

## **BES WEBDEVELOPER (5XBDR)**

### **Système de gestion de bases de données**

L'étudiant sera capable :

*dans le cadre de mise en situations simples et de l'observation de cas réels, en disposant d'une station informatique opérationnelle équipée d'un logiciel "Bases de données" et en développant des compétences de communication:*

- ◆ d'identifier une base de données ;
- ◆ de présenter les éléments essentiels d'un système de gestion de bases de données (SGBD) ;
- ◆ d'expliciter les mécanismes relationnels et le schéma relationnel dans une base de données ;
- ◆ de schématiser une base de données à partir d'un problème pratique en justifiant les choix effectués ;

*à l'aide d'un outil approprié et d'un langage de requête tel que SQL (Structured Query Language) :*

- ◆ de créer une base de données par :
  - l'identification et la création de tables,
  - l'identification et la création d'index (clé primaire, clé étrangère,...),
  - l'identification des champs et la définition à bon escient du type de données,
  - l'identification et la création des relations entre les tables ;
- ◆ de mettre à jour une base de données par:

## **BACHELIER EN INFORMATIQUE DE GESTION (5IBD1)**

### **Initiation aux bases de données**

L'étudiant sera capable :

*dans le cadre d'applications issues des environnements informatique et technique, en disposant d'une station informatique opérationnelle équipée d'un logiciel « Bases de données » et en développant des compétences de communication,*

- ◆ de définir une base de données ;
- ◆ de présenter les éléments essentiels d'un système de gestion de bases de données (SGBD) ;
- ◆ de créer une table, un index en utilisant différents types de données et de formats d'affichage de ces données sur un système de gestion de bases de données relationnelles ;
- ◆ d'expliciter les mécanismes relationnels et le schéma relationnel dans une base de données ;
- ◆ d'implémenter sur des exemples pratiques le schéma relationnel ;
- ◆ d'utiliser une clé primaire et les vues ;
- ◆ d'introduire et d'utiliser des tables à jonctions (jointure) ;
- ◆ d'utiliser les éléments essentiels d'un langage tel que SQL ;
- ◆ de créer des tables à l'aide du langage choisi ;

- la modification de la structure d'une table,
  - la modification des index,
  - la modification des relations entre les tables ;
  - ◆ d'intervenir sur le contenu de la base de données par :
    - l'ajout, la modification et la suppression de données,
  - ◆ d'interroger une base de données par ;
    - des requêtes de sélection (simple, multiple, avec tri, avec filtre, avec jointure, avec regroupement, calculée, ...),
    - des requêtes ensemblistes (union, intersection, différence,...) ;
  - ◆ d'importer et d'exporter des données ;
- de recourir à bon escient à la documentation disponible.
- ◆ d'effectuer des sélections à l'aide du langage de requête : requêtes, tri simple, tri multiple, élimination des doublons, requêtes avec création de champs, jointure, regroupement,... applications pratiques sur un système de gestion de bases de données relationnelles ;
  - ◆ d'appliquer les opérations ensemblistes en SQL,... ;
  - ◆ d'importer et d'exporter des données.

# Table des matières

1. Introduction – Théorie des bases de données relationnelles .....	7
1.1 Définitions.....	7
2 Administration avec l'outil phpMyAdmin.....	9
2.1 Présentation .....	9
2.2 Pourquoi utiliser la ligne de commande ? .....	10
3. Création d'une base de données .....	10
3.1 Identification et création de tables .....	10
3.2 Identification des champs et leurs types de données .....	12
3.2.1 Types des attributs.....	12
3.2.2 Entiers.....	13
3.2.3 Flottants .....	13
3.2.4 Chaînes .....	14
3.2.5 Dates et heures.....	15
3.2.6 Ensembles .....	15
3.3 Création des relations entre les tables .....	16
3.3.1 Clé primaire .....	16
3.3.2 Attribut non nul.....	16
3.3.3 Valeur par défaut .....	16
3.3.4 Attribut sans doublon .....	17
3.3.5 Index .....	17
3.3.6 Exercices – création de table .....	18
4. Gestion du contenu de la base de données.....	19
4.1 Ajout de données .....	19
4.2 Modification de données.....	21
4.3 Suppression de données.....	22
4.4 Exercices – insertion de données.....	23
5. Intégrité référentielle .....	23

5.1 Exercices – Analyse/ Intégrité référentielle .....	26
6. Mise à jour d'une base de données .....	29
6.1 Modification d'une table .....	29
6.1.1 Ajouter un attribut .....	29
6.1.2 Supprimer un attribut .....	29
6.1.3 Créer une clé primaire .....	30
6.1.4 Supprimer une clé primaire .....	30
6.1.5 Ajout d'une contrainte d'unicité .....	31
6.1.6 Changer la valeur par défaut d'un attribut .....	31
6.1.7 Changer la définition d'un attribut .....	31
6.1.8 Changer le nom d'une relation .....	32
6.1.9 Ajouter un index .....	32
6.1.10 Supprimer un index .....	33
6.2 Suppression d'une table .....	33
6.3 Exercices – mise à jour d'une base de données .....	33
7 Interrogation de base de données .....	35
7.1 Requêtes de sélection .....	35
7.1.1 Simple .....	35
7.1.2 Avec filtre .....	36
7.1.3 Avec tri .....	37
7.1.4 Exercices – Sélection simple/filtre/tri .....	38
7.1.5 Avec jointure .....	38
7.1.6 Multiple .....	39
7.1.6.1 Requêtes imbriquées : .....	41
7.1.6.2 Requêtes sélection multi-tables: .....	41
7.1.7 Avec regroupement .....	42
7.1.7.1 MySQL gère-t-il le GROUP BY comme les autres SGBD ? .....	44
7.1.7.2 Regroupement sur plusieurs critères : .....	44
7.1.8 Champs calculés .....	45

7.1.9 Fonctions de MySQL .....	45
7.1.9.1 Quelques exemples .....	46
7.1.9.2 Fonctions de comparaison de chaînes.....	46
7.1.9.3 Fonctions mathématiques .....	47
7.1.9.4 Fonctions de chaînes .....	47
7.1.9.5 Fonctions de dates et heures .....	48
7.1.9.6 Fonctions à utiliser dans les GROUP BY .....	48
7.1.10 Les jointures internes.....	50
7.1.11 Les jointures externes.....	51
7.1.11.1 left outer .....	51
7.1.11.2 right outer .....	52
7.1.12 Exercices - jointures internes/externes/requêtes jointures/multiples/avec regroupement/champ calculé .....	54
7.2 Requêtes ensemblistes .....	58
7.2.1 Union .....	58
7.2.2 Intersection .....	59
7.2.3 Différence .....	60
7.2.4 Le produit cartésien .....	60
7.2.5 Exercices - Requêtes ensemblistes .....	62
7.3 fonction Coalesce .....	62
7.3.1 Exercices - coalesce.....	63
7.4 GROUP BY WITH ROLLUP .....	63
7.4.1 Exercices - WITH ROLLUP .....	64
8. Importation/Exportation des données .....	64
8.1 Réalisation d'un backup sous PHPMyAdmin.....	64
8.2 Restaurer une base de donnée sous PHPMyAdmin .....	66
8.2.1 Exemple d'importation sous OVH .....	66
8.3 Exporter une base de données MySQL.....	68
8.4 Sauvegarder une base de données MySQL avec mysqldump et PHP .....	68

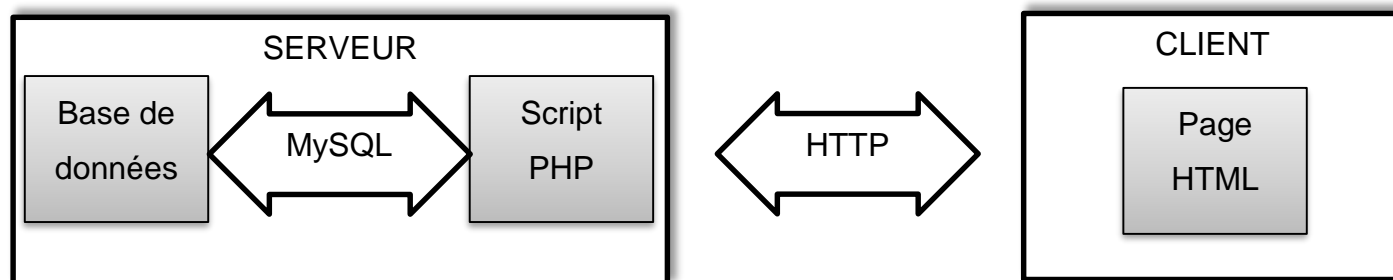
8.4 Exercices – mysqldump .....	70
9. Cas pratiques – Analyse .....	70
9.1 Service achats .....	70
9.2 Élections en Syldavie .....	70
9.3 Bibliothèque.....	70
9.4 Région Bruxelloises – administration .....	71
9.5 Région Bruxelloises – bâtiments .....	71
9.6 Région Bruxelloises – administration/bâtiments V1 .....	71
9.7 Région Bruxelloises – administration/bâtiments V2.....	71
9.8 IFOSUP .....	72
9.9 Agriculteur .....	72
9.10 Forum .....	73
9.11 Camping .....	73
9.12 Fleuves.....	73
9.13 Gestion d’anciens .....	74
9.14 Gestion des logements dans une agence immobilière .....	74
9.15 Pairi Daiza .....	75
9.16 Toutenbois.....	76
9.17 UGC .....	76
10 Cas pratiques – Requêtes.....	77
10.1 bibliothèque .....	77
10.2 Clinique .....	78
10.3 Montagnes.....	79
10.4 Spectacles.....	80
10.5 voyages .....	81
10.6 World .....	82
10.7 Gestion d’une école.....	83
10.8 Comptoir.....	87
12. Evaluation .....	89

## 1. Introduction – Théorie des bases de données relationnelles

« En informatique, une base de données relationnelle est un stock d'informations décomposées et organisées dans des matrices appelées relations ou tables » ©Wikipedia

MySQL dérive directement de SQL (Structured Query Language) qui est un langage de requête vers les bases de données exploitant le modèle relationnel.

Le serveur de base de données MySQL est très souvent utilisé avec le langage de création de pages web dynamiques : PHP. Il sera discuté ici des commandes MySQL utilisables via PHP dans les conditions typiques d'utilisation dans le cadre de la gestion d'un site web.



### 1.1 Définitions

Domaine : ensemble des valeurs d'un attribut.

Relation : sous ensemble du produit cartésien d'une liste de domaines. C'est en fait un tableau à deux dimensions dont les colonnes correspondent aux domaines et dont les lignes contiennent des enregistrements (=tuples). On associe un nom à chaque colonne.

Une *relation* est une *table* comportant des *colonnes* (appelées aussi *attributs*) dont le *nom* et le *type* caractérisent le contenu qui sera inséré dans la table.

Imaginons que l'on veuille stocker dans notre base de données notre carnet d'adresses. On va donc créer la relation **Personne** qui aura pour attributs : *nom*, *prénom*, *adresse*, *téléphone*.

Autrement dit, c'est une table nommée **Personne** possédant les colonnes : *nom*, *prénom*, *adresse*, *téléphone*.

Les *lignes* que contiendra cette table seront appelées *enregistrements* ou *tuples*.

Personnes			
Nom	Prenom	Adresse	telephone
Dupond	Marc	8, Rue de l'octet	0123456789

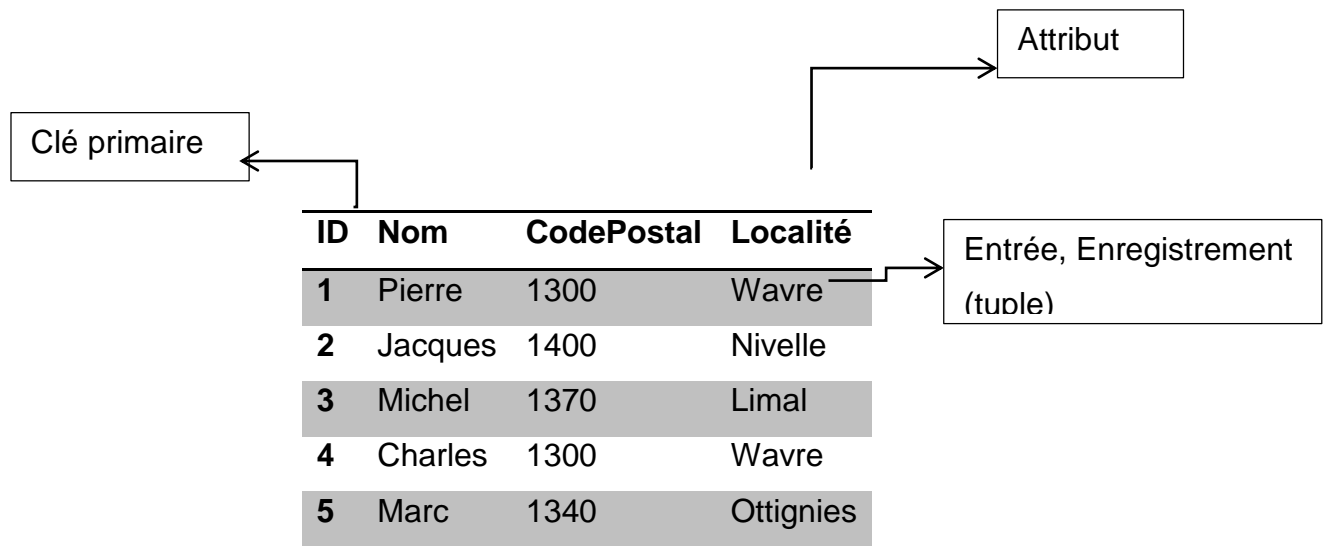
L'algèbre relationnelle regroupe toutes les opérations possibles sur les relations.

**Attribut** : une colonne d'une relation, caractérisé par un nom.

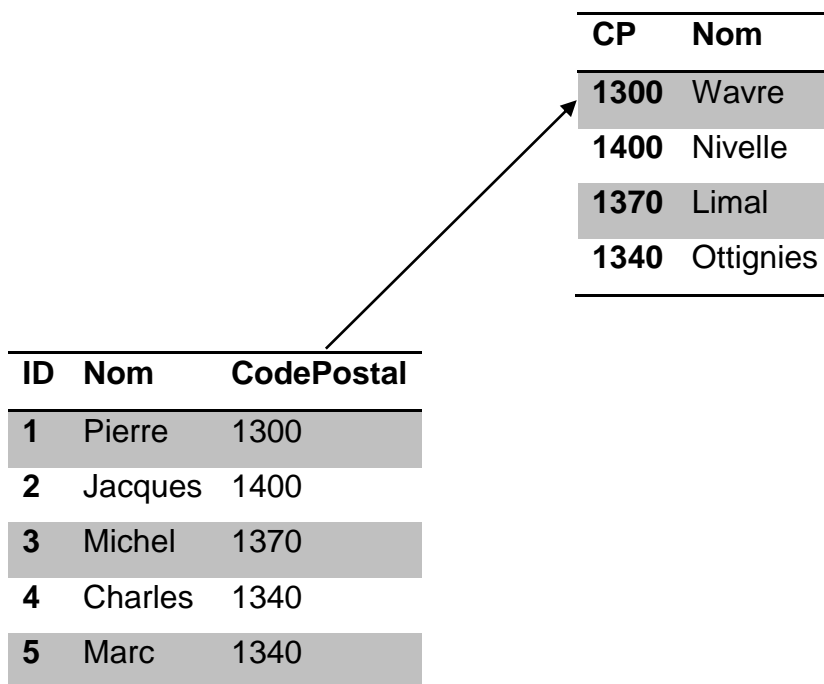
**Enregistrement ou Tuple** : liste des valeurs d'une ligne d'une relation.

Une relation est un peu une classe (programmation orientée objet) qui ne posséderait que des attributs et donc chaque instance représenterait un enregistrement.

**Clé primaire** : ensemble minimal de colonnes qui permet d'identifier de manière **unique** chaque enregistrement.



**Clé étrangère** : référence dans la majorité des cas une clé primaire d'une autre table.



Une personne est domiciliée dans une localité.

Une localité est habitée par plusieurs personnes.



Il existe deux grands types de liens : Un - Plusieurs (comme le précédent) et Plusieurs -

Plusieurs. La réalisation de ce dernier type de liens, un peu plus complexe, passe par l'utilisation d'une table intermédiaire dont la clé primaire est formée des clés étrangères des tables qu'elle relie.

### Conclusion :

- ✧ Une base de données est un outil qui stocke vos données de manière organisée et vous permet de les retrouver facilement par la suite.
- ✧ On communique avec MySQL grâce au langage SQL. Ce langage est commun à tous les systèmes de gestion de base de données (avec quelques petites différences néanmoins pour certaines fonctionnalités plus avancées).
- ✧ PHP fait l'intermédiaire entre vous et MySQL.
- ✧ Une base de données contient plusieurs tables.
- ✧ Chaque table est un tableau où les colonnes sont appelées "champs" et les lignes "entrées".

## 2 Administration avec l'outil phpMyAdmin

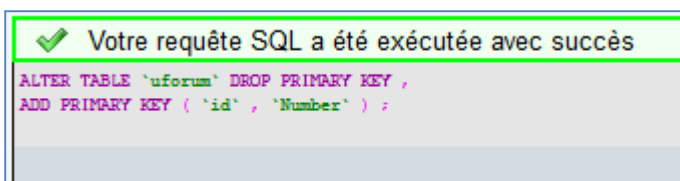
### 2.1 Présentation

L'outil phpMyAdmin est développé en PHP et offre une interface intuitive pour l'administration des bases de données du serveur.

Cet outil permet de :

- créer de nouvelles bases;
- créer/modifier/supprimer des tables;
- afficher/ajouter/modifier/supprimer des tuples dans des tables;
- effectuer des sauvegardes de la structure et/ou des données;
- effectuer n'importe quelle requête;
- gérer les privilèges des utilisateurs.

Toutes ces commandes sont disponibles via cette interface graphique, toutefois, phpMyAdmin vous montrera toujours la commande SQL qui sera effectuée :



Il est possible d'utiliser MySQL en mode "ligne de commande". Consultez les sites suivants pour plus d'information :

- ✧ [https://www.sqlfacile.com/apprendre\\_bases\\_de\\_donnees/executer\\_requete\\_mysql\\_en\\_ligne\\_de\\_commande](https://www.sqlfacile.com/apprendre_bases_de_donnees/executer_requete_mysql_en_ligne_de_commande)
- ✧ <https://tecfa.unige.ch/guides/tie/html/mysql-intro/mysql-intro-7.html>

## 2.2 Pourquoi utiliser la ligne de commande ?

- ✧ Les interfaces graphiques permettent de faire pas mal de choses, mais une fois que vous vous mettez à faire des choses subtiles et compliquées, il faudra obligatoirement écrire vous-même vos requêtes ;
- ✧ Il est fort probable que vous désiriez utiliser MySQL en combinaison avec un autre langage de programmation. Or, dans du code PHP (ou Java, ou Python, etc.), on ne va pas écrire "Ouvre PhpMyAdmin et clique sur le bon bouton pour que je puisse insérer une donnée dans la base". On va devoir écrire en dur les requêtes. Il faut donc que vous sachiez comment faire.
- ✧ un des principal intérêt du SQL est la **portabilité**. Cela veut dire qu'un logiciel qui utilise une base de données peut fonctionner avec n'importe quelle base de données. Il suffira de lui indiquer avec quelle base de données il doit dialoguer. *(si pour une raison X, on doit changer la base, il suffit de modifier la relation entre le logiciel et la base de données)*

## 3. Création d'une base de données

- ✧ Chaque instruction SQL se termine par un point-virgule, mais dans PHPmyAdmin, lors de l'entrée directe des instructions SQL, elle est facultative;
- ✧ Les instructions SQL ne sont pas sensibles à la casse.

```
CREATE DATABASE NomBaseDeDonnees ;
```

### 3.1 Identification et création de tables

Création d'une table :

```
CREATE [TEMPORARY ] TABLE [IF NOT EXISTS] [NomBase.]Nom_de_la_table
( Colonne1 Type1 [NOT NULL | NULL] [DEFAULT valeur1] [COMMENT 'chaîne1']
  Colonne2 Type2 [NOT NULL | NULL] [DEFAULT valeur2] [COMMENT 'chaîne2']
  Colonne3 Type3 [NOT NULL | NULL] [DEFAULT valeur3] [COMMENT 'chaîne3']
  ...
  [CONSTRAINT nomContrainte1 typeContrainte1]
);
```

- ✧ TEMPORARY : créer une table qui n'existera que durant la session courante

- ✧ IF NOT EXISTS : éviter qu'une erreur se produise si la table existe déjà
- ✧ Nom\_de\_la\_table : jusqu'à 64 caractères (sauf "/" "\" et ".")
- ✧ Colonnei typei : nom d'une colonne et son type (integer, char, date, ...)
- ✧ DEFAULT : fixe la valeur par défaut
- ✧ NOT NULL | NULL : indique que la valeur de la cellule soit NULL ou pas
- ✧ COMMENT : permet de commenter une colonne
- ✧ NomContrainte typeContrainte : nom de la contrainte et son type (clé primaire, clé étrangère, etc.)

