Plan : Vues et droits

1. Vues
    1. Définition et utilisation
    2. Vues concrètes
    3. Mise à jour
    4. Recalcul des vues concrètes
2. Droits.
    1. Catalogue
    2. Gestion des droits proprement dite

1. Vues 1. Définition et utilisation

Vue ?

Table virtuelle dont le schéma et le contenu sont dérivés de la base réelle par une question.

Exemple :

- VETEMENT (CodeProduit, Prix, CodeTVA, Couleur)
- TAUXTVA (CodeTVA, Taux)

- **Souvent des prix TTC ?**

```
CREATE VIEW RESUMETTC (PRIXTTC, CODE) AS
        SELECT  Prix*(1+Taux), CodeProduit    FROM VETEMENTS, TAUXTVA
            WHERE TAUXTVA.CodeTVA=VETEMENTS.CodeTVA ;
```

- **Un des magasins ne traite que les articles rouges ?**

```
CREATE VIEW VETEMENTROUGE (CodeProduit, Prix,CodeTVA, Couleur) AS
        SELECT  *  FROM VETEMENTS WHERE Couleur =  ''Rouge''
        WITH CHECK OPTION ;
```

1. Vues 1. Définition et utilisation

Utilisée comme une table :

- SELECT * FROM VETEMENTROUGE
  WHERE CodeTVA = ‘’Luxe’’;
  ⇒ Remplacé par sa définition.
  SELECT CodeProduit FROM **VETEMENTS**
  WHERE CodeTVA = ‘’Luxe’’ **AND Couleur=‘’Rouge’’** ;

- INSERT INTO VETEMENTROUGE
  VALUES (12345, 120.50, ‘’Luxe’’, ‘’Rouge’’) ;

- INSERT INTO VETEMENTROUGE
  VALUES (12345, 120.50, ‘’Luxe’’, ‘’Bleue’’) ;

- UPDATE VETEMENTSROUGE
  SET PRIX = PRIX *1.1 ;

1. Vues 1. Définition et utilisation

Quelques remarques :

A travers la vue, les mises à jours des données ne sont pas toujours autorisées (voir les problèmes posés plus loin), cela dépend de la vue.

- Est-elle monotable ?
- Y a-t-il des champs calculés (comme le prixTTC dans RESUMETTC) ?
- Est-ce un agrégat ?

et du SGBD….

1. Vues 1. Définition et utilisation

Utilisation :

- Schéma Externe :
  - Restreindre la vue de la base suivant l'utilisateur.
  - Dissocier le schéma interne et Schéma externe.
- Améliorer la lisibilité de l'écriture de certaines requêtes (comme un sous-programme)
- **Optimisation dans le cas de vues concrètes.**

1. Vues 2. Vues concrètes.

Mécanisme normal :

vue remplacée par sa définition au moment de l'appel.

Si vue souvent utilisée, et requête associée couteuse à évaluer ?

=> Utilisation d'une vue concrète.

CREATE CONCRETE VIEW RESUMETTC(PrixTTC,Code) as … ;

La vue est calculée et stockée en dur sous forme d'une table.

- SELECT * FROM RESUMETTC WHERE PrixTTC > 120 ; pas de recalcul !
- *ATTENTION* : ***modification*** des tables référencées = ***recalcul*** de la vue…

1. Vues 3. Mise à jour à travers une vue.

Modification des données à travers une vue ? Exemples :

- UPDATE VETEMENTROUGE SET prix = prix*1.2 WHERE CodeTVA = ‘’Luxe’’ ;
- INSERT INTO VETEMENTROUGE VALUES (12345, 120.50, ‘’Luxe’’, ‘’Rouge’’) ;
- DELETE FROM VETEMENTROUGE WHERE Prix =123.50 ;

V = Vue avant modification : $V = X(B)$

V’ = Vue après modification : $V' = U_V(V)$

Problème : trouver la mise à jour U_B de la base telle que

$$V' = U_V(V) = X(U_B(B))$$

Exemple : trouver les modifications à faire à VETEMENT pour chacune de ces mises à jour de VETEMENTROUGE.

Compliqué dans le cas général => règles pour autoriser les mises à jour à travers une vue.

1. Vues 3. Mise à jour à travers une vue.

a) En SQL 92

- Conditions pour pouvoir faire une mise à jour à travers une vue (la vue est ‘’mettable à jour’’) :
  - Ni UNION, JOIN, INTERSECT, EXCEPT
  - Ni DISTINCT, ni GROUP BY, ni HAVING.
  - Pas d’expression dans les champs de la requête.
  - Une seule table **T** dans le FROM de la requête,
  - Si le WHERE contient un select imbriqué, celui-ci ne doit pas faire référence à la table **T**

- La mise à jour est totale : Soit les 3 droits (UPDATE, INSERT, DELETE) sont accordés, soit aucun.

