Exercice 1 : Index non-dense

Pour cet exercice, nous considérons l'instance de la relation film visible dans la figure 1. On considère qu'un enregistrement à une taille de 256 octets et une page à une taille de 512 octets (on suppose qu'une page peut contenir deux enregistrements). Un pointeur sur une page (ou RID) occupe 6 octets et la date 4 octets.

MID	TITLE	YEAR	LANGUAGE
2	Ariel	1988	FI
5	Star Wars	1977	EN
3	Shadow in Paradise	1986	FI
7	Dancer in the Dark	2000	EN
80	Nausicaä	1984	JA
71	Billy Elliot	2000	EN
75	Mars Attacks!	1996	EN
62	2001: A Space Odyssey	1968	EN
19	Metropolis	1927	DE
128	Princess Mononoke	1997	JA
103	Taxi Driver	1976	EN
98	Gladiator	2000	EN
391	A Fistful of dollars	1964	IT

Une instance de Film

Question 1 : Combien d'entrée d'index peut contenir une page d'un index de type 2 trié sur les dates ?

Solution

- Une entrée de l'index utilise 10 octets (4 + 6)
- La taille d'une page est de 512 octets

$\frac{512}{x} = 51,2$, donc 51 entrées de l'index

Question 2 : Donnez le nombre d E/S (nombre de lecture/écriture sur les pages) nécessaires pour mettre à jour les données et l'index en considérant la figure 1 pour l'ajout de l'entrée (4, Barry Lyndon, 1975, EN) pour un fichier trié (croissant) ? pour un index de type 2 trié (en supposant que le fichier contenant la relation est un tas)?

Solution

Dans tous les cas, il faut d'abord effectuer une recherche puis décaler tous les enregistrements succédant à l'enregistrement courant

Pour un index de type 1 Pour retrouver, on cherche le premier enregistrement où la date est supérieure à 1975. On peut utiliser a recherche par dichotomie, ce qui implique au maximum la lecture de $\log_2(P)$ pages ici P vaut 7, donc 3 pages. Par ailleurs, le nouvel enregistrement sera dans la première page, il faut donc décaler toutes les pages, donc écrire sur 7 pages. Nous avons donc 3 lectures et 7 écritures

Pour un index de type 2 La compléxité est la même mais une unique page compose l'index. Il y aura 1 lecture et 1 écriture sur l'index et 1 écriture sur le fichier.

Question 3 : Répondez à la même question si la relation est contenu dans un fichier de 10 000 pages, et que après recherche nous ajoutons l'enregistrement sur la première page.

Solution

- **Index de type 1** il faut $\approx log(10\ 000)$ lectures, donc 13 lectures pour retrouver l'emplacement où sera ajouter l'enregistrement. En revanche il faut faire un décalage dans toutes les pages, donc $\approx 10\ 000$ écritures (précisément 10 001 car nous devons créer une nouvelle page). Donc il faut 13 lectures et 10 000 écritures
- **Index de type 2** Le fichier d'index est formé de $\lceil \frac{10\ 000}{51} \rceil = 197$ pages. il faut donc $\approx \lceil \log_2(197) \rceil = 8$ lectures et décaler toutes les pages d'un enregistrement donc 197 écritures

Question 4 : On se replace dans le cas où la relation n'a que 13 enregistrements (voir la figure). À partir de maintenant on va considérer que le fichier de la relation est un tas et qu'une page de l'index de type 2 peut contenir seulement 3 entrées de l'index sur les dates, on nomme cet index trié A. Donnez les entrées d'index de l'index B non-dense crée sur l'index A. Combien de pages sont modifiées pour l'ajout de l'enregistrement (4, Barry Lyndon, 1975, EN) dans l'index B ?

Solution

Clef	Page
1927	1
1976	2
1986	3
1997	4
2000	5

Il y a seulement deux pages dans B, donc 2 pages sont modifiées dans cet index.

Question 5 : Si on utilise cet index, est-il nécessaire que les pages soient consécutivement ordonnées dans l'index A (en gardant l'ordre des enregistrements dans les pages)? Combien de pages aura t-on besoin de modifier dans ce dernier cas ?

Solution

Tant que l'index B est mis à jour, il n'est pas nécessaire que les pages de l'index A soient ordonnées. Dans ce cas, on pourrait par exemple ajouter une nouvelle page dans l'index A contenant la nouvelle entrée de l'index et la faire pointer sur une nouvelle entrée dans le fichier de données. Il faudra dans tous les cas mettre à jour l'index B.
Donc 1 création de page dans A puis un décalage dans B mais aussi un ajout dans le fichier de données. Il faut donc 4 accés disques au lieu de 5.

