. Éditions Ellipses.
- Boubaker, L., Mellal, L. & Djebabra, M. (2010). Modèle DIC (Données-Informations-Connaissances) Outil support pour le développement des mémoires projets. *La Revue des Sciences de Gestion*, 243–244, 153–159.
- Bruillard, E. (2025). Intelligence artificielle générative et éducation scolaire : quelques réflexions. *Sciences et Technologies de l'Information et de la Communication pour l'Éducation et la Formation*, 32(1), 1–25. DOI : 10.23709/STICEF.32.1.6
- Codd, E. F. (1970). A Relational Model of Data for Large Shared Data Banks. *Communications of the ACM*, 13(6), 377–387.
- Delobel, C. & Adiba, M. (1982). *Bases de données et systèmes relationnels*. Dunod Informatique.
- Gardarin, G. (2003). *Bases de données, 5<sup>e</sup> édition*. Éditions Eyrolles, collection Informatique.
- Hainaut, J.-L. (2022). *Bases de données - Concepts, utilisation et développement, 5<sup>e</sup> édition*. Dunod.
- Higuera, C. de la & Iyer, J. (2024). *IA pour les enseignants : Un manuel ouvert*. <https://pressbooks.pub/iapourlesenseignants/>
- Keserovic, D., Brdjanin, D., Banjac, G. & Banjac, D. (2025). Speed: An Online Multilingual Speech-based Database Design Tool. *Actes de la 33rd International Conference on Information Systems Development (ISD 2025)*, Belgrade, Serbie.
- Kodo, L (2025). *Tour d'horizon et analyse des modèles relationnels en enseignement des bases de données*. [Rapport de stage de Licence Informatique 3<sup>e</sup> année, Nantes Université]. Disponible sur demande.
- Morley, C., Bia-Figueiredo, M. & Gillette, Y. (2011). *Processus métiers et S.I., 3<sup>e</sup> édition*. Dunod.
- Nanci, D. & Espinasse, B. (2001). *Ingénierie des systèmes d'information : Merise - Deuxième génération, 4<sup>e</sup> édition*. Éditions Vuibert.
- UNESCO (2023). *Guidance for generative IA in education and research*. <https://www.unesco.org/fr/articles/orientations-pour-lintelligence-artificielle-generative-dans-leducation-et-la-recherche>

### Références institutionnelles

- CNIL (Commission Nationale Informatique et Libertés) (s.d.). *Enseignant : Comment utiliser un système d'IA dans le cadre de vos missions ?* <https://www.cnil.fr/fr/enseignant-usage-systeme-ia> (consulté le 5 septembre 2025).
- Germain, E., Kirchner, C. & Tessier, C. (dir.). (2022). *Pour une éthique du numérique*. Comité National d'Éthique du Numérique. PUF.

---

**MODÉLISATION ASSISTÉE PAR L'IA  
POUR L'ENSEIGNEMENT DES  
BASES DE DONNÉES**


---

MENJS (2019a). Programme d'enseignement de spécialité de numérique et sciences informatiques de la classe de première de la voie générale. *Bulletin officiel spécial n° 1 du 22 janvier 2019*.

<https://www.educationgouv.fr/bo/19/Special1/MENE1901633A.htm>

[https://cache.media.education.gouv.fr/file/SP1-MEN-22-1-2019/26/8/spe633\\_annexe\\_1063268.pdf](https://cache.media.education.gouv.fr/file/SP1-MEN-22-1-2019/26/8/spe633_annexe_1063268.pdf)

MENJS (2019b). Programme de l'enseignement de spécialité de numérique et sciences informatiques de la classe terminale de la voie générale. *Bulletin officiel spécial n° 8 du 25 juillet 2019*.

<https://www.education.gouv.fr/bo/19/Special8/MENE1921247A.htm>

[https://cache.media.education.gouv.fr/file/SPE8\\_MENJ\\_25\\_7\\_2019/93/3/spe247\\_annexe\\_1158933.pdf](https://cache.media.education.gouv.fr/file/SPE8_MENJ_25_7_2019/93/3/spe247_annexe_1158933.pdf)

MENESR (s.d.). Cadre d'usage de l'IA en éducation. (s. d.).

<https://www.education.gouv.fr/cadre-d-usage-de-l-ia-en-education-450647>  
(consulté 17 juin 2025).

Nantes Université (2025). *Nantes Université s'engage pour une utilisation responsable et inclusive des IA génératives*.

