Les vues et les règles

Comment protéger des tables

LES VUES

Présentation

- **Introduction**
  - **Les vues permettent :**
    - **de simplifier le schéma relationnel ;**
    - **D'affiner les privilèges ;**
    - **De cacher certaines colonnes d'une table ;**
    - **Regrouper des informations au sein d'une entité**

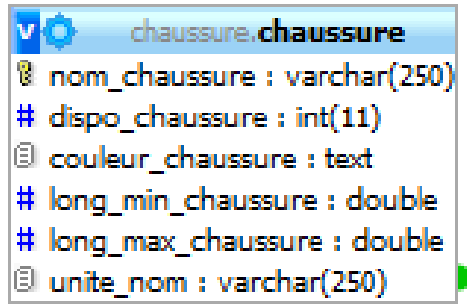
Création de vues

Syntaxe de création d'une vue

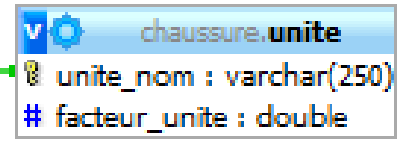
```
CREATE  [ OR REPLACE ] [ TEMP | TEMPORARY ]
VIEW nom [ (nom de colonne [, ...] ) ]
AS REQUÊTE
```

- ***TEMPORARY ou TEMP*** **La vue est temporaire., elles sont automatiquement supprimées en fin de session.**
- ***nom*** **Le nom de la vue à créer , il ne peut être le même que le nom d'une table**
- ***nom de colonne*** **Une liste optionnelle de noms à utiliser pour les colonnes de la vue. Si elle n'est pas donnée, le nom des colonnes est déduit de la requête.**
- ***requête*** **Une commande SELECT qui fournira les colonnes et lignes de la vue.**

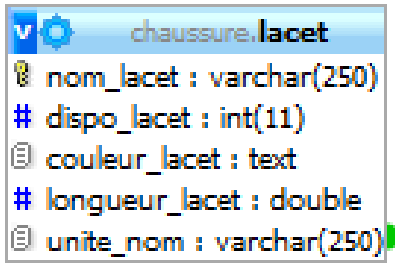
Schéma de données des exemples



nom_chaussure	dispo_chaussure	couleur_chaussure	long_min_chaussure	long_max_chaussure	unite_nom
sh1	2	black	70	90	cm
sh2	0	black	30	40	inch
sh3	4	brown	50	65	cm
sh4	3	brown	40	50	inch



unite_nom	facteur_unite
cm	1
m	100
inch	2.54



nom_lacet	dispo_lacet	couleur_lacet	longueur_lacet	unite_nom
sl1	5	black	80	cm
sl2	6	black	100	cm
sl3	0	black	35	inch
sl4	8	black	40	inch
sl5	4	brown	1	m
sl6	0	brown	0.9	m
sl7	7	brown	60	cm
sl8	1	brown	40	inch

Exemple de création de vues

- **Création d'une vue qui permet d'avoir les longueurs min et max des chaussures en centimètre**

```
CREATE VIEW view_chaussure_cm
AS SELECT nom_chaussure, dispo_chaussure, couleur_chaussure,
     long_min_chaussure, long_max_chaussure, s.unite_nom,
     long_min_chaussure * facteur_unite AS long_min_chaussure_cm,
     long_max_chaussure * facteur_unite AS long_max_chaussure_cm
   FROM chaussure s JOIN unite u ON s.unite_nom = u.unite_nom;

SELECT *
FROM view_chaussure_cm ;
```

nom_chaussure	dispo_chaussure	couleur_chaussure	long_min_chaussure	long_max_chaussure	unite_nom	long_min_chaussure_cm	long_max_chaussure_cm
sh1	2	black	70	90	cm	70	90
sh2	0	black	30	40	inch	76.2	101.6
sh3	4	brown	50	65	cm	50	65
sh4	3	brown	40	50	inch	101.6	127

