

# Bases de données

## Cours 3

### Requêtes SQL (suite)

Marie Pelleau

[marie.pelleau@univ-cotedazur.fr](mailto:marie.pelleau@univ-cotedazur.fr)

13 mai 2025

- 1 Notations et exemple fil-rouge
  - Exemple : bibliothèque
  - Notations
- 2 Jointures
- 3 Opérations ensemblistes
- 4 Exercices
- 5 Requêtes de définition de données
- 6 Sous-requêtes

# Exemple fil-rouge : bibliothèque

## Attributs des entités

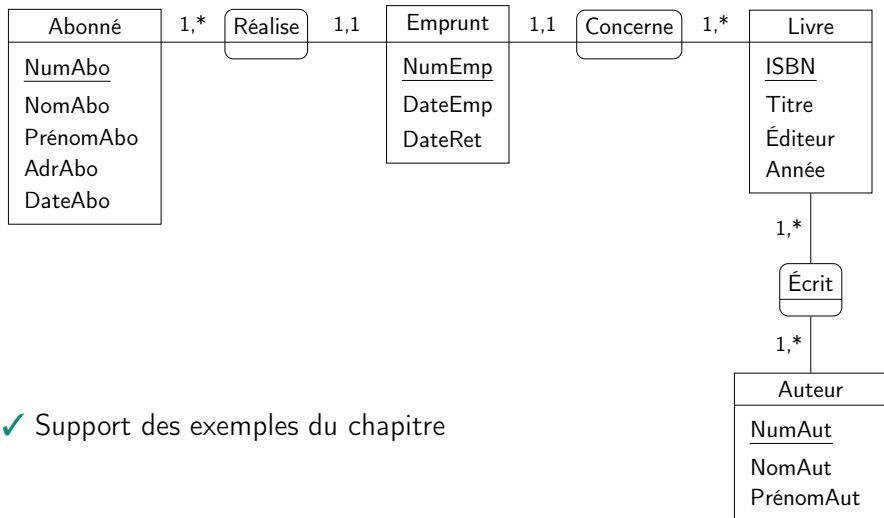
- Chaque **abonné** a un *numéro d'abonné* unique, un *nom*, un *prénom*, une *adresse* et une *date d'abonnement*.
- Les **livres** ont tous un *numéro ISBN*, un *titre*, un *éditeur* et une *année de publication*.
- Les **auteurs** qui écrivent les **livres** sont identifiés par un *numéro d'auteur*, et on stocke leur *nom* et *prénom*.
- Lorsqu'un **abonné** réalise un **emprunt** d'un **livre**, on enregistre le *numéro* et la *date de l'emprunt*.
- Lorsqu'il le restitue, on mémorise la *date de retour*.

# Dictionnaire de données

On le résume dans un tableau.

| Libellé   | Type    | Description                                      |
|-----------|---------|--|
| NumAbo    | entier  | Numéro de l'abonné                               |
| NomAbo    | car(20) | Nom de l'abonné                                  |
| PrénomAbo | car(20) | Prénom de l'abonné                               |
| AdrAbo    | car(80) | Adresse de l'abonné                              |
| DateAbo   | date    | Date de l'abonnement (AAAA-MM-JJ)                |
| NumAut    | entier  | Numéro de l'auteur                               |
| NomAut    | car(20) | Nom de l'auteur                                  |
| PrénomAut | car(20) | Prénom de l'auteur                               |
| ISBN      | car(13) | Code ISBN identifiant un livre                   |
| Titre     | car(80) | Titre du livre                                   |
| Editeur   | car(20) | Nom de l'editeur                                 |
| Année     | entier  | Année de publication                             |
| NumEmp    | entier  | Numéro d'emprunt                                 |
| DateEmp   | date    | Date de l'emprunt d'un livre par un abonné       |
| DateRet   | date    | Date de retour d'un livre emprunté par un abonné |

## Exemple BD Bibliothèque : schéma E-A



✓ Support des exemples du chapitre

## Exemple BD Bibliothèque : schéma relationnel

- Schéma relationnel de la BD

ABONNÉ(NumAbo, NomAbo, PrénomAbo, AdrAbo, DateAbo)

LIVRE(ISBN, Titre, Éditeur, Année)

AUTEUR(NumAut, NomAut, PrénomAut)

ÉCRIT(ISBN, NumAut)

EMPRUNT(NumEmp, NumAbo, ISBN, DateEmp, DateRet)

# Exemple BD Bibliothèque : tables

- Tables (relations).

