Les vues

Outre le contrôle de l'accès aux données (privilèges), la confidentialité des informations est un aspect important qu'un SGBD relationnel doit prendre en compte. La confidentialité est assurée par l'utilisation de vues (views). Les vues correspondent à ce qu'on appelle le niveau externe qui reflète la partie visible de la base de données pour chaque utilisateur.

Seules les tables contiennent des données et pourtant, pour l'utilisateur, une vue apparaît comme une table. En théorie, les utilisateurs ne devraient accéder aux informations qu'en questionnant des vues. **Ces dernières masquant la structure des tables interrogées**. En pratique, beaucoup d'applications se passent de ce concept en manipulant directement les tables.

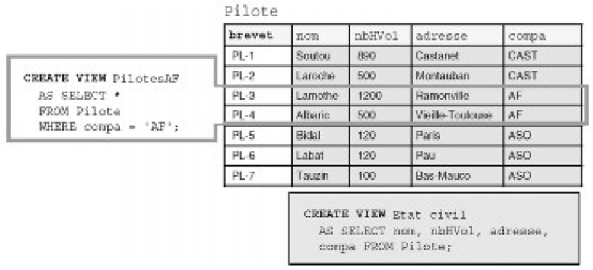
Outre le fait d'assurer la confidentialité des informations, une vue est capable de réaliser des contrôles de contraintes d'intégrité et de simplifier la formulation de requêtes complexes.

1. Création d'une vue (CREATE VIEW)

Pour pouvoir créer une vue dans votre schéma vous devez posséder le privilège CREATE VIEW. Pour créer des vues dans d'autres schémas, le privilège CREATE ANY VIEW est requis. Des exemples de création en SQL sont donnés plus tard. Pour pouvoir supprimer une vue, vous devez en être propriétaire ou posséder le privilège DROP ANY VIEW.

1.1 Vues monotables

Les mécanismes présentés ci-après s'appliquent aussi, pour la plupart, aux vues multitables. Considérons les deux vues illustrées par la figure suivante et dérivées de la table Pilote. La vue PilotesAF décrit les pilotes d'Air France à l'aide d'une restriction (éléments du WHERE). La vue Etat_civil est constituée par une projection de certaines colonnes (éléments du SELECT).



Une fois créée, une vue s'interroge comme une table par tout utilisateur, sous réserve qu'il ait obtenu le privilège en lecture directement (GRANT SELECT ON nomVue TO…) ou via un rôle.

CREATE TABLE pilote (brevet CHAR(8), nom CHAR(10), nbHVol INTEGER, adresse CHAR(20), compa CHAR(8), CONSTRAINT pk_pilote PRIMARY KEY(brevet));

CREATE VIEW PilotesAF
AS SELECT * FROM Pilote
WHERE compa = 'AF';

CREATE VIEW Etat_civil
AS SELECT nom, nbHVol, adresse, compa
FROM Pilote;

Question 1 Donnez la requête d'interrogation des vues pour obtenir :

- Somme des heures de vol des pilotes d'Air France.
- Nombre de pilotes.

À partir de cette table et de ces vues, nous allons étudier certaines autres options de l'instruction CREATE VIEW.

2. Alias

Les alias, s'ils sont utilisés, désignent le nom de chaque colonne de la vue. Ce mécanisme permet de mieux contrôler les noms de colonnes. Quand un alias n'est pas présent, la colonne prend le nom de l'expression renvoyée par la requête de définition. Ce mécanisme sert à **masquer les noms des colonnes de l'objet source**.

Les vues suivantes sont créées avec des alias qui masquent le nom des colonnes de la table source.

Écriture 1	Écriture 2
`CREATE OR REPLACE VIEW`	`CREATE OR REPLACE VIEW`
`PilotesPasAF`	`PilotesPasAF`
`(codepil,nomPil,heuresPil, adressePil, société)`	`AS SELECT brevet codepil, nom nomPil, nbHVol`
`AS SELECT *`	`heuresPil , adresse adressePil, compa société`
`FROM Pilote`	`FROM Pilote`
`WHERE NOT (compa = 'AF');`	`WHERE NOT (compa = 'AF');`

L'objet source d'une vue est en général une table mais peut aussi être une vue. La vue suivante est définie à partir de la vue PilotesPasAF précédemment créée.

```
CREATE OR REPLACE VIEW
EtatCivilPilotesPasAF
AS SELECT nomPil,heuresPil,adressePil
FROM PilotesPasAF ;
```

Question 2 : Est-il possible d'utiliser des alias pour renommer à nouveau les colonnes de la nouvelle vue ?