Exemple de création de vues

- **La même mais en changeant le nom des colonnes**

ATTENTION impossible de remplacer une vue existante en lui changeant le nom des colonnes

```
CREATE OR REPLACE VIEW view_chaussure_cm (nom, stock, couleur, long_min, long_max,
                                                    unite,long_min_cm,long_max_cm)
AS SELECT nom_chaussure, dispo_chaussure, couleur_chaussure,
     long_min_chaussure, long_max_chaussure, s.unite_nom,
     long_min_chaussure * facteur_unite AS long_min_chaussure_cm,
     long_max_chaussure * facteur_unite AS long_max_chaussure_cm
   FROM chaussure s JOIN unite u ON s.unite_nom = u.unite_nom;

SELECT *
FROM view_chaussure_cm ;
```

nom	stock	couleur	long_min	long_max	unite	long_min_cm	long_max_cm
sh1	2	black	70	90	cm	70	90
sh2	0	black	30	40	inch	76.2	101.6
sh3	4	brown	50	65	cm	50	65
sh4	3	brown	40	50	inch	101.6	127

Utilisation d'une vue

- **L'interrogation de données**
  - **Une vue peut être interrogée via une requête SELECT au même titre que les tables**
  - **Exemple**
    - **Requête simple**

```
SELECT nom,stock,couleur
FROM view_chaussure_cm
WHERE long_max_cm <=90 ;
```

nom	stock	couleur
sh1	2	black
sh3	4	brown

Utilisation d'une vue

- L'interrogation de données
  - Exemple
    - Requête avec jointure

```
SELECT nom,stock,couleur,unite, facteur_unite
FROM view_chaussure_cm JOIN unite ON unite=unite_nom
WHERE long_max_cm <=90 ;
```

nom	stock	couleur	unite	facteur_unite
sh1	2	black	cm	1
sh3	4	brown	cm	1

Utilisation d'une vue

- L'interrogation de données
  - Exemple
    - Requête avec sous requête

```
SELECT nom, stock, couleur, unite
FROM view_chaussure_cm Vc
WHERE exists ( SELECT *
               FROM lacet
               WHERE couleur_lacet= Vc.couleur) ;
```

nom	stock	couleur	unite
sh1	2	black	cm
sh2	0	black	inch
sh3	4	brown	cm
sh4	3	brown	inch

Utilisation d'une vue

- **Mise à jours, suppression et insertion de données**
  - **les vues sont en lecture seule : le système n'autorise pas une insertion, une mise à jour ou une suppression sur une vue.**
  - **Si l'on veut actualiser (insert, update ou delete) des données des tables à travers les vues il faut créer des règles (RULES) sur les vues .**

Utilisation d'une vue

- **Mise à jours, suppression et insertion de données**
  - **Sans ce mécanisme de règles (RULES) sur les vues, si l'on tente de d'actualiser une vue on aura l'erreur suivante :**

```
UPDATE view_chaussure_cm
SET stock = 13
WHERE nom = 'sh4';
```

```
Erreur SQL :

ERREUR:  ne peut pas mettre à jour la vue « view_chaussure_cm »
HINT:  Vous avez besoin d'une règle non conditionnelle ON UPDATE DO INSTEAD ou d'un trigger INSTEAD OF UPDATE.
```