LIVRE

| ISBN          | Titre            | Éditeur  | Année |
|---------------|------------------|----------|-------|
| 9782212112818 | Bases de Données | Eyrolles | 1989  |
| 9782225805158 | Le Langage C     | Masson   | 1985  |
| 9782207257357 | Fondation        | Denoël   | 2006  |

AUTEUR

| NumAut | NomAut    | PrénomAut |
|--------|-----------|-----------|
| 1      | Gardarin  | Georges   |
| 2      | Kernighan | Brian     |
| 3      | Ritchie   | Dennis    |
| 4      | Asimov    | Isaac     |

ÉCRIT

| ISBN          | NumAut |
|---------------|--------|
| 9782212112818 | 1      |
| 9782225805158 | 2      |
| 9782225805158 | 3      |
| 9782207257357 | 4      |

ABONNE

| NumAbo | NomAbo  | PrénomAbo | DateAbo    |
|--------|---------|-----------|------------|
| 1      | Dupont  | Philippe  | 2008-06-18 |
| 2      | Durand  | Arthur    | 2009-01-02 |
| 3      | Dupont  | Charlie   | 2015-05-03 |
| 4      | Ducros  | Marie     | 2020-07-04 |
| 5      | Vernier | Alain     | 2021-09-15 |

EMPRUNT

| NumEmp | ISBN          | NumAbo | DateEmp    | DateRet    |
|--------|---------------|--------|------------|------------|
| 1      | 9782225805158 | 2      | 2021-09-06 | 2021-09-20 |
| 2      | 9782225805158 | 3      | 2021-09-25 | 2021-10-11 |
| 3      | 9782212112818 | 1      | 2021-10-28 | 2021-11-10 |
| 4      | 9782212112818 | 1      | 2021-11-08 | NULL       |

# Conventions de notations

- Mots-clés de SQL : caractères COURIER majuscules
- Paramètres des requêtes : caractères courier minuscules
- Paramètres optionnels : [option]
- Valeurs multiples possibles : valeur<sub>1</sub> | valeur<sub>2</sub>
- Options multiples : [option<sub>1</sub> | option<sub>2</sub>]



- 1 Notations et exemple fil-rouge
- 2 Jointures
  - Présentation
  - Copies multiples d'une table
- 3 Opérations ensemblistes
- 4 Exercices
- 5 Requêtes de définition de données
- 6 Sous-requêtes

# Requêtes multi-relations : jointures

- Requêtes utilisant plusieurs tables :
  - Clause FROM : permet de spécifier les tables sources de données
  - Clause WHERE : jointure sur les attributs liés

```
SELECT *
FROM Abonné, Emprunt
WHERE Abonné.NumAbo = Emprunt.NumAbo;
```


| NumEmp | ISBN       | NumAbo | ... |
|--------|------------|--------|-----|
| 1      | 1234123410 | 1      | ... |
| 2      | 1234123425 | 2      | ... |
| 3      | 1234123410 | 2      | ... |
| 4      | 1234123475 | 3      | ... |
| ...    | ...        | ...    | ... |

| NumAbo | NomAbo  | PrénomAbo | DateAbo    |
|--------|---------|-----------|------------|
| 1      | Vernier | Alain     | 2017-02-01 |
| 2      | Ducros  | Marie     | 2019-09-04 |
| 3      | Durand  | Arthur    | 2021-09-17 |
| ...    | ...     | ...       | ...        |

# Requêtes multi-relations : jointures (2)

- La sortie est une table contenant le produit conditionnel des relations
  - Les attributs de la sortie sont l'union des attributs.



