**RAPPORT BD**

**Tugrul Ghellab Troles**

**SOMMAIRE**

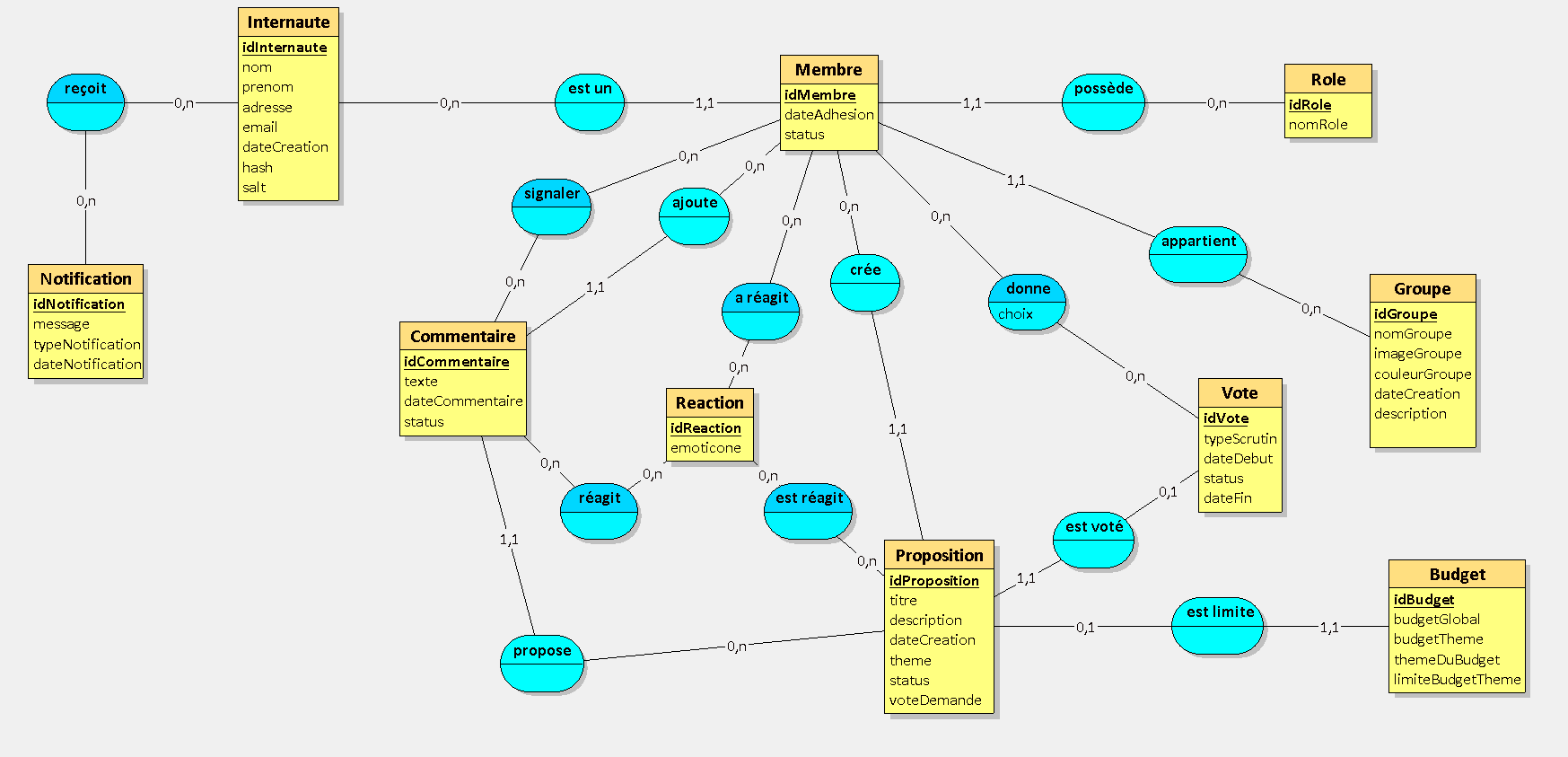
[**I-Modélisation de la base de données** 3](#_Toc187917484)

