BASES DE DONNEES
TP 5 : Notion de Sous–Schéma
Restriction de la vision (VIEW) et des actions (GRANT)

Les vues sont des objets virtuels : ils n'ont pas d'existence propre (ou physique). A l'appel de la vue (select), le sgbd reconstruit son contenu avec la définition conservée dans le dictionnaire de données (user_views). On parle ainsi de tables de base et de tables virtuelles.

Les vues sont utilisées principalement pour :

-modifier la présentation de la base de données à certains utilisateurs. On parlera alors de "**Schéma Externe**" en opposition du "**Schéma Physique**" correspondant à l'implémentation des tables.

-simplifier la construction de certaines requêtes difficiles. Les vues sont alors crées et utilisées comme des résultats intermédiaires. Une vue pouvant être utilisée dans la création d'une autre vue.

-calculer directement un résultat qui nécessiterait une requête complexe. C'est le cas souvent pour certaines jointures pas trés évidentes à réaliser.

-assurer la **confidentialité** de la base de données en construisant des **sous-schémas** adaptés à chaque utilisateur (ou famille d'utilisateur). Selon le principe que tout le monde ne peut **voir** et **faire** n'importe quoi ; dans un premier temps on réduit le champ de vision de certaines tables (restriction horizontale, verticale ou mixte) en construisant des vues puis, ensuite, on réduit les actions sur ces vues avec l'ordre GRANT. Chaque table de base peut être contrôlée par un ensemble de contraintes (constraints).

Tous les utilisateurs travaillant sur une partie de table de base à travers une vue sont contrôlés par ces contraintes générales. Il est aussi possible de donner des contraintes spécifiques pour chaque vue ou 'morceau' de table ; ces contraintes sont crées avec l'option de contrôle (with check option).

Une vue peut trés bien avoir des colonnes de base (existantes dans la table) ou des colonnes virtuelles (calculées). Ces colonnes calculées ne pourront jamais être mises à jour dans la vue.

Une vue mono-table peut recevoir un ordre du langage de manipulation (insert, update ou delete). Ces ordres sont effectivement exécutés dans la table de base correspondante. Un premier contrôle est effectué sur les contraintes de la vue, puis sur celles de la table. Une vue multi-table ne peut, en aucun cas, être modifiée.

Ecrire toutes les requêtes demandées dans un fichier pour l'imprimer à la fin.

1 - Création et utilisation de vues mono-table sans contraintes

(a) Créer la vue mono-table CHERCHEUR_O à partir de la table CHERCHEUR ne contenant que les chercheurs spécialisés en 'objet' (Restriction Horizontale avec la chaine 'objet' présente dans la colonne nom_spécialité de la table SPECIALITE). Certaines colonnes sont conservées mais l'ordre et les noms changent :

CHERCHEUR_O(n_o, nom_o, univ_o, n_equ_o,spec_o)

Attention : cette vue doit être 'mono-table'.

(b) Vérifier le contenu de la vue. Augmentez le paramètre LONG (SET LONG 1000) et vérifiez dans la table USER_VIEWS du dictionnaire de données.

(c) Insérer deux tuples dans la vue : un chercheur appartenant à la spécialté 'objet' et un autre d'une autre spécialité.

(d) Vérifier le contenu de la vue et celui de la table. Comprendre. Supprimer ces deux tuples.

(e) Créer une vue COUT_PROJET à partir de la table PROJET contenant tous les projets avec le coût par semaine(fonction du nombre de jours/chercheurs) :

COUT_PROJET(n_projet, nom_projet, nb_jours, cout)

On compte 500 Euros par jour et par chercheur. Vérifier le contenu de la vue.

(f) A partir de cette vue, afficher les projets dont le coût dépasse un montant entré au clavier. Ecrire la même requête sans utiliser la vue.

2-Création et utilisation de vues multi-tables sans contraintes

(a) Reprendre la dernière vue mono-table (COUT_PROJET) en tenant compte maintenant de la spécialité des chercheurs intervenant dans le projet. Le tarif journalier dépend maintenant de la spécialité du chercheur.

Pour créer cette vue, il faut d'abord spécifier,pour chaque spécialité, le tarif demandé par un chercheur pour une journée :

- Ajouter une colonne TARIF (number) à la table SPECIALITE,
- Mettre à jour cette colonne avec les tarifs journaliers suivants :

specialite	→	tarif
bd	→	800
si	→	200
oo	→	700
rx	→	300
se	→	500

Créer la vue COUT_PROJET (n_projet, nom_projet, cout) en tenant compte de la spécialité et du nombre de jours de chaque chercheur qui travaillle dans le projet.

Vérifier le contenu. Vous devez avoir :

p1	objet-relationnel	5000
p2	intranet	3000
p3	sans fil	900
p4	groupware	800
p5	uml	800

(b) A partir de la vue, afficher les noms de projet dont le coût dépasse 1000 Euros. Faire la même requête (même résultat) sans utiliser la vue.

(c) Créer la vue TRAVAUX permettant d'obtenir toutes les chercheurs affectés (qui travaillent) dans chaque projet :

TRAVAUX(nom_du_projet, nom_du_chercheur, nombre_jours)

Vérifier.

3-Création et utilisation de vues mono-tables avec contraintes

La vue doit être crée avec l'option de contrôle : le prédicat sert alors de contrainte.

Exemple :

```
CREATE VIEW emp_info
AS SELECT * FROM emp
WHERE service='info'
        AND salaire BETWEEN 7500 AND 25000
WITH CHECK OPTION CONSTRAINT sal_emp_info;
```

(a) Reprendre la vue CHERCHEUR_2 de la première question en rajoutant l'option de contrôle.

(b) Tenter d'insérer un chercheur n'appartenant pas à une spécialté 'objet'.

(c) Créer la vue PROJET_E1 à partir de la table TRAVAILLER ne permettant de travailler qu'avec les chercheurs de l'équipe 'e1' et les projets d'un coût supérieur à 1000 Euros :

PROJET_E1(n_projet, n_chercheur, nb_jour_sem)

Tenter d'insérer deux tuples dans cette vue ne correspondant pas à ces deux contraintes :

.un chercheur d'une autre équipe que 'e1',
.un projet de coût < 1000 Euros.

4-Restriction des actions : GRANT

Travailler deux par deux. Chaque table crée est enregistrée dans le dictionnaire de données avec le nom de son propriétaire (owner) sous la forme : 'nom_propritaire.nom_table', par exemple : 'ora1.projet'. On appellera respectivement user1 et user2 les 2 utilisateurs travaillant ensemble (user1 et user2).

(a) User1 donne à User2 le droit de travailler sur sa table PROJET avec les privilèges suivants : select, insert, update(nom_projet).

(b) User2 fait une modification d'un nom de projet dans la table 'user1.projet'. User2 et User1 vérifient le contenu de la table. Que se passe-t-il et pourquoi ? User2 fait un 'commit'. User1 vérifie le contenu.

(c) User2 tente de supprimer une ligne dans la table 'User1.projet'.

(d) User1 révoque les droits accordés.

5-Résultats à rendre

.les requêtes demandées
.les requêtes avec exécution (trace),
.les nouvelles contraintes mises en oeuvre (user_constraints),
.la table user_views avec mise en page.