NI401 - Administration des Bases de Données Réparties

examen 1ère session - 21 janvier 2013

Seuls les documents de cours sont autorisés – **Durée : 2h**.

La **qualité et la précision de la rédaction** seront prises en compte. **Argumenter et justifier** chaque réponse (une réponse non justifiée est insuffisante). Préciser toutes les hypothèses supplémentaires sur lesquelles repose votre raisonnement. Ecrire à l'encre bleue ou noire.

Exercice : Système de gestion de données de type NOSQL 20 pts

On considère la base de données d'une application pour gérer une sélection d'articles de presse. Un utilisateur choisit des conseillers. Il peut lui-même être choisi comme conseiller par d'autres utilisateurs. Un utilisateur demande à lire une sélection d'articles qu'il peut ensuite noter. Les notes que les utilisateurs attribuent, servent à filtrer les articles à inclure dans une sélection.

Personne (num, nom, prénom, ville, pays) // un utilisateur avec son adresse.

Conseiller (pers, conseiller, jour) // les conseillers de la personne *pers*. Les attributs *pers* et *conseiller* sont des numéros de personne. L'attribut *jour* indique la date à laquelle une personne a choisi un conseiller.

Article (num, titre, desc, catégorie) // l'attribut *desc* est le contenu textuel de l'article. Un article est dans une seule catégorie. Il y a 5000 catégories distinctes.

Note (pers, article, note, jour) // une personne attribue une note à un article à une certaine date (attribut jour). L'attribut *pers* est un numéro de personne. L'attribut *article* est un numéro d'article.

L'attribut jour dans Conseiller et Note est une valeur entière dans [1, 3000].

Les requêtes sont les suivantes :

R1(p) : retrouver le numéro et le titre des articles notés par un conseiller de *p*, et que *p* n'a pas notés.

R2(p) : lister les couples (a, m) tels que *a* est un numéro d'article noté par les conseillers de *p*, et *m* est la note moyenne de l'article. Trier le résultat par moyenne décroissante. Rmq : la note moyenne de *a* est calculée à partir des notes attribuées par les conseillers de *p*.

R3 : Afficher tous les couples (A, B) des numéros d'utilisateurs qui se conseillent mutuellement. Autrement dit, B est un conseiller de A, et A est un conseiller de B.

Les opérations de modification sont les suivantes :

AjoutN(p, a, n, j) : L'utilisateur p attribue la note n à l'article a.

AjoutC(p, c, j) : L'utilisateur p ajoute le conseiller c.

SupprC(p, c) : Supprimer le conseiller c de l'utilisateur p.

Rmq : On suppose que tous les utilisateurs et les articles existent initialement.

On souhaite utiliser SimpleDB pour gérer les données. La syntaxe simplifiée pour l'accès à SimpleDB est la suivante.

Syntaxe pour invoquer SimpleDB et poser une requête :

R = **requete**(*req*).

Le résultat *R* est un ensemble d'items qui satisfont la requete *req*. La syntaxe de *req* est

select distinct liste d'attributs **from** domaine // un seul domaine

where attribut1 = valeur **AND** attribut2=valeur etc…

[**order by** attribut]

Les autres opérateurs de comparaison possibles dans un where sont : > valeur, < valeur, **in**(liste de valeurs). On suppose qu'il n'y a pas de limite sur la taille de l'expression formant la requête. En particulier, une requête peut contenir un prédicat *in(liste)* dont la liste contient un nombre quelconque de valeurs. On peut aussi ajouter **NOT** devant un prédicat et relier deux prédicats par **OR**.

Il n'y a **pas** de requêtes imbriquées ni de clause group by.

Syntaxe pour itérer sur le résultat d'une requête ou sur un ensemble.

Pour chaque r **dans** R
…
Fin pour

Syntaxe pour ajouter/remplacer un item : **put**(domaine, nom de l'item, liste de paires (nom d'attribut, valeur))

Syntaxe pour ajouter un attribut dans un item : **put**(domaine, item , ajoute(nom, valeur), …).

Syntaxe pour remplacer un attribut déjà existant dans un item : **put**(domaine, item , remplace(nom, valeur), …).

