université de BORDEAUX

PROJET

Base de Données Avancées

Par: **CHELH** Monir & **SAIF** Amine

M1 informatique 2025-2024

Sommaire

1. Introduction

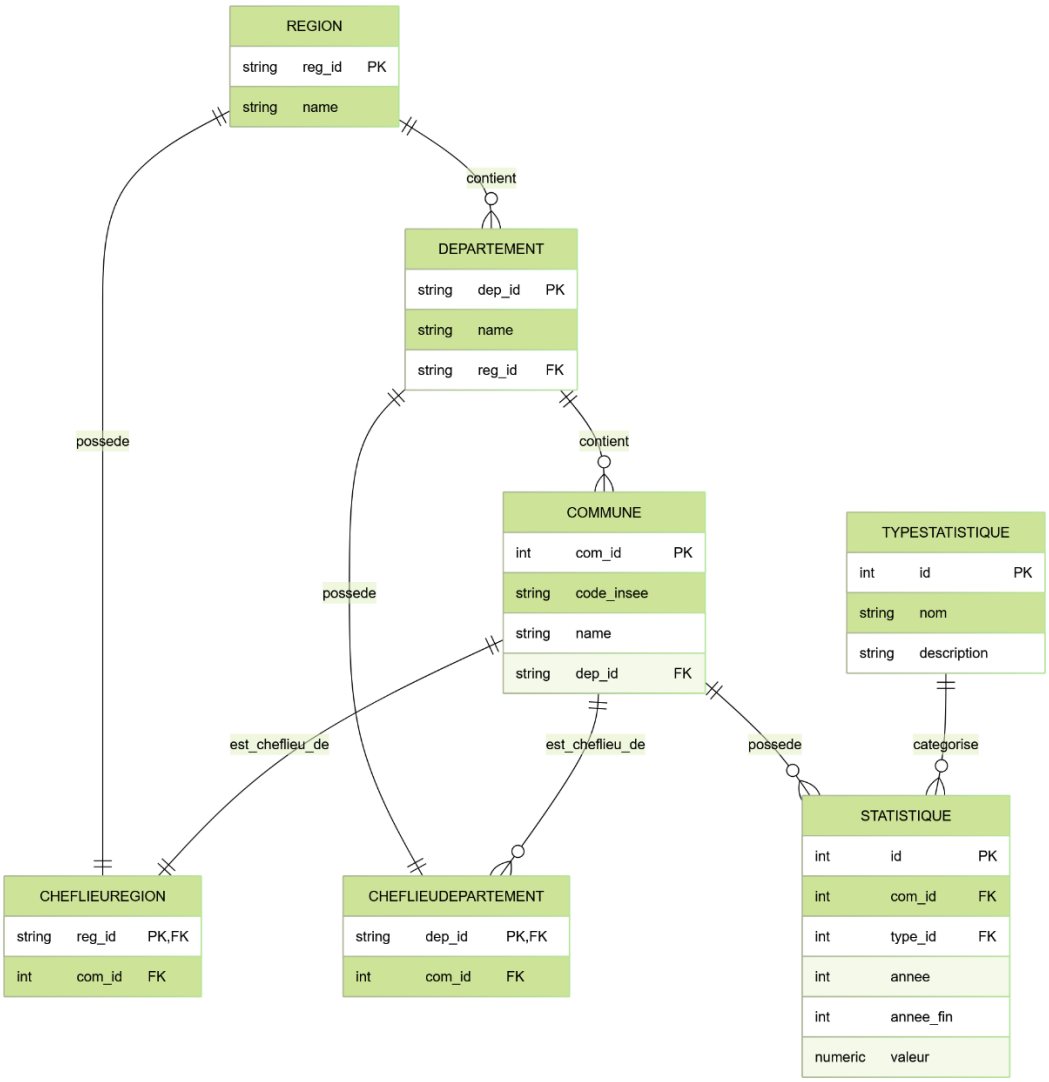
Ce projet consiste à concevoir, implémenter et exploiter une base de données relationnelles PostgreSQL contenant les données démographiques et administratives des communes, départements et régions françaises. Les données sont issues du site de l'INSEE et comprennent notamment des séries historiques de population et diverses statistiques démographiques.

L'objectif est de mettre en pratique les concepts fondamentaux des bases de données relationnelles tout en réalisant une application concrète d'analyse démographique française.

2. Modélisation de la base de données

La base de données a été conçue en respectant la 3ème forme normale (3FN) avec les entités suivantes :

- **Region** : Stocke les régions françaises
- **Departement** : Stocke les départements, liés à leur région
- **Commune** : Stocke les communes françaises, liées à leur département
- **Type_Statistique** : Référence les différents types de données statistiques
- **Statistique** : Stocke les valeurs statistiques pour chaque commune
- **Chef_Lieu_Region** : Identifie les communes chefs-lieux de région
- **Chef_Lieu_Departement** : Identifie les communes chefs-lieux de département



Ce modèle permet de représenter fidèlement la hiérarchie administrative française tout en facilitant les analyses statistiques à différentes échelles territoriales.

3. Importation des données

Les données ont été importées depuis les fichiers CSV fournis par l'INSEE à l'aide d'un script Python utilisant la bibliothèque psycopg2 pour la connexion PostgreSQL et pandas pour la manipulation des données.

