Parcours DataScale
Architecture des Gestionnaires de Données

Examen session 2 14 mars 2017

Durée : 2h30. Tous (vos) documents autorisés.
Soyez précis dans vos réponses, toute réponse non justifiée sera considérée comme fausse.
Utilisez deux feuilles séparées pour les Parties 1 et 2.

PARTIE 1 (14 points) : Mécanismes internes des Gestionnaires de Données

EXERCICE 1 (5 points) : Transactions

Supposons une exécution parallèle de 3 transactions T_1, T_2 et T_3 produisant, sans contrôle de concurrence, la séquence d'opérations suivante :

1	2	3	4	5	6	7	8
$R_2(b)$	$W_1(a)$	$R_3(a)$	$R_3(c)$	$W_1(b)$	$W_3(c)$	$R_2(c)$	$R_2(b)$

$R_i(x)$ (resp. $W_i(x)$) représente une opération de lecture (resp. écriture) par la transaction T_i de l'objet x.

Question 1.1 : Cette exécution est-elle sérialisable ? Quelle que soit la réponse, donnez-en la preuve.

Question 1.2 : Quelle est l'histoire produite par un protocole d'estampillage ? On suppose pour cela que les transactions sont lancées dans l'ordre T1, T2, T3.

Question 1.3 : Quelle est l'histoire produite par un protocole de certification ? On suppose pour cela qu'une transaction se termine juste après sa dernière opération.

Question 1.4 : Quelle est l'histoire produite par un protocole de verrouillage à 2 phases ?

Question 1.5 : Modifier cette exécution pour y introduire un verrou mortel.

Pour cet exercice, indiquer simplement les actions intéressantes et leur étape (de 1 à 8 ou ultérieure).

EXERCICE 2 (5 points) : Indexation

Soit la table `Produits(IdProduit, NomFournisseur, Prix, QuantitéStock)` contenant 100.000 tuples stockés dans 1000 pages disque de 4KB chacune. On souhaite indexer cette table sur l'attribut `NomFournisseur` variant uniformément sur un domaine de 80 valeurs distinctes.

(a) Quel est le meilleur choix entre un index secondaire de type B-Tree ou Bitmap et pourquoi ?

(b) Quelle serait la taille, en nombre de pages, de l'index bitmap ?

(c) On souhaite également indexer en bitmap l'attribut `QuantitéStock` pour faire des requêtes de type Range (>v1, <v1, [v1,v2]) en supposant que la quantité maximale en stock ne peut dépasser 20 et que les données sont équi-distribuées. Est-ce possible ? Si oui, combien faudrait-il d'entrée/sorties disque pour répondre à la requête `(NomFournisseur = Thomson) and (QuantitéStock > 15)`? Combien y-a-t-il de tuples résultats à cette requête ?

EXERCICE 3 (4 points) : Quiz

La réponse à chacune des questions ci-dessous devra être justifiée avec précision.

Question 3.1 : (a) dans le mécanisme de *Database Cracking* mis en œuvre dans MonetDB, le gain obtenu par rapport à un scan séquentiel augmente au fur et à mesure que l'on exécute des requêtes ; (b) ce gain augmente également au fur et à mesure par rapport à un mécanisme d'indexation classique (i.e., création initiale d'un index complet). Pour chacune de ces affirmations, dire si elle est vraie ou fausse.

Question 3.2 : On veut indexer une même table T sur 3 attributs A, B et C avec un seul index multi-attributs afin de pouvoir accélérer des requêtes qui portent sur une combinaison quelconque de ces attributs. Vaut-il mieux utiliser un B^+-Tree multi-attributs ou un hachage multi-attributs ?

Question 3.3 : Dans un contexte In-Memory Database Systems (IMDS), quelle optimisation est apportée par un modèle de stockage et d'indexation Graph-based par rapport à un modèle Pointer-based ?

Question 3.4 : Dans un contexte IoT, l'architecture hardware des objets intelligents repose principalement sur un microcontrôleur relié à de la mémoire Flash. Parmi Hachage, B-Tree, Bloom filter et index bitmap, lesquels vous semblent adaptés et lesquels ne le sont pas et pourquoi ?

PARTIE 2 (6 points) : Systèmes de gestion de données dans le cloud

Question 2.1 (4 points): Partitionnement statique

Considérons une application de type e-commerce gérant principalement trois collections de données :

- collection de 500.000 clients avec pour chaque client son profil (de l'ordre de 2 K octets par client) ;
- collection de 200.000 articles à vendre classés en 50 catégories, avec pour chaque article une description détaillée, un prix et une quantité en stock (de l'ordre de 5 K octets par article) ;
- collection de 2.500.000 commandes, en moyenne 5 par client, avec un minimum de 1 et un maximum de 100 (de l'ordre de 1K Octet par commande).

Sur cette base de données, on exécute principalement les transactions suivantes :

- ajouter un client avec son profil ;
- passer une commande (associe un client, un article et une quantité) ;
- top100 des articles les plus vendus ;
- ajouter un article dans une catégorie (avec description, prix et stock).

On veut mettre en place cette application dans le cloud en utilisant un système de gestion de données de type clé-valeur. On suppose que les données sont distribuées sur trois serveurs (A, B et C). Le modèle de fragmentation M1 est défini comme suit :

M1 : Clients est fragmenté horizontalement sur les 3 serveurs en fonction des valeurs de la clé (environ 1/3 des clients sur chaque serveur). Commandes est fragmenté par voisinage de Clients (toutes les commandes d'un client sont sur le même serveur que le client). Articles est répliqué intégralement sur les 3 serveurs.

Le modèle de fragmentation M2 est défini comme suit :

M2 : Articles est fragmenté horizontalement sur chaque serveur en fonction de la clé (environ 1/3 des articles sur chaque serveur). Commandes est fragmenté par voisinage de Articles (toutes les commandes d'un article sont sur le même serveur que l'article). Clients est répliqué intégralement sur les 3 serveurs.

a) Donner pour chaque modèle de fragmentation, le volume total de données stockées sur chaque serveur.
b) Donner pour chacune des quatre transactions ci-dessus et chaque modèle de fragmentation, le volume de données à lire et à écrire.
c) Expliquer pour chaque modèle de fragmentation et chaque transaction si l'exécution manipulera des données présentes sur plusieurs serveurs ou non. Plus précisément, vous indiquerez s'il y a besoin de mises à jour distribuées ou si le modèle de partitionnement garantit que la transaction ne modifie pas des données sur plusieurs serveurs.
d) Quel est le modèle de fragmentation à préconiser selon vous entre M1 et M2 et pourquoi.

Question 2.2 (2 points) : Multi-tenancy

Supposons que l'on ait deux applications A1 et A2 utilisant chacune leur propre schéma de bases de données (S1 et S2). Si on veut gérer ces deux applications dans le modèle « shared table » de multitenancy, faut il modifier leur schéma de données. Si oui, quelle opération faut-il faire sur les schémas S1 et S2 ? Y-a –t-il des modifications à faire dans le code des requêtes portant sur S1 et S2 ?