Question 6 : On réitère l'opération avec un index C non-dense sur l'index B. Dessinez le nouvel index et déterminez le coût d'une recherche par rapport au nombre d'éléments stockés dans la relation ?

Solution

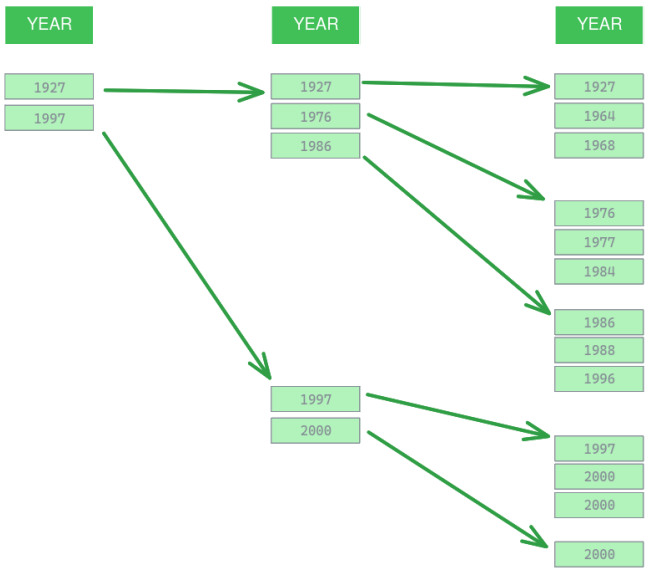
Voir figure de la question suivante pour le dessin. il faut accéder à 3 pages d'index (hauteur de l'arbre) puis un page du fichier de donnés. On pourrait chaîner les pages de l'index A pour améliorer le temps de calcul.

Question 7 : A quoi s'apparente la structure que nous avons crée ? Comment améliorer le coût de l'insertion ?

Solution

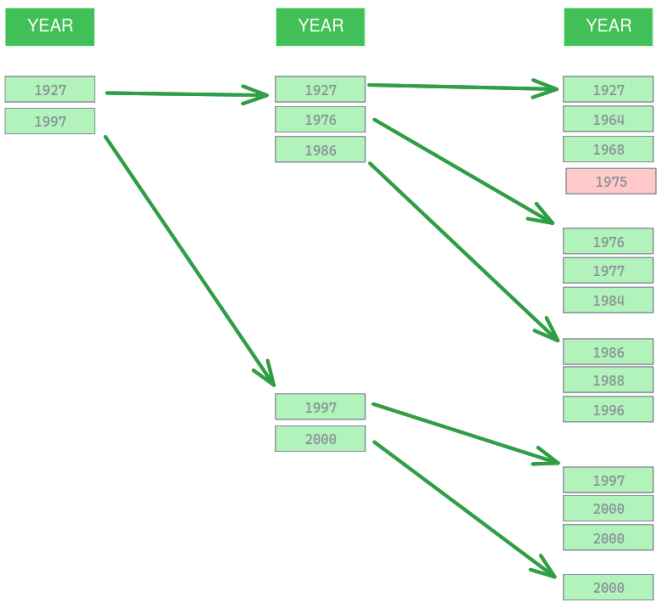
- Ajouter le noeud au bon endroit, puis si trop de noeuds procéder à un éclatement (il s'agit de l'algorithme d'insertion des arbres B+), notons que la structure est légèrement différentes puisque l'on a dans les noeuds interne une répétition du premier élément du fils.

- L'insertion n'est pas optimale dans notre cas à cause de la taille des pages (seulement 3 enregistrements) en revanche on remarquera que l'on créer au pire des cas la valeur de la hauteur en éclatement.

