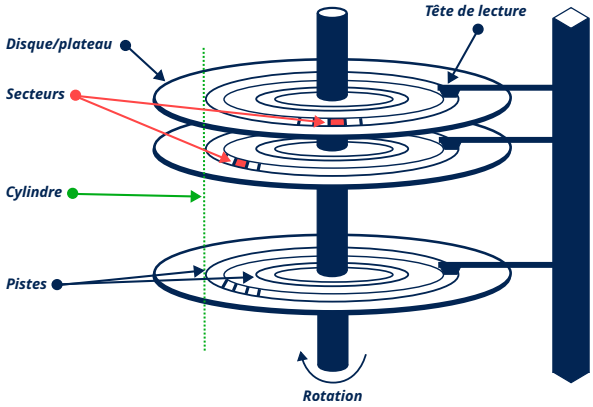
Exercice n°1 : Gestionnaire de mémoire

Dans cette première partie, on considéra comme média de stockage un disque dur à plateaux. Celui-ci est constitué de différents éléments : les **secteurs** correspondant à la plus petite unité de lecture, les secteurs sont contenus dans des **blocs** correspondant à une suite consécutive de secteurs, enfin ces blocs sont ensuite arrangés en cercle concentrique sur les **disques ou plateaux**, ces espaces de données sont appelés des **pistes**.



Question n°1 : Quels sont les sous-modules du gestionnaire de mémoire ?

Solution

Celui-ci est composé de 2 sous-gestionnaires communiquants entre eux :

Le gestionnaire de disque. L'objectif étant de placer les données de manière optimale sur le disque pour accélérer la lecture des pages.

Le gestionnaire de cache. L'objectif du cache est de minimiser le nombre d'écritures/lectures (R/W) sur le stockage persistant. Plusieurs stratégies sont mises en place conserver les pages utiles en mémoire vive, comme les algorithmes LRU ou Clock.

Question n°2 : Expliquez les différentes opérations pour la lecture d'un secteur sur un stockage de type disque à plateaux (HDD).

Solution

1. Positionnement de la tête sur la piste (**recherche**)
2. Rotations
   - Rotation jusqu'au début du bloc (**rotation**)
   - Rotation de lecture sur les blocs (**transfert**)

Question n°3 : Considérons un disque où la taille des secteurs vaut 1024 octets, chaque piste contient 50 secteurs et chacune des surfaces à une capacité de 102400 octets, enfin le HDD contient deux plateaux (double face).

1. Quelle est la capacité d'une piste ? du disque ?
2. Combien y-a t-il de secteurs sur une surface ?

Solution

L'objectif ici n'est pas tant de savoir si l'on sait calculer mais de savoir si les différentes parties d'un disque sont comprises (ceci est un exemple jouet, il n'est pas représentatif de la capacité des disques durs)

- La capacité d'une piste vaut $Taille\_secteur * nb\_secteur = 1024 * 50 = 51200$ octets
- La capacité d'une surface vaut $Taille\_surface * nb\_surface = 102400 * 4 = 409600$ octets
- On peut calculer le nombre de pistes $\frac{Taille\_surface}{Taille\_secteur * nb\_secteur} = 2$ le nombre de secteurs total est alors $nb\_pistes * nb\_secteur = 100$

Question n°4 : Dans le cadre du gestionnaire de cache, quel est l'objectif du stockage du

`dirty_bit` pour chaque page présente dans les cellules du cache ? Que se passe t-il si l'on chercher à remplacer une cellule contenant une page qui a un `dirty_bit` égal à 1

Solution

Le dirty_bit correspond à un bit de stockage valant 1 si des modifications ont été effectuées sur la page associée dans le cache. Dans le cas d'une suppression ou d'un remplacement d'une page ayant une $dirty\_bit = 1$, avant le remplacement on fait une demande d'écriture de la page du cache vers le stockage persistent (gestionnaire de disques)

Question n°5 : Qu'est-ce que le `pin_count` ? Quand est-il incrémenté ? Décrémenté ?

Solution

Le `pin_count` stocke le nombre de processus utilisant la page, sa valeur est incrémenté de 1 lorsque un processus demande l'accès à une page, sa valeur est décrémentée lorsque un processus libère la page

Question n°6 : On considère la stratégie de remplacement FIFO, on considère dans cette exemple la lecture des pages seulement (qui sont automatiquement libérées ensuite). Quel est le nombre de lecture sur le disque de la séquences de demande de pages suivantes (pour un cache de 3 cellules) :

$$p_1 \to p_2 \to p_3 \to p_1 \to p_2 \to p_4 \to p_1 \to p_2 \to p_5 \to p_6$$

Solution

Cellule/page	p_1	p_2	p_3	p_1	p_2	p_4	p_1	p_2	p_5	p_6
0	p_1			_		p_4			p_5	
1		p_2			_		p_1			p_6
2			p_3					p_2		

8 lectures sur le disque !!

