Rapport Partie 2 Projet

Base de données 2

Réalisé par :

Houssam EL BOUHI : 22014521

Redha BOUZIDI : 22118033

1. Ce qui est fait et difficultés rencontrées :

Le projet est terminé à 100%, toutes les fonctionnalités demandées sont faites et bien testés et valide.

- Script de création et tests : Fait
- Requêtes : Fait
- Procédures/fonctions (PL/SQL) : Fait
- Contraintes d'intégrités : Fait
- Triggers demandés : Fait
- Index : Fait

Les difficultés rencontrées étaient surtout dans la 5ème requête et 2ème procédure.

2. Choix d'implémentation :

N.B : Le mot-clé « wish » dans le nom de la liste concerne les listes que l'utilisateur prévoit acheter, si ce mot n'existe pas c'est-à-dire qu'il possède les éléments de cette liste.

- 1ère requête :

On réalise un count des utilisateurs au début qui ont des listes dont le nom de la liste ne contient pas un « wish » et on le compare avec le count des types de listes qu'on a dans la base, si ils sont égaux alors on retourne le nom et prénom de l'utilisateur.

- 2ème requête :

On réalise un count des objets dans les listes possédées par les utilisateurs et on regarde si un objet est présent 20 fois ou plus dans les listes, si c'est le cas on calcule la moyenne des évaluations de cet objet qui doit être supérieur à 14, si c'est le cas on retourne son nom, l'implémentation est faite grâce à la table liste_objet qui réunit chaque objet à sa liste correspondante.

- 3ème requête :

On réalise un test de minimalité, si un utilisateur pendant toutes ses évaluations, la plus basse évaluation est supérieure à 8, alors il n'a jamais évalué de moins de 8 et donc on retourne son nom et prénom

- 4ème requête :

On prend au premier lieu les objets qui sont commentés la semaine dernière en utilisant (sysdate – 7) puis on rassemble les objets qui ont le plus nombre de commentaire dans la base à l'aide de la fonction MAX().

- 5ème requête :

On récupère les objets notés au cours de la année dernière en faisant un EXTRACT(year), puis pour vérifier que les mois sont consécutifs aussi avec un EXTRACT(month). Ensuite pour récupérer le nombre des objets qui prévoit acheter et qui possède en fait une différenciation par le mot-clé « wish ». Et à la fin en routourne la longueur de la liste d'objet minimal, et celle de nombre maximal d'objet puis la moyenne des longueurs et joindre l'utilisateur associé à ses données.

- Fonction PL/SQL :

On SELECT le COUNT des objets dont l'identifiant est passé en commentaire, on teste si ce count de cet objet dans les listes est présent au moins 20 fois, on récupère donc son nom et la moyenne de ces évaluations dans la base et on l'affiche, sinon on return 0.

Score_moyen(1) retourne « john wick » de moyenne 15.05

- 1ère Procédure PL/SQL :

Le premier select cherche les identifiants des film / jeu vidéo / livre de l'utilisateur. Puis on passe à une boucle qui retourne pour ces types donnés les objets par un ordre décroissant jusqu'à 10 objets

seulement. Si une liste d'un type ne possède pas au moins 10 éléments ne se génére pas grâce à un count() dans le premier SELECT.

- 2ème Procédure PL/SQL :

suggestion consiste a seulement une boucle for dans un select , le select , prend le tableau location , et effectue une jointure avec un autre select , ce dernier prends les quotation ou l'id user est équivalent a la variable entrée , et effectue une autre jointure , ou on compte le nombre d'utilisateurs commun pour chaque objets , ensuite le premier select effectue une deuxième jointure avec utilisateurs différents de la variable id , et on rajoute la condition ou id_user = id pour le select de base , en plus de ça , on fait un left join sur quotation pour récupérer la moyenne , et on projette tout ça pour seulement récupérer le nom d'objet et le numéro de colonne.

- 1er déclencheur demandé :

Dans le trigger **objects_tr** qui se déclenche après l'insertion, on affecte à une variable l'année et le mois d'insertion avec sysdate, ensuite on cherche l'identifiant de la liste dans la table nouveaute, si on trouve l'identifiant , on fait un insert dans la table nouveaute_objet, sinon, on insère dans la table nouveaute les valeurs t de la date actuelle et ensuite on insère dans la table nouveaute_objet

- 2ème déclencheur demandé :

Ajout en premier lieu de deux tables (archive & ar_liste_objet) pour stocker les listes supprimer (la fonction de la table ar_liste_objet sera comme celle de liste_objet).