Exemple :**Un seul champ**

Soit la création d'une table T\_Copain, pourvue d'un seul champ texte NomClient de taille 50

```
CREATE TABLE T_Copain (NomClient VARCHAR(50));
```

**2 champs**

```
CREATE TABLE T_Copain
(
  NomCopain varchar(30),
  Prenom  varchar(20)
);
```

- ✧ Il est possible d'écrire sur plusieurs lignes pour rendre le code SQL plus clair
- ✧ Les champs sont séparés par des virgules

**Avec une valeur par défaut, et l'interdiction de laisser un champ NULL**

```
CREATE TABLE T_Copain
(
  NomCopain VARCHAR (20) NOT NULL,
  Pays  VARCHAR (20) DEFAULT 'Belgique'
);
```

**Avec une clé primaire et un auto-incrément**

L'exemple qui suit crée une table T\_Copain, pourvue d'une clé primaire IDCopain, qui s'auto-incrémente. Utilisation de la fonction DATETIME dans le champ DateCreation (valeur par défaut dynamique)

```
CREATE TABLE T_Copain
(
  IDCopain INT(11) auto_increment,
  NomCopain VARCHAR(20),
  DateCreation DATETIME,
  PRIMARY KEY (IDCopain)
);
```

### Cumul d'options de création de champs

```
CREATE TABLE T_Copain
(
  IDCopain INT(11) auto_increment PRIMARY KEY,
  NomCopain VARCHAR(20),
  DateCreation DATETIME
)
```

#### Remarque :

- ✧ Il est recommandé de préfixer par ID (ou pk\_ pour primary key) les contraintes de clé primaires (IDRef (ou fk\_ foreign key) les clés étrangères).
- ✧ Les mots SQL seront par convention tapés en majuscule, et les noms des champs/tables en minuscule sauf la première lettre.
- ✧ PRIMARY KEY = Unique + NOT NULL + Index

## 3.2 Identification des champs et leurs types de données

### 3.2.1 Types des attributs

Les propriétés de vos objets peuvent être de types très différents :

- Nombre entier signé ou non (température, quantité commandée, âge);
- Nombre à virgule (prix, taille);
- Chaîne de caractères (nom, adresse, article de presse);
- Date et heure (date de naissance, heure de parution);
- Énumération (une couleur parmi une liste prédéfinie);
- Ensemble (une ou des monnaies parmi une liste prédéfinie).

Il s'agit de choisir le plus adapté à vos besoins.

Ces types requièrent une plus ou moins grande quantité de données à stocker. Par exemple, ne pas choisir un LONGTEXT pour stocker un prénom mais plutôt un VARCHAR(40) !

### 3.2.2 Entiers

nom	borne inférieure	borne supérieure
<b>TINYINT</b>	-128	127
<b>TINYINT UNSIGNED</b>	0	255
<b>SMALLINT</b>	-32768	32767
<b>SMALLINT UNSIGNED</b>	0	65535
<b>MEDIUMINT</b>	-8388608	8388607
<b>MEDIUMINT UNSIGNED</b>	0	16777215
<b>INT*</b>	-2147483648	2147483647
<b>INT* UNSIGNED</b>	0	4294967295
<b>BIGINT</b>	-9223372036854775808	9223372036854775807
<b>BIGINT UNSIGNED</b>	0	18446744073709551615

(\*) : **INT** est un synonyme de **INTEGER**.

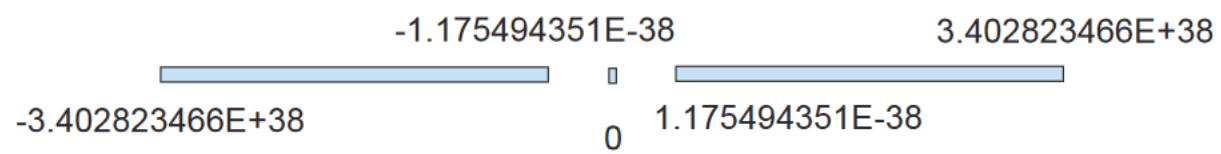
**UNSIGNED** permet d'avoir un type non signé.

**ZEROFILL** : remplissage des zéros non significatifs.

### 3.2.3 Flottants

Les flottants – dits aussi nombres réels – sont des nombres à virgule. Contrairement aux entiers, leur domaine n'est pas continu du fait de l'impossibilité de les représenter avec une précision absolue.

Exemple du type **FLOAT** :



nom	domaine négatif : borne inférieure borne supérieure	Domaine positif : borne inférieure borne supérieure
<b>FLOAT</b>	-3.402823466E+38 -1.175494351E-38	1.175494351E-38 3.402823466E+38
<b>DOUBLE*</b>	-7.976931348623157E+308 -2.2250738585072014E-308	2.2250738585072014E-308 1.7976931348623157E+308

(\*) : **DOUBLE** est un synonyme de **REAL**.

### 3.2.4 Chaînes

nom	longueur
<b>CHAR(M)</b>	Chaîne de taille fixée à M, où $1 < M < 255$ , complétée avec des espaces si nécessaire.
<b>CHAR(M) BINARY</b>	Idem, mais insensible à la casse lors des tris et recherches.
<b>VARCHAR(M)</b>	Chaîne de taille variable, de taille maximum M, où $1 < M < 255$ , complété avec des espaces si nécessaire.
<b>VARCHAR(M) BINARY</b>	Idem, mais insensible à la casse lors des tris et recherches.
<b>TINYTEXT</b>	Longueur maximale de 255 caractères.
<b>TEXT</b>	Longueur maximale de 65535 caractères.
<b>MEDIUMTEXT</b>	Longueur maximale de 16777215 caractères.
<b>LONGTEXT</b>	Longueur maximale de 4294967295 caractères.
<b>DECIMAL(M,D)</b>	Simule un nombre flottant de D chiffres après la virgule et de M chiffres au maximum. Chaque chiffre ainsi que la virgule et le signe moins (pas le plus) occupent un caractère.

#### Remarques :

"Les types `CHAR` et `VARCHAR` sont similaires, mais diffèrent dans la manière dont ils sont stockés et récupérés.

La longueur d'une colonne `CHAR` est fixée à la longueur que vous avez défini lors de la création de la table. La longueur peut être n'importe quelle valeur entre 1 et 255. (Dans la version 3.23 de MySQL, la longueur est comprise entre 0 et 255.) Quand une valeur `CHAR` est enregistrée, elle est complétée à droite avec des espaces jusqu'à atteindre la valeur fixée. Quand une valeur de `CHAR` est lue, les espaces en trop sont retirés.

Les valeurs contenues dans les colonnes de type `VARCHAR` sont de tailles variables. Vous pouvez déclarer une colonne `VARCHAR` pour que sa taille soit comprise entre 1 et 255, exactement comme pour les colonnes `CHAR`. Par contre, contrairement à `CHAR`, les valeurs de `VARCHAR` sont stockées en utilisant autant de caractères que nécessaire, plus un octet pour mémoriser la longueur. Les valeurs ne sont pas complétées. Au contraire, les espaces finaux sont supprimés avant stockage (ce qui ne fait pas partie des spécifications ANSI SQL).

Si vous assignez une chaîne de caractères qui dépasse la capacité de la colonne `CHAR` ou `VARCHAR`, celle-ci est tronquée jusqu'à la taille maximale du champ."

**NUMERIC** est un synonyme de **DECIMAL**.

Les types TINYTEXT, TEXT, MEDIUMTEXT et LONGTEXT peuvent être judicieusement remplacés respectivement par TINYBLOB, BLOB, MEDIUMBLOB et LONGBLOB.

Ils ne diffèrent que par la sensibilité à la casse qui caractérise la famille des BLOB. Alors que la famille des TEXT est insensible à la casse lors des tris et recherches.

Les BLOB peuvent être utilisés pour stocker des données binaires.

Les VARCHAR, TEXT et BLOB sont de taille variable. Alors que les CHAR et DECIMAL sont de taille fixe.

### 3.2.5 Dates et heures

nom	description
<b>DATE</b>	Date au format anglophone AAAA-MM-JJ.
<b>DATETIME</b>	Date et heure au format anglophone AAAA-MM-JJ HH:MM:SS.
<b>TIMESTAMP</b>	Affiche la date et l'heure sans séparateur : AAAAMMJJHHMMSS.
<b>TIMESTAMP(M)</b>	Idem mais M vaut un entier pair entre 2 et 14. Affiche les M premiers caractères de TIMESTAMP.
<b>TIME</b>	Heure au format HH:MM:SS.
<b>YEAR</b>	Année au format AAAA.

nom	description
<b>TIMESTAMP(2)</b>	AA
<b>TIMESTAMP(4)</b>	AAMM
<b>TIMESTAMP(6)</b>	AAMMJJ
<b>TIMESTAMP(8)</b>	AAAAMMJJ
<b>TIMESTAMP(10)</b>	AAMMJJHHMM
<b>TIMESTAMP(12)</b>	AAMMJJHHMMSS
<b>TIMESTAMP(14)</b>	AAAAMMJJHHMMSS

En cas d'insertion d'un enregistrement en laissant vide un attribut de type TIMESTAMP, celui-ci prendra automatiquement la date et heure de l'insertion.

Contrairement à Unix (où le timestamp est le nombre de secondes écoulées depuis le 1er janvier 1970), en

### 3.2.6 Ensembles

nom	description
<b>ENUM('valeur','valeur2',...)</b>	Une énumération <code>ENUM</code> est une chaîne dont la valeur est choisie parmi une liste de valeurs autorisées lors de la création de la table. (65535 valeurs max.)
<b>SET('valeur','valeur2',...)</b>	Un <code>SET</code> est une chaîne qui peut avoir zéro ou plusieurs valeurs, chacune doit être choisie dans une liste de valeurs définies lors de la création de la table. (64 valeurs max.)

### 3.3 Création des relations entre les tables

#### 3.3.1 Clé primaire

Toute table doit posséder un champ qui joue le rôle de clé primaire. La clé primaire permet d'identifier de manière unique une entrée dans la table. Par exemple, chaque article de votre site Internet doit pouvoir être identifié de manière unique. Le moyen le plus simple pour cela est de lui donner un numéro unique, dans un champ nommé "id". **Il ne peut pas y avoir deux articles avec le même id** – si deux articles ont le même numéro, on ne pourra pas les différencier !

Exemples de clé primaire : Numéro de registre national, numéro ISBN, ADN, etc...

#### 3.3.2 Attribut non nul

Considérons que l'on souhaite que certains attributs aient obligatoirement une valeur. On utilisera l'option **NOT NULL**.

Dans ce cas, si malgré tout, aucune valeur n'est fournie, la valeur par défaut – si elle est déclarée à la création de la relation – sera automatiquement affectée à cet attribut dans l'enregistrement.

Si aucune valeur par défaut n'est déclarée :

- ✧ la chaîne vide " " sera affectée à l'attribut s'il est de type chaîne de caractères;
- ✧ la valeur zéro 0 s'il est de type nombre;
- ✧ la date nulle 0000-00-00 et/ou l'heure nulle 00:00:00 s'il est de type date, heure ou date et heure.

Exemple :

```
adresse TINYTEXT NOT NULL
```

Au contraire, on utilisera l'option **NULL** si on autorise l'absence de valeur.

#### 3.3.3 Valeur par défaut

Pour donner une valeur par défaut à un attribut, on utilise l'option **DEFAULT**.

Lors de l'ajout d'un enregistrement cette valeur sera affectée à l'attribut si aucune valeur n'est donnée.

Exemple :

```
'telephone' DECIMAL(10,0) DEFAULT '0123456789'
```

Les attributs de type chaîne de caractères de la famille TEXT et BLOB ne peuvent pas avoir de valeur par défaut.



### 3.3.4 Attribut sans doublon

Pour interdire l'apparition de doublon pour un attribut, on utilise l'option **UNIQUE**.

Syntaxe :

**UNIQUE** [nomdelacontrainte](liste des attributs)

Exemple, pour interdire tout doublon de l'attribut *nom* :

**UNIQUE**(*nom*)

Pour interdire les doublons sur l'attribut *nom* mais les interdire aussi sur '*prénom*', tout en les laissant indépendants :

**UNIQUE**(*nom*)

**UNIQUE**(*prénom*)

nom	prénom
Dupond	Marc
Dupont	Pierre
Martin	Marc

Enregistrement interdit car 'Marc' est un doublon dans la colonne 'prénom'

Pour interdire tout doublon à un ensemble d'attributs (tuple), on passe en paramètre à **UNIQUE** la liste des attributs concernés.

Pour interdire tout doublon du couple (*nom*, *prénom*) :

**UNIQUE**(*nom*,*prénom*)

nom	prénom
Dupond	Marc
Dupont	Pierre
Martin	Marc
Martin	Pierre
Martin	Marc

Enregistrement interdit car le couple ('Martin', 'Marc') est un doublon du couple (nom,prénom)

### 3.3.5 Index

Lors de la recherche d'informations dans une relation, MySQL parcourt la table correspondante dans n'importe quel ordre. Dans le cas d'un grand nombre de lignes, cette recherche est très très longue du fait du parcours de TOUTE la table.

Pour y remédier, une optimisation possible et FORTEMENT recommandée, est d'utiliser des index.

La création d'un index associé à un attribut ou à un ensemble ordonné d'attributs va créer une liste ordonnée des valeurs de ces attributs et de l'adresse de la ligne associée. C'est sur les valeurs de cette liste que se feront les recherches et les tris. Les algorithmes de recherche et de tri sur des ensembles ordonnés sont énormément plus rapides !

Ainsi, d'une recherche à coût prohibitif, on passe à une recherche sur un ensemble déjà trié. On gagne donc énormément en temps d'accès aux informations. Bien que cela ralentisse les mises à jour (insertion, suppression, modification de clé).

On choisira de créer des index sur les attributs qui seront les plus sollicités par les recherches ou utilisés comme critère de jointure. Par contre, on épargnera les attributs qui contiennent peu de valeurs différentes les uns des autres et ceux dont les valeurs sont très fréquemment modifiées.

Syntaxe :

**INDEX index (liste des attributs)**

Exemple, pour créer un index sur les 3 premiers caractères seulement de l'attribut *nom* :

**INDEX idx\_nom (nom(3))**

Exemple, pour créer un index sur le couple (*nom*,*prenom*) :

**INDEX idx\_nom\_prenom (nom,prenom)**

- ✧ Un index peut porter sur 15 colonnes maximum.
- ✧ Une table peut posséder au maximum 16 index.
- ✧ Un index peut avoir une taille d'au maximum 256 octets et ne doit porter que sur des attributs NOT NULL.

### 3.3.6 Exercices – création de table

- ✧ Créez un système d' « article » pour votre site. Les informations relatives à un article sont les suivantes : titre, texte, date de parution, auteur, rubrique (soit "économie", "sports", "international", "politique" ou "culture")
- ✧ Créez une table "T\_contact" qui contiendrait toutes les informations relatives aux personnes de votre carnet d'adresses. Quels champs seraient-ils utiles d'indexer ?

## 4. Gestion du contenu de la base de données

### 4.1 Ajout de données

```
INSERT [LOW_PRIORITY | DELAYED | HIGH_PRIORITY] [IGNORE]
[INTO] [nomBase.] { nomTable | nomVue } [(nomColonne,...)]
VALUES ({expression | DEFAULT},...),(...),...
[ON DUPLICATE KEY UPDATE nomColonne = expression,...]
```

- ✧ **DELAYED** indique que l'insertion est différée (si la table est modifiée par ailleurs, le serveur attend qu'elle se libère pour y insérer périodiquement de nouveaux enregistrements si elle redevient active entre-temps).
- ✧ **LOW\_PRIORITY** indique que l'insertion est différée à la libération complète de la table (option à ne pas utiliser sur des tables `MyISAM`).
- ✧ **HIGH\_PRIORITY** annule l'option *low priority* du serveur.
- ✧ **IGNORE** indique que les éventuelles erreurs déclenchées suite à l'insertion seront considérées en tant que *warnings*.
- ✧ **ON DUPLICATE KEY UPDATE** permet de mettre à jour l'enregistrement présent dans la table, qui a déclenché l'erreur de doublon (dans le cas d'un index `UNIQUE` ou d'une clé primaire). Dans ce cas le nouvel enregistrement n'est pas inséré, seul l'ancien est mis à jour.

Ajouter un enregistrement à une relation revient à ajouter une ligne à la table. Pour cela, pour chacun des attributs, il faudra en préciser la valeur. Si certaines valeurs sont omises, alors les valeurs par défaut définies lors de la création de la relation seront utilisées. Si on ne dispose pas non plus de ces valeurs par défaut, alors MySQL mettra 0 pour un nombre, "" pour une chaîne, 0000-00-00 pour une date, 00:00:00 pour une heure, 0000000000000000 pour un timestamp (si la contrainte NOT NULL est présente). Dans le cas où l'attribut porte la contrainte NULL (par défaut) alors la valeur par défaut de l'attribut – quel soit son type – sera la suivante : NULL.

Syntaxe d'une "insertion étendue" :

```
INSERT INTO Table(liste des attributs) VALUES(liste des valeurs)
```

Exemple :

```
INSERT INTO Personnes(nom,prenom) VALUES('Martin','Paolo')
```

REPLACE est un synonyme de INSERT, mais sans doublon. (Respect des contraintes d'unicité (UNIQUE, PRIMARY KEY).

Une syntaxe plus courte mais plus ambiguë permet d'insérer un enregistrement dans une table. Elle consiste à omettre la liste des noms d'attribut à la suite du nom de la relation. Cela impose

que la liste des valeurs suivant le mot clé VALUES soit exactement celle définie dans la table et qu'elle soit dans l'ordre défini dans la définition de la table ; sinon des erreurs se produiront.

Syntaxe d'une "insertion standard" :

**INSERT INTO *table* VALUES(liste exhaustive et ordonnée des valeurs)**

Exemple :

```
CREATE TABLE Ballon (  
    taille INT(4) NOT NULL,  
    couleur VARCHAR(40)  
)
```

```
INSERT INTO Ballon VALUES(20, 'rouge')           →ok
```

```
INSERT INTO Ballon VALUES('rouge', 20)           →faux
```

```
INSERT INTO Ballon VALUES('rouge')               →faux
```

Syntaxe d'une "insertion complète ":

**INSERT INTO relation VALUES (liste des valeurs), (liste d'autres valeurs), (liste d'encore d'autres valeurs), ...**

Exemple :

```
INSERT INTO Ballon VALUES (20, 'rouge'), (35, 'vert fluo'), (17, 'orange'), (28,  
'mauve')
```

Cet exemple est équivalent aux requêtes suivantes :

```
INSERT INTO Ballon VALUES(20, 'rouge')  
INSERT INTO Ballon VALUES(35, 'vert fluo')  
INSERT INTO Ballon VALUES(17, 'orange')  
INSERT INTO Ballon VALUES(28, 'mauve')
```

## 4.2 Modification de données

**UPDATE** [LOW\_PRIORITY] [IGNORE] [*nomBase.*] *nomTable*

SET *col\_name1=expr1* [, *col\_name2=expr2 ...*]

SET *colonne1 = expression1* | (*requête\_SELECT*) | DEFAULT

[*,colonne2 = expression2...*]

[WHERE (*condition*)]

[ORDER BY *listeColonnes*]

[LIMIT *nbreLimite*]

- ✧ **LOW\_PRIORITY** indique que la modification est différée à la libération complète de la table (option à ne pas utiliser sur des tables **MyISAM**).
- ✧ **IGNORE** signifie que les éventuelles erreurs déclenchées suite aux modifications seront considérées en tant que *warnings*.
- ✧ La clause **SET** actualise une colonne en lui affectant une expression (valeur, valeur par défaut, calcul ou résultat d'une requête).
- ✧ La condition du **WHERE** filtre les lignes à mettre à jour dans la table. Si aucune condition n'est précisée, tous les enregistrements seront actualisés. Si la condition ne filtre aucune ligne, aucune mise à jour ne sera réalisée.
- ✧ **ORDER BY** indique l'ordre de modification des colonnes.

**LIMIT** spécifie le nombre maximum d'enregistrements à changer (par ordre de clé primaire croissante). Permet de n'appliquer la commande qu'aux **nbreLimite** premiers enregistrements satisfaisant la condition définie par **WHERE**.

Pour modifier un ou des enregistrement(s) d'une relation, il faut donc préciser un critère de sélection des enregistrements à modifier (clause **WHERE**), il faut aussi dire quels sont les attributs dont on va modifier la valeur et quelles sont ces nouvelles valeurs (clause **SET**).

Exemple :

```
UPDATE Personnes SET téléphone='0156281469' WHERE nom='Martin' AND prénom = 'Pierre'
```

Cet exemple modifie le numéro de téléphone de Martin Pierre.

Il est possible de modifier les valeurs d'autant d'attributs que la relation en contient.

Exemple pour modifier plusieurs attributs :

```
UPDATE Personnes SET téléphone='0156281469', fax='0156281812' WHERE id = 102
```

Pour appliquer la modification à tous les enregistrements de la relation, il suffit de ne pas mettre de clause **WHERE**.

Autre exemple :

```
UPDATE Produits SET Prix=Prix*0.15
```

Il est donc possible de modifier la valeur d'un attribut relativement à sa valeur déjà existante.

### 4.3 Suppression de données

```
DELETE [LOW_PRIORITY] [QUICK] [IGNORE] FROM [nomBase.] nomTable
[WHERE (condition)]
[ORDER BY listeColonnes]
[LIMIT nbreLimite]
```

- ✧ LOW\_PRIORITY, IGNORE et LIMIT ont la même signification que pour UPDATE.
- ✧ QUICK (pour les tables de type MyISAM) ne met pas à jour les index associés pour accélérer le traitement.
- ✧ La condition du WHERE sélectionne les lignes à supprimer dans la table. Si aucune condition n'est précisée, toutes les lignes seront détruites. Si la condition ne sélectionne aucune ligne, aucun enregistrement ne sera supprimé.
- ✧ ORDER BY réalise un tri des enregistrements qui seront effacés dans cet ordre.

#### **Attention, la suppression est définitive !**

Exemple :

```
DELETE FROM Personnes WHERE nom='Martin' AND prénom='Marc'
```

→ Effacement de toutes les personnes qui s'appellent Martin Marc

```
DELETE FROM Personnes WHERE id="154"
```

→ Effacement uniquement de la personne dont l'ID est 154

Pour vider une table de tous ces éléments, ne pas mettre de clause WHERE. Cela efface et recrée la table, au lieu de supprimer un à un chacun des enregistrements de la table (ce qui serait très long).