Syntaxe pour supprimer des attributs : **delete**(domaine, item, liste d'attributs). Si tous les attributs sont supprimés, alors l'item est supprimé.

Syntaxe pour supprimer un item : **delete**(domaine, item).

Question 1 : On définit un domaine par relation. Le **nom** d'un item (itemname) est *unique* dans son domaine.

- Le domaine **Personne** contient les items suivants :

  nom : le numéro (*num*) de la personne,

  attributs : *nom, prénom, ville, pays*

  De la même façon, un item du domaine **Article** a pour nom le numéro (num) de l'article, et ses attributs sont *titre*, *desc* et *catégorie*.

- Le domaine **Conseiller** contient les items suivants :

  nom : un numéro unique (par exemple, le rang *i* du $i^{ème}$ nuplet, sans ordre particulier).

  attributs : *pers, conseiller, jour*.

  De la même façon, un item du domaine **Note** a pour nom un numéro unique et pour attributs *pres, article, note, jour*.

Donner le pseudo code de chaque requête de l'application, en utilisant la syntaxe ci-dessus.

Question 2 : Proposer une solution pour organiser les données dans SimpleDB de telle sorte que **chaque requête R1, R2 et R3** de l'application soit réalisée avec le minimum d'invocation à SimpleDB (une seule invocation). Rmq : les opérations de modifications AjoutN, AjoutC, SupprC peuvent en contrepartie nécessiter plusieurs invocations.

a) Préciser chaque domaine et la structure de chaque item. Préciser si un attribut a une ou plusieurs valeurs.

b) Donner le pseudo code de chaque requête et chaque opération de modification de l'application, en utilisant la syntaxe ci-dessus.

Question 3 :

a) Donner la structure des clés reflétant le schéma relationnel.

b) Proposer une solution pour stocker les données de l'application dans **kvstore** de telle sorte que toutes les requêtes de l'application soient réalisées avec le minimum d'invocation au store. La clé est composée d'une liste de mots (sans limite sur le nombre de mots composant une clé). Cela permet d'organiser hiérarchiquement les données gérées dans le store. Il est possible en une seule requête get(p), de lire l'ensemble des couples (k,v) dont les clés ont le même préfixe p (p pouvant être une liste de plusieurs mots).

c) Donner le pseudo code de chaque requête et chaque opération de modification de l'application, en utilisant la syntaxe get(p), put(k,v), et delete(k).

Question 4 : On suppose que les données de l'application sont gérées par un système garantissant les propriétés AP (*cf.* la fiche de lecture). Préciser ce que cela signifie, en particulier :

a) Toutes les données sont-elles répliquées ?

b) Deux répliques d'une même donnée sont-elles toujours identiques ?

c) Pour chaque requête et chaque opération de modification de l'application, donner un exemple de situation où les données ne sont pas strictement cohérentes.

Question 5 : Est-il préférable de gérer les données de l'application avec un système AP ou CP ? Justifier en donnant les avantages de la solution que vous retenez.

Question 6 : Répartition des données.

On suppose que les données de l'application sont gérées dans 100 kvstores nommés S_1 à S_{100}. Chaque store est sur un seul site. Le store Si contient les données relatives à la personne p telle que h(p)=i (i est une valeur dans [1,100]).

On suppose (dans les questions a) et b)) que l'application se limite seulement à la requête **R1**(p) et à la modification **AjoutN**(p, a, n, j).

Chaque store S_i contient toutes les données nécessaires, et seulement celles-ci, pour traiter R1(p) quel que soit p tel que h(p)=i.

a) Décrire précisément le contenu de S_i .

b) L'ajout d'une note peut nécessiter de 'propager' la nouvelle note auprès des sites qui auront besoin de cette nouvelle note pour évaluer leurs requêtes R1. Donner le pseudo code de **AjoutN**(p, a, n, j).

c) On considère maintenant la transaction T(u1,u2,j) qui est la séquence :

AjoutC(u1,u2 ,j) suivie de **AjoutC**(u2,u1, j).

Proposer une solution pour traiter T entièrement sur un seul site. Est-ce que cela nécessite de déplacer des données d'un site vers un autre site ? Si oui, expliquer comment l'application peut ensuite continuer de traiter les requêtes R1(p) directement en accédant à un seul site.