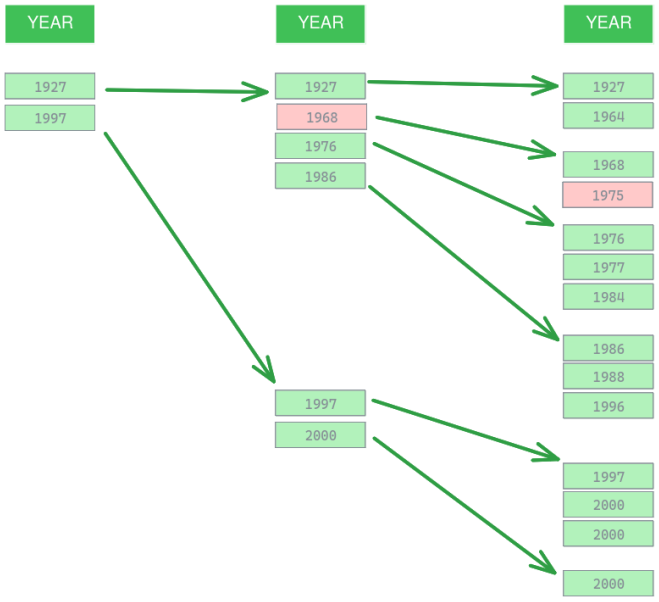


MID	TITLE	YEAR	LANGUAGE
2	Ariel	1988	FI
5	Star Wars	1977	EN
3	Shadow in Paradise	1986	FI
7	Dancer in the Dark	2000	EN
80	Nausicaä	1984	JA
71	Billy Elliot	2000	EN
75	Mars Attacks!	1996	EN
62	2001: A Space Odyssey	1968	EN
19	Metropolis	1927	DE
128	Princess Mononoke	1997	JA
103	Taxi Driver	1976	EN
98	Gladiator	2000	EN
391	A Fistful of dollars	1964	IT

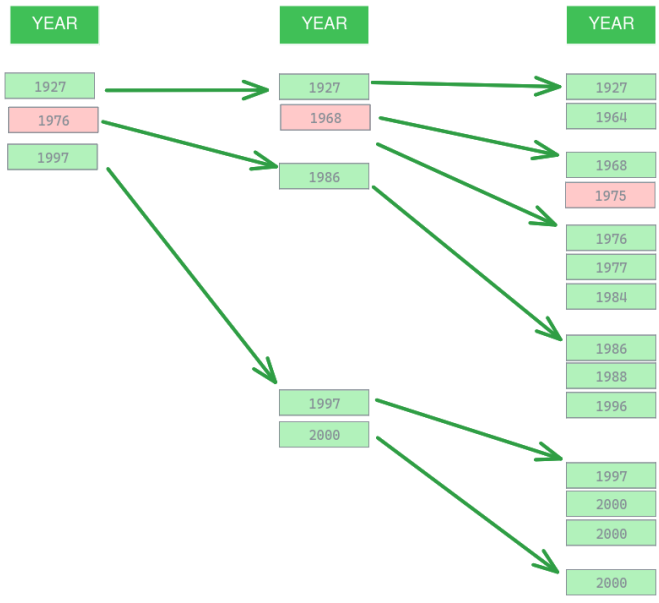
Solution



MID	TITLE	YEAR	LANGUAGE
2	Ariel	1988	FI
5	Star Wars	1977	EN
3	Shadow in Paradise	1986	FI
7	Dancer in the Dark	2000	EN
80	Nausicaä	1984	JA
71	Billy Elliot	2000	EN
75	Mars Attacks!	1996	EN
62	2001: A Space Odyssey	1968	EN
19	Metropolis	1927	DE
128	Princess Mononoke	1997	JA
103	Taxi Driver	1976	EN
98	Gladiator	2000	EN
391	A Fistful of dollars	1964	IT
4	Barry Lyndon	1975	EN



MID	TITLE	YEAR	LANGUAGE
2	Ariel	1988	FI
5	Star Wars	1977	EN
3	Shadow in Paradise	1986	FI
7	Dancer in the Dark	2000	EN
80	Nausicaä	1984	JA
71	Billy Elliot	2000	EN
75	Mars Attacks!	1996	EN
62	2001: A Space Odyssey	1968	EN
19	Metropolis	1927	DE
128	Princess Mononoke	1997	JA
103	Taxi Driver	1976	EN
98	Gladiator	2000	EN
391	A Fistful of dollars	1964	IT
4	Barry Lyndon	1975	EN



MID	TITLE	YEAR	LANGUAGE
2	Ariel	1988	FI
5	Star Wars	1977	EN
3	Shadow in Paradise	1986	FI
7	Dancer in the Dark	2000	EN
80	Nausicaä	1984	JA
71	Billy Elliot	2000	EN
75	Mars Attacks!	1996	EN
62	2001: A Space Odyssey	1968	EN
19	Metropolis	1927	DE
128	Princess Mononoke	1997	JA
103	Taxi Driver	1976	EN
98	Gladiator	2000	EN
391	A Fistful of dollars	1964	IT
4	Barry Lyndon	1975	EN

