

# SI3: Exploitation des données

## CHAP II - Introduction au SQL

Initiation au langage de manipulation des données.





## Sommaire

1. Le langage SQL .....	3
1.1 Présentation .....	3
1.2 Classification des commandes.....	3
2. Présentation du thème .....	4
2.1 Modèle Logique des Données (schéma relationnel) .....	4
2.3 Modèle Physique des Données .....	4
2.4 Contenu de la base de données .....	5
3. Consultation des données .....	6
3.1 La projection (consultation simple) .....	6
3.2 La sélection (ou restriction) .....	7
3.2.1 Condition logique simple.....	7
3.2.2 Conditions logiques composées .....	8
3.2.3 Restriction sur une valeur manquante .....	9
3.3 Interrogation de plusieurs relations : la jointure.....	9
3.3.1 Syntaxe SQL89 (SQL-1) .....	9
3.3.2 Syntaxe SQL92 (SQL-2) .....	10
3.4 Tri des résultats .....	11
3.5 Calculs arithmétiques .....	12
3.5.1 Calculs sur attributs.....	12
3.5.2 Les fonctions intégrées.....	12
3.6 Regroupement des résultats .....	13
4. Mise à jour des données .....	14
4.1 Ajout de données .....	14
4.2 Modification de données.....	14
4.3 Suppression de données .....	15
5. Application.....	16
5.1 Base « Vols ».....	16
5.2 Base « Bibliothèque » .....	16



## 1. Le langage SQL

---

### 1.1 Présentation

---

Le **SQL** (Structured Query Language) est un « langage de programmation » particulier qui permet de **manipuler** des **bases de données relationnelles**.

SQL se retrouve aujourd'hui dans la très grande majorité des SGBD, et fonctionne sur des plates-formes allant des gros systèmes aux systèmes embarqués<sup>1</sup> en passant par les micro-ordinateurs et les serveurs.

Exemples de SGBD utilisant SQL pour la gestion des données : MS-ACCESS, ORACLE, SQL SERVER, MySQL, PostgreSQL, SQLite, etc.

SQL n'est pas un langage de programmation au sens classique du terme, c'est un langage qui permet de faire des manipulations sur des bases de données : Les **requêtes**.

### 1.2 Classification des commandes

---

Le langage SQL contient trois grandes familles de commandes :

- ♦ Le langage de description de données (**LDD**) qui permet la création et la modification de la structure de bases de données (tables et attributs, vues, états, index, contraintes d'intégrité ...)
- ♦ Le langage de contrôle des données (**LCD**) qui assure la sécurité des données, et leur confidentialité (qui a le droit de faire quoi, et sur quoi ?), réservé à l'administrateur de la base.
- ♦ Le langage de manipulation de données (**LMD**) qui permet la gestion des données se trouvant dans les tables (consultation, mise à jour, ...)

Dans ce chapitre, nous étudierons uniquement le langage de manipulation des données.

Toutes les requêtes<sup>2</sup> mises en œuvre dans ce cours peuvent être testées en ligne sur [www.defay.net](http://www.defay.net) (Suivre les liens [Ressources](#)>[Modélisation et SGBDR](#)>[Initiation au SQL](#)>[SQL Live](#))

---

<sup>1</sup> C'est le cas par exemple de certaines applications pour smartphones (iPhone, Android,...) qui utilisent une base de données SQLite.

<sup>2</sup> Excepté les requêtes de mise à jour des données.



## 2. Présentation du thème

Pour illustrer les commandes de base du langage de manipulation de données SQL, nous utiliserons le système d'information décrit ci-après.

### 2.1 Modèle Logique des Données (schéma relationnel)

a) Représentation graphique :

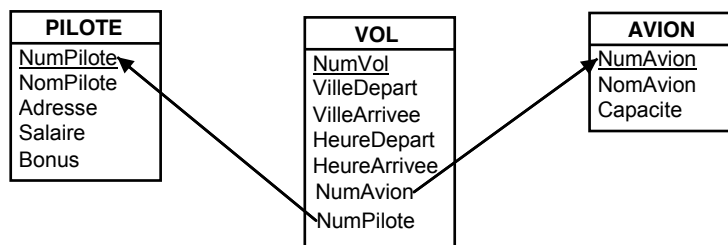


Figure 1 : MLD « vols »

b) Représentation en intention :

