

# Projet Bases de Données : Big Brother



Hergaux Alexandre  
Berne--Gallieri Tom  
Bernardin Paul






# Projet Bases de Données : Big Brother



## Sommaire :

- I) Présentation du sujet et problématique
  - II) Modélisation MCD
  - III) Modélisation des tables
  - IV) Réponse technique à la problématique
  - V) Les requêtes
  - VI) Problèmes rencontrés et points à améliorer
- 






# I) Présentation du sujet et problématique



Big Brother fait référence à un concept de surveillance omniprésente et autoritaire, souvent associé au contrôle gouvernemental sur les individus. L'expression vient du roman *1984* de George Orwell, où "Big Brother" est un dirigeant fictif qui symbolise un régime totalitaire.



Dans ce contexte, Big Brother surveille constamment la population à travers des dispositifs comme les télécrans et impose une propagande visant à contrôler les pensées et les actions des citoyens.





# I) Présentation du sujet et problématique



Nous avons donc imaginé un SGBD qui possède des données sur des personnes et qui peut se nourrir de données externes en agrandissant ses données internes et ainsi faire grandir sa base de données.

Les données externes sont des fichiers .csv de données sensibles de différents domaines : données bancaires, sécurité sociale ou même des listes d'employés hauts placés d'entreprises.







# I) Présentation du sujet et problématique

Notre base de données se crée de la forme suivante :

- Création d'une table "Personne" qui est le coeur de la base de donnée
- Lecture d'un fichier .csv qui comporte une grande quantité de données sensibles et personnelles
- Création d'une table à partir du fichier.csv
- Normaliser les données importées
- Vérifier et créer les tables de références pour les nouveaux attributs
- Remplir ces tables de références avec les bonnes données
- Créer la table cible avec les bonnes données
- Remplir cette table cible
- Supprimer la table temporaire





