Introduction aux SGBD

Talel.Abdessalem@enst.fr

www.enst.fr/~talel/ens.html

Base de Données ?

- Une collection de données cohérentes entre elles, généralement de taille importante.
- Modélise une *entreprise* du monde réel
  - Entités (ex., étudiants, Briques)
  - Associations (ex., Paul est inscrit en BD)
- Un *Système de Gestion de Bases de Données (SGBD)* est un logiciel destiné au stockage et à la manipulation de bases de données.

Pourquoi un SGBD?

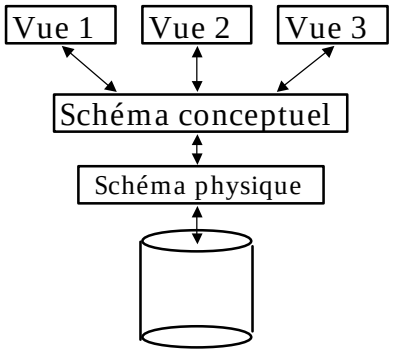
- Indépendance des données / applications et sûreté d'accès aux données.
- Temps de développement d'application réduit.
- Intégrité des données et sécurité des accès.
- Administration des données uniforme.
- Concurrence des accès et reprise sur panne.

Modèle de données

- Un *modèle de données* est un ensemble de concepts sur les données.
- Un *schéma* est une description d'un ensemble de données, s'appuyant sur un modèle de données.
- Le *modèle relationnel* est le plus répandu.
  - Concepts de base: *relation*, table avec tuples et des colonnes.
  - Chaque relation a un *schéma*, qui décrit ses colonnes.
- Les modèles objet et objet-relationnel sont utilisés pour gérer des données complexes.
- Les modèles semi-structurés se cherchent une place dans les applications web, intégration de données hétérogènes, ...

Niveaux d'abstraction

- Plusieurs *vues*, un *schéma conceptuel* et un *schéma physique*.
  - Les vues décrivent comment l'utilisateur voit les données.
  - Le schéma conceptuel définit la structure logique des données.
  - Le schéma physique décrit la structure physique, de stockage, des données.



- *DDL : langage de définition des données; DML : langage de manipulation des données.*

Exemple: Base de données

- Schéma conceptuel :
  - *élèves(ide: string, nom: string, login: string, age: integer)*
  - *Cours(idc: string, nomc:string, ths_ens: integer, ths_tp: integer)*
  - *inscription(ide:string, idc:string)*
- Schéma physique :
  - fichiers non séquentiels et index en b-arbres sur les attributs clés.
- Schéma externe (Vues) :
  - *Volume_th_elves(ide:string, nb_ths:integer)*

Indépendance des données

- Les applications sont isolées des changements de structure et du mode de stockage des données.
- *Indépendance logique des données*: Protection des changements de structure des données au niveau *logique*.
- *Indépendance physique des données*: Protection des changement de structure au niveau physique.
  - *Un des plus importants bénéfices de l'utilisation des SGBD*

Contrôle de concurrence

- L'exécution concurrente de programmes est essentielle pour un SGBD.
  - Les accès disque sont fréquents et relativement *lents*, il est important que l'unité centrale puisse exécuter de façon concurrente les programmes des utilisateurs.
- L'exécution partielle des actions de différents programmes peut aboutir à des incohérences: ex., opération de débit en même temps que la réalisation d'un virement.
- Les SGBD assurent que la concurrence soit réalisée sans problème: chaque utilisateur a l'impression d'être seul à travailler sur le système.

Transaction: Exécution d'un programme au-dessus d'une BD

- Concept clé : *transaction*, une séquence atomique d'actions sur une BD (lectures / écritures).
- Chaque transaction doit laisser la BD dans un *état cohérent* après l'avoir prise dans un état cohérent.
  - Les utilisateurs peuvent spécifier des *contraintes d'intégrité* simples sur les données et le SGBD se charge de les garder inviolables.
  - En dehors de ça, le SGBD n'a pas conscience de la sémantique des données (ex., il ne comprend pas comment les intérêts d'un compte bancaire sont calculés).
  - Le fait qu'une transaction préserve la cohérence de la BD est au bout du compte de la responsabilité de l'utilisateur!