PILOTE(NumPilote, NomPilote, Adresse, Salaire, Bonus)

AVION(NumAvion, NomAvion, Capacite)

VOL(NumVol, VilleDepart, VilleArrivee, HeureDepart, HeureArrivee, #NumAvion, #NumPilote)

c) Représentation détaillée :

PILOTE(NumPilote, NomPilote, Adresse, Salaire, Bonus)

NumPilote : Clé primaire

AVION(NumAvion, NomAvion, Capacite)

NumAvion : Clé primaire

VOL(NumVol, VilleDepart, VilleArrivee, HeureDepart, HeureArrivee, NumAvion, NumPilote)

NumVol : Clé primaire

NumAvion : Clé étrangère en référence à NumAvion de AVION

NumPilote : Clé étrangère en référence à NumPilote de PILOTE

### 2.3 Modèle Physique des Données

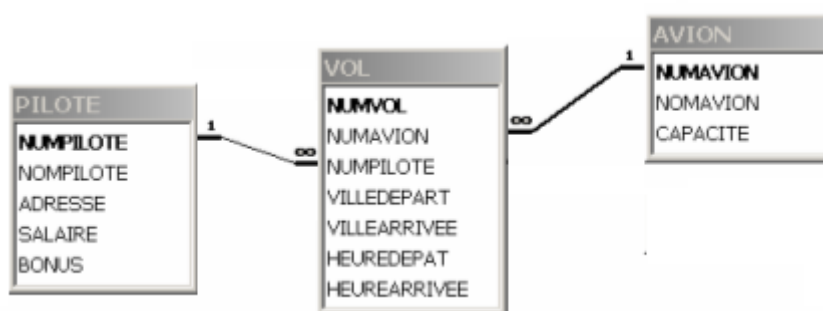



Figure 2 : MPD « vols » sous Ms-Access

Remarques :

- Le lien  matérialise une relation de type « Un à plusieurs »

Entre les tables AVION et VOL, il précise que le champ *NumAvion* est présent une seule fois dans la table AVION et plusieurs fois dans VOL.

- Dans la table VOL, les clés étrangères (*NumAvion* et *NumPilote*) ne se distinguent pas des autres propriétés (absence du # présent dans le MLD)



## 2.4 Contenu de la base de données<sup>3</sup>

PILOTE : Table							
	NUMPILOTE	NOMPILOTE	ADRESSE	SALAIRE	BONUS		
▶	1	TOTO	97438 Sainte Marie	1000	1100		
	2	PAYET	97487 Saint Denis	3000	800		
	3	HOAREAU	97420 Le Port	3300	950		
	4	DUPUIS	75130 Paris	3200	900		
	5	DUPOND	75150 Paris	3500	1000		
	6	PETIT	34000 Montpellier	2000	500		
	7	VITRI	34070 Montpellier	2500			
*	(NuméroAuto)						
Enr: 1 sur 7							

**Figure 3 : Contenu de la table PILOTE**

AVION : Table			
	NUMAVION	NOMAVION	CAPACITE
▶	1	A340-300	295
	2	ATR72	70
	3	B747-400	420
	4	A319	120
	5	ATR42/72	50
	6	B747-400	420
*	NuméroAuto)		
Enr:	1	sur 6	

**Figure 4 : Contenu de la table AVION**

VOL : Table							
	NUMVOL	NUMAVION	NUMPILOTE	VILLEDEPART	VILLEARRIVEE	HEUREDEPAT	HEUREARRIVEE
▶	AF 380	3	1	PARIS ORLY	GILLOT	21:00:00	10:45:00
	AF 7684	4	6	PARIS CDG	MONTPELLIER	20:45:00	22:10:00
	MK 203	5	4	GILLOT	MAURICE	07:30:00	08:45:00
	MK 230	2	3	GILLOT	MAURICE	11:40:00	12:25:00
	MK 40	6	2	MAURICE	PARIS CDG	09:40:00	19:05:00
	MK 45	1	7	PARIS CDG	GILLOT	11:40:00	12:25:00
*							
Enr: 1 sur 6							

**Figure 5 : Contenu de la table VOL**

<sup>3</sup> Cette base de données est à considérer « pour l'exercice ».  
La pertinence des informations qu'elle contient doit être occultée...



