Plan du module

- **Partie 1- I. Introduction aux SGBDs**
  - Chapitre 1: Présentation des SGBDs
  - Chapitre 2: Rappel. Définition et Evolution des données
  - Chapitre 3: Contrôle des données
  - **Chapitre 4: Gestion des objets utilisateurs**
    - **(Vues, séquences et Index)**
- **Partie 2- II. Langage procédural: PL/SQL**
- **Partie 3- III. Gestion des Transactions**

ENICARTHAGE
المدرسة الوطنية للمهندسين بقرطاج
Ecole Nationale d'Ingénieurs de Carthage

I.4 Séquences, Vues & Index

2ème Ingénieurs info

Année Universitaire 2020-2021

B- Les vues virtuelles

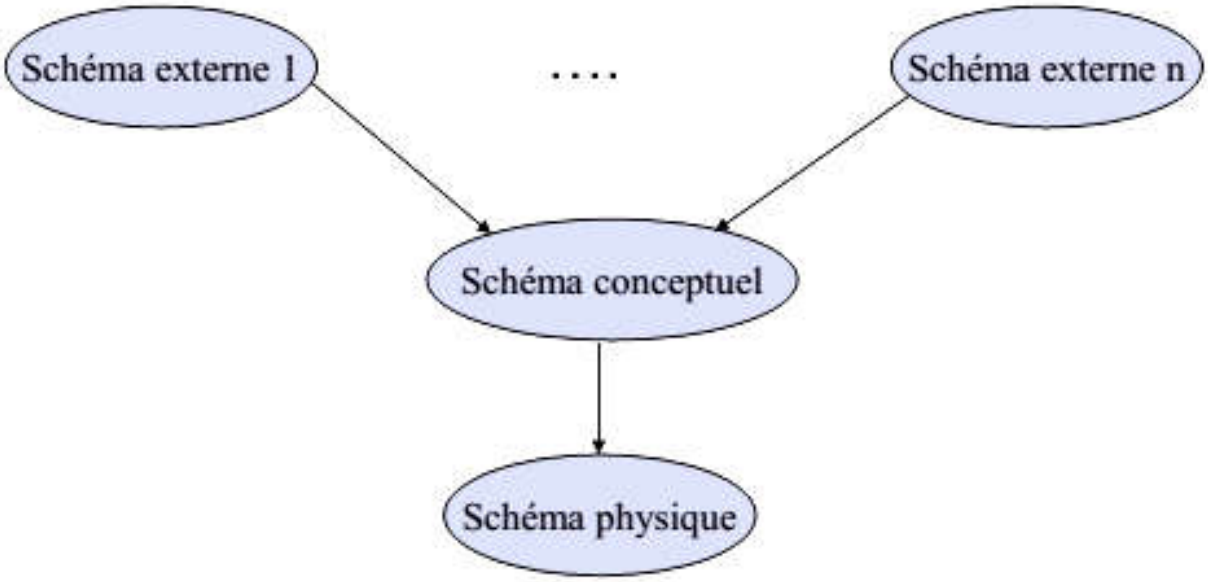
Plan

- Introduction
- Définitions d'une vue.
  - Création.
  - Suppression.
  - Renommage.
  - Interrogation.
- Utilités des vues.
- Mise à jour au travers des vues.
- Cas d'Oracle.

