Collections interconnectées dans une chaîne de bibliothèques

1- Reformulation du projet

Un client voudrait informatiser sa bibliothèque. Cette informatisation passe par la création d'une base de données pour gérer les différentes bibliothèques de son réseau dans le but de centraliser toutes les données et facilement gérer son personnel mais aussi les abonnés et leurs actions (prêts, réservations, pénalités…).
Notre base de donnée doit donc répondre aux critères suivants:

- Avoir un suivi des ouvrages disponibles ainsi que leur disponibilité et leur emplacement.

- Avoir un suivi des prêts effectués dans les bibliothèques et respecter les règles d'emprunts (3 emprunts max, limitation en fonction des abonnements/pénalités…)

- Avoir un suivi des réservations et bloquer à l'emprunt les livres réservés

- Avoir un suivi des pénalités appliquées ou en cours d'application d'une personne enregistrée (Bibliothécaire, abonné(e)...)

- Gérer les différents transferts entre bibliothèques

- Gérer les fin d'abonnements (retirer les réservations de l'abonné, bloquer si le prêt n'est pas terminé)

- Gérer les différents événements organisés dans la bibliothèque, pouvoir enregistrer les intervenants et les participants, abonnés ou pas

2- Schéma de la base de donnée

Voir dossier doc du projet.

3- L'expression en SQL des requêtes

Voir dossier src du projet.

4- Description du jeu de donné

Chaque requête est accompagnée de son jeu de donné dans le dossier test du projet. Chaque test est complet et permet de traiter les cas de base comme les cas limites générant des exceptions.
Pour ce qui est du test global du projet, on utilise le fichier delete_table.sql dans le dossier Table pour assurer la suppression préalable de tables qui auraient pu rester (fichier créé par ChatGPT).
Dans un second temps, l'on doit exécuter le script to_test dans Test. Ce script crée les tables puis les remplit conformément à notre vision des tables de ce projet ensuite, il crée les fonctions et les instancie à leurs homologues (triggers/vues/groupes).
Ensuite chacun des tests indépendants dans Test peuvent être exécutés.*
Il faut noter cependant que certains tests

5- Analyse critique du déroulement du projet

Lors de la première séance, nous avons d'abord reformulé le sujet pour en récupérer ses tenants et aboutissants dans le but de déceler les zones d'ombres à éclaircir avec le client. Nous avons donc rédigé des questions et les avons posées au client. La seconde séance consistait à définir un schéma entité-association basé sur les réponses que nous avions récupérées du client. Nous avons beaucoup travaillé sur cette partie car une bonne base de données nous facilite la suite du projet. Bien que notre base de données était bien pensée en général, il nous a fallu quelques retouches lors du codage de la base de données pour s'assurer de la forme normale (Nous sommes actuellement en 5NF).
Lors de la dernière séance, nous nous sommes vu attribués des dépôts github.
La dernière étape fut de remplir la base de données et traiter les modifications de celles-ci à l'aide d'outils avancés SQL (triggers, vues, sous requêtes, …). Celle-ci fut la plus complexe car certaines requêtes nécessitent un travail de fond et une bonne compréhension de la syntaxe PostgreSQL.

Nous avons rencontré de nombreux problèmes lors de la création des US pour se répartir des tâches. En effet, les consignes n'avaient pas étés bien assimilés et nous nous étions éloignés des consignes pour se concentrer sur les réponses du client et nous nous sommes rendus compte 1 jour avant la deadline que la compréhension globale du projet n'était pas bonne. Nous avons donc dû refaire et modifier certaines lignes de code et modifier une partie de l'architecture de la base de données nullifiant donc nos efforts sur cette partie du projet et nous faisant perdre un temps précieux.
Malgré ces oublis notre base de données reste fonctionnelle et testée. Mais elle sera en effet incomplète par rapport au sujet de base et en retard de quelques heures par rapport à la date de rendu initiale, ce pour quoi l'équipe s'excuse platement.

Pour ce qui est de la répartition des tâches, nous avions découpé les requêtes à faire entre différentes user story comportant des tasks (chaque task était des triggers/vues à réaliser en rapport avec une seule table de la base de donnée). Cette manière de faire bien que pratique car elle nous permettait de ne rien omettre s'est avérée moins bonne au niveau de la répartition des tâches. Certains se sont retrouvés avec peu de code à faire tandis que d'autres en avaient beaucoup. Cela a entraîné un déséquilibre de travail sur cette partie entre les différents membres du groupe.
De plus, le système de branche que nous avions instanciés pour nous faciliter la tâche et éviter des pertes d'infos et des bugs de merges incessants que nous pensions intéressant pour ce projet s'est avéré être plus problématique pour ce projet que nous le pensions. En effet, ce système n'était pas maîtrisé par le groupe et nous avons eu de nombreux problèmes de merges lors du merge final des branches. Le projet a en partie été supprimé à ce moment. C'est aussi une des raisons du retard du groupe dans le rendu.

Finalement, en prenant en compte ce que nous avons effectué nous pensons qu'avec moins de problèmes extérieurs (comme github ou pgadmin 4 qui nous ont posés beaucoup de problèmes comme dit précédemment) et une réflexion à tête reposé sur le sujet, nous aurions pu aller plus loin, plus vite et de manière bien plus optimisée en évitant de diverger du projet.

Cette expérience était cependant très intéressante et nous avons beaucoup appris sur le SQL, la création de base de données, la relation client-développeur ainsi que le fonctionnement des branches sur github.

Cordialement,
HELLAL Mathias, MIGNY Léo, BIZOT Patrick, MESKINI Driss