UNIVERSITÉ de la NOUVELLE-CALÉDONIE

Chapitre 2: Définir et structurer les bases de données

Définir des vues des données

(dérivé du cours du Pr. Jeffrey Ullman, Stanford University)

Université de la Nouvelle-Calédonie
frederic.flouvat@univ-nc.nc

Les Vues

- Une *vue* est une relation définie en fonction de tables stockées (appelée *tables de base*) et d'autres vues.

- Avantages:
  - Améliore la sécurité (limite l'accès à certaines données et masque la conception de la base de données)
  - Facilite la conception côté client (simplifie l'accès aux données)
  - Peut beaucoup améliorer le temps d'interrogation (select)
- Inconvénients :
  - Difficilement modifiable (insert/delete/update) directement
  - Peut consommer des ressources supplémentaires

- Il existe deux types de vues:
  1. *Virtuelles* = pas stockées dans la base de données; juste une requête enregistrée construisant à chaque appel la relation.
  2. *Matérialisées* = relation réellement construite et enregistrée.

Déclarer une Vue

- Déclaré par l'instruction SQL:

  CREATE [MATERIALIZED] VIEW <name> AS <query>;

  - Par défaut, la vue est virtuelle.

- Exemple:
  - CanDrink(drinker, beer) est une vue « contenant » les pairs buveur-bière tel que le buveur fréquente au moins un bar proposant la bière:

```
CREATE VIEW CanDrink AS
        SELECT drinker, beer
        FROM Frequents, Sells
        WHERE Frequents.bar = Sells.bar;
```

Accéder à une Vue

- Interroger une vue comme si c'était une table de base.
- Dans certains cas, possibilité de modifier le contenu d'une vue si celle-ci ne dépend que d'une seule table de base.
  - modifier = INSERT, DELETE, UPDATE
  - par contre ALTER toujours possible
- Exemple:

```
SELECT beer FROM CanDrink
WHERE drinker = 'Sally';
```

Modification de Vues grâce à des Triggers

- En général, il est impossible de modifier directement une vue virtuelle, tout simplement parce qu'elle n'existe pas concrètement.
  - Dans PostgreSQL, modification automatique des données sous-jacentes (table complète ou certaines colonnes) possible ssi
    - une seule table dans le FROM,
    - pas de DISTINCT, GROUP BY, LIMIT, UNION, INTERSECT, EXCEPT, etc
    - aucun opérateur d'agrégation et aucune fonction retournant des ensembles dans le SELECT

- Mais l'option d'événement INSTEAD OF des triggers permet d'interpréter des modifications sur des vues d'une manière qui fait sens.