| NumEmp | ISBN       | NumAbo | DateEmp    | DateRet    | NumAbo | NomAbo  | PrénomAbo | DateAbo    |
|--------|------------|--------|------------|------------|--------|---------|-----------|------------|
| 1      | 1234123410 | 1      | 2017-03-01 | 2017-04-02 | 1      | Vernier | Alain     | 2017-02-01 |
| 2      | 1234123425 | 2      | 2019-12-12 | 2020-01-05 | 2      | Ducros  | Marie     | 2019-09-04 |
| 3      | 1234123410 | 2      | 2021-10-15 | 2021-11-17 | 2      | Ducros  | Marie     | 2019-09-04 |
| 4      | 1234123475 | 3      | 2021-11-22 | NULL       | 3      | Durand  | Arthur    | 2021-09-17 |
| ...    | ...        | ...    | ...        | ...        | ...    | ...     | ...       | ...        |

- Équi-jointure entre les deux tables (clé primaire et clé étrangère).

# Notation pointée


| Notation       | Représente                              |
|----------------|---|
| *              | tous les attributs de toutes les tables |
| Table.Attribut | l'attribut Attribut de la table Table   |
| Table.*        | tous les attributs de la table Table    |

- La notation `Table.Attribut` est nécessaire pour faire référence à `Attribut` s'il est présent dans plusieurs tables de la requête.
- C'est vrai pour toutes les clauses de la requête (`SELECT`, `WHERE`, etc.).
- Il n'est pas possible de spécifier « tous les attributs sauf certains », il faut alors tous les énumérer.

# Requêtes multi-relations : jointures obligatoires

- Sans jointure, le résultat est le produit cartésien des tables.
- En général, ce n'est pas le résultat attendu !

```
SELECT *
FROM Emprunt, Abonné;
```




| NumEmp | ISBN       | NumAbo | DateEmp    | DateRet    | NumAbo | NomAbo  | PrénomAbo | DateAbo    |
|--------|------------|--------|------------|------------|--------|---------|-----------|------------|
| 1      | 1234123410 | 1      | 2017-03-01 | 2017-04-02 | 1      | Vernier | Alain     | 2017-02-01 |
| 1      | 1234123410 | 1      | 2017-03-01 | 2017-04-02 | 2      | Ducros  | Marie     | 2019-09-04 |
| 1      | 1234123410 | 1      | 2017-03-01 | 2017-04-02 | 3      | Durand  | Arthur    | 2021-09-17 |
| ...    | ...        | ...    | ...        | ...        | ...    | ...     | ...       | ...        |
| 1      | 1234123410 | 1      | 2017-03-01 | 2017-04-02 | 148    | Bernard | Françoise | 2021-11-20 |
| 2      | 1234123425 | 2      | 2019-12-12 | 2020-01-05 | 1      | Vernier | Alain     | 2017-02-01 |
| 2      | 1234123425 | 2      | 2019-12-12 | 2020-01-05 | 2      | Ducros  | Marie     | 2019-09-04 |
| 2      | 1234123425 | 2      | 2019-12-12 | 2020-01-05 | 3      | Durand  | Arthur    | 2021-09-17 |
| ...    | ...        | ...    | ...        | ...        | ...    | ...     | ...       | ...        |

# Jointures multiples

- Si la requête utilise **N tables**, il faut **N-1 jointures**.
- Exemple.

```
SELECT *
FROM Livre, Ecrit, Auteur
WHERE Livre.ISBN = Ecrit.ISBN
AND Ecrit.NumAut = Auteur.NumAut;
```



| ISBN          | Titre                | Éditeur | Année     | Édition | ISBN          | NumAut | NumAut | NomAut    | PrénomAut |
|---------------|----------------------|---------|-----------|---------|---------------|--------|--------|-----------|-----------|
| 9782073012722 | Les Vestiges du jour | 1997    | Gallimard | 1       | 9782073012722 | 1      | 1      | Ishiguro  | Kazuo     |
| 9782225805158 | Le langage C         | 1985    | Masson    | 1       | 9782225805158 | 2      | 2      | Kernighan | Brian     |
| 9782225805158 | Le langage C         | 1985    | Masson    | 1       | 9782225805158 | 2      | 2      | Ritchie   | Dennis    |
| ...           | ...                  | ...     | ...       | ...     | ...           | ...    | ...    | ...       | ...       |

# Équi-Jointure

- La jointure supprime les n-uplets n'apparaissant pas dans l'une des tables.
- Exemple :

```
SELECT Abonné.*  
FROM Abonné, Emprunt  
WHERE Abonné.NumAbo = Emprunt.NumAbo;
```

- Les abonnés dont le numéro n'apparaît pas dans Emprunt sont supprimés du résultat.
- Résultat : liste des abonnés qui ont déjà effectué (au moins) un emprunt.
- Chaque abonné apparaîtra dans le résultat autant de fois qu'il a fait d'emprunts.