Ordonnancement et concurrence des transactions

- Les SGBD assurent que l'exécution de {T1, ... , Tn} soit équivalente à une exécution *en série* T1 '... Tn '.
  - Avant de lire / écrire un élément, chaque transaction demande à émettre un verrou sur l'élément et attend que le SGBD lui accorde ce verrou. Tous les verrous sont relâchés à la fin de la transaction (protocole V2P strict.)
  - Idée: Si une action de Ti (ex., écrire X) affecte Tj (qui effectue une lecture sur X), une des deux, disons Ti, obtient le verrou sur X la première et Tj est forcée à attendre la fin de Ti; cette façon de faire permet d'ordonner les transactions.
  - Et si Tj a déjà verrouillé Y et que Ti demande par la suite à verrouiller Y? (Deadlock!) Ti ou Tj est abandonnée (aborted) et remise en concurrence!

Atomicité

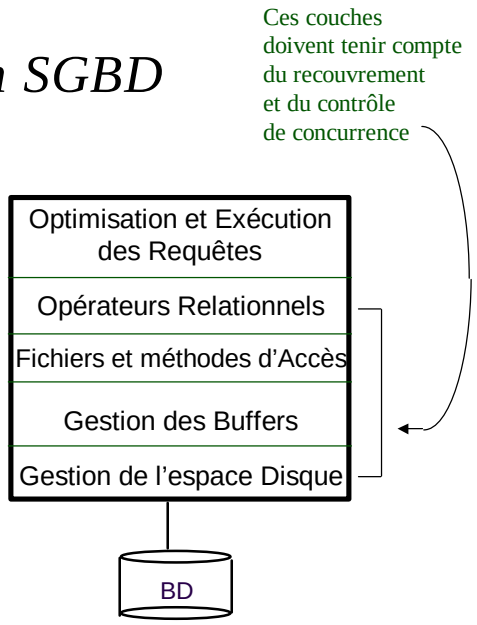
- Les SGBD assurent l'*atomicité* (tout ou rien) même si un crashes surgit au milieu d'une exécution de transaction.
- Idée: garder un *journal ou log* (histoire) de toutes les actions réalisées par le SGBD :
  - Avant qu'un changement ne soit réalisé sur la BD, l'action est tracée dans un log file.
  - Après un crash, les effets d'une exécution partielle d'une transaction sont défaites à l'aide du fichier log.

Le journal

- Les actions suivantes sont mémorisées dans le journal:
  - *Ti écrit un élément*: l'ancienne et la nouvelle valeur.
    - L'enregistrement correspondant du journal doit être stocké sur le disque *avant* la page de données modifiée !
  - *Ti valide(commit)/abandonne(abort)*: un enregistrement du journal mémorise cette action.
- Les enregistrements du journal contiennent l'identifiant de la transaction, ainsi il est facile de défaire une transaction spécifique (ex., en cas de deadlock).
- Le journal est souvent dupliqué et archivé sur un support « sûr ».
- Toute l'activité enregistrée dans le journal (ex. verrouillage / déverrouillage, deadlocks etc.) est gérée de façon transparente par le SGBD.

Architecture d'un SGBD

- Un SGBD typique possède une architecture en plusieurs couches.
- Le figure ne montre pas le module de recouvrement et de contrôle de concurrence.
- Ceci est une architecture possible; chaque système possède sa propre variante.



Résumé

- Les SGBD sont utilisés pour maintenir et interroger un volume de données important.
- Quelques bénéfices : reprise sur panne, accès concurrent, développement rapide d'applications, intégrité et sécurité des données.
- Les niveaux d'abstraction permettent l'indépendance des données.
- Un SGBD possède une architecture en couches.

Enseignements de bases de données

- Niveau Système
  - Les transactions et la gestion de la concurrence
  - L'indexation et l'accès aux données
  - L'organisation physique et la gestion du stockage
  - L'optimisation de requêtes
  - Le tunning des SGBD
  - Se mettre à l'échelle de la carte …
- Niveau modèles et langages
  - Les modèles post-relationnels et les langages associés : SQL, SQL3, OQL, Xquery, Lore, etc.
  - Les modèles conceptuels, la théorie de la normalisation, l'intégrité et le maintien de la cohérence des données.
- Mini-mémoires
  - Sujets de R&D proposés par les profs ou à l'initiative des élèves.

Bibliographie

- J.D. Ullman, Principles of Database and Knowledge-Base Systems, volume 1 et 2, Ed. Computer Science Press.
- A. Silberschatz, H. F. Korth et S. Sudarshan, Database System Concepts, Ed. Mc-Graw Hill.
- ElMasri et Navathe, Fundamentals of Database Systems, Ed. The Benjamin / Cummings Publishing compagny.