TP 2 : Projet BDD

(Conception , Sécurité des BDD etSQL : LDD, LMD LCD)

Au départ, votre client, gestionnaire d'une grande bibliothèque moderne, vous demande de mettre en place la base de données pour la gestion de l'ensemble des activités de la bibliothèque. Il souhaite pouvoir enregistrer les livres avec leurs titres, résumés, dates de publication, nombre d'exemplaires, et leur date d'ajout à la collection. De plus, il veut conserver des informations détaillées sur les auteurs, les éditeurs, et les catégories littéraires auxquelles les livres appartiennent. La base devra également gérer les membres de la bibliothèque avec leurs coordonnées et plusieurs adresses (domicile, courrier), ainsi que les emprunts et réservations effectués par ces membres. Enfin, le client souhaite enregistrer les avis laissés sur les livres et suivre l'historique des amendes en cas de retards ou de dommages.

- Vous prenez note de ces besoins et réalisez, en premier lieu, un schéma conceptuel en identifiant les entités essentielles et leurs relations (par exemple : un livre peut avoir plusieurs auteurs, appartenir à plusieurs catégories, et être édité par un éditeur ; un membre peut effectuer plusieurs emprunts ou réservations, etc.).
- Vous rédigez un premier rapport expliquant votre modèle conceptuel et vous le modélisez ensuite à l'aide d'un outil tel que MySQL Workbench pour obtenir un schéma relationnel.
- Vous créez la base de données en écrivant l'ensemble des commandes LDD. Vous utiliserez notamment :
  - La création de tables
  - La création de tables à partir d'une sélection via **CREATE TABLE <nom_table> [(<colonne1>, <colonne2>, …)] AS SELECT …**
  - La vérification des tables existantes avec la requête **SELECT**
  - Inspection de leur structure et **métaschéma**;
  - La définition des contraintes d'intégrité : **clé primaire (PRIMARY KEY), unicité (UNIQUE), non nullité (NOT NULL), clé étrangère (FOREIGN KEY), et contraintes de domaine via CHECK (par exemple, pour vérifier qu'un nombre d'exemplaires soit toujours positif)**
  - L'utilisation d'options comme **ON DELETE CASCADE** pour maintenir l'intégrité référentielle
  - La liste des contraintes existantes avec la requête avec la requête **SELECT**;
  - Le renommage d'objets;
  - L'utilisation **d'ALTER TABLE** pour ajouter, modifier ou supprimer des colonnes, ainsi que pour gérer les contraintes (ajout, suppression, renommage, activation ou désactivation)
  - La suppression des tables avec et sans **CASCADE CONSTRAINTS**;
- Vous remplissez ensuite votre base de données avec un jeu de données conséquent (au moins 30 tuples par table – pour les livres, auteurs, membres, etc. – et le maximum d'associations possibles). Pour cela, vous utiliserez le langage de manipulation des données (LMD) :
  - **INSERT INTO <nom_table> [(liste de colonnes)] VALUES (liste des valeurs);**
  - **INSERT INTO <nom_table> [(liste de colonnes)] SELECT …;**
  - La création de séquences pour générer des identifiants automatiquement.
  - Des mises à jour des données avec **UPDATE** avec et sans **SELECT**;
  - Des suppressions avec DELETE FROM <nom_table> [WHERE condition];

Peu après, le client revient vers vous et explique qu'il souhaite lancer des événements littéraires et des promotions temporaires (par exemple, expositions ou remises sur certains livres). Il veut pouvoir activer ou désactiver ces événements et appliquer des remises ou avantages aux livres concernés.

- Vous notez cette demande et modifiez votre schéma en conséquence, en ajoutant par exemple une nouvelle entité « Promotions/Événements » associée de façon optionnelle aux livres via une clé étrangère nullable ou une table d'association.
- Vous mettrez à jour votre schéma avec des commandes LDD (ajout de nouvelles tables et contraintes, modification via ALTER TABLE, etc.).

Quelques jours plus tard, le client estime que la gestion des adresses des membres est trop complexe. Plutôt que de conserver plusieurs adresses pour chaque membre, il souhaite simplifier en ne gardant qu'une adresse principale, tout en laissant la possibilité d'ajouter des adresses supplémentaires si nécessaire.

- Vous envisagez deux approches : fusionner l'adresse principale dans la fiche membre, ou conserver une table d'adresses avec une indication explicite de l'adresse principale (par exemple, via un attribut booléen ou une contrainte d'unicité).
- Vous discutez des avantages et inconvénients de chaque solution et proposez la modification la plus adaptée, en utilisant les commandes LDD nécessaires (création ou modification de tables et de clés étrangères).

