	Université de Corse - Pasquale PAOLI	
	Diplôme : **M1 DFS-DE**	**2025-2026**
	Module : Gestion de Bases de données et dataWarehouses **TP1 - Base relationnelle PostgreSQL** **Modéliser l'écosystème de la recherche** **Enseignant : Evelyne VITTORI**	

Ce TP constitue la première étape d'un projet fil rouge **« Écosystème de la recherche : de la modélisation relationnelle à l'entrepôt de données »** qui porte sur la gestion et l'analyse des données de la recherche universitaire. Il est organisé en trois étapes complémentaires : modéliser l'ensemble de l'**écosystème de la recherche** dans une base relationnelle PostgreSQL (TP1), stocker les **données effectives** des jeux de données scientifiques dans MongoDB (TP2), puis construire un **Data Warehouse** pour réaliser des analyses décisionnelles avec SQL avancé (TP3). L'ensemble illustre la complémentarité entre bases relationnelles, NoSQL et entrepôts de données.

Objectif TP1

Il s'agit de concevoir et implémenter une base de données relationnelle pour stocker les informations sur les institutions (Université de Corse, CNRS…), les laboratoires de recherche (UMR SPE, UMR LISA, Stella Mare…), les chercheurs et leurs contrats de recherche (ANR, Région, Europe…), ainsi que les projets structurants menés au sein de ces laboratoires. Elle doit également permettre de gérer les publications des chercheurs et les métadonnées des jeux de données produits par ces projets (description, auteurs, licences, conditions d'accès, etc.).

Cette base sera utilisée par plusieurs profils : les administrateurs et data managers, qui pourront suivre les projets, gérer les contrats, vérifier les dépôts de données et contrôler la cohérence des métadonnées, et les chercheurs, qui pourront déclarer leurs projets, consulter leurs contrats, publier ou modifier les informations relatives aux jeux de données.

Le travail que vous devrez réaliser consiste à créer la base de données sous PostgreSQL, y insérer des données fictives de manière automatique (ou utiliser des données réelles lorsqu'elles existent (laboratoires, UMR)) et vous appuyer sur ces données pour mettre en pratique les notions abordées dans le cadre du cours : rôles et privilèges, requêtes et optimisation, triggers et procédures stockées.

Attention, vous pouvez utiliser quelques données réelles mais l'objectif n'est pas de passer du temps à collecter des données réelles, mais de construire un corpus cohérent et représentatif permettant de tester les fonctionnalités attendues.

Dans un premier temps, vous devrez mettre à l'épreuve votre maitrise des concepts des bases de données relationnelles pour être en mesure d'utiliser des outils avancés d'intelligence artificielle et de génération automatique pour concevoir une base de données pertinente et cohérente.

Objectifs Clés du TP

1. **Conception guidée :** Utiliser un (ou plusieurs) outil d'IA génératives pour faciliter et optimiser la phase de conception, en adoptant une posture critique. Vous devrez évaluer constamment les recommandations proposées afin de parvenir à la création d'une base de données valide.
2. **Peuplement éclairé :** Utiliser (partiellement) Python Faker pour peupler la base de données de données fictives et réalistes, avec une attention particulière à la gestion des clés étrangères pour assurer l'intégrité référentielle.
3. **Gestion Sécurisée des Accès :** Mettre en œuvre une stratégie robuste de gestion des utilisateurs à travers des rôles, des vues et des privilèges pour contrôler l'accès aux données.
4. **Optimisation et Performance :** Explorer des techniques avancées pour améliorer l'efficacité des requêtes, notamment par l'analyse des plans d'exécution et la mise en place d'index stratégiques.
5. **Maîtrise de la Programmation PL-SQL :** Acquérir des compétences en PL-SQL en mettant en place des triggers et en élaborant des procédures stockées pour des opérations courantes et complexes.

Rendus

Date rendu : jeudi 12 novembre

Ce travail doit être réalisé en groupes de 3 étudiants.
Le rendu sera effectué dans l'onglet travaux sur l'ENT sous la forme d'un **seul fichier archive (zip ou rar)** contenant l'ensemble des fichiers demandés :

- Rapport synthétique contenant :
  - Description détaillée de la conception de la base.
  - Méthodologie pour l'utilisation des outils d'IA générative et Python Faker.
  - Notice explicative de vos scripts.
  - Comparaison des requêtes complexes avec captures d'écran des outils de diagnostic.
- Scripts SQL
  - Création du schéma de la base de données et insertion des données
  - Création des utilisateurs, des vues et attribution des privilèges
  - Requêtes complexes (différentes versions)
  - Création des procédures stockées et des triggers.
- Scripts Python utilisés pour générer les données des différentes tables.

