	Université de Corse - Pasquale PAOLI	
	***Diplôme :* M1 DE et DFS**	**2024-2025**
	EC : Bases de données et optimisation **Projet BD Partie 1** Conception assistée et Optimisation BD Données Formations - Etudiants **Enseignant : Evelyne VITTORI**	

On souhaite créer une base de données pour gérer les différentes formations d'informatique d'une université (ex : université de corse). La base de données doit stocker les informations sur les différentes formations proposées (licences et master) et les résultats des étudiants.

Cette base de données est destinée à être associée à une application dont les utilisateurs seront les directeurs des études qui pourront y effectuer des mises à jour sur les maquettes de formation et les notes des étudiants, les enseignants qui pourront effectuer des recherches sur l'ensemble de la base de donnée, et les étudiants qui ne pourront réaliser des recherches qu'au niveau des maquettes de formations mais n'auront pas accès aux résultats.

Le travail que vous devez réaliser consiste à concevoir et implémenter la base de données sous PostgreSQL, y insérer des données à partir de jeux de données en partie fictifs générés aléatoirement selon différentes contraintes. Vous mettrez ensuite en pratique les notions abordées dans le cadre du cours : rôles et privilèges, requêtes et optimisation, triggers et procédures stockées.

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 outil d'IA générative 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 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 et le respect des contraintes spécifiques aux données manipulées (règles métiers).
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-PGSQL :** Acquérir des compétences en PL-PGSQL en mettant en place des triggers et en élaborant des procédures stockées pour des opérations courantes et complexes.

Rendus

Date rendu : lundi 11 novembre

Ce travail doit être réalisé en groupes de 2 à 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 d'une IA générative (préciser l'outil utilisé) 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 d'une IA générative (type chatgpt)

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 chatgpt et Python Faker
   - Qualité de la documentation fournie, incluant les commentaires dans le code
4. **Qualité de la soutenance orale**

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 maquettes de formations et des résultats d'étudiants dans les filières informatique de l'université de corse (L3, M1, M2).

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'une IA générative (ex : chatGPT) 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 ce type d'outil 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 les informations concernant des formations de niveau Licence et Master.
- Chaque année de formation (ex : L1, L2, …) accueille un nombre maximum d'étudiant défini par le directeur des études.
- Chaque diplôme est caractérisé par un niveau (licence ou master) et comporte pour chaque année de formation une liste d'unités d'enseignement UE par semestre. Chaque UE est divisée en éléments constitutifs (de 1 à 20).
- Les éléments constitutifs sont caractérisés par un nombre d'heures d'enseignement divisés en nombre d'heures de cours, nombre d'heures de TD et nombre d'heures de TP.
- Le règlement des études défini la nature des évaluations dans chaque élément constitutif et les modalités de calcul des moyennes par semestre pour chaque année de formation en attribuant un coefficient à chaque UE. La note d'une UE est obtenue par la moyenne des notes de ses éléments constitutifs.
- Le directeur des études saisi les maquettes des formations en définissant pour chaque diplome et pour chaque année de formation, la liste des UE, la liste des éléments constitutifs et leur description. Il attribue un enseignant responsable à chaque élément constitutif.
- Les étudiants pourront consulter toutes les informations relatives aux maquettes mais ils n'auront pas accès aux résultats.
- Chaque semestre, les enseignants responsables des éléments constitutifs saisiront la note obtenue par chaque étudiant dans chaque élément constitutif. A

titre de simplification, une seule note sera saisie par étudiant pour un élément constitutif.
- A la fin de chaque semestre, le directeur des études utilisera la base de données pour préparer les délibérations de chaque année de formation en calculant pour chaque étudiant inscrit sa moyenne générale du semestre. La moyenne sera calculée selon les coefficients attribués aux UE dans les années de formation. On suppose que toutes les UEs se compensent entre elles.
- A la fin du semestre 2, le directeur des études calcule aussi la moyenne des notes des 2 semestres pour chaque étudiant et définit ainsi la réussite ou l'ajournement de l'étudiant à l'année de formation considérée. En cas de réussite, il identifie la mention obtenue.

