Contrôle continu 1 : 06-01-2022

Mécanismes Internes des SGBD

Durée : 1h00

Ce devoir peut être rédigé par groupe de 1 ou 2.

Le devoir est à déposer sur Celene ***EN FORMAT PDF*** *avec éventuellement des images (1 seul dépôt en indiquant les 2 noms le cas échéant).*

L'utilisation d'internet est autorisée, sauf pour une communication avec une personne autre que votre binôme.

Préliminaires :

On considère le schéma de la base de données du TP en ligne :

```
CREATE TABLE Artiste (
   id integer,
  nom varchar(30),
  prenom varchar(30),
  annee_naissance integer,
  primary key (id)
  )

CREATE TABLE Film (
        id integer,
        titre varchar(50),
        annee integer
        id_realisateur integer,
        genre varchar(30),
        resume text,
        code_pays varchar(4),
        version integer,
        primary key (id),
        foreign key (id_realisateur) references Artiste(id),
        foreign key (code_pays) references Pays(code)
        )

CREATE TABLE Role (
           id_film integer,
           id_acteur integer,
           nom_role varchar(60),
           primary key (id_film, id_acteur) ,
           foreign key (id_film) references Film(id),
           foreign key (id_acteur) references Artiste(id)
        )
```

Vous justifierez bien toutes vos réponses.

Exercice 1 : Optimisation

On pose la requête suivante :

```
SELECT F.TITRE
FROM FILM F, ARTISTE A
WHERE A.NOM = 'Hitchcock'
AND F.id_realisateur = A.id
```

L'exécution de cette requête sur une base de données non indexée donne le plan d'exécution suivant :

```
Jointure de hachage (temps de réponse:28.95 ; temps d'exécution:48.01 ;
nombre de nuplets:2 ; mémoire allouée:18) condition de jointure
(f.id_realisateur = a.id)

-> parcours séquentiel de film f (temps de réponse:0.00 ; temps
d'exécution:18.21 ; nombre de nuplets:221 ; mémoire allouée:22)

-> hachage (temps de réponse:28.81 ; temps d'exécution:28.81 ; nombre de
nuplets:11 ; mémoire allouée:4)

      -> parcours séquentiel de artiste a (temps de réponse:0.00 ; temps
      d'exécution:28.81 ; nombre de nuplets:11 ; mémoire allouée:4) filter:
      ((nom) = 'hitchcock')
```

Q1.0 : Que calcule cette requête ?
Q1.1 : Dessinez le plan d'exécution de cette requête. Est-ce que l'algorithme de jointure utilisé est bloquant ?
Q1.2 : Quelle table a été stockée dans la table de hachage pour effectuer l'algorithme de jointure de hachage ? Pourquoi ?