Soutenance Orale

- Date prévue : 12 novembre
- Présentation de 20 minutes décrivant la démarche et les résultats obtenus :
  Schéma, données, vues, requetes, procédures, triggers
- Démonstration
- Proposition d'une méthodologie sur l'utilisation des outils d'IA génératives

Critères d'évaluation

1. **Conception de la base de données**
    - Pertinence et intégrité du schéma
    - Cohérence des données générées
    - Originalité de la solution proposée et justification des choix
2. **Maitrise des solutions proposées**
    - **Vues et privilèges**
        - Pertinence et efficacité des vues créées et de l'attribution des privilèges
    - **Requêtes complexes**
        - Pertinence et exactitude des requêtes
        - Efficacité et optimisation des requêtes
        - Comparaison des performances et justification des choix
    - **Procédures stockées et triggers**
        - Fonctionnalité et efficacité des procédures stockées
        - Efficacité et pertinence des triggers
3. **Documentation et méthodologie**
    - Clarté de la méthodologie et analyse critique de l'aide apportée par les outils d'IA génératives et Python Faker
    - Qualité de la documentation fournie, incluant les commentaires dans le code

Partie 1 - Conception de la Base de Données

Il s'agit ici d'élaborer un schéma conceptuel pour une base de données destinée à la gestion des données de recherche d'une université.

L'un des défis centraux de cette partie réside dans l'intégration efficace et judicieuse d'outils d'intelligence artificielle dans le processus traditionnel de conception d'une base de données. En particulier, l'utilisation d'outils d'IA génératives s'avère être un outil précieux pour faciliter et accélérer la phase de conception. Toutefois, il est crucial d'être capable non seulement d'utiliser une IA générative comme un outil d'assistance, mais aussi de porter un regard critique sur les suggestions et les aides proposées. Analyser la pertinence, la précision et l'adéquation des solutions offertes par ces outils avec les exigences réelles du projet est essentiel pour garantir une conception robuste et pertinente de la base de données.

Cahier des charges indicatif

La base de données doit permettre de gérer :

- Institutions : nom, type (Université, organisme de recherche, partenaire privé), adresse.
- Laboratoires (UMR) : identifiant, nom, rattachement institutionnel. Chaque chercheur appartient à un seul laboratoire.
- Projets de recherche structurants : identifiant, titre, description, discipline, budget annuel, dates de début et de fin, laboratoire pilote, contrats associés. Un projet implique plusieurs chercheurs (voir https://www.universita.corsica/fr/recherche/axes-de-recherche/ pour des exemples de projets structurant au sein de l'université de corse).

- Un projet de recherche implique plusieurs chercheurs mais il est dirigé par un seul chercheur responsable du projet.
- Contrats de financement : type (ANR, H2020, Région…), financeur, intitulé, montant, durée, date de début et de fin. A titre de simplification on supposera qu'un contrat ne finance qu'un seul projet structurant. En revanche, un projet structurant peut etre financé par plusieurs contrats.
- Chercheurs : identité, … , discipline, laboratoire de rattachement. A titre de simplification on supposera qu'un chercheur n'est impliqué que dans un seul projet structurant.
- Publications : elles sont rédigées par un ou plusieurs chercheurs. On stocke dans la base de données, les métadonnées de ces publications uniquement (titre, auteurs, taille, DOI, date, …). Les fichiers pdf des publications n'ont pas à être stockés dans la base de données, ils sont supposés déposés sur un dépôt d'archives ouvertes type HAL(https://doc.hal.science/). Seul lee lien vers le fichier sur ce depot sera associé à chaque publication dans ses métadonnées.
- Un jeu de données est un ensemble de données (dataset) associées à un contrat de recherche (résultats de recherche). Il peut bien sur y avoir plusieurs jeux de données pour un même contrat. On stocke ici uniquement les descriptifs de ces jeux de données : les Métadonnées des jeux de données. Il s'agit des caractéristiques suivantes : description, auteur(chercheur), conditions d'accès, licences, date de dépôt, …. On ne stocke ici que les descriptions des jeux de données pas les jeux de données eux-même. Ceux-ci seront stockés dans une BD NoSQL (cf. TP2).
- Chaque contrat de recherche doit être associé à un Plan de Gestion des Données (DMP). Le DMP décrit la manière dont les données issues des projets financés par ce contrat seront produites, documentées, partagées et archivées. Dans la base de données, on conservera pour chaque contrat : le statut du DMP (brouillon, soumis, validé), la date de validation, un lien vers le document complet (stocké sur un support externe).
- Les jeux de données financés par un contrat ne peuvent être officiellement déposés que si le DMP associé au contrat est validé

⚠ Vous pouvez enrichir ce modèle avec vos propres choix ou hypothèses (à préciser).

