LOG660 - Base de données de haute performance

Chapitre 18: Bases de données parallèles et réparties

Exercices:

QUESTION 1

a)

BD parallèles:

Les données peuvent être distribuées sur plusieurs disques d'un même site, et l'exécution des requêtes peut être parallélisée sur les différentes unités de traitement (CPU) du site.

BD réparties:

Les données sont distribuées et/ou dupliquées sur différents sites du réseau (ex: internet) qui possèdent un certain degré d'autonomie. Chaque site peut comporter une BD parallèle.

Avantages des BD réparties:

- Performance : En rapprochant les données des applications utilisant ces données (ex : stockant les comptes des clients montréalais dans un site à localisé à Montréal), on peut réduire les coûts de transfert sur le réseau et, ainsi, augmenter la performance des requêtes sur ces données.

- Fiabilité : En dupliquant certaines données importantes sur plusieurs sites, on minimise l'impact d'une panne sur un site. De même, en cas de panne, on peut rediriger le traitement d'une requête vers un autre site disponible.

- Extensibilité : Si les besoins en espace de stockage et en puissance de traitement augmentent on peut facilement rajouter un nouveau nœud (site), sans avoir à remplacer le serveur (ex : approche Google).

b)

Stratégie de semi-jointure :

La stratégie par semi-jointure permet de réduire le coût d'une jointure en limitant la quantité de données transférées sur le réseau.

Supposons que l'on veuille calculer $T_1 \bowtie T_2$ où la table T_i est située sur le site i. Au lieu de transférer une table complète d'un site à un autre, on envoie seulement les colonnes nécessaires à la jointure (la *clé*). Par exemple, on envoie $\Pi_{\text{clé}}(T_2)$ au site 1 et on fait la jointure avec T_1:

$$R = T_1 \bowtie \Pi_{\text{clé}}(T_2)$$

Ceci correspond à faire la semi-jointure entre T_1 et T_2. Ensuite, on envoie le résultat R au site 2 pour faire la jointure avec T2:

$$T = R \bowtie T_2 = T_1 \bowtie T_2$$

Les données transférées sont celles de $\Pi_{\text{clé}}(T_2)$ et de R, et ont une taille potentiellement moins grande que celle de T_1 ou de T_2.

c)

Répartition cyclique par bloc :

Au lieu de disposer les blocs d'une table séquentiellement sur un même disque, la répartition cyclique les dispose en alternance sur plusieurs disques. Par exemple,

Disque 1	**Disque 2**	**Disque 3**
bloc 1	bloc 2	bloc 3
bloc 4	bloc 5	bloc 6
...	...	...

Le but de cette stratégie est de permettre la lecture / écriture de plusieurs blocs en parallèle (un dans chaque disque).

d)

Fragmentation *horizontale* :

Chaque fragment contient un sous-ensemble de lignes de la table. Par exemple, on découpe la table Client selon la provenance (ex : province, état, etc.) d'un client.

Fragmentation *verticale* :

Chaque fragment contient un sous-ensemble de colonnes de la table. En pratique, ce type de fragmentation est rarement employé.

e)

Avantages de la fragmentation :

- La fragmentation horizontale permet de répartir les lignes d'une table sur les sites où le traitement de ces lignes est souvent fait, réduisant ainsi les temps de transfert sur le réseau.

- En cas de panne d'un site, l'information stockée sur les autres sites reste disponible.

f)

Avantages de la duplication :

- Réduit les coûts de transfert en dupliquant sur les différentes l'information globale à tous les sites. Par exemple, les codes et les frais associés aux transactions bancaires.

- Assure la disponibilité des données dupliquées dans le cas où un ou plusieurs sites tombent en panne.

g)

Duplication *synchrone* :

Une transaction modifiant des données de plusieurs sites n'est confirmée qu'au moment ou tous les sites ont confirmés les changements.

Duplication *asynchrone* :

Les mises à jour sont d'abord faites sur la copie primaire des tables, et les autres copies sont mises à jour en différé.

h)

Vues matérialisées :

Permet de créer une copie locale d'une table située sur un autre site distant. En somme, elles permettent d'implémenter le concept de la duplication (synchrone ou asynchrone).

i)

Optimisation dans les BD réparties

- Coût de communication: Contrairement aux BD centralisées, l'optimisation de requêtes utilisant des données sur plusieurs sites doit également tenir compte du coût de transfert sur le réseau.
- Ressources multiples: L'optimiseur doit également tenir compte de la localisation des données et des diverses ressources à sa disposition. Par exemple, plusieurs sites peuvent contribuer en parallèle à répondre à la requête selon les données qu'ils renferment.

j)

RAID 1:

Le niveau RAID 1 est basé sur la duplication des données sur des disques miroirs. Cette architecture est robuste aux pannes survenant sur un ou plusieurs disques. De plus, elle permet la lecture en parallèle sur les différents disques (mais pas l'écriture). Par contre, cette architecture est plutôt gourmande en terme d'espace.

RAID 5:

Contrairement au niveau RAID 1, le niveau RAID 5 ne duplique pas les données. En revanche, ce niveau emploi la répartition cyclique par bloc ce qui permet de faire des lectures ET des écritures en parallèle. Par ailleurs, elle permet une certaine forme de fiabilité à l'aide de bits de parité stockés séparément des données.

k)

Sélection dans BD parallèles:

En supposant que la table sur laquelle opère la sélection est fragmentée, on peut effectuer en parallèle une recherche sur chacun des fragments et ensuite combiner les résultats de ces recherches. Par ailleurs, si la fragmentation est faite selon la clé de sélection, on peut limiter la recherche aux fragments correspondants.

l)

Mémoire partagée:

Plusieurs processeurs (CPU) partagent la même mémoire vive (RAM). L'avantage est que les processeurs peuvent communiquer efficacement à travers la mémoire RAM. Cependant, la mémoire RAM constitue un goulot d'étranglement qui limite le nombre de CPU possibles.

Disques partagés:

Contrairement à la précédente, les CPU de cette architecture ont chacun leur propre RAM. Cela facilité l'extension de l'architecture (ajout de nouveau CPU) mais complexifie un peu la communication entre les processeurs. En pratique, cette architecture est celle employée le plus souvent.

QUESTION 2

Puisque l'application Web réside du côté de la succursale québécoise, pour accéder au catalogue français, il faut créer un lien entre la BD québécoise et la BD française :

```
CREATE PUBLIC DATABASE LINK serveur-bd.microdur.france.com
CONNECT TO nom_schema IDENTIFIED BY mot_de_passe;
```

Par ailleurs, puisque les délais doivent être minimisés, on choisit de une approche de duplication qui crée une copie locale du catalogue français dans la BD situé au Québec. Cette approche peut être implémentée à l'aide du vue matérialisée :

```
CREATE MATERIALIZED VIEW CatalogueFrance
REFRESH FAST ON COMMIT AS
SELECT * FROM Catalogue@serveur-bd.microdur.france.com
```

Le paramètre `ON COMMIT` est employé dans ce cas pour que les changements faits au catalogue français soient immédiatement visibles à l'application. Le paramètre `FAST` assure une mise à jour incrémentale rapide.

Enfin, pour rendre transparent à l'application la localisation des items, on crée une vue regroupant tous les produits :

```
CREATE VIEW CatalogueGlobal AS
(SELECT * FROM Catalogue) -- le catalogue local
UNION
(SELECT * FROM CatalogueFrance) -- le catalogue francais
```