Les bases de données

Séance 9

Les vues, index et les droits

Sommaire

- L’objet VUE
- L’objet INDEX
- Les privilèges d’accès

Définition

- Une ***vue*** est une relation définie à partir des tables (*tables de base*) et des autres vues.
- 2 types :
  - ***Virtuelle*** : les données de la vue ne sont pas stockées physiquement, seule la requête décrivant la vue est stockée.
  - ***Matérialisée*** : données construites et stockée

Les vues

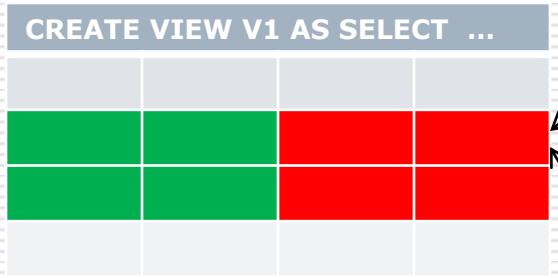
- Une vue est une **table virtuelle** de la base de données dont le contenu est défini par une requête SELECT.

- CREATE [MATERIALIZED] VIEW <name> AS <query>;
- Par défaut : *virtuelle*

Table 1

Id	Nom	Prénom	..
1	Dupond	Marc	..
2	Dupont	Jean	
3	Martin	Tom	
4	Javelot	Pierre	

Vue 1



User ⟷

Table 2

Id	IdClient	DateC	..
1	2	25/09/2008	..
2	2	30/07/2009	
3	1	29/04/2000	
4	3	15/07/2006	

Exemple

- CanDrink(drinker, beer)

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

Exemple

```
CREATE VIEW dep20_view AS
   SELECT last_name, salary*12 AS annual_salary
   FROM employees
   WHERE department_id = 20;
```

=> Vue des employés du département numéro 20 avec leur salaire annuel.

Les vues : avantages

- Sécurité
- Simplicité des requêtes
- Simplicité structurelle
- Isolation des modifications
- Intégrité des données

Les vues : inconvénients

- Performances
  - La traduction de la requête peut être longue.
- Restriction des mises à jour
  - possibilité de mise à jour qu'à partir de vues simples.

Créer et gérer des vues

```
CREATE [OR REPLACE]
      VIEW nom_vue [col,...] AS
        <syntaxe_d_un_select>
      [WITH CHECK OPTION
        [CONSTRAINT nom]];

DROP VIEW nom_vue ;

RENAME ancien_nom TO nouveau_nom ;
```

Requêtes sur une vue

- On peut faire des requêtes sur une vue comme sur une table de base :

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

- Les mises à jour d'une vue sont limitées !

Les vues : mise à jour de données

- Conditions :
  - Construite sur une seule table.
  - Absence de GROUP BY.
  - Les colonnes résultats doivent être des colonnes réelles de la table et non des résultats calculés.
  - La vue contient **toutes** les colonnes NOT NULL de la table.
- La mise à jour de vue permet d'insérer et de modifier des lignes dans une table.

Vue "mettable à jour" : exemple

```
CREATE VIEW clerk AS
   SELECT employee_id, last_name, department_id, job_id
   FROM employees
   WHERE job_id = 'PU_CLERK'
   OR job_id = 'SH_CLERK'
   OR job_id = 'ST_CLERK';
```

- On peut faire une mise à jour sur la table de base Employees via la vue Clerk :

```
UPDATE clerk SET job_id = 'PU_MAN' WHERE employee_id = 118;
```

Les vues : contrôle d'intégrité

- La clause **WITH CHECK OPTION** permet d'utiliser une vue en prévision d'un contrôle des données.
- Cette clause interdit d'insérer à travers la vue des lignes qui ne répondraient plus à la **clause de définition** de la vue

Les vues : contrôle d'intégrité

```
CREATE VIEW clerk AS
    SELECT employee_id, last_name, department_id, job_id
    FROM employees
    WHERE job_id = 'PU_CLERK'
    OR job_id = 'SH_CLERK'
    OR job_id = 'ST_CLERK'
    WITH CHECK OPTION;
```

- On ne peut plus faire cette mise à jour !

```
UPDATE clerk SET job_id = 'PU_MAN' WHERE employee_id = 118;
```

Les vues : contrôle d'intégrité

```
CREATE VIEW  vue_client AS
  SELECT * FROM client
  WHERE region = 'Paris'
  WITH CHECK OPTION
  CONSTRAINT client_paris;

INSERT INTO vue_client VALUES
  ('c1','','Rouen');
                *
```

ERROR at line 1:

ORA-01402: view WITH CHECK OPTION where-clause violation

Les indexes

- *Index* = structure de données permettant d'accélérer l'accès aux lignes d'une table!

