INF1611

Optimisation

L'archive donnees.zip contient les fichiers nécessaires à la réalisation du sujet.

Partie 1 : Plans d'exécution de requêtes

Préparation du TP

Les plans d'exécution de requêtes sont copiés dans la table PLAN_TABLE. En général, cette table est déjà créée dans le SGBD. Si ceci n'est pas le cas, vous devez la créer au préalable en exécutant le script utlxplan.

Le script creerTablesPlanExec.sql contient des instructions de création des tables SALARIES et SERVICES et des instructions d'insertions de tuples dans ces tables. Lancez ce script pour créer les tables dans votre schéma.

SALARIES (idSalarie, ename, job, sal, idService) : 28672 tuples
SERVICES (idService, dname, loc) : 10 tuples

Aucune des tables n'a de clé primaire et aucun des attributs n'est initialement indexé.

Pour examiner un plan d'exécution de requête, vous allez procéder de la manière suivante :

::: réalisation du plan

```
EXPLAIN PLAN
SET statement_id = <<identificateur>>
FOR
<<requête>> ;
COMMIT ;
```

::: consultation du résultat

```
SELECT operation, options, id, parent_id, object_name
FROM plan_table
WHERE statement_id = <<identificateur>>
ORDER BY id;
```

::: pour connaître l'option d'optimisation

```
SELECT optimizer FROM plan_table
WHERE statement_id = <<identificateur>>;
```

::: pour modifier l'option d'optimisation

```
ALTER SESSION SET OPTIMIZER_MODE=mode
Mode=ALL_ROWS (défaut) ou FIRST_ROWS ou RULE ou CHOOSE
```

::: pour vider plan_table

```
DELETE FROM plan_table ;
COMMIT;
```

Examen des plans d'exécution de requêtes

Donnez l'algorithme correspondant au plan d'exécution de chacune des requêtes (Ri) suivantes (mode ALL_ROWS par défaut) :

```
SELECT count(*) FROM salaries ; (R1)
```

Créez un index *idxIdSalarieSalarie* sur l'attribut *idSalarie* de la table *salaries* :

```
CREATE INDEX idxIdSalarieSalarie ON salaries (idSalarie)
```

```
SELECT count(*) FROM salaries; (R2)
```

Supprimez l'index sur l'attribut *idSalarie* de la table *salaries* :

```
DROP INDEX idxIdSalarieSalarie
```

Ajoutez la contrainte de clé primaire *pkSalarieIdSalarie* sur l'attribut *idSalarie* de la table *salaries* (ie on indexe cet attribut avec un arbre B+) :

```
ALTER TABLE salaries ADD CONSTRAINT pkSalarieIdSalarie PRIMARY
KEY (idSalarie)
```

```
SELECT count(*) FROM salaries; (R3)
```

Passez l'optimiseur en mode RULE

```
SELECT count(*) FROM salaries; (R4)
```

Remettez l'optimiseur en mode ALL_ROWS

```
SELECT *       (R5)
FROM salaries;

SELECT *       (R6)
FROM salaries
WHERE idSalarie = 5000;

SELECT *       (R7)
FROM salaries
WHERE ename ='SCOTT';
```

Créez un index *idxEnameSalaries* sur l'attribut *ename* de la table *salaries* :

```
CREATE INDEX idxEnameSalaries ON salaries (ename)
```

Pour chacune des options de l'optimiseur ALL_ROWS, FIRST_ROWS, RULE : donnez l'algorithme correspondant au plan d'exécution de la requête :

```
SELECT *       (R8)
FROM salaries
WHERE ename='SCOTT';
```

Insérez un salarié de nom ‘PETIT’ dans la table salaries :

```
INSERT INTO Salaries values (100000, 'PETIT', 'CLERK', 1000, 3);
```

Pour l’option de l’optimiseur ALL_ROWS : donnez l’algorithme correspondant au plan d’exécution de la requête :

```
SELECT *       (R9)
FROM salaries
WHERE ename='PETIT';
```

Pour chacune des options de l’optimiseur ALL_ROWS, FIRST_ROWS, RULE : donnez l’algorithme correspondant au plan d’exécution de la requête :

```
SELECT *       (R10)
FROM salaries, services
WHERE salaries.idService=services.idService;
```

Modifiez la table *services* de manière à ce que l’attribut *idService* soit clé (contrainte *pkServicesIdService*) :

```
ALTER TABLE services ADD CONSTRAINT pkServicesIdService PRIMARY
KEY (idService)
```

Pour chacune des options de l’optimiseur ALL_ROWS, FIRST_ROWS, RULE : donnez l’algorithme correspondant au plan d’exécution de la requête :

```
SELECT *       (R11)
FROM salaries, services
WHERE salaries.idService=services.idService;
```

Créez un index *idxIdServiceSalaries* sur l’attribut *idService* de la table *salaries* :

```
CREATE INDEX idxIdServiceSalaries ON salaries (idService)
```

Pour chacune des options de l’optimiseur ALL_ROWS, FIRST_ROWS, RULE : donnez l’algorithme correspondant au plan d’exécution de la requête :

```
SELECT *       (R12)
FROM salaries, services
WHERE salaries.idService=services.idService;
```

Partie 2 : Optimisation des ressources de mémoire : calcul de ratios

- optimisation de la mémoire - shared pool area, zone LC : PIN RATIO
- optimisation de la mémoire - shared pool area , zone DC : GET RATIO
- optimisation de la mémoire - buffer de données : HIT RATIO
- optimisation de la mémoire - buffer de reprise : KEY RATIO

Calculer chacun de ces ratios et indiquez si la mémoire concerné est suffisamment optimisé .