

# Base de données

---

## Séance 7

### Requêtes avec SELECT

# Plan

---

- Sous-requêtes
- Union
- Intersection
- Différence
- Jointure :
  1. Jointure avec pivot
  2. Jointure interne
  3. Jointure naturelle

# Sous-requêtes

---

```
select nom
from pokemon
where espece = (select espece
                from pokemon
                where lower(nom) like '%pikachu%');
```

---

# Sous-requêtes SQL

---

- Une sous-requête permet de faire une requête sur la base du résultat d'une autre requête.
- Le résultat peut être utilisé à la place d'une constante dans un calcul ou un prédicat
- Le résultat de la sous-requête peut renvoyer :
  - une ligne contenant :
    - une valeur
    - un n-uplet
  - une liste de lignes
  - aucun résultat

# Sous-requêtes renvoyant une valeur

---

- Sélection par comparaison avec une constante :

`WHERE poste = 'Manager'`

- Est équivalent à :

`WHERE poste = (SELECT poste from ...)`

- Accepte tous les opérateurs tels que `<`, `>`, `<=`, `>=`, `<>`

# Sous-requêtes renvoyant une valeur

---

```
SELECT nom
FROM employe
WHERE poste = (SELECT poste
                FROM employe
                WHERE nom = 'Martin') ;
```

Renvoie tous les employés qui ont le même poste que Martin

*Note : On considère que nom est unique*

# Sous-requêtes sur colonnes multiples

---

```
SELECT nom
FROM employe
WHERE (poste, salaire) =
      ( SELECT poste, salaire
        FROM employe
        WHERE nom = 'Martin' ) ;
```

Renvoie tous les employés qui ont le même poste et le même salaire que Martin

*Note : On considère que nom est unique*

# Sous-requêtes renvoyant une liste

---

- ❖ ...**WHERE** poste **IN** (**SELECT** poste **FROM** ..);
  - La valeur doit être trouvée dans le résultat de la sous requête
  - On peut utiliser **NOT IN**
- ❖ ...**WHERE** numero **>** **ALL** (**SELECT** numero **FROM** ..);
  - La valeur testée doit être supérieure à toutes les valeurs ramenées par la sous requête
- ❖ ...**WHERE** poste **>** **ANY** (**SELECT** poste **FROM** ..);
  - La valeur testée doit être supérieure à au moins une valeur obtenue par la sous requête

# Sous-requêtes renvoyant une liste

---

- opérateur **IN, NOT IN**
- opérateur simple **=, <>, <, >, <=, >=** suivi de **ALL** ou **ANY**.
  - **= ANY** est équivalent à **IN**
  - **!= ALL** est équivalent à **NOT IN**

# Sous-requêtes renvoyant une liste

---

## □ Exemple

```
SELECT nom
FROM client
WHERE num_client
IN (SELECT num_client
    FROM commande
    WHERE date_commande = '05-JUN-98');
```

# Union en SQL

---

- Exemple : nom et prénom des étudiants et professeurs de l'EISTI

```
SELECT nom, prenom FROM etudiant
```

```
UNION
```

```
SELECT nom, prenom FROM professeur
```

---

# Intersection en SQL

---

- Exemple :
  - nom et prénom des professeurs étudiants :

```
SELECT nom, prenom FROM etudiant
```

```
INTERSECT
```

```
SELECT nom, prenom FROM professeur;
```

---

# Différence en SQL

---

## □ Exemple :

- nom et prénom des étudiants qui ne sont pas professeurs :

```
SELECT nom, prenom FROM etudiant
```

**MINUS**

```
SELECT nom, prenom FROM professeur;
```

---

# Jointure avec pivot

---

- La jointure de tables se réalise en effectuant une restriction sur le produit cartésien à l'aide d'un **pivot**.

```
SELECT * FROM A, B
```

```
WHERE A.c1 = B.c2;
```

- Le pivot introduit une contrainte qui réduit les croisements possibles.

# Croiser plus de deux tables

---

- Pour effectuer des **jointures multiples** sur plus de deux tables, on étend le pivot :

```
SELECT * FROM T1, T2, T3  
WHERE <PIVOT1>  
AND <PIVOT2>;
```

## Jointure sur un identifiant : ambiguïtés

---

- Attention aux ambiguïtés lorsqu'un attribut du pivot est présent dans les 2 relations

```
SELECT * FROM employe, manager
WHERE id_manager = id;
```

id ?  
employe  
OU manager

- Préfixer ou renommer :

```
SELECT * FROM employe, manager m
WHERE id_manager = m.id;
```