Suppression d'une vue

- **Supprimer une vue**

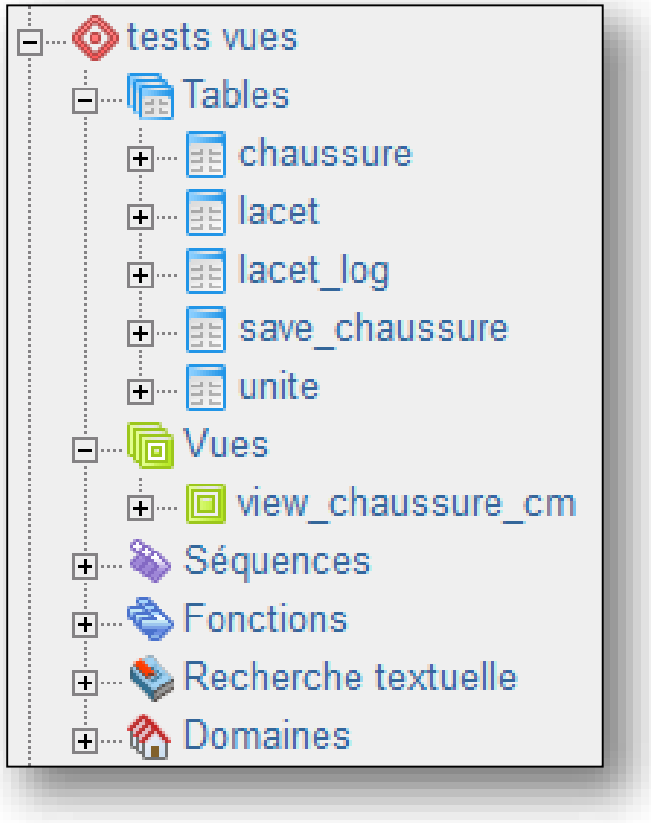
```
DROP VIEW  VIEW_NAME [, VIEW_NAME] ...
```

Exemple :

```
DROP VIEW view_chaussure_cm;
```

Les vues sous phppgadmin

- **Une vue est un objet situé au même niveau que les tables**



LES RÈGLES

Les règles

Présentation

- **Le système de règles autorise la définition d'actions alternatives sur les insertions, mises à jour ou suppressions dans les tables ou les vues.**
- **Une règle permet d'imposer des commandes supplémentaires, de remplacer les commandes par d'autres, voire d'empêcher sa réalisation lors de l'exécution d'une instruction sur une table ou une vue donnée.**
- **Une règle est déclenchée sur un des événements INSERT, UPDATE ou DELETE**

Syntaxe de création d'une règle

```
CREATE [ OR REPLACE ] RULE nom
AS ON événement TO table ou vue [ WHERE condition ]
DO [ ALSO | INSTEAD ]
{commande | (commande; commande..) }
```

nom : **Le nom de la règle à créer.**

événement : **SELECT, INSERT, UPDATE ou DELETE.**

condition : **Toute expression SQL conditionnelle (renvoyant un type boolean). L'expression de la condition ne peut pas faire référence à une table autre que NEW ou OLD ni contenir de fonction d'agrégat.**

INSTEAD : **Les commandes sont exécutées à la place de la commande originale.**

ALSO : **Les commandes sont exécutées *en plus de* la commande originale (par défaut).**

Les structures OLD et NEW

- **Elles sont automatiquement créées par le système sur une action de type UPDATE, DELETE ou INSERT sur une table, lors du déclenchement d'une règle ou d'un trigger.**

- **Elles ont la structure de la table sur laquelle à lieu l'action**

- **La structure OLD**
  - **Existe après un UPDATE ou un DELETE**
  - **Contient la ligne qui a été supprimé ou celle avant modification**

- **La structure NEW**
  - **Existe après un UPDATE ou un INSERT**
  - **Contient la ligne qui a ajouter ou celle après modification**

Les structures OLD et NEW

- **Exemple sur un UPDATE**

```
UPDATE chaussure
SET dispo = 12,
    couleur_chaussure ='Red'
WHERE nom_chaussure = 'sh1';
```

table avant le update

nom_chaussure	dispo_chaussure	couleur_chaussure
sh1	2	black
sh2	0	black
sh3	4	brown
sh4	3	brown

- **Pour accéder à la colonne d'une des structures OLD ou NEW**

OLD.dispo_chaussure contient *2*
OLD.couleur_chaussure contient *black*
et
NEW.dispo_chaussure contient *12*
NEW.couleur_chaussure contient *Red*

Les structures OLD et NEW

- **Exemple sur un INSERT**

```
INSERT INTO chaussure values ('sh5', 12,'red',45,69,'cm');
```

nom_chaussure	dispo_chaussure	couleur_chaussure	long_min_chaussure	long_max_chaussure	unite_nom
sh1	2	black	70	90	cm
sh2	0	black	30	40	inch
sh3	4	brown	50	65	cm
sh4	3	brown	40	50	inch
sh5	12	red	45	69	cm

