INFORMATIQUE Développement d'applications
BLOC 1
UE05 Bases de données

Chapitre 4.2 : Les vues

Vincent Reip

Février 2025

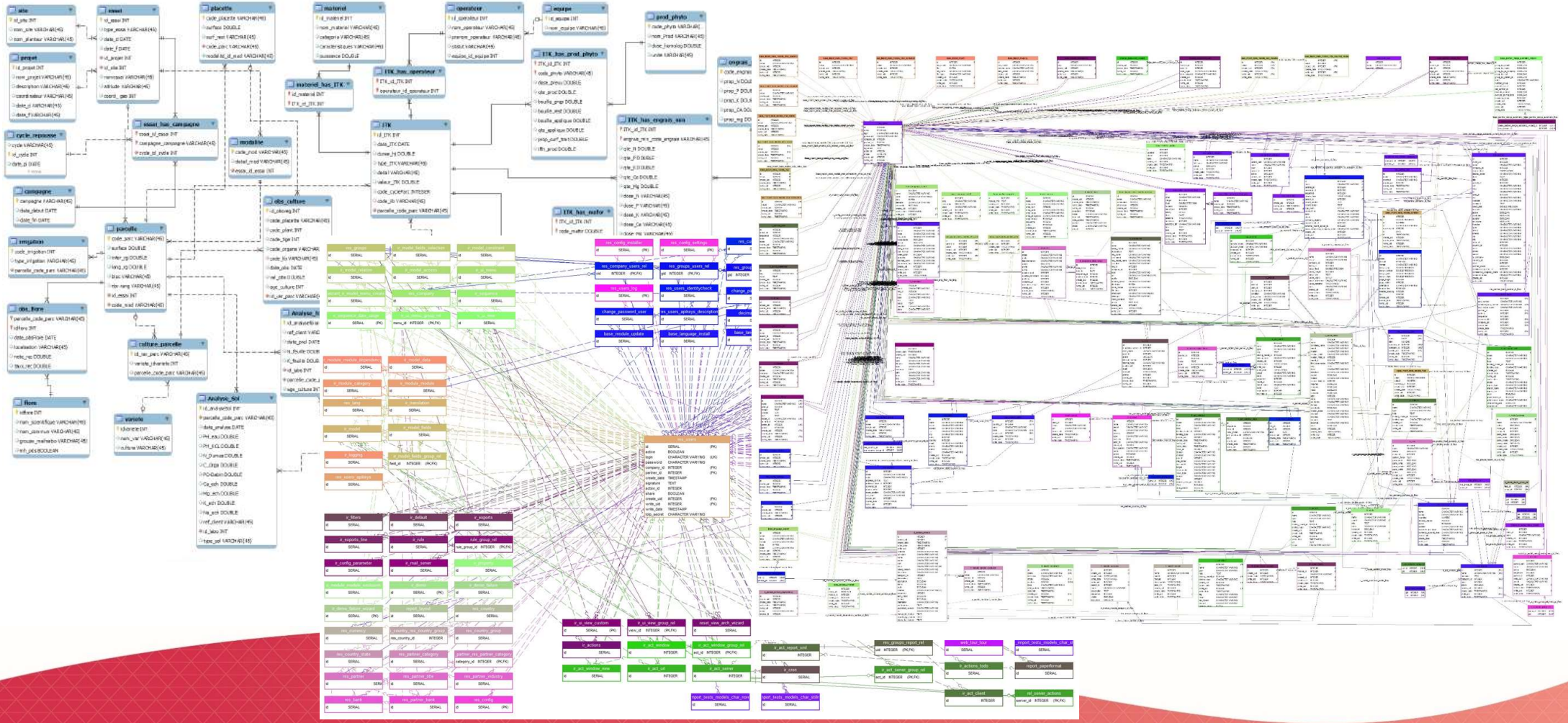
Objectif

- Au terme de ce chapitre, l'étudiant sera capable de :
  - comprendre et expliquer le mécanisme des vues

Les vues : introduction

- La majorité des bases de données respectent les formes normales. Dans un schéma normalisé, les données sont réparties sur plusieurs tables
  - Ce type de schémas se révèle peu commode pour l'utilisateur
    - Difficultés de retrouver toutes les données qui lui sont utiles
    - Besoin de requêtes complexes afin d'obtenir une information intéressante pour l'utilisateur
      - Jointure, projection, …
    - Difficulté de restreindre l'accès à une partie des données présentes dans une table

