



Bases de données

SQL - vues



Sommaire

- 1. Définitions, principe
- 2. Création d'une vue
- 3. Interrogation d'une vue
- 4. Modification et suppression d'une vue
- 5. Modification des données via une vue
- 6. Vues matérialisées



1.1 Vue : définition

■ Vue

- = résultat qui a reçu un nom...
- ... d'une requête `SELECT` stockée dans le SGBD

- Assimilable à une (sous-)requête nommée...
 - résultat d'une sous-requête = table
 - vue = table
 - obtenue par requête
 - pour être utilisée dans une autre requête
 - cf. résultat de sous-requête



1.2 Principe

- **Vue = table virtuelle**
 - stockage
 - de la requête qui définit la vue
 - (re-) calcul
 - de la table correspondante à chaque utilisation

→ table à jour, mais temps de calcul...

- **Vue matérialisée = table physique**
 - Stockage
 - du résultat de la requête qui définit la vue

→ rapidité d'accès, mais mise à jour ?



1.3 Intérêt

- **Simplifier l'écriture des requêtes**
 - Vue = table intermédiaire (cf. sous-requête)
 - Renommage de tables/colonnes
- ➔ écriture plus intuitive de certaines requêtes
 - cf. programmation modulaire (fonctions)



1.3 Intérêt

- **Structurer les données de façon plus intuitive**
 - Restriction des données
 - ex : vue des pilotes de la saison 2014
 - Jointures
 - ex : vue des films avec nom et prénom des réalisateurs
 - Renommages
 - ex : table avec noms de colonnes en français
 - ...



1.3 Intérêt

- **Restreindre l'accès = protection des données**
 - granularité du droit d'accès SELECT = table
 - colonnes confidentielles ?
 - → vue sans ces colonnes !
 - pour consultation, modification...
- **Exemple**
 - Adresse, téléphone, email d'un membre du club...
 - présents dans la table des membres de la BD, pour gestionnaire
 - supprimés de la vue accessible aux autres membres



1.3 Intérêt

- **Créer des rapports / résumés sur les données**
 - Ajouts de calculs horizontaux
 - ex : âge des pilotes calculé à partir de la date de naissance
 - Ajout de calculs verticaux
 - ex : total de points de la saison...
... ajouté à la vue des pilotes de courses de la saison 2014



1.3 Intérêt

- Faciliter interface Programme / Données

- Permettre certaines modification de structure...
... sans avoir à refaire le programme

- Exemple

- programme qui utilise Anaissance de artiste
 - si Anaissance transformée en « date » de naissance
→ programme HS
- programme qui utilise Anaissance d'une **vue sur** artiste
 - si Anaissance transformée en « date » de naissance
 - + modification de la vue (calc Anaissance à partir de « date »)
→ programme reste fonctionnel



Sommaire

- 1. Définitions, principe
- 2. Création d'une vue
- 3. Interrogation d'une vue
- 4. Modification et suppression d'une vue
- 5. Modification des données via une vue
- 6. Vues matérialisées



2.1 Requête de création de vue

- Syntaxe simplifiée

```
CREATE VIEW <nom_vue> AS  
SELECT ...
```

- <nom_vue>
 - se forme comme un nom de table
 - conseil : préfixer par « **v_** »
 - ex : **v_racedrivers2014**
- **SELECT** ...
 - requête de définition de la vue
 - peut porter sur tables et autres vues de la BD



2.2 Création ou modification d'une vue

- Si la vue existe déjà ?
 - → message d'erreur, et requête non exécutée
 - pour éviter ce problème : requête de création/modification
- Syntaxe « création/modification » de vue

```
CREATE OR REPLACE VIEW <nom_vue> AS  
SELECT ...
```

- Si la vue existe, elle est redéfinie



2.3 Choix des noms de colonnes

- **Noms des colonnes**
 - ceux de la table renvoyée par SELECT
 - avec éventuel renommage par alias
 - (`SELECT *` ... au moment de la création de la vue : pas recalculé !)
 - ou donnés dans une liste après le nom de la vue
- **Syntaxe avec la liste des noms de colonnes**

```
CREATE OR REPLACE VIEW <nom_vue> (<liste_noms>) AS  
SELECT ...
```

- (<liste_noms>)
 - noms distincts, séparés par des virgules (et entre parenthèses)
 - appariement dans l'ordre d'écriture
 - taille de la liste = nombre de colonnes renvoyées par la requête



2.4 Requête de définition

- Requête **SELECT**
= requête standard avec restrictions
 - pas de sous-requête dans le **FROM**
 - **ORDER BY**
 - donne l'ordre des lignes « par défaut »
 - peut être redéfini par la requête qui utilise la vue
 - **SELECT DISTINCT**
 - donne des résultats imprévisibles : à éviter !



2.5 Droits sur la vue

- **Droits lors de l'utilisation d'une vue : 2 modes**
 - **DEFINER**
 - contexte de l'utilisateur qui a défini la vue
 - par défaut
 - ex : accès à des données protégées pour des calculs verticaux
 - **INVOKER**
 - contexte de l'utilisateur qui utilise la vue
 - à préciser lors de la création de la vue
 - plus restrictif...
- **Serveur pedago de l'université**
 - problème de reconnaissance du **DEFINER**...