Exemple :

```
DELETE FROM Personnes
```

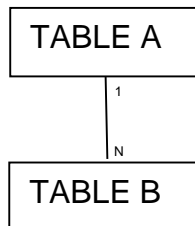
#### 4.4 Exercices – insertion de données

Insérez en sql les personnes suivantes dans la table « Contact » créée à l'exercice précédent :

titre	nom	prenom	Adresse	Numéro	CP	Ville	gsm	téléphone	email
Mr	AUBOISDORMANT	Abel	Rue des arbres	12	1300	Wavre	0479/794513	010/212121	aa@yahoo.fr
Dr	ZIEUVAIR	Bruno	Clos de l'Armoise	1323	1300	Wavre	0479/784513	010/212122	zb@yahoo.fr

### 5. Intégrité référentielle

La **normalisation** correspond au processus d'organiser ses données afin de limiter les redondances, divisant une table en plusieurs, et en les reliant entre elles par des clefs primaires et étrangères. L'objectif est d'isoler les données afin que l'ajout, l'effacement ou la modification d'un champ puisse se faire sur une seule table, et se propager au reste de la base par le biais des relations.

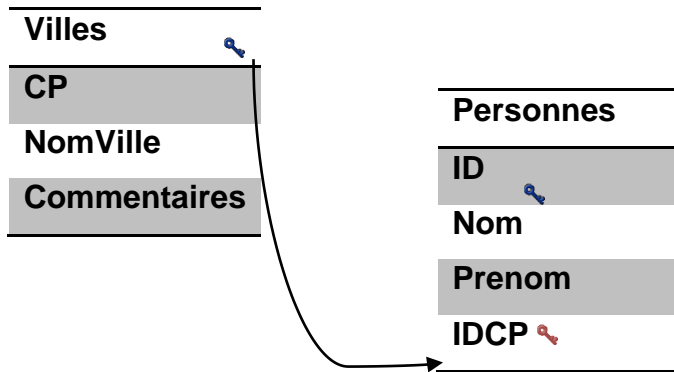


1. Clé primaire : identifiant unique et non nul
2. On ne peut pas ajouter un élément B pour un A inexistant
3. On ne peut pas supprimer un élément A s'il lui correspond un ou plusieurs éléments B

```
[CONSTRAINT nomContrainte] FOREIGN KEY [id] (listeColonneEnfant)  
REFERENCES nomTable (listeColonneParent)  
[ON DELETE {RESTRICT | CASCADE | SET NULL | NO ACTION}]  
[ON UPDATE {RESTRICT | CASCADE | SET NULL | NO ACTION}]
```

- ✧ La cohérence du "fils" vers le "père" : on ne doit pas pouvoir insérer un enregistrement "fils" (ou modifier sa clé étrangère) rattaché à un enregistrement "père" inexistant. Il est cependant possible d'insérer un "fils" (ou de modifier sa clé étrangère) sans rattacher d'enregistrement "père", à la condition qu'il n'existe pas de contrainte `NOT NULL` au niveau de la clé étrangère.
- ✧ La cohérence du "père" vers le "fils" : on ne doit pas pouvoir supprimer un enregistrement "père" si un enregistrement "fils" y est encore rattaché. Il est possible de supprimer les "fils" associés (`DELETE CASCADE`), d'affecter la valeur nulle aux clés étrangères des "fils" associés (`DELETE SET NULL`) ou de répercuter une modification de la clé primaire du père (`UPDATE CASCADE` et `UPDATE SET NULL`).

Exemple :



```
CREATE TABLE Villes(
  CP INT(4) PRIMARY KEY,
  NomVille VARCHAR(80),
  Commentaires TEXT
);
```

```
CREATE TABLE Personnes(
  ID INT(11) auto_increment PRIMARY KEY,
  Nom VARCHAR (40),
  Prenom VARCHAR(40),
  IDCP INT(4),
  CONSTRAINT FK_CP FOREIGN KEY (IDCP)
  REFERENCES Villes (CP));
```

```
INSERT INTO Villes VALUES(1300, 'Wavre','');
INSERT INTO Villes VALUES(1342, 'Limelette','');
INSERT INTO Villes VALUES(1340, 'Ottignies','');
INSERT INTO Villes VALUES(1348, 'LLN','');
INSERT INTO Villes VALUES(1330, 'Rixensart','');
```

```
INSERT INTO Personnes VALUES(NULL,'Martin','Bruno',1342); → ok
```

```
INSERT INTO Personnes VALUES(NULL,'Martin','Charles',1301); → erreur
```

#1452 - Cannot add or update a child row: a foreign key constraint fails



```
"CONSTRAINT FK_CP FOREIGN KEY (IDCP)
REFERENCES Villes (CP)
ON DELETE NO ACTION ON UPDATE NO ACTION"
```

Si tentative de suppression d'une ville → "foreign key constraint fails"

```
"CONSTRAINT FK_CP FOREIGN KEY (IDCP)
REFERENCES Villes (CP)
ON DELETE CASCADE ON UPDATE CASCADE"
```

Si tentative de suppression d'une ville → suppression de toutes les personnes

Si tentative de modification d'une ville (code postal par exemple) → mise à jour automatique dans la table personnes.

```
"CONSTRAINT FK_CP FOREIGN KEY (IDCP)
REFERENCES Villes (CP)
ON DELETE SET NULL"
```

Si tentative de suppression d'une ville → Affecter la valeur nulle dans la table personnes

## Résumé

Instructions	Table "père"	Table "fils"
<b>INSERT</b>	Correcte si la clé primaire est unique.	Correcte si la clé étrangère est référencée dans la table "père" ou est nulle.
<b>UPDATE</b>	Correcte si l'instruction ne laisse pas d'enregistrements dans la table "fils" ayant une clé étrangère non référencée.	Correcte si la nouvelle clé étrangère référence un enregistrement "père" existant.
<b>DELETE</b>	Correcte si aucun enregistrement de la table "fils" ne référence le ou les enregistrements détruits.	Correcte sans condition.
<b>DELETE Cascade</b>	Correcte s'il n'y a pas de NOT NULL dans la table "fils".	Sans objet.
<b>DELETE SET NULL</b>	Correcte sans condition.	Sans objet.
<b>UPDATE Cascade</b>	Correcte sans condition.	Sans objet.
<b>UPDATE SET NULL</b>	Correcte s'il n'y a pas de NOT NULL dans la table "fils".	Sans objet.

### 5.1 Exercices – Analyse/ Intégrité référentielle

- ✧ Réalisez les tables qui permettraient de normaliser le tableau suivant. (L'idée est de ne pas "répéter" la fonction pour chaque employé – et par exemple de pouvoir ajouter des informations relatives à la fonction (barème, descriptif, etc.) sans devoir répéter ces informations pour chaque personne).

N°	Nom	Prénom	Fonction
1	DURANT	Louis	Commercial
2	DURAND	Pierrette	Webmaster
3	MARÉCHAL	Juan	Webmaster
4	MICHEL	Louissette	Commercial
5	TLUAUP	Yves	PDG
6	BACH	Gérard	Graphiste
7	DELLILE	Georgette	Graphiste
8	DELAGRANGE	Maurice	Comptable

✧ Réalisez les tables qui contiendraient toutes les informations relatives aux avions ainsi qu'à leurs vols

**AVIONS**

AviID	AviModel	AviNombreDePlaces	AviLocalite
1	A300	300	NICE
2	A310	300	NICE
3	B707	250	PARIS
4	A300	280	LYON
5	CONCORDE	160	NICE
6	B747	460	PARIS
7	B707	250	PARIS
8	A310	300	TOULOUSE
9	MERCURE	180	LYON
10	CONCORDE	160	PARIS

**VOLS**

VolID	VolDate	VolDestinationDepart	VolDestinationArrivé	VolHeureDepart	VoleHeureArrivé	VolAviID
1	2013-10-30	NICE	TOULOUSE	11h00	12h30	1
2	2013-10-30	PARIS	TOULOUSE	17h00	18h30	8
3	2013-10-30	TOULOUSE	LYON	14h00	16h00	1
4	2013-10-30	TOULOUSE	LYON	18h00	20h00	3
5	2013-10-30	PARIS	NICE	06h45	08h15	1
6	2013-10-30	LYON	NICE	11h00	12h00	2
7	2013-10-30	PARIS	LYON	08h00	09h00	4
8	2013-10-30	NICE	PARIS	07h15	08h45	4
9	2013-10-31	NANTES	LYON	09h00	15h30	8
10	2013-10-31	NICE	PARIS	12h15	13h45	2
11	2013-10-31	PARIS	LYON	15h00	16h00	2
12	2013-10-31	LYON	NANTES	16h30	20h00	2
13	2013-10-31	NICE	LENS	11h00	14h00	5
14	2013-10-31	LENS	PARIS	15h00	16h00	5
15	2013-10-31	PARIS	TOULOUSE	17h00	18h00	9
16	2013-10-31	PARIS	TOULOUSE	18h00	19h00	5

## 6. Mise à jour d'une base de données

### 6.1 Modification d'une table

#### ALTER TABLE

Il est possible de modifier la définition d'une table par la suite, à tout moment par la commande **ALTER TABLE**.

Voici ce qu'il est possible de réaliser :

- ajouter/supprimer un attribut
- créer/supprimer une clé primaire
- ajouter une contrainte d'unicité (interdire les doublons)
- changer la valeur par défaut d'un attribut
- changer totalement la définition d'un attribut
- changer le nom de la relation
- ajouter/supprimer un index

#### 6.1.1 Ajouter un attribut

Syntaxe :

```
ALTER TABLE relation ADD definition [ FIRST | AFTER attribut]
```

Ajoutons l'attribut *fax* qui est une chaîne représentant un nombre de 10 chiffres:

```
ALTER TABLE Personnes ADD fax DECIMAL(10,0)
```

Nous aurions pu forcer la place où doit apparaître cet attribut. Pour le mettre en tête de la liste des attributs de la relation, il faut ajouter l'option **FIRST** en fin de commande. Pour le mettre après l'attribut '*téléphone*', il aurait fallu ajouter **AFTER** '*téléphone*'.

**Note** : il ne peut pas déjà avoir dans la relation un attribut du même nom !

#### 6.1.2 Supprimer un attribut

Attention, supprimer un attribut implique la suppression des valeurs qui se trouvent dans la colonne qui correspond à cet attribut.

Syntaxe :

```
ALTER TABLE relation DROP attribut
```


Exemple :

```
ALTER TABLE Personnes DROP 'prenom'
```

La suppression d'un attribut peut incidemment provoquer des erreurs sur les contraintes clé primaire (**PRIMARY KEY**) et unique (**UNIQUE**).

```
CREATE TABLE Personne (
    id SMALLINT UNSIGNED PRIMARY KEY AUTO_INCREMENT,
    nom VARCHAR(40),
    'prénom' VARCHAR(40),
    adresse TINYTEXT,
    'téléphone' DECIMAL(10,0),
    UNIQUE(nom, 'prénom')
)
```

ALTER TABLE *Personnes* DROP '*prénom*'



<i>Nom</i>	<i>Prénom</i>
<b>Dupond</b>	Marc
<b>Martin</b>	Marc
<b>Martin</b>	Pierre

<i>Nom</i>
<b>Dupond</b>
<b>Martin</b>
<b>Martin</b>

Refus d'opérer la suppression, car cela contredirait la contrainte d'unicité qui resterait sur l'attribut *nom*.

### 6.1.3 Créer une clé primaire

La création d'une clé primaire n'est possible qu'en l'absence de clé primaire dans la relation.

Syntaxe :

```
ALTER TABLE relation ADD PRIMARY KEY (attribut)
```

Exemple :

```
ALTER TABLE Personnes ADD PRIMARY KEY (nom, prénom)
```

### 6.1.4 Supprimer une clé primaire

Comme une clé primaire est unique, il n'y a aucune ambiguïté lors de la suppression.

Syntaxe :

```
ALTER TABLE relation DROP PRIMARY KEY
```

Exemple :

```
ALTER TABLE Personnes DROP PRIMARY KEY
```

S'il n'y a aucune clé primaire lorsque cette commande est exécutée, aucun message d'erreur ne sera généré, la commande sera simplement ignorée.

### 6.1.5 Ajout d'une contrainte d'unicité

Il est possible (facultatif) de donner un nom à la contrainte.

Cette contrainte peut s'appliquer à plusieurs attributs.

Si les valeurs déjà présentes dans la relation sont en contradiction avec cette nouvelle contrainte, alors cette dernière ne sera pas appliquée et une erreur sera générée.

Syntaxe :

```
ALTER TABLE relation ADD UNIQUE [contrainte] (attributs)
```

Exemple pour interdire tout doublon sur l'attribut *fax* de la relation **Personnes** :

```
ALTER TABLE Personnes ADD UNIQUE u_fax (fax)
```

Autre exemple fictif :

```
ALTER TABLE Moto ADD UNIQUE u_coul_vitre (couleur,vitre)
```

### 6.1.6 Changer la valeur par défaut d'un attribut

Pour changer ou supprimer la valeur par défaut d'un attribut.

Attention aux types qui n'acceptent pas de valeur par défaut (**BLOB** et **TEXT**).

Syntaxe :

```
ALTER TABLE relation ALTER attribut { SET DEFAULT valeur | DROP DEFAULT }
```

Changer sa valeur par défaut :

```
ALTER TABLE Personnes ALTER 'téléphone' SET DEFAULT '9999999999'
```

Supprimer sa valeur par défaut :

```
ALTER TABLE Personnes ALTER 'téléphone' DROP DEFAULT
```

Le changement ou la suppression n'affecte en rien les enregistrements qui ont eu recours à cette valeur lors de leur insertion.

### 6.1.7 Changer la définition d'un attribut

Pour changer la définition de l'attribut sans le renommer :

```
ALTER TABLE relation MODIFY attribut definition_relative
```

Exemple 1 :

```
ALTER TABLE Personnes MODIFY fax VARCHAR(14)
```

Pour changer sa définition en le renommant :

```
ALTER TABLE relation CHANGE attribut definition_absolue
```

Exemple 2 :

```
ALTER TABLE Personnes CHANGE fax num_fax VARCHAR(14)
```

Attention, si le nouveau type appliqué à l'attribut est incompatible avec les valeurs des enregistrements déjà présents dans la relation, alors les valeurs risquent d'être modifiées ou remises à zéro !

### 6.1.8 Changer le nom d'une relation

Syntaxe :

```
ALTER TABLE relation RENAME nouveau_nom
```

Exemple :

```
ALTER TABLE Personnes RENAME Carnet
```

Cela consiste à renommer la table, et donc le fichier qui la stocke.

### 6.1.9 Ajouter un index

Une table ne peut comporter que 32 index.

Et un index ne peut porter que sur 16 attributs maximum à la fois.

Syntaxe :

```
ALTER TABLE relation ADD INDEX index (attributs)
```

Exemple :

```
ALTER TABLE Personnes ADD INDEX nom_complet (nom,prénom)
```

Dans cet exemple, on a ajouté à la relation **Personnes** un index que l'on nomme *nom\_complet* et qui s'applique aux deux attributs '*nom*' et '*prénom*'. Ainsi, les recherches et les tris sur les attributs '*nom*' et '*prénom*' seront grandement améliorés. Car un index apporte les changements sous-jacents permettant d'optimiser les performances du serveur de base de données.

### Remarques :

Après la suppression de grandes parties d'une table contenant des index, les index des tuples supprimés sont conservés, rallongeant d'autant les sélections. Pour supprimer ces index obsolètes et vider les "trous", il faut l'optimiser.

Syntaxe :

```
OPTIMIZE TABLE Relation
```

Exemple :

```
OPTIMIZE TABLE Personnes
```



### 6.1.10 Supprimer un index

Syntaxe :

```
ALTER TABLE relation DROP INDEX index
```

Exemple :

```
ALTER TABLE Personnes DROP INDEX nom_complet
```

Cet exemple permet de supprimer l'index nommé *nom\_complet* de la relation **Personnes**.

### 6.2 Suppression d'une table

**DROP TABLE**

La commande **DROP TABLE** prend en paramètre le nom de la table à supprimer. Toutes les données qu'elle contient sont supprimées et sa définition aussi.

Syntaxe :

```
DROP TABLE relation
```

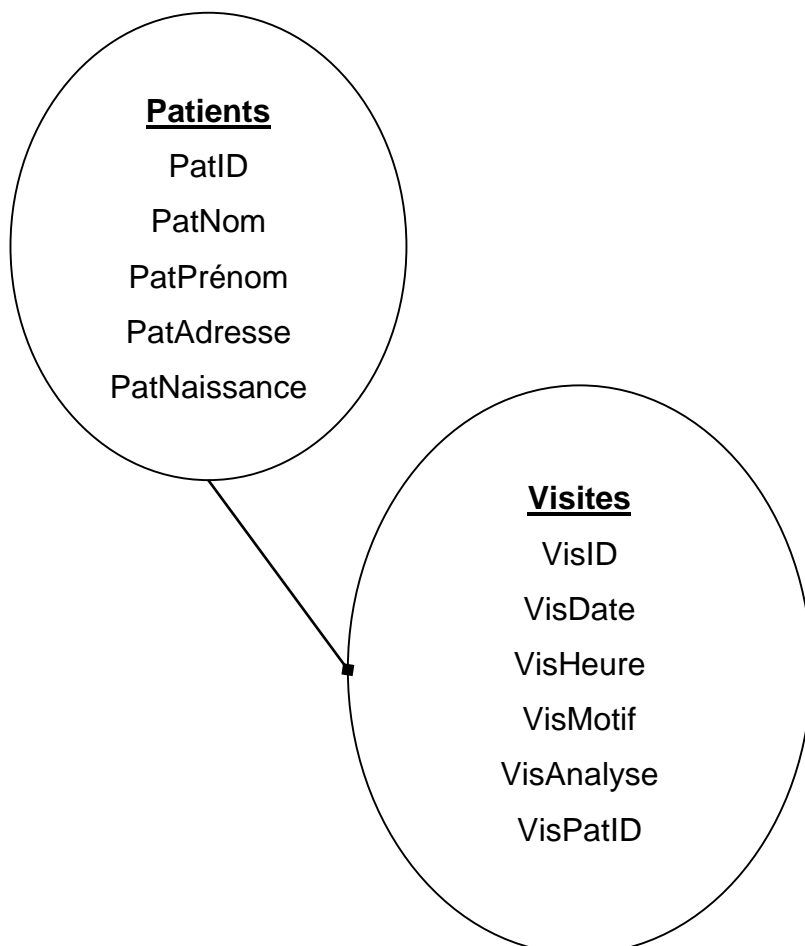
Exemple :

```
DROP TABLE Personnes
```

### 6.3 Exercices – mise à jour d'une base de données

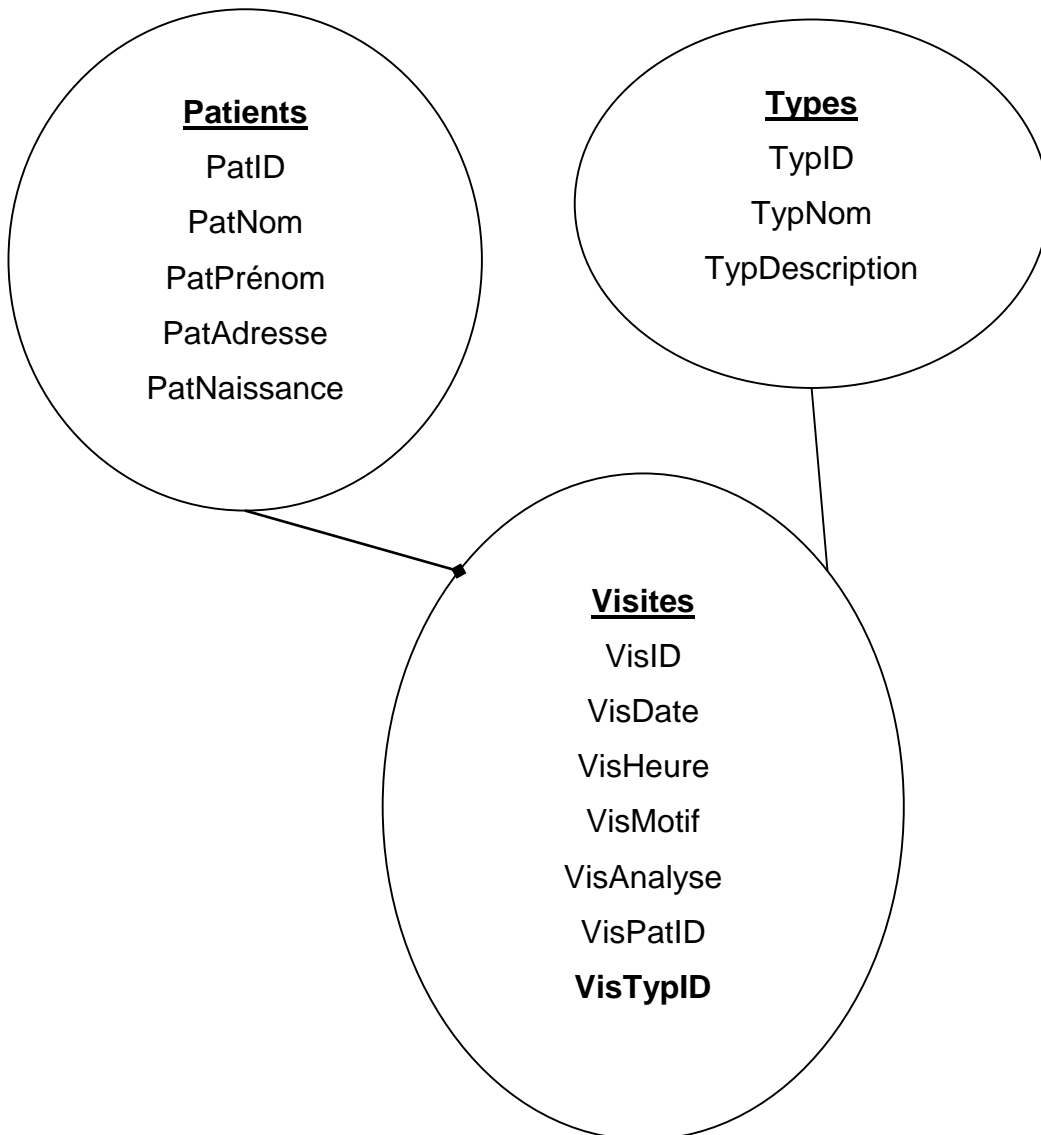
Gestion des visites de patients chez le médecin.

