

# MODULE EAO<sub>4</sub>

Introduction aux bases  
de données



# Sommaire

---

---

- Système de gestion de base de données (SGBD)
- Conception
  - Modèle conceptuel de données
  - Modèle relationnel de données
- Exploitation
  - Les langages de requête (SQL)

# Système de gestion de bases de données

---

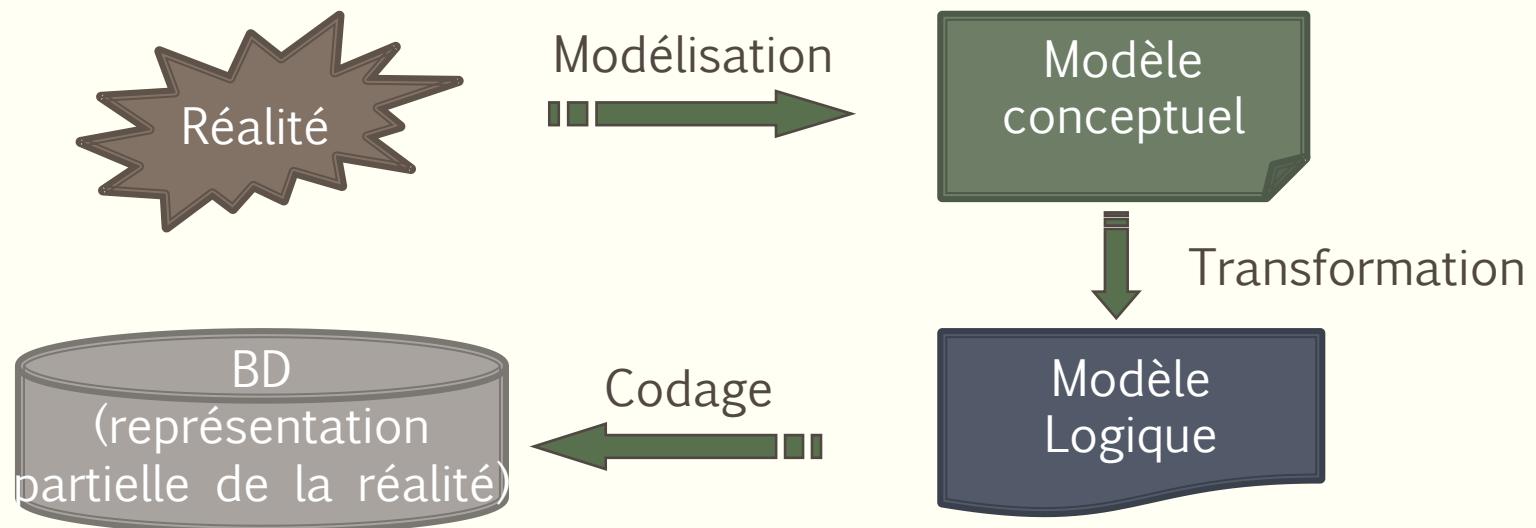
- Une base de données
  - Ensemble d'informations structurées et mémorisées sur un support permanent. (ex. fichier sur disque dur)
- Un Système de Gestion de Bases de Données (SGDB)
  - un logiciel de haut niveau qui permet de manipuler les informations stockées dans une base de données.
  - Permet notamment
    - Une gestion de données relativement structurées
    - Une meilleure interprétation des données
    - L'automatisation des traitements
    - Un stockage cohérent (et optimisé) de l'information

# Méthode d'analyse MERISE

---

## ■ La méthode MERISE

- MCD : concevoir un Modèle Conceptuel de Données
- MLDR : transposer en Modèle Logique de Données Relationnelles
- MPD : générer le Modèle Physique correspondant





# MODÈLES CONCEPTUEL DE DONNÉES (MCD)

Modèle Entité/Association

# Notion de modèle

---

- Un modèle : pour quoi faire ?
  - Identifier l'information pertinente
  - Identifier les éléments qui relient entre elles les informations
  - Structurer l'information de manière cohérente en fonction des besoins réels
  - Optimiser la gestion et le stockage de l'information
- Objectif final : faciliter l'implantation physique de la base de données

# Modèle Entités/Associations

---

## ■ Entité

- Objet clairement identifiable et pertinent pour l'application
- Peut représenter une notion concrète (l'abonné nommé « Varnier Christophe ») ou une notion abstraite (l'abonnement à EDF du client habitant au 3 grande rue à Besançon)

## ■ Association

- Lien sémantique (porteur d'une signification) entre deux ou plusieurs entités.
- Lien non orienté (les abonnés empruntent des films veut dire également que les films peuvent être empruntés)

# Modèle Entités/Associations

---

---

- Propriétés d'une entité ou Attributs
  - Données élémentaires permettant de décrire une entité
  - Un attribut est désigné par un nom
  - Un attribut prend des valeurs dans un domaine défini
- Propriétés ou attributs d'une association
  - Données élémentaires permettant de décrire l'association
- Quelques règles
  - Une propriété ne peut pas figurer sur deux objets différents (éviter la redondance d'information)
  - Une entité possède au moins une propriété
    - Un identifiant qui permet de repérer un objet de cette entité de façon unique
    - Ex : un numéro de sécurité sociale, une plaque d'immatriculation, etc.
  - Une association peut ne pas avoir de propriété

# Modèle Entités/Associations

---

- Exemple

- Développement d'un logiciel ayant pour but de gérer une vidéothèque

*Les films sont classés par catégorie. Les abonnés indiquent quelles sont les catégories de films qu'ils préfèrent. Un abonné peut emprunter un film à la fois*