[**1-** **Modèle conceptuel de données** 3](#_Toc187917485)

[**2-Schéma relationnel** 4](#_Toc187917486)

# **I-Modélisation de la base de données**

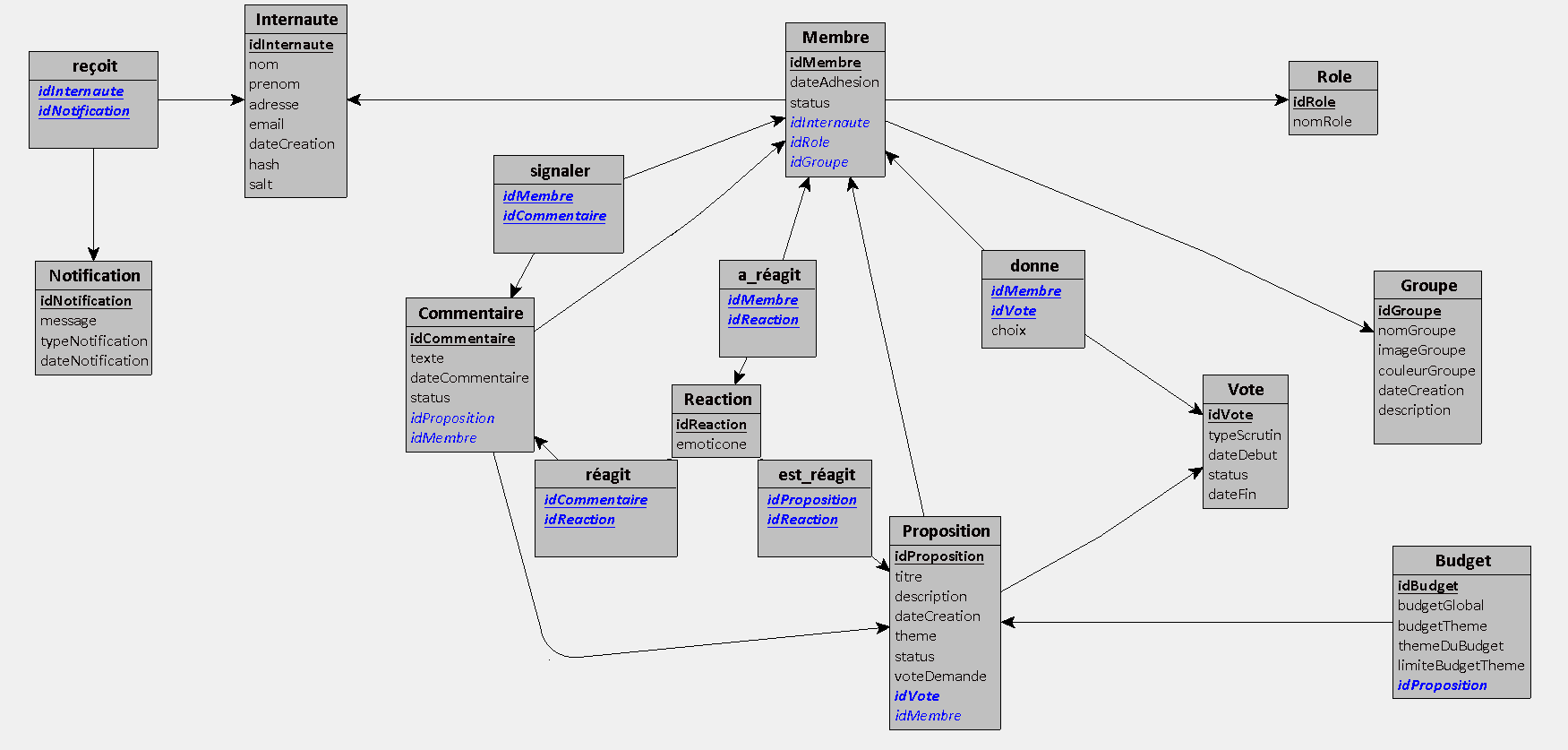
## **Modèle conceptuel de données**



Commentaire du MCD :