Exercice 2 : Index B-tree

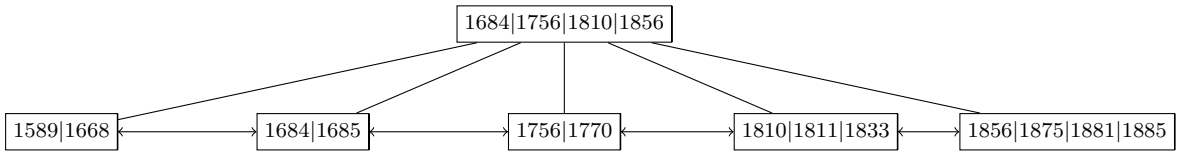
Nom	Date
Monteverdi	1589
Couperin	1668
Rameau	1684
Bach	1685
Ravel	1875
Mozart	1756

Nom	Date
Faure	1856
Beethoven	1770
Chopin	1810
Liszt	1811
Bartok	1881
Brahms	1833
Berg	1885
Bizet	1876

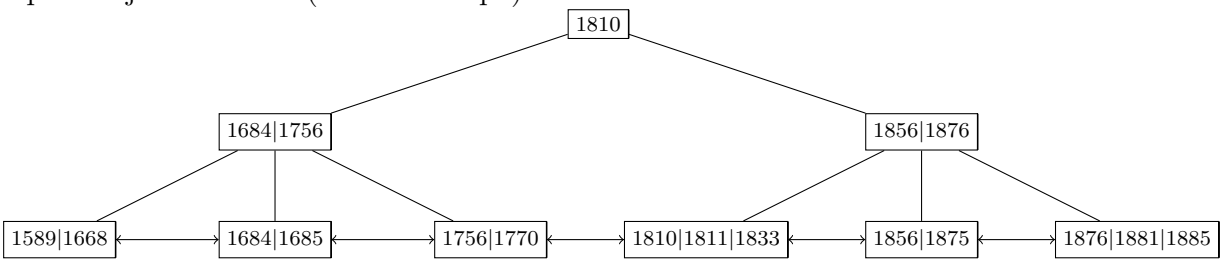
Question 1 : Créer un arbre B+ (d'ordre 2) sur la table pour la clef date par insertions successives dans l'ordre de la table (compter le nombre d'écritures mais aussi de lectures de pages)

Solution

Après l'ajout de 1885 (avant dernière étape) :

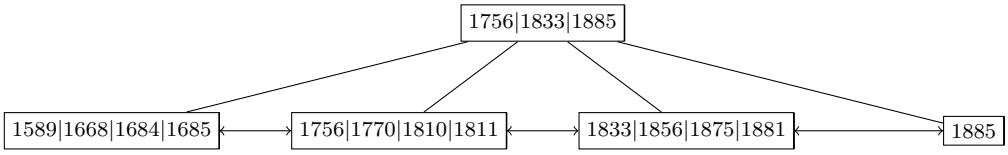


Après l'ajout de 1876 (dernière étape) :



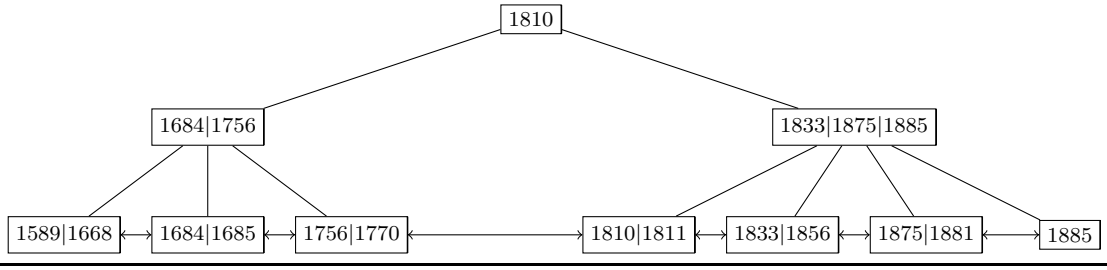
Question 2 : Créer un arbre B+ (d'ordre 2) sur la table de la table précédente pour la clef date par bulkloading (remplissage des feuilles à 100%), comparez le nombre d'opérations en lecture et écriture.