- **Le UPDATE est total** : soit tous les champs de la vue sont ‘’mettables à jour’’ soit aucun.

1. Vues 3. Mise à jour à travers une vue.

b) Quelques remarques

- En théorie, beaucoup plus de possibilités.
- En pratique, très dépendant du SGBD.
- Dans l'idéal, la mise à jour doit modifier la *vue*, puisque c'est à travers la vue qu'on modifie les données.

  INSERT INTO VETEMENTROUGE VALUES (12345, 120.50, ''Luxe'', ''Bleue'') ;

  SELECT * FROM VETEMENTROUGE ; ***Ne voit pas l'article inséré !***

  On peut penser que l'ajout n'a pas fonctionné.

  ⇒ **Fortement conseillé** de mettre WITH CHECK OPTION.

1. Vues 3. Mise à jour à travers une vue.

b) Quelques remarques

- Vues équivalentes ? Comportement différent pour la mise à jour.

  – CREATE VIEW V1 AS SELECT * FROM VETEMENT
    WHERE Couleur = ‘’Bleu’’ OR Couleur = ‘’Jaune’’ ;

  – CREATE VIEW V1 AS SELECT * FROM VETEMENT
    WHERE Couleur = ‘’Bleu’’
    UNION
    CREATE VIEW V1 AS SELECT * FROM VETEMENT
    WHERE Couleur = ‘’Jaune’’ ;

  V1 est ‘’updatable’’ mais pas **V2** ! Pourtant elles calculent la même chose.

1. Vues 4. Recalcul de vues concrètes

Modification des données référencées par une vue concrète V
=> Recalcul de la vue V.

a) Recalcul complet systématique de V ? Lourd, à éviter.

b) Recalcul uniquement de la modification.

Principe :

1) calcul de la vue lors de sa création.
2) à chaque mise à jour Δ des relations **Ri**,

$$\mathbf{Ri} <= (\mathbf{Ri} \setminus \mathbf{Ri}^{-}) \cup \mathbf{Ri}^{+}$$

Plutôt que de recalculer complètement la vue, on va se servir si possible de $\mathbf{Ri}^{+}$, $\mathbf{Ri}^{-}$ et **V**

1. Vues 4. Recalcul de vues concrètes

Vue automaintenable : Une vue V (référençant des tables Ri) est *automaintenable* si lors d'une mise à jour, la connaissance des Ri^+, Ri^- et de V, suffit à calculer V+ et V-.

1. Vues 4. Recalcul de vues concrètes

Exemples :

- CREATE CONCRETE VIEW V1 (PrixMin, Catégorie) AS
  SELECT MIN(PRIX), CodeTVA FROM VETEMENT GROUP BY CodeTVA ;

Automaintenable pour le INSERT. mais ni pour le DELETE, ni le UPDATE.

- CREATE CONCRETE VIEW V2 (Prix, Categorie, CodeProduit) AS
  SELECT Prix, CodeTVA, CodeProduit FROM VETEMENT WHERE Prix > 20 ;

Automaintenable pour le INSERT, DELETE (et donc UPDATE).

- CREATE CONCRETE VIEW V3 (Prix, Categorie) AS
  SELECT DISTINCT Prix, CodeTVA FROM VETEMENT
  WHERE Prix > 200 ;

Automaintenable pour le INSERT, mais pas le DELETE (et donc pas le UPDATE).

1. Vues 4. Recalcul de vues concrètes

Exemple pour une jointure :

- CREATE CONCRETE VIEW V4 (CodeProduit, Taux) AS
  SELECT CodeProduit , Taux FROM Vetement as V, TauxTva as T
  WHERE T.CodeTva= V.CodeTva ;

Automaintenable ni pour le INSERT (insertion d'un CodeTVA?)
ni pour le DELETE (suppression d'un CodeTVA ?).

En général : si une vue **V** utilisant des jointures sur R1,R2, …Rk
- conserve tous les champs nécessaires aux jointures (et en particulier les clés),
- est sans élimination des doublons,

Alors **V** est automaintenable.

1. Vues 4. Recalcul de vues concrètes

Agrégats automaintenables :

Les fonctions d'agrégats ;

- **Distributives : On peut séparer le calcul.**

  Exemple : SUM (t1,t2, t3,t4) = SUM (SUM(t1,t2) , SUM(t3,t4)).
  MIN, MAX, COUNT (en ne considérant pas le distinct).