Les vues : introduction



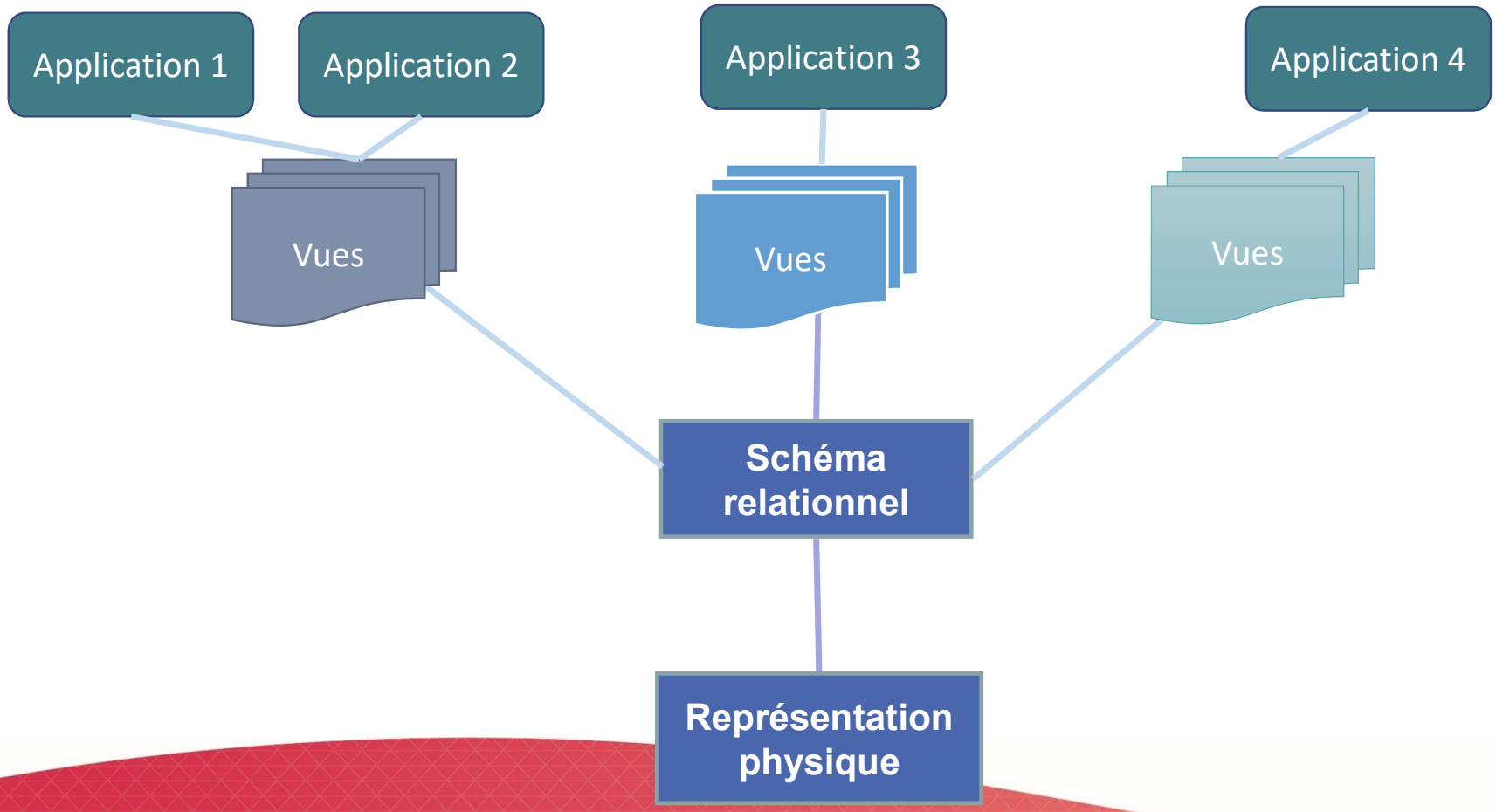
Les vues : introduction

- Dans une base de données, le schéma interne définit le schéma logique (relationnel) de la base de données
- Les vues correspondent aux schémas externes. Ces schémas externes sont propres aux applications ou aux groupes d'utilisateurs d'une base de données
- Une vue désigne une relation virtuelle basée sur une sélection entre des relations ou d'autres vues.

