Collège de Maisonneuve
Formation continue

Professeur: Ahmed Imed Eddine Rabah
Mail: arabah@cmaisonneuve.qc.ca
Cours: Base De Donnee - Laboratoire

Projet 2 : Mini Doctolib

Consigne :

- **Respectez strictement les échéances fixées pour chaque étape du projet. Aucun retard ne sera accepté sans justification.**
- **Rapport sous format PDF qui contient Explication détaillée de toutes les étapes.**
- **un seul fichier SQL qui contient la définition du schéma de la base donné normalisé.**
- **Dossier lab1/ contenant le fichier PDF (E/A, relationnel, algèbre...)**
- **Dossier lab2/ avec le fichier .sql**

Cahier de charge client :

Je travaille dans une petite clinique privée et j'aimerais mettre en place un système de gestion des rendez-vous médicaux. Ce n'est pas un gros projet comme Doctolib, mais j'aimerais quelque chose de fonctionnel, adapté à nos besoins quotidiens. Je ne m'y connais pas en base de données, donc je vous explique comment on fonctionne, et à vous de structurer ça comme il faut.

Dans notre clinique, nous avons plusieurs médecins qui exercent dans des spécialités différentes (comme dermatologie, pédiatrie, cardiologie, etc.). Chaque médecin a un nom, un prénom, une adresse email, une spécialité et consulte dans une ou plusieurs villes (certains font des consultations mobiles dans d'autres cliniques).

Des patients peuvent prendre des rendez-vous avec les médecins. Pour cela, ils remplissent un formulaire avec leur nom, prénom, adresse email, numéro de téléphone, ville, et parfois une remarque à propos du rendez-vous (douleur, suivi, etc.). Les patients n'ont pas besoin de créer un compte, ils réservent directement.

Un rendez-vous correspond à une plage horaire précise pour un médecin donné, à une date précise. Chaque médecin a un planning : certaines plages sont disponibles, d'autres sont bloquées. On aimerait savoir si un créneau est disponible ou non. Pour chaque rendez-vous, on doit savoir qui est le patient, quel est le médecin, la date, l'heure, la spécialité, et s'il a été confirmé.

Il arrive aussi que les patients annulent ou modifient un rendez-vous. Dans ce cas, on garde la trace de l'annulation avec la date à laquelle ça a été fait, et la raison si elle est fournie.

On aimerait aussi pouvoir envoyer des notifications ou rappels par email (ou sms fictifs) aux patients avant leur rendez-vous. Ces notifications ne sont pas obligatoires, mais on garde un historique : contenu, date d'envoi, et à qui elles ont été envoyées.

Enfin, de temps en temps, les médecins écrivent un compte rendu après la consultation (bilan, recommandations, etc.). Ce compte rendu est lié à un rendez-vous, mais il n'est pas toujours rempli.

Voilà, j'espère que ça vous donne une bonne idée de notre fonctionnement. On voudrait que vous organisiez tout ça de manière claire et logique dans une base de données. Merci de bien penser aux liens entre les informations, à la possibilité de retrouver rapidement les rendez-vous à venir, les disponibilités des médecins, et les historiques patients.

On aimerait que vous soyez rigoureux dans l'organisation des données, qu'il n'y ait pas de doublons, et que vous puissiez facilement retrouver des informations comme :

1.) Rechercher les rendez-vous confirmés à venir pour un médecin donné.

2.) Lister les médecins avec le nombre total de rendez-vous confirmés qu'ils ont effectués

3.) Trouver les créneaux disponibles pour un médecin donné à une date donnée

4.) Lister tous les rendez-vous annulés avec la date et la raison d'annulation

5.) Retrouver tous les patients qui ont pris au moins 3 rendez-vous avec le même médecin

6.) Historique des notifications envoyées à un patient spécifique

7.) Liste des rendez-vous sans compte rendu médical

8.) Les patients ayant annulé plus de 2 fois

9.) Moyenne de rendez-vous par médecin par semaine

10.) Liste des médecins qui n'ont aucun rendez-vous cette semaine

Bonne chance,
Le responsable administratif.

Travail a faire

- **Lab 1** – Modélisation et requête.
- **Lab 2** : Normalisation et Implémentation avec SQL

Ce projet vous permet de **réaliser un seul projet évolutif** qui sera évalué en deux étapes.

Organisation du projet

Partie	Contenu principal	Pondération	Échéance
Lab 1	Modélisation et Requête	20%	21 avril 2025
Lab 2	Normalisation et SQL	20%	05 mai 2025

Remarque : Le projet final est cumulatif. Il intègre les composantes de Lab 1 et Lab 2 tout en les améliorant.

