Exercice 01 :

Soit le schéma relationnel suivant :

Propriétaire (numPropriétaire, prénom, nom, adresse, numtél).

Propriété (numPropriété, ville, type, location, numPropriétaire)

Client (numClient, prénom, nom, adresse, typepréf, locMax)

Visite (numPropriété, numClient, datevisite)

1. Exprimer en SQL la requête suivante : pour le compte de locataires potentiels qui recherchent des **appartements**, trouver tous les logements qui correspondent à leurs exigences et qui appartiennent au propriétaire de numéro **CP93**.

SELECT P.numPropriété, P.ville

FROM Client C, Visite V, Propriété P

WHERE typepréf=’ **appartements’**

AND typepré= type

AND locMax>= location

AND numPropriétaire=’ **CP93’**

AND C. numClient=V. numClient

AND V. numPropriété=P. numPropriété

1. Donner l’arbre algébrique qui correspond à la requête SQL.

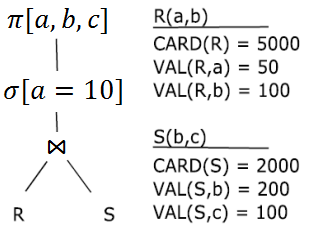
|  |
| --- |
| numPropriété, ville  **σ** ((typepréf=’ appartements’) (typepré= type) (locMax>= location) (numPropriétaire=’ CP93’))  Propriété    Visite  Client |

1. Optimiser la requête en utilisant les règles de transformation.

|  |
| --- |
| numPropriété, ville  **σ** ((typepré= type) (locMax>= location))  **σ** (numPropriétaire=’ CP93’)    Propriété  Visite  **σ** (typepréf=’ appartements’)  Client |

|  |
| --- |
| numPropriété, ville  **σ** (locMax>= location)  numPropriété, location, ville    **σ** ((type=’ appartements’) (numPropriétaire=’ CP93’))  numClient, numPropriété  numClient, locMax  Propriété  Visite  **σ** (typepréf=’ appartements’)  Client |

**Exercice 02 :** Soit un arbre algébrique et des informations statistiques sur les tables qu’il manipule.



Indiquez les exécutions possibles et évaluez leur coût d’exécution.

|  |
| --- |
| TR : 103\*(size(a)+size(b) +size(c))  a, b, c  S  R  S  TR : 103\*(size(a)+size(b) +size(c))  5\*104/50=103 tuples  a, b, c  **σ** (a=10)  (102 \*2\*103)/200=103 tuples  (5\*103 \*2\*103)/200=5\*104 tuples  **σ** (a=10) 5\*103/50=102    R  Coût du Plan 1 : 5\*104+103=51\*103  opérations Coût du Plan 2 : 102+103=1.1\*103 opérations |

**Exercice 03 :**

**Information tirées à partir des données de l’énoncé :**

• Il y a 10 000 enregistrements de la table étudiants stockés sur 1000 pages 🡪 un bloc étudiant contient 10 tuples et card(Etudiant)=10000.

• Il y a 50 000 enregistrements de la table livres stockés sur 5000 pages 🡪 un bloc livre contient 10 tuples et card(Livre)=50000.

• Il y a 300 000 enregistrements de la table emprunter stockés sur 15000 pages 🡪 un bloc livre contient 20 tuples et card(Emprunter)=300000.

• Il y a 500 auteurs différents 🡪 VAL(auteur, Livre)=500

• Les âges des étudiants vont de 7 à 24 ans 🡪 VAL(age, etudiant)=18, max(age)=24 et min(age)=7.

a. Donner un plan d’exécution pour cette requête, en supposant qu'il n'y a pas d'index et que les données ne sont pas triées sur n'importe quel attribut.

|  |
| --- |
| nom : à l’exécution  **σ** ((auteur=’ DATE) (age > 12) (age<20)) : à l’exécution  Jointure par boucle imbriquée « nested loop » sur les tuples  Livre : scanner 50000  Jointure par boucle imbriquée sur les blocs    Emprunter : Scanner 300000 tuples  Etudiant : Scanner 10000 tuples |

b. Calculez le coût de ce plan et la cardinalité du résultat.