Le cahier des charges est volontairement assez vague afin que chaque groupe fasse preuve de créativité et d'ingéniosité pour proposer un modèle original. En particulier, il s'agit de définir au minimum les champs (colonnes) permettant de représenter l'ensemble des informations évoquées dans le cahier des charges et dans les questions des différentes parties. Vous êtes libres d'ajouter toutes les informations supplémentaires vous paraissant intéressantes.

Attention : Vous devez absolument lire intégralement le sujet car les questions des parties suivantes (en particulier requêtes, triggers et procédures stockées) contiennent des informations complémentaires sur les fonctionnalités attendues.

Travail à faire

- Définir un schéma relationnel comportant au minimum sept tables et répondant aux besoins décrits ci-dessus (et dans les parties suivantes) et créer le script SQL de création de la base de données sous PostGres.

- Utiliser une IA générative comme outil de soutien lors de la conception, mais évaluer de manière critique les suggestions fournies.

Rendus

- Script SQL PostgreSQL d'implémentation du schéma choisi. Ce script devra :
  - Créer les tables avec les types de données pertinents.
  - Définir les clés primaires et étrangères.
  - Définir des contraintes "CHECK" là où elles sont pertinentes.
  - Définir toutes les contraintes nécessaires pour garantir l'intégrité des données.
- Documentation de la méthodologie suivie pour l'utilisation des outils d'IA génératives pour créer la base. Vous préciserez en particulier votre interprétation du cahier des charges en expliquant vos choix au niveau des fonctionnalités prises en compte.

Partie 2 - Peuplement de la base

L'objectif de cette partie est de générer partiellement (ou entièrement) automatiquement des données fictives réalistes et de taille importante afin d'avoir une base de données représentative pour mettre en œuvre les techniques d'optimisation.

Python Faker propose un moyen efficace de peupler la base de données avec des données fictives réalistes. Cependant, sa véritable complexité et valeur ajoutée résident dans la gestion des tables comportant des relations, notamment celles avec des clés étrangères. La génération de données pour de telles tables nécessite une attention particulière pour assurer l'intégrité référentielle. Il s'agit en particulier de s'assurer que les données générées pour les tables enfant respectent les contraintes imposées par les clés étrangères des tables parentes.

Travail à faire

- Utiliser Python avec la bibliothèque Faker (cf. Documentation annexe 1) et un outil d'IA générative pour générer des données fictives adaptées à la structure de la base. Votre script Python doit permettre la création de fichiers SQL prêts à être exécutés dans PostGres.
- **Contraintes**
  - Générer au minimum 200 chercheurs, 100 contrats de recherche, plus de 1000 descriptifs de jeux de données et plus de 500 publications. Le nombre de laboratoires pourra être plus réduit (5) ainsi que le nombre de projets structurants (8 à l'université de corse).
  - Assurez-vous que vos scripts de peuplement des tables dynamiques (avec clés étrangères) respectent l'intégrité référentielle
  - Assurez-vous de la cohérence des données générées : chaque chercheur doit être affilié à un labo, chaque projet doit au moins avoir un jeu de données, les dates de projets et de contrats doivent être compatibles, ect…
- **Rendus**

  - Scripts Python utilisés pour générer les données des différentes tables.
  - Scripts SQL de peuplement

Partie 3 - Gestion des Utilisateurs et Création de Vues

Cette partie vise à assurer la sécurité des données et faciliter l'accès pour différents utilisateurs.

Travail à faire

- Créer au minimum trois rôles distincts pour la base de données :
  - Chercheur : accès restreint à ses projets et données.
  - Data Manager : accès élargi aux métadonnées.
  - Administrateur : accès complet.
- Concevoir au moins **cinq vues** pour répondre aux besoins des différents utilisateurs.
- Attribuer les privilèges nécessaires aux différents rôles en fonction des vues créées.