Solution



Question 3 : Même question avec des feuilles remplis à 50% au maximum.

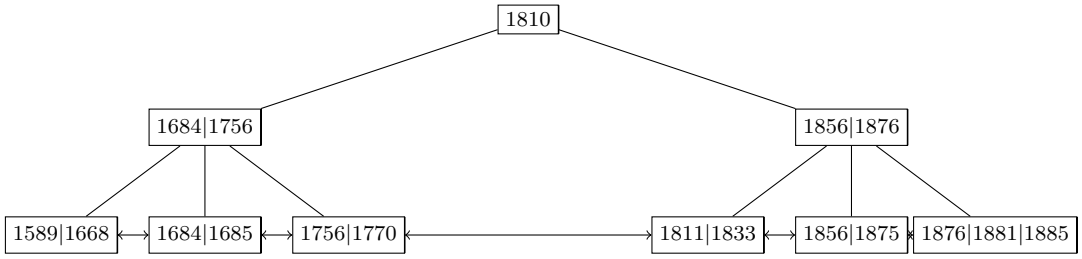
Solution



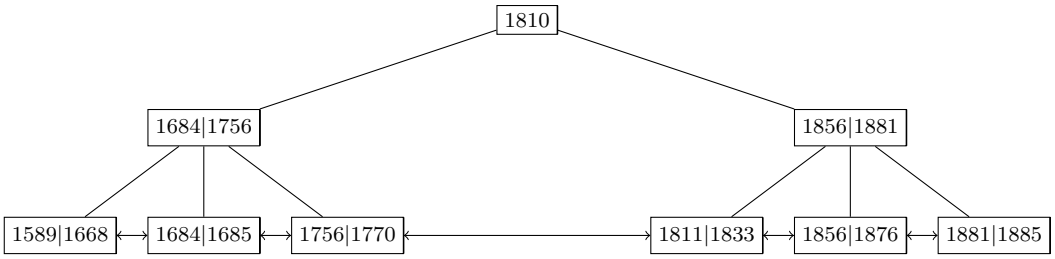
Question 4 : Proposez deux suppressions sur l'arbre produit à la question nř1 de telle sorte qu'une redistribution des nuds de même niveau et une fusion de nuds soient nécessaires

Solution

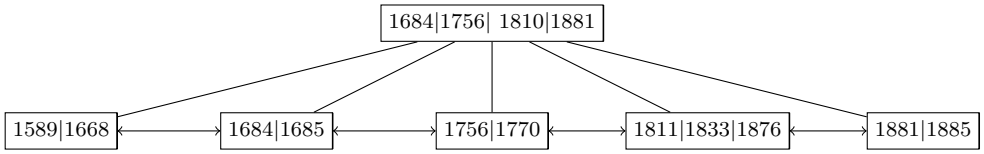
Suppression de 1810 :



Suppression de 1875 (en fait un échange avec le voisin 1876)



Suppression de 1856 (on fusionne des noeuds, ici les deux premiers du sous arbre droit)



Pour aller plus loin, vous pouvez utiliser le script python du cours : `https://colab.research.google.com/drive/1EYxLfDPXPMmme-lTIA2Kf_3bDr7PDrM3?usp=sharing`

Exercice 3 : Index B-tree, taille et compléxité

Supposons quun index B+ (entrées de type 2) a été construit sur un fichier non trié contenant 2.104 enregistrements. Le champ correspondant à la clé de recherche de lindex est une chaîne de caractères de 40 octets. La clé de la recherche est une clé de la relation. On suppose que les pointeurs utilisent (au plus) 10 octets. La taille dune page est de 1000 octets.

Question 1 : Combien de feuilles aura t-on si tout les noeuds de l'arbre sont pleins ?

Solution

On considère que chaque enregistrement contient deux "types" de données :

- Une chaîne de caractère (la valeur de la clef de recherche) → 40 octets
- Un identifiant pointant sur sur les données de l'enregistrement associé à la clef de recherche → 10 octets

Ainsi pour chaque feuille nous avons donc besoin de 50 octets. On sait qu'une page peut contenir jusqu'à 1000 octets et que nous avons 2.104 enregistrements. Une feuille peut contenir $\frac{1000}{50} = 20$ références sur des enregistrements, ainsi nous avons donc $\frac{2104}{20} = 105,2$ feuilles, cela signifie que nous aurons 105 feuilles remplis à 100% et une dernière feuille remplis à 20%, nous aurons au final 106 feuilles.

