UQAC
Université du Québec
à Chicoutimi

Travail Pratique 3

Étudiants	**Mamadou Billo Diallo & Alpha Macky Daff**
Cours	**8TRD157** **Base de données avancée**
Professeur	Éric Dallaire

Partie A : Indexation

Identification de 5 endroits où il serait approprié de faire de l'indexation

En analysant bien notre base de données en fonction des questions posés dans le document tp2 on remarque qu'il est plus avantageux d'indexer certains des champs des tables mère de cette façon le temps pour retourner les données de ces champs diminue et accélère les procédures, fonctions, ...

Index 1 :

• **Identification de la table et le/les champs affecté(s)**

Table : Periode

Champs affectés : CodeForfait

• **Requêtes de création et de suppression de l'index**

```
CREATE INDEX CodeForfait_INDEX
   ON PERIODE(CodeForfait) ;
   DROP INDEX CodeForfait_INDEX;
```

• **Description de l'impact (probable et théorique) du changement**.

Impact théorique : En **prolongeant le forfait, l**'index sur le champ CodeForfait pourrait accélérer le résultat de recherche du bon code de forfait.

Impact probable : la rapidité de l'exécution de cette requête augmente donc plus rapide de retrouver la période avec le code de forfait entré.

Index 2 :

• **Identification de la table et le/les champs affecté(s)**

Table : SUCCES

Champs affectés : IdSucces

• **Requêtes de création et de suppression de l'index**

```
CREATE INDEX IdSucces_INDEX
ON SUCCES(IdSucces);
 DROP INDEX IdSucces_INDEX;
```

• **Description de l'impact (probable et théorique) du changement**.

Impact théorique : le titre d'un succès pourrait être retrouvé plus rapidement.

Impact probable : Evidement le titre d'un succès ressort plus rapidement.

Index 3 :

• **Identification de la table et le/les champs affecté(s)**

Table : Joueur

Champs affectés : Actif

• **Requêtes de création et de suppression de l'index**

```
CREATE INDEX Actif_INDEX
  ON Joueur (Actif) ;
 DROP INDEX Actif_INDEX;
```

• **Description de l'impact (probable et théorique) du changement**.

Impact théorique : En mettant l'index sur ce champ ça peut faciliter la recherche du joueur le plus populaire.

Impact probable : il est rapide de retrouver quel joueur est actif ou pas.

Index 4 :

• **Identification de la table et le/les champs affecté(s)**

Table : JEU

Champs affectés : CodeESRB

• **Requêtes de création et de suppression de l'index**

```
CREATE INDEX CodeESRB_INDEX
```

ON Jeu (CodeESRB) ;
DROP INDEX CodeESRB_INDEX;

• **Description de l'impact (probable et théorique) du changement.**

Impact théorique : pourrait aider à retrouver le **jeuLePlusPopulaire.**

Impact probable : il est plus rapide de retrouver le **jeuLePlusPopulaire.**

Index 5 :

• **Identification de la table et le/les champs affecté(s)**

Table : JEU

Champs affectés : Nom

• **Requêtes de création et de suppression de l'index**

CREATE INDEX Nom_INDEX
ON Jeu (Nom) ;
DROP INDEX Nom_INDEX;

• **Description de l'impact (probable et théorique) du changement.**

Impact théorique : retrouver plus rapidement le nom d'un jeu.

Impact probable : il sera est rapide de retrouver le nom d'un jeu afin de le supprimer de la liste.

Pour finir avec la partie A on peut rajouter un Impact théorique et probable qui est que l'utilisation des index prend beaucoup d'espace mémoire.

Partie B : Optimisation des Scripts

Identification d'endroits dans nos scripts PL/SQL où il serait approprié de faire de l'optimisation.

Optimisation :

• **Le script de départ et le script modifié**

Le script de départ :

```
CREATE OR REPLACE FUNCTION jeuLePlusPopulaire ( Code IN CHAR)
RETURN  VARCHAR2
IS
    nomP VARCHAR2(100);
BEGIN

      SELECT nomJeu INTO nomP
    FROM (SELECT J.Nom AS nomJeu, COUNT(DISTINCT CP.NoJoueur) AS nbJoueurs
          FROM JEU J
          JOIN CONTENU C ON J.IdJeu=C.IdJeu
          JOIN SUCCES_REALISE SR ON C.IdContenu=SR.IdContenu
          JOIN CONTENU_POSSEDE CP ON CP.IdContenu=SR.IdContenu
          JOIN JOUEUR JO ON CP.NoJoueur=JO.NoJoueur
          WHERE J.CodeESRB=Code
          AND C.TypeContenu='J'
          AND JO.Actif='1'
          AND CP.NoJoueur IS NOT NULL
          GROUP BY J.Nom
          ORDER BY COUNT(DISTINCT CP.NoJoueur) DESC)
    WHERE ROWNUM = 1;

    RETURN nomP;

END jeuLePlusPopulaire;
/
```

Le script modifié :

```
CREATE OR REPLACE FUNCTION jeuLePlusPopulaire ( Code IN CHAR)
RETURN  VARCHAR2
IS
    nomP VARCHAR2(100);
BEGIN

      SELECT nomJeu INTO nomP
    FROM (SELECT J.Nom AS nomJeu, COUNT(DISTINCT CP.NoJoueur) AS nbJoueurs
          FROM JEU J
          JOIN CONTENU C ON J.IdJeu=C.IdJeu
          JOIN SUCCES_REALISE SR ON C.IdContenu=SR.IdContenu
          JOIN CONTENU_POSSEDE CP ON CP.IdContenu=SR.IdContenu
          JOIN JOUEUR JO ON CP.NoJoueur=JO.NoJoueur
          WHERE J.CodeESRB=Code
          AND C.TypeContenu='J'
          AND JO.Actif='1'
          AND CP.NoJoueur IS NOT NULL
          GROUP BY J.Nom
          ORDER BY nbJoueurs DESC)
    WHERE ROWNUM = 1;

    RETURN nomP;

END jeuLePlusPopulaire;
/
```

• **Identification et description clair de la raison des modifications**
Sur le premier script on a défini COUNT(DISTINCT CP.NoJoueur) AS nbJoueurs.
Mais dans ORDER BY on a oublié d'utiliser nbJoueurs et on a utilisé la même syntaxe et on a demandé à Oracle de refaire le même calcul.
• Description de l'impact (probable et théorique) de votre changement
Impact théorique **:** ceci évite à oracle de répéter le même calcul pour la même syntaxe.
Impact probable : Il n'y a pas de duplication de code donc il y'a gain de temps.

Bien évidemment pour cette partie, on ne dit pas qu'Il n'y en a pas, peut-être qu'il en a, mais selon notre expérience on n'arrive pas à retrouver d'autre élément où il faut optimiser les syntaxes de nos requêtes.
Merci.