2.5 Droits sur la vue

- **Syntaxe avec précision du contexte des droits**
 - préciser **SQL SECURITY** suivi de **DEFINER** ou **INVOKER**

```
CREATE OR REPLACE SQL SECURITY INVOKER VIEW <nom_vue> AS  
SELECT ...
```

- mode à utiliser en TD



2.6 Exemples

- **Vues monotables**

- artistes français

```
CREATE VIEW v_artiste_fr AS
SELECT *
FROM artiste
WHERE Anat = 'FRANCE'
```

- artistes avec juste nom, prénom, âge

```
CREATE VIEW v_artiste_npa (nom, prenom, age) AS
SELECT Anom, Aprenom, YEAR(CURRENT_DATE()) - Anaissance
FROM artiste
```



2.6 Exemples

■ Vues multi-tables

- films avec nom et prénom du réalisateur au lieu de l'id

```
CREATE VIEW v_film_real AS
SELECT FilmID, Ftitre, Anom, Aprenom, Fgenre,
       Fnat, Fduree, Fannee, FRemakeDe
FROM film JOIN artiste ON FrealisateurID = ArtisteID
```

- Id, titre, nationalité des films avec nb de notes et moyenne

```
CREATE VIEW v_film_notes
      (id, titre, nat, nbnotes, moyenne) AS
SELECT FilmID, Ftitre, Fnat, COUNT(Nnote), AVG(Nnote)
FROM film LEFT JOIN note ON filmID = NfilmID
GROUP BY FilmID
ORDER BY AVG(Nnote)
```



Sommaire

- 1. Définitions, principe
- 2. Création d'une vue
- 3. Interrogation d'une vue
- 4. Modification et suppression d'une vue
- 5. Modification des données via une vue
- 6. Vues matérialisées



3.1 Interrogation d'une vue

- Une vue s'utilise comme une table !
- Interrogation d'une vue
 - ➔ par une requête `SELECT` qui utilise la vue



3.2 Exemples

- Titre et nom du réalisateur des drames

```
SELECT Ftitre, Anom  
FROM v_film_real  
WHERE Fgenre = 'Drame'
```

- Nom des artistes et leur âge

```
SELECT nom, age AS 'âge'  
FROM v_artiste_npa  
ORDER BY age DESC
```

- Exercice
donner les requêtes équivalentes sans utiliser de vue



3.2 Exemples

- Le (les ?) film américain qui a eu le plus de notes

```
SELECT titre
FROM v_film_notes
WHERE nbnotes = (SELECT MAX(nbnotes)
                 FROM v_film_notes)
AND nat = 'USA'
```

- Exercice
donner la requête équivalente sans utiliser la vue...



Sommaire

- 1. Définitions, principe
- 2. Création d'une vue
- 3. Interrogation d'une vue
- 4. Modification et suppression d'une vue
- 5. Modification des données via une vue
- 6. Vues matérialisées



4.1 Modification d'une vue

- Utilisation de la syntaxe « création/modification »

```
CREATE OR REPLACE VIEW <nom_vue> (<liste_noms>) AS  
SELECT ...
```




4.2 Suppression d'une vue

- **Syntaxe simple**

```
DROP VIEW <nom_vue>
```

- produit une erreur si la vue n'existe pas

- **Syntaxe alternative**

```
DROP VIEW IF EXISTS <nom_vue>
```

- ne fait rien si la vue n'existe pas



Sommaire

- 1. Définitions, principe
- 2. Création d'une vue
- 3. Interrogation d'une vue
- 4. Modification et suppression d'une vue
- 5. Modification des données via une vue
- 6. Vues matérialisées



5.1 Mise à jour des données d'une vue

- **Màj possible si**

- pas de **DISTINCT** dans la requête de définition
 - pas de calculs verticaux
 - pas de **GROUP BY** (ni **HAVING**)
 - pas d'**UNION**
 - pas de sous-requête dans **SELECT**
-
- et s'il y a des jointures, les mises à jour ne peuvent porter que sur une seule table



5.1 Mise à jour des données d'une vue

- **Requête de mise à jour**
 - = requête **UPDATE** sur la vue
 - porte sur les attributs présents dans la vue...
... provenant d'UNE SEULE table de la BD



5.2 Insertion de données

- **Mêmes restrictions que pour la mise à jour... plus...**
 - pas de calculs horizontaux
 - colonnes de la table sous-jacente doivent être **toutes** valuées
 - par valeurs fournies
 - par valeurs par défaut
 - par auto-incrément
- **Requête d'insertion**
 - = requête **INSERT INTO** la vue



5.3 Option de vérification de la vue

- **Restrictions sur les valeurs des données**
 - dans clause **WHERE** de la requête de définition
 - données de la table qui ne respectent pas les restrictions pas visibles dans la vue

- **Ajout/màj qui ne respectent pas ces restrictions**
 - = ajout/màj de données dans une table...
... via la vue ...
... et qui ne sont pas visibles dans la vue !

 - interdire ces ajouts/màj est légitime !



5.3 Option de vérification de la vue

- Syntaxe de création avec vérification des restrictions du **WHERE**

```
CREATE OR REPLACE VIEW <nom_vue> (<liste_noms>) AS  
SELECT ...  
...  
WITH CHECK OPTION
```



5.3 Option de vérification de la vue

■ Exemple

```
CREATE OR REPLACE VIEW v_film_fr AS  
SELECT *  
FROM film  
WHERE Fnat = 'FRANCE'
```

Permet l'ajout de
n'importe quel film

```
CREATE OR REPLACE VIEW v_film_fr AS  
SELECT *  
FROM film  
WHERE Fnat = 'FRANCE'  
WITH CHECK OPTION
```

Permet l'ajout
seulement de
films français



Sommaire

- 1. Définitions, principe
- 2. Création d'une vue
- 3. Interrogation d'une vue
- 4. Modification et suppression d'une vue
- 5. Modification des données via une vue
- 6. Vues matérialisées



6. Vues matérialisées

- Partie donnée oralement...



Questions ?