# Inéqui-jointures

- Une jointure avec un opérateur arithmétique autre que = est appelée **inéqui-jointure**.
  - Exemple :
    - Liste des abonnés ayant emprunté un livre à une date autre que celle de leur inscription
- ```
SELECT Abonné.*  
FROM Abonné, Emprunt  
WHERE Abonné.NumAbo = Emprunt.NumAbo  
AND Abonné.DateAbo <> Emprunt.DateEmp;
```
- Si un abonné n'a fait des emprunts que lors de sa journée d'inscription, il n'apparaîtra pas dans le résultat.



# Jointure et sélection : clause WHERE

- Exemple : sélection des abonnés de nom « Vernier »

```
SELECT * FROM Abonné, Emprunt  
WHERE Abonné.NumAbo = Emprunt.NumAbo  
AND Abonné.NomAbo = 'Vernier';
```

Emprunt

| NumEmp | ISBN       | NumAbo | ... |
|--------|------------|--------|-----|
| 1      | 1234123410 | 1      | ... |
| 2      | 1234123425 | 2      | ... |
| 3      | 1234123410 | 2      | ... |
| 4      | 1234123475 | 3      | ... |
| 5      | 1234123475 | 1      | ... |
| ...    | ...        | ...    | ... |

Abonné

| NumAbo | NomAbo  | PrénomAbo | DateAbo    |
|--------|---------|-----------|------------|
| 1      | Vernier | Alain     | 2017-02-01 |
| 2      | Ducros  | Marie     | 2019-09-04 |
| 3      | Durand  | Arthur    | 2021-09-17 |
| ...    | ...     | ...       | ...        |

# Copies multiples d'une table

- Certaines requêtes nécessitent plusieurs fois la même table : on crée des alias pour la table.
- Exemple : paires d'abonnés qui se sont inscrits à la même date

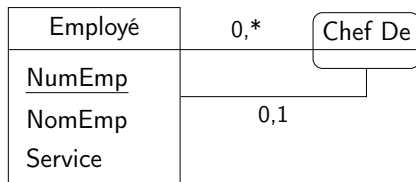
```
SELECT A1.NumAbo, A1.NomAbo, A2.NumAbo,  
       A2.NomAbo, A2.DateAbo  
FROM Abonné A1, Abonné A2  
WHERE A1.DateAbo = A2.DateAbo  
AND A1.NumAbo < A2.NumAbo;
```

- Tout se passe comme si on avait deux tables identiques indépendantes.
- La condition `A1.NumAbo < A2.NumAbo` permet d'éviter d'associer un abonné avec lui-même (car présent dans les deux tables).

| A1.NumAbo | A1.NomAbo | A2.NumAbo | A2.NomAbo | A2.DateAbo |
|-----------|-----------|-----------|-----------|------------|
| 19        | Khammar   | 23        | Caranta   | 2017-10-01 |
| 73        | Ennola    | 91        | Deuring   | 2019-10-04 |
| ...       | ...       | ...       | ...       | ...        |

## Copies multiples d'une table (2)

- Exemple 2

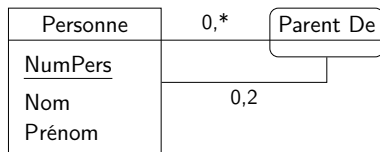


- Le numéro du chef d'un employé est représenté en mettant la clé primaire d'Employé comme clé externe d'Employé :  
`EMPLOYÉ(NumEmp, NomEmp, Service, NumChef)`
- Liste des employés avec leur chef :

```
SELECT Employé.*, Chef.*  
FROM Employé, Employé Chef  
WHERE Employé.NumChef = Chef.NumEmp;
```

## Copies multiples d'une table (3)

- Exemple 3



- Les numéros des parents sont représentés comme des clés étrangères dans la relation Personne.