Question 2 : Combien d'éléments peux contenir un nud interne (au maximum)?

Solution

Un noeud interne contient $2d$ clef de de recherche et $2d+1$ pointeurs au maximum (d étant le degré de l'arbre). Donc il faut que $2d \times 40 + (2d+1) \times 10 \leq 1000$. On peut donc résoudre le système d'équation :

$$100d + 10 \leq 1000$$

donc $d \leq \frac{990}{100} = 9,9$, ainsi $d = 9$ et donc nous avons 18 clefs de recherche et 19 pointeurs au maximum pour les noeuds internes.

Question 3 : Combien de niveaux a l'arbre si les nuds et les feuilles sont remplis aux maximum ?

Solution

Dans le cours, nous avons vu que le nombre de niveaux de l'arbre dépend du nombre de feuilles et du degré avec $h = log_{2d+1}(F)$ où F est le nombre de feuilles. Le nombre de niveaux vaut la hauteur de l'arbre où l'on ajoute 1 pour la racine. Ici, on aura donc

$$h = \lceil log_{19}(106) \rceil$$

$$log_{19}(19) < log_{19}(106) < log_{19}(19^2)$$

$$1 < log_{19}106 < 2$$

Donc l'arbre est de hauteur 2 et comporte 3 étages/niveaux

Question 4 : Combien y a t-il de nuds à chaque niveau si les nuds et les feuilles sont remplis aux maximum ?

Solution

- Niveau 3 : 106 noeuds (feuilles)
- Niveau 2 : $5 < \frac{106}{19} < 6$ donc 6 noeuds (intermédiaire)
- Niveau 1 : $6 < 19$ la racine comporte 1 page

Question 5 : Supposons que lon utilise une passe de compression pour diminuer la taille des clés de recherche. La taille dune clé compressée est de 10 octets. Combien de niveaux aurait alors lindex ?

Solution

$\frac{990}{40} = 24,75$ donc $d = 24$ donc les noeuds internes peuvent contenir 49 données. Le nombre de feuilles est 43 (car $\frac{2104}{50} = 42,08$) . Ainsi $0 < log_{49}(43) < 1$, donc de hauteur 1 (2 niveaux) et pour chaque niveau de l'arbre on a :

- Niveau 1 : 1
- Niveau 2 : 43

Question 6 : Combien de de niveaux aurait lindex, sans compression, en supposant un taux de remplissage des pages de 60% ?

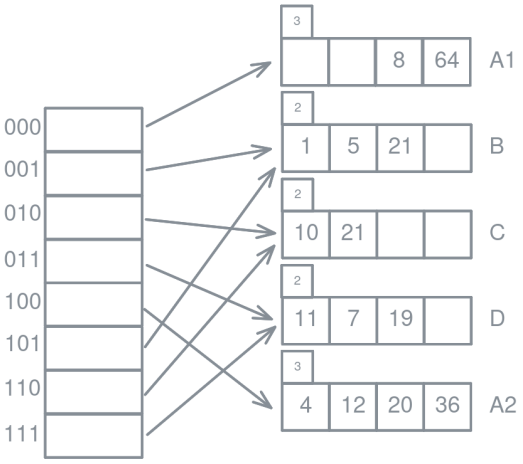
Solution

Si $0.6 * 1000 = 600$ donc $600/50 = 11$ $\frac{2104}{11} = 191.2$ donc 192 feuilles, puis au niveau suivant on considère aussi ce taux de remplissage
$192/11 = 11.61$ puis $12/11$ donc 2 noeuds

- Niveau 4 : 192 (feuilles)
- Niveau 3 : 12
- Niveau 2 : 2
- Niveau 1 : 1 (racine)

Exercice 4 : Index haché

On considère dans cette exercice un index haché dynamique comme dans la figure ci-contre.



Question 1 : Si il n'y a pas eu de suppressions dans l'index, est-il possible que la dernière opération soit un partitionnement ? Pourquoi ?

Solution

Non car A1 et A2 on à eux deux 6 éléments (la taille d'une page est de 4). Sachant qu'aucune suppression n'est possible, après un partitionnement, la somme du nombre d'éléments partitionnés vaut 5 (car les pages sont de tailles 4 le partitionnement a lieu quand l'on ajoute un élément donc 5). Ici le seul partitionnement a été effectué sur A1, A2. Le nombre d'éléments dans A1 ajouté aux nombre d'éléments de A2 vaut 6 donc pas de partitionnement possible à la dernière étape.