### 3. Consultation des données

La consultation (ou l'interrogation) des données constitue l'opération la plus fréquemment utilisée en langage SQL. Elle est réalisée en utilisant la commande **SELECT**.

Sa syntaxe générale est la suivante :

```
SELECT    [ALL] | [DISTINCT] <liste des noms de colonnes> | *  
FROM      <Liste des tables>  
[WHERE <condition logique>]
```

Il existe d'autres options pour la commande SELECT :

ORDER BY ...

GROUP BY ...

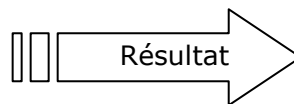
HAVING ...

Note : Les instructions entre crochets sont facultatives.

#### 3.1 La projection (consultation simple)

Cette opération permet de sélectionner une partie des attributs (les colonnes) d'une ou plusieurs tables.

**R1** : « Quels renseignements possédons-nous sur tous les avions ? »

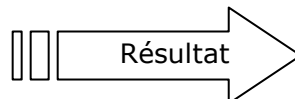


Requête1		
NumAvion	NomAvion	Capacite
1	A340-300	295
2	ATR72	70
3	B747-400	420
4	A319	120
5	ATR42/72	50
6	B747-400	420

Tous les tuples de la relation AVION sont ainsi sélectionnés.

La projection peut être limitée à un choix d'attributs en indiquant, à la place de l'astérisque, une liste de noms d'attributs.

**R2** : « Quels sont les noms et les capacités des avions de la compagnie ? »



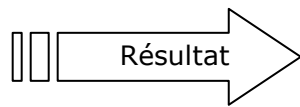
Requête2	
NomAvion	Capacite
A340-300	295
ATR72	70
B747-400	420
A319	120
ATR42/72	50
B747-400	420



La clause **DISTINCT** ajoutée à la clause **SELECT** permet d'éliminer les doublons.

☞ Si dans le résultat plusieurs tuples sont identiques, un seul sera conservé.

**R3** : « Quels sont les différents avions de la compagnie et leur capacité ? »



NomAvion	Capacite
A319	120
A340-300	295
ATR42/72	50
ATR72	70
B747-400	420

Note : L'option **ALL** est, par opposition à l'option **DISTINCT**, l'option par défaut. Elle permet de sélectionner l'ensemble des lignes.

### 3.2 La sélection (ou restriction)

La sélection est l'opération qui permet de **sélectionner des lignes** d'une ou plusieurs tables répondant à certains **critères**.

En SQL, les restrictions s'expriment à l'aide de la clause **WHERE** suivie d'une condition logique exprimée à l'aide d'opérateurs logiques.

La condition (expression logique ayant soit la valeur « vrai », soit la valeur « faux ») sera évaluée pour chaque tuple de la relation résultante.

Les tuples pour lesquels la condition est « vraie » sont ainsi sélectionnés.

#### 3.2.1 Condition logique simple

C'est le résultat de la comparaison de deux expressions au moyen d'un opérateur de comparaison :

- ♦ { = } ; { != ou < > } ; { > } ; { < } ; { <= } ; { >= } (égalité, différence...)
- ♦ **BETWEEN** <expression 1> **AND** <expression 2>

Le prédicat **BETWEEN** permet de vérifier qu'une valeur se trouve dans un intervalle.

- ♦ **LIKE** <chaîne de caractères>

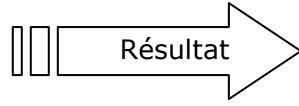
Le prédicat **LIKE** permet de faire des comparaisons sur des chaînes grâce à des caractères, appelés caractères jokers<sup>4</sup> :

- Le caractère **%** permet de remplacer une séquence de caractères (éventuellement nulle)
- Le caractère **\_** permet de remplacer un caractère (l'équivalent du "blanc" au scrabble ;-)
- Les caractères **[ - ]** permettent de définir un intervalle de caractères (par exemple [A-D])

- ♦ **IN** (<expression 1, expression 2, ...>)

Le prédicat **IN** permet de vérifier qu'une valeur appartient à une liste de valeurs:

**R4** : « Quels sont les avions de capacité supérieure ou égale à 100 »



NomAvion	Capacite
A340-300	295
B747-400	420
A319	120
B747-400	420

<sup>4</sup> Ces caractères peuvent être différents suivant le SGBDR utilisé.



