R2.06 - Exploitation d'une base de données
Cours 5 - Vues

A. Ridard

A propos de ce document

- Pour naviguer dans le document, vous pouvez utiliser :
  - le menu (en haut à gauche)
  - l'icône en dessous du logo IUT
  - les différents liens
- Pour signaler une erreur, vous pouvez envoyer un message à l'adresse suivante : anthony.ridard@univ-ubs.fr

Plan du cours

Vous savez déjà qu'un SGBD permet de définir (LDD) et de manipuler (LMD) des données, il assure également le contrôle des données (LCD).

Plus précisément, il garantit la ***confidentialité*** et l'***intégrité*** des données.
Les vues participent à ce contrôle, mais d'autres possibilités sont offertes...

Un SGBD garantit la confidentialité des données en gérant :

- l'accès aux données[1] avec les ***utilisateurs***, les ***rôles*** et les ***privilèges***[2]
- le niveau externe[3] avec les ***vues*** qui agissent comme des "fenêtres"

Une vue est considérée comme une table virtuelle[4] car elle ne nécessite aucune allocation en mémoire pour contenir les données. Une vue n'a pas d'existence propre car seule sa structure est stockée dans le dictionnaire des données[5].

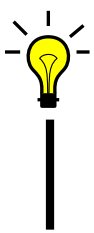
---

1. On étudiera ce point en période 4
2. Il s'agit ici des privilèges en consultation
3. En comparaison avec le niveau :
   - conceptuel (diagramme de classes UML)
   - logique (schéma relationnel)
   - physique (base de données)
4. En opposition aux tables « réelles » implantées physiquement sur le serveur
5. On reviendra sur cette notion plus tard

Une vue est créée à l'aide d'une instruction **SELECT** appelée *requête de définition*.

Syntaxe SQL (simplifiée)

```
CREATE [OR REPLACE] VIEW [schéma.]nomVue
[ ( alias [, alias ] ) ]
AS requêteSELECT
```



- Cette requête interroge une ou plusieurs table(s) ou vue(s)
- Une vue se recharge chaque fois qu'elle est interrogée

Pour créer une vue, l'utilisateur "connecté" doit en posséder le privilège [a].

Si vous avez respecté les consignes du TP1 (R1.05) lors de l'installation du serveur Oracle Database et du client Oracle SQL Developer, vous avez dû créer votre utilisateur avec les instructions suivantes :

```
CREATE USER nom_user IDENTIFIED BY mdp_user ;
GRANT CONNECT, RESOURCE TO nom_user ;
```

A la création de votre première vue [b], vous aurez un message d'erreur :

```
/*

Rapport d'erreur -
ORA-01031: insufficient privileges
01031. 00000 - "insufficient privileges"
*Cause:    An attempt was made to perform a database operation
     without
           the necessary privileges.
*Action:   Ask your database administrator or designated security
           administrator to grant you the necessary privileges

*/
```

a. Toutes ces notions seront étudiées en période 4

b. Avec votre utilisateur bien entendu, et non SYSTEM (mauvaise pratique à éviter)

Vous devez alors vous connecter, depuis Oracle SQL Developer ou Run SQL, en tant que SYSTEM avec le mot de passe[a] utilisé lors de l'installation de Oracle Database !

Puis, vous devez ajouter à votre utilisateur le privilège de création de vue en exécutant l'instruction suivante :

```
GRANT CREATE VIEW TO nom_user ;
```

a. Si vous l'avez oublié, tout n'est pas encore perdu, mais vous aurez besoin d'aide...

Vue sans alias

```
CREATE OR REPLACE VIEW vue_PiloteEtCompagnie
AS
SELECT nomPilote, nomComp
FROM Pilote
     JOIN Compagnie ON compPil = idComp
;

SELECT *
FROM vue_PiloteEtCompagnie
;
```

RESULT

```
View VUE_PILOTEETCOMPAGNIE créé(e).
```

NOMPILOTE	NOMCOMP
Ridard	Air France
Naert	EasyJet
Godin	Ryanair
Fleurquin	Air France
Pham	American Airlines
Kamp	American Airlines

Vue avec alias dans la requête

```
CREATE OR REPLACE VIEW vue_PiloteEtCompagnie
AS
SELECT nomPilote, nomComp nom_Compagnie
FROM Pilote
     JOIN Compagnie ON compPil = idComp
;

SELECT *
FROM vue_PiloteEtCompagnie
;
```