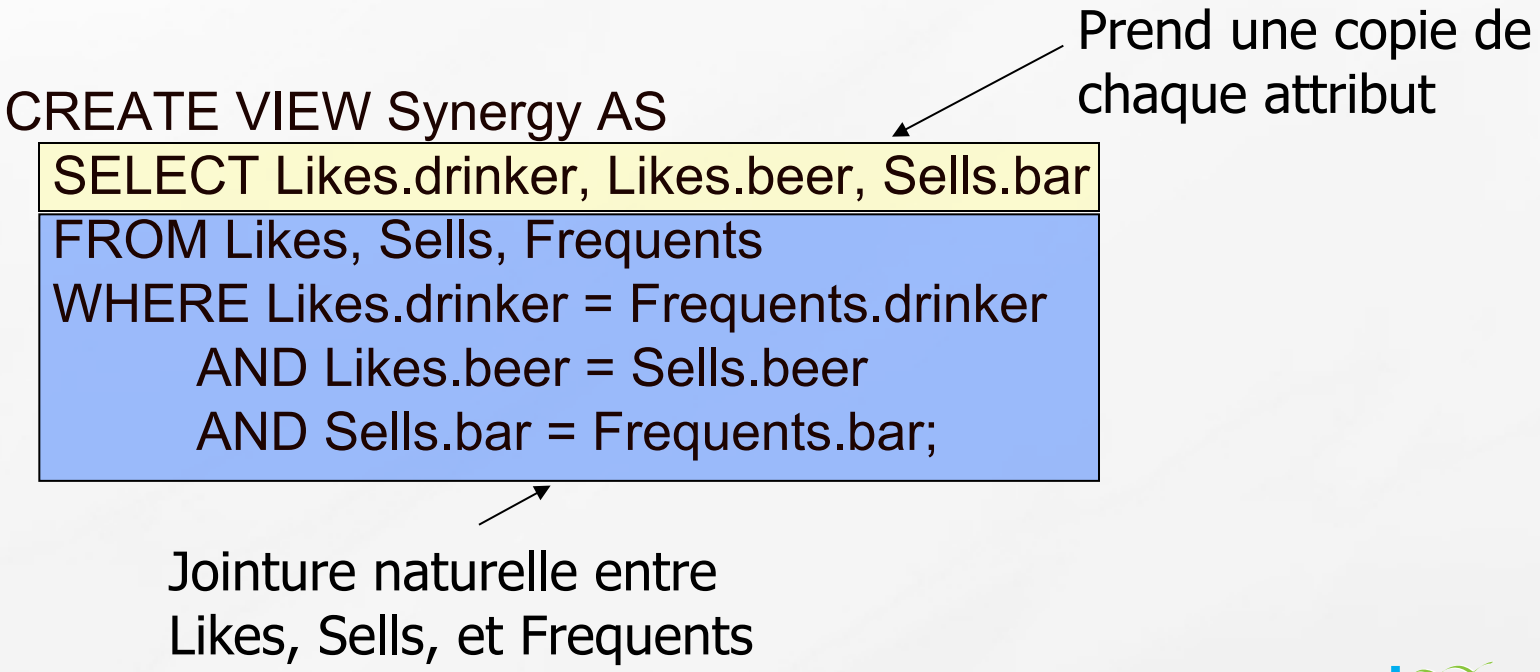
CREATE TRIGGER <nom>

INSTEAD OF [INSERT, DELETE, UPDATE] ON <nomVue>

[FOR EACH ROW]
WHEN <condition>
<fonction trigger PL/pgSQL>

Modification de Vues grâce à des Triggers

- Exemple:
  - La vue Synergy est composée du triplet (drinker, beer, bar) et représente les buveurs, les bières qu'ils aiment, ainsi que les bars qui les servent.



Modification de Vues grâce à des Triggers

- Exemple (suite):
  - Nous ne pouvons pas insérer dans Synergy --- c'est une vue virtuelle.
  - Mais nous pouvons utiliser à la place un trigger avec INSTEAD OF pour transformer l'insertion d'un tuple (drinker, beer, bar) en trois insertions sur Likes, Sells, et Frequents.
    - Sells.price aura la valeur NULL.

```
CREATE TRIGGER SynergyViewTrig
   INSTEAD OF INSERT ON Synergy
   FOR EACH ROW
   EXECUTE PROCEDURE insertSynergyView();

CREATE FUNCTION insertSynergyView() RETURNS TRIGGER AS $$
   BEGIN
      INSERT INTO LIKES VALUES(new.drinker, new.beer);
      INSERT INTO SELLS(bar, beer) VALUES(new.bar, new.beer);
      INSERT INTO FREQUENTS VALUES(new.drinker, new.bar);
      RETURN NULL;
   END;
$$ LANGUAGE plpgsql;
```

Les vues Matérialisées

CREATE MATERIALIZED VIEW <name> [(<colomn_name [,…])] AS <query>;

- Problème: chaque fois qu'une table de base change, la vue matérialisée peut changer.
  - Il faut reconstruire la vue à chaque changement.

- Solution: Reconstruction périodique de la vue matérialisée, qui sans cela serait "périmée".
  - P.ex. en utilisant *cron* (un planificateur de tâches sous Unix)

➢ **L'implémentation dépend du SGBD.**

- Dans PostgreSQL,
  - uniquement disponible depuis la version 9.3
  - utiliser la commande REFRESH MATERIALIZED VIEW <name> pour rafraîchir les données de la vue
  - pas de mises à jours directes

Les vues Matérialisées

- Exemples:
  - Mailing List d'une classe
    - La mailing list d'une classe peut être obtenue par une vue matérialisée sur la table de base contenant tous les étudiants.
    - si la vue est mise à jour 4 fois par jours, les étudiants enregistrés entre deux mises à jour ne recevront pas les mails.
  - Un entrepôt de données
    - Les magasins Wal-Mart stockent les ventes de tous les magasins dans une base de données.
    - Pendant la nuit, les ventes du jour sont utilisées pour mettre à jour un *entrepôt de données* (data warehouse) = des vues matérialisées sur les ventes.
    - L'entrepôt est utilisé par les preneurs de décision pour prédire les tendances et envoyer des produits là où ils se vendent le mieux.