Créez la base de données "clinique" avec les tables suivantes



Ajoutez une contrainte d'unicité sur le champ "date" et "heure" (un patient ne peut pas avoir un rendez-vous avec un autre patient !)

Ajoutez la table "Types", et modifiez la table "Visites" pour lui ajouter la clé étrangère "VisTypID" :



Ajoutez un index sur le nom et le prénom de la table patients

## 7 Interrogation de base de données

### 7.1 Requêtes de sélection

Syntaxe générale :

```
SELECT [ DISTINCT ] attributs
      [ INTO OUTFILE fichier ]
      [ FROM relation ]
      [ WHERE condition ]
      [ GROUP BY attributs ]
      [ HAVING condition ]
      [ ORDER BY attributs [ ASC | DESC ] ]
      [ LIMIT [a,] b ]
```

Nom	Description
<b>SELECT</b>	Spécifie les attributs dont on souhaite connaître les valeurs.
<b>DISTINCT</b>	Permet d'ignorer les doublons de ligne de résultat.
<b>INTO OUTFILE</b>	Spécifie le fichier sur lequel effectuer la sélection.
<b>FROM</b>	Spécifie le ou les relations sur lesquelles effectuer la sélection.
<b>WHERE</b>	Définit le ou les critères de sélection sur des attributs.
<b>GROUP BY</b>	Permet de grouper les lignes de résultats selon un ou des attributs.
<b>HAVING</b>	Définit un ou des critères de sélection sur des ensembles de valeurs d'attributs après groupement.
<b>ORDER BY</b>	Permet de définir l'ordre (ASCendant par défaut ou DESCendant) dans l'envoi des résultats.
<b>LIMIT</b>	Permet de limiter le nombre de lignes du résultat

#### 7.1.1 Simple

Projection (vue) : on ne sélectionne qu'un ou plusieurs attributs d'une relation (on ignore les autres).

##### Personnes

Nom	Prénom	Adresse	Téléphone
<b>Martin</b>	Pierre	7 Place des Ronds	0258941236
<b>Dupond</b>	Jean	32 Rue des Poivrots	0526389152
<b>Dupond</b>	Marc	8 Rue des Soularde	0123456789

On projette la table Personnes  
sur les colonnes nom et  
prénom

SELECT nom,  
prénom  
FROM Personnes

Nom	Prénom
Martin	Pierre
Dupond	Jean
Dupond	Marc

Pour sélectionner tous les enregistrements d'une relation :

```
SELECT * FROM relation
```

Pour sélectionner toutes les valeurs d'un seul attribut :

```
SELECT attribut FROM relation
```

Pour éliminer les doublons :

```
SELECT DISTINCT attribut FROM relation
```

### 7.1.2 Avec filtre

Le filtre s'effectue sur un ou plusieurs attributs. La relation résultante a la même structure tandis que le contenu résultera des critères de sélection qui portent sur les valeurs des attributs.

#### Personnes

Nom	Prénom	Adresse	Téléphone
Martin	Pierre	7 Place des Ronds	0258941236
Dupond	Jean	32 Rue des Poivrots	0526389152
Dupond	Marc	8 Rue des Soularde	0123456789

SELECT \*  
FROM Personnes  
WHERE nom = "Dupond"

On ne sélectionne que les tuples dont  
l'attribut nom est égal à « Dupond »

Nom	Prénom	Adresse	Téléphone
Dupond	Jean	32 Rue des Poivrots	0526389152
Dupond	Marc	8 Rue des Soularde	0123456789

Par exemple, extraction de votre base de données de la liste des personnes de votre carnet d'adresse qui vivent à Paris.

```
SELECT nom,prénom FROM Personnes WHERE adresse LIKE '%paris%'
```

### 7.1.3 Avec tri

Le tri s'effectue sur un ou plusieurs attributs, dans l'ordre croissant ou décroissant. La relation résultante a la même structure et le même contenu que la relation de départ.

La commande du tri est la clause `ORDER BY` et à défaut de spécifications `DESC` (Descending) ou `ASC` (Ascending), le tri est croissant.

**Personnes**

Nom	Prénom	Adresse	Téléphone
Martin	Pierre	7 Place des Ronds	0258941236
Dupond	Jean	32 Rue des Poivrots	0526389152
Dupond	Marc	8 Rue des Soularde	0123456789

On sélectionne tous les tuples en les triant par ordre alphabétique de « prénom »

SELECT \*  
FROM Personne  
ORDER BY prénom ASC

Nom	Prénom	Adresse	Téléphone
Dupond	Jean	32 Rue des Poivrots	0526389152
Dupond	Marc	8 Rue des Soularde	0123456789
Martin	Pierre	7 Place des Ronds	0258941236

Pour trier les valeurs en ordre croissant :

```
SELECT DISTINCT attribut FROM relation ORDER BY attribut ASC
```

Pour se limiter aux **num** premiers résultats :

```
SELECT DISTINCT attribut FROM relation ORDER BY attribut ASC LIMIT num
```

Pour ne sélectionner que ceux qui satisfont à une condition :

```
SELECT DISTINCT attribut FROM relation WHERE condition ORDER BY attribut ASC LIMIT num
```

### 7.1.4 Exercices – Sélection simple/filtre/tri

Soit la table "gens"

Nom	Prenom	Age
Dupond	Pierre	24
Martin	Marc	48
Dupont	Jean	51
Martin	Paul	36
Dupond	Lionel	68
Chirac	Jacques	70

- 1) Sélectionnez toute la table:
- 2) Sélectionnez uniquement les "noms" des personnes de la table "gens":
- 3) Sélectionnez uniquement les "noms" des personnes de la table "gens" en supprimant les doublons:
- 4) Sélectionnez uniquement les "noms" par ordre alphabétique, des personnes de la table "gens" en supprimant les doublons:
- 5) Affichez uniquement les deux premiers résultats de la dernière requête:
- 6) Affichez uniquement les deux premiers résultats de la dernière requête sauf si le nom est égal à "Chirac":

### 7.1.5 Avec jointure

La jointure permet de fabriquer une nouvelle relation à partir de 2 ou

plusieurs autres en prenant comme pivot 1 ou plusieurs attributs. Par exemple, on concatène la table du carnet d'adresses et celle des inscrits à la bibliothèque en fonction du nom de famille (c'est typiquement du recoupement de fichiers).

Personnes

Nom	Prénom	Adresse	Téléphone
Martin	Pierre	7 Place des Ronds	0258941236
Dupond	Jean	32 Rue des Poivrots	0526389152

Bibliothèque

Nom	Dernierlivre
Dupond	Robinson
Jospin	Faust
Martin	Les piliers de la terre

SELECT prénom,Personnes.Nom, dernierlivre  
FROM Personnes, Bibliothèque  
WHERE Personnes.nom = Bibliothèque.nom

On joint les deux tables, grâce à la colonne nom.

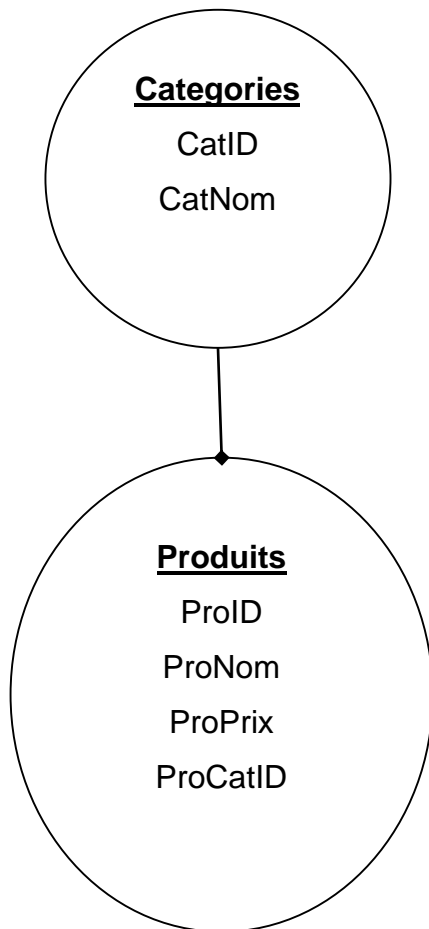
Et on combine cette jointure à une projection sur les attributs nom et dernier livre.

Prénom	Nom	Dernierlivre
Jean	Dupond	Robinson
Pierre	Martin	Les piliers de la terre

Attention à lever toute ambiguïté sur les noms d'attribut dans le cas où deux tables possèdent des colonnes de même nom.

### 7.1.6 Multiple

Soit un système de gestion catégories/produits :



```
CREATE TABLE Categories (  
    CatID int(3) PRIMARY KEY auto_increment,  
    CatNom varchar(30)  
);  
  
CREATE TABLE Produits (  
    ProID int(3) PRIMARY KEY auto_increment,  
    ProNom varchar(255),  
    ProPrix int(7),  
    ProCatID int(3),  
    CONSTRAINT FK_Cat FOREIGN KEY(ProCatID) REFERENCES Categories(CatID)  
);
```

```
INSERT INTO Categories(CatNom) VALUES( 'Immobilier');
INSERT INTO Categories(CatNom) VALUES( 'Services');
INSERT INTO Categories(CatNom) VALUES( 'Auto');
INSERT INTO Categories(CatNom) VALUES( 'Cinéma');
INSERT INTO Produits(ProNom, ProPrix, ProCatID) VALUES ( 'Maison', '150000', '1') ;
INSERT INTO Produits(ProNom, ProPrix, ProCatID) VALUES ( 'Appartement', '60000', '1') ;
INSERT INTO Produits(ProNom, ProPrix, ProCatID) VALUES ( 'Yourthe', '10000', '1') ;
INSERT INTO Produits(ProNom, ProPrix, ProCatID) VALUES ( 'Nettoyage', '' , '2') ;
INSERT INTO Produits(ProNom, ProPrix, ProCatID) VALUES ( 'Repassage', '' , '2') ;
INSERT INTO Produits(ProNom, ProPrix, ProCatID) VALUES ( 'Brad Pitt', '1000' , '4') ;
INSERT INTO Produits(ProNom, ProPrix, ProCatID) VALUES ( 'Mel Gibson', '500' , '4') ;
INSERT INTO Produits(ProNom, ProPrix, ProCatID) VALUES ( 'Jean Paul Belmondo', '10' , '4') ;
```

SELECT \* FROM Categories

CatID	CatNom
1	Immobilier
2	Services
3	Auto
4	Cinéma

SELECT \* FROM Produits

ProID	ProNom	ProPrix	ProCatID
1	Maison	150000	1
2	Appartement	60000	1
3	Yourthe	10000	1
4	Nettoyage	0	2
5	Repassage	0	2
8	Brad Pitt	1000	4
9	Mel Gibson	500	4
10	Jean Paul Belmondo	10	4



### 7.1.6.1 Requêtes imbriquées :

Lorsqu'il y a plusieurs tables, et que nous voulons récupérer par exemple les "produits" de la catégorie "Immobilier", le programmeur non initié correctement va faire minimum deux requêtes, une première pour récupérer l'id du produits dans la table Produits :

```
SELECT CatID FROM Categories WHERE CatNom='Immobilier';
```

→ Retourne la valeur "1"

Puis une deuxième pour récupérer les produits :

```
SELECT * FROM Produits WHERE ProCatID='1';
```

Le résultat de la 1ère requête peut être directement injecté dans la deuxième :

```
SELECT * FROM Produits WHERE ProCatID=( SELECT CatID FROM Categories WHERE  
CatNom='Immobilier') ;
```

### 7.1.6.2 Requêtes sélection multi-tables:

L'inconvénient de la méthode précédente est que le SGBD va quand même exécuter deux requêtes. La solution est d'utiliser une requête de sélection multi-tables :

```
SELECT *  
FROM Categories, Produits  
WHERE Categories.CatID = Produits.ProCatID  
AND Categories.CatNom="Immobilier"
```

**Remarque** : L'instruction AS permet de définir un alias, ce qui évite de réécrire le nom de la table au complet.

```
SELECT *  
FROM Categories AS c, Produits AS p  
WHERE c.CatID = p.ProCatID  
AND c.CatNom="Immobilier"
```

---

*Au lieu des requêtes multi-tables, la clause INNER JOIN a fait son apparition avec la version 2 de SQL, parce que le besoin s'était fait sentir de préciser à quel type de jointure appartenait une relation. Les puristes préconiseront donc le "Inner join" (voir point 7.1.10).*

---

### 7.1.7 Avec regroupement

L'instruction "Group by" permet de regrouper les lignes selon un critère.

```
SELECT ...  
FROM nom_table  
[WHERE condition]  
GROUP BY nom_colonne;
```

Soit la table "animal" suivante :

```
CREATE TABLE Animal (  
id SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT,  
espece VARCHAR(40) NOT NULL,  
sexe CHAR(1),  
date_naissance DATETIME NOT NULL,  
nom VARCHAR(30),  
commentaires TEXT,  
PRIMARY KEY (id)  
);
```

id	espece	sexe	date_naissance	nom	commentaires
1	chien	F	2008-02-20 15:45:00	Canaille	NULL
2	chien	F	2009-05-26 08:54:00	Cali	NULL
3	chien	F	2007-04-24 12:54:00	Rouquine	Mordille parfois
4	chien	F	2009-05-26 08:56:00	Fila	NULL
5	chien	F	2008-02-20 15:47:00	Anya	NULL
6	chien	F	2009-05-26 08:50:00	Louya	NULL
7	chien	F	2008-03-10 13:45:00	Welva	Dangereux
8	chien	F	2007-04-24 12:59:00	Zira	NULL
9	chien	F	2009-05-26 09:02:00	Java	NULL
10	chien	M	2007-04-24 12:45:00	Balou	NULL
11	chien	M	2008-03-10 13:43:00	Pataud	Paresseux
12	chien	M	2007-04-24 12:42:00	Bouli	NULL
13	chien	M	2009-03-05 13:54:00	Zoulou	NULL
14	chien	M	2007-04-12 05:23:00	Cartouche	NULL
15	chien	M	2006-05-14 15:50:00	Zambo	NULL
16	chien	M	2006-05-14 15:48:00	Samba	NULL

17	chien	M	2008-03-10 13:40:00	Moka	Boit du lait
18	chien	M	2006-05-14 15:40:00	Pilou	NULL
19	chat	M	2009-05-14 06:30:00	Fiero	NULL
20	chat	M	2007-03-12 12:05:00	Zonko	NULL
21	chat	M	2008-02-20 15:45:00	Filou	NULL
22	chat	M	2007-03-12 12:07:00	Farceur	NULL
23	chat	M	2006-05-19 16:17:00	Caribou	NULL
24	chat	M	2008-04-20 03:22:00	Capou	NULL
25	chat	M	2006-05-19 16:56:00	Raccou	Pas de queue depuis la naissance

L'instruction suivante

```
SELECT espece, COUNT(*) AS nb_animaux  
FROM Animal  
GROUP BY espece;
```

Retournera :

espece	nb_animaux
chat	7
chien	18

Si on ne souhaite prendre que les mâles :

```
SELECT espece, COUNT(*) AS nb_males  
FROM Animal  
WHERE sexe = 'M'  
GROUP BY espece;
```

Retournera :

espece	nb_males
chat	7
chien	9

**Remarque :**

Lorsque l'on fait un groupement dans une requête, avec GROUP BY, on ne peut sélectionner que deux types d'éléments dans la clause SELECT :

- une ou des colonnes **ayant servi de critère** pour le regroupement ;
- **une fonction d'agrégation** (agissant sur n'importe quelle colonne).

**7.1.7.1 MySQL gère-t-il le GROUP BY comme les autres SGBD ?**

Non, MySQL applique une optimisation spécifique qui constitue une "extension de la norme" et est donc en contradiction avec celle-ci.

MySQL est un SGBD extrêmement permissif. Dans certains cas, c'est bien pratique, mais c'est toujours dangereux.

Et notamment en ce qui concerne GROUP BY, MySQL ne sera pas perturbé si vous sélectionnez une colonne qui n'est pas dans les critères de regroupement. Reprenons la requête qui sélectionne la colonne date\_naissance alors que le regroupement se fait sur la base de l'espèce. Cette requête ne respecte pas la norme SQL, et n'a aucun sens. La plupart des SGBD vous renverront une erreur si vous tentez de l'exécuter.

```
SELECT espece, COUNT(*) AS nb_animaux, date_naissance
FROM Animal
GROUP BY espece;
```

Retournera

espece	nb_animaux	date_naissance
chat	7	2009-05-14 06:30:00
chien	18	2008-02-20 15:45:00

MySQL a tout simplement pris n'importe quelle valeur parmi celles du groupe pour la date de naissance. Soyez donc très prudents lorsque vous utilisez GROUP BY. Vous faites peut-être des requêtes qui n'ont aucun sens, et MySQL ne vous en avertira pas !

**7.1.7.2 Regroupement sur plusieurs critères :**

```
SELECT espece, sexe, COUNT(*) as nb_animaux
FROM Animal
GROUP BY espece, sexe;
```

espece	sexe	nb_animaux
chat	M	7
chien	F	9
chien	M	9

### 7.1.8 Champs calculés

Un champ calculé dans une requête est le résultat d'un calcul ou d'une fonction sur un des champs de la table.

Exemple :

```
SELECT MIN(ProPrix) AS Minimum, MAX(ProPrix) AS Maximum FROM Produits
```

Minimum	Maximum
0	150000

```
SELECT avg( ProPrix ) FROM `produits`
```

avg(ProPrix)
27688.7500

Calcul de 21% sur le prix :

```
SELECT ProNom, ProPrix*1.21 FROM produits
```

ProNom	ProPrix*1.21
Maison	181500.00
Appartement	72600.00
Yourthe	12100.00
Nettoyage	0.00
Repassage	0.00
Brad Pitt	1210.00
Mel Gibson	605.00
Jean Paul Belmondo	12.10
Un machin Bidule	121.00

### 7.1.9 Fonctions de MySQL

Ces fonctions sont à ajouter à vos requêtes dans un SELECT, WHERE, GROUP BY ou encore HAVING.

Vous avez à votre disposition :

- les parenthèses ( ),
- les opérateurs arithmétiques (+, -, \*, /, %),
- les opérateurs binaires (<, <=, >, >=, |, &),
- les opérateurs logiques qui retournent 0 (faux) ou 1 (vrai) (AND, OR, NOT, BETWEEN, IN),
- les opérateurs relationnels (<, <=, =, >, >=, <>).

Les opérateurs et les fonctions peuvent être composés entre eux pour donner des expressions très complexes.

### 7.1.9.1 Quelques exemples

```
SELECT nom
FROM produits
WHERE prix <= 100.5
```

Liste du nom des produits dont le prix est inférieur ou égal à 100.5 EUR.

```
SELECT nom,prénom
FROM élèves
WHERE age BETWEEN 12 AND 16
```

Liste des nom et prénom des élèves dont l'âge est compris entre 12 et 16 ans.

```
SELECT modèle
FROM voitures
WHERE couleur IN ('rouge', 'blanc', 'noir')
```

Liste des modèles de voiture dont la couleur est dans la liste : rouge, blanc, noir.

```
SELECT modèle
FROM voitures
WHERE couleur NOT IN ('rose', 'violet')
```

Liste des modèles de voiture dont la couleur n'est pas dans la liste : rose, violet.

### 7.1.9.2 Fonctions de comparaison de chaînes

- Le mot clé **LIKE** permet de comparer deux chaînes.
- Le caractère '%' est spécial et signifie : 0 ou plusieurs caractères.
- Le caractère '\_' est spécial et signifie : 1 seul caractère, n'importe lequel.

L'exemple suivant permet de rechercher tous les clients dont le prénom commence par 'Jean', cela peut être 'Jean-Pierre', etc... :

```
SELECT nom
FROM clients
WHERE prénom LIKE 'Jean%'
```

Pour utiliser les caractères spéciaux ci-dessus en leur enlevant leur fonction spéciale, il faut les faire précéder de l'antislash : '\'.

Exemple, pour lister les produits dont le code commence par la chaîne '\_XE' :

```
SELECT *
FROM produit
WHERE code LIKE '\_XE%'
```

### 7.1.9.3 Fonctions mathématiques

Fonction	Description
<b>ABS(x)</b>	Valeur absolue de X.
<b>SIGN(x)</b>	Signe de X, retourne -1, 0 ou 1.
<b>FLOOR(x)</b>	Arrondi à l'entier inférieur.
<b>CEILING(x)</b>	Arrondi à l'entier supérieur.
<b>ROUND(x)</b>	Arrondi à l'entier le plus proche.
<b>EXP(x), LOG(x), SIN(x), COS(x), TAN(x), PI()</b>	fonctions mathématiques de base...
<b>POW(x,y)</b>	Retourne X à la puissance Y.
<b>RAND(), RAND(x)</b>	Retourne un nombre aléatoire entre 0 et 1.0 Si x est spécifié, entre 0 et X
<b>TRUNCATE(x,y)</b>	Tronque le nombre X à la Yème décimale.

```
SELECT nom  
FROM filiales  
WHERE SIGN(ca) = -1  
ORDER BY RAND()
```

Cet exemple affiche dans un ordre aléatoire le nom des filiales dont le chiffre d'affaire est négatif.