Dans la foulée, le client demande d'enrichir le suivi des emprunts. Il veut connaître non seulement le statut actuel de l'emprunt (par exemple, « En cours », « Retourné », « En retard » ou « Annulé »), mais aussi le mode de réservation (sur place, en ligne, etc.) et la date de retour prévue.

- Vous ajoutez de nouvelles colonnes dans la table des emprunts et ajustez les contraintes existantes (via CHECK, par exemple) pour garantir la cohérence des données.
- Vous testez ces modifications via des opérations de mise à jour (LMD).

Ensuite, pour garantir la qualité des données, le client insiste pour que certains critères soient vérifiés automatiquement. Par exemple, le nombre d'exemplaires d'un livre ne peut jamais être nul ou négatif, et la durée d'emprunt doit respecter des bornes définies.

- Vous ajoutez des contraintes CHECK dans vos définitions de tables pour vérifier ces règles à la création et lors des mises à jour des enregistrements.

Un peu plus tard, soucieux de la sécurité, le client vous informe qu'il ne souhaite plus stocker les mots de passe des membres en clair et veut externaliser l'authentification vers un service spécialisé.

- Vous retirez la colonne de mot de passe de la fiche membre et proposez une solution alternative, comme l'utilisation d'un hash sécurisé ou l'intégration d'un système d'authentification externe. Ces modifications seront effectuées via ALTER TABLE.

Puis, dans un souci d'amélioration du service, le client souhaite suivre les retours de livres (par exemple, les retours anticipés ou les retours avec remarques sur l'état du livre). Il désire connaître la date du retour, le livre concerné, l'emprunt associé, le motif du retour et le statut de traitement (par exemple, « En attente », « Accepté » ou « Refusé »).

- Vous proposez la création d'une nouvelle entité « Retours » et vous l'intégrez dans votre modèle en créant une table dédiée avec les clés appropriées et les contraintes nécessaires.

Parallèlement, le client veut améliorer la traçabilité des emprunts. Il souhaite que chaque modification (par exemple, prolongation de la période d'emprunt ou changement de statut) soit enregistrée avec la date de modification et une description du changement.

- Vous proposez de mettre en place une table d'historique pour les emprunts et, éventuellement, des triggers qui enregistrent automatiquement chaque modification.

Le client se rend compte que les informations relatives aux éditeurs sont trop détaillées. Il souhaite se concentrer sur l'essentiel en supprimant certains attributs (comme le numéro de téléphone) et en mettant l'accent sur l'adresse email et le contact principal.

- Vous adaptez le schéma des éditeurs en modifiant la table correspondante via ALTER TABLE.

Plus tard, le client vous demande d'automatiser l'attribution des promotions aux livres. Il souhaite que certains livres soient automatiquement associés à un événement promotionnel en fonction de critères comme leur ancienneté dans la collection ou le nombre d'exemplaires disponibles.

- Vous envisagez l'utilisation de triggers ou de procédures stockées pour réaliser cette association dynamique.

Le client souhaite ensuite obtenir une vue synthétique des activités de la bibliothèque, afin d'aider le service de gestion. Il vous demande de créer un tableau de bord virtuel qui agrège, par exemple, le nombre d'emprunts mensuel, les retours, et les amendes par catégorie de livres.

- Vous créez une vue SQL (avec CREATE VIEW) qui rassemble ces informations à l'aide de fonctions d'agrégation, des jointures et des regroupements (GROUP BY), avec un tri adapté (ORDER BY).

Au fil de l'évolution, le client décide de revoir la gestion des catégories littéraires. Il souhaite que, chaque fois qu'un livre est associé à une catégorie, la date de cette association soit enregistrée pour suivre l'historique des modifications.

- Vous modifiez la table d'association entre livres et catégories pour y ajouter un attribut de date et ajustez la logique d'insertion via des requêtes LMD.

Dans un autre changement, le client trouve que le suivi des réservations est trop détaillé et demande de simplifier certaines informations. Vous devez alors revoir la structure de la gestion des réservations et fusionner certains attributs pour alléger le modèle sans perdre d'information pertinente.

- Vous réorganisez la table ou les tables concernées à l'aide de commandes d'ALTER TABLE et ajustez les requêtes d'insertion/mise à jour.

À un moment donné, le client souhaite diversifier les modes d'emprunt (prêt sur place, prêt à domicile, etc.) et qu'ils soient clairement identifiés dans chaque emprunt.

- Vous ajoutez une colonne dédiée aux modes d'emprunt dans la table correspondante, en garantissant la validité des données via des contraintes CHECK ou des tables de référence.