**R5** : « Quels sont les pilotes (Nom, Bonus et Salaire) dont le bonus est supérieur au salaire ? »

Requête5		
NomPilote	Bonus	Salaire
TOTO	1100	1000

**R6** : « Quels sont les pilotes dont le bonus est compris entre 800 et 1000 € ? »

Requête6	
NomPilote	Bonus
PAYET	800
HOAREAU	950
DUPUIS	900
DUPOND	1000

**R7** : « Quels sont les pilotes dont le nom commence par "DUP" ? »

Requête7	
NOMPILOTE	
DUPUIS	
DUPOND	

Remarque : Sous MS-Access le "%" est remplacé par "\*"

**R8** : « Quels sont les numéros des vols dont la ville d'arrivée est "GILLOT" ou "MAURICE" ? »

Requête8	
NumVol	
AF 380	
MK 203	
MK 230	
MK 45	

### 3.2.2 Conditions logiques composées

Les opérateurs logiques **AND** et **OR** sont utilisés pour combiner plusieurs conditions.

Ainsi, les requêtes R6 et R8 pouvaient également être écrites comme suit :

Requête R6 Bis

Requête R8 Bis

**R9** : « Quels sont les pilotes dont le bonus est inférieur à 800€ (peu importe la ville où ils résident) et ceux dont le salaire est inférieur à 3500€ mais habitant PARIS ? »

Requête9			
NomPilote	Bonus	Salaire	Adresse
DUPUIS	900	3200	75130 Paris
PETIT	500	2000	34000 Montpellier

Remarques :

- L'opérateur **AND** est prioritaire par rapport à l'opérateur **OR**.
- Si les interrogations concernant la ville des pilotes sont fréquentes, l'attribut *Adresse* devra être **décomposé** afin d'accélérer les traitements et d'assurer la pertinence des résultats.





**R9bis** : « Quels sont les pilotes qui habitent à Ste-Marie ou à St-Denis et dont le bonus est supérieur à 1000€ ? »

Requête9bis	
NomPilote	Bonus
TOTO	1100

### 3.2.3 Restriction sur une valeur manquante

Lorsqu'un champ n'est pas renseigné, le SGBD lui attribue une valeur spéciale que l'on note **NULL**. La recherche de cette valeur ne peut pas se faire à l'aide des opérateurs classiques, il faut utiliser les prédicats **IS NULL** ou bien **IS NOT NULL**.

**R10** : « Quels sont les pilotes n'ayant pas de bonus ? »

Requête10	
NumPilote	NomPilote
7	VITRI

## 3.3 Interrogation de plusieurs relations : la jointure

La jointure est l'opération permettant d'obtenir des informations provenant de plusieurs relations.

Une jointure est formulée en spécifiant :

- les relations concernées dans la clause **FROM**,
- la ou les conditions dans la clause **WHERE** portant sur l'égalité des attributs communs aux relations concernées par la jointure.

Note : Les jointures portent toujours sur les **clés primaires** et **les clés étrangères** des relations.

Exemple (R11) : « Pour chaque numéro de vol, on désire connaître le nom des avions »

Cette requête concerne des attributs provenant de plusieurs relations :

- *NumVol* dans la relation VOL
- *NomAvion* dans la relation AVION

### 3.3.1 Syntaxe SQL89 (SQL-1)

La clause **FROM** devra indiquer la liste des relations concernées : VOL et AVION.

La clause **WHERE** devra contenir :

- une condition d'égalité pour les attributs communs aux deux relations.
- si les attributs concernés portent le même nom, on les prefixera du nom de la relation correspondante.