- Exemples d'entités

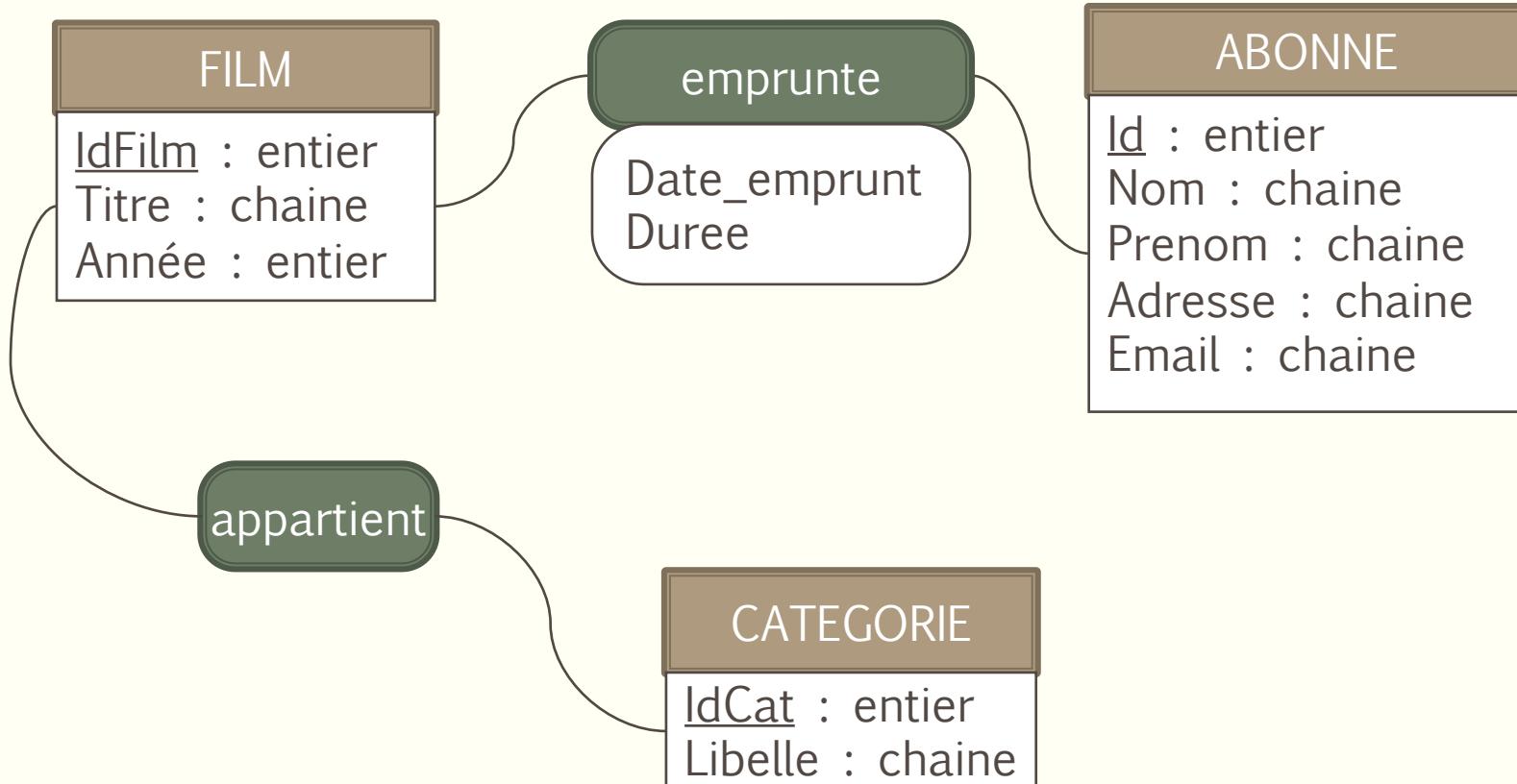
- Un **abonné** est caractérisé par son nom, son prénom, son âge, son adresse et son email.
- Un **film** est caractérisé par son titre, son année de sortie et son réalisateur