# Jointure interne

---

```
SELECT * FROM T1 [INNER] JOIN T2  
ON T1.Ai op T2.Bi;
```

- Équi-jointure : =
  - $\theta$ -jointure : <,  $\leq$ , >,  $\geq$ , <>
-

# Jointure interne ou jointure avec pivot ?

tab1		tab2	
col11	col12	col21	col22
a	x	a	1
b	y	b	2
c	z	d	3

```
SELECT *  
FROM tab1 t1,tab2 t2  
WHERE t1.col11 = t2.col21;
```

```
SELECT * FROM tab1  
INNER JOIN tab2  
ON tab1.col11 = tab2.col21
```

a	x	a	1
b	y	b	2

# Jointure naturelle

---

- La jointure se fait sur les attributs de même nom (avec les mêmes types)
- Les attributs de la table résultat sont l'union des attributs de deux tables

```
SELECT * FROM T1 NATURAL JOIN T2
```

---

# Jointure naturelle vs équi-jointure

tab1

tab2

col1 col12 col1 col22

a x a 1

b y b 2

c z d 3

```
SELECT *  
FROM tab1  
NATURAL JOIN tab2;
```



a x 1

b y 2

```
SELECT * FROM tab1 t1  
JOIN tab2 t2  
ON t1.col1 = t2.col1;
```



a x a 1

b y b 2

# Exemple

---

Film(num film, titre, genre, annee)

Cinema(num cine, nom, adresse)

Projection(#num cine, #num film, pdate)

---

num_cine	num_film	pdate
02	05	01/05/2002
02	05	02/05/2002
02	05	03/05/2002
02	04	02/12/1996
01	01	07/05/1996
02	07	09/05/1985
01	04	02/08/1996
04	03	08/04/1994
03	06	02/12/1990
02	02	25/09/1990
03	03	05/11/1994
04	03	06/11/1994
01	06	05/07/1980
02	04	02/09/1996
04	06	01/08/2002
03	06	09/11/1960
01	02	12/03/1988

num_cine	nom	adresse
02	Le Fontenelle	78160 Marly-le-Roi
01	Le Renoir	13100 Aix-en-Provence
03	Gaumont Wilson	31000 Toulouse
04	Espace Ciné	93800 Epinay-sur-Seine

num_film	titre	genre	année
05	Dogville	Drame	2002
04	Breaking the waves	Drame	1996
03	Pulp Fiction	Policier	1994
02	Faux-Semblants	Epouvante	1988
01	Crash	Drame	1996
06	Alamo	Western	1960
07	Dangereusement vôtre	Espionnage	1985

# Exemple

---

Film(num film, titre, genre, annee)

Cinema(num cine, nom, adresse)

Projection(#num cine, #num film, pdate)

- *Quels sont les cinémas qui ont projeté le film Dogville ?*
    1. Jointure avec pivot
    2. Jointure interne
    3. Jointure naturelle ?
-

# Réponse

---

Film(num film, titre, genre, annee)

Projection(#num cine, #num film, pdate)

Cinema(num cine, nom, adresse)

---

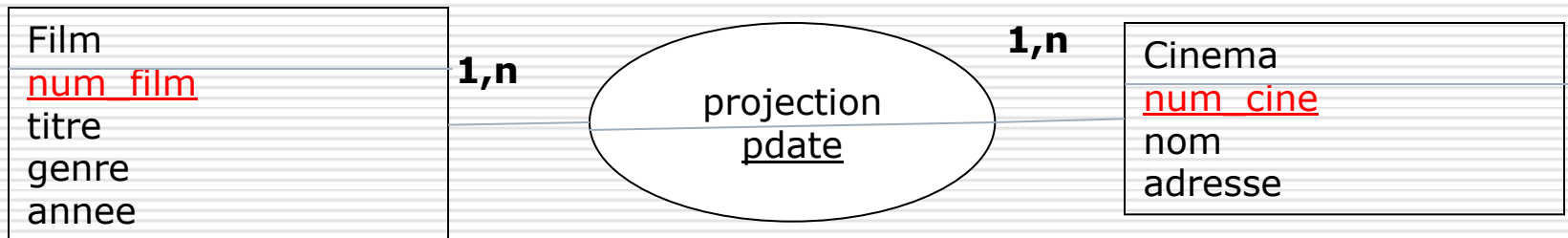
# Réponse

---

Film(num\_film, titre, genre, annee)

Projection(#num\_cine, #num\_film, pdate)

Cinema(num\_cine, nom, adresse)



# 1. Jointure avec pivot

---

```
SELECT nom, adresse
FROM cinema c, film f, projection p
WHERE c.num_cine = p.num_cine
AND f.num_film = p.num_film
AND titre = 'Dogville';
```

---

## 2. Jointure interne

---

```
SELECT nom, adresse  
FROM cinema c  
JOIN projection p ON c.num_cine = p.num_cine  
JOIN film f ON f.num_film = p.num_film  
WHERE titre = 'Dogville';
```

---

### 3. Jointure naturelle

---

```
SELECT nom, adresse  
FROM cinema NATURAL JOIN projection  
NATURAL JOIN film  
WHERE titre = 'Dogville';
```

---