<https://www.univ-nantes.fr/universite/vision-strategie-et-grands-projets/nantes-universite-sengage-pour-une-utilisation-responsable-et-inclusive-des-iagen>

## Annexes

### Annexe 1 – Tableau txt2sql

| Type | Modèle                    | Support | Temp. | RAG | Contexte | Think  | Note | Remarque                                                                                                                                                   |
|------|---------------------------|---------|-------|-----|----------|--------|------|------------------------------------------------------------------------------------------------------------------------------------------------------------|
| LLM  | Anthropic Claude 4 Sonnet | POE     | Std   | ✗   | ✗        | faible | 5    | Bien. Sauf « association activité » à la place de « propose »                                                                                              |
| LLM  | Anthropic Claude 4 Sonnet | POE     | Std   | ✗   | ✓        | faible | 5    |                                                                                                                                                            |
| LLM  | Anthropic Claude 4 Sonnet | POE     | Std   | ✓   | ✗        | faible | 3    | Propose des triggers, mais qui ne sont pas corrects. Il manque aussi des jointures.                                                                        |
| LLM  | Anthropic Claude 4 Sonnet | POE     | Std   | ✓   | ✓        | faible | 4    | Bon modèle, avec la présence de contraintes. Une petite erreur sur une contrainte.                                                                         |
| LLM  | Mistral AI Mistral Pro    | Le Chat | Std   | ✗   | ✗        | faible | 3    | Le modèle est basique et peu contrôlé. Le nom des relations est composé du nom des tables. Utilise le concept de rôle pour « président » et « trésorier ». |
| LLM  | Mistral AI Mistral Pro    | Le Chat | Std   | ✗   | ✓        | faible | 3    | Table de rôles intéressante, qui n'impose pas un président ou trésorier. Mauvaise gestion des responsables (pas d'unicité).                                |
| LLM  | Mistral AI Mistral Pro    | Le Chat | Std   | ✓   | ✗        | faible | 2    | Gestion des rôles encore différente. Maladie des « id ». « Unique » sur clé primaire !                                                                     |
| LLM  | Mistral AI Mistral Pro    | Le Chat | Std   | ✓   | ✓        | faible | 2    | Deux tables pour président et trésorier. Aucune gestion des contraintes d'unicité.                                                                         |

**MODÉLISATION ASSISTÉE PAR L'IA  
POUR L'ENSEIGNEMENT DES  
BASES DE DONNÉES**

| Type | Modèle                  | Support   | Temp. | RAG | Contexte | Think  | Note | Remarque                                                                                                                                                                                  |
|------|-------------------------|-----------|-------|-----|----------|--------|------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| LLM  | Open AI GPT 4o          | POE       | Std   | ✗   | ✗        |        | 3    | Modèle avec une table de rôle comme pour Mistral, avec le même défaut. Gère l'unicité des rôles.                                                                                          |
| LLM  | Open AI GPT 4o          | POE       | Std   | ✗   | ✓        |        | 3    | Pas de changement par rapport à la version précédente.                                                                                                                                    |
| LLM  | Open AI GPT 4o          | GPT4A LL  | 0.05  | ✓   | ✗        |        | 4    | Bon modèle. Bonne gestion des rôles.                                                                                                                                                      |
| LLM  | Open AI GPT 4o          | GPT4A LL  | 0.05  | ✓   | ✓        |        | 2    | Gestion des rôles mauvaise qui autorise plusieurs fois le même rôle dans l'association. Même problème pour le responsable.                                                                |
| LLM  | Open AI GPT 5           | POE       | Std   | ✗   | ✗        | faible | 4    | Bon modèle, mais présente des défauts. Table « commune » inutile. Bonne gestion des unicités. Triggers de validation.                                                                     |
| LLM  | Open AI GPT 5           | POE       | Std   | ✗   | ✓        | faible | 5    | Bonne gestion des rôles. Propose des solutions alternatives en commentaire.                                                                                                               |
| LLM  | Open AI GPT 5           | POE       | Std   | ✗   | ✓        | Fort   | 5    | Traitement long. Gestion du responsable curieuse avec une clé étrangère vers une autre table de relation.                                                                                 |
| LLM  | Meta Llama 4 Scout-T    | POE       | Std   | ✗   | ✗        | faible | 3    | Approche atypique des rôles (président, trésorier et responsable). Pas cohérent pour responsable. Ne gère pas l'unicité.                                                                  |
| LLM  | Meta Llama 4 Scout-T    | POE       | Std   | ✗   | ✓        | faible | 2    | Une table par rôle. Ne gère pas l'obligation d'avoir les rôles. Dans les tables Président et Trésorier, « unique » faux (empêche une personne d'être président de plusieurs associations) |
| SLM  | Google Gemma3n e4b      | Ollama    | 0     | ✗   | ✗        |        | 2    | Table « Mairie » inutile. Tables de rôle. Unicité avec « Unique ». Maladie des « id ». Pas de gestion du responsable.                                                                     |
| SLM  | Google Gemma3n e4b      | Ollama    | 0     | ✗   | ✓        |        | 2    | Quelques problèmes terminologiques. Ne gère pas les unicités ni les obligations de rôle.                                                                                                  |
| SLM  | Mistral AI Mathstral 7B | LM Studio | 0     | ✗   | ✗        |        | 4    | Bonne structuration. Gestion des unicités bonnes, manque l'obligation du président et du trésorier.                                                                                       |
| SLM  | Mistral AI Mathstral 7b | LM Studio | 0     | ✗   | ✓        |        | 3    | Problème avec les obligations.                                                                                                                                                            |
| SLM  | Mistral AI Mathstral 7b | LM Studio | 0     | ✓   | ✗        |        | 4    | Bonne structure. Gère bien président et trésorier. Présence d'une table « Mairie ».                                                                                                       |
| SLM  | Mistral AI Mathstral 7b | LM Studio | 0     | ✓   | ✓        |        | 3    | Problèmes d'obligations. Problème d'unicité du responsable.                                                                                                                               |