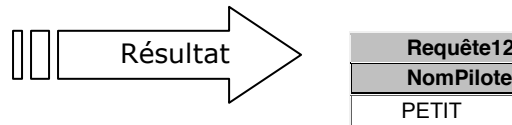
Soit :

Requête11	
NumVol	NomAvion
MK 45	A340-300
MK 230	ATR72
AF 380	B747-400
AF 7684	A319
MK 203	ATR42/72
MK 40	B747-400

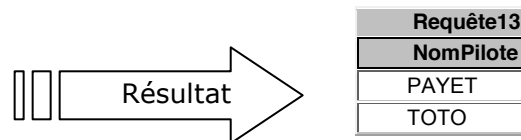


Une jointure peut être associée à une restriction :

**R12** : « Nom du pilote qui assure le vol "AF7684" »



**R13** : « Nom des pilotes qui ont déjà piloté un "BOEING 747-400" ? »



### 3.3.2 Syntaxe SQL92 (SQL-2)

La clause **FROM**, combinée avec **INNER JOIN**<sup>5</sup>, devra indiquer la liste des relations concernées : VOL et AVION ainsi que les champs sur lesquels portera la jointure.

La clause **WHERE** contiendra les éventuelles restrictions.

👉 C'est le point fort du SQL-2 : **séparer ce qui relève de la jointure du reste de la requête.**

Ainsi, pour les requêtes précédentes, on obtiendra :

**R11** : « Pour chaque numéro de vol, on désire connaître le nom des avions »

**R12** : « Nom du pilote qui assure le vol "AF7684" »

**R13** : « Nom des pilotes qui ont déjà piloté un "BOEING 747-400" ? »

<sup>5</sup> INNER JOIN correspond à une jointure classique en SQL-1. Les autres clauses (LEFT JOIN, RIGHT JOIN...) seront abordées plus tard.



### 3.4 Tri des résultats

Les tuples constituant le résultat d'une requête sont obtenus dans un ordre indéterminé dépendant des mécanismes internes du SGBDR utilisé.

On peut demander en fin d'instruction **SELECT** que le résultat soit ordonné de manière ascendante ou descendante suivant un ou plusieurs attributs.

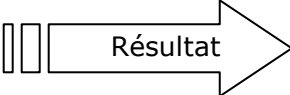
Les critères de tri sont indiqués dans la clause **ORDER BY** dont la syntaxe est la suivante :

```
ORDER BY <Attribut 1> [ASC|DESC] [, <Attribut 2> [ASC|DESC]] ...
```

Le tri se fait tout d'abord selon le premier attribut, puis les tuples ayant une même valeur pour ce premier attribut sont classés selon le deuxième attribut, etc...

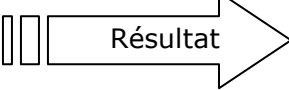
Par attribut, le tri peut être ascendant (**ASC** par défaut) ou descendant (**DESC**).

**R14** : « Liste détaillée des avions triée par ordre alphabétique croissant (sur NomAvion) »



NumAvion	NomAvion	Capacite
4	A319	120
1	A340-300	295
5	ATR42/72	50
2	ATR72	70
6	B747-400	420
3	B747-400	420

**R15** : « Liste des pilotes ayant un bonus. Les bonus seront classés dans l'ordre décroissant. »




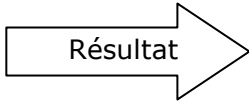
NomPilote	Bonus
TOTO	1100
DUPOND	1000
HOAREAU	950
DUPUIS	900
PAYET	800
PETIT	500

## 3.5 Calculs arithmétiques

### 3.5.1 Calculs sur attributs

Les attributs numériques d'une table peuvent être utilisés pour réaliser des calculs.

**R16** : « Quels sont les pilotes qui gagnent plus de 4200 € (bonus compris) ? »

		Résultat	Requête16
			NomPilote
			HOAREAU
			DUPOND

### 3.5.2 Les fonctions intégrées

Ces fonctions du langage SQL effectuent un **calcul** sur des **ensembles de valeurs**.

**AVG** (<Attribut>) : Moyenne arithmétique


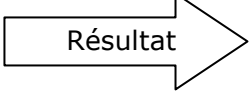
**SUM** (<Attribut>) : Somme arithmétique

**MAX** (<Attribut>) : Valeur maximum


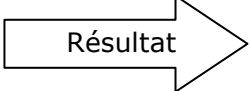
**MIN** (<Attribut>) : Valeur minimum

**COUNT** (\*), **COUNT** (<Attribut>), **COUNT** (**DISTINCT** <Attribut>) : Nombre de tuples

**R17** : « Quel est le salaire moyen des pilotes ? »


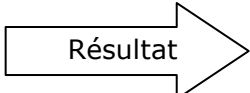
		Résultat	Requête17
			Expr1000
			2642,85714285714

**R18** : « Quel est le plus gros salaire ? »


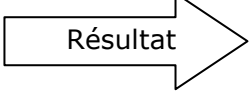
		Résultat	Requête18
			PlusGrosSalaire
			3500

Note : L'utilisation de **AS** permet de donner un nom d'alias à une colonne créée.