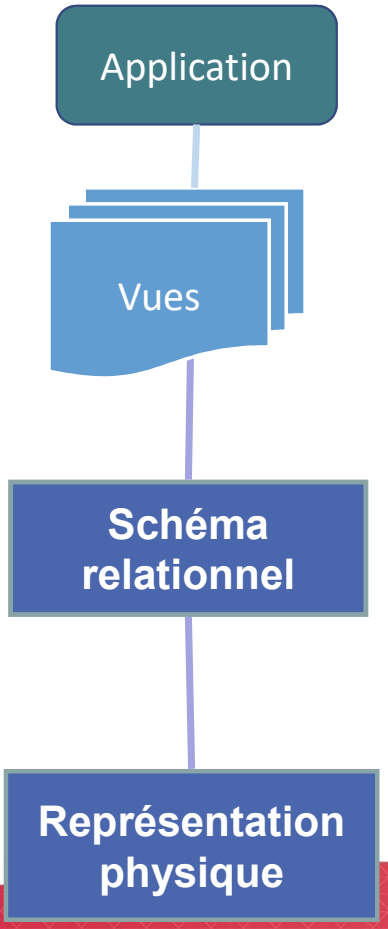
Les vues : introduction

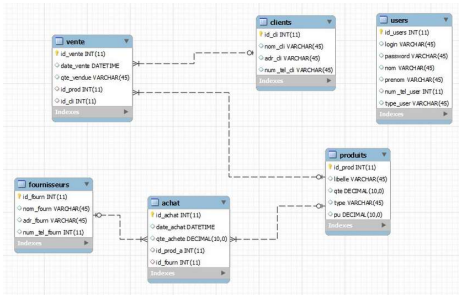
- Les vues répondent au besoin d'adéquation entre les données utiles / autorisées à un utilisateur et le schéma interne de la base de données
- Une vue
  - Présente une version simplifiée de la structure d'une base de données
  - Permet de présenter les données de manières différentes à différentes classes d'utilisateurs
  - Reprend des données utiles / autorisées à une classe d'utilisateurs
  - Fournit un certain degré d'indépendance logique

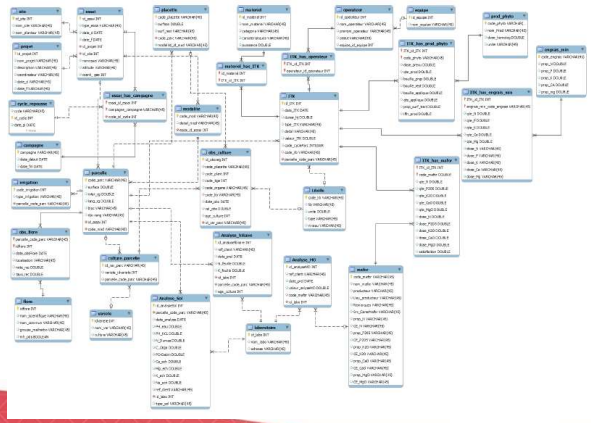
Les vues : introduction



Les vues : introduction







➔ On crée un schéma simplifié grâce aux vues

Schéma relationnel complexe
- l'application n'a pas besoin d'accéder à toutes les tables du schéma (pour réaliser ses UC)
- on veut permettre aux développeurs de l'application de manipuler un schéma simple qui se concentre sur leurs besoins

Les vues : simplification

- Les vues permettent de simplifier la recherche et la présentation des données pour certaines classes d'utilisateurs
  - Les vues permettent d'abstraire la complexité du modèle relationnel : l'utilisateur de la vue n'a pas besoin de connaître l'ensemble des relations et les règles liées. Les requêtes qu'il devra écrire s'en trouveront simplifiées.
  - Certains aspects liés au formatage des données peuvent être effectués « au sein de la vue »
    - Format spécifiques de dates, opérations arithmétiques sur les attributs, manipulation de chaînes de caractères, …

Les vues : simplification

PERSONNE			
id	nom	prenom	id_voiture
1	Potter	Ronald	2
2	Weasley	Hermione	3
3	Granger	Harry	1

FK →

VOITURE				
id	marque	modele	couleur	dateAchat
1	Audi	A4	Rouge	01-02-2012
2	BMW	X1	Bleue	24-05-2015
3	Opel	Corsa	Noire	16-10-2001

```
CREATE VIEW VUE_VOITURE_PERSONNE (voiture, annee_achat, proprietaire) AS
SELECT v.marque || ' ' || v.modele || ' (' || v.couleur || ')', EXTRACT(YEAR FROM v.dateAchat), p.prenom || ' ' || p.nom
FROM PERSONNE p
JOIN VOITURE v ON p.id_voiture = v.id
```

VUE_VOITURE_PERSONNE		
voiture	annee_achat	proprietaire
Audi A4 (Rouge)	2012	Harry Granger
BMW X1 (Bleue)	2015	Ronald Potter
Opel Corsa (Noire)	2001	Hermione Weasley

```
SELECT *
FROM VUE_VOITURE_PERSONNE
```

Les vues : sécurité

- Les vues offrent un certain niveau de sécurité
  - Elles restreignent l'accessibilité d'un groupe d'utilisateurs à un ensemble limité de données. Ces utilisateurs n'ont pas accès au reste de la base de données
  - Chaque groupe d'utilisateurs jouit d'un ensemble de droits sur une vue (ainsi que sur tous les objets de la base de données)

Les vues : sécurité

REVOKE ALL ON EMPLOYE **FROM PUBLIC**;
GRANT SELECT, **INSERT**, **UPDATE**, **DELETE ON** EMPLOYE **TO** USER_HR;

EMPLOYE			
id	**nom**	**prenom**	**salaire**
1	Potter	Ronald	2000
2	Weasley	Hermione	6000
3	Granger	Harry	1500

