Applications Bases de Données

1)Les vues

1.1)Définition et utilisations

Une vue est une table virtuelle (relation temporaire)

-> Ses données ne sont pas matérialisées, ne sont pas physiquement stockées sur disque

-> On l'utilise comme une relation de la base dans des requêtes ou pour des mises à jour (INSERT,UPDATE,DELETE) qui sont « à travers » la vue dans la relation associée à la vue, avec certaines restrictions

-> Une vue est le résultat d'une requête

-> Les tuples de la vue sont calculés au moment où on utilise la vue

Syntaxe :

CREATE [OR REPLACE] VIEW nom_vue AS requetededéfinitondelavue

Exemple :

ETUDIANT (Num_Et, NOM_ET,ANNEE,GROUPE,DEPNT)

CREATE OR REPLACE VIEW Etud Info AS SELECT * FROM ETUDIANT

WHERE DEPNT = 'INFORMATIQUE '

Utilisation de Etud Info

1. SELECT * FROM Etud Info
   WHERE ANNEE = 2

   Au moment de l'exécution de la dernière requête, le système la combine avec la requête de définition de la vue. Donc ce qui est exécuté par le système est :

   SELECT * FROM ETUDIANT
   WHERE DEPNT = 'INFORMATIQUE' AND ANNEE = 2

2. INSERT INTO Etud Info

   VALUES(2101,'DUPONT',..2,1,'INFORMATIQUE ')

   Le tuple inséré à travers la vue Etud Info est physiquement stocké dans la table ETUDIANT et il apparait quand on consulte aussi bien Etud Info que ETUDIANT

3. INSERT INTO Etud Info
   VALUES(2102,'DURAND ',..,2,1,'GENIE MECANIQUE)

Le tuple est inséré dans la relation ETUDIANT et apparait lorsqu'on consulte ETUDIANT mais jamais quand on consulte la vue Etud Info car ils ne vérifie pas la condition DEPNT = 'INFORMATIQUE'

1.2 Objectif des vues

On utilise les vues pour 3 objectifs :

- Assurer la confidentialité des données : un utilisateur ne pourra accéder qu'aux données qu'il a le droit de consulter
- Facilité l'expression de requêtes complexe (pour son utilisateur final, on lui épargne GROUP BY, CONNECT BY ..)
- Vérifier les contraintes d'intégrité statiques (domaine, relation, référence) et dynamiques avec la classe WITH CHECK OPTION après la requête de définition de la vue

Avec WITH CHECK OPTION tous les tuples manipulés à travers la vue doivent respecter les conditions de la requête de définition de la vue.

Ex :

CREATE VIEW Etud Info 2 AS

SELECT * FROM ETUDIANT

WHERE DEPNT = 'INFORMATIQUE'

WITH CHECK OPTION

L'insertion suivante échoue car on ne respecte pas la condition DEPNT = 'Informatique'

INSERT INTO Etud Info 2

VALUES (2103,'DUMONT',..,2,1,'GENIE MECANIQUE')

1.1) Restrictions sur les mises à jour à travers

Les opérations de mises à jour travaillent sur une seule relation. Donc dans la requête de définition de la vue, il ne faut qu'une seule relation (mais on peut faire des jointures imbriquées)

Dans le 1 er bloc de la requête de définition de la vue, il est interdit de :

- D'utilises DISTINCT dans le SELECT car cela signifie avoir des duplicats donc ne pas projeter la clef primaire

- D'utiliser des clauses complexes comme GROUP BY (car on aggrège les données), ou CONNECT BY (equivalent a des auto jointures) ou ORDER BY
- D'utiliser les opérateur ensenblistes UNION, INTERSECT, MINUS car ils éliminent les duplicats
- Tous les attributs ayant été déclarés avec une contrainte NOTNULL doivent apparaitre dans la vue

1.4)Exemples :

- Vue donnant les effectifs par département et par année

```
CREATE OR REPLACE VIEW Effectifs AS
SELECT DEPNT,ANNEE,COUNT(*)
FROM ETUDIANT
GROUP BY DEPNT,ANNEE
```

  Elle n'est utilisable qu'en consultation puisqu'on a un GROUP BY dans le 1 er bloc

- Vue donnant la liste des étudiants des départements ayant le plus grand effectif

```
CREATE OR REPLACE VIEW Etud As
SELECT * FROM ETUDIANT
WHERE DEPNT IN
        (SELECT DEPNT FROM ETUDIANT
        GROUP BY DEPNT
        HAVING COUNT(*) >= ALL
            (SELECT COUNT(*) FROM
            ETUDIANT GROUP BY DEPNT)
```

  Cette vue peut être utilisé en consultation mais aussi en mise à jour car les GROUP BY sont dans des blocs imbriqués

- Vue permettant de vérifier une CI de domaine

```
CREATE OR REPLACE VIEW Dom-DEPNT AS
SELECT * FROM ETUDIANT
WHERE DEPNT IN ('INFORMATIQUE', 'TC','GEA ',…)
WITH CHECK OPTION
```

- Vue interdisant plus de 29 élèves par groupe

```
CREATE OR REPLACE VIEW Etud 4 AS
SELECT * FROM ETUDIANT
WHERE (GROUPE,ANNEE) IN
        (SELECT GROUPE,ANNEE
        FROM ETUDIANT
        GROUP BY GROUPE,ANNEE
        HAVING COUNT(*) < 30)
        WITH CHECK OPTION
```

1) Les triggers (déclencheurs)

Sémantique des triggers :

Quand un évenement survient
Si une condition est vérifiée
Alors une action est exécutée

Les triggers permettent de vérifier des contraintes d'intégrité automatiquement, de déclencher des alertes (par exemple des que la valeur d'un attribut atteint un certain seuil on déclenche une opération)

2.1) Evènement

Un évènement est la détection d'un ordre SQL par le système

-ordres du Langage de définition de données (LDD) : CREATE ,ALTER

-ordres du Langage de Contrôle de données (LCD) : GRANT , REVOKE

-ordres du Langage de Manipulation de données (LMD) : INSERT, UPDATE, DELETE

Chronologie entre évènement et actions

Pour un trigger de type BEFORE l'action est exécutée d'abord puis l'ordre SQL dont la détection à déclenché le trigger est exécuté.

-Granularité des triggers LMD déclenchés par la détection d'une mise à jour

Ils peuvent être :

-orienté ensemble : leur action est exécuté une et une seule fois pour tous les tuples concernés.

-Orientés tuple : leur action est executée pour chaque tuple concerné par la mise à jour.

2.2) Condition

C'est une condition simple portant sur une unique relation

Attribut comparateur constante.

On peut les combiner avec AND, OR

On peut utiliser les prédicats de sélection de SQL : IN, BETWEEN, Val1 AND Val2, LIKE, IS NULL et leur négation.

- SUITE AU PROCHAIN AMPHI