A noter que :  $\text{SIGN}(ca) = -1 \Leftrightarrow ca < 0$

### 7.1.9.4 Fonctions de chaînes

Fonction	Description
<b>TRIM(x)</b>	Supprime les espaces de début et de fin de chaîne.
<b>LOWER(x)</b>	Converti en minuscules.
<b>UPPER(x)</b>	Converti en majuscules.
<b>LONGUEUR(x)</b>	Retourne la taille de la chaîne.
<b>LOCATE(x,y)</b>	Retourne la position de la dernière occurrence de x dans y. Retourne 0 si x n'est pas trouvé dans y.
<b>CONCAT(x,y,...)</b>	Concatène ses arguments.
<b>SUBSTRING(s,i,n)</b>	Retourne les n derniers caractères de s en commençant à partir de la position i.
<b>SOUNDEX(x)</b>	Retourne une représentation phonétique de x.

```
SELECT UPPER(nom)
FROM clients
WHERE SOUNDEX(nom) = SOUNDEX('Dupond')
```

On affiche en majuscules le nom de tous les clients dont le nom ressemble à 'Dupond'.

#### 7.1.9.5 Fonctions de dates et heures

Fonction	Description
<b>NOW()</b>	Retourne la date et heure du jour.
<b>TO_DAYS(x)</b>	Conversion de la date X en nombre de jours depuis le 1er janvier 1970.
<b>DAYOFWEEK(x)</b>	Retourne le jour de la semaine de la date x sous la forme d'un index qui commence à 1 (1=dimanche, 2=lundi...)
<b>DAYOFMONTH(x)</b>	Retourne le jour du mois (entre 1 et 31).
<b>DAYOFYEAR(x)</b>	Retourne le jour de l'année (entre 1 et 366).
<b>SECOND(x), MINUTE(x), HOUR(x), MONTH(x), YEAR(x), WEEK(x)</b>	Retournent respectivement les secondes, minutes, heures, mois, année et semaine de la date.
<b>DATEDIFF(expr,expr2)</b>	retourne le nombre de jours entre la date de début expr et la date de fin expr2

Cet exemple affiche le titre des articles parus il y a moins de 30 jours.

```
SELECT titre
FROM article
WHERE (TO_DAYS(NOW()) - TO_DAYS(parution)) < 30
```

Celui-ci affiche le nom et prénom des personnes dont l'année de la date est "2002"

```
SELECT Nom, Prénom, Date
FROM Personnes
WHERE Year([Date])=2002;
```

#### 7.1.9.6 Fonctions à utiliser dans les GROUP BY

Fonction	Description
<b>COUNT([DISTINCT]x,y,...)</b>	Décompte des tuples du résultat par projection sur le ou les attributs spécifiés (ou tous avec '*'). L'option DISTINCT élimine les doublons.
<b>MIN(x), MAX(x), AVG(x), SUM(x)</b>	Calculent respectivement le minimum, le maximum, la moyenne et la somme des valeurs de l'attribut X.



```
SELECT DISTINCT model
FROM voiture
GROUP BY model
HAVING COUNT(couleur) > 10
```

Ici on affiche les modèles de voitures qui proposent un choix de plus de 10 couleurs.

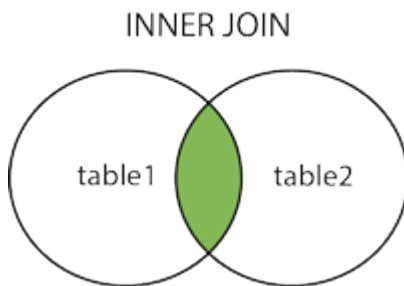
```
SELECT COUNT(*)
FROM client
```

Affichage du nombre de clients.

```
SELECT DISTINCT produit.nom, SUM(vente.qt * produit.prix) AS total
FROM produit, vente
WHERE produit.id = vente.produit_idx
GROUP BY produit.nom
ORDER BY total
```

Classement des produits par la valeur totale vendue.

### 7.1.10 Les jointures internes



Les jointures **internes**<sup>1</sup> (*inner joins*) ; à chaque ligne de la 1<sup>er</sup> table est associée toute ligne de la 2<sup>ème</sup> table vérifiant la condition.

Par exemple, pour obtenir les tuples des catégories qui contiennent des produits.

```
SELECT *  
FROM categories  
INNER JOIN produits ON categories.CatID = produits.ProCatID
```

CatID	CatNom	ProID	ProNom	ProPrix	ProCatID
1	Immobilier	1	Maison	150000	1
1	Immobilier	2	Appartement	60000	1
1	Immobilier	3	Yourthe	10000	1
2	Services	4	Nettoyage	0	2
2	Services	5	Repassage	0	2
4	Cinéma	8	Brad Pitt	1000	4
4	Cinéma	9	Mel Gibson	500	4
4	Cinéma	10	Jean Paul Belmondo	10	4

Ce premier type de jointure complexe apporte finalement peu de nouveautés puisque le "INNER JOIN" fonctionne exactement de la même manière qu'une jointure simple (avec la clause WHERE). Son principal intérêt est d'apporter une certaine lisibilité et de mieux distinguer les jointures internes (INNER) des jointures externes (OUTER).

---

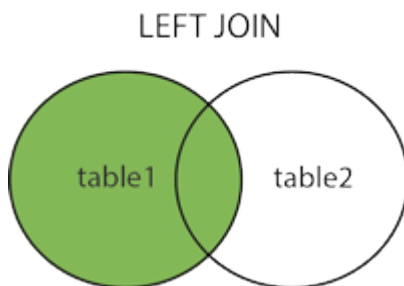
<sup>1</sup> Quand le type de jointure est omis, c'est automatiquement une jointure interne (inner)

### 7.1.11 Les jointures externes

**La jointure externe** (*outer join*), la plus compliquée, qui favorise une table (dite "dominante") par rapport à l'autre (dite "subordonnée"). Les lignes de la table dominante sont retournées même si elles ne satisfont pas aux conditions de jointure.

Pour bien comprendre les jointures externes :

#### 7.1.11.1 left outer



Une jointure interne est d'abord effectuée, puis y est ajoutée toute ligne de la 1<sup>ère</sup> table sans concordance avec la 2<sup>ème</sup> table et alors complétée par des "null".

Par exemple pour obtenir toutes les catégories, chacune avec tous les produits s'il y en existe.

```
SELECT *
FROM categories
LEFT OUTER JOIN produits ON categories.CatID = produits.ProCatID
```

CatID	CatNom	ProID	ProNom	ProPrix	ProCatID
1	Immobilier	1	Maison	150000	1
1	Immobilier	2	Appartement	60000	1
1	Immobilier	3	Yourthe	10000	1
2	Services	4	Nettoyage	0	2
2	Services	5	Repassage	0	2
3	Auto	NULL	NULL	NULL	NULL
4	Cinéma	8	Brad Pitt	1000	4
4	Cinéma	9	Mel Gibson	500	4
4	Cinéma	10	Jean Paul Belmondo	10	4

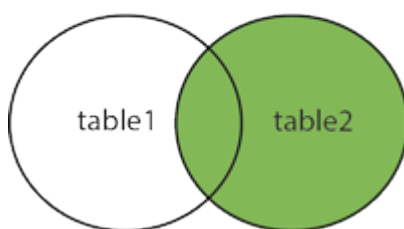
Par exemple pour obtenir tous les produits chacun avec sa catégorie s'il y en a une.

```
SELECT *
FROM produits
LEFT OUTER JOIN categories ON produits.ProCatID = categories.CatID
```

ProID	ProNom	ProPrix	ProCatID	CatID	CatNom
1	Maison	150000	1	1	Immobilier
2	Appartement	60000	1	1	Immobilier
3	Yourthe	10000	1	1	Immobilier
4	Nettoyage	0	2	2	Services
5	Repassage	0	2	2	Services
8	Brad Pitt	1000	4	4	Cinéma
9	Mel Gibson	500	4	4	Cinéma
10	Jean Paul Belmondo	10	4	4	Cinéma

#### 7.1.11.2 right outer

RIGHT JOIN



Une jointure interne est d'abord effectuée, puis y est ajoutée toute ligne de la 2<sup>ème</sup> table sans concordance avec la 1<sup>ère</sup> table et alors complétée par des "null".

Par exemple pour obtenir toutes les catégories, chacune avec tous les produits s'il y en existe.

```
SELECT *
FROM produits
RIGHT OUTER JOIN categories ON produits.ProCatID = categories.CatID
```

ProID	ProNom	ProPrix	ProCatID	CatID	CatNom
1	Maison	150000	1	1	Immobilier
2	Appartement	60000	1	1	Immobilier
3	Yourthe	10000	1	1	Immobilier
4	Nettoyage	0	2	2	Services
5	Repassage	0	2	2	Services
NULL	NULL	NULL	NULL	3	Auto
8	Brad Pitt	1000	4	4	Cinéma
9	Mel Gibson	500	4	4	Cinéma
10	Jean Paul Belmondo	10	4	4	Cinéma

Par exemple pour obtenir tous les produits chacun avec sa catégorie s'il y en a une.

```
SELECT *  
FROM categories  
RIGHT OUTER JOIN produits ON categories.CatID = produits.ProCatID
```

CatID	CatNom	ProID	ProNom	ProPrix	ProCatID
1	Immobilier	1	Maison	150000	1
1	Immobilier	2	Appartement	60000	1
1	Immobilier	3	Yourthe	10000	1
2	Services	4	Nettoyage	0	2
2	Services	5	Repassage	0	2
4	Cinéma	8	Brad Pitt	1000	4
4	Cinéma	9	Mel Gibson	500	4
4	Cinéma	10	Jean Paul Belmondo	10	4

Autre exemple d'explication : <http://www.epershand.net/developpement/mysql-bdd/comprendre-jointures-inner-left-right-join-mysql>

**7.1.12 Exercices - jointures internes/externes/requêtes jointures/multiples/avec regroupement/champ calculé**

Ajoutez la table Pilotes, modèles et villes à l'exercice concernant les avions (point 5.1) :

**MODELES**

ModID	ModNom	ModVitesse
1	Boeing 747	12000km/h
2	A300	700km/h
3	A310	915km/h
4	Boeing 707	984km/h
5	Concorde	2145km/h
6	Mercure	932km/h

**VILLES**

VilID	VilNom	VilNbHabitants	VilSuperficie
1	Nice	343304	71,92
2	Paris	2244000	105,4
3	Lyon	484344	47,95
4	Toulouse	441802	118,3
5	Lens	4277	46,61
6	Nantes	284970	65,19

**AVIONS**

AviID	AviModID	AviNombreDePlaces	AviViID
1	2	300	1
2	3	300	1
3	5	250	2
4	2	280	3
5	5	160	1
6	1	460	2
7	4	250	2
8	3	300	4
9	6	180	3
10	5	160	2

**PILOTES**

PiID	PiNom	PiPrenom	PiViID	PiSalaire
1	TIM	Vincent	2	26000.00
2	CLETTE	Lara	4	21000.00
3	AIPAN	Ahmed	1	18000.00
4	TERIEUR	Alain	2	17000.00
5	LAIBOUL	Ella	4	19000.00
6	TERIEUR	Alex	2	18000.00
7	DON	Guy	1	17000.00
8	RATAMAIR	Waldi	3	15000.00
9	OUIN	Serge	5	18000.00
10	GRAFFE	Otto	1	

**VOLS**

VolID	VolPilID	VolDate	VolDepartVilID	VolArriveVilID	VolHeureDepart	VolHeurearrive	VolAvilID
1	1	2013-10-30	1	4	11 :00	12 :30	1
2	1	2013-10-30	2	4	17 :00	18 :30	8
3	2	2013-10-30	4	3	14 :00	16 :00	1
4	5	2013-10-30	4	3	18 :00	20 :00	3
5	9	2013-10-30	2	1	06 :45	08 :15	1
6	10	2013-10-30	3	1	11 :00	12 :00	2
7	1	2013-10-30	2	3	08 :00	09 :00	4
8	8	2013-10-30	1	2	07 :15	08 :45	4
9	1	2013-10-31	6	3	09 :00	15 :30	8
10	8	2013-10-31	1	2	12 :15	13 :45	2
11	9	2013-10-31	2	3	15 :00	16 :00	2
12	1	2013-10-31	3	6	16 :30	20 :00	2
13	4	2013-10-31	1	5	11 :00	14 :00	5
14	3	2013-10-31	5	2	15 :00	16 :00	5
15	8	2013-10-31	2	4	17 :00	18 :00	9
16	7	2013-10-31	2	4	18 :00	19 :00	5



Reproduisez le schéma des tables en relation

Créez les tables et insérez les données ci-dessus.

1. Donner toutes les informations sur les pilotes
2. Donner le nom et le prénom des pilotes
3. Sélectionner l'identificateur et le nom de la ville de chaque ville
4. Sélectionner les noms des pilotes gagnant plus de 25000 €
5. Quels sont les noms des pilotes gagnant entre 20000 et 25000 € ?
6. Quelle est la vitesse des boeings?
7. Quels sont les noms des pilotes dont le salaire est inconnu?
8. Quelles sont les villes de départ des différents vols
9. Sélectionner les noms des pilotes habitant Paris
10. Quelles sont les capacités des avions de type Airbus ?
11. Quels sont les vols au départ de Nice desservant Paris
12. Quels sont les avions (identifiant de l'avion + nom du modèles) de capacité supérieure à 250 personnes ou localisés à Paris ?
13. Quels sont les vols au départ de Paris et dont l'heure d'arrivée est inférieure ou égale à 15h00 ?
14. Quel est le salaire moyen des pilotes parisiens ?
15. Trouver le nombre de vols au départ de Paris.
16. Trouver le nom des pilotes effectuant des vols au départ de Paris ?
17. Trouver le nom des pilotes effectuant des vols au départ de Paris sur des Airbus.
18. Quels sont les avions localisés dans la même ville que l'avion numéro 3.
19. Quels sont les pilotes dont le salaire est plus élevé que le salaire moyen des pilotes ?
20. Quel est le pilote qui gagne le moins
21. Quels sont les noms des pilotes niçois qui gagnent plus que tous les pilotes parisiens ?
22. Donner le nom des pilotes niçois qui gagnent plus qu'au moins un pilote parisien.
23. Rechercher les pilotes ayant même ville et même salaire que TIM.
24. Donner la liste des pilotes parisiens par ordre de salaire décroissant puis par ordre alphabétique des noms.
25. Quel est le nombre de vols effectués (ou pas) par chacun des pilotes ?
26. Trouver le nombre de vols effectués (ou pas) par pilote, en affichant à chaque fois le modèle et le numéro d'avion.
27. Donner le nombre de vols par pilote seulement s'il est supérieur à 5.
28. Donner le nom des pilotes effectuant au moins 5 vols.
29. Quels sont les numéros d'avions qui ne volent pas ?

## 7.2 Requêtes ensemblistes

Soit deux tables de même schéma :

Client		Fournisseur	
Nom	Chiffre	Nom	Chiffre
Belgacom	1150	ABComputer	28500
Dupont	19950	Belgacom	6250
Electrabel	3750	Carglass	1125
Informatifacile	3560	Electrabel	9520
Format Logique	15750	Informatifacile	3560
New Form	16840	Transport Claude	12500

### 7.2.1 Union

L'opérateur UNION produit un résultat qui possède les attributs des relations d'origine et tous les tuples de chacune.

Pour obtenir l'union des relations Client et Fournisseur :

```
SELECT * FROM Client
UNION
SELECT * FROM Fournisseur;
```

Nom	Chiffre
Belgacom	1150
Dupont	19950
Electrabel	3750
Informatifacile	3560
Format Logique	15750
New Form	16840
ABComputer	28500
Belgacom	6250
Carglass	1125
Electrabel	9520
Informatifacile	3560
Transport Claude	12500

Pour n'obtenir que les noms des relations **Client** et **Fournisseur**.

```
SELECT NOM FROM Client
UNION
SELECT NOM FROM Fournisseur;
```

### 7.2.2 Intersection

L'intersection<sup>2</sup> produit un résultat qui possède les tuples identiques de chacune sur base d'un ou plusieurs attributs spécifiés.

```
SELECT Nom FROM Client
WHERE Nom IN (SELECT Nom FROM Fournisseur);
```

Ou

```
SELECT Nom FROM Client
INNER JOIN FOURNISSEUR
USING (Nom) ;
```

Pour obtenir les noms des tiers qui sont à la fois client et fournisseur avec leur chiffre d'affaires côté fournisseur.

Client		Fournisseur	
<i>Nom</i>	<i>Chiffre</i>	<i>Nom</i>	<i>Chiffre</i>
Belgacom	1150	ABComputer	28500
Dupont	19950	Belgacom	6250
Electrabel	3750	Carglass	1125
Informatifacile	3560	Electrabel	9520
Format Logique	15750	Informatifacile	3560
New Form	16840	Transport Claude	12500



<i>Nom</i>	
Belgacom	6250
Electrabel	9520
Informatifacile	3560

```
SELECT Nom, Chiffre FROM Fournisseur
WHERE Nom IN (SELECT Nom FROM Client);
```

ou

```
SELECT Nom, Chiffre FROM Fournisseur
INNER JOIN Client
```

<sup>2</sup> La commande SQL INTERSECT n'existe pas en MySQL !

USING (Nom)

### 7.2.3 Différence

Client		Fournisseur		→	
Nom	Chiffre	Nom	Chiffre		Nom
Belgacom	1150	ABComputer	28500		Dupont
Dupont	19950	Belgacom	6250		Format Logique
Electrabel	3750	Carglass	1125		New Form
Informatifacile	3560	Electrabel	9520		
Format Logique	15750	Informatifacile	3560		
New Form	16840	Transport Claude	12500		

Le résultat de la différence entre les tuples de la première table qui, sur base d'un ou plusieurs attributs spécifiés, n'appartiennent pas à la deuxième.

```
SELECT Nom FROM Client
WHERE Nom NOT IN (SELECT Nom FROM Fournisseur);
```

Pour obtenir les noms et chiffres d'affaires des tiers qui sont uniquement fournisseurs.

```
SELECT Nom, Chiffre FROM Fournisseur
WHERE Nom NOT IN (SELECT Nom FROM Client);
```

ou (forme utile avec un SGBDR qui ne supporte ni l'**EXCEPT**, ni les **SELECT** imbriqués)

```
SELECT Nom, Chiffre
FROM Client LEFT OUTER JOIN Fournisseur ON Client.Nom = Fournisseur.Nom
WHERE Fournisseur.Nom IS NULL;
```

### 7.2.4 Le produit cartésien

Le produit cartésien produit un résultat qui possède les tuples de la première table, chacun associé à l'ensemble des tuples de la deuxième table.

Exemple sur une base de données des personnes :

Personne		Communication	
IdPers	Prenom	IdCom	LibCom
1	Pierre	1	Tél.privé
2	Marc	2	GSM
3	Michel		

Le produit cartésien donnera

<i>IdPers</i>	<i>Prenom</i>	<i>IdCom</i>	<i>LibCom</i>
1	<b>Pierre</b>	1	Tél.privé
1	<b>Pierre</b>	2	GSM
2	<b>Marc</b>	1	Tél.privé
2	<b>Marc</b>	2	GSM
3	<b>Michel</b>	1	Tél.privé
3	Michel	2	GSM

```
SELECT * FROM Personne, Communication;
```

Bien sûr le résultat du produit cartésien de cet exemple ne présente pas d'informations très utiles.

Le suivant permet d'établir la liste des étudiants susceptibles de se présenter à chaque examen organisé.

Etudiant		Examen
<i>IdEtudiant</i>	<b>Nom</b>	<i>LibExamen</i>
1	<b>Panzani</b>	Analyse
2	<b>Ergoton</b>	Langage C
		VB.Net

```
SELECT * FROM Examen, Etudiant;
```

<i>LibExamen</i>	<i>IdEtudiant</i>	<i>Nom</i>
Analyse	1	Panzani
Analyse	2	Ergoton
Langage C	1	Panzani
Langage C	2	Ergoton
VB.Net	1	Panzani
VB.Net	2	Ergoton

### 7.2.5 Exercices - Requêtes ensemblistes

Reprenez la base de données Vols/Avions/Pilotes et effectuez les requêtes suivantes :

1. Sélectionner les avions dont le modèle est A310, A320 ou A330
2. Quels sont les avions dont le modèle est différent de A310, A320 et A330?
3. Quels sont les numéros des pilotes pilotant les avions 2 et 4 ?
4. Quels sont les numéros des pilotes pilotant les avions 2 ou 4 ? (en utilisant une requête ensembliste !)
5. Quels sont les numéros des pilotes pilotant l'avion 2 mais jamais le 4 ?
6. Quels sont les numéros des pilotes qui pilotent tous les avions de la compagnie ?

### 7.3 fonction Coalesce

Le fonction COALESCE de MySQL permet de renvoyer une valeur par défaut si la valeur null est renvoyée initialement.

COALESCE() renverra la première valeur non NULL de la liste, ou NULL s'il n'y a pas de valeur non NULL.

Avec MySQL la valeur NULL est particulière et a parfois un comportement inattendu.

*Par exemple* : une table avec un champ ayant des valeurs NULL.

```
CREATE TABLE noms(  
  nom VARCHAR(45)  
);
```

avec les données suivantes:

nom
Pierre
Paul
NULL
NULL
Pierre

Comptons les lignes ayant pour nom 'Pierre':

```
SELECT count(*) FROM noms WHERE nom = 'Pierre';
```

On obtient 2.

Comptons les autres lignes :

```
SELECT count(*) FROM noms WHERE nom <> 'Pierre';
```

On obtient 1 (et non 3)!!

Explication: MySQL écarte automatiquement les valeurs NULL avant de faire une comparaison.

La fonction **COALESCE** qui permet de remplacer NULL par une valeur donnée.

```
SELECT count(*) FROM noms WHERE coalesce(nom, 'sans nom') <> 'Pierre';
```

On obtient bien 3.