Question n°7 : Même question avec un remplacement en utilisant la stratégie LRU (la page la moins récemment utilisée)

Solution

Cellule/page	p_1	p_2	p_3	p_1	p_2	p_4	p_1	p_2	p_5	p_6
0	p_1			+			+			p_6
1		p_2			+			+		
2			p_3			p_4			p_5	

6 lectures sur le disque !!

Exercice n°2 : Disque et temps d'accès

On considère un disque ayant les caractéristiques suivantes : chaque face contient 2 000 pistes, chaque piste contient 50 blocs, et il a 5 plateaux double face. On suppose que le temps moyen de recherche d'une piste (mouvement du bras du disque) est de 10 msec et que la taille d'un bloc a été fixée à 1024 octets, considérons un fichier de 100 000 enregistrements de 100 octets chacun devant être stocké sur le disque (le chevauchement d'un enregistrement sur plusieurs blocs n'est

pas autorisé).

Question n°1 : Combien d'enregistrements peuvent être stockés dans un bloc ?

Solution

Sur un bloc on peut stocker 1024 octets un enregistrement occupe 100 octets, $1024 = 10 * 100 + 24$ donc un bloc peut contenir au plus 10 enregistrements

Question n°2 : Combien de blocs sont nécessaires pour le stockage du fichier ?

Solution

il y a 100 000 enregistrements et 10 enregistrements par page/block il faut donc 10 000 blocs

Question n°3 : Si le fichier est placé séquentiellement sur le disque, combien de cylindres sont nécessaires pour le stockage du fichier ? Combien de temps est nécessaire pour lire le fichier ?

Solution

Calculons la taille en bloc d'une piste est de 50 blocs, $10\,000 = 50*200+0$, donc 200 pistes sont nécessaires, ainsi il faudra 200 recherches (10ms) mais aussi faire un tour complet à chaque fois ($\frac{1}{5400}$ secondes, donc $\approx .1$ ms). DOnc le temps total est $200 \times 10 + 200 \times 0.1 = 2000 + 20$ms

Question n°4 : Même question en supposant que le disque puisse lire en parallèle avec ses 10 têtes (on suppose que les données sont placées sur les mêmes cylindres) ?

Solution

Il suffit de diviser par 10, donc 201 ms.

Exercice n°3 : Fichiers et pages

On suppose qu'une relation est stockée sur 1000 pages et que chaque page contient 10 enregistrements.

Question n°1 : On souhaite accéder à l'enregistrement ayant pour $v\_id = 1002$. Si toutes les pages contiennent 10 enregistrements, sur quelle page peut-on retrouver l'enregistrement si le fichier trié sur v_id ? Si le fichier est un tas ?

Solution

- Pour un fichier trié il faut au maximum $\lceil \log_2(1000) \rceil$ ouvertures de pages soit 10 pages
- Pour le tas, dans le pire cas il faut lire l'intégralité du fichier donc 1000 pages

Question n°2 : On considère que une page p_{1002} ayant comme adresse de base dans le cache $0x1020$, qu'une page a une entête de 100 octets et que l'on cherche a accéder au 3éme enregistrement (1 pointeur vaut 2 octets) de la page. Donnez l'adresse du pointeur sur le début de l'enregistrement.

Solution

Pour accéder au pointeur sur l'enregistrement, il faut ajouter à l'adresse de la page l'entête ainsi que les pointeurs précédents le pointeur cible. Avant cela convertissons en hexadécimal la taille de l'entête :

- 100 octets s'écrit 64 en hexadécimal (6 * 16 + 4)

L'adresse du pointeur est donc $1020 + 64 + 2 \times 2 = 1088$

Question n°3 : On suppose que le fichier est trié, déterminez combien de lecture et d'écriture de blocs sont nécessaires pour l'ajout d'un enregistrement étant placé en 501éme position (selon l'ordre des valeurs triés).

Solution

Il faut d'une part accéder à l'enregistrement (10 ouvertures max) mais aussi décaler les enregistrements suivants dans le fichier.

- 10 pages pour retrouver l'enregistrement
- l'enregistrement est sur la 51 éme page, il faut donc modifier 949 pages (mais les lire aussi)

Donc $\approx 949 \times 2 + 10 = 1908$ I/O

Question n°4 : Même question mais cette fois-ci pour un tas.

Solution

Il n'est pas nécessaire de rechercher un enregistrement, il faudra en revanche écrire/modifier une page

Exercice n°4 : Introduction aux index

Question n°1 : Quelles sont les alternatives pour les entrées d'un index ?

Solution

- Une entrée de données k* est un enregistrement de données (avec la valeur de la clé de recherche k).
- Une entrée de données est une paire (k, rid) , où rid est l'identifiant d'un enregistrement de données avec la valeur de clé de recherche k.
- Une entrée de données est une paire (k, rid-list), où rid-list est une liste d'identifiants d'enregistrements de données ayant la valeur de clé de recherche k.

Question n°2 : Déterminer la structure de fichier la plus adaptée parmi le tas (*heap file*) ou le fichier trié (*sorted file*) si les opérations les plus fréquentes sont :

1. L'ajout de nouveaux enregistrements
2. Récupérer tout les éléments d'une instance de relation
3. La recherche d'un intervalle de valeurs

Expliquez pourquoi et donnez le nombre de chargement de page moyen approximatif (on considérera N pages dans le fichier) ?

