**TP (PL/SQL)**

**Application pour…….**

Durant la session 1, vous avez appris à créer des BD relationnelles et interagir avec ces BDs en utilisant SQL directement sur les tables, pour différents besoins.

Nous avons déjà expliqué, qu’il est recommandé d’accèder aux tables en utilisant :

* **Des vues** : encapsulation de la complexité SQL, sécurité, différentes vues des mêmes données,…
* **Des procédures stockées** : regroupement logique des traitements, réduction du nombre de va et vient sur le réseau,….

Dans ce travail, vous allez devoir créer un package qui va servir d’une API (Application Programming Interface) pour les diférentes tables de votre BD.

SQL Developer vous permet de générer le squelete des procédures stockées pour les opérations CRUD de base.

|  |
| --- |
|  |
|  |
|  |

Mais il y a **certainement** d’autres traitements, en plus d’ajouter, supprimer, modifier, qui sont PERTINENTS pour votre BD.

Vous devez donc créer une API qui va servir d’interface pour différents traitements sur la BD, en complétant la liste des procédures stockées par d’autres correspondant à des besoins liés à votre BD (**AU MOINS 4 de plus**).

**Consignes générales.**

* Le texte précédent vous donne un aperçu des traitements « essentiels » à prendre en charge. Mais rien ne vous empêche de compléter la liste avec d’autres procédures et fonctions que vous jugerez utiles.
* Au moins 2 des traitements DOIVENT impliquer PLUSIEURS tables avec une logique métier complexe (qui dépasse le niveau exécuter une requête et confirmer ou retourner des résultats).
* Vous pouvez aussi créer un (ou plusieurs) packages (sans corps éventuellement) pour regrouper les déclarations de constantes utiles à l’application ainsi que des exceptions.
* Les procédures et fonctions doivent utiliser les mécanismes de gestion d’exception lorsque nécessaire.
* Aucune utilisation des variables de substitution et de l’affichage à l’aide de DBMS\_OUTPUT dans les procédures et les fonctions.
* Vous devrez tester vos procédures et fonctions à l’aide de blocs anonymes.
* Le code doit être documenté (commentaires). Prévoir au moins une section au début du package et de chaque procédure (ou fonction).
* Il faut intégrer le maximum des connaissances vues en cours (curseurs, collections, traitements de masse,…)

OPTIONS POUR LE CHOIX DU TP

* **Pour la réalisation du TP, vous avez plusieurs options. Choisir une parmi les suivantes :**

**Option\_1**

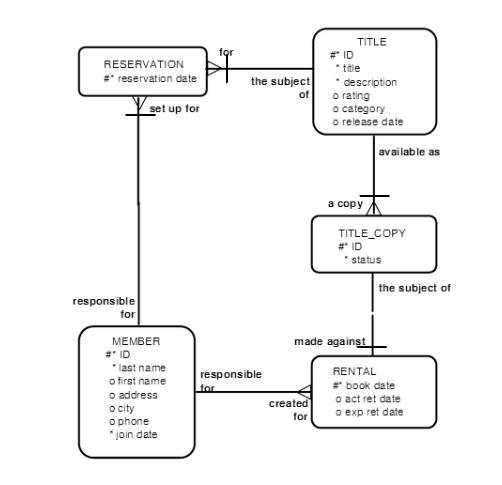
1. Créer une API, pour une BD pour un besoin que vous allez déterminer (choix de sujet dans la liste du document « dd » ou votre propre sujet). Le travail va se faire en 2 étapes :
   1. Modélisation et création de la BD
   2. Création de l’API avec différentes procédures et fonctions qui répondent aux besoins d’applications typiques qui interogent votre BD.

**Option\_2**

1. Créer une API pour une BD existante. Vous n’avez donc pas à modéliser et créer la BD.
2. La BD peut être :
   1. Celle utilisée en première session (produits/ventes….)
   2. Celle présentée sur la page suivante (prêts de titres)
   3. Autre à me proposer.

**Application de gestion de prets de Livres?, de films?….**

On dispose d’un schéma physique comme suit :



Les scripts pour créer le schéma sont avec les notes de cours (**creSchema**.sql et **Backup\_Données\_Tables**.sql)

Vous allez créer un package qui sera utilisé par l’application. Cette application permet aux clients :

* de devenir membre.
* Tous les membres peuvent :
  + emprunter un titre,
  + retourner des titres,
  + et réserver des titres.

Créer un package avec les procédures et les fonctions suivantes:

**NEW\_MEMBER**

Une procédure publique qui ajoute un nouveau membre à la table membres :

* Pour le member id, utilisez une séquence (MEMBER\_ID\_SEQ),
* pour la date de membership, utilisez SYSDATE.
* Passer toutes les autres valeurs ​​à être insérées dans une nouvelle ligne en tant que paramètres.

**NEW\_RENTAL**

Une fonction publique surchargée pour enregistrer une nouvelle location :

* Passez le title id du titre à emprunter et soit le nom du client ou lemember id, à la fonction.
* La fonction doit renvoyer la date de retour. Cette date est trois jours de la date de prêt.
* Si le statut d'un titre demandé est « AVAILABLE » dans le table TITLE\_COPY pour une copie de ce titre, alors mettre à jour la table TITLE\_COPY à « RENTED »
* S'il n'y a pas de copie disponible, la fonction doit retourner NULL.
* Ensuite, insérez un nouvel enregistrement dans la table RENTAL :
  + avec la date du jour comme date de prêt,
  + le numéro d'identification de la copie,
  + le member id du membre,
  + le numéro d'identification de titre
  + et la date de retour prévue.

**Attention** au cas de clients ayant le même nom de famille. Dans ce cas, par exemple, on peut :

* Faire en sorte que la fonction retourne NULL,
* et afficher une liste des membres ainsi que leur « member ID ».

**RETURN\_TITLE**

Une procédure publique qui met à jour le statut d'un titre dans la table TITLE\_COPY (available, rented, or damaged) et définit la date de retour :

* Transmettre le « title ID », le « copy ID » et le statut de la copie à la procédure.
* Vérifier s'il y a des réservations pour ce titre et afficher un message s’il est réservé.
* Mettre à jour la table RENTAL et définir la date de retour effectif (Actual return date) à la date d'aujourd'hui.
* Mettre à jour le statut dans la table TITLE\_COPY à l’aide du paramètre d'état passé à la procédure.

**RESERVE\_TITLE**

:Une procédure privée qui s’exécute uniquement si toutes les copies demandées dans la procédure NEW\_RENTAL ont un statut de « RENTED » :

* Passez le « member ID» et le «title ID » à cette procédure.
* Insérer un nouvel enregistrement dans la table RESERVATION (date, member ID number, and title ID number)

À la fin d’une réservation, afficher un message indiquant que le titre a été réservé ainsi que sa date de retour « prévue ».

**EXCEPTION\_HANDLER**

Une procédure privée qui sera appelée à partir des procédures publiques.

Cette procédure reçoit en paramètre :

* Le SQLCODE
* Le nom de la procédure (ou fonction) où a eu lieu l’erreur(chaîne de caractères).

(Utiliser aussi RAISE\_APPLICATION\_ERROR).

La procédure doit offrir un traitement pour toute autre erreur non spécifique (OTHERS)

Pour les erreurs à prendre en charge fiez-vous au schéma pour déterminer les erreurs les plus pertinentes à ce type d’application.

Autres traitements possibles :

* Nombre de retards pour un client
* titres les plus loués
* Recherche selon différents critères
* Etc.