

## R2.06 Exploitation BDD

M. T. Pham, M. Khayata

[minh-tan.pham@univ-ubs.fr](mailto:minh-tan.pham@univ-ubs.fr)

Cours 4

# Vues

# Plan

---

**Définition et Syntaxe**

**Garantir la confidentialité**

**Garantir l'intégrité**

# Définition

- Les **vues** en deux mots : des tables virtuelles
- Les vues en une phrase : une vue est une table (virtuelle) qui est le résultat d'une requête (**SELECT**) à laquelle on a donné un nom
- Le nom d'une vue peut être utilisé partout où on peut mettre le nom d'une table : **SELECT, UPDATE, DELETE, INSERT, GRANT**

# Création d'une vue : syntaxe



```
CREATE [OR REPLACE] VIEW nomVue  
[ (attr1, ..., attrn) ]  
AS requêteSELECT  
[WITH CHECK OPTION [CONSTRAINT nomContrainte]]  
[WITH READ ONLY]  
;
```

## Remarque:

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

# Utilisation d'une vue

- Pour une interrogation (requête)



```
SELECT ...  
FROM nomVue  
WHERE ...;
```

- Suppression d'une vue



```
DROP VIEW nomVue;
```

# Exemples pratiques

**Compagnie** (idComp (1), nomComp, pays, estLowCost)

**Pilote** (idPilote(1), nomPilote, nbHVol, compPil=@Compagnie(idComp))

**TypeAvion** (idTypeAvion(1),nbPassagers)

**Qualification** (unPilote=@Pilote(idPilote)(1),unTypeAvion=@TypeAvion(idTypeAvion)(1))

**Avion** (idAvion(1), leTypeAvion=@TypeAvion(idTypeAvion)(NN), compAv=@Compagnie(idComp)(NN) )

## Compagnie

idComp	nomComp	pays	estLowCost
1	Air France	France	0
2	Corsair International	France	0
3	EasyJet	Angleterre	1
4	American Airlines	Etats-Unis	0
5	Ryanair	Irlande	1

## TypeAvion

idTypeAvion	nbPassagers
A320	174
A350	324
B747	279

## Qualification

unPilote	unTypeAvion
1	A320
1	A350
2	A320
2	B747
3	A320
4	A320
4	A350
4	B747
5	A350
5	A320
7	A350
7	B747

## Pilote

idPilote	nomPilote	nbHVol	compPil
1	Ridard	1500	1
2	Naert	450	3
3	Godin	450	5
4	Fleurquin	3000	1
5	Pham	900	4
6	Kerbellec	900	
7	Kamp	3000	4

## Avion

idAvion	leTypeAvion	compAv
1	A320	1
2	A320	3
3	A350	1
4	A320	2
5	B747	1
6	A350	4
7	B747	4
8	A320	5
9	A320	5

# Exemples pratiques



## 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

# Exemples pratiques



## 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



# Exemples pratiques



## 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

# Pourquoi utiliser des vues ?

- Remplacer une requête compliquée par des requêtes plus simples
- Garantir la confidentialité
- Garantir l'intégrité

# Plan

---

Définition et Syntaxe

**Garantir la confidentialité**

Garantir l'intégrité

# Garantir la confidentialité

- Un SGBD garantit **la confidentialité** des données en gérant :
  - l'accès aux données avec les **utilisateurs**, les **rôles** et les **privileges** (en consultation) → on va étudier ce point dans le Cours 5.
  - le **niveau externe** avec les vues qui agissent comme des "fenêtres ». *En comparaison avec les niveaux:*
    - *conceptuel (diagramme UML)*
    - *logique (schéma relationnel)*
    - *physique (base de données)*
- Une vue peut servir à "**cacher**" certaines informations (colonnes et/ou lignes) sans pour autant priver complètement l'utilisateur

# Garantir la confidentialité

Pour créer une vue, l'utilisateur "connecté" doit en posséder le privilège<sup>a</sup>.