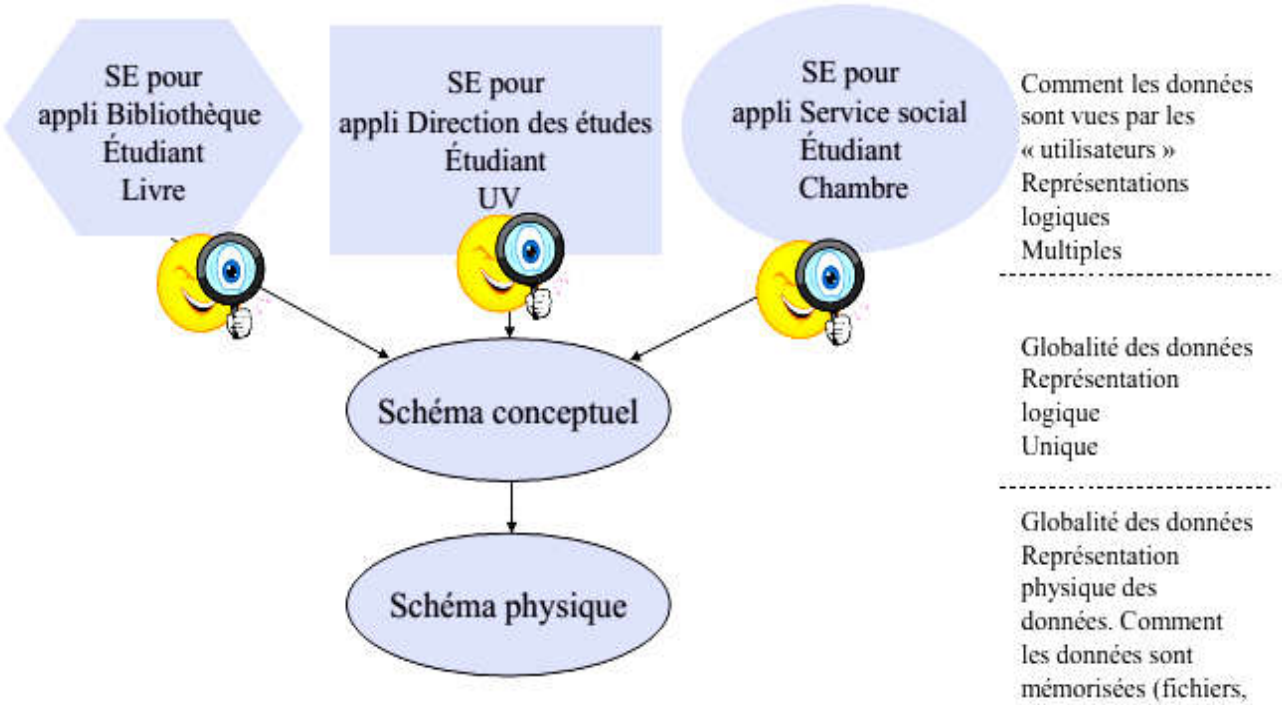
Introduction

- **Groupe ANSI/X3/SPARC (1975)**

3 niveaux d'abstraction



Introduction



Exemple d'une vue

1. Définitions

- Une vue est le résultat d'une requête à laquelle on a donné un nom:
  - Une vue est créée à l'aide d'une instruction SELECT appelée « requête de définition ».
  - Cette requête interroge une ou plusieurs table(s) ou vue(s).
- C'est une table virtuelle.
  - Ensemble de tuples qui n'existe pas physiquement
  - Ne nécessite aucune allocation en mémoire pour contenir les données.
  - Seule sa structure est stockée dans le dictionnaire de données.
  - Calculable à l'exécution: Une vue se recharge chaque fois qu'elle est interrogée.
- La vue sera vue par l'utilisateur comme une table réelle.
- Peut être utilisée pour définir une autre vue
- Le nom d'une vue peut être utilisé partout où on peut mettre le nom d'une table :
  - **SELECT**, **UPDATE**, **DELETE**, **INSERT**, **GRANT**

1. Définitions

Création d'une vue : Syntaxe Générale

```
CREATE VIEW nom-de-la-vue[Liste d'attributs]
AS  <requête de définition SELECT>
[WITH CHECK OPTION]
```

- La spécification des noms de colonnes (attributs) de la vue est facultative.
- Par défaut, les colonnes de la vue ont pour nom les noms des colonnes résultat de SELECT.
- La clause **WITH CHECK OPTION** empêche que l'utilisateur ajoute ou modifie dans une vue des lignes non conformes à la définition de la vue.