PERSONNE(NumPers, Nom, Prénom, NumParent1, NumParent2)

- Liste des personnes avec leurs deux parents :

```

SELECT Personne.*, Parent1.*, Parent2.*
FROM Personne, Personne Parent1, Personne Parent2
WHERE Personne.NumParent1 = Parent1.NumPers
AND Personne.NumParent2 = Parent2.NumPers;
  
```

- 1 Notations et exemple fil-rouge
- 2 Jointures
- 3 Opérations ensemblistes**
- 4 Exercices
- 5 Requêtes de définition de données
- 6 Sous-requêtes

# Union, intersection, différence

- Opérateurs ensemblistes :

```
SELECT ...  
UNION | INTERSECT | EXCEPT  
SELECT ... ;
```

applique l'opération ensembliste aux résultats des deux requêtes.

- UNION : union des résultats des requêtes
  - INTERSECT : intersection des résultats
  - EXCEPT : différence des résultats.
- Exemple :

```
SELECT NumAbo FROM Emprunt  
WHERE ISBN = '1234123410'  
INTERSECT  
SELECT NumAbo FROM Emprunt  
WHERE ISBN = '1234123475';
```

# Exemple 1

- Les deux requêtes doivent être compatibles :
  - même nombre d'attributs
  - et attributs de mêmes domaines.
- Union parfois nécessaire pour avoir tous les n-uplets.
  - Exemple :

```
SELECT Employé.*, Chef.NomEmp
FROM Employé, Employé Chef
WHERE Employé.NumChef = Chef.NumEmp
UNION
SELECT Employé.*, NULL
FROM Employé
WHERE NumChef IS NULL;
```

- La jointure de la 1<sup>ère</sup> requête supprime les employés n'ayant pas de chef.
- La 2<sup>e</sup> requête affiche ces employés.
- Résultat : tous les employés sont affichés, avec leur chef s'il existe.

## Exemple 2

- Exemple : livres disponibles :

- « livres qui sont disponibles (qui ne sont pas en cours d'emprunt) »

```
SELECT *  
FROM Livre  
EXCEPT  
SELECT Livre.*  
FROM Emprunt, Livre  
WHERE Livre.ISBN = Emprunt.ISBN  
AND Emprunt.DateRet IS NULL;
```

- La 1<sup>ère</sup> requête donne tous les livres (qu'ils apparaissent dans Emprunt ou non).
- La 2<sup>e</sup> requête donne les livres qui sont en cours d'emprunt.
- Résultat : la différence des deux.



- 1 Notations et exemple fil-rouge
- 2 Jointures
- 3 Opérations ensemblistes
- 4 Exercices**
- 5 Requêtes de définition de données
- 6 Sous-requêtes

# Exercices

- Prénom, nom des abonnés qui ont déjà mis plus de 30 jours à rendre un livre emprunté (éviter les répétitions)
  - sélection sur 2 attributs, jointure
- Prénom, nom des abonnés qui ont emprunté un livre par un auteur de même prénom qu'eux
  - 3 jointures, et sélection sur 2 attributs
- Titre des livres écrits par (au moins) deux auteurs qui portent le même prénom
  - copie (écrire), auto-jointure (e1, e2), jointure (e1, a1), jointure (e2, a2), jointure (e1 ou e2, livre)

- 1 Notations et exemple fil-rouge
- 2 Jointures
- 3 Opérations ensemblistes
- 4 Exercices
- 5 Requêtes de définition de données
  - Création de tables
  - Autres contraintes
- 6 Sous-requêtes

# Langage de définition de données

- Définition de table (relation) de la DB

```
CREATE TABLE table1 (  
    attribut1 type1 [contrainte1],  
    attribut2 type2 [contrainte2],  
    ...,  
    PRIMARY KEY (attribut1),  
    FOREIGN KEY (attribut2) REFERENCES tablei(attributj)  
    ...);
```

- Exemple :

```
CREATE TABLE Personne (  
    NumPers SMALLINT,  
    NomPers VARCHAR(12),  
    PrénomPers VARCHAR(12),  
    DateNaissance DATE );
```