Dans cette question nous nous sommes basés sur deux déclencheurs.

Le premier TRIGGER **ARCH** sert juste de stocker les identifiants de la liste et l'owner de la liste et le nom de cette liste.

Le deuxième TRIGGER **ARCH_OBJ**, après suppression d'une liaison liste_objet, on fait un select pour être sûr que sa liste correspondante existe, et on fait un insert tout simple. Si cela n'existe pas, on est obligé de créer sa liste correspondante et on insère après, ce test fait par EXCEPTION when NO_DATA_FOUND

N.B : Les Déclencheurs associés à la base sont décrits et expliquer dans la section suivante.

3. Contraintes d'intégrités :

- Unicité de login en utilisant UNIQUE
- Le premier caractère du login doit être composé du premier caractère du prénom et les 7 suivants des 7 premiers caractères du nom (tout en minuscule) et 2 chiffres à la fin.
- Mot de passe peut contenir que des chiffres ou lettres en majuscules ou minuscules et le caractère « _ » seulement.
- La note donnée par l'utilisateur pour un objet doit être entre 0 et 20.
- Unicité d'objet en utilisant UNIQUE.
- La date de naissance doit être inférieur à la date d'inscription.

Déclencheurs associés à la base :

- TRIGGER **REGISTRATION_CHECK** qui teste que la date d'inscription doit être inférieur à la date actuelle
- TRIGGER **NOTE_DATE_CHECK** qui teste que la date d'insertion de la note est inférieure à la date actuelle.
- TRIGGER **COMMENT_DATE_CHECK** qui teste que la date d'insertion du commentaire est inférieure à la date actuelle.

4. INDEX

Les index sont mises sur les identifiant de les tables UTILISATEUR (id_user), LISTE (id_list), OBJET (id_object) et LISTE_OBJET (id_object,id_list) car ils sont les plus utilisé pour chercher dans la base des données pour chercher plus rapide.
L'identifiant (id_quotation) de la table QUOTATION n'est pas indexé car on l'a jamais l'utilisé et donc il gaspillera juste de la mémoire.

5. Explication des insertions et résultats :

Les insertions sont faites pour tester la validité des requêtes et procédures.

- Pour la 1ère requête auquel on doit afficher les utilisateurs ayant crée une liste pour chacun des types, le mot « wish » dans le nom de la liste inséré met différence entre les listes que l'utilisateur souhaite acheter et ceux dont il possède. Donc l'utilisateur 1 et 5 sont valides qui ont des listes de tous les types sans le mot clé « wish ».
- Comme pour la 2ème requête qui demande un objet appartenant à 20 listes ou plus les insertions sont mises pour l'objet « John Wick » pour tester la requête.
- Comme pour la 4ème requête qui demande les insertions faites concernent l'heure actuel de l'implémentation de la base qui concerne cette semaine, lors de la correction du travail la requête ne retourne aucune donnée mais le code est toujours valide et fonctionne très bien.
- Pour la 5ème requête qui demande les données d'un utilisateur qui a noté des objets sur 3 mois consécutifs au cours de l'année précédente. Les insertions faites pour tester la validité de la requête sont pour l'année 2021, donc l'utilisateur 1 et 2 sont affichés si la correction est faite en 2023 aucune donnée sera affichée.
- Pour la fonction à implémenter, les insertions sont semblables à celles de la 2ème requête, alors elle renvoie « John Wick » et son score moyen.
- Pour la 1ère procédure les insertions sont ajoutées pour être conforme à l'objectif de la question, 10 meilleurs objets de chaque type « livre », « jeu vidéo » et « film » de chaque liste si chaque liste contient au moins 10 éléments de chaque type. Alors l'utilisateur 1 et 2 retournent des données car ils les insertions sont faites sur ces deux pour tester la validité. Sauf l'identifiant 1 et 2 peuvent retourner des résultats car sont les seuls qui ont des listes contenant des objets supérieurs à 10.

6. Les fichiers fournis :

Le fichier bd_collection contient :

- Les tables.
- Insertions.
- Suppression tables.
- Les contraintes d'intégrité dont triggers associés à la base.
- Les fonctions et procédures PL/SQL.
- Les déclencheurs demandés.

Les fichers req1, req2, req3, req4, req5 contiennent les requêtes demandées.

Les fichiers func1, proc1, proc2 contiennent les tests pour les procédures créer dans bd_collection.

Les fichers archivetest et nouveautetest contiennent les tests pour les triggers demandés.