- Exemples d'associations

- Association un film « **Appartient** » à une catégorie

# Modèle de données

## Entités/Associations



# Modèle Entités/Associations

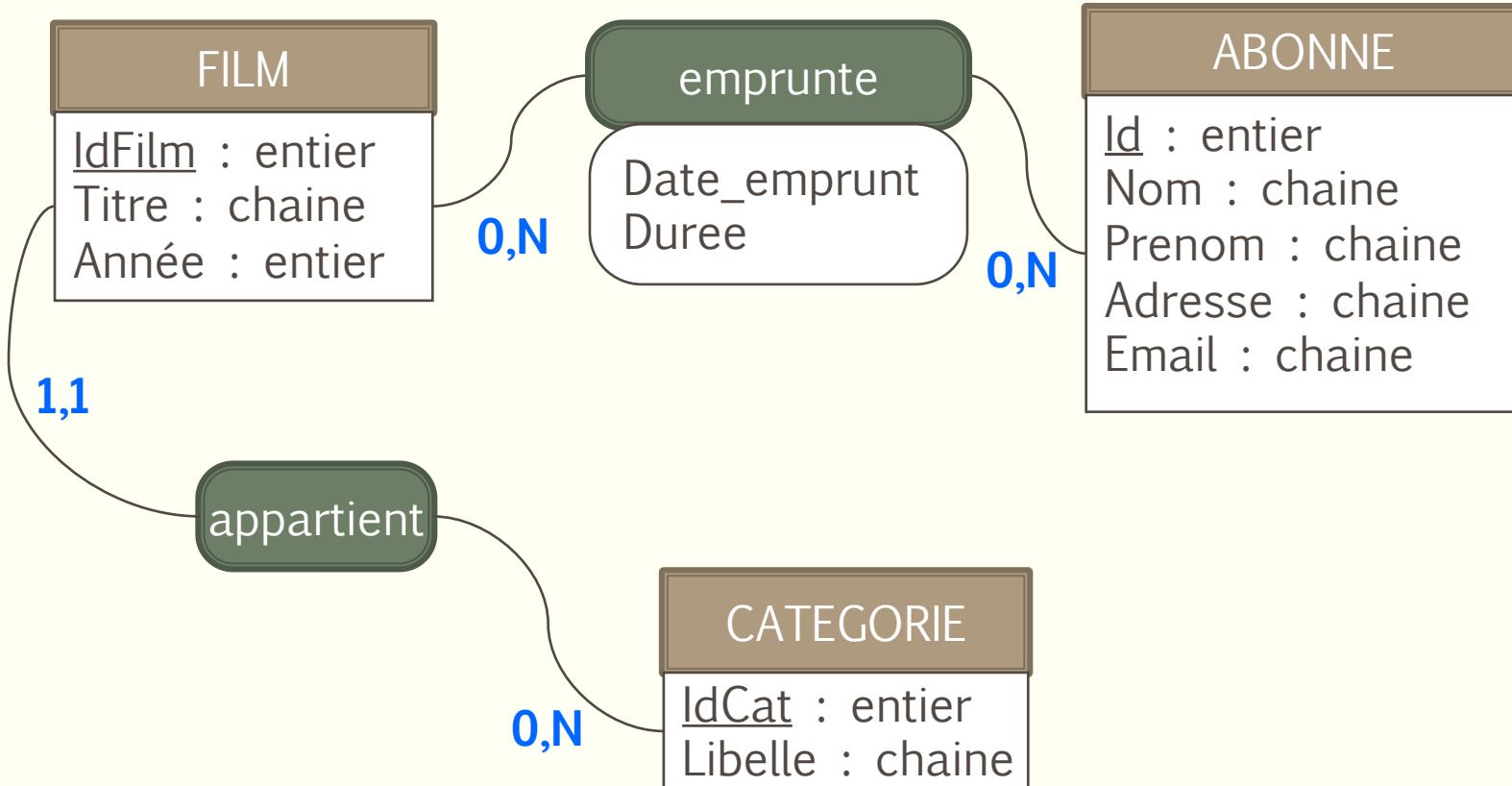
---

---

- Notion de **cardinalité** d'une association
  - La cardinalité établit le nombre possible d'associations entre entités
  - Cardinalité minimum et maximum
    - nombre minimal et maximal de fois qu'une entité peut intervenir dans l'association
    - Valeurs minimum possibles : 0 ou 1
    - Valeurs maximum possibles : 1 ou N ( $\infty$  ou \*)
  - La cardinalité est indiquée dans le modèle sur les arcs des associations

# Modèle de données

## Entités/Associations



Un film appartient à **au moins 1** et **au plus 1** catégorie

Une catégorie contient **au moins 0** film et **au plus N** films



# MODÈLES LOGIQUE DE DONNÉES (MLD)

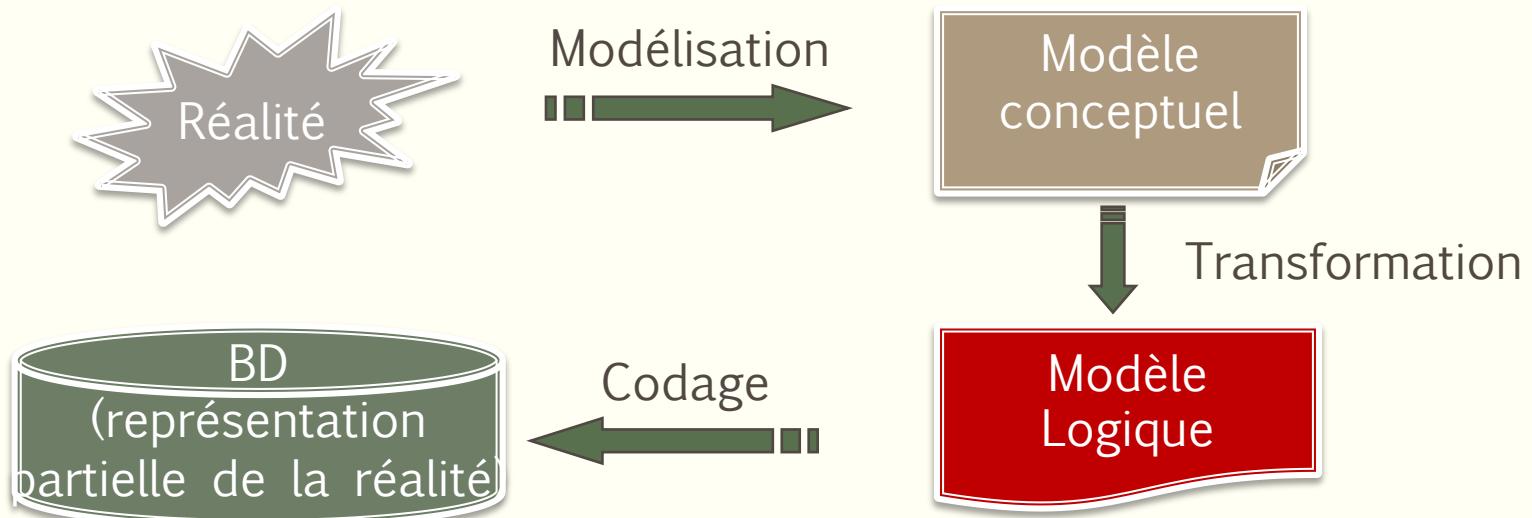
Modèle Relationnel

# Modèle Relationnel

---

## ■ Modèle logique

- Le modèle entité/association est un modèle conceptuel. Il permet de représenter de manière abstraite l'information et les liens existant entre les données
- Le modèle logique consiste à construire la structure des informations dans le but de les « coder » dans une base de donnée



# Modèle Relationnel

---

## ■ Notion de **Relation**

- La seule structure existante est la relation
- Équivalent à une table
  - Les colonnes de la table sont les attributs
  - Les lignes sont les enregistrements de la relation

## ■ Exemple : relation **ABONNE**

- « Colonne » : Nom, Prénom, Adresse, Email
- « Lignes » : Varnier, Christophe, ...

Id	Nom	Prénom	Adresse	Email
124	Varnier	Christophe	Rue A. Daudet	<a href="mailto:cvarnier@ens2m.fr">cvarnier@ens2m.fr</a>
127	Dupont	Roger	Grande rue	<a href="mailto.dupont@bidule.fr">dupont@bidule.fr</a>
131	Jobs	Steve	Rue des pommes	<a href="mailto:jobs@apple.com">jobs@apple.com</a>
...	...	...	...	...