Barèmes d'évaluation

Lab 1 – Modélisation (20%)

Critère	Points
Schéma E/A correct et complet	/6
Modèle relationnel cohérent	/5
Requêtes en algèbre relationnelle	/5
Arbres algébriques clairs	/4
Total	**/20**

Lab 2 – Normalisation et SQL (20%)

Critère	Points
Tables correctement normalisées (jusqu'à la forme FNBC)	/7
Création correcte des tables (DDL)	/6
Cohérence et réalisme des données insérées	/3
Requêtes fonctionnelles	/4
Total	**/20**

Durée : 30 jours

- **Semaine 1 à 2** : Modélisation et conception de votre base de données à partir du cahier des charges. Vous devez produire :
  - un modèle Entités/Associations clair, cohérent et complet ;
  - sa traduction en modèle relationnel, en veillant à intégrer des **contraintes d'intégrité logique et intuitive** (unicité, non-nullité, relations exclusives, etc.) ;
  - un ensemble de requêtes formulées en algèbre relationnelle ;
  - les arbres algébriques correspondants à ces requêtes.
- **Semaine 3 à 4** : Implémentation SQL de votre modèle conçu au Lab 1. Vous devez :

- créer toutes les tables (DDL) avec clés primaires et étrangères ;
- définir des **contraintes d'intégrité explicites et pertinentes** (NOT NULL, UNIQUE, CHECK, etc.) dans votre script SQL ;
- insérer des jeux de données réalistes et variés ;
- écrire des requêtes simples et complexes pour extraire des informations pertinentes ;
- vous assurer que toutes les tables sont normalisées jusqu'à la **forme normale de Boyce-Codd (FNBC)**.

Bonus général — Connecter votre projet Web à la base de données PostgreSQL

Objectif

Ce bonus vous permet de récupérer les données de votre base SQL (créée dans Lab 2) et de les afficher dynamiquement dans votre interface web (HTML/JS), sans utiliser de framework complexe.

Cette partie est facultative mais vous permettra de découvrir les bases du développement full-stack (front + back). Il peut ajouter jusqu'à 2 points bonus à votre Examen final pour chaque membre du groupe si bien présenté.

Feuille de route pour débutants

Prérequis :

- Node.js installé sur votre machine : https://nodejs.org
- PostgreSQL déjà installé et accessible en local

ÉTAPE 1 — Préparer votre base de données

1.) Lancez PostgreSQL

2.) Assurez-vous que votre base (ex. unisphere) est bien créée

3.) Elle doit contenir des données valides (cours, étudiants, notes, etc.)

ÉTAPE 2 — Créer un petit serveur pour interagir avec PostgreSQL

1.) Créez un nouveau dossier backend

```
mkdir backend
cd backend
npm init -y
npm install express pg cors
```

2.) Créez un fichier db.js

```
const { Pool } = require('pg');

const pool = new Pool({
  user: 'postgres',
```

```
  host: 'localhost',
  database: 'nom_de_votre_base',
  password: 'votre_mot_de_passe',
  port: 5432,
});

module.exports = pool;
```

3.) Créez un fichier server.js

```
const express = require('express');
const pool = require('./db');
const cors = require('cors');
const app = express();
app.use(cors());
// Exemple : récupérer tous les cours
app.get('/cours', async (req, res) => {
  try {
    const result = await pool.query('SELECT * FROM medecin');
    res.json(result.rows);
  } catch (err) {
    res.status(500).send('Erreur serveur');
  }
});
app.listen(3000, () => {
  console.log('Serveur en ligne sur http://localhost:3000');
});
```

ÉTAPE 3 — Démarrer votre serveur

```
node server.js
```

Vous devriez voir :

```
Serveur en ligne sur http://localhost:3000
```

ÉTAPE 4 — Récupérer les données côté HTML

Dans votre projet web (ex : UniSphere, TaskMaster...), ajoutez ce code dans un fichier js/app.js :

- app.js

```
fetch('http://localhost:3000/cours')
  .then(response => response.json())
  .then(data => {
    const container = document.getElementById('liste-cours');
    data.forEach(cours => {
      const div = document.createElement('div');
      div.className = 'card p-3 mb-2';
      div.innerHTML = `
        <h5>${cours.titre}</h5>
```

```
        <p>Crédits : ${cours.credits}</p>
      `;
      container.appendChild(div);
    });
  });
```

- Dans dashboard.html

```
<div class="container mt-5">
  <h2>Liste des cours</h2>
  <div id="liste-cours"></div>
</div>
<script src="js/app.js"></script>
```

Documentation et librairie d'aide

- ChatGPT : Utilisez-le intelligemment.
- W3School : Pour toute les fonctionnalités de PostgreSQL.

Outils collaboration et travail en groupe :

- **Gestion de temps et de tache**: Trello

Message de soutien

- Si vous ressentez une baisse de motivation, rencontrez des difficultés ou faites face à des imprévus, sachez que c'est tout à fait normal dans le cadre d'un projet qui représente 40% de la moyenne du cours. Travaillez en équipe, communiquez avec vos camarades, organisez votre temps efficacement et répartissez les tâches de manière équilibrée selon vos compétences et disponibilités.
- En cas de conflit ou de situation délicate, je reste disponible et à votre écoute. N'hésitez pas à me contacter si vous avez besoin de conseils ou d'assistance.

Bon courage à tous !