Bases de données
SQL

Relations virtuelles et « vues »

SQL_05a
v130a
2025-02-19

Christina.Khnaisser@USherbrooke.ca
Luc.Lavoie@USherbrooke.ca

Plan

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

Relations virtuelles

- Définitions
- Exemple
- Problématique
- Cas simple
- Risque d'incohérence
- Cas général

Du point de vue de la théorie relationnelle

Variables de relation

- La thérorie relationnelle définit deux formes de variables de relation (*relvar*).
- La relvar de base est définie par une ***énumération*** *de tuples*.
- La relvar virtuelle est définie par une ***expression*** *relationnelle*
  - *référant généralement à d'autres relvar.*
- En SQL
  - Une TABLE se veut une représentation d'une relvar de base.
  - Une VIEW se veut une représentation relvar virtuelle.

Relations virtuelles (vues)
Exemple inspiré par Gaspard et Madeleine

- L'information relative au *poids* des produits est exprimée en grammes dans la BD par la *table* suivante :

```
Produit (noProduit, typeProduit, poids)
```

- Les clients désirent plutôt recevoir l'information soit en kilogrammes soit en livres.
- Madeleine propose de définir la *vue* suivante :

```
CREATE VIEW ProduitN(noProduit, typeProduit, kg, lb) AS
    SELECT
        noProduit,
        typeProduit,
        poids/1000.0 AS kg,
        poids/453.5923699688862215 AS lb
    FROM Produit
```


*Gaspard a insisté pour dénommer l'attribut représentant la **masse** du produit par l'identifiant « poids »*

au prétexte qu'une erreur partagée par la majorité devient la vérité.

Qu'en pensez-vous ?

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 une vue comme étant une fonction « calculée » à chaque référence.

- Notes
  - Un SGBDR doit garantir cette sémantique, indépendamment de la représentation sous-jacente qu'il est libre d'adopter.
  - En pratique, il est donc possible de ne stocker aucun tuple (donc aucun attribut) propre à cette relvar.
  - Sans notre exemple, seuls les tuples (et donc les attributs) de la table Produit doivent l'être).

Relations virtuelles (vues)
Remarque 3

- Une relvar virtuelle (VIEW) peut-elle être utilisée comme une relvar de base (TABLE) ?
- En particulier, est-il possible d'y insérer (INSERT) ou d'en retirer (DELETE) des tuples ? De la modifier UPDATE ?
- Oui !

Relation virtuelle – insertion (INSERT)

- Lorsqu'on insère un tuple dans la relation virtuelle, il faut en fait insérer le tuple correspondant dans la relation de base.
- Le quadruplet de ProduitN est facile à «convertir» en triplet de Produit.

- Par exemple, la procédure

```
CREATE PROCEDURE insPN (no NP, type TP, kg KG, lb LB)
```

- pourrait être mise en oeuvre comme ceci :

```
INSERT INTO Produit (noProduit, typeProduit, poids)
  VALUES (no, type, kg*1000 as poids)
```

ou comme ceci :

```
INSERT INTO Produit (noProduit, typeProduit, poids)
  VALUES (no, type, lb*453.5923699688862215 as poids)
```


Pour simplifier l'utilisation des vues, on a recours, le plus souvent, à des automatismes (TRIGGER), que sous étudierons sous peu.

Relation virtuelle – possibilité d'incohérence

- Le problème est ailleurs!
- Que doit-il se passer on tente d'insérer dans ProduitN un produit dont la masse est à la fois d'un kg et de deux lb?

```
VALUES ('A01234', 'Flûte', 1, 2)
```

- Quelle devrait être la masse du produit dans Produit?

```
VALUES ('A01234', 'Flûte', 1000)
<->
VALUES ('A01234', 'Flûte', 907.18473993777244429)
```

- Le problème ne résulte pas d'une « erreur » de la théorie relationnelle pas plus qu'elle n'invalide notre hypothèse!