- *Exemple1*: Créer une vue des élèves ingénieurs du département informatique:

```
CREATE VIEW v_Ing_info (num_etud, nom, prenom, niveau,
specialite)
AS SELECT * FROM Etudiants
WHERE specialite = 'G-informatique'
```

1. Définitions

- *Exemple 2*: créer une vue des employés du département 10 (NUM, Nom, fonction):
  - Emp (Num ,Nom,FONCTION, NUMSUP, Embauche, SAL, COMM,DEPTNO)

```
SQL> CREATE VIEW      empvu10
  2  AS SELECT       NUM, NOM, FONCTION
  3  FROM            emp
  4  WHERE           deptno = 10;
View created.
```

```
NUM     NOM    FONCTION  NUMSUP EMBAUCHE   COMM  DEPTNO
```

```
 NUM    NOM      FONCTION
------ -------- -----------
  7ti39 KING      PRESIDENT
  77ti2 CLARK     MANAGER
  7934 MILLER    CLERK
```

```
                                                    10
                                                    20
                                                     20
                                                      20
                                                    20
                                                   20
                                                      30
7654  MARTIN   SALESMAN  769ti 2ti-SEP-ti1   1250  1400      30
```

Vue EMPVU10

1. Définitions

- *Exemple 3:* Créer une vue à partir de plusieurs tables:
  - Exemple: créer une vue comportant le nom des employés, le nom du département dans lequel ils exercent leurs fonctions et le lieu du travail.
  - Soit le schéma relationnel suivant:
    - Emp(Num,Nom,Fonction,NumSup,Embauche, Salaire, #DeptNo)
    - DEPT(DeptNo,Nom,Loc)

```
CREATE VIEW empDept (Nom_emp,
Nom_dept, loc_dept)
AS SELECT e.nom, d.nom, LOC
FROM EMP e, DEPT d
WHERE e.DEPTNO = d.DEPTNO;
```

1. Définitions

- **Suppression d'une vue**

```
DROP VIEW nom-de-la-vue
```

  - La suppression d'une vue n'entraîne pas la suppression des données.
  - Toutes les vues qui utilisent cette vue sont automatiquement détruites.
  - Les vues figurent dans les tables systèmes **ALL_CATALOG, USER_VIEWS et ALL_VIEWS**
    - USER_* :Décrit les objets qui appartiennent à l'utilisateur courant .
    - .ALL_* Décrit les objets qui sont accessibles à l'utilisateur courant .
    - DBA_* Décrit tous les objets (ces vues ne sont accessibles que par l'administrateur)

- **Renommage d'une vue**

```
RENAME ancien-nom TO nouveau-nom
```


1. Définitions

- **Interrogation:**
  - Pour récupérer les données de vues, on procédera comme si l'on était en face d'une table classique.

➔*Exemple 4*: `SELECT * FROM empDept`…

[Vues matérialisées]

- Attention: il existe des vues réelles dites concrètes:
  - Vue « réelle » : fenêtre sur le contenu de la base de données MAIS …
    - Objet réel : recopie physique des données concernées
    - Actualisation périodique : mise à jour automatique avec les données

	Vue	Vue concrète
Données observées	virtuelles	réelles
Accès aux données sources	direct	indirect
Mise à jour des données	immédiate	périodique / sur demande

    - Pré-agrégation des données - data warehouse (entrepôt de données)
    - Pré-récupération des données - bases de données distribuées
    - Sécurisation des données - plus d'accès aux données sources

[Vues matérialisées]

- Syntaxe:

```
CREATE MATERIALIZED VIEW <nom_vue_concrete>
TABLESPACE <nom_tablespace>
BUILD [IMMEDIATE | DEFERRED]
[ENABLE QUERY REWRITE]
REFRESH [FAST | COMPLETE | FORCE |NEVER]
[START WITH date] [NEXT date]]
AS <requête SQL> ;
```

Dans cette syntaxe on définit:

•Chargement des données à la création ou lors de la première requête

• Fréquence d'actualisation des données

• Mode d'actualisation des données : recopie complète ou seulement incrémentale (plus rapide).