- **Pour accéder à la colonne de la structure NEW**

```
NEW.dispo_chaussure contient 12
NEW.couleur_chaussure contient Red
NEW.nom_chaussure contient sh5
……
```

La structure OLD n'existe pas sur un INSERT

Les structures OLD et NEW

- **Exemple sur un DELETE**

```
DELETE FROM chaussure
WHERE nom_chaussure = 'sh5';
```

- **Pour accéder à la colonne de la structure OLD**

```
OLD.dispo_chaussure contient 12
OLD.couleur_chaussure contient Red
OLD.nom_chaussure contient sh5
……
```

La structure NEW n'existe pas sur un DELETE

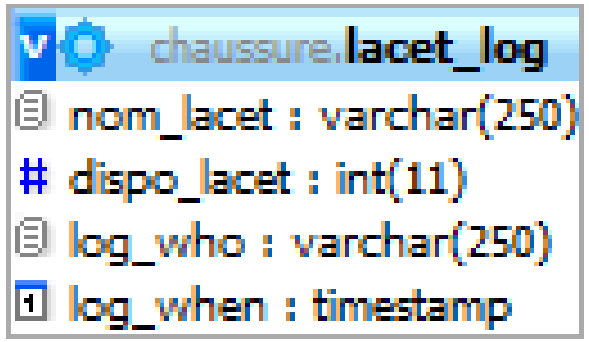
Exemple de règle sur une table

- **Exemple 1 : ajout d'action avec condition**

Nous voulons tracer les modifications dans la colonne *dispo_lacet* de la table *lacet*.

Créons une table de log et une règle qui va écrire une entrée lorsqu'un *update* est lancé sur la table *lacet*.

La table lacet_log



Exemple de règle sur une table

- **Exemple 1 : ajout d'action avec condition**

  Création de la règle sur la table lacet

```
CREATE RULE log_lacet AS ON UPDATE TO lacet
WHERE NEW.dispo_lacet <> OLD.dispo_lacet
DO
INSERT INTO lacet_log
  VALUES (NEW.nom_lacet,NEW.dispo_lacet,current_user,current_timestamp);
```

La règle *log_lacet* sera déclenchée sur un UPDATE sur la table *lacet*, uniquement si la valeur de la colonne *dispo_lacet* a changé, alors en plus de l'action sur la table *lacet* on ajoute une ligne dans la table des logs (lacet_log)

Exemple de règle sur une table

- **Exemple 1 : ajout d'action avec condition**

  Mise à jour sur la table lacet

```
UPDATE lacet
SET dispo_lacet = 12,
    couleur_lacet ='Red',
    longueur_lacet=50
WHERE nom_lacet = 'sl8';
```

nom_lacet	dispo_lacet	couleur_lacet	longueur_lacet	unite_nom
sl1	5	black	80	cm
sl2	6	black	100	cm
sl3	0	black	35	inch
sl4	8	black	40	inch
sl5	4	brown	1	m
sl6	0	brown	0.9	m
sl7	7	brown	60	cm
sl8	12	Red	50	inch

Le update est appliqué sur la table *lacet* et une ligne a été ajouté dans la table *lacet_logs*

nom_lacet	dispo_lacet	log_who	log_when
sl8	12	postgres	2014-03-20 18:16:11.158

Exemple de règle sur une table

- **Exemple 1 : ajout d'action avec condition**

  On peut mettre autant d'action que nécessaire dans la règle

```
CREATE RULE log_lacet AS ON UPDATE TO lacet
WHERE NEW.dispo_lacet <> OLD.dispo_lacet
DO
( INSERT INTO lacet_log
        VALUES ( NEW.nom_lacet, NEW.dispo_lacet,
                 current_user, current_timestamp );
  SELECT * FROM lacet_log;
)
```

Exemple de règle sur une table

- **Exemple 2 : remplacer les actions de mises à jour avec condition**

Nous allons dérouter l'action d'insertion si la valeur de la clé primaire existe dans la table *unite*.