# Types de données numériques

| Type         | Représente                                                              |
|--------------|-------------------------------------------------------------------------|
| SMALLINT     | Nombre entier [-32768, 32767]                                           |
| INT          | Nombre entier [-2147483648, 2147483647]                                 |
| BIGINT       | Nombre entier [-9223372036854775808, 9223372036854775807]               |
| SERIAL       | Nombre entier incrémenté automatiquement (le SGBD gère la numérotation) |
| DOUBLE       | Nombre réel [-1.7976931348623157E+308, 1.7976931348623157E+308]         |
| FLOAT        | Nombre réel [-3.402823466E+38, 3.402823466E+38]                         |
| NUMERIC(n,d) | Nombre réel de n chiffres dont d décimales                              |

- Nombres entiers positifs seulement : UNSIGNED
  - Les valeurs possibles sont décalées (ex : de 0 à 65535 pour SMALLINT)

```
CREATE TABLE Personne (  
    NumPers UNSIGNED SMALLINT,  
    ...);
```

# Types de données autres

| Type       | Représente                    |
|------------|-------------------------------|
| VARCHAR(n) | Chaîne d'au plus n caractères |
| CHAR(n)    | Chaîne d'exactly n caractères |
| TEXT       | Chaîne de taille non limitée  |
| DATE       | Date au format 'aaaa-mm-jj'   |
| TIME       | Heure au format 'hh:mm:ss'    |
| BIT(n)     | Vecteurs de n bits            |

- Chaînes de caractères :
  - VARCHAR si le nombre de caractères peut varier
  - CHAR s'il est fixe (Exemple : immatriculation d'un véhicule)
  - TEXT si le texte peut être très long (plusieurs centaines de caractères)

# Contraintes et clés primaires

- Contraintes :
  - `DEFAULT val` : prend la valeur `val` par défaut
  - `NOT NULL` : valeur `NULL` interdite
  - `UNIQUE` : valeur unique pour chaque n-uplet
  - `REFERENCES Tab(att)` : référence l'attribut `att` dans la table `Tab`, i.e. ses valeurs autorisées sont celles de `att` (nécessaire pour les clés étrangères)
- Clé primaires :
  - `PRIMARY KEY (att)`
  - Le ou les attributs `att` forme(nt) la clé primaire.
  - Entraîne une contrainte d'unicité `UNIQUE` automatiquement.

# Contraintes et clé primaire : exemples

- Exemples :

```
CREATE TABLE Abonné(  
    NumAbo UNSIGNED INT NOT NULL,  
    NomAbo VARCHAR(12) NOT NULL,  
    PrénomAbo VARCHAR(12) DEFAULT 'inconnu',  
    DateAbo DATE,  
    PRIMARY KEY (NumAbo));
```

```
CREATE TABLE Livre(  
    ISBN CHAR(10) NOT NULL,  
    Titre VARCHAR(100) NOT NULL,  
    Editeur VARCHAR(30),  
    Annee UNSIGNED SMALLINT,  
    Edition UNSIGNED TINYINT,  
    PRIMARY KEY (ISBN));
```



# Clés étrangères

- Clés étrangères :

```
FOREIGN KEY (att1) REFERENCES tab2(att2)
```

- L'attribut att<sub>1</sub> est une clé étrangère qui référence l'attribut att<sub>2</sub> qui est clé primaire dans la table tab<sub>2</sub>

- Exemple :

```
CREATE TABLE Écrit(  
  ISBN VARCHAR(10) NOT NULL,  
  NumAut UNSIGNED SMALLINT NOT NULL,  
  PRIMARY KEY (ISBN, NumAut),  
  FOREIGN KEY (ISBN) REFERENCES Livre(ISBN),  
  FOREIGN KEY (NumAut) REFERENCES Auteur(NumAut));
```