La fonction **coalesce** permettra aussi de remplacer les valeurs « NULL » par 0 ou toute autre valeur

### 7.3.1 Exercices - coalesce

1. Affichez la liste des pilotes (numéro, nom, prénom) ainsi que leurs numéros de vols respectifs (ou N/A si ils n'ont pas volé)

## 7.4 GROUP BY WITH ROLLUP

La commande GROUP BY est utilisée en SQL pour grouper plusieurs résultats et utiliser des fonctions sur le groupement. L'ajout de la commande WITH ROLLUP permet quant à elle d'ajouter une ligne supplémentaire qui fonctionne tel qu'un système de « super-agrégateur » sur l'ensemble des résultats.

```
SELECT colonne1, fonction(colonne2)
FROM table
GROUP BY colonne1 WITH ROLLUP
```

Si par exemple, nous demandons le nombre de clients pour chaque code postal ou le nombre de commandes pour chaque code postal, etc. Nous pouvons en plus demander le nombre de clients total et Le nombre de commandes totales :

```
SELECT code_postal, COUNT(*), SUM(commande)
FROM clients
GROUP BY code_postal WITH ROLLUP
```

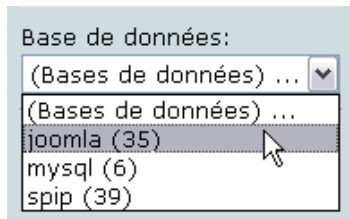
### 7.4.1 Exercices - WITH ROLLUP

1. Affichez le nombre de pilotes par villes avec le nombre total de pilote à la fin...

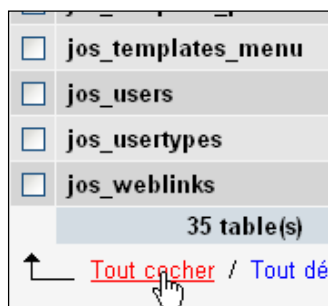
## 8. Importation/Exportation des données

### 8.1 Réalisation d'un backup sous PHPMyAdmin

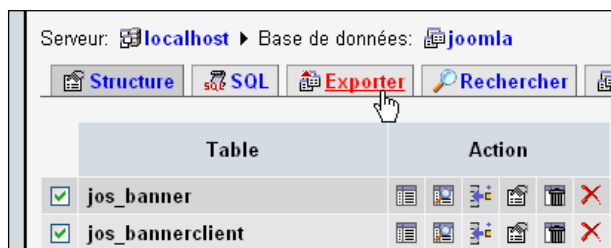
- a. Localhost/phpmyadmin
- b. Sélectionnez la base de données par exemple "Joomla" :



- c. Sélectionnez toutes les tables :



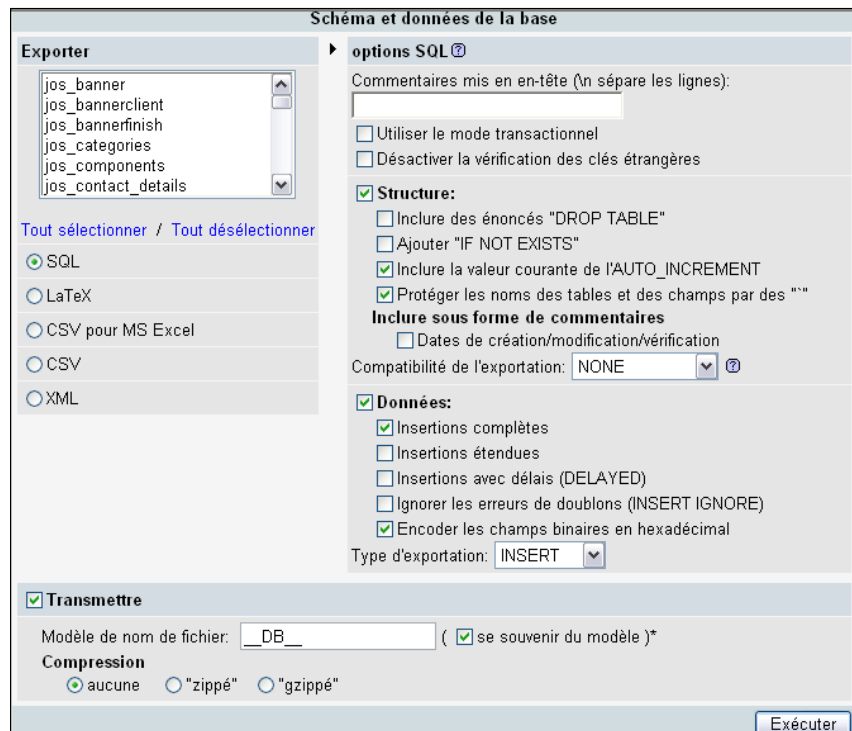
- d. Utilisez la fonction "Exporter" :



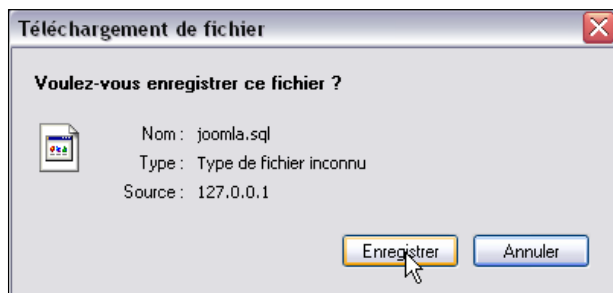
- e. Sélectionnez les options :
  - "Inclure la valeur courante de l'AUTO\_INCREMENT"
  - Protéger les noms des tables
  - Insertions complètes



• Transmettre

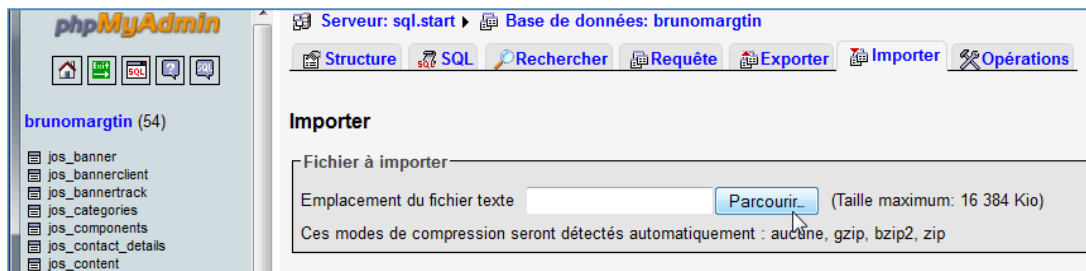


- f. Et appuyez sur "Exécuter"
- g. Donnez la destination où le fichier sera copié.

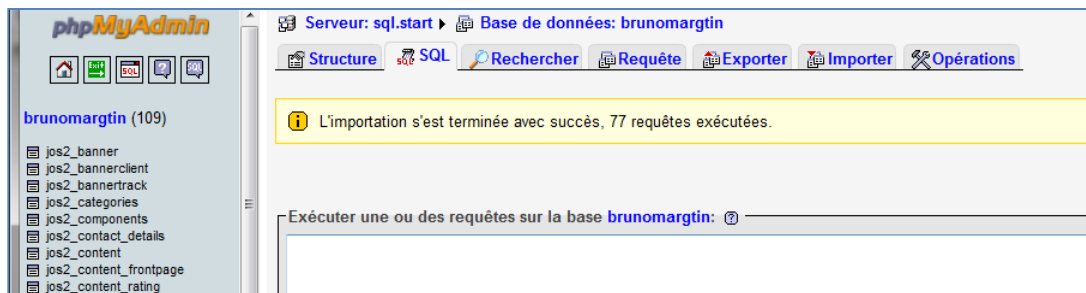


## 8.2 Restaurer une base de donnée sous PHPMYAdmin

- Sélectionnez ou créez une base de données.
- Cliquez sur "Importer" puis "parcourir" pour aller rechercher votre fichier ".sql"

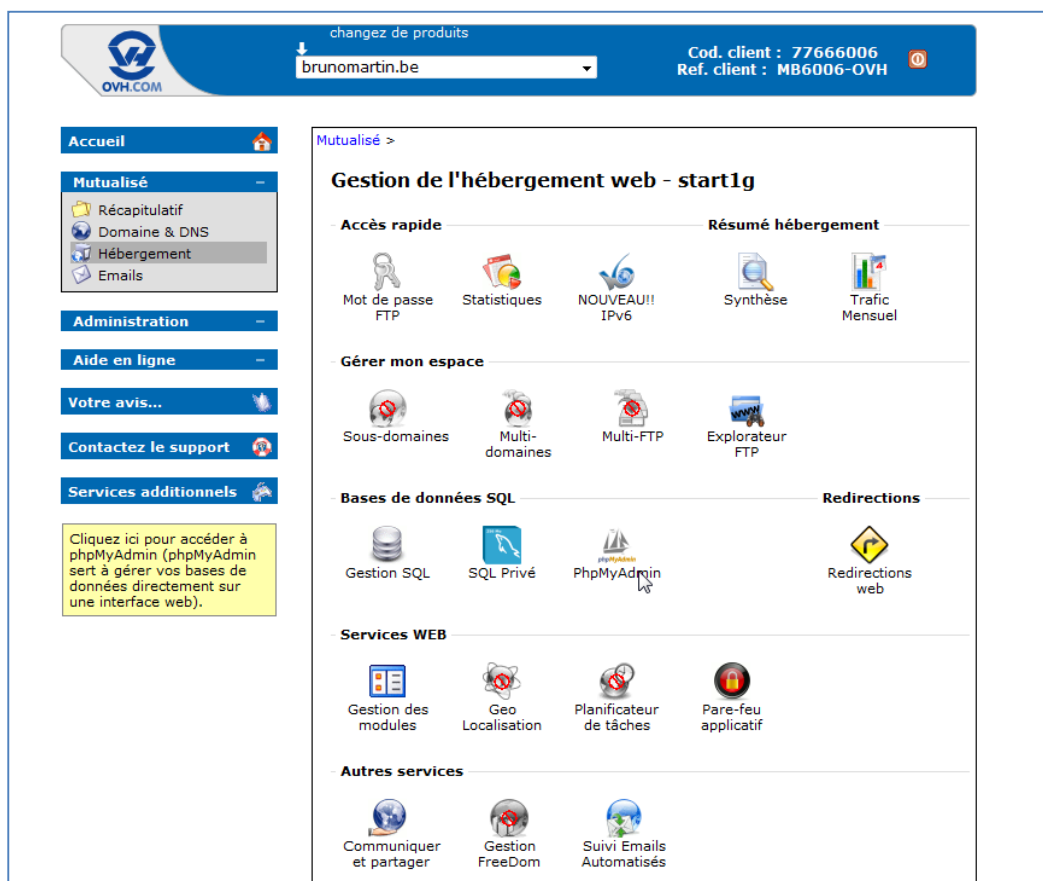


- Les nouvelles tables apparaissent dans la base de données sélectionnée :

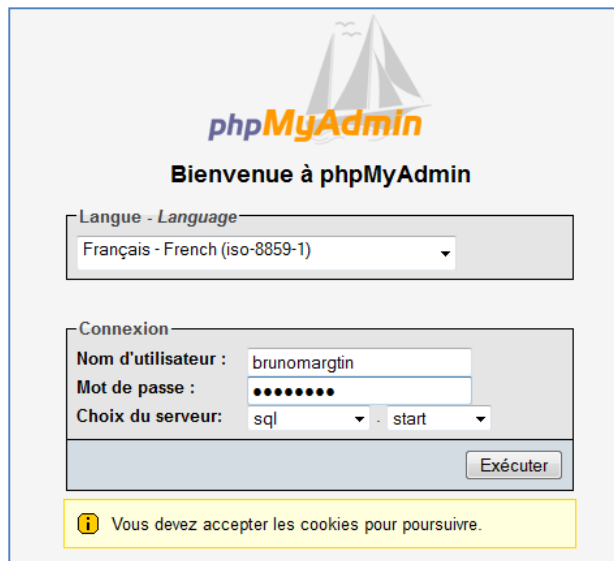


### 8.2.1 Exemple d'importation sous OVH

- Sous ovh, connectez-vous à votre manager à l'aide de votre login et mot de passe, sous la rubrique "Hébergement", cliquez sur "phpmyadmin" :



- b. Introduisez votre login et mot de passe pour accéder à votre base de données



- c. Importer vos tables dans la base de données existante :

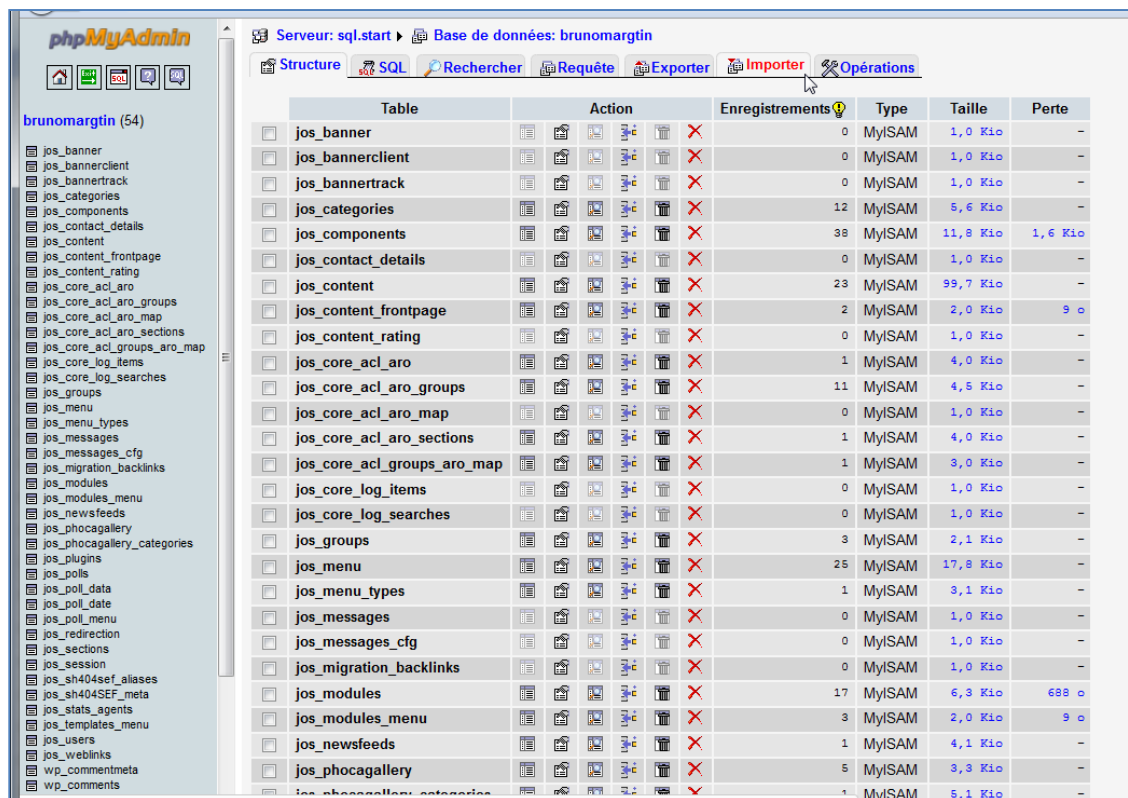
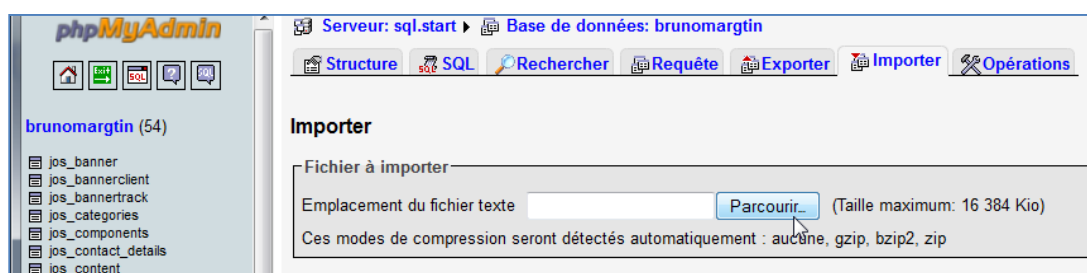
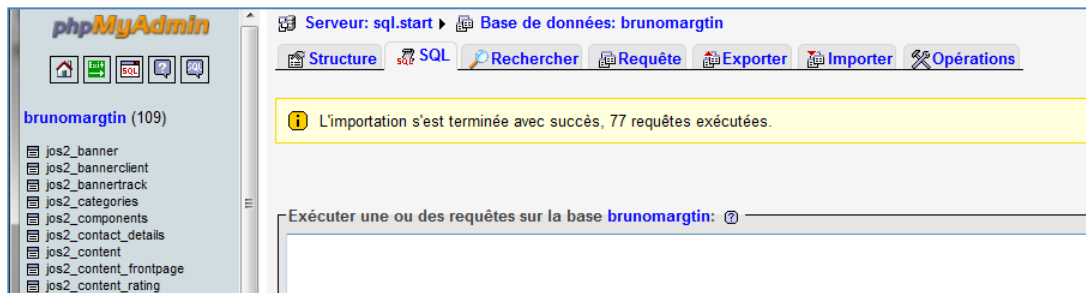


Table	Action	Enregistrements	Type	Taille	Perte
jos_banner		0	MyISAM	1,0 Kio	-
jos_bannerclient		0	MyISAM	1,0 Kio	-
jos_bannertrack		0	MyISAM	1,0 Kio	-
jos_categories		12	MyISAM	5,6 Kio	-
jos_components		38	MyISAM	11,8 Kio	1,6 Kio
jos_contact_details		0	MyISAM	1,0 Kio	-
jos_content		23	MyISAM	99,7 Kio	-
jos_content_frontpage		2	MyISAM	2,0 Kio	9 o
jos_content_rating		0	MyISAM	1,0 Kio	-
jos_core_acl_aro		1	MyISAM	4,0 Kio	-
jos_core_acl_aro_groups		11	MyISAM	4,5 Kio	-
jos_core_acl_aro_map		0	MyISAM	1,0 Kio	-
jos_core_acl_aro_sections		1	MyISAM	4,0 Kio	-
jos_core_acl_groups_aro_map		1	MyISAM	3,0 Kio	-
jos_core_log_items		0	MyISAM	1,0 Kio	-
jos_core_log_searches		0	MyISAM	1,0 Kio	-
jos_groups		3	MyISAM	2,1 Kio	-
jos_menu		25	MyISAM	17,8 Kio	-
jos_menu_types		1	MyISAM	3,1 Kio	-
jos_messages		0	MyISAM	1,0 Kio	-
jos_messages_cfg		0	MyISAM	1,0 Kio	-
jos_migration_backlinks		0	MyISAM	1,0 Kio	-
jos_modules		17	MyISAM	6,3 Kio	689 o
jos_modules_menu		3	MyISAM	2,0 Kio	9 o
jos_newsfeeds		1	MyISAM	4,1 Kio	-
jos_phocagallery		5	MyISAM	3,3 Kio	-
jos_phocagallery_categories		1	MyISAM	5,1 Kio	-

- d. Cliquez sur "Importer" puis "parcourir" pour aller rechercher votre fichier ".sql"



e. Les nouvelles tables (préfixées différemment !) apparaissent :



### 8.3 Exporter une base de données MySQL

<sup>3</sup>MySQL propose plusieurs façon d'exporter des données. La principale est la commande en ligne *mysql* permettant de réaliser à peu près n'importe quelle action sur les bases de données qu'elle contient à partir d'une simple ligne de commande :

```
mysql -h host -u user -p password base_de_donnees > fichier_dump
```

La notation suivante est également possible :

```
mysql --host host --user user --pass password base_de_donnees > fichier_dump
```

- *host* représente le nom ou l'adresse IP de la machine sur laquelle la base de données que vous désirez exporter est installée. Par défaut il s'agit de *localhost*, c'est-à-dire la machine à partir de laquelle la commande *mysql* est lancée
- *user* représente l'utilisateur avec lequel vous désirez vous connecter. Par défaut il s'agit de l'utilisateur *root*
- *password* représente le mot de passe de l'utilisateur avec lequel vous désirez vous connecter. Si vous n'indiquez pas de mot de passe, celui-ci sera demandé de manière interactive. Il ne doit pas y avoir d'espace entre *-p* et le mot de passe fourni, contrairement aux autres champs
- *base\_de\_donnees* est le nom de la base de données à exporter.
- *fichier\_dump* est le nom du fichier dans lequel la base va être exportée. Si aucun chemin absolu n'est précisé, le fichier sera stocké dans le même répertoire que la commande *mysql*. Attention de ne pas lui donner un nom d'un fichier existant dans ce répertoire !"

### 8.4 Sauvegarder une base de données MySQL avec mysqldump et PHP

Parfois, la base de données étant trop "lourde" l'exécution de l'exportation/importation risque de donner un "Time Out". La solution consiste alors à exporter les données via la commande "mysqldump" (soit en ligne de commande dos/linux – soit dans un script PHP comme ci-dessous.)

<sup>3</sup> Source <http://www.commentcamarche.net/contents/mysql/mysqlimport.php3>

«<sup>4</sup> Cette solution est intéressante, car elle vous permet d'importer des dumps importants et est accessible pour tous les hébergements.

Il faut donc éditer un script php :

Dans les scripts ci-dessous, remplacez "**nom\_de\_la\_base.sql**" par le nom de votre fichier, "**serveur\_sql**" par le nom du serveur sur lequel votre base est installée, "**nom\_de\_la\_base**" par le nom de votre base de données et "**mot\_de\_passe**" par le mot de passe associé à votre base.