• Optimisation par réécriture de la requête de définition : forcer l'optimisateur de requête à réécrire son plan d'exécution pour accélérer l'ex

2. Utilité des vues

Les vues permettent:

- Des accès simplifiés aux données:
  - Effet macro : remplacer une requête compliquée nécessitant plusieurs étapes par des requêtes plus simples.
  - Masquer la complexité du schéma de la base.
- Une indépendance logique des données:
  - le programme restera invariant aux modifications de schéma s'il accède à la base via une vue qui l'isole de celle-ci.
- La sécurité/ Confidentialité des données:
  - L'utilisateur ne peut accéder qu'aux données des vues auxquelles il a droit d'accès
- L'Intégrité de la base:
  - Lorsqu'elles sont utilisées pour les mises à jour : on dit alors qu'on effectue une mise à jour au travers d'une vue.

2.1- Assurer la confidentialité des données

Restreindre l'accès à certaines colonnes;

- *Exemple 6:* donner au service comptable *(service_comptable)* d'une entreprise la possibilité du suivi des salaires de façon anonyme (sans savoir le nom) pour les titulaires de grands salaires à partir de la table
  - emp (num, nom, prenom, adresse, sal)
- l'utilisateur service_comptable ne peut connaître ni le nom, ni le prénom, ni l'adresse des employés gagnant plus de 1 0 000 d

1)
```
CREATE VIEW V_employés_GSal
AS SELECT num,sal,
FROM Emp
WHERE sal> 10000
```

2) Accorder le privilège au service_comptale (un utilisateur identifié sur la base).

```
GRANT SELECT ON V_employés_Gsal TO
service_comptale;
```

2.2- Assurer la sécurité des données

Emettre des restrictions d'accès en fonction du contexte.

- **Utilisation des variables d'environnement**
  - Une requête de définition d'une vue peut utiliser des fonctions SQL relatives aux variables d'environnement d'Oracle.
  - Le tableau suivant décrit ces variables :

Variable / Fonction	Signification	
`USER`	Nom de l'utilisateur connecté. *(Qui démarre la session et lance le programme d'application)*	
`UID`	Numéro d'identification de l'utilisateur connecté.	
`USERENV('paramètre')`	Fonction utilisant un des paramètres ci-contre.	`SESSIONID` : numéro de la session.
		`TERMINAL` : nom du terminal dans le système d'exploitation hôte.
		`ENTRYID` : numéro chronologique de la commande SQL dans la session.
		`LANGUAGE` : langage utilisé.

2.2- Assurer la sécurité des données

Emettre des restrictions d'accès en fonction du contexte.

Exemple 7 : Créer une vue et octroyer les droits aux utilisateurs concernés par cette vue (un utilisateur peut visualiser ses propres infos)

Soit la relation Personnel(num_pers, nom, salaire, prime_mois, adresse, tel, email, #depno)

RQ: on suppose que l'email est utilisé comme username dans l'application de gestion des employés de l'entreprise

1) Chaque utilisateur peut visualiser un seul tuple de la relation: celui qui le concerne.

```
CREATE VIEW v_info_privee
AS SELECT * FROM personnel
WHERE upper(email)=upper(USER) :  permet de définir une vue dépendant du contexte en utilisant la variable d'environnement USER
```

2.2- Assurer la sécurité des données

2) Donner les privilèges de lecture sur la vue à chaque utilisateur:

```
GRANT SELECT ON v_info_privee TO PUBLIC;
```

- Attribuer maintenant les privilèges *de modification sur certaines colonnes de la vue* ***(adresse, tel):***
- *Exemple 8*: **Personnel(num_pers, nom, salaire, prime_mois, adresse, tel,email, #depno)**
  - ```
    CREATE VIEW v_info_privee_employe
    AS SELECT * from Employé
    WHERE upper(email)=upper(user);
    ```