- **Algébriques : Fonctions algébriques de fonctions distributives.**
  Exemple : AVG(…) = SUM(…)/COUNT(…)

- **Les autres**

1. Vues 4. Recalcul de vues concrètes

SUM, MIN, MAX, COUNT automaintenables en INSERTION,

SUM,* COUNT* automaintenables en SUPPRESSION.

=> AVG automaintenable en INSERTION/SUPPRESSION.

*

1. Vues 4. Recalcul de vues concrètes

Exemple :

```
CREATE VIEW TOTAL AS
SELECT SUM(PRIX),CodeTVA FROM VETEMENT
GROUP BY CodeTVA ;
```

Comportement de la vue vis-à-vis des suppressions suivantes ?

- DELETE FROM VETEMENT WHERE CodeProduit = 20 ;
- DELETE FROM VETEMENT WHERE CodeProduit = 20 and Prix=10 and CodeTVA =‘’Luxe’’

1. Gestion des droits.

Partie très très variable suivant le SGBD…

1. Gestion des droits. 1. Le catalogue.

Catalogue : Dictionnaire des données, partie du SGBD stockant la définition et caractéristiques des

- Bases,
- Tables,
- Index,
- Vues,
- Utilisateurs
- Contraintes,
- Déclencheurs et procédures
- et tout objet défini dans le SGBD.

Catalogue stocké dans … une ou des bases du SGBD !

Toute création de d'utilisateur, de droit, etc, se traduit dans le catalogue.

1. Gestion des droits. 1. Le catalogue.

- SYBASE :
  - Une base **MASTER** : description des bases et du serveur, nb de connections, stockage, etc…
  - Une base **MODEL** par base de données, description des tables, index, utilisateurs de cette base, Index, etc…
    Exemple : tables SYSUSERS (utilisateurs), SYSOBJECTS(tables, vues, etc…) SYSPROTECT (droits).

- ORACLE :
  - **SYSTEM**, un espace dédié au sein de chaque base, avec un utilisateur privilégié (SYS).
  - La table **DICTIONNARY** contient la liste des tables et vues de ce dictionnaire. Exemple :
    - Description des objets : ALL_TABLES, USER_TABLES, DBA_TABLES,
    - Description physique, du stockage, des logs, etc : tables V$xxxx réservées à l'utilisateur SYS.

1. Gestion des droits. 2. Gestion de droits.

a) Plusieurs types d'utilisateurs.

- Administrateur système (DBA)
- Les autres,

b) Les droits ou privilèges = droits sur les objets.

SELECT, UPDATE, DELETE, INSERT, REFERENCE, EXECUTE.

c) Les rôles (pas toujours présents)**:**

Ensemble de droits, permet de grouper les utilisateurs.

Exemple :

1) créer le rôle COMPTABLE,
2) Affecter les droits au rôle COMPTABLE,
3) Affecter le rôle COMPTABLE aux utilisateurs U1, U2, U3.

Il est possible d'affecter plusieurs rôles à un même utilisateur.

1. Gestion des droits. 2. Gestion de droits.

Quelques commandes :

a) Gestion des droits de base (assez standard).

- GRANT [ALL PRIVILEGES | {PRIV1, PRIV2,…}]
  ON [TABLE|VIEW|…] [nom1|{nom1,nom2,…}]
  TO [PUBLIC | {user1,user2,….}]
  [WITH GRANT OPTION]

- REVOKE [ALL PRIVILEGES | {PRIV1, PRIV2,…}]
  ON [TABLE|VIEW|…] [nom1|{nom1,nom2,…}]
  FROM [PUBLIC | {user1,user2,….}]

PRIVILEGES ?

- SELECT,
- INSERT,
- DELETE,
- REFERENCES [COL1{,Col2,Col3…}]
- UPDATE [COL1{,Col2,Col3…}]

1. Gestion des droits. 2. Gestion de droits.

b) Gestion des rôles (pour ORACLE)

- commandes CREATE ROLE, DROP ROLE, GRANT, SET,
- rôle prédéfini : SYSDBA

Exemple :

1) **Création et octroi du rôle.**

> CREATE ROLE LES_IND ;
> GRANT CREATE ANY INDEX, ALTER ANY INDEX, DROP ANY INDEX TO LES_IND ;
> GRANT LES_IND TO TITI.

2) **Session de TITI lorsqu'il veut manipuler les index.**

> SET LES_IND ON ;
> ….
> SET LES_IND OFF ;