Pour la conception de ce MCD nous avons commencé par analyser le sujet en détails dans le but de relever toutes les fonctionnalités nécessaires qui nécessitent d’être modéliser dans notre base de données. Une fois cette première étape passée nous avons entamé une réflexion sur la manière de conceptualiser notre base. Le principale point de réflexion était la manière dont était géré les internautes et les membres d’un groupe. Notre première piste était de créer une table internaute et de permettre à ses internautes de rejoindre un ou plusieurs groupes. Cependant nous nous sommes rapidement aperçus que cette modélisation était limitée. En effet chaque internaute peut avoir un rôle différent suivant les groupes auxquelles il appartient comme par exemple administrateur ou assesseur. C’est à ce moment que nous avons décidé de créer une table intermédiaire intitulée « membre ». L’idée est la suivante : un internaute est unique mais il peut appartenir à plusieurs groupes où son rôle n’est pas le même. Cette conceptualisation nous permet de stocker les informations différentes de chaque internaute sur chacun des groupes auxquelles il appartient. Le reste de la modélisation est plus intuitive, un membre peut commenter, voter ou encore réagir à un commentaire et à une proposition. La table vote est associée à la table proposition, cette même table est reliée à la table budget puisque qu’une proposition est budgétée. Enfin un internaute peut recevoir différentes notifications.

## **2-Schéma relationnel**



# **II- Jeu de test et population de la base de données**

## **1- Vues**

VueMembresGroupes :

Cette vue permet d’afficher la liste de tous les membres avec toutes leurs informations comme leurs groupes, rôles, la date de leur adhésion ainsi que leur nom et leur prénom. Grâce à la vue, les administrateurs ou responsables peuvent facilement visualiser la répartition des rôles au sein d'un groupe. Par exemple, cela permet de savoir combien de membres sont administrateurs, modérateurs ou simples utilisateurs dans un groupe donné. Cela peut être utile pour équilibrer les rôles ou pour ajuster les permissions en fonction des besoins du groupe.

La vue VueMembresGroupes aide à gérer, analyser et suivre les membres d'un groupe en fonction de leurs rôles et de leur adhésion. Elle permet de centraliser les données, d'améliorer la communication, de faciliter la gestion des permissions et d’offrir des outils puissants pour l'analyse des groupes et de leurs membres.

VuePropositions :

Cette vue permet de lister toutes les propositions avec leur auteur, leur date de création et de fin (si fini), leur statut, leur thème et enfin le type de scrutin.

La vue VuePropositions permet la gestion et le suivi des propositions ou de gestion de groupe. Elle permet aux administrateurs et aux utilisateurs de suivre facilement l'état de chaque proposition, de gérer les délais, d'analyser la répartition des propositions par thème ou type de scrutin, et d'assurer un suivi transparent et efficace du processus décisionnel.

En connaissant le statut des propositions (en cours, fermées, etc.), les administrateurs ou les décideurs peuvent gérer le traitement des propositions. Par exemple, une proposition qui est arrivée à sa date de fin et qui a un statut "en attente de résultat" peut être priorisée pour un traitement immédiat, à l’inverse d'une proposition "en cours".

VueCommentairesSignales :

Cette vue permet de retourner les informations d’un commentaire qui a été signalé. Elle renvoie sa date, son contenu, son statut ainsi que toutes les personnes l’ayant signalé. De plus elle renvoie l’auteur du commentaire (à ajouter dans la view).

Cette vue est particulièrement utile pour les modérateurs, car elle facilite leur travail en rassemblant toutes les informations nécessaires pour évaluer un commentaire signalé. En un coup d'œil, les modérateurs peuvent voir quel membre a signalé le commentaire, ce qui peut être utile pour analyser si le commentaire enfreint les règles ou s'il s'agit d'une erreur.

