Projet de Base de Données : Application Web pour l'Exploration et l'Analyse des E-mails d'Enron

Souheid Nasser Djama et Jules Malard

15 octobre 2025

1 Introduction

L'affaire Enron est considérée comme l'un des plus grands cas de manipulation financière aux États-Unis. Parmi les différentes pistes d'enquête, l'analyse des e-mails échangés au sein de l'entreprise a joué un rôle essentiel pour comprendre les interactions entre les acteurs impliqués. Ce projet a pour objectif de développer une application web destinée aux enquêteurs afin de leur permettre d'explorer, de visualiser et d'analyser efficacement les données issues des e-mails d'Enron.

2 Contexte

2.1 Le Sujet

L'objectif est de concevoir une application intuitive utilisable par des enquêteurs ne disposant pas nécessairement de compétences techniques avancées. Toutes les requêtes doivent être formulées en SQL. L'application proposera un ensemble de formulaires correspondant à des interrogations types, chacun permettant à l'utilisateur de définir les paramètres nécessaires à la génération automatique d'une requête SQL.

2.2 Les Données

Pour ce projet, nous disposons :
- d'un fichier XML répertoriant chaque employé avec son nom, prénom, une adresse principale de messagerie, une liste d'adresses e-mail secondaires et, le cas échéant, sa catégorie dans l'entreprise ;
- d'une structure arborescente de fichiers texte représentant les e-mails. Chaque fichier contient le corps du message ainsi que diverses métadonnées (expéditeur, destinataires, date, sujet, etc.).

La racine de cette structure est un dossier `maildir`, contenant un sous-répertoire par boîte mail employé. Chaque sous-répertoire renferme les fichiers correspondant aux e-mails reçus et envoyés.

2.3 Implémentation Technique

Les principaux outils utilisés sont :
- **Framework Web** : Django
- **SGBD** : PostgreSQL
- **Peuplement de la base** : Python
- **Gestion de versions** : Git (dépôt hébergé sur GitHub)

3 Base de Données

3.1 Modélisation de la BDD

La première étape du projet a consisté à concevoir un schéma relationnel efficace, permettant :

1. d'exécuter facilement les requêtes SQL nécessaires pour répondre aux besoins fonctionnels ;
2. de peupler la base rapidement et sans opérations manuelles lourdes.

Deux tables principales structurent notre modèle :
— `mail` : identifiée par le `message-ID`, contenant l'expéditeur, la date, le sujet et le contenu du mail ;
— `mailbox` : identifiée par une adresse e-mail, avec un champ `is_internal` indiquant si l'adresse est interne à l'entreprise.

`mailbox` est liée à `employee` par une relation « plusieurs à au plus un » : un employé peut posséder plusieurs boîtes mail, mais une boîte mail est associée à un seul employé au maximum. Les tables `mail` et `mailbox` sont reliées par des relations « plusieurs à plusieurs » via trois tables : `to`, `cc` et `bcc`, car un mail peut avoir plusieurs destinataires et inversement.

Pour accélérer certaines requêtes, nous avons ajouté une table `word`, en relation « plusieurs à plusieurs » avec `mail`, afin d'associer chaque message aux mots qu'il contient. Cela permet notamment d'optimiser la recherche par mots-clés.

3.2 Gestion des Conversations

La requête liée à l'identification des conversations a constitué un défi majeur. Nous avons choisi de regrouper dans une même conversation tous les e-mails partageant le même sujet et au moins un message commençant par « Re : ». Cette approche, bien que simplificatrice, permet une implémentation efficace et une exploitation rapide par l'utilisateur. Nous avons également ajouté une table `conversation`, liée aux tables `mail` et `mailbox`, pour modéliser la participation des différentes boîtes aux discussions.

3.3 Script de Peuplement

Le peuplement est assuré par un script Python composé de deux fonctions principales :
— `fill_employee_and_mailbox` : extrait les informations du fichier XML pour remplir les tables `employee` et `mailbox` ;
— `fill_mails` : parcourt la structure de fichiers des e-mails et remplit les tables `mail`, `conversation`, `to/cc/bcc` et `word`.

Nous utilisons le module `xml.etree.ElementTree` pour le XML et `re` (regex) pour extraire les informations des fichiers mails.

Les premières versions du script étaient trop lentes (plusieurs jours d'exécution). Pour y remédier :
— nous avons adopté la méthode `bulk_create` afin d'insérer les données en lot ;
— nous avons compilé les regex une seule fois pour éviter les recompilations coûteuses.

Ces optimisations ont permis de réduire drastiquement le temps de traitement à quelques heures.

4 Application

Au lancement, l'utilisateur accède à une page d'accueil proposant huit formulaires. Chaque formulaire correspond à une requête SQL spécifique, générée automatiquement à partir des paramètres saisis.