## Annexe 2 – Tableau txt2MLD

| Type | Modèle                    | Support | Temp. | RAG | Contexte | Think  | Note | Remarque                                                                                                                  |
|------|---------------------------|---------|-------|-----|----------|--------|------|---------------------------------------------------------------------------------------------------------------------------|
| LLM  | Anthropic Claude 4 Sonnet | POE     | Std   | ✗   | ✓        | Faible | 5    | Structure ok et PlantUML bien fait                                                                                        |
| LLM  | Anthropic Claude 4 Sonnet | POE     | Std   | ✗   | ✓        | Fort   | 4    | Clés primaires pas identifiées. PlantUML avec un Problème sur la contrainte d'adhésion pour le président et le trésorier. |

**MODÉLISATION ASSISTÉE PAR L'IA  
POUR L'ENSEIGNEMENT DES  
BASES DE DONNÉES**

| Type | Modèle                       | Support | Temp. | RAG | Contexte | Think  | Note | Remarque                                                                                                                                                                             |
|------|------------------------------|---------|-------|-----|----------|--------|------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| LLM  | Open AI GPT 5                | POE     | Std   | ✗   | ✓        | Faible | 4    | Bon modèle relationnel. Gestion du responsable mal fait (obligation absente, mais unicité). PlantUML bien écrit mais avec de liens mal faits (pas cohérents avec le modèle textuel). |
| LLM  | Open AI GPT 5                | POE     | Std   | ✗   | ✓        | Fort   | 5    | Bon modèle et bon schéma. Mais « réflexion » très longue...                                                                                                                          |
| LLM  | Mistral AI Mistral Pro       | Le Chat | Std   | ✗   | ✓        | Faible | 3    | Ne gère pas l'unicité ni l'obligation. Modèle textuel sans mettre les clés primaires.                                                                                                |
| LLM  | Meta Llama 4 Scout-T         | POE     | Std   | ✗   | ✓        | Faible | 2    | Modèle PlantUML avec beaucoup de Problème. Pas d'indication des clés primaires. Ne gère pas les contraintes.                                                                         |
| SLM  | Google Gemma3n e4b           | Ollama  | 0     | ✗   | ✓        |        | 2    | PlantUML incorrect. Pas de clés. Rôle à l'adhésion. Ne gère pas les contraintes.                                                                                                     |
| SLM  | Meta Codellama 7b            | Ollama  | 0     | ✗   | ✓        |        | 2    | Modèle de base. Mélange entre président/trésorier et responsable. PlantUML pas complet, mais structure correcte.                                                                     |
| SLM  | Mistral AI Mistral NeMo      | Ollama  | 0     | ✗   | ✓        |        | 3    | Modèle textuel avec des caractères parasites. Maladie des « id ». Aucune gestion des contraintes. PlantUML pas respecté.                                                             |
| SLM  | Mistral AI Mistral Mathstral | Ollama  | 0     | ✗   | ✓        |        | 1    | Modèle relationnel en SQL. Pas de gestion des contraintes (min et max). Version PlantUML avec des manques.                                                                           |