Rendus

- Scripts SQL commentés de Création des utilisateurs, des vues et de l'attribution des privilèges

Partie 4 - Requêtes et optimisation

Travail à faire

On considère les trois requêtes suivantes :

- **R1** — Nombre total de jeux de données déposés par projet et par année civile, ainsi que la moyenne du délai de dépôt (en jours) = date_depot – date_creation des datasets, pour les projets ayant impliqué > 5 chercheurs (tous rôles confondus) depuis 2018.
- **R2** — Pour le laboratoire LISA (vous pouvez choisir un autre labo selon vos données) en 2024, donner la liste des projets (intitulé) et le nom du responsable qui n'ont aucun chercheur dont le nombre de publications en 2024 est strictement inférieur à la moyenne du nombre de publications par chercheur de ce même laboratoire en 2024.
- **R3** — Laboratoires n'ayant eu aucun jeu de données non conforme en 2024, où "non conforme" = jeu de données sans licence ou sans date_depot (métadonnées cœur manquantes).

Pour chacune des requêtes :

1. Proposez au moins deux écritures possibles
2. Comparez les performances des deux solutions à l'aide des outils de postGreSQL, notamment les plans d'exécution et les temps d'exécution réels. Choisissez la meilleure version en fonction des résultats obtenus.
3. Définissez éventuellement des index pertinents pour optimiser les performances de la solution choisie

Rendus

- Scripts SQL de création des différentes versions de chaque requete

- Rapport synthétique expliquant les résultats obtenus lors de la démarche d'optimisation et la justification des éventuels index créés

Partie 5 - Triggers et Procédures Stockées

Travail à faire

Question 1 - Définissez les deux triggers d'insertion/modification suivants :

1. **Limite de participants à un projet**
   Définir un trigger qui, lors de l'insertion d'une nouvelle participation d'un chercheur à un projet, vérifie que la capacité maximale de participants fixée pour ce projet n'est pas dépassée.
2. **Vérification DMP d'un jeu de données**
   Définir un trigger qui empêche le passage d'un jeu de données (métadonnées) en statut « déposé » si le contrat de financement associé n'a pas un DMP validé.

Question 2 - Définissez et testez quatre fonctions/procédures stockées pour répondre aux besoins opérationnels suivants :

1. **Nombre de publications d'un projet**
   Écrire une fonction qui calcule et renvoie le nombre de publications pour un projet donné et une année donnée.
2. **Préparation du bilan d'un projet**
   Écrire une procédure qui, pour une année donnée, met à jour dans une table de bilan le nombre de publications et de jeux de données déposés pour chaque projet.
3. **Fiche projet**
   Écrire une fonction qui, pour un projet donné, renvoie la liste des publications et jeux de données associés avec leurs principales informations (année, DOI ou statut, etc.).
4. **Archivage des contrats échus**
   Écrire une procédure qui déplace les contrats dont la date de fin est antérieure à une date seuil dans une table d'archives dédiée.

Prenez soin de documenter votre code en particulier au niveau des variables et paramètres utilisés.

Rendus

- Scripts de définition des procédures et fonctions
- Scripts de définition des triggers
- Scripts de test des procédures et fonctions réalisées

ANNEXE - Guide de Démarrage Rapide : Utilisation de Python Faker pour Générer des Données Fictives sous Forme d'un Fichier SQL

1. Prérequis

- Python 3.6 ou supérieur
- Installation de Faker: pip install Faker

2. Exemple Simple : Table "Personne"

Supposons que nous ayons une table Personne définie comme suit:

```
CREATE TABLE Personne (
   ID int PRIMARY KEY,
   Nom varchar2(50),
   Prenom varchar2(50),
   Email varchar2(100),
   DateNaissance date
);
```

3. Génération de Données avec Python Faker

Exemple de script :

```
from faker import Faker
import random

fake = Faker()

number_of_entries = 100

with open("data.sql", "w") as file:
   for _ in range(number_of_entries):
      id = _
      nom = fake.last_name()
      prenom = fake.first_name()
      email = fake.email()
      date_naissance = fake.date_of_birth(tzinfo=None, minimum_age=18, maximum_age=90)

      insert_line = f"INSERT INTO Personne (ID, Nom, Prenom, Email, DateNaissance) VALUES ({id}, '{nom}', '{prenom}', '{email}', '{date_naissance}');\n"

      file.write(insert_line)
```

3. Exécution du Script

Lorsque vous exécutez le script ci-dessus, un fichier data.sql sera généré dans le répertoire actuel.