# Modèle Relationnel

---

---

- Définition d'une relation ou table
  - Son nom
  - Ses attributs (nom des « champs ») et leur domaine
- Exemple
  - **ABONNE** ( Id : entier , nom : chaine(20) ,  
prenom : chaine(20) , adresse : chaine(100) ,  
email : chaine(40) )
- Clé(s) d'une relation
  - Clé primaire
    - Identification unique de chaque élément de la relation
    - Souligné dans le formalisme
  - Clés secondaires
    - Clé dite étrangère appartenant à une autre relation
    - précédée de # dans le formalisme

# Passage de MCD vers de MLD

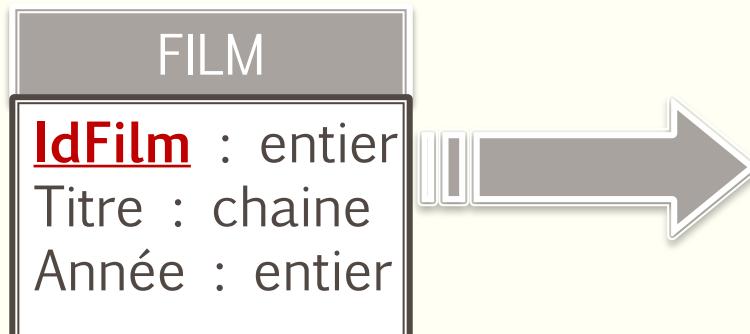
---

- Règles générales

- Une entité devint une relation
- La relation porte le même nom que l'entité
- Les propriétés des entités deviennent les attributs de la relation
- L'attribut identifiant l'entité devient la **clé primaire** de la relation

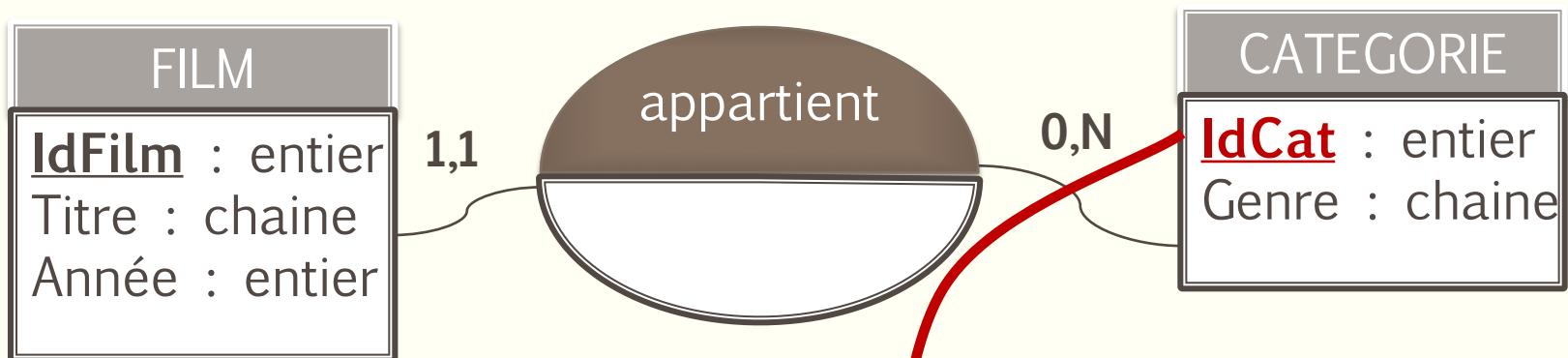
- Remarque :

- perte des associations du modèle entité/Association



# Passage de MCD vers de MLD

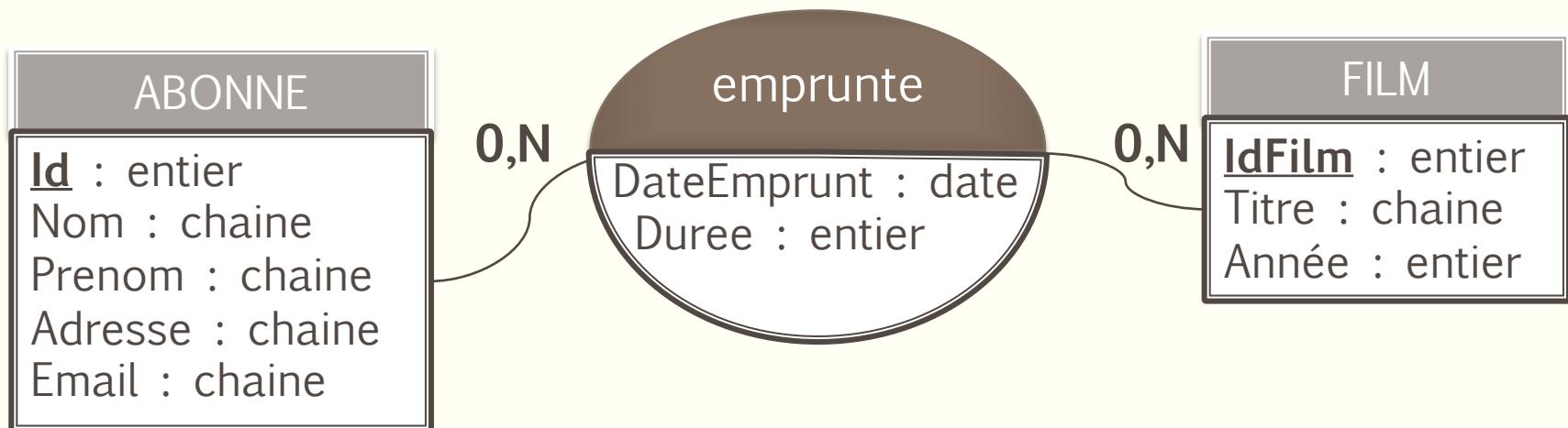
- Une association de cardinalité (1,1) - (x,N)  
avec x = 0 ou 1
  - La clé primaire de la relation ayant la cardinalité (x,N) devient clé secondaire de la relation de cardinalité (1,1)



CATEGORIE ( IdCat : entier , Libelle : chaine )  
FILM ( IdFilm : entier , Titre : chaine ,  
Année : entier , **#IdCat** : entier )