En php (backupbase.php) :

```
<?php
echo "Votre base est en cours de sauvegarde.....";
system("mysqldump --host=serveur_sql --user=nom_de_la_base --
password=mot_de_passe nom_de_la_base > nom_de_la_base.sql");
echo "C'est fini. Vous pouvez récupérer la base par FTP";
?>
```

**Remarque 1** : Dans wamp, en local, utiliser le chemin complet de la commande mysqldump (ou définissez ce chemin dans la variable \$PATH). Utiliser les paramètres "dos" "-u" "-h" et non "--user" "--host".

```
system("C:\wamp\bin\mysql\mysql5.5.20\bin\mysqldump -u root -h localhost hubbihobby >
hubbihobby.sql");
```

**Remarque 2** : Si jamais votre base est trop volumineuse, vous pouvez faire un dump table par table en ajoutant l'option "**--tables nom\_de\_la\_table**" à la fin pour avoir cette commande :

```
mysqldump --host=serveur_sql --user=nom_de_la_base --password=mot_de_passe
nom_de_la_base --tables nom_de_la_table > nom_de_la_base.sql
```

**Remarque 3** : Vous pouvez aussi compresser ce fichier pour mieux le télécharger sur votre ordinateur (par FTP ou par le web).

Pour compresser le fichier, exécutez la commande gzip ce qui créera le fichier par l'extension .sql.gz :

```
system("gzip nom_de_la_base.sql");
```

Pour aller plus loin :

<http://cedric-duprez.developpez.com/tutoriels/mysql/sauvegarde6/>

---

<sup>4</sup> Source ovh : <http://guide.ovh.com/BackupBaseMySQL>

## 8.4 Exercices – mysqldump

Réalisez un backup complet de la base de données concernant les avions.

Réalisez ensuite un backup avec un fichier par table.

## 9. Cas pratiques – Analyse

### 9.1 Service achats

Dans le service achats, on a besoin de connaître pour chaque article quels sont les fournisseurs qui fournissent ledit article. Un article peut être fourni par plusieurs fournisseurs différents et un fournisseur fournit plusieurs articles. Réalisez le diagramme de la base de données avec l'indication des clefs primaires.

### 9.2 Élections en Syldavie

Vous devez gérer l'informatisation des élections communales de Syldavie. Le principe électoral syldave est dominé par les **partis**. On vote pour un parti et non pour une personne. Une dizaine de partis étaient présents aux dernières élections mais on s'attend à une légère augmentation. Tous les partis ne se présentent pas dans toutes les **communes**. Pour être reconnus les partis doivent déposer une liste de 500 signatures au Ministère de l'intérieur. Chaque commune, identifiée sans ambiguïté par le code postal doit déposer au ministère de l'intérieur les listes des partis qui se présentent dans cette commune. Chaque commune dresse la liste des **électeurs**, c'est-à-dire la liste des citoyens syldaves qui ont plus de 18 ans le jour des élections domiciliés dans cette commune à cette date. Les communes de plus de 5000 habitants organisent plusieurs **bureaux** de vote (un bureau par 5000 habitants) et signalent sur la convocation aux électeurs le numéro du bureau où ils doivent se présenter. Réalisez le diagramme de la base de données, avec l'indication des clefs primaires, qui permet d'enregistrer les résultats des élections.

### 9.3 Bibliothèque

Réalisez le schéma de la base de données d'une gestion de bibliothèque.

La base de données contiendra la liste de tous les livres (titre, date de parution, prix, format, nombre de page, ISBN) de la bibliothèque ainsi que leurs éditeurs (coordonnées complètes) et auteurs (**un seul par livre**) (date de naissance et éventuellement date de décès ainsi que leur bibliographie).

Il faut à tout moment savoir qui a emprunté un livre et à quelle date. La date de retour d'un livre sera aussi enregistrée (afin de pouvoir comptabiliser les jours de retard). Les informations sur les emprunteurs seront classiquement nom, prénom, adresse complète, téléphone, Email, sexe (H ou F) et la date de naissance.

## **9.4 Région Bruxelloises – administration**

La région Bruxelloise se compose de plusieurs administrations. Chacune est divisée en services.

Tout membre du personnel est attaché à un service et y occupe une fonction principale. Dans certains cas, il peut occuper également une autre fonction. Par exemple, une secrétaire pourrait remplacer la téléphoniste chaque fois que celle-ci est absente. (Ceci uniquement pour une certaine durée et dans un même service et forcément une même administration)

Réalisez le diagramme des tables et décrivez les attributs de celles-ci avec les clés primaires/clés étrangères

## **9.5 Région Bruxelloises – bâtiments**

La région Bruxelloise possède plusieurs bâtiments.

Chaque bâtiment est composé de plusieurs étages et chaque étage comprend plusieurs bureaux.

Un bureau peut être équipé de plusieurs ordinateurs.

On désire répertorier tous ces ordinateurs et leurs emplacements. On souhaite également disposer d'autres informations comme leur état, l'année d'achat, les logiciels installés, marque, fournisseur, ...

## **9.6 Région Bruxelloises – administration/bâtiments V1**

Sachant qu'un ordinateur est « attaché » à une fonction exercée par un membre, comment lier les deux diagrammes précédents ?

## **9.7 Région Bruxelloises – administration/bâtiments V2**

Sachant qu'un même ordinateur peut être utilisé par plusieurs membres et que ces membres, s'ils remplacent une personne, peuvent-être amenés à utiliser plusieurs ordinateurs, comment lier des deux diagrammes précédents ? *En partant du principe qu'un ordinateur peut être utilisé par plusieurs personnes et qu'une fonction exercée peut nécessiter plusieurs ordinateurs.*



## 9.8 IFOSUP

L'IFOSUP est une école supérieure qui comporte 4 sections (informatique, comptabilité, marketing, communication). Il s'agit de bachelier en 3 ans. Dans chaque section, il y a environ 50 étudiants en 1<sup>er</sup>, 30 en seconde et 20 en troisième. Chaque année comporte entre 8 à 15 cours. Un cours n'est donné que dans une seule année d'étude. Mais un cours peut être donné dans plusieurs sections. Un cours est donné par un seul professeur, un professeur peut donner plusieurs cours. Un étudiant peut s'inscrire soit à tous les cours d'une section, soit à certain cours d'une section, soit à certains cours dans plusieurs sections différentes. Réaliser le schéma des tables nécessaire à l'inscription des étudiants, la gestion des professeurs ainsi que les cours.

## 9.9 Agriculteur

Un agriculteur désire connaître les productions réalisées sur ses parcelles culturales. Une parcelle peut comporter plusieurs productions dans la même année civile.

Pour chaque parcelle, on veut connaître quelles productions ont été réalisées et à quelles dates. On désire également connaître le rendement de chaque production par parcelle, ainsi que les apports en N, P, K pour une période donnée.

Enfin, on doit pouvoir disposer de la quantité et du nom de l'engrais qui a été épandu sur chaque parcelle (à une date donnée).



L'agriculteur vous fournit les informations suivantes :

- Le nom de la parcelle et sa surface
- Les coordonnées géographiques de la parcelle
- Le nom de la production et son unité de production
- Les dates début et fin de production
- La quantité produite
- Le nom de l'engrais
- La proportion de N (Azote), de P (Phosphore) et de K (Potassium) dans l'engrais
- La quantité d'engrais épandue à une date donnée sur une parcelle donnée



## 9.10 Forum

Réalisez le schéma relationnel d'un forum (de base)

## 9.11 Camping

Gestion de réservation d'un camping.

Le camping des ifosupiens est constitué d'un terrain découpé en emplacement de différents tarifs. Les emplacements peuvent être de plusieurs types (vide, avec mobile home, avec caravane) et disposer d'une place de parking ou non.

Lorsqu'un client veut réserver un emplacement, il spécifie les dates de début et de fin de séjour, ainsi que type d'emplacement souhaité. Il est défini par un identifiant, un nom, un prénom, une adresse, un numéro de téléphone, une adresse de courrier électronique. Les emplacements sont numérotés. Ils font partie d'une zone et leur situation (à l'ombre ou non) doit être spécifiée ainsi que leur raccordement à l'électricité.

Le camping propose des activités (payantes ou gratuites) comme un cinéma, un restaurant, une piscine, un court de tennis, des terrains de pétanque, de volley, soirée dansante, ... Ces activités sont situées dans une zone précise du camping.

Une fois la réservation effectuée, le client a 15 jours pour envoyer un acompte (20% du prix total du séjour). Au-delà de ce délai, la réservation est annulée et l'emplacement libéré pour la période considérée.

Dessiner le schéma des tables qui permettrait d'enregistrer les réservations, en précisant bien quelles sont les clés primaires/étrangères. Retranscrivez le code SQL de la création de la table « Emplacements ».

## 9.12 Fleuves

Réalisez le schéma relationnel de la BD suivante, en définissant clairement les clés primaires ainsi que les clés étrangères :

« On souhaite créer une base de données destinée à la gestion des pays et de ces fleuves. Un pays est connu par un nom, son identifiant, une superficie, un nombre d'habitants et la liste des fleuves qui le traversent. Un fleuve est connu par son identifiant, son nom, sa longueur, le nom du pays dans lequel il prend sa source, la liste des pays qu'il traverse, la distance parcourue dans chacun des pays. »

### 9.13 Gestion d'anciens

« Gestion d'une association d'anciens »

L'association des anciens étudiants d'une école souhaite informatiser la gestion de son fichier des « anciens ».

Pour tout ancien étudiant, elle détient son nom, prénom et adresse ainsi que son numéro d'inscription à l'école et elle connaît les diplômes qu'il a obtenus, ainsi que leurs dates d'obtention.

Elle essaie également de connaître la liste des entreprises qui les ont employé par le passé et celles dans laquelle ils sont actuellement employé ainsi que et la fonction ou les fonctions qu'ils y occupent ou qu'ils ont occupés. Ils peuvent bien sûr avoir exercés plusieurs fonctions différentes dans plusieurs sociétés différentes.

En vue de produire des statistiques relatives à l'emploi des anciens étudiants, il serait également intéressant de connaître la durée de chaque emploi presté.

Réalisez le schéma de la base de données ainsi que les relations entre les tables. (N'oubliez pas de bien préciser les clés primaires et clés étrangères.) Reproduisez le code SQL de la création de la table « **Anciens** » (uniquement celle-là) avec les clés primaires (*autoincrement*) et les clés étrangères s'il y en a.

### 9.14 Gestion des logements dans une agence immobilière

Une agence de location de maisons et d'appartements désire gérer sa liste de logements. Elle voudrait en effet connaître l'implantation de chaque logement (nom de la commune et du quartier) ainsi que les personnes qui les occupent (les signataires uniquement). Le loyer dépend d'un logement. Les logements sont organisés en fonction de différents types (maison, studio, T1, T2...)

L'agence facturera toujours en plus du loyer la même somme forfaitaire (lié aux types) à ses clients. Par exemple, le prix d'un studio sera toujours égal au prix du loyer + 30 de charges forfaitaires par mois.

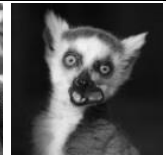
Pour chaque logement, on veut disposer également de l'adresse, de la superficie ainsi que du loyer.

Quant aux individus qui occupent les logements (les signataires du contrat uniquement), on se contentera de leurs noms, prénoms, date de naissance et numéro de téléphone. Pour chaque commune, on désire connaître le nombre d'habitants ainsi que la distance séparant la commune de l'agence.

On souhaite gérer l'historique de l'occupation des logements par les individus.



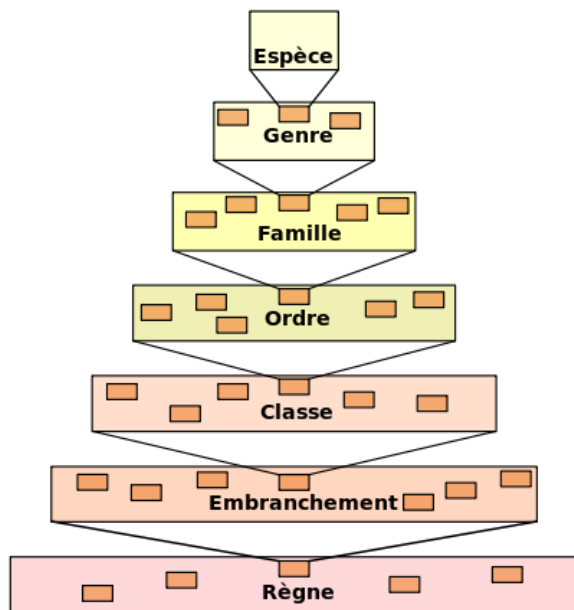
## 9.15 Pairi Daiza



### Gestion du parc animalier

Pairi Daiza a besoin de répertorier tous les animaux de son parc. Créez le schéma de la base de données, en précisant bien quelles sont les clés primaires/clés étrangères.

Pour la bonne gestion de celui-ci, les informations nécessaires à savoir sur les animaux, sont : nom, date de naissance, poids, taille, qui est son père et qui est sa mère. Chaque animal est rangé et catégorisé selon le schéma suivant :



Par exemple le **KOALA**

**Règne** : Animalia

**Sous rubrique de règne** :

**Embranchement** : Vertébré

**Sous rubrique d'embranchement** :

**Classe** : Mammalia Marsupialia

**Sous rubrique de classe** :

**Ordre** : Diprotodontia Vombatiformes

**Sous rubrique d'ordre** :

**Famille** : Phascolarctidae

**Sous rubrique de famille** :

**Genre** : Phascolarctos

**Sous rubrique de genre** : l'animal

**Espèce** : *Phascolarctos cinereus*

L'animal doit aussi être classé selon ses origines (il peut se trouver dans plusieurs pays et un pays peut bien entendu contenir plusieurs animaux).

Par exemple, le koala est originaire d'Australie. Ou encore l'éléphant est originaire de l'Ouganda, de la Tanzanie et du Kenya.

De plus, pour des raisons pratiques, le parc souhaiterait pouvoir gérer la **nourriture** sous forme de « menu » contenant une quantité de « viande » ainsi qu'une quantité de « légumes ». Certains menus pourront être les mêmes pour plusieurs animaux. Mais un animal n'a bien sûr droit qu'à un seul menu.

## 9.16 Toutenbois

L'entreprise toutenbois est un grossiste en bois et matériaux de recouvrement de toitures. Les clients de cette entreprise sont classés par catégories (particuliers, entrepreneurs, grossistes,...). Chaque client reçoit une facture en fin de mois. Chaque article appartient à une famille d'article. Pour chaque article, une description et un prix de vente sont déterminés. La base de données permettra aussi de gérer le stock disponible des articles afin d'augmenter la quantité de celui-ci pour éviter qu'il tombe en rupture de stock.

Réalisez le schéma de la base de données ainsi que les relations entre les tables. (N'oubliez pas de bien préciser les clés primaires et clés étrangères.) Reproduisez le code SQL de la création de la table « **Articles** » (uniquement celle-là) avec les clés primaires (*autoincrement*) et les clés étrangères s'il y en a.

## 9.17 UGC

L'UGC vous demande d'informatiser la gestion de séances de cinéma de leurs salles en Belgique ! Sachant qu'une ville peut contenir plusieurs cinémas, eux-mêmes possédant plusieurs salles (avec les informations suivantes : Capacité, climatisation, son numérique, 3D, taille de l'écran).

Dans chacune des salles plusieurs « séances » seront programmées (avec une date, une heure de début, et une heure de fin). Ces séances diffuseront un et un seul film. Pour les films, il faudra stocker les informations suivantes : Titre, année du film, réalisateur (1 seul par film !) et les rôles des différents acteurs (il peut y en avoir plusieurs). Pour les réalisateurs et les acteurs il faudra stocker leur nom et date de naissance. Attention qu'un réalisateur peut parfois avoir aussi un rôle en tant qu'acteur (même dans son propre film). *Par exemple, Tarantino peut jouer le rôle du « méchant cow-boy » dans « Django Unchained » tout en étant le réalisateur du film...*

Réalisez le schéma de la base de données ainsi que les relations entre les tables. (N'oubliez pas de bien préciser les clés primaires et clés étrangères.) Reproduisez le code SQL de la création de la table « **Cinema** » (uniquement celle-là) avec les clés primaires (*autoincrement*) et les clés étrangères si nécessaire.

## **10 Cas pratiques – Requêtes**

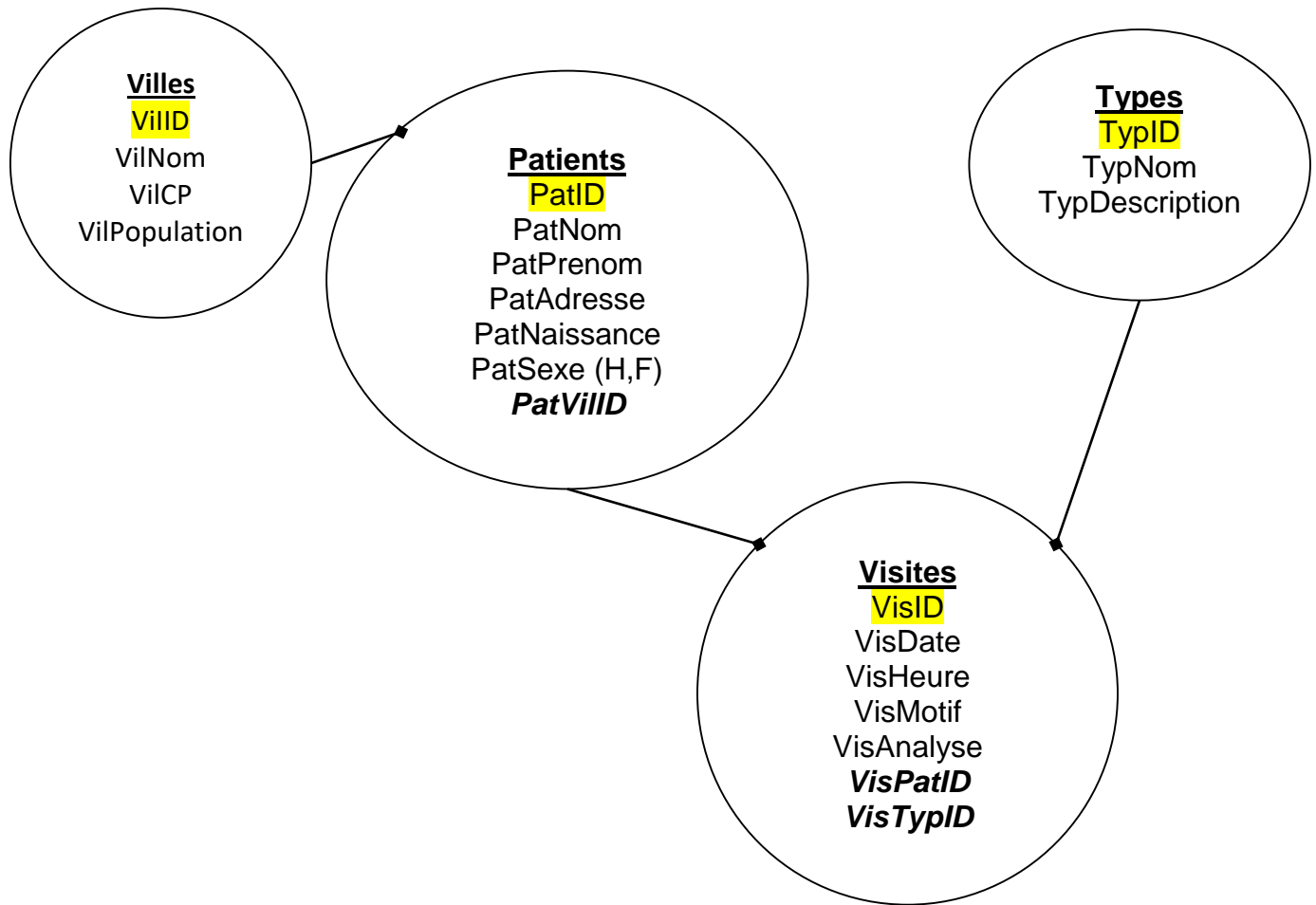
### **10.1 bibliothèque**

À l'aide de la DB « bibliotheque.sql »

1. En face de chaque titre d'ouvrage (par ordre alphabétique) afficher le nom et prénom de son auteur.
2. Affichez le titre et l'auteur des ouvrages empruntés suivi du nom de leur emprunteurs.
3. A la suite de problèmes de saisie informatique, il existe un certain nombre d'emprunteurs sans emprunts. Affichez les nom d'emprunteurs, suivi de l'identifiant de tous les emprunts qu'il y en ait un ou non
4. Réunissez dans une seule liste tous les titres de livres contenant le mot « nuit » et tous les titres contenant le mot « noire » par ordre décroissant de titre.
5. On veut afficher le titre des livres qui contiennent à la fois « noire » et « nuit ».
6. Affichez dans une colonne les noms et dans une colonne les numéros de téléphone des personnes et des editeurs.
7. Sélectionnez sur une colonne Lecteurs/Auteurs tous les personnnnes qui ont le même nom qu'un auteur.
8. Quels sont les livres qui n'ont jamais été empruntés ?
9. Quels sont les livres qui ont été rendus dans les 15 jours (par rapport à la date de prêt)
10. Quelles sont les personnes qui ont emprunté plusieurs fois un même livre ?