```
GRANT SELECT, UPDATE (adresse, tel)
  ON v_info_privee_employe TO PUBLIC;
```


2.2- Assurer la sécurité des données

Emettre des restrictions d'accès en fonction du contexte.

- Outre l'utilisation de variables d'environnement, il est possible de restreindre l'accès à des tables **en fonction du temps.**
  Les vues suivantes limitent temporellement les accès en lecture et en écriture à des tables:
- *Exemple 9:*

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

- *Restriction, en mise à jour de la table Compagnie, les samedi et dimanche. Lecture possible à tout moment*

Attribution du privilège	Signification
`GRANT SELECT ON VueDesCompagniesJoursFériés TO PUBLIC;`	Accès pour tous en lecture sur la vue VueDesCompagniesJoursFériés.
`GRANT INSERT ON VueDesCompagniesJoursFériés TO Paul;`	Accès pour *Paul* en écriture sur la vue VueDesCompagniesJoursFériés.
```

3. Mise à jour au travers des vues

Pour qu'une vue soit modifiable, il faut respecter les conditions suivantes :

1. L'expression de table associée à la vue doit être un simple SELECT, elle ne peut donc contenir les termes **JOIN**, **INTERSECT**, **UNION** ou **EXCEPT/MINUS** ;
2. La clause **FROM** ne peut contenir qu'une seule table de base ou une vue elle-même modifiable ;
3. L'expression **SELECT** ne peut contenir la clause **DISTINCT** ;
4. La liste des colonnes du **SELECT** ne peut comporter d'expression ;
5. Si le SELECT contient une requête imbriquée, celle-ci ne peut faire référence à la même table que la requête externe ;
6. Pas de fonction de calcul (AVG, COUNT, MAX, MIN, STDDEV, SUM, ou VARIANCE),
7. La requête SELECT ne peut contenir ni **GROUP BY**, ni **HAVING**.

Si une de ces conditions n'est pas remplie, la vue n'est pas modifiable : impossibilité d'utiliser les commandes **INSERT INTO**, **DELETE** ou **UPDATE**.

4. Cas d'Oracle

Syntaxe de création:

```
CREATE [OR REPLACE] [[NO]FORCE] VIEW NomVue [schéma]
[WITH { READ ONLY |CHECK OPTION [CONSTRAINT
nomContrainte] } ];
```

- **OR REPLACE**: remplace la vue par la nouvelle définition même si elle existait déjà (évite de détruire la vue avant de la recréer).
- **FORCE** pour créer la vue sans vérifier si les sources qui l'alimentent existent, ou si les privilèges adéquats (SELECT, INSERT, UPDATE, ou DELETE) sur ces objets sont acquis par l'utilisateur qui crée la vue. Par défaut c'est NO FORCE qui est utilisée.
- **WITH READ ONLY** déclare la vue non modifiable par INSERT, UPDATE, ou DELETE
- **WITH CHECK OPTION** garantit que toute mise à jour de la vue par INSERT ou UPDATE s'effectuera conformément à la définition de la vue.
- **CONSTRAINT** *nomContrainte* nomme la clause CHECK OPTION sous la forme d'un nom de contrainte.

4. Vues sous Oracle

Exemple 10:

```
SQL> CREATE OR REPLACE VIEW v_empDep20
  2  AS SELECT       *
  3  FROM   emp
  4  WHERE  deptno = 20
  5  WITH CHECK OPTION CONSTRAINT v_empDep20_ck;
View created.
```

- Toute tentative de modification du numéro de département dans une ligne de la vue échouera, car elle transgresse la contrainte **WITH CHECK OPTION**.

