Plouhinec Glenn
Le Badezet Benoît

TP4 – Plans d'exécution

Partie 1 : Prise en main de la base

Pour consulter le schéma de chacune des tables *facts, authors, collaborations, dates, publications, squads, supports*, nous utilisons la requête SQL :
DESCRIBE <nom_table>;

De plus, nous avons également compté le nombre de tuples de chacune de ces tables:
SELECT COUNT(*) FROM <nom_table>;

Ainsi, nous avons pu identifier les schémas des tables suivants, également présents dans le fichier *Part1.sql* :

Facts : (**PUBLICATION_ID, SQUAD_ID, DATE_ID, SUPPORT_ID**), 473 176 tuples

Authors : (AUTHOR_ID, FIRST_NAME, LAST_NAME), 454 807 tuples

Collaborations : (**SQUAD_ID, AUTHOR_ID**), 1 212 896 tuples

Dates : (DATE_ID, YEAR, MONTH_LABEL, MONTH, DAY), 407 tuples

Publications : (PUBLICATION_ID, TITLE, ABSTRACT, NB_PAGES), 481 659 tuples

Squads : (SQUAD_ID), 479 540 tuples

Supports : (SUPPORT_ID, TYPE, NUM), 11 243 tuples

Les clefs primaires de ces tables sont soulignées, et les clés étrangères sont en gras. Pour déterminer les clefs primaires de ces tables, il nous a fallu comparer le nombre d'occurrences (*SELECT COUNT (DISTINCT <nom_attribut>) FROM <nom_table>;*) de chacun des attributs, et les comparer avec le nombre de tuples, ce qui nous a permis d'identifier les clés les plus triviales. D'autres clés ont pu être déterminées en raisonnant par rapport au schéma des tables, ce qui nous a permis de déterminer notamment les clés de la table Collaborations.

Après avoir identifié le schéma de ces tables ainsi que les clés primaires et étrangères, on remarque que des jointures peuvent être faites entre Facts et Publication, Facts et Squads, Facts et Dates, Facts et Support, ainsi que Collaborations et Authors (voire Collaborations et Squads, mais il n'existe pas de réel intérêt), ou encore Publications avec Collaborations et Authors, etc...

Partie 2 : Plans d'execution

Nota Bene : Dans un but de lisibilité les résultats des plans d'exécution des requêtes se situes en commentaire sous chacune des requêtes dans le fichier *Part2.sql* .

REQUÊTE 1

Publication :
1- Parcours complet de la table
Filtre pour afficher les résultats
Temps : 00:00:23

Publication1 :
1- Parcours de la table en fonction des index posés
Accès pour afficher les résultats
Temps : 00:00:01

Cette différence s'explique grâce à l'indexation de la table publications1 qui permet à la recherche d'être plus optimisée.

REQUÊTES 2 & 3 :

Les différences sont mineures: le nombre de rows sélectionnées pour publications1 est supérieur à celui de publication, cela influe un peu sur le temps d'exécution (une seconde d'écart).

REQUÊTE 4

Publication :
On utilise des boucles imbriquées et des hachages afin de parvenir à notre table finale
On fait deux accessions puis un filtre puis un nouvel accès ou afficher nos résultats
On remarque aussi l'utilisation d'un index pour la table

Publication1 :
On n'utilise aucune boucle imbriquée pour parvenir a notre table finale, nous perdons plus de temps que pour publication (00:01:18 contre 00:00:55 pour les boucles imbriquées)
On fait trois accessions puis ensuite on filtre les résultats

La requête pour publication vas plus vite que la requête pour publication1 !

Partie 3 : Les index

Les index définis sur les différentes tables de la base peuvent être observés grâce à la requête suivante :

SELECT INDEX_NAME,INDEX_TYPE,TABLE_NAME,NUM_ROWS FROM ALL_INDEXES WHERE TABLE_NAME = '<nom_table>';

Les résultats de cette requête pour chacune des tables de la base peut être observé dans le fichier *Part3-1.txt*, mais globalement, nous avons pu identifier les index suivants :

PK_PUBLICATION pour la table Publications

INDEX_PUB1 et PUB1 pour la table Publications1

PK_FACT pour la table Facts

PK_AUTHOR pour la table Authors

PK_COLLABORATION pour la table Collaborations

PK_DATE_ID pour la table Dates

PK_SQUAD pour la table Squads

PK_SUPPORT pour la table Supports

On observe alors que la table publications1 possède un index PUB1 ayant un nombre de rows inférieur à celui de l'index PK_PUBLICATIONde la table publication, ce qui explique alors que les requêtes faites avec la table publications1 avaient un temps d'exécution plus faible en général.

Ensuite, nous visualisons le plan d'exécution de deux requêtes sur la table Avis, créée dans les TPs précédent :

*SELECT * FROM AVIS WHERE NOTE > 10;*
*SELECT * FROM AVIS WHERE ROUND(NOTE,2) > 10;*

Les résultats des plans d'exécution de ces requêtes se trouve dans le fichier *Part3-2.txt*.

Nous créons alors un index sur l'attribut note :

CREATE INDEX INDEX_NOTES ON AVIS (NOTE);

Puis nous relançons les mêmes requêtes. On observe alors que les requêtes ont un temps d'exécution similaires, étant donné qu'il n'y a que très peu de tuples dans cette table. Cependant on remarque que le coût CPU est tout de même inférieur après avoir créé un index.

Ainsi, la création d'indexes dans les tables d'une base de données permet de réduire de façon non négligeable le temps d'exécution des requêtes, ainsi que le coût CPU.