RESULT

```
View VUE_PILOTEETCOMPAGNIE créé(e).
```

NOMPILOTE	NOM_COMPAGNIE
Ridard	Air France
Naert	EasyJet
Godin	Ryanair
Fleurquin	Air France
Pham	American Airlines
Kamp	American Airlines

Vue avec alias dans la création

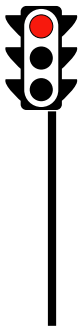
```
CREATE OR REPLACE VIEW vue_PiloteEtCompagnie
(
    nom_Pilote,
    nom_Compagnie
)
AS
SELECT nomPilote, nomComp
FROM Pilote
     JOIN Compagnie ON compPil = idComp
;

SELECT *
FROM vue_PiloteEtCompagnie
;
```

RESULT

```
View VUE_PILOTEETCOMPAGNIE créé(e).

NOM_PILOTE                NOM_COMPAGNIE
------------------------- ------------------------------
Ridard                    Air France
Naert                     EasyJet
Godin                     Ryanair
Fleurquin                 Air France
Pham                      American Airlines
Kamp                      American Airlines
```



- Une vue peut servir à "cacher" certaines informations (colonnes et/ou lignes) sans pour autant priver complètement l'utilisateur
- Une vue permet de dénormaliser virtuellement les tables et donc facilite l'interrogation pour l'utilisateur en limitant les jointures

Un SGBD garantit l'intégrité des données :

- la **sécurité** en se protégeant des attaques (volontaires)
- la **cohérence** en se préservant des erreurs (involontaires)

La sécurité des données est assurée en gérant :

- l'accès aux données[6] avec les ***utilisateurs***, les ***rôles*** et les ***privilèges***[7]

Quant à la cohérence des données, elle est assurée en gérant :

- les **accès concurrents** avec les ***transactions***[8]
- les **contraintes**
- la **redondance** (potentiellement incohérente)

6. On étudiera ce point en période 4
7. Il s'agit ici des privilèges en modification
8. Ce mécanisme assure les propriétés ACID (Atomicité, Cohérence, Isolation, Durabilité)

Les **contraintes** sont gérées différemment selon leur "type" :

- Les clés (primaires et étrangères), l'existence, l'unicité et les vérifications sont prises en compte directement dans le script de création de tables (déjà vu)
- Les autres (surjectivité, vrais cycles non modifiables, ...) sont programmées :
  - au niveau du serveur (SGBD) en PL/SQL avec des déclencheurs [9]
  - au niveau du client (application) en s'appuyant, éventuellement, sur des ***vues*** [10]

La **redondance** est gérée grâce à :

- La normalisation (déjà vue)
- La dérivation (attribut ou association) avec les ***vues***

9. On étudiera ce point au semestre 3
10. Ces vues expriment les "défauts de cohérence" et sont à la disposition du développeur de l'appl.

Complétons notre schéma relationnel avec les éléments suivants.

Contraintes :

- Une compagnie possède au moins un avion
- Un pilote possède au moins une qualification
- Un pilote d'une compagnie possède au moins une qualification pour un avion de sa compagnie

Attributs dérivables :

- nbAvion dans compagnie [a]
- nbQualification dans Pilote
- piloteExperimente (1 ou NULL) dans Pilote [b]

---

a. Savez-vous comment est représentée une telle contrainte sur le diagramme de classes UML ?

b. On dira qu'un pilote est expérimenté s'il a au moins 4 ans d'expérience, en sachant qu'un pilote vole en moyenne 650 h/an

Une compagnie possède au moins un avion

```
CREATE OR REPLACE VIEW vue_Compagnie_Sans_Avion
AS
SELECT idComp
FROM Compagnie
MINUS
SELECT compAv
FROM Avion
;

SELECT *
FROM vue_Compagnie_Sans_Avion
;
```

RESULT

```
View VUE_COMPAGNIE_SANS_AVION créé(e).

aucune ligne sélectionnée
```

Un pilote possède au moins une qualification

```
CREATE OR REPLACE VIEW vue_Pilote_Sans_Qualification
AS
SELECT idPilote
FROM Pilote
MINUS
SELECT unPilote
FROM Qualification
;

SELECT *
FROM vue_Pilote_Sans_Qualification
;
```