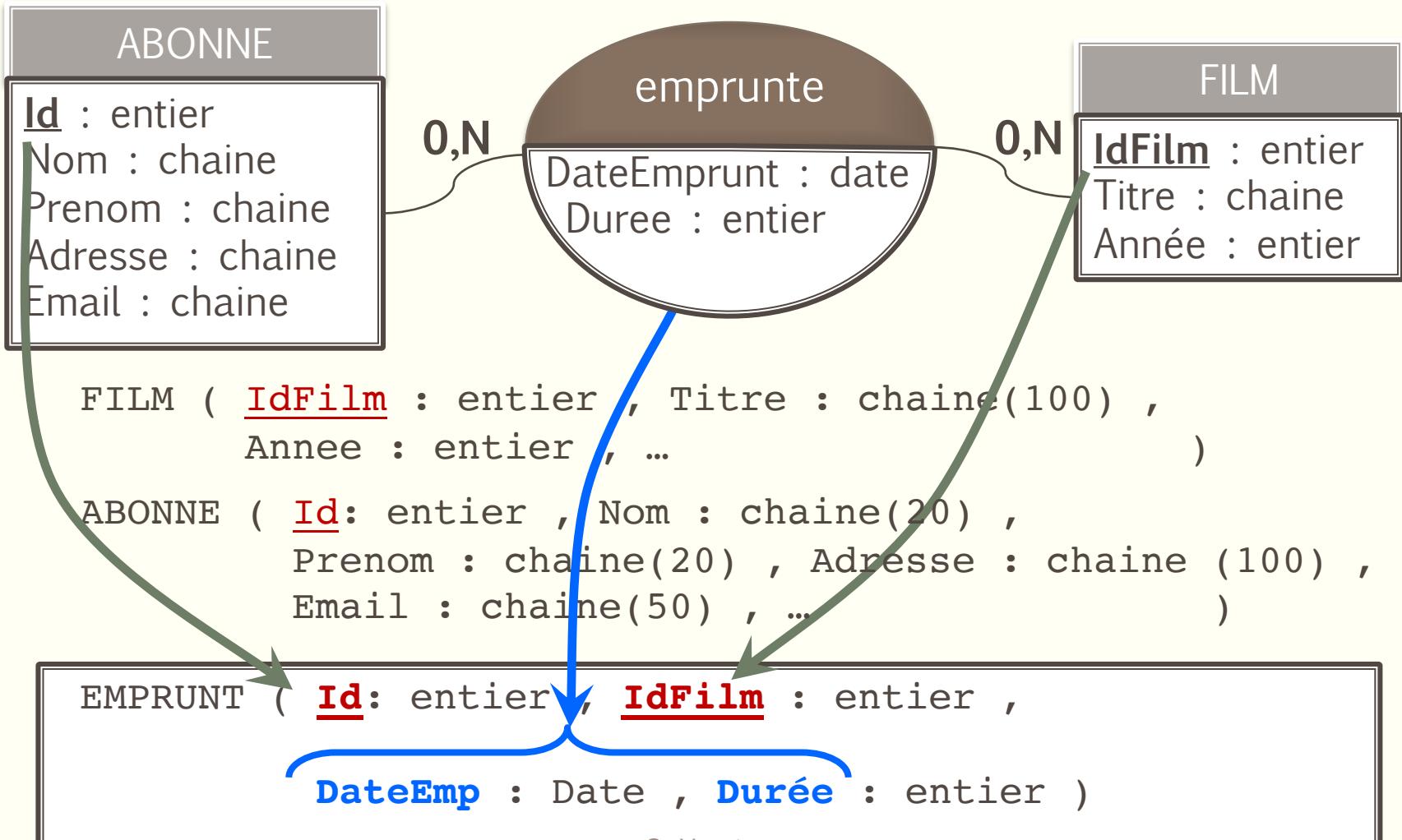
# Passage de MCD vers de MLD

- Une association de cardinalité  $(x,N) - (y,N)$  avec  $x$  et  $y = 0$  ou  $1$ 
  - Une nouvelle relation est créée
  - La clé primaire est constitués de l'ensemble des identifiants des entités associées.
  - Les propriétés de l'association deviennent des attributs de la relation