USER_HR

RW

NO_ACCESS

CREATE VIEW VUE_EMPLOYE_SANS_SALAIRE **AS**
SELECT id, nom, prenom
FROM EMPLOYE

VUE_EMPLOYE_SANS_SALAIRE		
id	**nom**	**prenom**
1	Potter	Ronald
2	Weasley	Hermione
3	Granger	Harry

GRANT SELECT ON VUE_EMPLOYE_SANS_SALAIRE **TO** USER_IT;

USER_IT

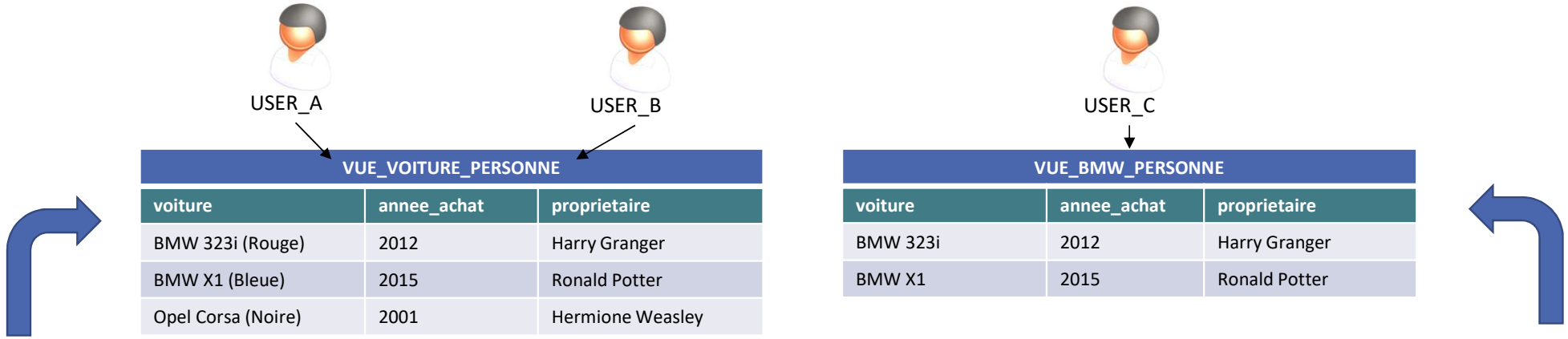
RO

SELECT *
FROM VUE_EMPLOYE_SANS_SALAIRE

Les vues : indépendance logique

- Les vues offrent un certain niveau d'indépendance logique
  - Si pour une raison ou une autre, il est nécessaire de restructurer la BD en modifiant l'emplacement logique des informations, l'utilisation de vues permettra de rendre ces modifications transparentes aux utilisateurs (moyennant une redéfinition de la vue)

Les vues : indépendance logique




```
CREATE VIEW VUE_VOITURE_PERSONNE (voiture, annee_achat, proprietaire) AS
SELECT v.marque || ' ' || v.modele || ' (' || v.couleur || ')',
       EXTRACT(YEAR FROM v.dateAchat), p.prenom || ' ' || p.nom
FROM PERSONNE p
JOIN VOITURE v ON p.id_voiture = v.id
```

```
CREATE VIEW VUE_BMW_PERSONNE (voiture, annee_achat, proprietaire) AS
SELECT v.marque || ' ' || v.modele', EXTRACT(YEAR FROM v.dateAchat),
       p.prenom || ' ' || p.nom
FROM PERSONNE p
JOIN VOITURE v ON p.id_voiture = v.id
WHERE v.marque = 'BMW'
```

PERSONNE

id	nom	prenom	id_voiture
1	Potter	Ronald	2
2	Weasley	Hermione	3
3	Granger	Harry	1

FK →

VOITURE

id	marque	modele	couleur	dateAchat
1	BMW	323i	Rouge	01-02-2012
2	BMW	X1	Bleue	24-05-2015
3	Opel	Corsa	Noire	16-10-2001

Les vues : indépendance logique



USER_A

USER_B

USER_C

VUE_VOITURE_PERSONNE

voiture	annee_achat	proprietaire
BMW_323i (Rouge)	2012	Harry Granger
BMW X1 (Bleue)	2015	Ronald Potter
Opel Corsa (Noire)	2001	Hermione Weasley

VUE_BMW_PERSONNE

voiture	annee_achat	proprietaire
BMW 323i	2012	Harry Granger
BMW X1	2015	Ronald Potter

CREATE OR REPLACE VIEW VUE_VOITURE_PERSONNE (voiture, annee_achat, proprietaire) **AS**

