Bases de données
SQL

LDD
Relations virtuelles et « vues »

BD109
v121a

2020-10-31

Département d'informatique
Faculté des sciences

UNIVERSITÉ DE SHERBROOKE

Christina.Khnaisser@USherbrooke.ca
http://info.USherbrooke.ca/ckhnaisser
Luc.Lavoie@USherbrooke.ca
http://info.USherbrooke.ca/llavoie

Plan

- Relations virtuelles
  - Définition
  - Motivation
- Vues
  - Syntaxe
  - Référabilité
  - Modifiabilité
  - Exemples

Relations virtuelles (vues)
Définition

- Une vue (VIEW en SQL) est
  - une variable de relation (virtuelle)
  - définie par une ***expression*** *(relationnelle)*
  - plutôt que par ***énumération*** *(des tuples)* comme pour une variable de relation de base (TABLE en SQL).

- Plus exactement, en regard de la théorie relationnelle
  - Une TABLE se veut une représentation d'une variable de relation (*relvar*) de base.
  - Une VIEW se veut une représentation d'une variable de relation (*relvar*) virtuelle.

Relations virtuelles (vues)
Exemple inspiré par Gaspard et Madeleine

- Plusieurs clients désirent recevoir l'information relative au poids des armes en kilogrammes, d'autres en livres. Or, celle-ci est représentée en grammes dans la relation

  `Arme (noProduit, typeArme, poids)`

- Madeleine propose de définir une vue

  `ArmeN (noProduit, typeArme, kg, lb)`

- Soit, en SQL :

```
CREATE VIEW ArmeN(noProduit, typeArme, kg, lb) AS
    SELECT
        noProduit,
        typeArme,
        poids*0.001 AS kg,
        poids*0.002204622622 AS lb
    FROM Arme
```

Relations virtuelles (vues)
Remarque 1

- L'attribut poids,
  - ayant servi à la définition de kg et lb,
  - n'y figure pas.
- Il le pourrait,
  - c'est le choix de Madeleine;
  - ... que nous soutenons ☺ ;
  - ... on remarque également que lb désigne la masse exprimée en livre-masse (égale à 1/14 de stone) et non en livre-poids (qui varie en fonction de la gravité locale).

Relations virtuelles (vues)
Remarque 2

- On peut représenter cette relation comme étant
  - « calculé » à chaque référence ;
  - aucun de ses attributs n'est stocké
    (du moins n'y a-t-il aucune exigence en ce sens,
    seuls ceux de la table Arme doivent l'être).

- Note
  - En pratique le SGBDR doit garantir ces comportements, indépendamment de la représentation sous-jacente qu'il est libre d'adopter.

VUES
SYNTAXE SIMPLIFIÉE

creation ::=
CREATE *objet*

objet ::=
objTable |
objVue |
objDomaine |
objType |
objAssertion |
<autres objets>

objVue ::=
VIEW nomVue **AS (***requête***)**

Syntaxe complète : https://www.postgresql.org/docs/11/sql-createview.html

VUES
UTILISATION – CONSULTATION (SELECT)

- Une vue devrait être utilisable de la même manière qu'une table.
- Cela est généralement le cas dans les requêtes (SELECT)
  → voir diapositive suivante.
- Note
  - Une vue est définie grâce à une expression relationnelle, ce qui comprend entre autres l'utilisation des opérateurs
    - JOIN
    - UNION, INTERSECT et EXCEPT
    - GROUP BY et HAVING
  - mais ne comprend ***pas***
    - ORDER BY
    - ni ce qui s'ensuit.

VUES
UTILISATION – CONSULTATION (SELECT)

- Par exemple, une relation X décomposée en deux relations Y et Z afin d'obtenir un schéma normalisé peut continuer à être utilisée grâce à

```
CREATE VIEW X AS
        SELECT * FROM Y NATURAL JOIN Z
```

- On peut également définir l'union de deux tables compatibles B et C comme suit

```
CREATE VIEW A AS
        SELECT * FROM B UNION SELECT * FROM C
```

VUES
UTILISATION – CONSULTATION (SELECT)

- Mais, dans ce cas
  - Quel sens faut-il donner à une insertion dans X ?
  - Quel sens faut-il donner à une insertion dans A ?