# I) Présentation du sujet et problématique

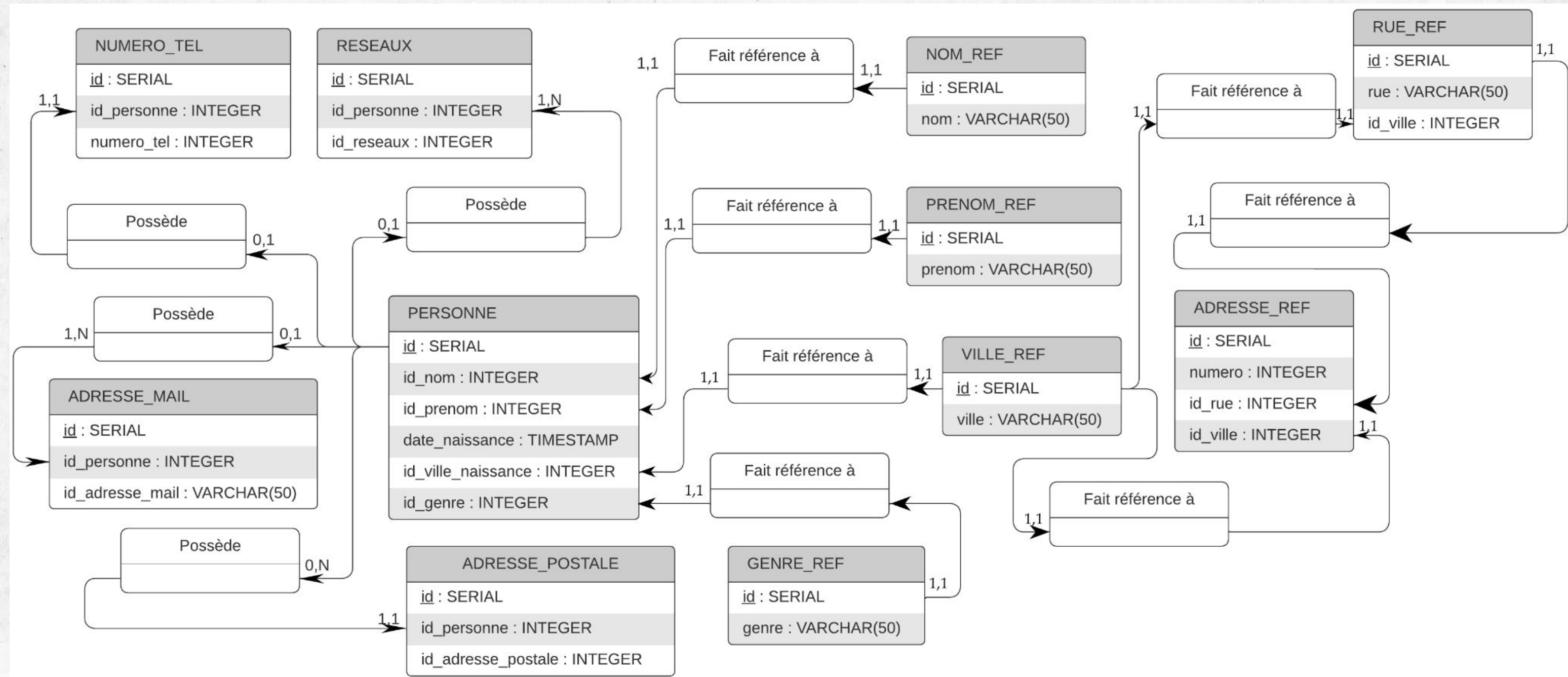


## Problématique :

Comment centraliser, normaliser et intégrer automatiquement des données provenant de sources diverses tout en garantissant leur cohérence et leur unicité dans une bases de données relationnelle ?

