SQL : vues

Achref El Mouelhi

Docteur de l'université d'Aix-Marseille
Chercheur en programmation par contrainte (IA)
Ingénieur en génie logiciel

elmouelhi.achref@gmail.com

Plan

SQL

Vue

- Comme une table virtuelle : physiquement inexistante
- Définie par un nom et une requête SQL
- Pouvant être interrogée comme une table
- Pouvant, sous certaines conditions, être mise à jour

SQL

Vue

- Comme une table virtuelle : physiquement inexistante
- Définie par un nom et une requête SQL
- Pouvant être interrogée comme une table
- Pouvant, sous certaines conditions, être mise à jour

Remarque

Une mise à jour des tables à partir desquelles une vue est définie se propage aux vues.

SQL

Avantages

- Rassembler les données logiques même si elles sont définies sur plusieurs tables
- Simplifier les requêtes en masquant la complexité du schéma
- Sécuriser l'accès à certaines colonnes si on ne les sélectionne pas avec la requête de la vue.

SQL

Avantages

- Rassembler les données logiques même si elles sont définies sur plusieurs tables
- Simplifier les requêtes en masquant la complexité du schéma
- Sécuriser l'accès à certaines colonnes si on ne les sélectionne pas avec la requête de la vue.

Inconvénients

- Restrictions sur les mises à jour
- Coût : chaque appel d'une vue vaut l'exécution de la requête `select` précisée à la création

SQL

Syntaxe de création

```
CREATE [OR REPLACE] VIEW nom_vue [ ( colonnes ) ] AS
requêteSelect
[WITH [CASCADED | LOCAL] CHECK OPTION]
```

SQL

Syntaxe de création

```
CREATE [OR REPLACE] VIEW nom_vue [ ( colonnes ) ] AS
requêteSelect
[WITH [CASCADED | LOCAL] CHECK OPTION]
```

Exemple

```
CREATE OR REPLACE VIEW marseillais  AS
SELECT num, nom, prenom, salaire
FROM personne
WHERE ville = 'Marseille';
```

SQL

Syntaxe de création

```
CREATE [OR REPLACE] VIEW nom_vue [ ( colonnes ) ] AS
requêteSelect
[WITH [CASCADED | LOCAL] CHECK OPTION]
```

Exemple

```
CREATE OR REPLACE VIEW marseillais  AS
SELECT num, nom, prenom, salaire
FROM personne
WHERE ville = 'Marseille';
```

Une vue peut être interrogée comme une table

```
SELECT *
FROM marseillais;
```

SQL

Syntaxe de suppression

```
DROP VIEW nom_vue;
```

SQL

On peut créer une vue multi-tables

```
CREATE OR REPLACE VIEW vehicule_marseillais AS
SELECT marque, modele, ville, nom, prenom
FROM personne, vehicule
WHERE ville = 'Marseille'
AND nump = num;
```

SQL

On peut créer une vue multi-tables

```
CREATE OR REPLACE VIEW vehicule_marseillais AS
SELECT marque, modele, ville, nom, prenom
FROM personne, vehicule
WHERE ville = 'Marseille'
AND nump = num;
```

Cette vue peut aussi être interrogée comme une table

```
SELECT *
FROM vehicule_marseillais;
```

SQL

On peut créer une vue à partir d'une autre vue

```
CREATE OR REPLACE VIEW cadre_marseillais AS
SELECT *
FROM marseillais
WHERE salaire >= 2000;
```

SQL

On peut créer une vue à partir d'une autre vue

```
CREATE OR REPLACE VIEW cadre_marseillais AS
SELECT *
FROM marseillais
WHERE salaire >= 2000;
```

Cette vue peut aussi être interrogée comme une table

```
SELECT *
FROM cadre_marseillais;
```

SQL

Exercice 1

En utilisant les vues, sélectionnez les personnes qui ont le plus grand nombre de voitures en utilisant les deux fonctions d'agrégation `max` et `count`.