# Contraintes sur les n-uplets

- Contrainte générale :

```
CONSTRAINT nomContrainte CHECK (expr)
```

- Exemple :

```
CREATE TABLE Emprunt (  
    NumEmp UNSIGNED SMALLINT NOT NULL,  
    NumAbo UNSIGNED SMALLINT NOT NULL,  
    ISBN VARCHAR(10) NOT NULL,  
    DateEmp DATE NOT NULL,  
    DateRet DATE,  
    CONSTRAINT Contrainte1 CHECK (DateRet >= DateEmp),  
    PRIMARY KEY (NumEmp),  
    FOREIGN KEY (NumAbo) REFERENCES Abonné(NumAbo),  
    FOREIGN KEY (ISBN) REFERENCES Livre(ISBN));
```

# Contraintes de valeurs

- Énumérer les valeurs possibles pour un attribut.
- Exemple :

```
CREATE TABLE JoueurDeTennis (  
    Numéro UNSIGNED SMALLINT NOT NULL,  
    Nom VARCHAR(20) NOT NULL,  
    Age UNSIGNED TINYINT NOT NULL,  
    Latéralité VARCHAR(8) NOT NULL,  
    CONSTRAINT LatéralitésPossibles  
    CHECK (Latéralité = 'Gaucher'  
    OR Latéralité = 'Droitier'),  
    PRIMARY KEY Numéro);
```

- Pour des valeurs nombreuses ou évolutives :
  - créer une table contenant ces valeurs et faire de l'attribut une clé étrangère vers cette table.

- 1 Notations et exemple fil-rouge
- 2 Jointures
- 3 Opérations ensemblistes
- 4 Exercices
- 5 Requêtes de définition de données
- 6 Sous-requêtes**
  - Cas de base
  - Cas avancés
  - Opérateurs
  - Manipulation de données

# Sous-requêtes

- Le résultat d'une requête de sélection peut être utilisé comme critère de sélection d'une autre requête.
  - Forme la plus simple : renvoie une seule valeur

```
SELECT * FROM Emprunt  
WHERE ISBN =  
    (SELECT ISBN FROM Livre  
     WHERE Titre = 'Bases de Données'  
     AND Éditeur = 'Eyrolles');
```

- Parenthèses obligatoires.
- Sous-requête : numéro ISBN du livre édité par Eyrolles intitulé « Bases de Données ».
- La sous-requête est remplacée par la valeur qu'elle renvoie.
- Résultat : emprunts du livre édité par Eyrolles intitulé « Bases de Données ».

## Sous-requêtes : opérateurs

- L'opérateur peut être une inégalité (<>, <, >, <=, >=).
- Exemple :

```
SELECT * FROM Emprunt  
WHERE DateEmp >  
    (SELECT DateAbo  
     FROM Abonné  
     WHERE NumAbo = 10);
```

- Sélections des emprunts effectués après l'abonnement de l'abonné 10.
- Opérateur BETWEEN

```
SELECT * FROM Emprunt  
WHERE DateEmp BETWEEN  
    (SELECT DateAbo FROM Abonné WHERE NumAbo = 1)  
AND (SELECT DateAbo FROM Abonné WHERE NumAbo = 10);
```

## Sous-requêtes : attributs multiples

- La sous-requête peut renvoyer plusieurs attributs

```
SELECT *  
FROM Livre  
WHERE (Titre, Éditeur) =  
      (SELECT Titre, Éditeur  
       FROM Livre  
       WHERE ISBN = '1234123410');
```

- Sous-requête : titre et éditeur du livre 1234123410
- La sous-requête est remplacée par l'ensemble des valeurs qu'elle renvoie.
- Le nombre et le type des attributs comparés doivent correspondre.

## Sous-requête : opérateur [NOT] IN

- Opérateur IN : la valeur doit être dans la liste
  - Exemple : sélection des livres qui ont déjà été empruntés

```
SELECT * FROM Livre
WHERE ISBN IN
      (SELECT ISBN FROM Emprunt);
```

- Opérateur NOT IN : la valeur ne doit pas être dans la liste.
  - Exemple : sélection des livres qui n'ont pas été empruntés depuis le 1<sup>er</sup> mars 2019

```
SELECT * FROM Livre
WHERE ISBN NOT IN
      (SELECT ISBN FROM Emprunt
       WHERE DateEmp >= '2019-03-01');
```