Votre base de données devra permettre au minimum de représenter l'ensemble des informations évoquées dans le cahier des charges et dans les questions des différentes parties mais 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 répondant aux besoins décrits ci-dessus (et dans les parties suivantes) et créer le script SQL de création de la base de données sous postGreSQL.
- Utiliser une IA générative comme outil de soutien lors de la conception, mais évaluer de manière critique les suggestions fournies.
- Vous pouvez créer un MCD pour vous aider dans votre démarche de conception mais ce n'est pas obligatoire et le MCD ne vous sera pas demandé dans les rendus.

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 d'une IA générative 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 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 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 une 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 postGreSQL.
- Vous n'êtes pas obligés de générer des données intégralement fictives, vous pouvez tout à fait définir une base réaliste en vous basant sur les diplomes de notre université et utiliser python faker pour compléter.
- **Contraintes**
    - Générer au minimum les données relatives aux années de formation de 2005 à 2024 avec au moins 1000 étudiants fictifs.
    - Assurez-vous que vos scripts de peuplement des tables dynamiques (avec clés étrangères) respectent l'intégrité référentielle
    - Le nombre d'inscrits à chaque année de formation devra être compris entre 2 et le nombre d'inscrit maximum défini dans la description de l'année de formation.
    - Un étudiant devra avoir une inscription minimum mais il ne pourra pas avoir plus de deux inscriptions pour la même année de diplôme.
    - Les dates d'inscriptions doivent être cohérentes : pour un même diplôme, les dates d'inscriptions doivent se suivre (ex : L1,L2, L3 avec éventuellement des redoublements mais pas L2 puis L1 ou L3 puis L2 par exemple).
    - Tous les diplomes de niveau licence doivent avoir trois années et les diplômes de niveau master deux.

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 moins trois rôles distincts pour la base de données : étudiant, enseignant et directeur des études. Seul le directeur des études pourra modifier les formations. Les enseignants pourront mettre à jour les notes des étudiants. Les étudiants ne pourront que faire des recherches sur les formations et déposer des avis.

- 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 d'étudiants par année de formation, par année et moyenne des notes finales (note de l'année), pour les formations ayant accueilli plus de 5 étudiants depuis 2010.
R2 - Pour l'année de formation 1 du diplôme de licence Sciences et Techniques (*vous pouvez choisir un autre diplôme selon vos données*) en 2023-2024 donner l'Intitulé des matières (éléments constitutif) et le nom de l'enseignant responsable qui n'ont pas eu d'étudiant ayant obtenu une note en dessous de la moyenne générale des notes finales des étudiants (moyenne de l'année pour l'année de formation considérée).
R3 - Filières n'ayant pas eu d'étudiants qui ont obtenu une note (moyenne générale) inférieure à 4 au cours de l'année 2023-2024.

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. Trigger pour vérifier la limite d'étudiants inscrits dans une année de formation lors de l'insertion d'une inscription.
2. Trigger pour empêcher l'inscription d'un étudiant à une année de formation si l'étudiant n'a pas validé l'année précédente.

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

1. **Calcul de la Moyenne d'un étudiant** : Une fonction qui calcule et renvoie la moyenne générale d'un étudiant inscrit pour une année universitaire, un semestre et une année de formation donnée.

2. **Préparation délibération** : Pour une année de formation donnée, une procédure qui met à jour le résultat de chaque étudiant inscrit (validé ou ajourné) après calcul de ses moyennes au semestre 1 et au semestre 2.
3. **Fiche de notes** : Une procédure qui renvoie pour un étudiant donné, une année de formation et un semestre, la liste de ses notes aux différentes UE.
4. **Archivage des Résultats Anciens** : Une procédure qui, lorsqu'elle est exécutée, déplace les résultats de plus d'un an de la table principale vers une table d'archive Résultats_Archives.

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.