La gestion des vues

Définition d’une vue

- Une vue est une relation virtuelle calculée à partir des relations de base ou à partir d'autres vues à l'aide d'une sous-requête.

Avantages des vues

- Elles limitent l'accès aux données en affichant une sélection de colonnes de la table
- Elles permettent de créer des interrogations simples pour extraire les résultats d'interrogations complexes. (par exemple, l'utilisateur peut interroger plusieurs tables sans savoir écrire une instruction de jointure).
- Elles garantissent l'indépendance des données pour des utilisateurs et des programmes d'application.
- Elles présentent les mêmes données sous différentes vues.

Création d'une vue (1)

- **Syntaxe:**

  CREATE [OR REPLACE] [FORCE | NOFORCE] VIEW <nom-vue>
  [(alias [, alias].....)]
  AS <sous-requête>
  [WITH CHECK OPTION [CONSTRAINT <nom-contrainte>]]
  [WITH READ ONLY [CONSTRAINT <nom-contrainte>]]

Remarques:

- La sous-requête peut être simple ou complexe (jointure, sous-interrogation, groupement, ….).
- On peut afficher le schéma de la vue à l'aide de la commande DESCRIBE.
- La sous-requête ne doit pas contenir la clause ORDER BY

Création d'une vue (2)

- OR REPLACE: modifie la définition de la vue si elle existe déjà, sans la supprimer ni la recréer
- FORCE: crée la vue, que les tables de base existent ou non
- NOFORCE: ne crée la vue que si les tables de base existent déjà (valeur par défaut)
- VIEW: correspond au nom de la vue
- ALIAS: indique les noms des expressions sélectionnées par l'interrogation de la vue. Le nombre d'alias doit être égal au nombre d'expressions sélectionnées par la vue
- Sous-requête: introduit une instruction SELECT complète. On peut utiliser des alias de colonnes dans la liste SELECT
- WITH CHECK OPTION: n'autorise l'insertion et la mise à jour que pour les lignes auxquelles la vue peut accéder.
- Constraint: correspond au nom attribué à la contrainte CHECK OPTION
- WITH READ ONLY: garantit qu'aucune opération LMD ne peut être effectuée sur la vue.

Création de vue : exemple1

1) Créer la vue empvue01 qui doit contenir des informations sur les employés du département 01:

```
CREATE  VIEW empvue01
AS  SELECT  codemp, nom_emp, salaire
FROM employe
WHERE coddep = 01;
```

Création de vue : exemple2

- créer une vue contenant les informations suivantes de chaque employé du département 02: le code de l'employé avec l'alias de colonne numemp, le nom de l'employé avec l'alias nom, le salaire annuel avec l'alias salaire_annuel:

- CREATE VIEW empvue02
  AS SELECT codemp numemp , nom_emp nom , salaire* 12
  salaire_annuel
  FROM employe
  WHERE coddep = 02;

- Ou bien:
  CREATE VIEW empvue02(numemp, nom, salaire_annuel)
  AS SELECT codemp , nom_emp , salaire* 12
  FROM employe
  WHER e coddep = 02;

Opérations LMD sur les vues:

- Vous pouvez exécuter des opérations LMD (insert, update, delete) sur des vues simples
- Vous ne pouvez pas supprimer une ligne d'une vue si la vue contient :
  - Des fonctions de groupe
  - Une clause Group By
  - Le mot clé Distinct
  - La pseudo colonne ROWNUM
- Vous ne pouvez pas modifier les données d'une vue si elle contient :
  - Des fonctions de groupe
  - Une clause Group By
  - Le mot clé Distinct
  - La pseudo colonne ROWNUM

Opérations LMD sur les vues:

- Vous ne pouvez ajouter des données à la vue si elle contient :
  - Des fonctions de groupe
  - Une clause Group By
  - Le mot clé Distinct
  - La pseudo colonne ROWNUM
  - Des colonnes définies par des expressions arithmétiques (note* 3 par exemple)
  - Des colonnes NOT NULL se trouvant dans la table de base qui ne sont pas sélectionnées par la vue (toutes les valeurs doivent figurer dans la vue)

La clause WITH CHECK OPTION :

- on peut s'assurer que les opérations LMD effectuées sur la vue restent dans le domaine de la vue à l'aide de la clause WITH CHECK OPTION. La clause n'autorise pas la création ou la modification de lignes de la table de base que la vue ne peut pas sélectionner.

La clause READ ONLY :

- Elle permet de garantir qu'aucune opération LMD ne sera exécutée dans la vue.