Question 2 : Montrez l'index après l'insertion de 68 (1000100)

Solution

partitionnement de A2 (utilisation du 4 bit): 4:0100, 12:1100, 20 : 10100, 36: 100100 donc

- **A21** 4, 20, 36, 68
- **A22** 12

Exercice 5 : Sélection

Soit le schéma Film(IdFilm, Titre, Année, Genre, Résumé, IdMES, CodePays).

Question 1 : On suppose quun index de type 2 trié, B+ groupant et dense de clé de recherche Année a été créé sur Film. Expliquez comment le SGBD peut évaluer la requête

```
1 SELECT count(*) FROM Film GROUP BY Année ;
```

Solution

Il faudra pour compter tous les éléments groupé par années accéder à toutes les champs années de la relation. Ainsi un parcours (scan) de l'index est suffisant.

Question 2 : La même question lorsque l'index est non-groupant ?

Solution

Il faudra pour compter tous les éléments groupé par années accéder à toutes les champs années de la relation. Ainsi un parcours (scan) de l'index est suffisant.

Question 3 : Quel index doit être créé pour pouvoir évaluer la requête ci-dessous par une consultation de lindex seulement.

```
1 SELECT Année count(*) AS NbFilm FROM Film  WHERE Année > 1984 AND année < 1999
    GROUP BY Année ;
```

Solution

Un index trié sur l'année seulement

Question 4 : Discuter de lévaluation de la requête ci-dessous en supposant quil existe deux index sur Film, tous les deux ont des entrées de type 2, lune des clef de recherche est Année et lautre Genre.

```
1 SELECT Titre FROM Film WHERE Année=2017 AND Genre=Comédie
```

Solution

On peut supposer que le pourcentage de films ayant pour année 2017 est relativement faible par rapport au nombre d'entrées de la base. On utilisera donc l'index de type 2. Dérouler l'index sur le genre n'est pas forcement nécessaire si le nombre de page est restreint. Pour sélectionner les films/titre on pourra utiliser la méthode de bitmap index scan puis de bitmap heap scan pour charger les pages cibles.

Exercice 6 : Choix de l'index

Dans cet exercice on considère la relation suivante :

Restaurants(rid : **integer**, nom : **string**, note_moyenne : **float**)

Les données de la relation sont stockées dans un fichier sous forme d'un tas. On suppose qu'une page peut contenir 8000 octets de données, les entiers et les flottants occupent un espace 4 octets chacun, une chaîne de caractère est limité à 192 octets et un pointeur (ou rid) occupe 6 octets. La relation contient 400 000 d'enregistrements avec des valeurs comprises entre 0 à 399 999. De plus on supposera que les notes sont uniformément réparties de 1 à 20.

Question 1 : Combien d'enregistrement peut contenir une page ?

Solution

Un enregistrement occupe 200 octets (4 + 4 + 192), donc $\frac{8000}{200} = 40$ enregistrements par page

Question 2 : Estimer la sélectivité de :

1. $\sigma_{restaurant.rid<50.000}(Restaurants)$
2. $\sigma_{restaurant.rid=50.000}(Restaurants)$
3. $\sigma_{restaurant.rid>50.000 \wedge restaurant.rid<50.020}(Restaurants)$
4. $\sigma_{restaurant.rid\neq 50.000}(Restaurants)$
5. $\sigma_{restaurant.moyenne=20}(Restaurants)$
6. $\sigma_{restaurant.moyenne>15}(Restaurants)$

Solution

$$\sigma_{\mathbf{restaurant.rid<50.000}}(\mathbf{Restaurants})$$

Pour la requête ci-dessus $\frac{50000}{400000} = \frac{1}{8}$ 0.125% des données seront sélectionnées

$$\sigma_{\mathbf{restaurant.rid=50.000}}(\mathbf{Restaurants})$$

Pour la requête ci-dessus$\frac{1}{400000}$ donc $\frac{1}{4000}$%des données seront sélectionnées

$$\text{sigma}_{restaurant.rid>50.000 \wedge restaurant.rid<50.020}(Restaurants)$$