Relation virtuelle – possibilité d'incohérence (suite)

- C'est notre modélisation qui est erronée, comme nous le montrera la théorie de la normalisation relationnelle.
- Pour éviter cette situation nous aurions du définir deux relations virtuelles (et les procédures d'insertion correspondantes) :

```
CREATE VIEW Produit_kg
  (noProduit, typeProduit, kg)
AS
  SELECT
    noProduit,
    typeProduit,
    poids/1000.0 AS kg
  FROM Produit
```

```
CREATE VIEW Produit_lb
  (noProduit, typeProduit, lb)
AS
  SELECT
    noProduit,
    typeProduit,
    poids/453.5923699... AS lb
  FROM Produit
```


Mais cela ne résout pas forcément le problème d'incohérence lorsqu'on prend en compte la séquence des modifications potentielles

et les erreurs d'arrondissement des représentations inexactes (telles que FLOAT et DOUBLE PRECISION).

D'autres approches sont possibles :

* permettre la définition d'attributs virtuels
* utiliser des fonctions scalaires
* utiliser des fonctions non scalaires

Dans tous les cas, ces solutions, si elles sont bien appliquées, seront équivalentes à celle présente.

En fait, il semble que la nature des solutions soit plus d'ordre syntaxique que sémantique.

En pratique, le choix de la solution dépend souvent plus du dialecte SQL utilisé que de principes fondamentaux ou de règles de pratique.

Relation virtuelle – retrait (DELETE)

- Puisqu'il y a une correspondance biunivoque (un-à-un) entre les tuples de la relation de base et ceux de la relation virtuelle et que la même clé est utilisée pour les deux relations, logiquement la réponse doit être positive.

Relation virtuelle – retour à la théorie

- On remarque que l'opération réalisée pour obtenir la relation virtuelle à partir de de la relation de base est une extension.
- Qu'en est-il des autres opérations ?
- Il y en beaucoup!
- Grâce à la théorie, nous savons toutefois qu'elles peuvent toutes être exprimées à l'aide de six opérations (renommage, projection, restriction, jointure, union et différence).
- Il suffit donc d'examiner ces six opérations pour obtenir la réponse à cette question.

- ... ce qui sera étudié dans le cadre d'une autre activité pédagogique (par exemple IGE487 ou INFO323).

VIEW

- Syntaxe
- Évaluation
- Modification
- Exemple

Du point de vue SQL

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/16/sql-createview.html

<view definition> ::=
CREATE [RECURSIVE] VIEW <table name>
<view specification> AS <query expression>
[WITH [<levels clause>] CHECK OPTION]

<view specification> ::=
<regular view specification>
| *<referenceable view specification>*

<regular view specification> ::=
[(<view column list>)]