RESULT

```
View VUE_PILOTE_SANS_QUALIFICATION créé(e).

  IDPILOTE
----------
         6
```

Un pilote d'une compagnie possède au moins une qualification pour un avion de sa compagnie

```
CREATE OR REPLACE VIEW vue_Pilote_Illegitime
AS
SELECT idPilote -- les pilotes travaillant pous une compagnie
FROM Pilote
WHERE compPil IS NOT NULL
MINUS
SELECT unPilote -- les pilotes respectant la contrainte
FROM Qualification
    JOIN Pilote ON unPilote = idPilote
        JOIN Avion ON compPil = compAv
WHERE unTypeAvion = leTypeAvion
;

SELECT *
FROM vue_Pilote_Illegitime
;
```

RESULT

```
View VUE_PILOTE_ILLEGITIME créé(e).

aucune ligne sélectionnée
```

nbAvion dans compagnie

```
CREATE OR REPLACE VIEW vue_nbAvion
AS
SELECT idComp, COUNT(idAvion) nb_Avion
FROM Compagnie
    LEFT JOIN Avion ON idComp = compAv
GROUP BY idComp
;

SELECT *
FROM vue_nbAvion
;
```

RESULT

```
View VUE_NBAVION créé(e).

    IDCOMP    NB_AVION
__________ ___________
         1           3
         2           1
         3           1
         4           2
         5           2
```

Compagnie complétée

```
CREATE OR REPLACE VIEW vue_Compagnie_Complete
AS
SELECT Compagnie.idComp, nomComp, pays, estLowCost, nb_Avion
FROM Compagnie
     JOIN vue_nbAvion ON Compagnie.idComp = vue_nbAvion.idComp
;

SELECT idComp, nomComp, nb_Avion -- pour l'affichage sous LaTeX
FROM vue_Compagnie_Complete
;
```

RESULT

```
View VUE_COMPAGNIE_COMPLETE créé(e).
```

IDCOMP	NOMCOMP	NB_AVION
5	Ryanair	2
1	Air France	3
3	EasyJet	1
2	Corsair International	1
4	American Airlines	2

nbQualification dans Pilote

```
CREATE OR REPLACE VIEW vue_nbQualification
AS
SELECT idPilote, COUNT(unTypeAvion) nb_Qualification
FROM Pilote
    LEFT JOIN Qualification ON idPilote = unPilote
GROUP BY idPilote
;

SELECT *
FROM vue_nbQualification
;
```

RESULT

```
View VUE_NBQUALIFICATION créé(e).

  IDPILOTE NB_QUALIFICATION
---------- ----------------
         1                2
         2                2
         3                1
         4                3
         5                2
         6                0
         7                2
```

piloteExperimente (1 ou NULL) dans Pilote

```
CREATE OR REPLACE VIEW vue_Pilote_Experimente
AS
SELECT idPilote, 1 est_Experimente
FROM Pilote
WHERE nbHVol >= (650*4)
;

SELECT *
FROM vue_Pilote_Experimente
;
```

RESULT

```
View VUE_PILOTE_EXPERIMENTE créé(e).

   IDPILOTE EST_EXPERIMENTE
___________ _______________
          4               1
          7               1
```

Pilote complétée

```
CREATE OR REPLACE VIEW vue_Pilote_Complete
AS
SELECT P.idPilote, nomPilote, nbHVol, compPil, nb_Qualification,
     est_Experimente
FROM Pilote P
     JOIN vue_nbQualification vue_Q ON P.idPilote = vue_Q.idPilote
         LEFT JOIN vue_Pilote_Experimente vue_P ON P.idPilote =
     vue_P.idPilote
;

SELECT idPilote, nomPilote, nb_Qualification, est_Experimente
FROM vue_Pilote_Complete
;
```

RESULT

```
View VUE_PILOTE_COMPLETE créé(e).
```

IDPILOTE	NOMPILOTE	NB_QUALIFICATION	EST_EXPERIMENTE
1	Ridard	2	
4	Fleurquin	3	1
5	Pham	2	
6	Kerbellec	0	
3	Godin	1	
7	Kamp	2	1
2	Naert	2	