SELECT b.brand || ' ' || c.model || ' (' || c.color || ')',

EXTRACT(YEAR FROM c.buyingDate), p.firstname || ' ' || p.name

FROM PEOPLE p

JOIN CAR c ON p.id_car = c.id

JOIN BRAND b ON c.brand = b.id

CREATE OR REPLACE VIEW VUE_BMW_PERSONNE (voiture, annee_achat, proprietaire) **AS**

SELECT b.brand || ' ' || c.model', EXTRACT(YEAR FROM c.buyingDate),

p.firstname || ' ' || p.name

FROM PEOPLE p

JOIN CAR c ON p.id_car = c.id

JOIN BRAND b ON c.brand = b.id

WHERE b.brand = 'BMW'

PEOPLE

id	name	firstname	id_car
1	Potter	Ronald	2
2	Weasley	Hermione	3
3	Granger	Harry	1

FK

CAR

id	model	color	buyingDate	brand
1	323i	Rouge	01-02-2012	1
2	X1	Bleue	24-05-2015	1
3	Corsa	Noire	16-10-2001	3

FK

BRAND

id	brand
1	BMW
2	Audi
3	Opel

Les vues : en pratique

- Une vue désigne une relation virtuelle
  - Une vue n'a pas d'existence propre.
    - Son existence s'achève au plus tard en même temps que celles des tables sur lesquelles elle se base
  - Une vue associe un identifiant (un nom) à une requête SQL. Cette association est stockée dans le **catalogue** de la base de données.
  - La requête SQL n'est pas exécutée lors de la création de la vue mais bien à chaque fois qu'on y fait référence
    - Ce qui implique quoi ?

Les vues : création

```
CREATE [ OR REPLACE ] VIEW nom_de_la_vue  [(nom-attr1, nom-attr2…)]
AS requête_SQL
[WITH READ ONLY | WITH CHECK OPTION]
```

- La requête SQL
  - Les opérations de jointure, union et autres sont autorisées
  - La liste des colonnes est facultative (si non précisée, la liste des colonnes de la requête SQL sera reprise)
  - Sur certains SGBD (SQLServer), l'utilisation d'une clause ORDER BY n'est pas autorisée
- Les clauses
  - WITH CHECK OPTION[1] restreint l'insertion/modification de tuples à partir de la vue seulement si ceux-ci respectent les contraintes de la requête (c'est-à-dire qu'ils « font partie de la vue »).
  - WITH READ ONLY n'autorise pas l'insertion/modification de tuples à partir de la vue.
  - Si aucune des clauses n'est mentionnée, il pourrait être possible d'insérer, à partir de la vue, des tuples ne faisant pas partie de la vue (1).

(1) concerne les vues modifiables

Les vues : suppression

```
DROP VIEW nom_de_la_vue  [CASCADE| RESTRICT]
```

- L'option CASCADE indique que les objets basés sur la vue à supprimer seront également supprimés
- L'option RESTRICT indique que si cette vue est utilisée dans la définition d'autres objets, la suppression sera refusée
  - Si aucune option n'est indiquée, le SGBD applique par défaut l'option RESTRICT (principe de précaution)

Les vues : modification des données

- Sous certaines conditions, une vue peut être utilisée pour modifier les données sous-jacentes
- La modification des tuples « au travers » d'une vue induit une modification au sein des tables sur lesquelles elle se base.
- La répercussion des modifications dans les tables sources peut être à l'origine de problèmes potentiels
  - Pourquoi ?
    - Il faut toujours respecter l'ensemble des contraintes des tables sous-jacentes.

Les vues : modification des données

```
UPDATE nom_de_la_vue
SET nom-attr1 = expression, nom-attr2 = expression….
[ WHERE condition ]
```

- UPDATE modifie les tuples désignés par la vue (qui sont stockés dans la table sous-jacente).

```
INSERT INTO nom_de_la_vue  [ (attr1, attr2, …) ]
VALUES
( val1, val2, …)
```

- INSERT INTO insère des tuples dans la table sous-jacente.

Les vues : modification des données

```
CREATE TABLE VOITURE (
IDVOITURE INTEGER PRIMARY KEY,
MARQUE VARCHAR2(50) NOT NULL,
ANNEE INTEGER NOT NULL);

INSERT INTO VOITURE VALUES (1, 'AUDI', 2018);
INSERT INTO VOITURE VALUES (2, 'SKODA', 2021);
INSERT INTO VOITURE VALUES (3, 'FORD', 2017);
INSERT INTO VOITURE VALUES (4, 'BMW', 2022);

SELECT *
FROM VOITURE;
```

IDVOITURE	MARQUE	ANNEE
1	AUDI	2018
2	SKODA	2021
3	FORD	2017
4	BMW	2022

```
CREATE OR REPLACE VIEW VW_VOITURE_RECENTE
AS
SELECT *
FROM VOITURE
WHERE ANNEE >= 2020;

SELECT *
FROM VW_VOITURE_RECENTE;
```