Solution

1. Les fichiers en forme de tas sont plus efficace, car il n'est pas nécessaire de modifier la structure ou d'effectuer un décalage comme sur les fichiers triés. Ainsi pour ajouter un enregistrements, on sélectionne la première page avec un espace libre suffisant (sinon on créera une nouvelle page) puis on écrit l'enregistrement (donc accès à la dernière page + écriture de l'enregistrement page). Pour le tas : $O(1)$ 1 page Pour le trie : $O(N)$ car décalage à chaque insertion
2. Les deux types de fichiers sont aussi performant, il faudra dans tous les cas scanner l'intégralité du fichier $O(N)$
3. Le fichier trié est plus efficace, il implique une recherche $O(log_2(N))$ (par dichotomie) puis un parcours où le nombre d'enregistrement parcourus est égale au nombre d'éléments dans l'intervalle. Pour le Tas il faudra faire un scanner l'intégralité du fichier donc $O(N)$ en terme de chargement de pages

Question n°3 : Quelle est la différence entre un index groupant et un index non groupant ? Est-il possible d'avoir plusieurs indexes groupants pour un fichier de données ?

Solution

Dans un index groupant, les données du fichier sont dans le même ordre que l'index (sur chaque page) à l'inverse dans un index non-groupant les données. Il n'est pas possible de créer plus d'un index groupant par fichier d'index, puisque l'index suit le même ordre que le fichier de données.

Question n°4 : On dispose de l'instance de relation suivante :

NID	NOM	LOGIN	21	Moyenne
53831	Madrian	madayan@etu.fr	21	11.5
53832	Guads	guads@etu.fr	21	13
53833	Jean	jean@etu.fr	24	15.7
53838	Paul	Paul@etu.fr	22	13.4
53839	Jacques	jacques@etu.fr	22	10.4
53855	Jacques	jacques2@etu.fr	19	19.4

On obtient par un scan de la relation Etudiant la table précédente. Une page peut contenir au plus 3 enregistrements. Le rid d'un enregistrement est un couple numéro de page, numéro d'emplacement. Listez les entrées des index pour les caractéristiques suivantes :

1. Un index dense sur NID de type 2
2. Un index dense sur age de type 1
3. Un index dense sur age de type 2
4. Un index dense sur age de type 3
5. Un index non-dense de type 2 sur NID
6. Un index non-dense sur moyenne de type 2
7. Un index dense sur Nom de type 3

Solution

Un index dense sur NID de type 2 : même ordre que le fichier qui est déjà trié (53831, (1,1)) , (53832, (1,2)),(53833, (1,3)), (53838, (2,1)), (53838, (2,2)),(53838, (2,3))

Un index dense sur age de type 1 : Comme l'index de type 1 considère les enregistrements comme entrées d'index, il n'est pas possible de réorganiser l'ordre des données.

Un index dense sur age de type 2 : Possible (19, (2,3)) , (21, (1,1)),(21, (1,2)), (22, (2,1)), (22, (2,2)),(24, (1,3))

Un index dense sur age de type 2 : Possible (19, (2,3)) , (21, (1,1), (1,2)) (22, (2,1), (2,2)), (24, (1,3))

Un index non-dense de type 2 sur NID les entrées de l'index sont (53838, (1,1)), (53838, (2,1))

Un index non-dense sur moyenne de type 2 Les données ne sont pas triés selon la moyenne donc impossible d'envisager un index non dense (propriété groupante nécéssaire)

Un index dense sur Nom de type 3 (Guads, (2,1)), (Jacques, (2,1),(2,2)), (Jean, (1,3)), (Madrian, (1,1)), (Paul, (2,1)

Question n°5 : On considère l'instance de relation précédente, on dispose d'un index trié sur la Moyenne (de type 2), en parcourant l'index pour les moyennes ≤ 16. Quelles sont les pages successivement chargées ? Expliquez les risques liés aux indexes non-groupant en terme de performance.

Solution

Page 2 (10.4), Page 1 (11.5), Page 1 (13), Page 2 (13.4), Page 1 (15.7) . Le risque ici est le coût en lecture, si l'on charge alternativement la page 1,2 alors on effectue 4 chargements (je ne compte pas lorsque l'on charge la même page deux fois de suite).

Question n°6 : Expliquez le principe du "bitmap index scan" et du "bitmap Heap Scan".En considérant le tableau de la question 5 et un index de type 2 trié, combien de lectures de pages sont effectuées au maximum ?

Solution

Dessiner ! ! Le bitmap index scan permet en parcourant l'index de retenir dans une bitmap les pages qui seront chargées. Ainsi les chargements multiples de pages seront évités. En revanche on perd l'information sur les enregistrements forçant ainsi la lecture des pages chargés (des enregistrements). Dans la question précédente, seulement deux lectures de pages auraient été au total nécessaire.