

SGBD 1

SupNum

Les Vues

Les Vues

- ❑ Les vues en deux mots : des tables virtuelles
- ❑ Les vues en une phrase : une vue est une table qui est le résultat d'une requête (SELECT) à laquelle on a donné un nom
- ❑ Le nom d'une vue peut être utilisé partout où on peut mettre le nom d'une table : *SELECT, UPDATE, DELETE, INSERT, GRANT*

Quatre raisons d'utiliser des vues

- 1) Effet macro : remplacer une requête compliquée par des requêtes plus simples
- 2) Confidentialité
- 3) Contraintes d'intégrité
- 4) Augmenter l'indépendance logique

1) Effet macro

Remplacer une requête compliquée par des requêtes plus simples

2) Confidentialité :

exemple

```
CREATE VIEW emprunteurRestreint AS  
SELECT login, nomClient, prenomClient FROM  
client
```

3) Contraintes d'intégrité (CHECK OPTION) :

```
CREATE VIEW anciensExemplaires  
AS SELECT * FROM Exemple WHERE  
    numExemple < 2000  
    WITH CHECK OPTION;
```

```
UPDATE anciensExemplaires SET  
= 3812 WHERE numExemple = 1318;
```

Sans 'WITH CHECK OPTION', c'est possible.
Avec 'WITH CHECK OPTION', c'est impossible.

4) Augmenter l'indépendance logique

Les applications utilisant les tables de la base ne doivent pas être modifiées si on change le schéma de la base

Conditions de mise à jour pour les vues

Pour **UPDATE, DELETE, INSERT** la vue ne doit pas contenir :

- Un opérateur ensembliste (**UNION, MINUS, INTERSECT**)
- Un opérateur **DISTINCT**
- Une fonction d'agrégation comme attribut
- Une clause **GROUP BY**
- En cas de jointure :
 - La suppression n'est pas possible
 - Les opérations de mise à jour et l'insertion marchent si les données insérées ou mise à jour concernent une et une seule table.

Conditions de mise à jour pour les vues

Pour **UPDATE, DELETE, INSERT** la vue ne doit pas contenir :

- Les colonnes résultats de l'ordre SELECT doivent être des colonnes réelles d'une table de la base et non des expressions
- Si la vue est construite à partir d'une autre vue, cette dernière doit elle-même vérifier les conditions ci-dessus

Les Vues

Les vues sont des objets de la base de données, constitués d'**un nom** et d'une **requête de sélection**.

il arrive régulièrement que l'on veuille trouver des renseignements sur nos étudiants, y compris leurs notes dans chaque élément de module.

```
SELECT e.matriculeetudiant, nom, prenom, d.nom, n.note FROM etudiant e left  
join notespartielles n on e.matriculeetudiant = n.matriculeetudiant;
```

Il serait donc bien pratique de pouvoir stocker cette requête plutôt que de devoir la retaper en entier à chaque fois.

C'est très exactement le principe d'une vue : on stocke une requête SELECT en lui donnant un nom, et on peut ensuite appeler directement la vue par son nom.

CREATE VIEW

```
CREATE [OR REPLACE] VIEW nom_vue  
AS requete_select;
```

Exemple :

```
CREATE OR REPLACE VIEW v_Note AS  
  
SELECT e.matriculeetudiant, nom, prenom, d.nom,  
n.note FROM etudiant e left join notespartielles n on  
e.matriculeetudiant = n.matriculeetudiant;
```

Dorénavant, plus besoin de retaper cette requête, il suffit de travailler à partir de la vue, comme s'il s'agissait d'une table :

```
SELECT matriculeetudiant, nom, prenom, note  
FROM v_Note  
  
WHERE matriculeetudiant IN (23001,21004,23005) ;
```

Maintenant en consultant les tables de notre BD, on y retrouve notre vue:

```
Mysql > show tables;
```

CREATE VIEW (suite)

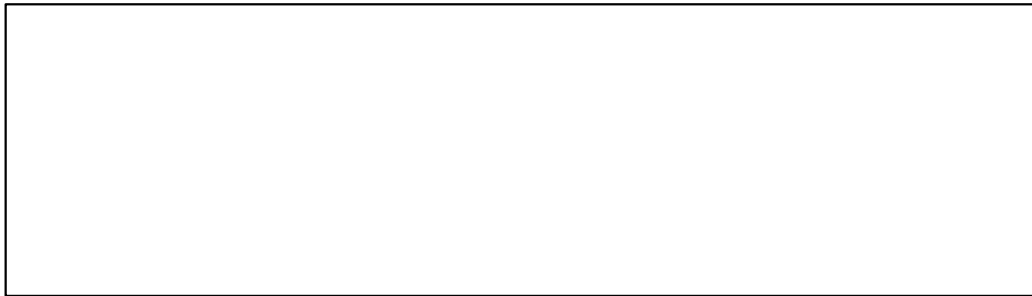
Ou bien

```
CREATE VIEW v_etudiant(matricule, etu_nom, etu_note)
```

```
AS SELECT e.matriculeetudiant, nom, n.note FROM etudiant e left join  
notespartielles n on e.matriculeetudiant = n.matriculeetudiant;
```

Faite desc **v_etudiant** pour voir la description de **v_etudiant**.

Creer une vue qui affiche le nombre d'étudiant par matière:



```
Mysql> select * from  
vNbreEtuParMatiere ;
```

CREATE VIEW (suite)

```
CREATE VIEW V_etu  
AS SELECT *  
FROM etudiant;
```

Une fois cette vue créée, elle est figée, et l'ajout d'une nouvelle colonne dans la table *etudiant* **ne changera pas** les résultats d'une requête sur *V_etu*: Cette vue ne sélectionnera jamais la nouvelle colonne.

Si on modifie la vue *v_etu*, la table *etudiant* est modifiée en conséquence et vice-versa.

Pour supprimer une vue, on écrit: `DROP VIEW nom_vue;`

Pour renommer une vue: `RENAME ancien-nom TO nouveau-nom`