IDVOITURE	MARQUE	ANNEE
2	SKODA	2021
4	BMW	2022

Les vues : modification des données

```
INSERT INTO VW_VOITURE_RECENTE VALUES (5, 'RENAULT', 2022);
INSERT INTO VW_VOITURE_RECENTE (IDVOITURE, ANNEE, MARQUE)
VALUES (6, 2021, 'BENTLEY');

SELECT *
FROM VW_VOITURE_RECENTE;
```

IDVOITURE	MARQUE	ANNEE
2	SKODA	2021
4	BMW	2022
5	RENAULT	2022
6	BENTLEY	2021

```
UPDATE VW_VOITURE_RECENTE
SET ANNEE = 2023
WHERE IDVOITURE = 1;
```

0 lignes mis à jour.

```
UPDATE VW_VOITURE_RECENTE
SET ANNEE = 2018
WHERE IDVOITURE = 6;

SELECT *
FROM VW_VOITURE_RECENTE;
```

IDVOITURE	MARQUE	ANNEE
2	SKODA	2021
4	BMW	2022
5	RENAULT	2022

```
SELECT *
FROM VOITURE;
```

IDVOITURE	MARQUE	ANNEE
1	AUDI	2018
2	SKODA	2021
3	FORD	2017
4	BMW	2022
5	RENAULT	2022
6	BENTLEY	2018

Les vues : modification de données

- Pour qu'une vue soit « modifiable », il faut que sa requête de sélection (norme SQL92) …
  - Soit une clause SELECT … FROM … WHERE basique
    - *Pas de jointure*
    - Pas d'union, d'intersection…
  - Ne mentionne pas de clause DISTINCT
  - N'utilise pas d'expressions dans la liste du SELECT
  - Ne fasse intervenir qu'une seule table au sein du FROM
  - Ne contienne pas de fonctions de calcul ni de GROUP BY
  - *Si la clause contient une requête imbriquée, elle ne peut pas référencer la même table que la requête externe*.

Les vues : modification de données

- ORACLE 11g (notre version utilisée en labo) est toutefois moins contraignant que la norme SLQ-92
  - Les jointures sont autorisées sous certaines conditions (Key-Preserved Table)
  - Les sous-requêtes portant sur une table invoqué dans la requête principale sont autorisées

- Une Key-Preserved Table est une table dont la clé primaire (ou une clé unique) est toujours conservée dans le résultat d'une vue contenant un JOIN.
  - Si une table participant à une vue possède une clé primaire (ou unique) et que chaque ligne de cette table apparaît au maximum une fois dans la vue, alors c'est une Key-Preserved Table.
- Si une table est key-preserved, alors Oracle autorise la mise à jour directe de ses colonnes via la vue

En savoir plus : A Look at Oracle Updatable View

Vue modifiable ?


```
CREATE VIEW VW_LIVRE_JULESVERNES
AS
SELECT L.ID_LIVRE, L.TITRE, L.IDAUTEUR
FROM LIVRE L
WHERE L.IDAUTEUR = 1
```

Vue modifiable car elle expose les attributs obligatoires de LIVRE

Vue modifiable ?


```
CREATE VIEW VW_LIVRE_JULESVERNES
AS
SELECT L.ID_LIVRE, L.TITRE
FROM LIVRE L
WHERE L.IDAUTEUR = 1
```

Vue non-modifiable car elle n'expose pas les attributs obligatoires de LIVRE

Vue modifiable ?


```
CREATE VIEW VW_LIVRE_JULESVERNES
AS
SELECT L.ID_LIVRE, L.TITRE, L.IDAUTEUR
FROM LIVRE L
WHERE L.IDAUTEUR = 1
WITH READ-ONLY
```

Vue non-modifiable car on a spécifiquement précisé ‘READ-ONLY’

Vue modifiable ?


```
CREATE VIEW VW_LIVRE_JULESVERNES
AS
SELECT L.ID_LIVRE, L.TITRE, L.IDAUTEUR
FROM LIVRE L
WHERE L.IDAUTEUR = 1
WITH CHECK OPTION
```

Vue modifiable mais on ne pourra insérer que des livres écrits par l'auteur 1 (Jules Vernes)

Vue modifiable ?


```
CREATE VIEW VW_LIVRE_JULESVERNES
AS
SELECT L.TITRE, A.NOM, A.PRENOM
FROM LIVRE L
JOIN AUTEUR A ON L.IDAUTEUR = A.IDAUTEUR
WHERE A.NOM = 'Vernes' AND A.PRENOM = 'Jules'
```

Vue non-modifiable car elle n'expose pas les attributs obligatoires (ni de AUTEUR, ni de LIVRE)

```
SELECT *
FROM VW_LIVRE_JULESVERNES
```

TITRE	NOM	PRENOM
Le tour du monde en 80 jours	Vernes	Jules
Michel Strogoff	Vernes	Jules