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<sup>b</sup>, 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 dans le Cours 5

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

# Garantir la confidentialité



Vous devez alors vous connecter, depuis Oracle SQL Developer ou Run SQL, en tant que SYSTEM avec le mot de passe<sup>a</sup> 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 ;
```

# Plan

---

Définition et Syntaxe

Garantir la confidentialité

**Garantir l'intégrité**

# Garantir l'intégrité

- Un SGBD garantit *l'intégrité* des données en gérant :
  - 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 avec les utilisateurs, les rôles et les privilèges
  
- **La cohérence** des données, elle est assurée en gérant :
  - les contraintes
  - la redondance (potentiellement incohérente)



# Garantir l'intégrité

- **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 la création de tables (déjà vu en R1.05)
  - Les autres (surjectivité, vrais cycles non modifiables, ...) sont programmées :
    - au niveau du client (application) en s'appuyant, éventuellement, sur des **vues**
    - au niveau du serveur (SGBD) en PL/SQL avec des **déclencheurs** (à voir en BUT2)
  
- **La redondance** est gérée grâce à :
  - La **normalisation** (déjà vue en R1.05)
  - La **dérivation** (attribut ou association) avec **les vues**

# Exemples pratiques

**Compagnie** (idComp (1), nomComp, pays, estLowCost)

**Pilote** (idPilote(1), nomPilote, nbHVol, compPil=@Compagnie(idComp))

**TypeAvion** (idTypeAvion(1),nbPassagers)

**Qualification** (unPilote=@Pilote(idPilote)(1),unTypeAvion=@TypeAvion(idTypeAvion)(1))

**Avion** (idAvion(1), leTypeAvion=@TypeAvion(idTypeAvion)(NN), compAv=@Compagnie(idComp)(NN) )

## Compagnie

idComp	nomComp	pays	estLowCost
1	Air France	France	0
2	Corsair International	France	0
3	EasyJet	Angleterre	1
4	American Airlines	Etats-Unis	0
5	Ryanair	Irlande	1

## TypeAvion

idTypeAvion	nbPassagers
A320	174
A350	324
B747	279

## Qualification

unPilote	unTypeAvion
1	A320
1	A350
2	A320
2	B747
3	A320
4	A320
4	A350
4	B747
5	A350
5	A320
7	A350
7	B747

## Pilote

idPilote	nomPilote	nbHVol	compPil
1	Ridard	1500	1
2	Naert	450	3
3	Godin	450	5
4	Fleurquin	3000	1
5	Pham	900	4
6	Kerbellec	900	
7	Kamp	3000	4

## Avion

idAvion	leTypeAvion	compAv
1	A320	1
2	A320	3
3	A350	1
4	A320	2
5	B747	1
6	A350	4
7	B747	4
8	A320	5
9	A320	5

# Exemples pratiques



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<sup>a</sup>
- nbQualification dans Pilote
- piloteExperimente (1 ou NULL) dans Pilote<sup>b</sup>

---

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

# Exemples pratiques

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
```

# Exemples pratiques

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
---

# Exemples pratiques

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
```

# Exemples pratiques



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<sup>a</sup>
- nbQualification dans Pilote
- piloteExperimente (1 ou NULL) dans Pilote<sup>b</sup>

- 
- 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



# Exemples pratiques

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



# Exemples pratiques

## 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

# Exemples pratiques

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

# Exemples pratiques

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

# Exemples pratiques

## 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	

# Références

- <https://fr.wikipedia.org/>
- Cours BDD Anthony Ridard, <https://math-ridard.fr/>
- Cours BDD Francesca Fiorenzi, <https://www.lri.fr/~fiorenzi/>
- Christian Soutou, *Modélisation des bases de données : UML et les modèles entité-association*, 3<sup>ème</sup> édition, Groupe Eyrolles.
- Christian Soutou, *SQL pour Oracle : Optimisation des requêtes et schémas*, 5<sup>ème</sup> édition, Groupe Eyrolles.
- Elisabetta De Mria, Cours L2 Informatique, UFR Sciences, Université Côte d'Azur, <https://www.i3s.unice.fr/%18edemaria/>
- Oracle SQL Tutorials, <https://www.w3schools.com/sql/default.asp>
- MySQL Tutorials, <https://www.w3schools.com/MySQL/default.asp>