1. Analyse de profil employé

Permet de consulter les informations relatives à un employé (identité, catégorie, adresses e-mail).

```
SELECT e.id, e.firstname, e.lastname, e.category, m.address
FROM mail_form_employee e JOIN mail_form_mailbox m
ON e.id = m.employee_id
WHERE e.lastname = 'Badeer';
```

2. Analyse des flux de messagerie

Permet d'identifier les employés ayant envoyé ou reçu plus (ou moins) d'un certain nombre de mails dans un intervalle donné.

```
SELECT e.id, e.firstname, e.lastname, e.category, mb.address
FROM mail_form_employee e
INNER JOIN mail_form_mailbox mb ON e.id=mb.employee_id
WHERE (SELECT COUNT(DISTINCT mm.message_id)
    FROM mail_form_mailbox m
    INNER JOIN mail_form_mail mm ON mm.sender_id = m.address
    INNER JOIN mail_form_mail_to mt ON mm.message_id = mt.mail_id
    INNER JOIN mail_form_mailbox m_to ON mt.mailbox_id = m_to.address
    WHERE m.employee_id = e.id
    AND mm.date BETWEEN '01/01/2001' AND '10/01/2002') > 500;
```

3. Analyse des communications inter-employés

Permet d'obtenir la liste des employés ayant communiqué avec une personne donnée.

```
SELECT e.id, e.firstname, e.lastname, e.category, mbox.address
FROM mail_form_mail m
INNER JOIN mail_form_mailbox mb_sender ON mb_sender.address = m.sender_id
INNER JOIN mail_form_mail_to t ON t.mail_id = m.message_id
INNER JOIN mail_form_mailbox mb_to ON t.mailbox_id = mb_to.address
INNER JOIN mail_form_employee e_sender ON mb_sender.employee_id = e_sender.id
INNER JOIN mail_form_employee e_to ON mb_to.employee_id = e_to.id
INNER JOIN mail_form_employee e ON (e.id = e_to.id OR e.id = e_sender.id )
    AND e.id != 62
INNER JOIN mail_form_mailbox mbox ON mbox.employee_id = e.id
WHERE (e_sender.id = 62 OR e_to.id = 62)
AND m.date BETWEEN '01/01/2001' AND '10/01/2002'
GROUP BY e.id, mbox.address;
```

4. Analyse des échanges par paires

Affiche les couples d'employés ayant le plus échangé dans une période donnée.

```
SELECT e1.id, e1.firstname, e1.lastname, e1.category,
       e2.id, e2.firstname, e2.lastname, e2.category, pairs.mail_count
FROM (
    SELECT DISTINCT LEAST(mb_sender.employee_id, mb_to.employee_id) AS id_1,
        GREATEST(mb_sender.employee_id, mb_to.employee_id) AS id_2,
        COUNT(m.message_id) AS mail_count
    FROM mail_form_mail m
    INNER JOIN mail_form_mailbox mb_sender ON mb_sender.address = m.sender_id
    INNER JOIN mail_form_mail_to t ON t.mail_id = m.message_id
    INNER JOIN mail_form_mailbox mb_to ON t.mailbox_id = mb_to.address
    WHERE m.date BETWEEN '01/01/2001' AND '10/01/2002'
      AND mb_sender.employee_id IS NOT NULL
      AND mb_to.employee_id IS NOT NULL
    GROUP BY id_1, id_2
) AS pairs
INNER JOIN mail_form_employee e1 ON pairs.id_1 = e1.id
INNER JOIN mail_form_employee e2 ON pairs.id_2 = e2.id
ORDER BY pairs.mail_count DESC
```

```
LIMIT 10;
```

5. Analyse des pics d'échanges quotidiens

Identifie les journées les plus actives en termes d'échanges internes ou externes.

```
SELECT DATE(m.date), COUNT(*)
FROM mail_form_mail m
WHERE date BETWEEN '01/01/2001' AND '10/01/2001'
AND m.message_id IN (
    SELECT mm.message_id
    FROM mail_form_mail mm
    INNER JOIN mail_form_mailbox mb_sender ON mb_sender.address = mm.sender_id
    INNER JOIN mail_form_mail_to t ON t.mail_id = mm.message_id
    INNER JOIN mail_form_mailbox mb_to ON t.mailbox_id = mb_to.address
    WHERE mb_sender.is_internal = TRUE AND mb_to.is_internal = TRUE
)
GROUP BY DATE(date)
ORDER BY COUNT(*) DESC;
```

6. Recherche par mots-clés

Permet de rechercher les mails contenant une liste de mots.

```
SELECT m.message_id, m.subject, m.sender_id, t.mailbox_id, m.content, m.date
FROM mail_form_mail m
LEFT JOIN mail_form_mail_to t ON m.message_id = t.mail_id
INNER JOIN mail_form_mail_word mw1 ON mw1.mail_id = m.message_id
INNER JOIN mail_form_mail_word mw2 ON mw2.mail_id = m.message_id
INNER JOIN mail_form_mail_word mw3 ON mw3.mail_id = m.message_id
WHERE mw1.word_id='mail' AND mw2.word_id='Enron' AND mw3.word_id='Company'
ORDER BY m.subject;
```

7. Analyse des conversations

Affiche les mails associés à une conversation donnée.

```
SELECT c.subject, cp2.mailbox_id
FROM mail_form_conversation c
INNER JOIN mail_form_conversation_participants cp
    ON cp.conversation_id = c.subject
INNER JOIN mail_form_mailbox mb
    ON mb.address = cp.mailbox_id
INNER JOIN mail_form_employee e
    ON mb.employee_id = e.id
INNER JOIN mail_form_conversation_participants cp2
    ON cp2.conversation_id = c.subject
WHERE e.id = 87;
```

8. Requête SQL libre

Un formulaire libre permet à l'utilisateur d'exécuter toute requête SQL valide et de visualiser le schéma de la base.

5 Conclusion

Cette application web basée sur Django et PostgreSQL offre une solution performante et flexible pour l'exploration des données de l'affaire Enron. Elle permet de réaliser des analyses ciblées sans nécessiter d'expertise informatique approfondie.

Des améliorations sont néanmoins envisageables :

— affiner l'algorithme de détection des conversations ;
— enrichir les visualisations interactives ;
— intégrer des outils de statistiques avancées pour mieux exploiter les flux de communication.