### Annexe 3 – Tableau txt2MCD

| Type | Modèle                    | Support  | Temp. | RAG | Conte xte | Think  | Moco-do | Nor-malisé | Note | Remarque                                                                                                                                                    |
|------|---------------------------|----------|-------|-----|-----------|--------|---------|------------|------|-------------------------------------------------------------------------------------------------------------------------------------------------------------|
| LLM  | Anthropic Claude 4 Sonnet | POE      | Std   | ✗   | ✗         | Faible | ✓       | ✓          | 3    | Beaucoup de cardinalités pas correctes.                                                                                                                     |
| LLM  | Anthropic Claude 4 Sonnet | POE      | Std   | ✓   | ✗         | Faible | ✓       | ✗          | 3    | Toujours de grosses difficultés avec les cardinalités.                                                                                                      |
| LLM  | Anthropic Claude 4 Sonnet | POE      | Std   | ✗   | ✓         | Faible | ✓       | ✗          | 4    | Encore quelques soucis de cardinalités (cohérence), mais bien mieux.                                                                                        |
| LLM  | Anthropic Claude 4 Sonnet | POE      | Std   | ✓   | ✓         | Faible | ✓       | ✓          | 4    | Des progrès sur les cardinalités, mais toujours des Problème de cohérence. Propose des contraintes (Trésorier est adhérent...), mais pas toujours correcte. |
| LLM  | Open AI GPT 4o            | POE      | Std   | ✗   | ✗         |        | ✗       |            | 2    | Mauvaise gestion des cardinalités.                                                                                                                          |
| LLM  | Open AI GPT 4o            | POE      | Std   | ✗   | ✓         |        | ✓       | ✗          | 3    | Gros bénéfice du contexte de langage. Syntaxiquement correcte. Cardinalités mauvaises. Presque « inversées ».                                               |
| LLM  | Open AI GPT 4o            | GPT4A LL | 0.05  | ✓   | ✗         |        | ✗       |            | 2    | Légère faute de syntaxe. Manque beaucoup de propriétés. Présence entités et association ok. Gros problèmes sur les cardinalités.                            |