Vue modifiable ?


```
CREATE VIEW VW_LIVRE_JULESVERNES
AS
SELECT L.TITRE, A.IDAUTEUR, A.NOM, A.PRENOM
FROM LIVRE L
JOIN AUTEUR A ON L.IDAUTEUR = A.IDAUTEUR
WHERE A.NOM = 'Vernes' AND A.PRENOM = 'Jules'
```

Vue non-modifiable : elle expose les attributs obligatoires de AUTEUR mais la table AUTEUR n'est pas 'key-preserved' (la valeur d'id d'un auteur ayant écrit plusieurs livres apparaîtra plusieurs fois dans la vue)

```
SELECT *
FROM VW_LIVRE_JULESVERNES
```

TITRE	IDAUTEUR	NOM	PRENOM
Le tour du monde en 80 jours	1	Vernes	Jules
Michel Strogoff	1	Vernes	Jules

Vue modifiable ?


```
CREATE VIEW VW_LIVRE_JULESVERNES
AS
SELECT L.IDLIVRE,L.TITRE, L.IDAUTEUR, A.NOM, A.PRENOM
FROM LIVRE L
JOIN AUTEUR A ON L.IDAUTEUR = A.IDAUTEUR
WHERE A.NOM = 'Vernes' AND A.PRENOM = 'Jules'
```

Vue modifiable : des modifications de tuples de la table LIVRE sont possibles elle expose les attributs obligatoires de LIVRE et que la table LIVRE est 'key-preserved' (la clé IDLIVRE est préservée dans la vue)

```
INSERT INTO VW_LIVRE_JULESVERNES (L.IDLIVRE, L.TITRE, L.IDAUTEUR)
VALUES (6, 'Michel Strogoff',1);
```

Les vues : WITH CHECK OPTION

```
CREATE TABLE VOITURE (
IDVOITURE INTEGER PRIMARY KEY,
MARQUE VARCHAR2(50) NOT NULL,
ANNEE INTEGER NOT NULL);

INSERT INTO VOITURE VALUES (1, 'AUDI', 2018);
INSERT INTO VOITURE VALUES (2, 'SKODA', 2021);
INSERT INTO VOITURE VALUES (3, 'FORD', 2017);
INSERT INTO VOITURE VALUES (4, 'BMW', 2022);

SELECT *
FROM VOITURE;
```

IDVOITURE	MARQUE	ANNEE
1	AUDI	2018
2	SKODA	2021
3	FORD	2017
4	BMW	2022

```
CREATE OR REPLACE VIEW VW_VOITURE_RECENTE
AS
SELECT *
FROM VOITURE
WHERE ANNEE >= 2020
WITH CHECK OPTION;

SELECT *
FROM VW_VOITURE_RECENTE;
```

IDVOITURE	MARQUE	ANNEE
2	SKODA	2021
4	BMW	2022

Les vues : WITH CHECK OPTION

INSERT INTO VW_VOITURE_RECENTE VALUES (5, 'RENAULT', 2022);

INSERT INTO VW_VOITURE_RECENTE (IDVOITURE, ANNEE, MARQUE) VALUES (6, 2018, 'BENTLEY');

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

SELECT *

FROM VW_VOITURE_RECENTE;

IDVOITURE	MARQUE	ANNEE
2	SKODA	2021
4	BMW	2022
5	RENAULT	2022

UPDATE VW_VOITURE_RECENTE

SET ANNEE = 2023

WHERE IDVOITURE = 1;

0 lignes mis à jour.

UPDATE VW_VOITURE_RECENTE

SET ANNEE = 2023

WHERE IDVOITURE = 2;

UPDATE VW_VOITURE_RECENTE

SET ANNEE = 2018

WHERE IDVOITURE = 2;

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

SELECT *

FROM VW_VOITURE_RECENTE;

IDVOITURE	MARQUE	ANNEE
2	SKODA	2023
4	BMW	2022
5	RENAULT	2022

Les vues : WITH READ ONLY

```
CREATE TABLE VOITURE (
IDVOITURE INTEGER PRIMARY KEY,
MARQUE VARCHAR2(50) NOT NULL,
ANNEE INTEGER NOT NULL);

INSERT INTO VOITURE VALUES (1, 'AUDI', 2018);
INSERT INTO VOITURE VALUES (2, 'SKODA', 2021);
INSERT INTO VOITURE VALUES (3, 'FORD', 2017);
INSERT INTO VOITURE VALUES (4, 'BMW', 2022);

SELECT *
FROM VOITURE;
```

IDVOITURE	MARQUE	ANNEE
1	AUDI	2018
2	SKODA	2021
3	FORD	2017
4	BMW	2022

```
CREATE OR REPLACE VIEW VW_VOITURE_RECENTE
AS
SELECT *
FROM VOITURE
WHERE ANNEE >= 2020
WITH READ ONLY;

SELECT *
FROM VW_VOITURE_RECENTE;
```