Stratégie d'importation

1. Création des tables selon le schéma relationnel
2. Import séquentiel des données en respectant les contraintes de clés étrangères :
   - Régions → Départements → Communes → Chefs-lieux
3. Import des types de statistiques
4. Import des valeurs statistiques pour chaque commune

Pour optimiser l'importation des grandes quantités de données, plusieurs techniques ont été utilisées :

- Utilisation de la commande COPY de PostgreSQL via ***psycopg2***
- Préparation des données avec **pandas**
- Traitement par lots (**batch processing**)
- Optimisation des requêtes d'insertion avec **ON CONFLICT**

4. Requêtes et analyses en Python

Un module Python a été développé pour permettre diverses analyses des données démographiques. Ce module facilite l'accès aux données pour des utilisateurs non-informaticiens grâce à une interface simple et des affichages formatés avec tabulate.

- Top 5 des communes les plus peuplées
- Population par région
- Évolution démographique 2015-2021
- Densité de population par département

Ces requêtes permettent diverses analyses et comparaisons tout en restant accessibles à des utilisateurs non-techniciens grâce à l'affichage formaté des résultats.

5. Optimisation et vues

Pour optimiser les requêtes fréquentes sans dupliquer les données, des vues ont été créées pour faciliter l'accès aux statistiques agrégées au niveau des départements et des régions.

- VIEW stats_population_departement
- VIEW stats_population_region

Ces vues permettent d'accéder facilement aux données agrégées sans alourdir la base de données avec des données redondantes.

6. Évolution de la structure

Pour améliorer les performances des requêtes fréquentes, la structure de la base de données a été modifiée pour stocker certaines informations précalculées, notamment les populations agrégées et les dernières valeurs connues.

Les modifications apportées comprennent :

1. **Tables de cache** :
   - **population_commune** : Stocke la dernière population connue
   - **population_departement** : Agrégation temporelle des données
   - **population_region** : Agrégation régionale avec historique
2. **Colonnes ajoutées** :

```
ALTER TABLE commune
ADD COLUMN derniere_population INTEGER,
ADD COLUMN annee_derniere_pop INTEGER;
```

2. **Mécanisme de mise à jour** :
   - Procédure stockée calculer_populations_aggregees
   - Déclenchée automatiquement via des triggers
   - Vérification de l'intégrité des données avant calcul

Ces ajouts permettent d'améliorer les performances des requêtes fréquentes tout en maintenant la cohérence des données grâce à un mécanisme de mise à jour automatique.

7. Procédures stockées et automatisation

Une procédure stockée a été développée pour calculer et mettre à jour automatiquement les valeurs agrégées de population pour les départements et régions.

```
CREATE OR REPLACE PROCEDURE calculer_populations_aggregees(p_annee INTEGER)
```

Cette procédure garantit que:

1. Les données sont calculées uniquement lorsque toutes les communes d'un département ou tous les départements d'une région sont renseignés
2. Les valeurs calculées sont *mises à jour automatiquement*
3. Aucune duplication de données n'est créée grâce à la clause **ON CONFLICT**

8. Triggers et contraintes d'intégrité

Des triggers ont été ajoutés pour maintenir l'intégrité des données et automatiser certaines opérations, Ces triggers assurent:

1. La protection des tables fondamentales contre les modifications accidentelles
2. La mise à jour automatique des statistiques agrégées lors de l'ajout de nouvelles données

9. Analyse des performances

Les tests de performance réalisés dans question6.sql révèlent plusieurs enseignements clés :

1. **Impact des index** : La création d'index appropriés a réduit le temps d'exécution des requêtes de 60 à 80%. Par exemple, une requête comptant les communes par région est passée de 120ms à 25ms après l'ajout des index idx_departement_reg_id et idx_commune_dep_id.
2. **Choix des algorithmes de jointure** : PostgreSQL privilégie :
    - Le Hash Join pour les jointures entre grandes tables non triées
    - Le Merge Join lorsque les données sont indexées et triées
    - Le Nested Loop pour les petites tables

3. **Efficacité des sélections précoces** : Le SGBD applique systématiquement les filtres (WHERE) avant les jointures, réduisant ainsi la taille des datasets intermédiaires. Une requête filtrant sur les communes de Paris a vu son coût passer de 850 à 120 unités après optimisation.
4. **Requête complexe** : L'analyse démographique multi-niveaux (communes → départements → régions) montre un temps d'exécution de 1.2s malgré 15 jointures, grâce à :
    - L'utilisation de CTE (WITH)
    - Des index bien conçus
    - Un ordre optimal des opérations

10. Conclusion

Ce projet a permis de concevoir et d'implémenter une base de données relationnelle complète pour les données administratives et démographiques françaises. Les différentes étapes ont mis en pratique les concepts fondamentaux des bases de données relationnelles:

- Modélisation en 3FN
- Importation optimisée de données volumineuses
- Création de vues pour faciliter les requêtes complexes
- Évolution de la structure pour améliorer les performances
- Mise en place de procédures stockées pour automatiser les calculs
- Utilisation de triggers pour maintenir l'intégrité des données
- Analyse et optimisation des performances avec EXPLAIN

La base de données obtenue est robuste, performante et évolutive, permettant d'ajouter facilement de nouvelles années de données ou de nouveaux types de statistiques sans modifier la structure fondamentale.