3. Vues en lecture seule

L'option WITH READ ONLY déclare la vue non modifiable par INSERT, UPDATE, ou DELETE. Redéfinissons la vue PilotesPasAF à l'aide de cette option.

```
CREATE OR REPLACE VIEW PilotesPasAFRO
AS SELECT *
FROM  Pilote
WHERE NOT (compa = 'AF')
WITH READ ONLY;
```

Question 3 : Exécuter les 3 requêtes suivantes puis expliquez pourquoi la mise à jour est ou n'est pas possible.

```
INSERT INTO PilotesPasAFRO VALUES
       ('PL-8', 'Ferry', 5, 'Paris', 'ASO');
UPDATE PilotesPasAFRO
      SET nbHvol=nbHvol+2;
DELETE FROM PilotesPasAFRO ;
```

4. Vues modifiables

Lorsqu'il est possible d'exécuter des instructions INSERT, UPDATE ou DELETE sur une vue, cette dernière est dite modifiable (updatable view). Vous pouvez créer une vue qui est modifiable intrinsèquement. Les mises à jour sont automatiquement répercutées au niveau d'une ou de plusieurs tables.

Pour mettre à jour une **vue simple**, il doit exister une correspondance biunivoque entre les lignes de la vue et celles de l'objet source. De plus, certaines conditions doivent être remplies (pas de distinct, pas de fonction, pas d'order by, pas de group by, pas de having).

Question 4 : Est-il possible d'effectuer ces mises à jour sur la vue Etat-civil ? Expliquez.
- Suppression des pilotes (d'une compagnie donnée)
- Un pilote donné double ses heures
- Ajout d'un pilote

Question 5 : Est-il possible d'effectuer ces mises à jour sur la vue PilotesAF ? Expliquez.
- Ajout d'un pilote AF
- Les pilotes AF doublent leurs heures
- Suppression d'un pilote AF
- Ajout d'un pilote qui n'est pas de 'AF'

5. Directive CHECK OPTION

La directive WITH CHECK OPTION empêche un ajout ou une modification non conforme à la définition de la vue. Interdisons l'ajout (ou la modification de la colonne compa) d'un pilote au travers de la vue PilotesAF, si le pilote n'appartient pas à la compagnie de code 'AF'.
Il est nécessaire de redéfinir la vue PilotesAF comme suit :

```
CREATE OR REPLACE VIEW PilotesAF
AS SELECT * FROM  pilote
WHERE compa = 'AF'
WITH CHECK OPTION;
```

Question 6 : Validez le bon fonctionnement de la clause WITH CHECK OPTION sur la vue PilotesAF par l'ajout d'un nouveau pilote qui n'est pas AF

6. Vues complexes

Une vue complexe est caractérisée par le fait de contenir, dans sa définition, plusieurs tables (jointures), et une fonction appliquée à des regroupements. La mise à jour de telles vues n'est pas toujours possible.

La vue multitable Pilotes_multi_AF est créée à partir d'une jointure entre les tables Compagnie et Pilote tandis que la vue Moyenne_Heures_Pil est créée à partir d'un regroupement de la table Pilote comme suit :

```
DROP TABLE Pilote;
CREATE TABLE Compagnie
        (comp CHAR(4), nrue NUMBER(3),
        rue CHAR(20), ville CHAR(15) DEFAULT 'Paris', nomComp CHAR(15),
        CONSTRAINT pk_Compagnie PRIMARY KEY(comp));

CREATE TABLE Pilote
        (brevet CHAR(6), nom CHAR(15), nbHVol NUMBER(7,2), compa CHAR(4),
         CONSTRAINT pk_Pilote PRIMARY KEY(brevet),
         CONSTRAINT nn_nom    CHECK (nom IS NOT NULL),
         CONSTRAINT fk_Pil_compa_Comp FOREIGN KEY(compa) REFERENCES
Compagnie(comp));

CREATE VIEW Pilotes_multi_AF
      AS SELECT  p.brevet, p.nom, p.nbHVol, c.ville, c.nomComp
         FROM  Pilote p, Compagnie c
         WHERE p.compa = c.comp
         AND   p.compa = 'AF';

CREATE VIEW Moyenne_Heures_Pil
 AS SELECT compa, AVG(nbHVol) moyenne
 FROM Pilote GROUP BY compa;
```

Question 7 : Peut-on insérer dans les deux vues ? Expliquez.

7. Cas d'une vue complexe avec table protégée par clé

Alors que la vue montable Moyenne_Heures_Pil n'est pas modifiable, ni par UPDATE ni par DELETE (message d'erreur ORA-01732), la vue multitable Pilotes_multi_AF est transformable dans une certaine mesure, car la table Pilote (qui entre dans sa composition) est dite « protégée par clé » (key preserved). Une table est dite protégée par sa clé (key preserved) si sa clé primaire est préservée dans la clause de jointure et se retrouve en tant que colonne de la vue multitable (peut jouer le rôle de clé primaire de la vue).