# Imbrication de plusieurs sous-requêtes

- Il est possible d'imbriquer plus de deux sous-requêtes.
  - Exemple

```
SELECT *  
FROM Abonné  
WHERE NumAbo IN  
      (SELECT NumAbo  
       FROM Emprunt  
       WHERE ISBN IN  
            (SELECT ISBN  
             FROM Livre  
             WHERE Éditeur = 'Gallimard'));
```

# Sous-requêtes : opérateur ALL

- Opérateur ALL

- La condition doit être vérifiée par **toutes** les valeurs renvoyées.

```
SELECT *  
FROM Abonné  
WHERE DateAbo >= ALL  
      (SELECT DateAbo  
       FROM Abonné);
```

- Sous-requête : liste des dates d'inscription des abonnés.
- Requête principale : un abonné est sélectionné si sa date d'inscription est supérieure ou égale à **toutes** les dates d'inscription.
- Résultat : liste des abonnés inscrits en dernier.

# Sous-requêtes : opérateur ANY

- Opérateur ANY

- La condition doit être vérifiée par **au moins une** des valeurs renvoyées.

```
SELECT *  
FROM Abonné  
WHERE DateAbo < ANY  
      (SELECT DateAbo  
       FROM Abonné);
```

- Sous-requête : liste des dates d'inscription des abonnés.
- Requête principale : un abonné est sélectionné si sa date d'inscription est strictement inférieure à **au moins une** date d'inscription.
- Résultat : liste des abonnés inscrits avant le(s) dernier(s) inscrit(s).

# Sous-requêtes : opérateur EXISTS

- Opérateur EXISTS

- La condition est vérifiée si la requête renvoie **au moins une** valeur.

```
SELECT *  
FROM Livre  
WHERE EXISTS  
    (SELECT *  
     FROM Emprunt  
     WHERE Emprunt.ISBN = Livre.ISBN);
```

- Pour chaque livre, la sous-requête teste s'il existe un emprunt de ce livre.
- Renvoie la liste des livres qui ont été empruntés **au moins une** fois.

# Sous-requêtes : opérateur NOT EXISTS

- Opérateur NOT EXISTS

- La condition est vérifiée si la requête ne renvoie **rien**.

```
SELECT *  
FROM Livre L  
WHERE NOT EXISTS  
    (SELECT *  
     FROM Emprunt  
     WHERE Emprunt.ISBN = L.ISBN  
     AND Emprunt.DateRet IS NULL);
```

- Pour chaque livre, la sous-requête teste s'il existe un emprunt non terminé de ce livre.
- Renvoie la liste des livres qui ne sont pas en cours d'emprunt.

# Opérateurs EXISTS et NOT EXISTS

- Certaines requêtes ne peuvent être exprimées qu'à l'aide de ces opérateurs.
  - Exemple

```
SELECT * FROM Abonné
WHERE NOT EXISTS
    (SELECT * FROM Livre
     WHERE NOT EXISTS
        (SELECT * FROM Emprunt
         WHERE Emprunt.NumAbo = Abonné.NumAbo
           AND Emprunt.ISBN = Livre.ISBN));
```

- Liste des abonnés pour lesquels il n'existe pas de livre pour lequel il n'existe pas d'emprunt par cet abonné.
- Résultat : liste des abonnés ayant emprunté chaque livre au moins une fois.

# Insertion et modification de données résultant de SELECT

- Insertion de n-uplets résultats de sélection :

```
INSERT INTO table [(attribut1, ..., attributi)]  
(SELECT ...);
```

- Insertion des n-uplets renvoyés par le SELECT.
- Les types des attributs renvoyés doivent correspondre aux types des attributs de la table.
- Exemple :

```
INSERT INTO Livre  
(SELECT '223401248', Titre, Editeur, 2007, Edition+1  
FROM Livre  
WHERE Titre = 'Bases de Données'  
AND Edition = 2);
```

- On peut utiliser IN (SELECT ...) pour des requêtes de modification ou suppression.
  - Exemple : ajouter 1000 aux numéros des abonnés ayant fait des emprunts.

```
UPDATE Abonné SET NumAbo = NumAbo + 1000  
WHERE NumAbo IN (SELECT NumAbo FROM Emprunt);
```

## À suivre

