Sabina Askerova
Adel Ettaoui
Bases de données 2
L3S5 Informatique
2022-2023

Projet collections d'objets culturels

Partie 2

Vous trouverez dans l'archive les fichiers de suppression (droptables.sql) et de création des tables (creation.sql) et des séquences (sequences.sql).
Le fichier requetes.sql contient la partie 'Requêtes SQL', et requetes_test.sql permet de les tester
De même pour la partie 'PL/SQL' avec les fichiers plsql.sql et data_tests_plsql.sql, ce dernier contenant les données et en fin de fichier les exécutions.
Enfin les déclencheurs dans triggers.sql et leurs tests dans triggers_test.sql.
Il est préférable de tester d'abord les requêtes SQL, puis le code PL/SQL, puis les triggers.

Requêtes SQL

1. Pour la première requête, la version qui utilise une requête imbriquée est plus intéressante que la version désimbriquée. En effet, comme il n'est pas possible de faire un JOIN entre la table LISTE et la table OBJET, SQL est obligé de faire un MERGE JOIN (produit cartésien). Cela résulte en x*y n-uplets traités, x étant le nombre de n-uplets dans LISTE et y le nombre de n-uplets dans OBJET.
   Par exemple:

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			1	4
FILTER				
Filter Predicates				
COUNT($vm_col_1)>= (SELECT COUNT($vm_col_1) FROM		(SELECT TYPE_OBJET $vm_col_1 FROM OBJET OBJET GROUP BY TYPE_OBJET) VM_NWVW_2		
HASH		GROUP BY	1	4
VIEW	SYS.VM_NWVW_1		7	4
HASH		GROUP BY	7	4
TABLE ACCESS	LISTE	FULL	30	3
SORT		AGGREGATE	1	
VIEW	SYS.VM_NWVW_2		3	4
SORT		GROUP BY	3	4
TABLE ACCESS	OBJET	FULL	40	3

version imbriquée

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			1	46
FILTER				
Filter Predicates				
COUNT(DISTINCT NLSSORT(L.TYPE_LISTE,'nls_sort="BINARY"'))>=COUNT(DISTINCT NLSSORT(O.TYPE_OBJET,'nls_sort="BINARY"'))				
SORT		GROUP BY	1	46
MERGE JOIN		CARTESIAN	1200	45
TABLE ACCESS	LISTE	FULL	30	3
BUFFER		SORT	40	43
TABLE ACCESS	OBJET	FULL	40	1

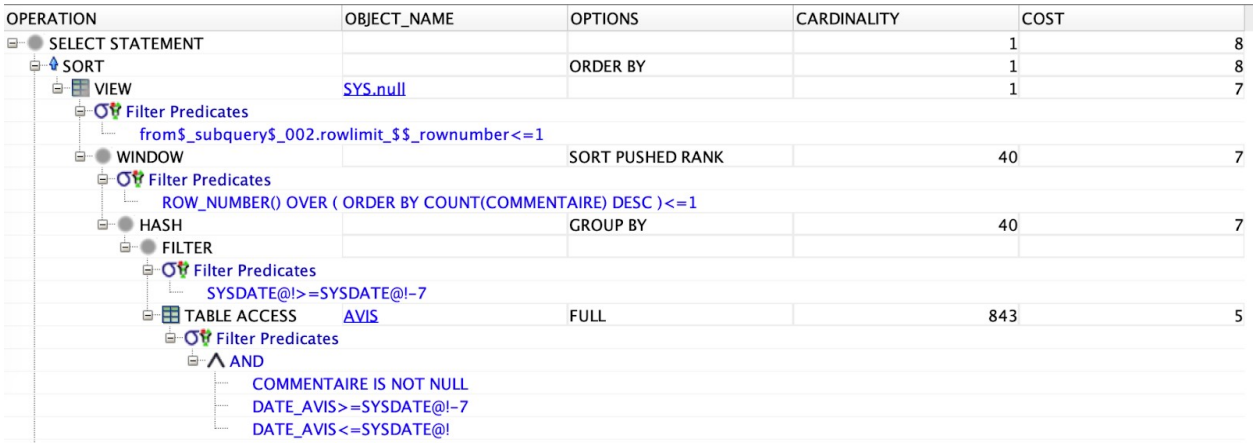
version désimbriquée

Ainsi définir l'index sur le nombre de types de listes créées par l'utilisateur est inutile.

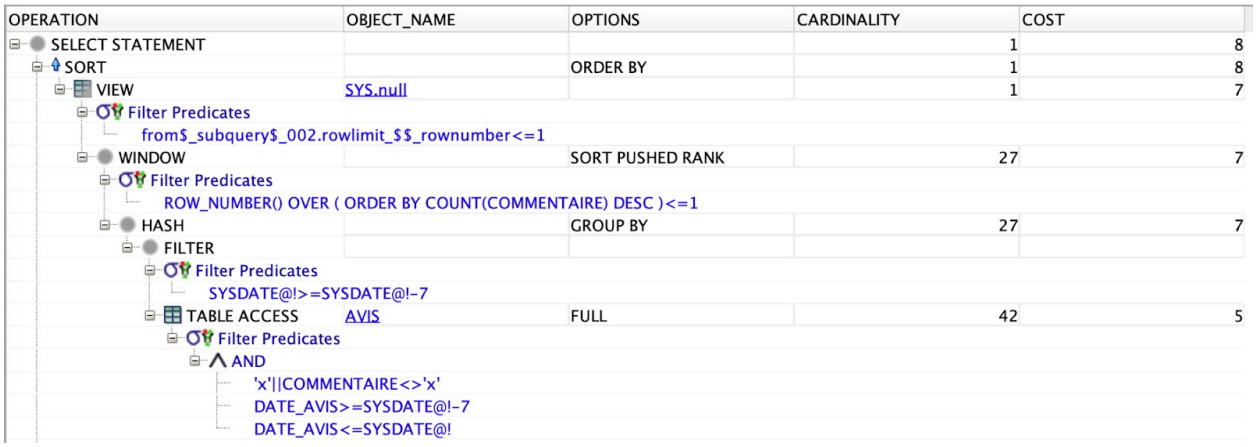
2. La deuxième requête sera aussi plus efficace en version qui utilise une requête imbriquée car les restrictions (sur la moyenne de notations et sur le nombre de collections) sont plus basses dans son arbre relationnel.