IDVOITURE	MARQUE	ANNEE
2	SKODA	2021
4	BMW	2022

Les vues : WITH READ ONLY

```
INSERT INTO VW_VOITURE_RECENTE VALUES (5, 'RENAULT', 2022);
```

ORA-42399: cannot perform a DML operation on a read-only view

```
SELECT *
FROM VW_VOITURE_RECENTE;
```

IDVOITURE	MARQUE	ANNEE
2	SKODA	2021
4	BMW	2022

```
UPDATE VW_VOITURE_RECENTE
SET ANNEE = 2023
WHERE IDVOITURE = 2;
```

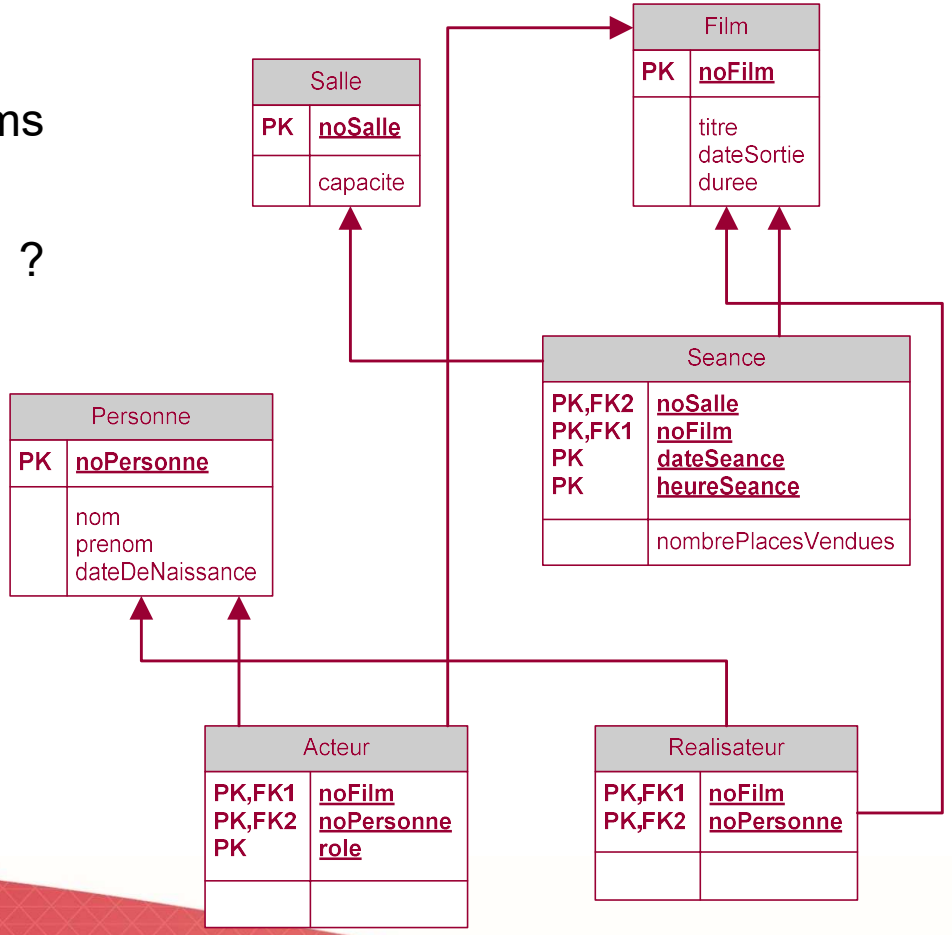
ORA-42399: cannot perform a DML operation on a read-only view

```
SELECT *
FROM VW_VOITURE_RECENTE;
```

IDVOITURE	MARQUE	ANNEE
2	SKODA	2023
4	BMW	2022

Les vues : exercices

- Exercices :
  - On désire une vue qui reprend les films diffusés en salle 1.
  - Cette vue est-elle modifiable ? Pourquoi ?



Les vues : exercices

```
CREATE VIEW Film_salle_1
AS SELECT f.noFilm, f.titre, f.dateSortie, f.duree
    FROM Film f
    JOIN Seance s ON f.noFilm = s.noFilm
     WHERE noSalle = 1
```

La vue n'est pas modifiable puisqu'elle utilise une jointure

Les vues : exercices

1) Créez une vue qui reprend les films dont la date de sortie est située dans les années 80’

2) Créez une vue qui reprend le nom et le prénom des acteurs ainsi que le jour et le mois de leur naissance (pas l’année). En utilisant cette vue, affichez tous les acteurs dont le nom commence par ‘P’.

3) Créer une vue modifiable qui reprend les réalisateurs nés dans les années 70’. En utilisant cette vue, insérez la personne suivante : Bouli Laners né le 20 mai 1965.