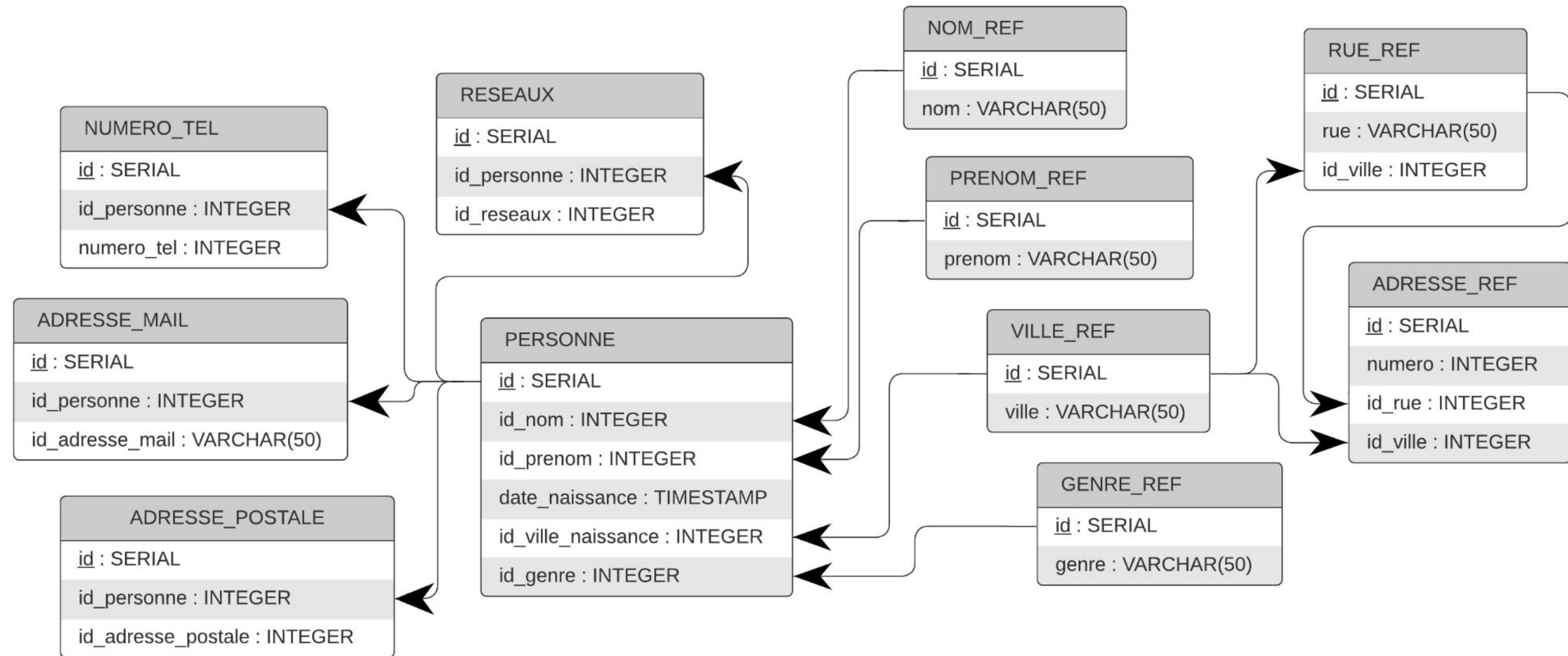


## II) Modélisation MCD





### III) Modélisation des tables







## IV) Réponse technique à la problématique

Les solutions techniques pour répondre à la problématique sont :

- a) Centralisation des données
- b) Normalisation des données
- c) Cohérence des données
- d) Audit et traçabilité des modifications






## IV) Réponse technique à la problématique



### a) Centralisation des données

- Lecture du fichier .csv
  - Création d'une table temporaire
  - Insertion des données brutes
  - Identification des correspondances avec les données préexistantes
  - Association : Si correspondance est trouvée -> l'enregistrement est fait  
Sinon -> création d'une nouvelle personne
  - Insertion dans la table personne et renvoie de leur identifiant
  - Liaison avec les autres tables
  - Centralisation et automatisation
- 





## IV) Réponse technique à la problématique



### b) Normalisation des données

- `normalize_table_contain` détecte les TEXT et retire les accents ainsi que “ ” et “-” -> “ \_”
- La normalisation est centralisée dans les tables de référence
- Insertion de données dans ces tables seulement si elle est bien normalisée






## IV) Réponse technique à la problématique



### c) Cohérence des données

- Gestion des conflits avec **ON CONFLICT DO NOTHING**
  - **lier\_personne\_import** identifie si les informations importées correspondent à une personne existante
  - **insert\_into\_corresponding\_table** vérifie si les données peuvent alimenter une table importante sinon ajoute ces données dans les tables correspondantes
- 






## IV) Réponse technique à la problématique



### d) Audit et traçabilité des modifications

- `create_audit_changes` crée une table d'audit pour stocker l'historique des modifications
  - `trigger_audit` définit les instructions de surveillance des modifications
  - Mise en place de trigger pour les tables importantes :
    - a) Enregistrement de l'utilisateur ayant effectué la modification
    - b) Enregistrement des données avant et après la modification
    - c) Enregistrement de la date et l'heure de l'opération
- 





## V) Les requêtes



Les requêtes demandées avaient comme prérequis :

Au moins 2 requêtes comportant 1 niveau de sous-requête

Au moins 2 requêtes comportant 2 niveaux de sous-requête

Au moins 5 requêtes comportant des jointures de tous types (LEFT / RIGHT / INNER / OUTER)

Au moins 2 requêtes de création de vue (persistantes ou non)

Au moins 2 requêtes utilisant ces vue

Au moins 1 requête justifiant un UNION / EXCEPT / INTERSECT







## V) Les requêtes







## VI) Problèmes rencontrés et points à améliorer



### Problèmes rencontrés :

- Différencier les tables minimales et maximales lors de l'initialisation de la base
- Quels choix prendre afin de normaliser, architecturer et faire les liens entre les données






## VI) Problèmes rencontrés et points à améliorer



### Problèmes rencontrés :

- Agencer ainsi que faire les liens entre les données prend différentes formes : il fallait donc s'accorder sur quel type de données nous allons recevoir.
  - Savoir si deux personnes avec des données très similaires sont bien la même personne. Il a fallu déléguer cette responsabilité à une règle statistique
- 





## VI) Problèmes rencontrés et points à améliorer



### Améliorations possibles :

Accepter des fichiers .csv autres que ceux décrivant une personne, tel que des entreprises ou partis politiques.

Cela aurait nécessité un algorithme liant les noms de tables avec des attributs préexistants qui ensuite remplace les valeurs par une clé étrangère faisant réfère au sujet de la table.

Ce genre d'algorithme nécessite cependant beaucoup de conditions afin qu'il puisse être fonctionnel