- Au tableau !

VUES
UTILISATION - MODIFICATION (UPDATE ET DELETE)

1. L'expression est un SELECT
2. L'expression n'inclut pas DISTINCT
3. Les dénotations de colonnes sont des références simples (pas d'expression)
4. Le FROM ne référence qu'une seule table ou une seule vue modifiable
5. Le WHERE ne contient pas de sous-requête relative à la table ou à la vue référencée par le FROM
6. L'expression ne contient pas de GROUP BY
7. L'expression ne contient pas de HAVING

VUES
UTILISATION - MODIFICATION (INSERT)

- Dans le cas d'un INSERT, aux conditions précédentes, il faut ajouter une dernière condition :
  - Les colonnes absentes de la table ultimement référencée doivent comporter une valeur par défaut.

VUES
UTILISATION - SYNTHÈSE

- Pour avoir un résultat correct et prévisible,
  - il **ne faut pas** spécifier
    WITH LOCAL CHECK OPTION

- SQL limite abusivement la modifiabilité.

- On peut aisément trouver des contre-exemples modifiables relativement à chacune des conditions.

- Plusieurs dialectes nuancent ces conditions… de façon non uniforme et non transportable.

Exemple de données

Étudiant

matricule	nom	adresse
15113150	Paul	ᐳᕽᖏᓂᖅ
15112354	Éliane	Blanc-Sablon
15113870	Mohamed	Tadoussac
15110132	Sergeï	Chandler

Activité

sigle	titre
IFT159	Analyse et programmation
IFT187	Éléments de bases de données
IMN117	Acquisition des médias numériques
IGE401	Gestion de projets
GMQ103	Géopositionnement

TypeÉvaluation

code	description
IN	Examen intra
FI	Examen final
TP	Travail pratique
PR	Projet

Résultat

matricule	TE	activité	trimestre	note
15113150	TP	IFT187	20133	80
15112354	FI	IFT187	20123	78
15113150	TP	IFT159	20133	75
15112354	FI	GMQ103	20123	85
15110132	IN	IMN117	20123	90
15110132	IN	IFT187	20133	45
15112354	FI	IFT159	20123	52

VUES (RELATIONS VIRTUELLES)
EXEMPLE

- Plusieurs étudiants désirent recevoir l'information relative à la note totale par cours.
- Définir une vue
  Bulletin (matricule, activité, trimestre, noteT)

- Soit, en SQL :

```
CREATE VIEW Bulletin
   (matricule, activite, trimestre, noteT) AS
SELECT
   matricule,
   activite,
   trimestre,
   sum(note) AS noteT,
FROM Resultat
GROUP BY matricule, activite, trimestre
ORDER BY noteT
```

VUES (RELATIONS VIRTUELLES)
EXEMPLE

- Vue

  Bulletin (matricule, activité, trimestre, noteT)

- Quelles sont les notes totales de l'étudiant identifié par la matricule '15113150' ?

```
SELECT noteT, activite, trimestre
FROM Bulletin
WHERE matricule = '15113150'
```

RÉFÉRENCES

- Elmasri et Navathe (4e éd.), chapitre 7
- Elmasri et Navathe (6e éd.), chapitre 4
- [Loney2008]
  Loney, Kevin ;
  Oracle Database 11g: The Complete Reference.
  Oracle Press/McGraw-Hill/Osborne, 2008.
  ISBN 978-0071598750.
- [Date2012]
  Date, Chris J. ;
  SQL and Relational Theory: How to Write Accurate SQL Code.
  2nd edition, O'Reilly, 2012.
  ISBN 978-1-449-31640-2.
- Le site d'Oracle (en anglais)
  - http://www.oracle.com/pls/db10g/portal.portal_demo3?selected=5
  - http://docs.oracle.com/cd/B19306_01/server.102/b14200/toc.htm
- Le site de PostgreSQL (en français)
  - http://docs.postgresqlfr.org