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

Indépendance des niveaux

L3 Informatique
Emmanuel Waller, LRI, Orsay

Rappel : les deux parties du cours

- création et gestion de la base
  - les problèmes BD :
    - liés à la construction de la base :

      modèle, conception, mise à jour, persistance, contraintes, indépendance des niveaux (L3), confidentialité
    - liés à l'interrogation de la base :

      interrogation (L3), grandes quantités
    - liés à la dynamique de la base :

      reprise sur panne, contrôle de concurrence
  - les traitements bas niveau en mode programme : PL/SQL
- accès à la base depuis un programme généraliste :

  les problèmes MP étudiés à travers (outre PL/SQL) : Java, PHP
- XML, XSQL, XSLT, intégration avec relationnel

Rappel du problème

- La situation :
  - Il peut être nécessaire de réorganiser certaines parties de l'application pour cause de (par exemple) :
    - Performances
    - Évolutivité
- Il faut :
  - Pouvoir le faire sans changer son comportement
  - « modularité » entre les différents niveau de l'application (analogue esprit objet)

Les 3 niveaux des applications de bases de données

- « Logique » : tables relationnelles
- « Physique » : les détails de leur organisation physique sur le disque
- « Externe » : les programmes accédant aux tables relationnelles
- Norme ANSI 1978

Motivation : exemple 1 : performances

- Performances d'une requête dépendent des détails du stockage sur le disque
- Il faut pouvoir modifier cette organisation sans modifier la structure logique des tables relationnelles (ex : pas possible en Java)
- Autrement dit : le niveau logique doit être indépendant du niveau physique (à préciser)

Motivation : exemple 2 : évolutivité

- Au cours du temps les fonctionnalités de l'application évoluent, en particulier les tables
- Mais les programmes utilisant les anciennes tables doivent toujours fonctionner avec le même comportement
- Le niveau externe doit être indépendant du niveau logique (à préciser)

Outils fournis par le SGBD

- Notion de vue : indépendance des niveaux externe et logique
- Le dictionnaire de données : indépendance des niveaux logique et physique

Exemple

- Situation :
  - Billet (client, destination, prix)
  - Secrétaire SNCF Toto gère uniquement les clients pour Saint Tropez
  - Tape toujours la même requête (ex : gros quotient) :

```
select client
from billet
where destination like  'St Trop%'
```

Problèmes

- situation insupportable pour quatre raisons
- Confort : il connaît très mal
  - SQL
  - Les tables de l'application
- Grandes quantités (performances) : elle est réexécutée à chaque fois

- Confidentialité : bien qu'il n'en ait pas besoin pour sa gestion, nécessite droits select sur
  - Colonne prix de billet (cas général : toutes tables)
  - Lignes de billet autres que Saint Tropez
- Indépendance des niveaux : en cas de réorganisation des tables par le programmeur de l'application en (par exemple) :

  Voyage(client, n°train) Train(n°train, destination, prix)

  la requête :

  select client from billet where destination like 'St Trop%'

  ne fonctionne plus

principe

- le programmeur de l'application (contenant la base) va associer un « alias » à cette requête
- Toto va utiliser uniquement cet alias

définition

- Une *vue* est une table virtuelle dont le contenu est défini par une requête
- Son contenu peut être au choix
  - Recalculé
  - Matérialisé (et maintenu par le système)

Ordres SQL

- create view billetsainttrop

  as

  select client

  from billet

  where destination like 'St Trop%'
- (Matérialisation ou non : hors programme)
- Rem : revoke select on billet from toto

Conséquences

- Toto tape simplement :

  select * from billetsainttrop
- Confort : pas de quotient
- Grandes quantités : pas d'attente (résultat matérialisé)
- Confidentialité respectée : il ne connaît que la table virtuelle billetsainttrop

- Indépendance des niveaux respectée : en cas de réorganisation des tables ci-dessus, le programmeur de l'application tape en même temps:

```
create or replace view billetsainttrop as
select client from voyage, train
where voyage.n°train = train.n°train
   and destination like 'St Trop%'
```

  - La requête de Toto

```
select * from billetsainttrop
```

    - Reste identique (inutile l'adapter)
    - Fonctionne
      - toujours (sauf 1ms entre les deux ordres)
      - Avec le même résultat
  - Toto n'a donc même pas besoin d'être prévenu