Pour les trois premières requêtes, les attributs utilisés pour le traitement sont les clés primaires, donc déjà indexées et optimisées.

4. On a une table AVIS assez grande.
Performance sans index :



Performance avec index sur l'attribut COMMENTAIRE (test IS NOT NULL) :



5. La ligne 'WHERE EXTRACT(YEAR FROM a.date_avis) = EXTRACT(YEAR FROM SYSDATE) - 1)' fait que cette requête ne devrait fonctionner qu'en 2023, pour que l'année précédente soit '2022'.

Procédures et fonctions PL/SQL

2. Pour cette procédure, on passe par une fonction intermédiaire ‘insert_fav_listes’ qui s’occupe de vérifier pour un type donné (‘film’, ‘livre’, ou ‘jeu video’) s’il y en a assez, c’est-à-dire au moins 10 que l’utilisateur a noté pour générer une liste des 10 les mieux notés.
La procédure ‘favorites’ s’occupe ensuite simplement d’appeler cette fonction pour chaque type d’objet et d’afficher un message pour prévenir l’utilisateur si la liste pour le type considéré a pu être générée ou non. Cela donne le résultat suivant :

```
Insertion d'une liste de film effectuée
Insertion d'une liste de livre effectuée
Insertion d'une liste de jeu video effectuée


Procédure PL/SQL terminée.

Insertion d'une liste de film effectuée
Insertion d'une liste de livre effectuée
Liste de jeu video non suffisante : pas générée


Procédure PL/SQL terminée.

Liste de film non suffisante : pas générée
Insertion d'une liste de livre effectuée
Insertion d'une liste de jeu video effectuée
```

3. Pour cette question on utilise également une fonction intermédiaire qui s’occupe de récupérer dans une table custom Y utilisateurs, ces utilisateurs ayant mis la même note que l’utilisateur en argument pour au moins Z objets**.**
La procédure ‘suggestions’ se sert ensuite de cette table pour récupérer X objets dont le score moyen est le plus élevé pour les objets communs à ces Y utilisateurs. Nous avons fait le choix de laisser apparaître comme suggestion des objets déjà notés par l’utilisateur tant qu’ils ne font pas partie d’une de ses listes (comme Netflix qui peut nous recommander des séries que l’on a déjà regardées). Si rien ne s’affiche après ‘Liste de suggestions pour l’utilisateur …’, cela peut signifier qu’il n’y a pas d’utilisateurs qui ont assez de notes en commun pour faire des suggestions.

```
Liste de suggestions pour l'utilisateur Apearson82 :
1. Identifiant de l'objet : 120, nom : 1984, score moyen : 19
2. Identifiant de l'objet : 138, nom : D'apres une histoire vraie, score moyen : 19
3. Identifiant de l'objet : 154, nom : Life Is Strange, score moyen : 18
4. Identifiant de l'objet : 121, nom : Animal Farm, score moyen : 18
5. Identifiant de l'objet : 107, nom : Stronger, score moyen : 18
6. Identifiant de l'objet : 132, nom : Vivre vite, score moyen : 18
7. Identifiant de l'objet : 142, nom : Overwatch, score moyen : 18
8. Identifiant de l'objet : 115, nom : Prisoners, score moyen : 18
9. Identifiant de l'objet : 159, nom : Uncharted, score moyen : 17
10. Identifiant de l'objet : 141, nom : Hollow Knight, score moyen : 17
```

Triggers

Les contraintes dynamiques de la base sont:

- Lors de l'ajout d'un objet dans une liste de l'utilisateur ordinaire, vérifier que l'objet est de même type que la liste (*t_appartient_liste*);

- Lors d'une inscription d'un utilisateur ordinaire, vérifier qu'un login est composé de la première lettre du prénom et des 7 premières lettres du nom (en minuscules) suivies de deux chiffres (*t_login*).

Quelques contraintes statiques de la première partie du projet sont devenues dynamiques dans cette partie en raison de la définition de l'utilisateur spécial 'nouveauté'.

Contrairement aux utilisateurs ordinaires, l'utilisateur 'nouveauté' a le nom, le prénom, le mdp, la date de naissance vides. Cet utilisateur est fictif et n'a que son login qui est défini. Cet utilisateur est créé une seule fois par le système (lors de l'exécution d'un trigger *t_liste_mois*).

Les listes de l'utilisateur 'nouveauté' ont tous les objets ajoutés pendant le mois XXX, sous-entendu que ces objets ont des types différents. donc ces listes n'ont pas de type (*t_liste*).

L'utilisateur 'nouveauté' n'a pas à respecter les contraintes dynamiques du login.