SQL

Conditions pour la mise à jour

- Si la vue est définie à partir de plusieurs tables, il faut que les modifications concernent une seule table.
- Il ne faut pas que la requête de la vue contienne les mots-clés suivants :
  - `DISTINCT`
  - `LIMIT`
  - `GROUP BY`
  - `HAVING`
  - `UNION`
  - `une fonction d'agrégation`

SQL

Modifier le modèle (de la table véhicule) est possible

```
UPDATE vehicule_marseillais
SET modele = 'focus'
WHERE marque = 'ford';
```

SQL

Modifier le modèle (de la table véhicule) est possible

```
UPDATE vehicule_marseillais
SET modele = 'focus'
WHERE marque = 'ford';
```

Modifier le nom (de la table personne) est possible

```
UPDATE vehicule_marseillais
SET nom = 'mercure'
WHERE prenom = 'sophie';
```

SQL

Modifier le modèle (de la table véhicule) est possible

```
UPDATE vehicule_marseillais
SET modele = 'focus'
WHERE marque = 'ford';
```

Modifier le nom (de la table personne) est possible

```
UPDATE vehicule_marseillais
SET nom = 'mercure'
WHERE prenom = 'sophie';
```

Modifier le nom (de la table personne) et le modèle (de la table véhicule) est impossible

```
UPDATE vehicule_marseillais
SET nom = 'mercure',
modele = 'focus'
WHERE prenom = 'sophie';
```

SQL

Conditions pour l'insertion = conditions pour la mise à jour +

- Les colonnes `not null` d'une table, n'ayant pas une valeur par défaut, doivent être présentes dans le `select` de création de la vue
- Il ne faut pas avoir de colonnes dupliquées dans la vue (`select *` dans une jointure)

SQL

Insérer le tuple suivant (dans la table véhicule) est possible

```
INSERT INTO vehicule_marseillais
SET modele = 'ibiza',
marque = 'seat';
```

SQL

Insérer le tuple suivant (dans la table véhicule) est possible

```
INSERT INTO vehicule_marseillais
SET modele = 'ibiza',
marque = 'seat';
```

Insérer un tuple dans une vue qui implique deux insertions dans deux tables différentes (un premier dans la table véhicule et un deuxième dans la table personne) est impossible

```
INSERT INTO vehicule_marseillais
SET nom = 'green',
prenom = 'michel',
modele = 'captur',
marque = 'renault';
```

SQL

Conditions pour la suppression = conditions pour la mise à jour +

- La vue est mono-table

SQL

Syntaxe de création

```
CREATE [OR REPLACE] VIEW nom_vue [ ( colonnes ) ] AS
requêteSelect
[WITH [CASCADED | LOCAL] CHECK OPTION]
```

SQL

Syntaxe de création

```
CREATE [OR REPLACE] VIEW nom_vue [ ( colonnes ) ] AS
requêteSelect
[WITH [CASCADED | LOCAL] CHECK OPTION]
```

Explication

- `check option` permet de modifier une vue à condition que le tuple modifié appartienne encore à la vue après la mise à jour
- `check option` interdit également l'insertion dans une vue d'un tuple contenant une valeur qui ne satisfait pas la condition de sélection de la vue.
- `WITH LOCAL CHECK OPTION` : on vérifie seulement les conditions de la vue.
- `WITH CASCADED CHECK OPTION` (par défaut) : on vérifie les conditions de la vue et celles des vues à partir desquelles on l'a créée.

SQL

Considérons la vue suivante

```
CREATE OR REPLACE VIEW cadre_marseillais AS
SELECT *
FROM marseillais
WHERE salaire >= 2000
WITH  CHECK OPTION;
```

SQL

Considérons la vue suivante

```
CREATE OR REPLACE VIEW cadre_marseillais AS
SELECT *
FROM marseillais
WHERE salaire >= 2000
WITH  CHECK OPTION;
```

Il est possible d'ajouter un nouveau tuple à cette vue

```
INSERT INTO cadre_marseillais VALUES(8, 'freeman', 'henri',
   2500);
```

SQL

Considérons la vue suivante

```
CREATE OR REPLACE VIEW cadre_marseillais AS
SELECT *
FROM marseillais
WHERE salaire >= 2000
WITH  CHECK OPTION;
```

Il est possible d'ajouter un nouveau tuple à cette vue

```
INSERT INTO cadre_marseillais VALUES(8, 'freeman', 'henri',
   2500);
```

Ajouter un tuple qui ne remplit pas les conditions de création d'une vue est impossible

```
INSERT INTO cadre_marseillais VALUES(9,'freuder','jennifer'
   ,1500);
```