Principes d'utilisation des systèmes de gestion de bases de données

Optimisation
Cours 6

Ecole informatique d'Orsay - M1
2005/06
Emmanuel Waller, LRI

Résumé des épisodes précédents

- But du module : savoir utiliser un SGBD pour résoudre les problèmes BD rencontrés par une application généraliste
- Les 12 problèmes BD
- Architecture, mode interactif, mode programme
- PL/SQL : code BD, triggers, procédures stockées

Les 12 problèmes de base de données

- modèle de données, conception, persistance, indépendance des niveaux
- contraintes d'intégrité, confidentialité
- mise à jour, interrogation
- reprise sur panne, contrôle de concurrence
- grandes quantités
- (répartition)

Cours 6

- Exercices concurrence
  - Jules et Jim
  - Examen 2004
- Projet
- Optimisation

Projet

- Suite énoncé détaillé : cours 8
- Anticiper :
  - Protocole commerce inter-entreprises
  - Gestion tous problèmes BD avec ordres SQL
  - Encapsulation tables par procédures/fonctions PL/SQL
  - Ecrans/menus de l'application solo/inter-entreprises
- Suite : JDBC, PHP (cours 7 à 11)

Rappel : présence au cours : indispensable

- Cause : TD seul ne suffit pas => ralentissement en TD : inacceptable
- Appel :
  - Cours (compter les présents)
  - Vérification en TD

optimisation : un premier contact

- but : rapidité
- évaluer une requête : stockage + algorithmes
- ordres SQL
- fonctionnement d'un index
- algorithmes pour opérateurs
- plan d'évaluation

optimisation : un premier contact

- pb 2 : grandes quantités
- but : évaluer les requêtes (select) en un temps raisonnable

Pb 2 : grandes quantités

- Certaines requêtes (select) : temps réponse énorme (heures, jours)
- Inacceptable
  - Cahier des charges
  - Ex : réservations SNCF (même si minute)
- Pourquoi ?
  - Voyons ce qu'il y a à faire
  - Où ça se passe
- Résoudre le problème des grandes quantités : optimiser

Évaluer une requête : approche naïve

- Paramètres :
  - Stockage : fichiers séquentiels non triés
  - Chaque opérateur :
    - Sélection : parcours total
    - Jointure : boucles imbriquées
  - Évaluation de la requête : bottom-up

Exemple

- Base
- Requête
- Taille RAM
- Taille disque
- Temps d'accés disque (ramener un bloc, écrire un bloc)

Coût

- Sur l'exemple :
  - Complexité
  - Complexité en nombre d'accès disque
  - Temps effectif sur l'exemple
- Conclusion :
  - Nombre d'accès disque simplement trop grand
  - Impossible augmenter taille RAM (futur ?)
  - Le SGBD et le programmeur ne peuvent être naïfs

Approches

- Complémentaires
- Stockage : structures de données (ex : index)
- Algorithmes opérateurs (ex : jointure)
- Algorithmes « globaux » (ex : pousser sélection)
- Statistiques serveur (ex : 70% notes 06-14)

  + théorie des probabilités
- Etc.

qui est concerné ?

- utilisateur : attente lors select
- programmeur :
  - responsable de la logique de son application
  - pas de la rapidité
- DBA :
  - responsable rapidité
  - Travail d'expert (outils non accessibles au programmeur)
- SGBD : algorithmes, stockage, etc.

évaluation d'une requête : comment fait le SGBD ?

- il doit gérer :
  - les données : stockées dans des fichiers
  - évaluation de la requête :
    - algorithme pour chaque opérateur
    - algorithme global pour la requête

- pb : accès disque lent (un million de fois plus que RAM) : naïf impossible
- approche :
  - stockage :
    - fichier
    - structures de données (nous : index)
  - algorithmes :
    - subtils
    - tirent parti des structures de données (choix de l'algorithme global en fonction des index existants)

remarque

- il existe beaucoup d'autres techniques
  - structures de données : arbres B, arbres B+, hachage, etc.
  - statistiques :
    - conservées par le serveur sur les requêtes antérieures et leurs résultats
    - probabilités
  - etc.

ordres SQL

- créer un index
- implicite : automatique lorsque
  - clé primaire
  - unique
  - etc.
- explicite :
  - create index idxew on t(a)

fonctionnement d'un index

- fichier des lignes :
  - rangé en séquence
  - ordre quelconque
- un deuxième fichier contient l'index :
  - structure de données
  - adresses physiques des lignes du fichier des lignes

structure de l'index

- arbre :
  - hauteur faible :
    - on trouve très vite la valeur cherchée
    - complexité = hauteur de l'arbre
  - noeuds internes : valeurs parmi lesquelles on cherche
  - feuilles : contiennent les adresse physiques de la ligne correspondant à chaque valeur de l'arbre

optimisation : un premier contact

- but : rapidité
- évaluer une requête : stockage + algorithmes
- ordres SQL
- fonctionnement d'un index
- algorithmes pour opérateurs
- plan d'évaluation

algorithme pour chaque opérateur

- select :
  - parcours (scan) séquentiel : renvoie ligne(s)
  - parcours d'index :
    - renvoie adresse(s)
    - récupération de la ligne(s) d'adresse(s) donnée(s) : renvoie ligne(s)
    - (voir déroulement sur le dessin de l'index)

- jointure :
  - simple : boucles imbriquées
  - subtile : tri fusion en mémoire externe (complexité : nlog(n))

plan d'évaluation d'une requête

- c'est l'algorithme « global » qui combine les algorithmes pour les opérateurs
- ex :
  - t (a primary key, b)
  - select b

    from t

    where a = 2005

plan d'évaluation

- peut se dessiner sous forme d'arbre
- s 'évalue bien sûr de bas en haut (comme l'arbre d'une expression arithmétique par exemple)
- select
  - accès par adresse
    - parcours d'index t(a)
    - t

exemple 2

- s (c, a foreign key references t)
- select c from s, t where s.a = t.a
- plan =

  select
  - boucles imbriquées
    - parcours séquentiel de s
    - accès par adresse
      - parcours d'index t(a)
      - t

résumé

- pour évaluer requête : SGBD doit choisir algorithmes pour chaque opérateur (sélection, jointure, etc.)
- or accès disque extrêmement lents : algorithmes naïfs impossibles
- index sur un fichier permet : accès très rapide aux enregistrements cherchés
  - amélioration peut être spectaculaire (heures en minutes)

- index sur un fichier permet : accès très rapide aux enregistrements cherchés
  - amélioration peut être spectaculaire (heures en minutes)
  - algorithmes de jointures utilisent les index qui sont disponibles, sinon ils rusent
  - création d'index :
    - explicite : create index...
    - implicite : clé primaire

- le plan d'évaluation d'une requête utilise :
  - index
  - algorithmes de jointure
  - il est obtenu par « compilation » de la requête
  - est consultable (explain sous Oracle)

Compétences à acquérir

- Construction : savoir mettre un index quand nécessaire
- Analyse :
  - savoir où sont les index
  - faire le travail du SGBD : choisir un plan d'évaluation tirant le meilleur parti des index

TD

- créer base et lire le schéma
- plans :
  - stocké dans table => créer cette table : utlxplan (utilitaire for Xecution plans)
  - calculer par Oracle le plan d'une requête (+ le stocker : explain plan...)
  - affichage : explain.sql + identificateur du plan
  - sur table avant machine

optimisation : un premier contact

- but : rapidité
- évaluer une requête : stockage + algorithmes
- ordres SQL
- fonctionnement d'un index
- algorithmes pour opérateurs
- plan d'évaluation