Pour la requête ci-dessous $\frac{20}{400000} = \frac{1}{20000}$ donc $\frac{1}{200}$%des données seront sélectionnées

$$\sigma_{\mathbf{restaurant.rid\neq 50.000}}(\mathbf{Restaurants})$$

Pour la requête ci-dessous $\frac{399999}{400000}$ donc 99.9997%des données seront sélectionnées

$$\sigma_{\mathbf{restaurant.moyenne=20}}(\mathbf{Restaurants})$$

D'après l'hypothèse d'uniformité sur la colonne moyenne, 1% des données seront sélectionnées (soit à peu près 20000 enregistrements)

$$\sigma_{\mathbf{restaurant.moyenne>15}}(\mathbf{Restaurants})$$

D'après l'hypothèse d'uniformité sur la colonne moyenne 25% des données seront sélectionnées (soit à peu près 100000 enregistrements)

Question 3 : Estimez le coût maximum (E/S sur les pages) des sélections précédentes pour une sélection utilisant un parcours du fichier

Solution

Dans tous les cas celui ci est du nombre de page du fichier soit 400 000/40 donc 10 000 pages ouvertes

Question 4 : En considérant les index de type 2 suivants :

- Un index groupant B+ de hauteur 3 sur rid, les feuilles contiennent en moyenne 500 entrées
- Un index non-groupant B+ de hauteur 3 sur rid, les feuilles contiennent en moyenne 500 entrées
- Un index haché sur rid
- Un index non-groupant B+ de hauteur 3 sur la note_moyenne, les feuilles contiennent en moyenne 500 entrées

Estimez le coût maximum (E/S) des sélections précédentes pour chaque index

Solution

$$\sigma_{\mathbf{restaurant.rid<50.000}}(\mathbf{Restaurants})$$

- **Index Groupant B+** : Le coût est de 3 pages de l'index pour la recherche mais aussi de $\frac{50000}{500} = 100$ feuilles de l'index, par ailleurs le nombre de page dans le fichier satisfaisant la condition est de $\frac{50000}{40} = 1250$ pages. En utilisant cet index groupant alors le coût sera approximativement de 1353 pages.
- **Index Non Groupant B+** : Le coût est de 3 pages de l'index pour la recherche mais aussi de $\frac{50000}{500} = 100$ feuilles de l'index. Comme l'index n'est pas groupant, dans le pire cas 50 000 pages sont ouvertes donc 50 104 pages ouvertes (ce qui est supérieur au parcours). Notons que nous pouvons utiliser la méthode Bitscan index, dans ce cas le nombre de lectures de pages dans le fichier n'excedera pas le nombre de page du fichier, ce qui tout de même équivaudrai à 10 104 pages ouvertes.
- **Index Haché** : Pas efficace

$$\sigma_{\mathbf{restaurant.rid=50.000}}(\mathbf{Restaurants})$$

- **Index Groupant B+** : Le coût est de 4 pages de l'index et 1 page du fichier
- **Index Non Groupant B+** : Le coût est de 4 pages de l'index et 1 page du fichier
- **Index Haché** : Le coût est de 1 page pour le bucket et 1 page du fichier

$$\text{sigma}_{restaurant.rid>50.000 \wedge restaurant.rid<50.020}(Restaurants)$$

- **Index Groupant B+** : Le coût est de 3 (hauteur de l'arbre) pages de l'index pour la recherche mais aussi de $\frac{20}{500} < 1$ (avec débordement possible donc 2) feuilles de l'index, par ailleurs le nombre de page dans le fichier satisfaisant la condition est de $\frac{20}{40} = 0.5$ pages (2 avec débordement). En utilisant cet index groupant alors le coût sera approximativement de 8 pages maximum.
- **Index Non Groupant B+** : Le coût est de 3 pages de l'index pour la recherche mais aussi de $\frac{20}{500} < 1$ feuilles de l'index. Comme l'index n'est pas groupant, dans le pire cas 21 (20 + débordement) pages sont ouvertes donc 27 pages ouvertes.
- **Index Haché** : Pas efficace

Les autres corrections sont similaires

Question 5 : Quel serait l'impact sur le coût dexécution des requêtes précédentes si les moyennes étaient réparties de la manière suivante :

- 5% des moyennes sont entre 0 et 5
- 90 % des moyennes entre 5 et 15
- 5 % des moyennes entre 15 et 20

On suppose aussi que sur chacun des trois intervalles les moyennes sont uniformément réparties.

Solution

Le coût des requêtes 5 et 6 seraient plus faible (calculer avec les nouveaux intervalles)