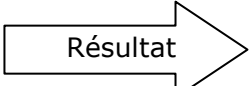
**R19** : « Combien d'avions disposent de plus de 100 places ? »

		Résultat	Requête19
			NbPlus100
			4

**R20** : « Combien d'avions différents existe-t-il ? »

		Résultat	Requête20
			NbAvion
			5

**R21** : « Quel est le pourcentage de pilotes avec bonus ? »

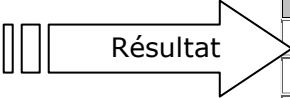
		Résultat	Requête21
			%PiloteAvecBonus
			85.71



### 3.6 Regroupement des résultats

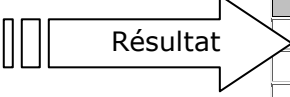
Il peut être intéressant de regrouper des résultats afin de faire des opérations par groupe (opérations statistiques par exemple). Cette opération se réalise à l'aide de la clause **GROUP BY**, suivie du nom de chaque colonne sur laquelle on veut effectuer des regroupements.

**R22** : « Combien y a-t-il de vol(s) au départ de chaque ville ? »



VilleDepart	NbVoIs
GILLOT	2
MAURICE	1
PARIS CDG	2
PARIS ORLY	1

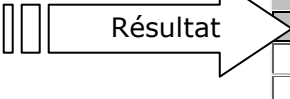
**R23** : « Combien y a-t-il de places tous vols confondus au départ de chaque ville ? »



VILLEDEPART	NbPlaces
GILLOT	120
MAURICE	420
PARIS CDG	415
PARIS ORLY	420

La clause **HAVING** va de pair avec la clause **GROUP BY**, elle permet d'appliquer une restriction sur les groupes créés grâce à la clause **GROUP BY**.

**R24** : « Quelles sont les villes pour lesquelles il y a au moins 2 vols à l'arrivée ? »



VilleArrivee	NbArrivee
GILLOT	2
MAURICE	2



## 4. Mise à jour des données

### 4.1 Ajout de données

L'insertion de nouvelles données dans une table se fait grâce à la commande **INSERT**.

Sa syntaxe générale est la suivante :

```
INSERT INTO Nom_de_la_table [(liste des noms de colonnes)]  
VALUES (Valeur1,Valeur2,Valeur3,...)
```

Notes :

- Les données sont affectées aux colonnes dans l'ordre dans lequel elles ont été créées.
- Lorsque les valeurs sont des chaînes de caractères (ou bien des dates), il faut les délimiter par des guillemets.

**Ra** : « Ajouter l'avion suivant dans la base : N°7 ; Modèle A319 ; capacité 100 »

### 4.2 Modification de données

La modification de données consiste à modifier des tuples (des lignes) dans une table à l'aide de la commande **UPDATE**. La modification à effectuer est précisée après la clause **SET**.

Sa syntaxe générale est la suivante :

```
UPDATE Nom_de_la_table  
SET Colonne1 = Valeur_Ou_Expression1 [, Colonne2 = Val_Ou_Expr2]...  
[WHERE Conditions]
```

Note :

- Valeur\_Ou\_Expression peut être une expression algébrique, une constante ou un résultat provenant d'une clause **SELECT**.
- La clause facultative **WHERE** permet de préciser les tuples sur lesquels la mise à jour aura lieu.

**Rb** : « Appliquer une augmentation de 5% sur le salaire de tous les pilotes »



**Rc** : « Pour tous les pilotes (y compris sans bonus), majorer de 25€ les bonus inférieurs à 1000€ »

☞ Cette mise à jour nécessite deux requêtes pour pouvoir prendre en compte les bonus ayant pour valeur « Null »

**Rc1**

**Rc2**

Remarque : Dans le cas présent, il était plus judicieux d'affecter au bonus une valeur nulle (zéro) par défaut.