Autre avantage

- Maintenance :
  - Dans le cas général la requête de toto est l'une des requêtes de l'application
  - Si elle apparaît plusieurs fois dans l'application, la remplacer par l' « appel à la vue » (select billetsainttrop) factorise son code
    - Pas de duplication de code
    - Comme fonction

vocabulaire

- On dit que :
  - Le programme du niveau externe est indépendant du niveau logique

    = les mises à jour du niveau logique ne l'affectent pas
  - Le programme du niveau externe satisfait la propriété d'indépendance du niveau externe par rapport au niveau logique

exemple

- démonstration

remarque

- La mise à jour ci-dessus du schéma est SPI par rapport à la vue :
  - elle ne perd pas d'information nécessaire à la vue
- Ex : mise à jour non SPI :
  - alter table billet drop column destination
  - Il n'y a aucun moyen de renvoyer le même résultat qu'avant
  - Dans ce cas l'indépendance des niveaux est impossible

- L'indépendance des niveaux est possible si, et seulement si la mise à jour est SPI

Remarque : mises à jour de la vue

- Ex : insert into billetsainttrop values ...
- dépend des cas
- Subtilités si, ex : jointure
- Hors programme

vues : en résumé

- Le concept de vue permet de gérer :
  - 3 problèmes de base de données :
    - Indépendance des niveaux
    - Confidentialité
    - Grandes quantités
  - Des aspects de confort et maintenance

Le dictionnaire de données

- utilisés pour deux aspects
- pour indépendance des niveaux :
  - but : traduction entre logique et physique
    - conséquence du modèle de haut niveau
  - comment : il stocke pour chaque table l'adresse du début de la liste chaînée des blocs physiques du disque qui contiennent les enregistrements de la table, etc.

- pour... dictionnaire :
  - but : répertorie informations concernant la base de données : tables, utilisateurs, procédures, événements, etc.
  - comment :
    - Stockées dans des... tables ! (« tables système »)
    - Accès par... vues
      - select * from dictionary
      - select table_name from user_tables

        select table_name from all_tables where owner = 'WALLER'
    - Voir ligne de t nécessite privilège select sur t

exemple

- Requêtes sur le dictionnaire de données

Indépendance du niveau physique par rapport aux mises à jour du niveau logique

- Certaines mises à jour n'obligent pas le système à réorganiser le disque
  - ex : ajout (ou suppression) d'une colonne d'une table
- Assuré par dictionnaire de données (et système)

Indépendance du niveau logique par rapport aux mises à jour du niveau externe

- Il n'y a pas de problème : le niveau logique (tables) ne dépend pas du niveau externe (vues)

Ce qu'il faut se rappeler

- Indépendance des niveaux :
  - Indépendance du niveau externe par rapport aux mises à jour du schéma du niveau logique : assurée par vues
  - Autre sens : pas de problème
  - Assurées par dictionnaire de données :
    - Indépendance du niveau logique par rapport aux mises à jour du niveau physique
    - Autre sens

- L'outil vue apporte solutions aussi pour :
  - Confidentialité : plus expressif que grant seul
  - Grandes quantités
  - Confort et maintenance

Compétences à acquérir

- Analyse :
  - Indépendance des niveaux : détecter et illustrer les occurrences de ce problème dans une application
  - Outil vues : savoir décrire le comportement d'une application en présence de vues
- Construction :
  - Indépendance des niveaux : savoir le gérer dans une application
  - Dictionnaire de données : savoir y récupérer les informations nécessaires

Compétences à acquérir

- Confidentialité (complément) : construction :
  - Utiliser les vues quand nécessaire

Résoudre un exercice

- Indépendance des niveaux
  - Les programmes du niveau externe (ordres SQL et procédures stockées) ne doivent accéder qu'à des vues (pas à des tables)
  - Pour les accès en mise à jour :
    - On pourra utiliser procédures stockées (vu ult.) au lieu vues (cause mise à jour vues hors programme)
- Confidentialité :
  - On considère les cases du tableau vu en TD
  - Une vue (SQL) associée à un grant (SQL) peut résoudre un problème de confidentialité