<view column list> ::=
<column name list>

<query expression> ::=
requête

<levels clause> ::=
CASCADED | LOCAL

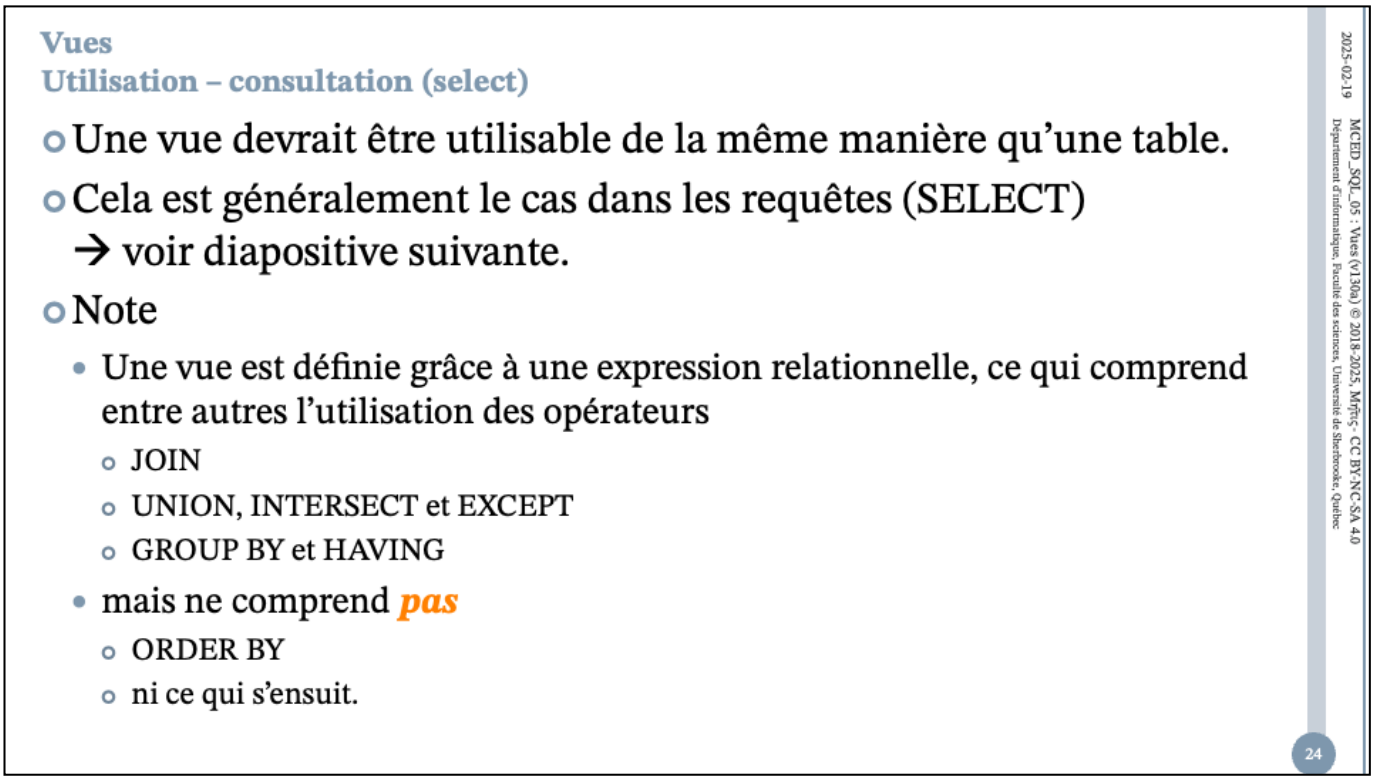
<referenceable view specification> ::=
OF <path-resolved user-defined type name>
[<subview clause>] [<view element list>]

<subview clause> ::=
UNDER <table name>

<view element list> ::=
(<view element> [{ <comma> <view element> }...])

<view element> ::=
<self-referencing column specification> | <view column option>

<view column option> ::=
<column name> WITH OPTIONS <scope clause>



Pourquoi cette exclusion ?

Selon la théorie relationnelle, c'est clair et évident : une relation est un ensemble de tuples (donc non ordonnés).

Mais en SQL, la table n'est-elle pas une liste (ou un tableau) de tuples ?

Eh bien, non, c'est une collection de tuples admettant certes la répétition, mais ne préservant pas (a priori) l'ordre.

En fait, le type de la table (et donc de la vue) et le type du résultat d'un SELECT ne sont pas les mêmes, bien qu'apparentés.

Le type associé à un SELECT est un tableau, celui associé à une table (comme à une vue) est une collection.

Lors d'une affectation (en particulier dans le contexte d'un énoncé INSERT), il y a donc une « conversion » de type.

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 A, 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 ?
- Il faut appliquer l'opération logique aux prédicats associés aux relations d'orgine, puis en simplifier l'expression.

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)
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.
- Nous verrons comment pallier ceci avec les TRIGGER !

Exercice au tableau

Exemple de données

Étudiant

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

TypeÉvaluation

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

Activité

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

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

On remarque l'omission de la dénotation des types des attributs.

C'est fréquemment le cas dans les représentations graphiques, afin de les alléger.

Cela ne porte pas à conséquence dans la mesure où une définition textuelle les accompagne, ce qui est le cas ici.

Nous verrons bientôt d'autres représentations graphiques plus complètes.

Note : >ᐃᖏᓂᑐᖅ se prononce (approximativement) Puvirnituq

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