Suppression d’une vue :

- Syntaxe : DROP VIEW<nom-vue>
- La suppression d’une vue n’a aucun effet sur les tables de la base

Les vues en ligne :

- Définition : une vue en ligne est une sous-requête intégrant un alias qu'on peut utiliser dans une instruction SQL. La sous-requête définit une source de données qui peut être référencée dans la requête principale.
- **Exemple :**

La requête principale suivante affiche le nom, le salaire, le code du département et le salaire maximum de tous les employés touchant un salaire inférieur au salaire maximum de leur département.

- **Requête:**

```
SELECT e.nomemp, e.salaire, e.coddep, d.salmax
FROM employe e, (SELECT coddep, max(salaire) salmax
                 FROM employe
                 GROUP BY coddep)  d
WHERE e.coddep = d.coddep
AND e.salaire <d.salmax
```

Requête de type n premiers:

- Les requêtes de type n premiers permettent d'identifier les n valeurs les plus petites ou les plus grandes dans une colonne
- Exemples :
  - quels sont les 10 produits les plus vendus ?
  - quels sont les trois étudiants de Master les mieux classés ?
  - quel est le dernier employé recruté dans la société ?

Requête de type n premiers:

- Syntaxe :

```
SELECT [liste de colonnes], ROWNUM
FROM (SELECT [liste de colonnes]
        FROM  < nom de table>
        ORDER BY top_N-colonne)
WHERE ROWNUM <= N;
```

Requête de type n premiers: Explication

- La sous requête ou la vue en ligne génère une liste de données triée à l'aide de la clause ORDER BY
- L'interrogation principale limite le nombre de lignes extraites par la vue en ligne. Elle comporte :
- Une pseudo-colonne ROWNUM qui affecte une valeur séquentielle commençant par 1 à chaque ligne renvoyée par la sous-requête
- Une clause WHERE qui indique le nombre n de lignes à renvoyer ; elle doit utiliser les opérateurs < ou <=

Requête de type n premiers : Exemple

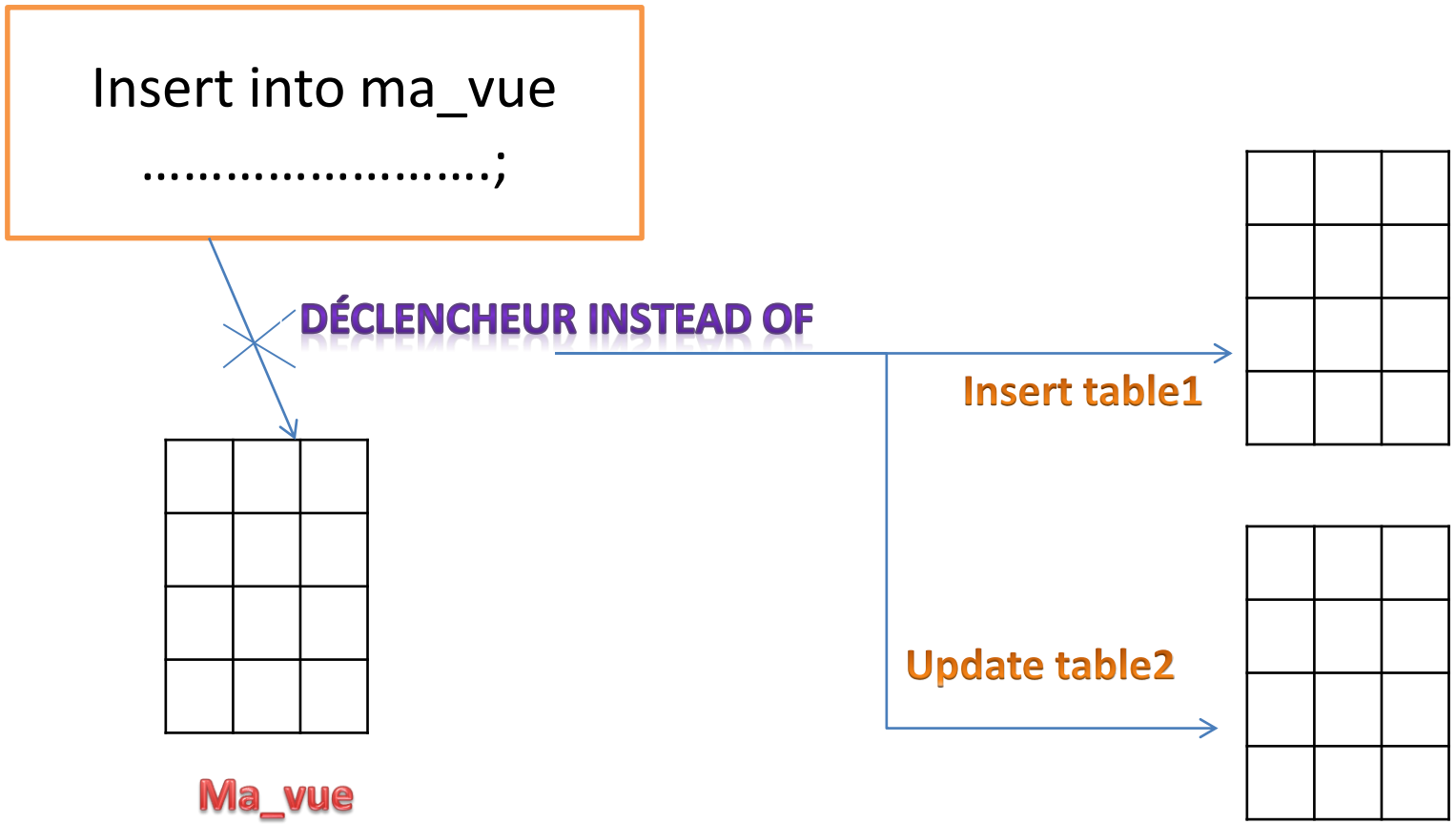
- Afficher le nom et le salaire des trois employés qui touchent les salaires les plus élevés :

```
Select ROWNUM as Rang, nomemp, salaire From
(select nomemp, salaire
        From employe
        Order by salaire DESC)
Where ROWNUM <= 3;
```