Dans ce cas nous ferons une mise à jour sur la ligne de la table *unite*

```
CREATE RULE insert_unite AS ON INSERT TO unite
WHERE exists ( SELECT * FROM unite
                WHERE unite_nom=NEW.unite_nom)
DO INSTEAD
 UPDATE unite
  SET facteur_unite=NEW.facteur_unite
  WHERE unite_nom=NEW.unite_nom
```

Exemple de règle sur une table

- **Exemple 2 : remplacer les actions de mises à jour avec condition**

  Insertion d'une donnée valide :

unite_nom	facteur_unite
cm	1
m	100
inch	2.54

```
INSERT INTO unite VALUES('mm',0.01);
```

La règle n'est pas déclenchée l'insertion se déroule normalement

unite_nom	facteur_unite
cm	1
m	100
inch	2.54
mm	0.01

Exemple de règle sur une table

- **Exemple 2 : remplacer les actions de mises à jour avec condition**

  Insertion d'une donné avec doublon sur la clé primaire :

```
INSERT INTO unite VALUES('mm',0.1);
```

unite_nom	facteur_unite
cm	1
m	100
inch	2.54
mm	0.01

La règle est déclenchée,l'insertion n'a pas lieu, mais la régle s'applique et la mise à jour de l'unite prend la place de l'insertion

unite_nom	facteur_unite
cm	1
m	100
inch	2.54
mm	0.1

Création d'une règle sur une vue

- **Limitation des règles sur les vues**
  - **l'utilisation de règles conditionnelles pour la mise à jour de vues n'est pas possible.**
  - **Chaque action autorisée sur la vue *doit* correspondre une règle INSTEAD inconditionnelle.**
  - **Si la règle est conditionnelle ou n'est pas une règle INSTEAD, alors le système rejette toute tentative de mise à jour.**

Création d'une règle sur une vue

- **Limitation des règles sur les vues**
  - **Exemple à ne pas faire**

```
CREATE or replace RULE update_view_chaussure
AS ON UPDATE TO view_chaussure_cm
WHERE NEW.nom = OLD.nom
DO
    UPDATE chaussure
    SET dispo_chaussure = new.stock,
     long_min_chaussure = new.long_min,
     long_max_chaussure = new.long_max;
```

Création d'une règle sur une vue

- **Limitation des règles sur les vues**
  - **Exemple à ne pas faire**

```
UPDATE view_chaussure_cm
SET stock = 21,
    long_min = 60
WHERE nom = 'sh6';
```

Suite à la définition de la règle précédente cette requête générera l'erreur suivante :

```
Erreur SQL :

ERREUR:  ne peut pas mettre à jour la vue « view_chaussure_cm »
HINT:  Vous avez besoin d'une règle non conditionnelle ON UPDATE DO INSTEAD ou d'un trigger INSTEAD OF UPDATE.
```

Création d'une règle sur une vue

- Limitation des règles sur les vues
  - Exemple à ne pas faire

```
CREATE or replace RULE update_view_chaussure
AS ON UPDATE TO view_chaussure_cm
WHERE NEW.nom = OLD.nom
DO
    UPDATE chaussure
    SET dispo_chaussure = new.stock,
    long_min_chaussure = new.long_min,
    long_max_chaussure = new.long_max;
```

Pas de condition et doit être obligatoirement être INSTEAD

Création d'une règle sur une vue

- **Limitation des règles sur les vues**
  - **Exemple à ne pas faire**

Doit être remplacé par :

```
CREATE or replace RULE update_view_chaussure
AS ON UPDATE TO view_chaussure_cm
DO INSTEAD
    UPDATE chaussure
    SET dispo_chaussure = new.stock,
        long_min_chaussure = new.long_min,
        long_max_chaussure = new.long_max
    WHERE nom_chaussure = OLD.nom;
```

Les règles sous phppgadmin

- **Les règles sont des objets contenu dans les tables ou les vues, en effet les règles s'appliquent à une et seule vue ou table**