- Techniques :
  - Table de hachage
  - Arbre de recherche équilibré (*B-tree*)

Indexes implicites et explicites

- Les indexes implicites
  - Créés automatiquement par le SGBD
  - clé primaire, colonne unique
- Les indexes explicites
  - Créés par l'administrateur de données
- Syntaxe générale

```
CREATE INDEX nom_index ON nom_table
    (nom_col [ASC|DESC], nom_col...,);

DROP INDEX nom_ind;
```

Comment choisir les indexes à ajouter

- On met en place des indexes sur :
  - les colonnes utilisées comme critère de jointure,
  - les colonnes servant de critères de sélection,
- sur une table de gros volume.

Problème de "tuning"

- Décider quels indexes à créer ?
  - Pour : un indexe accélère la requête qui l'utilise
  - Contre : un indexe ralentisse les mises à jour sur la table concernée car l'indexe doit être mis à jour aussi !

Exemple: Tuning

- Supposons les traitements suivants avec la base de données Bières :
  1. Insérer des nouveaux produits (10%).
  2. Trouver le prix d'une bière dans un bar (90%).

  - Indexe SellInd sur Sells(bar, beer)
  - Mais pas indexe BeerInd sur Beers !

Les indexes superflus

- Indexes déjà créés implicitement.
- Recherche de condition IS NOT NULL ou IS NULL.
- Pour les évaluations de requêtes par l'intermédiaire d'une fonction.

Exemples de cas où l'index est utilisé

- `SELECT * FROM client WHERE id = 500;`
- `SELECT * FROM client WHERE id > 10;`
-
```
SELECT * FROM client
   WHERE id BETWEEN 300 AND 500;
```
-
```
SELECT * FROM client
   WHERE nom ='Martin';
```
-
```
SELECT * FROM client
   WHERE nom LIKE 'M%' ;
```

Exemples de cas où l'index n'est pas utilisé

- `SELECT * FROM client;`
- `SELECT * FROM client`
  `WHERE nom IS NULL;`
- `SELECT * FROM client`
  `WHERE ca*10 >10000 ;`

Vérification d'un processus d'extraction

- Créer la table plan_table à l'aide du script utlxplan.sql
- Lancer la requête par l'intermédiaire d'un explain plan :
  - EXPLAIN PLAN FOR SELECT … ;
- Visualiser le contenu de la table plan_table

Vérification d'un processus d'extraction

- **EXPLAIN PLAN FOR SELECT nom FROM client WHERE nom = 'Toto';**
- **SELECT operation, object_name FROM plan_table;**

```
Plan Table
--------------------------------------
| Operation                    |  Name     |
--------------------------------------
| SELECT STATEMENT             |           |
|  TABLE ACCESS BY INDEX ROWI| CLIENT     |
|   INDEX RANGE SCAN           |NOM        |
--------------------------------------
```

Droits et privilèges

- Les bases de données permettent à plusieurs utilisateurs de travailler en toute sécurité sur la même base, selon leur fonction.
- Les ordres GRANT et REVOKE permettent de définir les droits de chaque utilisateur sur les objets de la base.

GRANT <privilège> ON <table> TO <utilisateur> [with grant option]

Droits et privilèges

- SELECT (droit de lecture)
- INSERT (droit d'insertion de lignes)
- UPDATE (droit de modification de lignes)
- UPDATE (col1, col2, …) (limité à certaines colonnes)
- DELETE (droit de suppression de lignes)
- ALTER (droit de modification de la définition d'une table)
- ALL (tous les droits ci-dessus)

Droits et privilèges

- Un utilisateur1 ayant accordé un privilège à l'utilisateur2 peut le reprendre à l'aide du contre-ordre :

REVOKE <privilège> ON <table> FROM <utilisateur2>