VueVotesChoix :

Cette vue permet de retourner les différents votes avec les choix effectués par les membres du groupe. Cela simplifie l'accès aux données nécessaires pour analyser les votes et leurs participants. Elle améliore la lisibilité, centralise les données utiles. Cette vue pourrait être utilisée comme base de données pour des interfaces utilisateur (par exemple, des tableaux dans une application web) afin de présenter les votes et les choix des membres de manière intuitive et lisible.

VueNotificationsReçues :

Cette vue regroupe les informations des internautes et leurs notifications dans une seule structure, facilitant l'accès aux détails des messages reçus par chaque internaute. La vue est utile pour suivre les notifications envoyées aux utilisateurs, avec des informations clés comme le type, la date, et le statut de chaque notification.

2-Fonctions

**compter\_signalements\_membre** :

Cette fonction permet de calculer le nombre de signalements effectués par un membre spécifique, en fonction de l'ID du membre fourni en paramètre. Elle renvoie le nombre total de signalements pour les commentaires associés à ce membre. Cette fonction est utile pour évaluer la fréquence des signalements d’un membre et peut être utilisée pour des analyses statistiques, ainsi que pour détecter les comportements indésirables ou pour ajuster la gestion des membres dans un système de modération.

**compter\_signalements\_commentaire** :

Cette fonction retourne le nombre de signalements pour un commentaire donné, en utilisant l'ID du commentaire passé en paramètre. Elle est utile pour surveiller l'activité des commentaires signalés et permettre une gestion efficace du contenu. En renvoyant le nombre total de signalements associés à un commentaire, cette fonction permet aux modérateurs ou administrateurs de prendre des mesures appropriées en fonction de la gravité du contenu signalé.

**createGroupe** :

Cette procédure permet de créer un nouveau groupe avec les informations relatives au nom, à l'image, à la couleur, à la description et à la date de création. Une fois le groupe créé, elle lie également un membre au groupe, en créant une adhésion pour un internaute existant. Cette procédure simplifie le processus de création et d'adhésion des groupes, tout en automatisant l'insertion des données dans les tables concernées. Elle peut être utilisée par les administrateurs pour organiser facilement des groupes et y inclure les membres de manière transparente.

**createInternaute** :

Cette procédure permet d’ajouter un nouvel internaute à la base de données, en incluant des informations telles que le nom, le prénom, l’adresse, l’email ainsi que les informations de sécurité (hash et salt pour le mot de passe). Elle permet ainsi de faciliter l’enregistrement d'un utilisateur dans le système, en veillant à sécuriser ses informations sensibles et en garantissant un processus d’inscription fluide et rapide. Elle est particulièrement utile pour les administrateurs souhaitant ajouter de nouveaux utilisateurs à la plateforme sans avoir à saisir manuellement toutes les données.

**notification\_suppression** :

Ce trigger est déclenché après la suppression d'une proposition. Lorsqu'une proposition est supprimée, il crée automatiquement une notification indiquant le titre de la proposition supprimée, puis l'insère dans la table Notification. Ensuite, il récupère l'ID de cette notification et l’associe à tous les membres du groupe lié à la proposition supprimée dans la table InternauteNotification. Ce trigger permet de notifier tous les membres concernés par la suppression d'une proposition, garantissant ainsi que tous les utilisateurs du groupe soient informés des changements pertinents. Il est particulièrement utile pour la transparence et la communication au sein des groupes, en assurant une gestion cohérente des modifications.

**trg\_before\_insert\_internaute** :

Ce trigger est déclenché avant l’insertion d’un nouvel internaute dans la table Internaute. Il vérifie d'abord si l'email fourni pour le nouvel internaute existe déjà dans la base de données. Si c’est le cas, une erreur est levée pour empêcher l'insertion et garantir qu’aucun doublon d'email ne soit créé. De plus, il prévoit la validation de l’email selon une expression régulière (commentée dans le code). Ce trigger est utile pour éviter les problèmes de duplication d'emails dans la base de données et pour garantir l'unicité des comptes utilisateur. Cela simplifie la gestion des utilisateurs et améliore la qualité des données dans le système.