Question 8 : Exécuter les requêtes suivantes et expliquez les résultats :

```
UPDATE Pilotes_multi_AF
SET nbHVol =  nbHVol * 2;
SELECT * FROM Pilotes_multi_AF;

UPDATE Pilotes_multi_AF
SET ville =  'autrenomville';
SELECT * FROM Pilotes_multi_AF;

DELETE FROM Pilotes_multi_AF ;
SELECT * FROM Pilote;
SELECT * FROM Compagnie;
```

Afin de savoir dans quelle mesure les colonnes d'une vue sont modifiables (en insertion ou suppression), on peut interroger la vue USER_UPDATABLE_COLUMNS du dictionnaire des données. La fonction UPPER peut être utilisée pour convertir en majuscules le nom de la table car tout est codé en majuscules dans le dictionnaire des données.

Question 9 : Exécuter la requête suivante et vérifier vos résultats précédents.

```
SELECT COLUMN_NAME, INSERTABLE,
UPDATABLE, DELETABLE
FROM  USER_UPDATABLE_COLUMNS
WHERE TABLE_NAME = UPPER('Pilotes_multi_AF');
```

8. Autres utilisations de vues

Les vues peuvent également servir pour renforcer la sécurité et la confidentialité, et simplifier des requêtes complexes.

```
DROP TABLE Pilote;
DROP TABLE Compagnie;

CREATE TABLE Compagnie
        (comp CHAR(4), nrue NUMBER(3),
        rue CHAR(20), ville CHAR(15), nomComp CHAR(15),
        CONSTRAINT pk_Compagnie PRIMARY KEY(comp));
CREATE TABLE Pilote
        (brevet CHAR(6), nom CHAR(15), nbHVol NUMBER(7,2), compa CHAR(4),
         CONSTRAINT pk_Pilote PRIMARY KEY(brevet));
```

8.1 Variables d'environnement

Une requête de définition d'une vue peut utiliser des fonctions SQL relatives aux variables d'environnement d'Oracle. A titre d'exemple la fonction USERENV('TERMINAL') retourne l'identifiant du terminal pour la session courante. C'est le même identifiant terminal que dans la vue V$SESSION.

Voic un exemple de la vue Said_Atlas_PilotesAF qui restituera les pilotes de la compagnie 'AF' pour uniquement l'utilisateur Said, ou pour un utilisateur connecté au terminal Atlas sous une version Oracle française.

```
CREATE VIEW Said_Atlas_PilotesAF
AS SELECT * FROM  Pilote WHERE compa = 'AF'
AND USER = 'Said'
OR (USERENV('TERMINAL')      =    'Atlas'
AND USERENV('LANGUAGE') LIKE 'FRENCH_FRANCE%');
```

8.2 Confidentialité

La confidentialité est une des vocations premières des vues. Outre l'utilisation de variables d'environnement, il est possible de restreindre l'accès à des tables en fonction des moments.

Question 10 : Les vues suivantes restreignent l'accès à des moments précis. Quel est le résultat retourné si une restriction n'est pas satisfaite ?

```
CREATE VIEW VueDesCompagniesJoursFériés
  AS SELECT * FROM Compagnie
  WHERE TO_CHAR(SYSDATE, 'DAY') IN
        ('SAMEDI', 'DIMANCHE');

CREATE VIEW VueDesPilotesJoursOuvrables
AS SELECT * FROM Pilote
WHERE TO_CHAR(SYSDATE,'HH24:MI')
      BETWEEN '8:30' AND '17:30'
AND TO_CHAR(SYSDATE,'DAY')
      NOT IN ('SAMEDI', 'DIMANCHE' )
WITH CHECK OPTION;
```

Question 11 : Redéfinir la vue `VueDesCompagniesJoursFériés` en limitant son utilisation à un utilisateur user1 uniquement ? Testez le fonctionnement de cette limitation.

9. Transmission des droits

Si un utilisateur désire transmettre des droits sur une partie d'une de ses tables, il utilisera une vue. Ainsi, seules les données appartenant à la vue seront accessibles aux bénéficiaires.

Les privilèges objets qu'il est possible d'attribuer sur une vue sont les mêmes que ceux applicables sur les tables (SELECT, INSERT, UPDATE sur une ou plusieurs colonnes, DELETE). Voici un exemples d'attribution de privilèges sur la vue `Pilotes_multi_AF`.

```
GRANT SELECT ON
Pilotes_multi_AF TO user1;
```

Question 12 : Vérifiez que l'accès aux données de la table compagnie est bien partiel pour l'utilisateur user1.