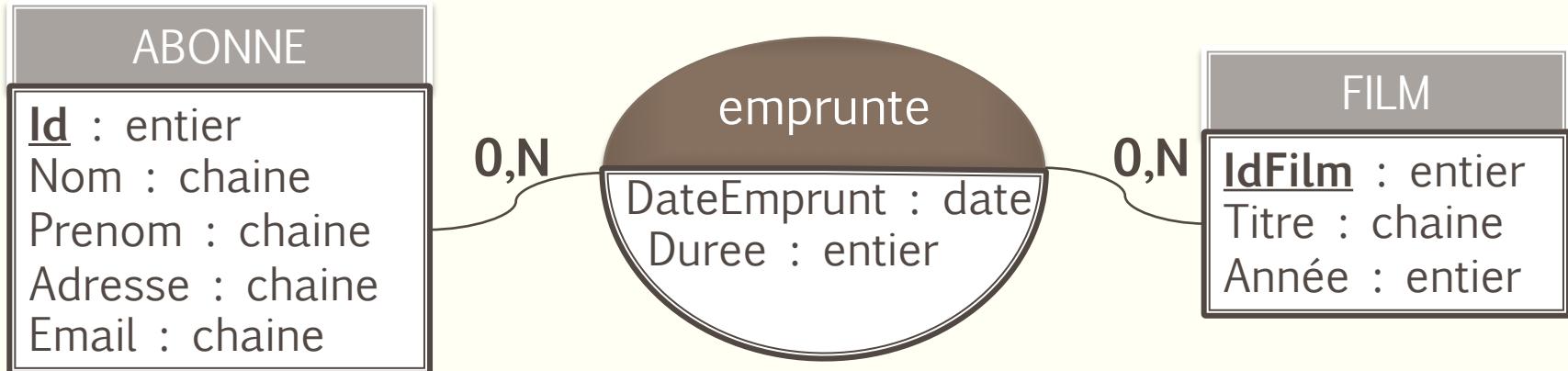
# Passage de MCD vers de MLD

20



# Passage de MCD vers de MLD

21

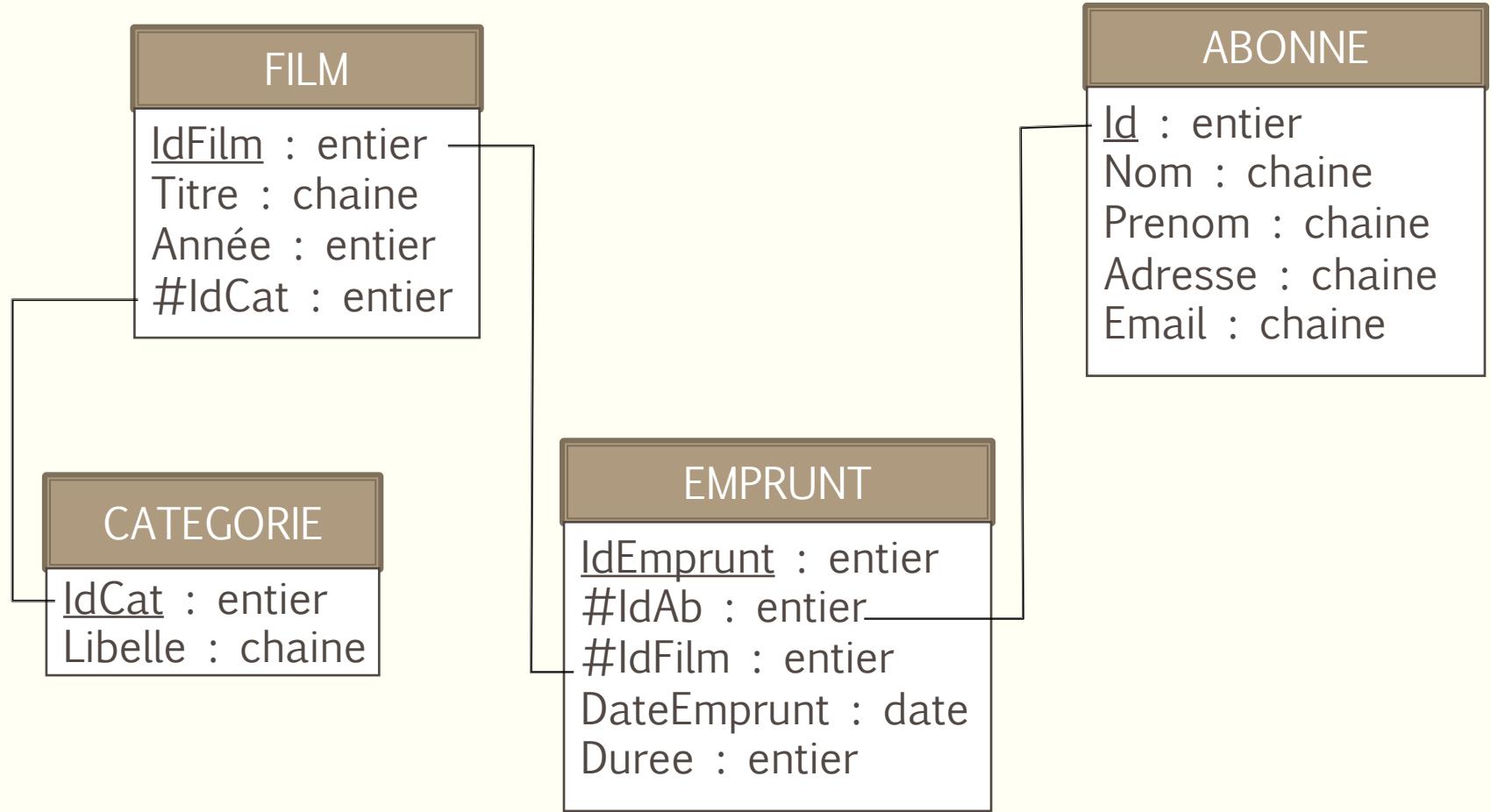


- Si il ne peut pas y avoir 2 enregistrements avec les mêmes clés
  - EMPRUNT ( **Id** : entier , **IdFilm** : entier ,  
DateEmp : Date , Durée : entier )
- Si il peut y avoir plus de 2 enregistrements avec les mêmes clés
  - EMPRUNT ( **Id** : entier , **IdFilm** : entier ,  
**DateEmp** : Date , Durée : entier )  
*OU*
  - EMPRUNT ( **IdEmprunt** : entier ,  
#Id : entier , #IdFilm : entier ,  
DateEmp : Date , Durée : entier )

# Modèle relationnel

---

---





# LANGAGES DE REQUÊTES

SQL

# SQL

---

---

- « *Structured Query Language* »
  - Langage pour la gestion de bases de données relationnelles
  - Conçu par IBM dans les années 70 et normalisé en 1986
- Langage qui permet
  - La définition des données et des relations
  - L'insertion de données
  - La mise à jour
  - La destruction
  - La sélection (interrogation de la base)
- Simple d'utilisation
  - Interface graphique disponible (ex. Access)
  - Peut être associé à des langages de programmation classiques (C, C++, Visual Basic par ex.)
  - Peut être utilisé dans des interfaces web (pages dynamiques avec PHP par ex.)

# Langage SQL

---

- Instructions pour la définition des bases ou des relations (tables)

- **CREATE DATABASE** : création d'une nouvelle base de données
    - Ex : `CREATE DATABASE Videotheque;`

- **CREATE TABLE** : création d'une nouvelle table (ou relation)
    - Syntaxe :

```
CREATE TABLE Nom_de_la_table
  ( nom_attribut1    type_attr1 [propriétés]*,
    nom_attribut2    type_attr2 [propriétés],
    ...
    nom_attributn    type_attrn [propriétés],
  )
```

Propriétés = **PRIMARY KEY**, **DEFAULT**, **NOT NULL** par exemple

- Exemple :

```
CREATE TABLE Film
  ( IdFilm      INT PRIMARY KEY NOT NULL,
    Titre       VARCHAR(100),
    Annee       SMALLINT,
    IdCat       INT
  )
```

\* [...] : options facultatives

# Langage SQL

---

- Instructions pour la destruction des bases ou des relations (tables)
  - **DROP DATABASE** : destruction de toutes les tables d'une base de données
    - Ex : **DROP DATABASE Videotheque ;**
  - **DROP TABLE** : destruction d'une ou plusieurs tables
    - Ex : **DROP TABLE Film ;**

# Langage SQL

---

- Instructions pour la modification des relations (tables)

- **ALTER TABLE** : Modifie les propriétés d'une table

- Syntaxe :

```
ALTER TABLE nom_table instruction
```

- Instructions :

- ADD : ajout d'un attribut
    - DROP : suppression d'un attribut
    - MODIFY : modification des propriétés d'un attribut
    - CHANGE : renommage d'un attribut

- Ex :

- ```
ALTER TABLE Film
      ADD Realisateur VARCHAR(40)
```
    - ```
ALTER TABLE Film
      DROP Realisateur
```
    - ```
ALTER TABLE Film
      MODIFY Annee BIGINT
```
    - ```
ALTER TABLE Film
      CHANGE Titre TitreDuFilm
```

# SQL

---

- Instruction pour la manipulation des données
  - Ajout
    - **INSERT** : ajout d'un ou plusieurs enregistrements dans une table
  - Mise à jour
    - **UPDATE** : modification des lignes d'une table
  - Interrogation (requête)
    - **SELECT** : permet de réaliser les opérations de l'algèbre relationnelle, vu précédemment
    - Instruction la plus utilisée
  - Suppression
    - **DELETE** : suppression de lignes d'une table

# SQL

---

- **INSERT** : ajout d'un ou plusieurs enregistrements dans une table

- Syntaxe :