4. Vues sous Oracle

Exemple 11:

```
SQL> CREATE OR REPLACE VIEW empvu10
  2        (employee_number, employee_name, job_title)
  3  AS SELECT       empno, ename, job
  4  FROM            emp
  5  WHERE           deptno = 10
  6  WITH READ ONLY;
View created.
```

- Toute tentative d'exécution d'un ordre du LMD sur une ligne de la vue génère l'erreur Oracle Server ORA-01752➔ Refus des Ordres du LMD

4. Vues sous Oracle

Etude de la modification au travers la vue

- Soit le schéma relationnel considéré dans les exemples suivants:
  - Companie(comp, nrue, rue, ville, nomComp)
  - Pilote(brevet, nom, nbHvol, #compa)

Compagnie

comp	nrue	rue	ville	nomComp
AF	124	Port Royal	Paris	Air France
SING	7	Camparols	Singapour	Singapore AL

Pilote

brevet	nom	nbHVol	compa
PL-1	Amélie Sulpice	450	AF
PL-2	Thomas Sulpice	900	AF
PL-3	Paul Soutou	1000	SING

4. Vues sous Oracle

- Vues monotables

Companie(comp, nrue, rue, ville, nomComp)
Pilote(brevet, nom, nbHvol,adresse, #compa)

  - *Exemple 12*: Soit la vue état-civil définie comme suit:

```
CREATE VIEW v_Etat_civil
AS SELECT   nom, nbHvol, adresse, compa
FROM        Pilote;
```

  - Dites si les opérations suivantes sont autorisées?
    - Suppression des pilotes de la companie ASO de la vue Etat_civil

```
DELETE FROM v_Etat_civil
WHERE compa = 'ASO';
```

➔ Autorisée

    - Le pilote *Yassin* double ses heures

```
UPDATE v_Etat_civil
SET nbHVol = nbHVol*2
WHERE nom = 'Yassin';
```

➔ Autorisée

4. Vues sous Oracle

- *Exemple 12 (suite)*

Companie(comp, nrue, rue, ville, nomComp)
Pilote(brevet, nom, nbHvol, #compa)

  - Ajout d'un pilote à travers la vue `(nom, nbHvol, adresse, compa)`

```
INSERT INTO v_Etat_civil
VALUES('Raffarin',10,'Poitiers','ASO');
```

➔**ORA-01400**: impossible d'insérer **NULL** dans (PILOTE.BREVET)

➔ Un tuple ne peut pas être inséré dans une vue qui ne contient pas la clé de la table.

4. Vues sous Oracle

- Impact de l'option **`WITH CHECK OPTION;`**
  - *Exemple 13* : créer une vue des pilotes appartenant à la compagnie aérienne 'AF' (clé)

Rappel du schéma:
•Companie(comp, nrue, rue, ville, nomComp)
•Pilote(brevet, nom, nbHvol, #compa)

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

- Opérations de mise à jour : Insertion d'un nouveau pilote:
  - **Cas 1:**

```
INSERT INTO v_PilotesAF VALUES
('PL-11','Alain',900,'Revel', 'AF' );
```

➔ 1 ligne créée.

4. Vues sous Oracle

- 2ème cas

```
INSERT INTO v_PilotesAF VALUES
('PL-10','Juppé',10,'Bordeaux', 'ASO' );
```

➔ **ORA-01402**: vue WITH CHECK OPTION violation de clause where

- *Exemple 13 (suite)*
  - Opérations de mise à jour:
    - Exemple de Modification de pilotes:

```
UPDATE PilotesAF SET compa='ASO'
```

- Impossible : violation de la définition de la vue. *
- Modification non effectuée grâce à WITH CHECK OPTION

4. Vues sous Oracle

- **Cas particulier des vues multi-tables:**

Rappel du schéma:

•*Companie(comp, nrue, rue, ville, nomComp)*

•*Pilote(brevet, nom, nbHvol, #compa)*

- *Exemple 14* : **Vue des pilotes AF avec la ville et le nom de la compagnie.**

```
CREATE OR REPLACE v_Pilotes_Multi_AF
AS SELECT p.Brevet, p.nom, p.nbHvol, c.ville, c.nomcomp
FROM pilote p, compagnie c
WHERE c.comp= p.compa AND p.compa= 'AF';
```