### 4.3 Suppression de données

La suppression de données dans une table se fait grâce à la commande **DELETE**.

- La clause **FROM** précise la table sur laquelle la suppression s'effectue.
- La clause **WHERE** précise l'ensemble des lignes qui seront supprimées.

Sa syntaxe générale est la suivante :

```
DELETE FROM Nom_de_la_table  
[WHERE Condition]
```

Note :

- La commande **DELETE** est à utiliser avec précaution car l'opération de suppression est irréversible.
- Il est préférable et surtout plus prudent de s'assurer dans un premier temps que les lignes sélectionnées sont bien les lignes que l'on désire supprimer !
- La clause **WHERE** est facultative mais sera la plupart du temps renseignée sans quoi, c'est le contenu complet de la table qui sera effacé.

**Rd** : « Supprimer l'avion ajouté par la requête Ra »



## 5. Application

### 5.1 Base « Vols »

Écrire les requêtes suivantes :

**R25** : Quels sont les vols (*NumVol*) triés par ordre croissant, assurés par Toto ?

**R26** : Combien de vols y a-t-il au départ de Gillot ?

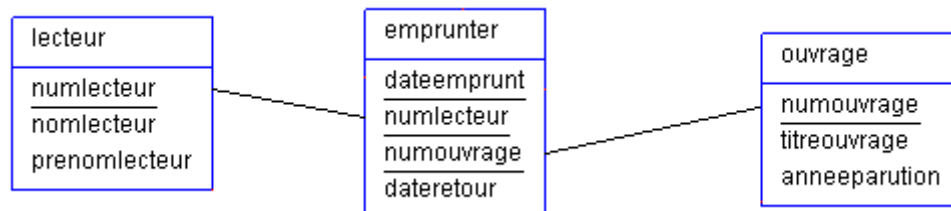
**R27** : Combien de vols sont assurés par des ATR ?

**R28** : Liste des vols (*NumVol* et *NumAvion*) au départ de Paris dont la capacité est supérieure à 400 places ?

**R29** : Liste des avions (*NumAvion* et *NomAvion*) pilotés par le pilote nommé Hoareau au départ de Gillot ?

### 5.2 Base « Bibliothèque »

Soit le système d'information simplifié d'une bibliothèque dont voici le schéma relationnel :



a) Expliquer pourquoi le champ *dateemprunt* fait partie de la clé primaire. N'y a-t-il pas une clé primaire plus simple pour la table « emprunter » ?

b) Écrire les requêtes suivantes :

**R1** : Qui est le lecteur n°30 ?

**R2** : Liste des lecteurs dont le nom de famille commence par la lettre « S »

**R3** : Liste des ouvrages qui traitent de la méthode Merise.

**R4** : Liste des ouvrages qui traitent du « registre » à partir de l'année 2000.

**R5** : Liste des lecteurs qui portent le nom (ou le prénom) « BERTRAND »

**R6** : Quels sont le titre et l'année de parution de l'ouvrage le plus ancien ☺

**R7** : Titre des ouvrages actuellement empruntés ou ayant déjà été empruntés

**R8** : Nom des lecteurs ayant effectué au moins un emprunt depuis le début de l'année 2013.

**R9** : Liste des emprunts (titres et dates) réalisés par le lecteur nommé « HAMSI »

**R10** : Liste des emprunts en cours. Pour chaque emprunt, nom du lecteur, titre de l'ouvrage et date d'emprunt.

**R11** : Nombre total d'emprunts enregistrés à ce jour.

**R12** : Ajouter l'ouvrage suivant dans la base : N°40, Titre ouvrage : « Réseaux de neurones », Année de parution : 2004

**R13** : Enregistrer le retour de l'ouvrage n°17 à la date du jour. La fonction *date()* retourne la date du système sous Access. Sous MySQL c'est la fonction *now()* ou la variable *current\_date*.

**R14** : Supprimer tous les emprunts régularisés (ouvrages restitués) de l'année 2011.

**R15** : Combien reste-t-il d'emprunts non régularisés pour l'année 2012 ?

**R16** : Nombre total d'emprunts réalisés par lecteurs.