**MODÉLISATION ASSISTÉE PAR L'IA  
POUR L'ENSEIGNEMENT DES  
BASES DE DONNÉES**

| Type | Modèle                                | Support      | Temp. | RAG | Conte<br>xte | Think  | Moco-<br>do | Nor-<br>malisé | Note | Remarque                                                                                                                   |
|------|---------------------------------------|--------------|-------|-----|--------------|--------|-------------|----------------|------|----------------------------------------------------------------------------------------------------------------------------|
| LLM  | Open AI<br>GPT 4o                     | GPT4A<br>LL  | 0.05  | ✓   | ✓            |        | ✓           | ✗              | 3    | Cardinalités plutôt satisfaisantes, sauf pour « responsable » (association fantôme) et « propose ». Manque des propriétés. |
| LLM  | Open AI<br>GPT 5                      | POE          | Std   | ✗   | ✗            | Faible | ✗           |                | 2    | Cardinalités correctes dans l'ensemble mais modèle mal construit.                                                          |
| LLM  | Open AI<br>GPT 5                      | POE          | Std   | ✗   | ✓            | Faible | ✓           | ✗              | 4    | Bonne construction, avec des contraintes. Une cardinalité fautive. Modèle non normalisé.                                   |
| LLM  | Open AI<br>GPT 5                      | POE          | Std   | ✗   | ✓            | Fort   | ✓           | ✗              | 4    | Bonne construction avec un soucis sur deux cardinalités. Une peut-être inversée.                                           |
| LLM  | Mistral AI<br>Mistral pro             | Le Chat      | Std   | ✗   | ✗            | Faible | ✗           |                | 2    | Manque des propriétés. Cardinalités mal faites.                                                                            |
| LLM  | Mistral AI<br>Mistral pro             | Le Chat      | Std   | ✗   | ✓            | Faible | ✓           | ✓              | 3    | Présence d'une contrainte non valide. Problème de cohérence de cardinalités. Activité, sans association !                  |
| LLM  | Mistral AI<br>Mistral pro             | Le Chat      | Std   | ✓   | ✗            | Faible | ✗           |                | 2    | Peu de propriétés, cardinalités inexistantes.                                                                              |
| LLM  | Mistral AI<br>Mistral pro             | Le Chat      | Std   | ✓   | ✓            | Faible | ✓           | ✓              | 4    | Toujours Problème de cohérence. Activité isolée. Sinon, plutôt bien.                                                       |
| LLM  | Meta<br>Llama 4 Scout-T               | POE          | Std   | ✗   | ✗            | Faible | ✗           |                | 2    | Confond modèle conceptuel et modèle relationnel                                                                            |
| LLM  | Meta Llama 4<br>Scout-T               | POE          | Std   | ✗   | ✓            | Faible | ✓           | ✓              | 3    | Cardinalités mauvaises : presque « inversée ». Activité isolée.                                                            |
| SLM  | NousResearch<br>Hermes 3<br>Llama 3.1 | LM<br>Studio | 0.05  | ✗   | ✗            |        | ✗           |                | 1    | Du Mermaid en sortie !                                                                                                     |
| SLM  | NousResearch<br>Hermes 3 Llama<br>3.1 | LM<br>Studio | 0.05  | ✗   | ✓            |        | ✓           | ✓              | 4    | Bon modèle. Des problèmes de cohérence de cardinalité (comme les grands).                                                  |
| SLM  | NousResearch<br>Hermes 3<br>Llama 3.1 | LM<br>Studio | 0.05  | ✓   | ✗            |        | ✗           |                | 1    | Malgré une analyse pas si mal, le modèle est totalement à côté.                                                            |
| SLM  | NousResearch<br>Hermes 3<br>Llama 3.1 | LM<br>Studio | 0.05  | ✓   | ✓            |        | ✓           | ✓              | 2    | Manque le président et le trésorier. Activité avec plusieurs responsables.                                                 |
| SLM  | Google<br>Gemini3n e4b                | Ollama       | 0     | ✗   | ✗            |        | ✗           |                | 2    | Génère le modèle relationnel en PlantUML !                                                                                 |
| SLM  | Google<br>Gemini3n e4b                | Ollama       | 0     | ✗   | ✓            |        | ✓           | ✓              | 2    | Présence inutile de la « Mairie ». La cardinalité ne sont pas correctes.                                                   |
| SLM  | Meta<br>Codellama 7b                  | Ollama       | 0     | ✗   | ✗            |        | ✗           |                | 1    | Complètement à côté. Juste les 3 entités Association, Activité et Personne.                                                |

**MODÉLISATION ASSISTÉE PAR L'IA  
POUR L'ENSEIGNEMENT DES  
BASES DE DONNÉES**

| Type | Modèle                              | Support      | Temp. | RAG | Conte<br>xte | Think | Moco-<br>do | Nor-<br>malisé | Note | Remarque                                                                                                                                                         |
|------|-------------------------------------|--------------|-------|-----|--------------|-------|-------------|----------------|------|------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| SLM  | Meta<br>Codellama 7b                | Ollama       | 0     | ✗   | ✓            |       | ✓           | ✗              | 2    | Les 3 entités et deux associations. Pas de gestion du président et du trésorier (juste un rôle dans « adhère »). Cardinalité autour de « responsable » à revoir. |
| SLM  | Mistral AI Mistral<br>NeMo-Instrict | LM<br>Studio | 0     | ✗   | ✓            |       | ✓           | ✗              | 3    | Membre vs Personne, Entité Président et Trésorier. Beaucoup d'entités redondante. Cardinalités à revoir.                                                         |
| SLM  | Mistral AI Mistral<br>NeMo-Instrict | LM<br>Studio | 0     | ✓   | ✓            |       | ✓           | ✗              | 3    | Encore des soucis. Soucis avec les cardinalités. Entité « Responsable ».                                                                                         |
| SLM  | Mistral AI<br>Mathstral 7b          | LM<br>Studio | 0     | ✗   | ✓            |       | ✓           | ✗              | 1    | Des entités isolées. 1 seule association.                                                                                                                        |
| SLM  | Mistral AI<br>Mathstral 7b          | LM<br>Studio | 0     | ✓   | ✓            |       | ✓           | ✗              | 3    | Personne vs Membre. Cardinalités à revoir.                                                                                                                       |