```
INSERT INTO Nom_de_table [(champ1, champ2, ...) ]
VALUES (val1, val2, ...)
[ , (val3, val4, ...) , ... ]
```

- Exemple :

```
INSERT INTO CATEGORIE
VALUES (12, 'Comédie musicale');
```

```
INSERT INTO CATEGORIE (IdCat, Libelle)
VALUES (13, 'Horreur');
```

```
INSERT INTO CATEGORIE (IdCat, Libelle)
VALUES (13, 'Horreur') , (14, 'peplum');
```

# SQL

---

## ■ **UPDATE** : modification des lignes d'une table

- Syntaxe :

```
UPDATE Nom_de_table
SET nom_de_champ = valeur [, champ = val2 , ...]
WHERE condition
```

- Exemple :

```
UPDATE ABONNE
SET Adresse = '10 grande rue, 25000 Besançon'
WHERE idAbonne = 1 ;
```

```
UPDATE EMPRUNT
SET Duree = Duree + 1
WHERE Date = '2014-12-01' ;
```

# SQL

---

---

- **SELECT** : La projection ( $\pi$ ) :  $\pi_{A_1, A_2, \dots, A_n}(\text{Relation})$

- Syntaxe :

```
SELECT [DISTINCT] A1, A2, ..., An  
FROM Relation
```

- Exemple :

$\pi_{\text{Titre}, \text{Année}}(\text{FILM}) \rightarrow$   
SELECT Titre, Anne FROM FILM ;

$\pi_{\text{Réalisateur}}(\text{FILM}) \rightarrow$   
SELECT DISTINCT Réalisateur FROM FILM ;

$\pi_{\text{tousLesChamps}}(\text{FILM}) \rightarrow$   
SELECT \* FROM FILM ;

# SQL

---

---

- **SELECT** : La sélection ( $\sigma$ ) :  $\sigma_{\text{critère}}(\text{Relation})$

- Syntaxe :

```
SELECT *
FROM Relation
WHERE Critère
```

- Exemple :

$\sigma_{\text{Année}<1990}(\text{FILM}) \rightarrow$

```
SELECT * FROM FILM
WHERE Année<1990 ;
```

$\sigma_{\text{Réalisateur}=\text{"James Cameron}}(\text{FILM}) \rightarrow$

```
SELECT * FROM FILM
WHERE Réalisateur = 'James Cameron' ;
```

# SQL

---

---

## ■ Critères :

- Opérateurs logiques : **AND**, **OR**, **NOT**
  - Condition1 OR Condition2
  - Condition1 AND Condition2
  - Condition1 ET (Condition2 OR NOT Condition3)
- Opérateurs ensemblistes : **IN**
  - Champ IN (val1, val2, ... )
- Intervalle : **BETWEEN**
  - Date\_inscription BETWEEN '2014-01-01' AND '2014-12-31'
- Chaines de caractères : **LIKE**
  - Champ LIKE 'ab%' (commence par 'ab', % est un caractère joker)
  - Champ LIKE '%yz' (termine par 'xy')
  - Champ LIKE '%ab%' (contient 'ab')
- Existence de valeur : **IS NULL** ou **IS NOT NULL**
  - Champ IS NULL

# SQL

---

---

- **SELECT** : Le renommage ( $\rho_{A \rightarrow B}$ ) :  $\rho_{A \rightarrow B}(\text{Relation})$

- Syntaxe :

```
SELECT A as B  
FROM Relation
```

- Exemple :

$\rho_{\text{Titre} \rightarrow \text{TitreDuFilm}}(\text{FILM})$

```
SELECT IdFilm, Titre as TitreDuFilm,  
       Realisateur, Année  
FROM FILM ;
```

# SQL

---

---

- Le produit cartésien ( $\times$ ) :  
**R1 \* R2**
  - Syntaxe :  
SELECT \*  
FROM R1 , R2
- L'union ( $\cup$ ) :  
**R1  $\cup$  R2**
  - Syntaxe :  
SELECT \* FROM R1  
**UNION**  
SELECT \* FROM R2
- L'intersection ( $\cap$ ) :  
**R1  $\cap$  R2**
  - Syntaxe :  
SELECT \* FROM R1  
INTERSECT  
SELECT \* FROM R2
- La différence (-) :  
**R1 - R2**
  - Syntaxe :  
SELECT \* FROM R1  
MINUS  
SELECT \* FROM R2

# SQL

---

---

- **SELECT** : La jointure ( $\bowtie$ )

- **INNER JOIN**

- Syntaxe :

```
SELECT * FROM A
INNER JOIN B ON A.cle = B.cle
```

- Exemple :

FILM	$\bowtie$	CATEGORIE
FILM.IdCat=CATEGORIE.IdCat		

```
SELECT * FROM FILM
INNER JOIN CATEGORIE
ON FILM.IdCat=CATEGORIE.IdCat
```

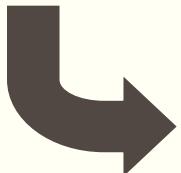
# Quelques exemples

- Chercher les films réalisés avant 1990