**Abréviation : ET : Etudiant, EM : Emprunter, L : livre, B : Bloc**

|  |  |  |  |
| --- | --- | --- | --- |
| **Opération** | **Coût** | **Cardinalité** | **Commentaire** |
| **ET EM** | B(ET) +B(ET) \* B(EM)  1000+1000\*15000  =15001000 opérations | * 300000 tuples(jointure sur clé étrangère (idet) : (300000\*10000)/ 10000). | * On utilise la boucle imbriquée basée sur les pages, chaque itération on charge une page étudiant et pour chaque page on charge les pages de la table emprunter, on suppose que les tuples résultants restent en mémoire centrale (il y a assez d’espace). Le tuple résultant sera utilisé pour la deuxième jointure. |
| **(ET EM) L**  **= C L** | C\*B(L) 300000\*5000  1500000000 opérations | * 300000 tuples(jointure sur clé étrangère (idl) : (300000\*5000)/ 5000). | * Les 300000 tuples résultants qui existent au niveau de la mémoire centrale vont être utilisés directement pour la deuxième jointure, on les nomme C. Pour chaque tuple de C on charge tous les blocs de la table livre. |
| **σ**  et | A l’exécution | * 300000\* **σauteur**\* **σage**   30000\*(1/500)\*(7/18)  234\* size (nom) | * On suppose qu’on a une distribution uniforme sur l’âge et auteur. |
| **Total** | 1515001000 opérations | 234 \* size (nom) octets |  |

c. Suggérer deux index et un autre plan pour cette requête.- On peut avoir plusieurs solution, pour illustration on suppose d’avoir un index de *type b-arbre n’est pas organisé en cluster* pour la table **Livre** défini sur l’attribut **auteur** et un index de *type b-arbre organisé en* cluster pour la table **Emprunter** sur l’attribut **idl**.

**- NB :**

|  |
| --- |
| * Il existe de nombreux types d’indexes, dont les plus connus et utilisés sont l’index en cluster (clustered index) et l’index non-cluster (non-clustered index). Et la différence entre les 2 est la suivante : * L’index en cluster (ou ordonné) est un index au sein duquel les enregistrements sont physiquement triés, et ses pages de données au niveau feuille et les numéros d’enregistrements au niveau intermédiaire. C’est ainsi qu’une table ne peut avoir qu’un index en cluster à la fois dans la mesure où les lignes ne peuvent être triées qu’en un seul ordre physique. * L’index non-cluster (ou non-ordonné) est un index au sein duquel le tri logique des enregistrements (i.e., tri des pages d’indexes) ne correspond pas au tri physique des enregistrements (sur le disque). De plus, au niveau structurel, seuls les numéros d’enregistrements sont placés au niveau feuille, ce qui permet donc, à une table, de pouvoir disposer de plusieurs indexes non-clusters. |

|  |
| --- |
| nom : à l’exécution  **σ** ((age > 12) (age<20)) : à l’exécution  Jointure par boucle imbriquée « nested loop » sur les blocs  ide  Etudiant : Scruter 10000 tuples  Jointure par boucle imbriquée indexée    idl  **σ** (auteur=’ DATE’)  Emprunter : Filtrage par index cluster (défini sur idl)  Livre : filtrage par index non cluster (défini sur auteur) |

d. Calculez le coût de nouveau plan.

|  |  |  |  |
| --- | --- | --- | --- |
| **Opération** | **Coût** | **Cardinalité** | **Commentaire** |
| **σ** (auteur=’ DATE’) :  Filtrage par un index sur la table Livre | Card(L)\*(1/Val(auteur, L))  =50000\*1/500  =100 opérations | * 100 tuples | * on suppose que toutes les pages index sont en mémoire centrale. L'index qu'on a défini et organisé en non cluster, la page index contient la clé (nom auteur) et l'idl (identifiant physique de livre), pour la jointure on a besoin que de idl d'ou on a besoin de faire une entrée/sortie pour les 100 tuples. |
| (L EM) | 100\*((card(EM)/Val(idl, EM))/N(EM))  100\*((300000/50000)/20)  =100 opérations | * (100\*300000)/ Max(100, Val(idl, Emprunter))   =600 tuples=C | * Avec l'index organisé en cluster défini sur la table Emprunter (idl) on aura (300000/50000)= 6 tuples seront chargés en une page(chaque page peut contenir 20 tuples), pour chaque tuple résultant (les 100 tuples résultants de l’étape précédente), on va charger cette page. Le résultat des 600 tuples vont rester en mémoire centrale. |
| **C ET** | B(ET)= 1000 opérations | * (10000\*600)/Max(600, Val(ide, Etudiant))   = 600 tuples | * Vu que les 600 tuples sont en mémoire centrale donc on a besoin de scanner que la table étudiant. |
| **σ**  et | A l’exécution | * 600\* **σage**   600\*(7/18)  234\* size (nom) | * On suppose qu’on a une distribution uniforme sur l’âge. |
| **Total** | 1200 opérations | 234 \* size (nom) octets |  |