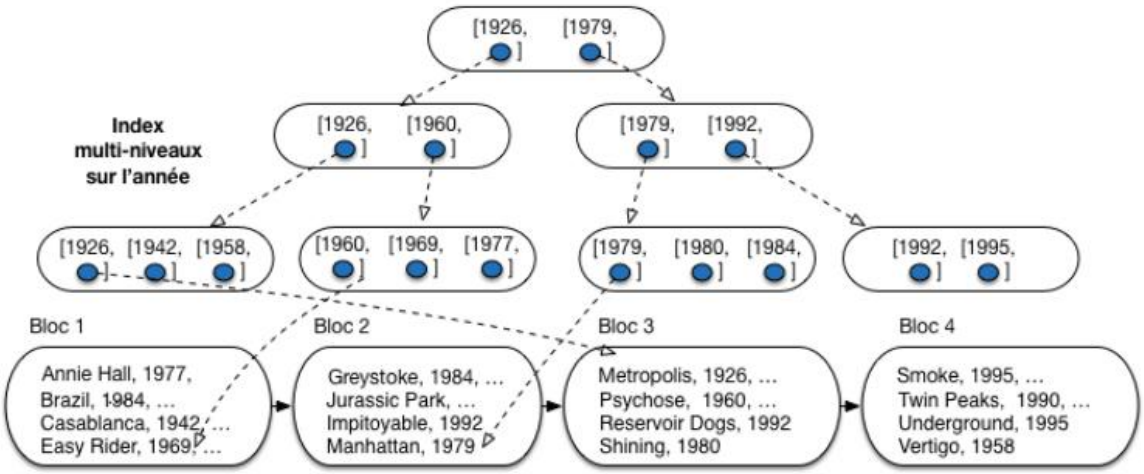
Si on lance cette même requête sur la grande base avec index, voici le résultat (on suppose que bitmap index scan correspond à une recherche dans l'index, et que bitmap heap scan correspond à un accès sur le disque à une page et une entrée donnée par l'index) :

```
Jointure de hachage (temps de réponse:110949.45 ; temps d'exécution:329699.35
; nombre de nuplets:18490 ; mémoire allouée:18) condition de jointure
(f.id_realisateur = a.id)

-> parcours séquentiel de film f (temps de réponse:0.00 ; temps
d'exécution:175598.00 ; nombre de nuplets:2210000 ; mémoire allouée:22)

-> hachage (temps de réponse:108883.48 ; temps d'exécution:108883.48 ; nombre
de nuplets:125918 ; mémoire allouée:4)

      -> bitmap heap scan on artiste a (temps de réponse:3036.42 ; temps
      d'exécution:108883.48 ; nombre de nuplets:125918 ; mémoire allouée:4)
      recheck cond: ((nom) = 'hitchcock')
            -> bitmap index scan on artiste_nom_prenom_idx (temps de
            réponse:0.00 ; temps d'exécution:3004.95 ; nombre de
            nuplets:125918 ; mémoire allouée:0) index cond: ((nom) =
            'hitchcock')
```

Q1.3 : Dessinez le plan d'exécution de cette requête.
Q1.4 : Sur quel attribut est l'index utilisé ici ? Est-ce que cet index est utilisé lors de l'exécution de l'algorithme de jointure ? Pourquoi ?

Exercice 2 : Arbre B+

Voici un dessin d'un arbre B+ indexant l'attribut année de la table FILM. Les nœuds de l'index peuvent avoir au plus 3 entrées.



On pose la requête

```
SELECT *
FROM FILM F
WHERE F.ANNEE = 1984
```

Q2.1 : En supposant qu'on utilise l'index, et que l'index tient en mémoire, combien de pages seront lues sur le disque ?

Q2.2 : En supposant qu'on n'utilise pas l'index, combien de pages seront lues sur le disque ?

Q2.3 : On insère un 5e bloc contenant le film « Superman » sorti en 1978. Dessinez l'index après cette insertion (inutile de dessiner les blocs contenant les données).

Exercice 3 : Implémentation d'opérateur

Dans cette question, on suppose disponible le code présenté en TD : https://benjamin-nguyen.fr/ENS/4ASTI-BD2/jointure.zip mais si vous le souhaitez vous pouvez coder les autres méthodes dans un autre langage ou en pseudo code.

On dispose d'une implémentation d'une Table, chaque colonne étant de type entier.

*Q3.1 : Implémentez l'opérateur **RESTRICT** qui prend en entrée 3 paramètres : op1 (un Operateur), col (un entier) et e, un entier. Cet opérateur doit produire à partir du flux d'entrée op1 uniquement les Tuples t vérifiant la condition t[col] == e. Est-ce que votre opérateur est bloquant ? Quelle est sa complexité ?*

*Q3.2 (Bonus) : Implémentez l'opérateur **HASHDISTINCT** qui prend en entrée 1 paramètre : op (un opérateur) et qui réalise la fonction **DISTINCT** de la manière suivante : a) on parcourt l'intégralité de l'operateur et on construit une table de hachage en hachant le tuple en entrée. On stocke dans cette table le tuple, à moins qu'il ne soit déjà présent dans la table. b) on parcourt la table de hachage et on produit chaque tuple distinct. Vous expliquerez votre choix de fonction de hachage et de taille de la table de hachage. On fera l'hypothèse que la table de hachage tient en mémoire. Est-ce que votre opérateur est bloquant ? Quelle est sa complexité ? Pourrait-on utiliser un algorithme à base de table de hachage si la table de hachage ne tient pas en mémoire ?*