```
SELECT Titre, Année, Réalisateur  
FROM FILM  
WHERE Année < 1990
```

FILM : Table

	IdFilm	Titre	Année	Réalisateur	IdCat	NomVid
+	1	Titanic	1997	James Cameron	2	MaVidéo
+	2	Il etait une fois dans l'ouest	1969	Sergo Leone	5	MaVidéo
+	3	Elephant Man	1980	David Lynch	2	MaVidéo
+	4	Pulp Fiction	1994	Quentin Tarantino	7	MaVidéo
+	5	Abyss	1989	James Cameron	3	MaVidéo



Fims avant 1990 : Requête Sélection

	Année	Titre	Réalisateur
	1969	Il etait une fois dans l'ouest	Sergo Leone
	1980	Elephant Man	David Lynch
	1989	Abyss	James Cameron

# Quelques exemples

- Produit cartésien simple

SELECT \* FROM FILM, CATEGORIE

FILM : Table					
	IdFilm	Titre	Année	Réalisateur	IdCat
+	1	Titanic	1997	James Cameron	2
+	2	Il etait une fois dans l'ouest	1969	Sergo Leone	5
+	3	Elephant Man	1980	David Lynch	2
+	4	Pulp Fiction	1994	Quentin Tarantino	7
+	5	Abyss	1989	James Cameron	3

X

CATEGORIE : Table		
	IdCat	Genre
+	1	Animation
+	2	Drame
+	3	Science fiction
+	4	Comédie
+	5	Western
+	6	Horreur
+	7	Policier
+	8	Suspense



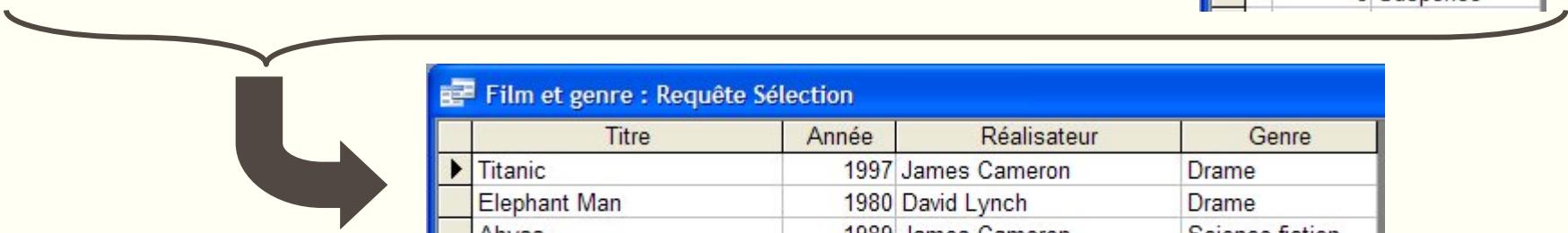
Produit FilmxCategorie : Requête Sélection							
	IdFilm	Titre	Année	Réalisateur	FILM.IdCat	CATEGORIE.IdCat	Genre
▶	1	Titanic	1997	James Cameron	2		1 Animation
	2	Il etait une fois dans l'ouest	1969	Sergo Leone	5		1 Animation
	3	Elephant Man	1980	David Lynch	2		1 Animation
	4	Pulp Fiction	1994	Quentin Tarantino	7		1 Animation
	5	Abyss	1989	James Cameron	3		1 Animation
	1	Titanic	1997	James Cameron	2		2 Drame
	2	Il etait une fois dans l'ouest	1969	Sergo Leone	5		2 Drame
	3	Elephant Man	1980	David Lynch	2		2 Drame
	4	Pulp Fiction	1994	Quentin Tarantino	7		2 Drame
	5	Abyss	1989	James Cameron	3		2 Drame
	1	Titanic	1997	James Cameron	2		3 Science fiction
	2	Il etait une fois dans l'ouest	1969	Sergo Leone	5		3 Science fiction
	3	Elephant Man	1980	David Lynch	2		3 Science fiction
	4	Pulp Fiction	1994	Quentin Tarantino	7		3 Science fiction
	5	Abyss	1989	James Cameron	3		3 Science fiction
	1	Titanic	1997	James Cameron	2		4 Comédie
	2	Il etait une fois dans l'ouest	1969	Sergo Leone	5		4 Comédie

# Quelques exemples

```
■ SELECT Titre, Année, Réalisateur, Genre  
FROM FILM  
INNER JOIN CATEGORIE  
ON FILM.IdCat=CATEGORIE.IdCat
```

FILM : Table					
	IdFilm	Titre	Année	Réalisateur	IdCat
+	1	Titanic	1997	James Cameron	2
+	2	Il etait une fois dans l'ouest	1969	Sergo Leone	5
+	3	Elephant Man	1980	David Lynch	3
+	4	Pulp Fiction	1994	Quentin Tarantino	7
+	5	Abyss	1989	James Cameron	3

CATEGORIE : Table		
	IdCat	Genre
+	1	Animation
+	2	Drame
+	3	Science fiction
+	4	Comédie
+	5	Western
+	6	Horreur
+	7	Policier
+	8	Suspence



Film et genre : Requête Sélection				
	Titre	Année	Réalisateur	Genre
►	Titanic	1997	James Cameron	Drame
	Elephant Man	1980	David Lynch	Drame
	Abyss	1989	James Cameron	Science fiction
	Il etait une fois dans l'ouest	1969	Sergo Leone	Western
	Pulp Fiction	1994	Quentin Tarantino	Policier

# Quelques exemples

---

- Chercher tous les films empruntés

```
SELECT Nom, Titre, DateEmprunt  
FROM FILM  
INNER JOIN EMPRUNT  
ON FILM.IdFilm=EMPRUNT.IdFilm  
INNER JOIN ABONNE  
ON EMPRUNT.Id=ABONNE.Id  
ORDER BY DateEmprunt DESC , Titre ASC;
```

	Nom	Titre	DateEmprunt
▶	Varnier	Elephant Man	14/11/2006
	Dupond	Titanic	14/11/2006
	Dupond	Abyss	13/11/2006
	Poquelin	Abyss	11/11/2006
	Poquelin	Pulp Fiction	11/11/2006
	Varnier	Il etait une fois dans l'ouest	10/11/2006
	Varnier	Titanic	10/11/2006

# Quelques exemples

---

- ```
SELECT Année, Count(Titre) AS NbreFilms  
FROM FILM  
GROUP BY Année;
```

| Film Groupe Année : Requête |       |           |
|-----------------------------|-------|-----------|
|                             | Année | NbreFilms |
| ▶                           | 1969  | 1         |
|                             | 1980  | 1         |
|                             | 1989  | 3         |
|                             | 1994  | 1         |
|                             | 1997  | 1         |
|                             | 2003  | 2         |

- ```
SELECT Année, Count(Titre) AS NbreFilms  
FROM FILM  
GROUP BY Année  
HAVING Count(Titre) >= 2;
```

Film Groupe Année Nbr >= 2		
	Année	NbreFilms
▶	1989	3
	2003	2

# Autres instructions SQL

---

- DELETE
  - Suppression de lignes de tables : **DELETE**
    - Supprimer le genre 'Drame'

```
DELETE FROM CATEGORIE  
WHERE Genre = 'Drame';
```