- **Insertion à travers une vue multi-table**
  - *Exemple 15* :Vue1: « Pilotes_multi_AF » (p.Brevet, p.nom, p.nbHvol, c.ville, c.nomcomp)

```
INSERT INTO v_Pilotes_multi_AF
VALUES('PL-4','Alex',400,'Paris','Aigle
Azur');
```

4. Vues sous Oracle

- **Modification et suppression à travers les vues multi-tables: Notion de table protégée:**
  - 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 multi-table (peut jouer le rôle de clé primaire de la vue).
  - Exemple 16:
    - p.Brevet, p.nom, p.nbHvol, c.ville, c.nomcomp)

```
UPDATE v_Pilotes_multi_AF
SET nbHVol = nbHVol *2
```

➔ 2 ligne(s) mise(s) à jour

```
DELETE FROM v_Pilotes_multi_AF;
```

➔ 2 ligne(s) mise(s) à jour

Il est à noter que seules les colonnes de la vue correspondant à la table protégée par clé peuvent être modifiées (ici nbHVol peut être mis à jour, en revanche, ville ne le peut pas)

```
SQL> SELECT * FROM v_Pilotes_multi_AF;
BREVET NOM            NBHVOL VILLE  NOMCOMP
------ -------------- ------ ------ ----------
PL-1   Amélie Sulpice    900 Paris  Air France
PL-2   Thomas Sulpice   1800 Paris  Air France
```

```
SQL> SELECT * FROM Pilote;
BREVET NOM                  NBHVOL COMP
------ --------------- ---------- ----
PL-3   Paul Soutou            1000 SING
```

```
SQL> SELECT * FROM Compagnie;
COMP  NRUE  RUE         VILLE      NOMCOMP
---- ------ ---------- --------- ------------
SING 7      Camparols  Singapour  Singapore AL
AF   124    Port Royal Paris      Air France
```


4. Vues sous Oracle

- *Exemple 16:*

```
CREATE VIEW Vue_Multi_Comp_Pil
 AS SELECT c.comp, c.nomComp,
    p.brevet, p.nom, p.nbHVol
 FROM  Pilote p, Compagnie c
 WHERE p.compa = c.comp;
```

```
SQL> SELECT * FROM Vue_Multi_Comp_Pil;

COMP NOMCOMP      BREVET NOM             NBHVOL
---- ------------ ------ -------------- --------
AF   Air France   PL-1   Amélie Sulpice      450
AF   Air France   PL-2   Thomas Sulpice      900
SING Singapore AL PL-3   Paul Soutou        1000
```

- Cela ne veut pas dire que cette vue est modifiable de toute manière.
  - Aucune insertion n'est permise dans la table compagnie.
  - Seules les modifications des colonnes de la table Pilote sont autorisées.
  - Les suppressions se répercuteront sur la table Pilote.
- La table préservée est la table Pilote, car la colonne brevet identifie chaque enregistrement extrait de la vue alors que la colonne comp ne le fait pas.

4. Dictionnaire de données

- **La table Système USER_UPDATABLE_COLUMNS**
- Afin de savoir dans quelle mesure les colonnes d'une vue sont modifiables (en insertion ou suppression), il faut interroger la vue
  - USER_UPDATABLE_COLUMNS du dictionnaire des données: c'est une table système qui contient la liste des colonnes modifiables des tables et des vues.
  - *Exemple 17*:

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

COLUMN_NAME	INS	UPD	DEL
COMP	NO	NO	NO
NOMCOMP	NO	NO	NO
BREVET	YES	YES	YES
NOM	YES	YES	YES
NBHVOL	YES	YES	YES

Références

Livres:

1. C. MAREE et G. LEDANT, SQL-2 : Initiation, Programmation, 2ème édition, Armand Colin, 1994, Paris.
2. P. DELMAL, SQL2 – SQL3 : application à Oracle, 3ème édition, De Boeck Université, 2001, Bruxelles.
3. C.SOUTOU: SQL pour Oracle, 3ème et 7ème édition, Eyrolles, Paris.

Supports de cours

1. D.BAYERS et L.SWINNEN, Laboratoire 5:Les vues et contraintes