Déclencheur INSTEAD OF: Rôle

- Permet de modifier les données lorsqu'une instruction LMD est exécutée pour une vue non modifiable (vue complexe).
- permet d'effectuer directement les opérations LMD sur les tables sous-jacentes
- Le serveur ORACLE exécute alors le corps du déclencheur à la place de l'instruction de déclenchement.

Déclencheur INSTEAD OF: Rôle



Créer un déclencheur INSTEAD OF

Create [or replace] trigger <nom_trigger>

Instead of

Event1 [or event2 or event3]

On <nom_vue>

Referencing old as *old* | new as *new*]

[for each row]

Corps du trigger

/*Les options after et before ne sont pas valides*/

Créer un déclencheur INSTEAD OF

Instead of	Indique que le trigger appartient à une vue
Event	Identifie l'opération LMD qui provoque l'exécution du trigger : insert |update of column |delete
referencing	Identifie la correspondance entre les anciennes et les nouvelles valeurs de la ligne en cours (les valeurs par défaut sont old et new)
For each row	Indique que le trigger est un trigger sur ligne ; optionnelle car par déf, un trigger instead of est un trigger sur ligne
Corps du trigger	Acction à effectuer par le déclencheur; commence par begin et se termine par end ou un appel de procédure

Exemple de déclencheur instead of

- create table Etudiant (id Number (5) primary key,nom varchar2 (20)) ;
- create table Note (etudiant references Etudiant (id),note Number (5)) ;
- Soit la vue non modifiable à cause, entre autres, de l'utilisation de AVG :

```
create view Moyenne (id, nom, moyenne) as
 select e.id, e.nom, nvl (to_char (avg (n.note)), 'pas de note')
 from Etudiant e left outer join Note n on e.id = n.etudiant
   group by e.id, e.nom ;
```

Exemple de déclencheur instead of(2)

- On veut qu'un ordre LMD sur la vue Moyenne se traduise par un ordre LMD similaire sur la table Etudiant :
  - un **insert** insère simplement le nouvel étudiant dans la table Etudiant,
  - un **update** met à jour uniquement le nom de l'étudiant (on pourrait aussi tenter de mettre à jour son id mais cela poserait des problèmes à cause de la clef étrangère de Note)
  - un **delete** supprime les notes de l'étudiant old.id puis l'étudiant.

Exemple: code du trigger instead of

```
create trigger LMD_sur_Moyenne
  instead of insert or delete or update
  on Moyenne
  for each row
begin
  if inserting then
    insert into Etudiant values (:new.id, :new.nom) ;
  elsif updating ('nom') then
    update Etudiant set nom = :new.nom where id = :old.id ;
  elsif deleting then
    delete from Note where etudiant = :old.id ;
    delete from Etudiant where id = :old.id ;
  end if ;
end ;
```