Le client revient ensuite sur les avis laissés par les membres : il souhaite que chaque avis comporte désormais un titre et un indicateur de validation par un bibliothécaire.

- Vous ajustez la structure de la table des avis en ajoutant ces nouvelles colonnes et en adaptant les requêtes LMD pour intégrer ces informations.

Toujours soucieux de la traçabilité, le client demande la mise en place d'un historique détaillé des modifications apportées aux livres (changement de prix, d'état, de nombre d'exemplaires, etc.).

- Vous proposez d'implémenter une solution d'historisation en créant une table dédiée et en automatisant l'enregistrement des modifications via des triggers.

Ensuite, le client souhaite modifier le format de stockage des dates pour harmoniser l'ensemble du système avec d'autres applications de gestion.

- Vous définissez une stratégie pour adapter le format des dates dans toutes les tables (en modifiant les types de données via ALTER TABLE) tout en veillant à la migration des données existantes.

Le client décide alors de fusionner certaines informations relatives aux amendes. Il estime que la gestion des amendes (liées aux retards ou aux dommages) est trop isolée et souhaiterait intégrer directement ces informations dans le suivi des emprunts.

- Vous étudiez les avantages et inconvénients d'une telle fusion et proposez une solution cohérente en modifiant la structure de la table des emprunts et en créant des jointures dans les vues.

Enfin, le client souhaite revoir la précision des données financières, comme les montants des amendes, pour qu'ils soient conformes aux standards comptables.

- Vous adaptez la définition des colonnes financières en modifiant leur type de données (par exemple, en passant à DECIMAL ou NUMERIC avec la précision appropriée).

Pour clore le projet, le client insiste sur le fait qu'il faut s'assurer que toutes ces évolutions n'ont pas compromis la cohérence globale du système. Vous devez donc mettre en place des mécanismes de validation automatisée (via des triggers ou procédures stockées) pour vérifier que, par exemple, la somme des amendes ou la cohérence des emprunts reste toujours valide.

En parallèle de ces évolutions, vous devez également gérer la manipulation des données. Cela signifie que, une fois la structure en place, vous devez insérer un jeu de données conséquent (par exemple, une dizaine de membres, une vingtaine de livres, quelques éditeurs et auteurs, plusieurs emprunts et réservations, ainsi que des avis et retours). Vous devez aussi réaliser des opérations de mise à jour (pour modifier les adresses, les statuts d'emprunts ou les montants des amendes) et des suppressions (pour retirer des livres obsolètes ou des avis inappropriés). Ces opérations représentent la partie LMD et vous permettront de tester et démontrer que votre base de données fonctionne correctement au quotidien.
Par ailleurs, le client vous demande de répondre à une vingtaine de questions d'intégration. Par exemple, vous devrez rédiger des requêtes pour :

- Lister les emprunts d'un membre avec tous les détails associés (livres, dates, modes d'emprunt, etc.).
- Calculer le nombre d'emprunts mensuel et les amendes générées.
- Afficher le top 5 des livres les mieux notés.
- Identifier les membres fidèles (ceux ayant emprunté plus de 10 livres et ayant peu ou pas d'amendes).
- Lister les livres en rupture d'exemplaires.
- Extraire les réservations en cours.
- Afficher les retours de livres avec leur motif et leur statut.
- Lister les livres en promotion et calculer leur nouveau tarif éventuel.
- Agréger les emprunts par catégorie littéraire.

- Afficher l'historique des modifications d'un emprunt ou d'un livre.
- Identifier les livres non empruntés depuis plus de six mois.
- Lister les emprunts avec des amendes en attente ou contestées.
- Calculer le délai moyen entre la date d'emprunt et la date de retour.
- Comparer les emprunts générés par chaque éditeur.
- Afficher le détail complet d'un emprunt donné.
- Calculer la note moyenne et le nombre d'avis pour chaque livre.
- Extraire l'évolution du statut d'un emprunt sur une période donnée.
- Lister les livres ajoutés récemment.
- Comparer le nombre d'emprunts selon le mode (sur place, à domicile, etc.).
- Créer une vue globale qui synthétise les activités de la bibliothèque par éditeur ou par catégorie.

Pour chacune de ces requêtes, vous devez :

- Expliquer la logique de formulation.
- Tester leur exécution.
- Utiliser le maximum d'options SQL disponibles (jointures, sous-requêtes, agrégations, opérations ensemblistes telles que UNION, INTERSECT, MINUS, tri avec ORDER BY, etc.).

L'objectif est que l'étudiant soit capable de traduire une description en langage naturel en une série complète de commandes SQL de définition (LDD), de manipulation (LMD) et d'interrogation (LID).