## 10.2 Clinique

À l'aide de la base de données suivante : (clinique.sql)

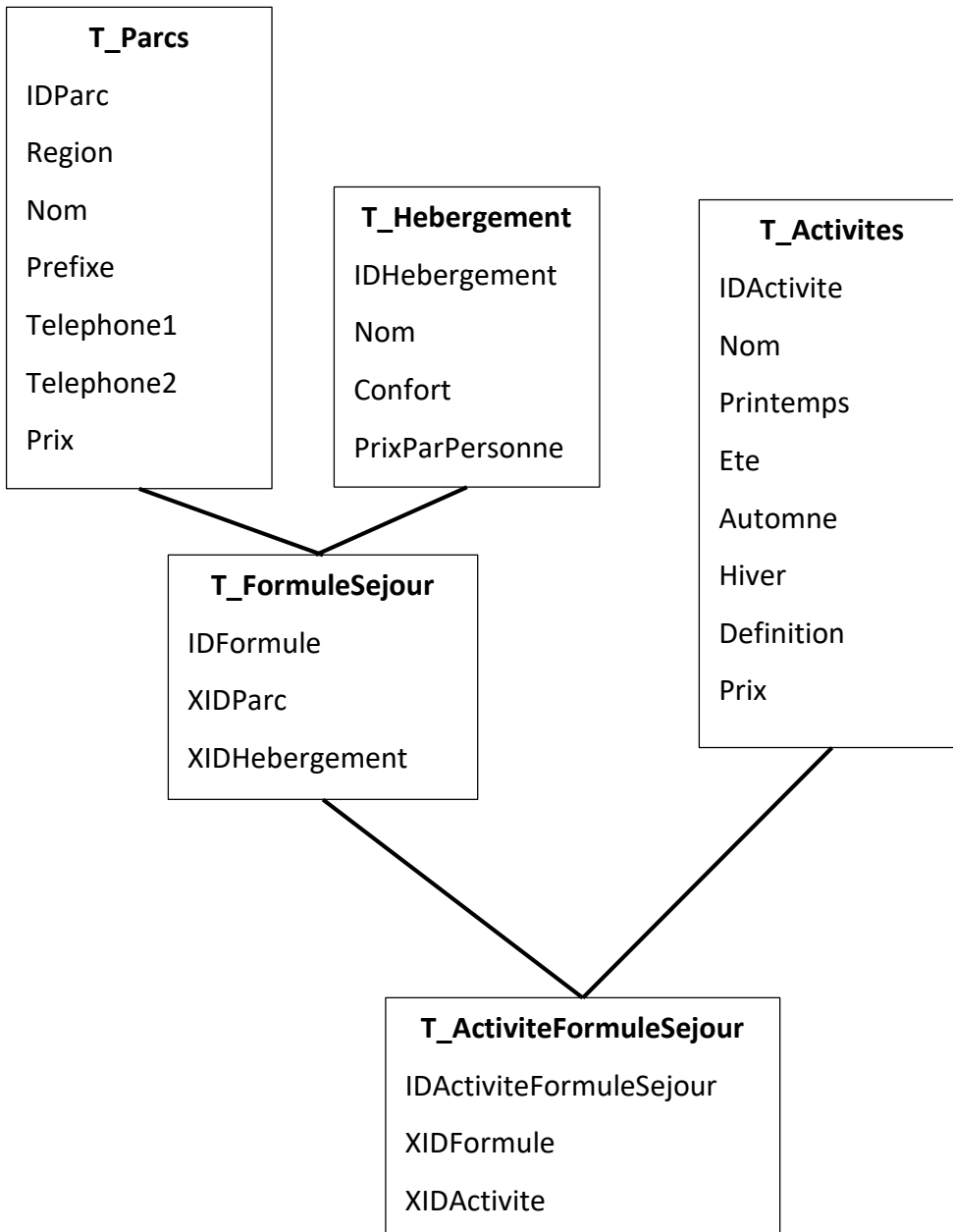


Réalisez les requêtes suivantes :

1. Affichez la liste des patients (nom et prénom), qui n'ont jamais eu de grippe
2. Affichez, pour chaque sexe (Homme, Femme) le nombre de visite qui ont été effectuée avant midi.
3. Affichez la liste des patients (nom et prénom), avec toutes les maladies et description de celle-ci qu'ils ont déjà eu.
4. Quelle est la ville qui a la population la plus élevée ?
5. Affichez la liste des nom et prénom de patients de la ville de Wavre par ordre inversement alphabétique de nom
6. Affichez le nom et prénom des patients qui n'ont jamais été consulter (ceux qui n'ont pas encore effectué de visite)
7. Affichez la liste des noms et prénoms de patients qui sont nés avant 2000 par ordre alphabétique de nom
8. Affichez le nom des patients qui ont eu la grippe autant de fois que "Jacques"

### 10.3 Montagnes

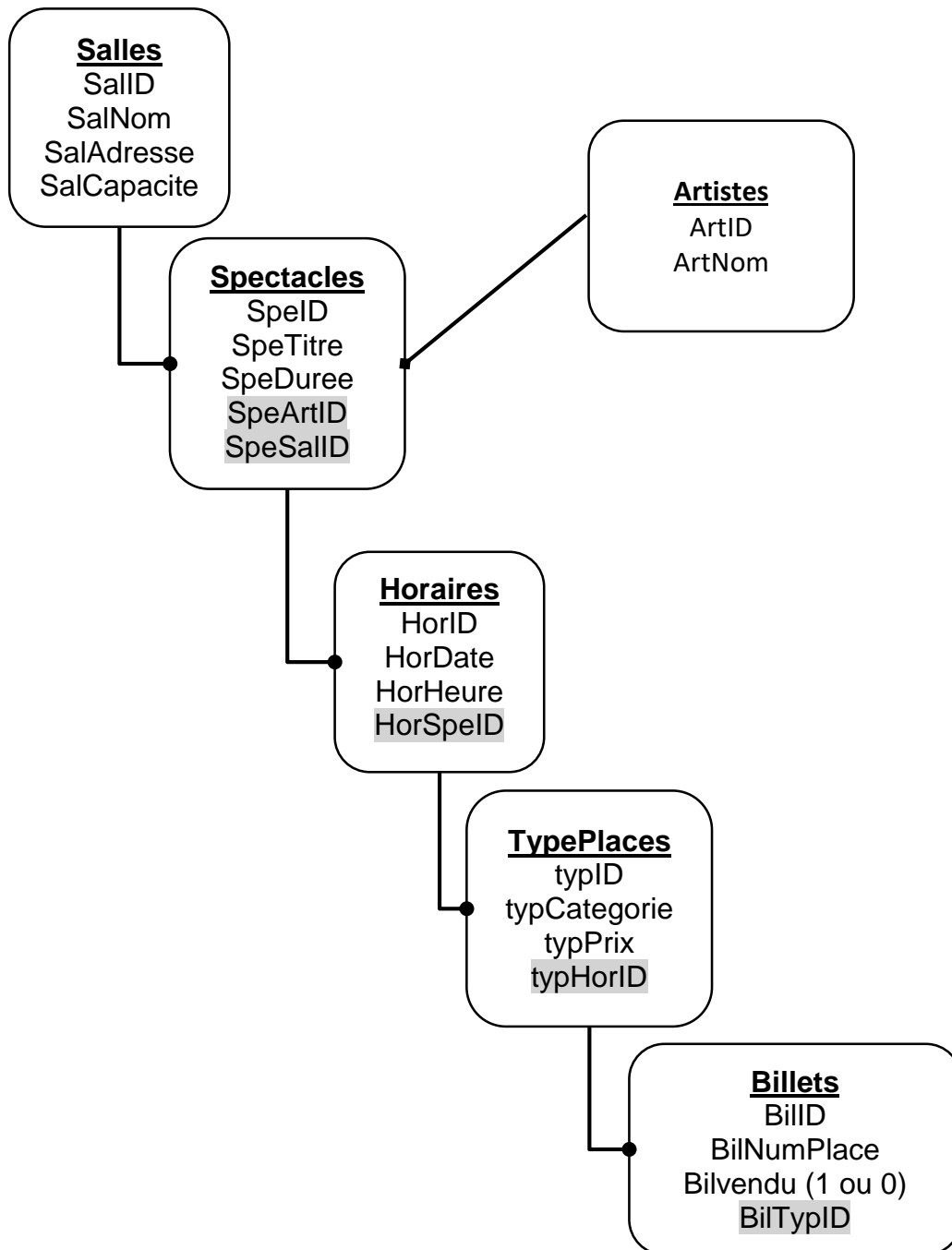
À l'aide de la base de données « montagne.sql », réalisez les requêtes ci-dessous.



- 1) Créez une requête qui n'affiche que les Parcs dont le prix n'est pas gratuit
- 2) Créez une requête qui affiche l'identifiant de la formule(IDFormule), Nom du parc, région et prix du parc, Nom de l'hébergement et prix de l'hébergement, Nom de l'activité, prix et les différentes saisons
- 3) Créez une requête qui affiche les parcs où il y a des activités en hiver
- 4) Créez une requête qui affiche le nombre d'hébergements par région
- 5) Quelles sont les activités dont le prix est supérieur au prix moyen des activités ?
- 6) Quelles sont les activités ou aucun séjour n'est proposé ?

## 10.4 Spectacles

Soit le fichier spectacle.sql reprennant la BD suivante :



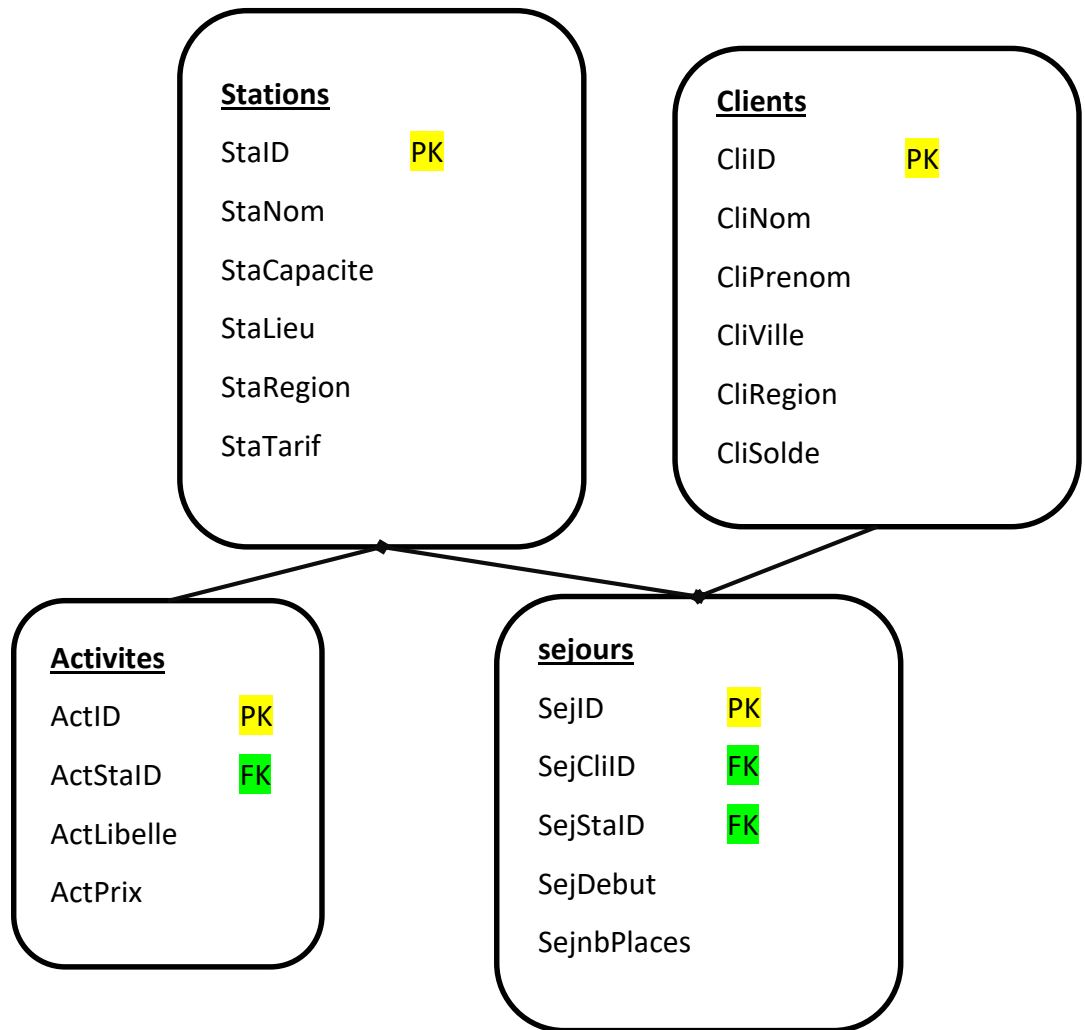
Réalisez les requêtes SQL permettant de répondre aux questions suivantes

1. Quelles sont les dates de concert à Forest National?
2. Quel est le nom de la salle ayant la plus grande capacité ?
3. Quels sont les artistes n'ayant jamais réalisé de spectacles à la Forest National ?
4. Combien de billets ont été vendus et invendus pour le concert de Cédric Gervy du 2014-10-06 ?
5. Quels sont les dates des concerts, les salles, les titres et le chanteur des spectacles pour lesquels il ne reste plus des billets invendu (=sold out)?



## 10.5 voyages

Soit le fichier « voyage.sql » reprenant la BD suivante :

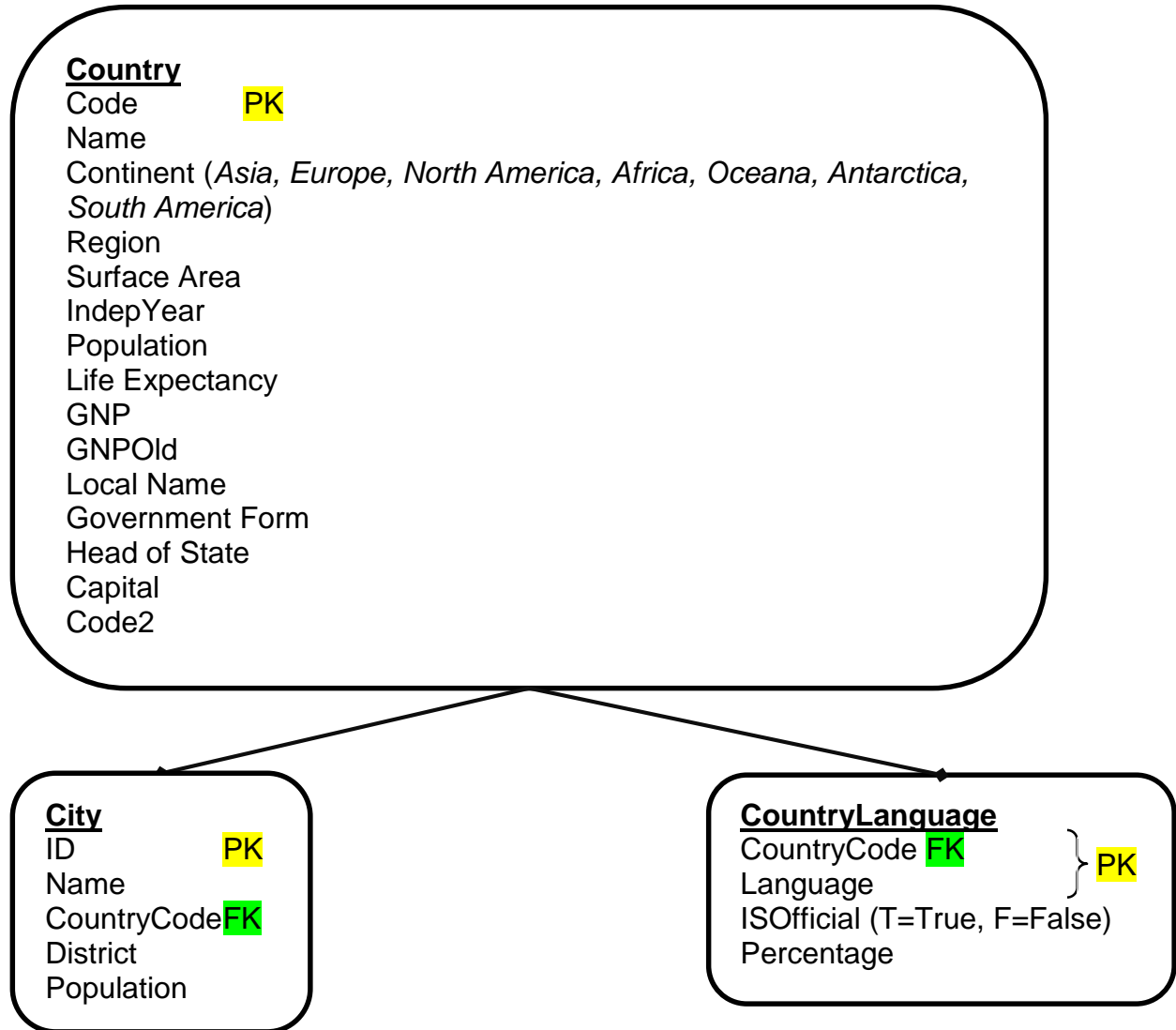


Réalisez les requêtes SQL permettant de répondre aux questions suivantes

1. Quel est le solde moyen des clients qui ont séjournés dans la station « Passac »
2. Affichez par stations, le nombre de clients différents
3. Affichez pour Mr Fogg les activités disponibles/proposées par ordre alphabétique, leurs prix et la station uniquement pour celles dont le prix est inférieur à 150 €.
4. Affichez les stations ainsi que leurs capacités, lieux, régions et tarifs pour lesquelles il n'y a aucune activité proposée.
5. Affichez le nom des clients qui ont réservé plus de 3 séjours.
6. Affichez le nom et prénom des clients qui n'ont jamais été à Passac.
7. Quelle est le nom et le coût de l'activité la plus chère de « Venusa » ?
8. Quelles sont les dates de séjour dont les nombres de place sont identiques

## 10.6 World

- 1) CREATE DATABASE world ;
- 2) Insertion et création des tables : « world\_innodb.sql »
- 3) Schéma :



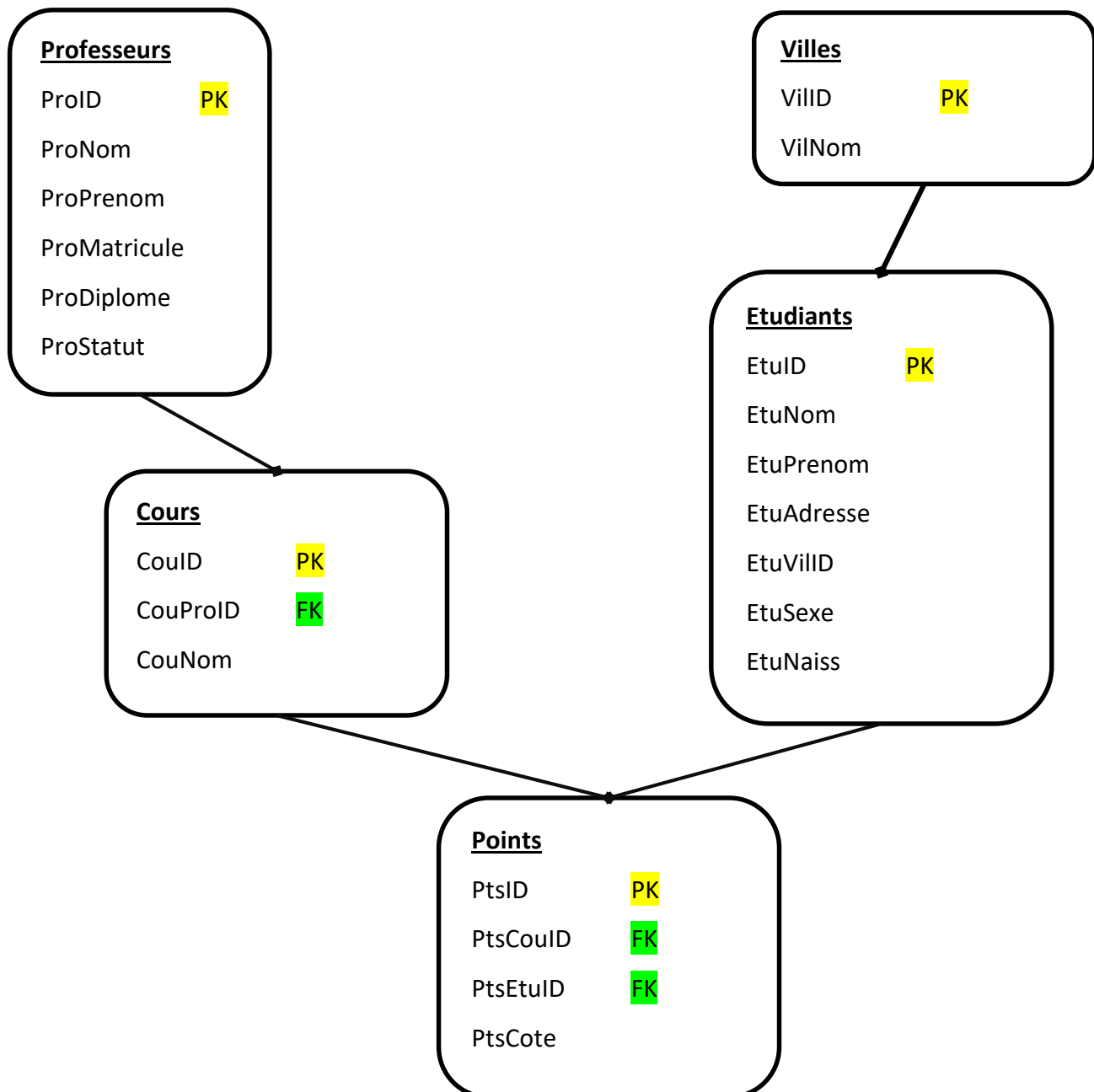
### Requêtes SQL

1. Affichez le nom des villes dont la population est inférieure ou égale à 50000 habitants.
2. Quelles sont les langues « Officielles » et leur pourcentage, qui sont parlées en Belgique (Belgium)
3. Combien de langues officielles et combien de langues non officielles sont parlées en Belgique ?
4. Quelle est la ville qui a la population la plus élevée ?
5. Affichez les « district » des Etats-Unis (USA) par ordre inversement alphabétique, dont la population totale du district dépasse les 3000000 habitants.
6. Affichez le nom des villes d'Europe où on parle français dans le pays.
7. Quels sont les pays qui n'ont pas de ville ?

8. Quelle(s) est/sont la ou les ville(s) dont la population est supérieure à la moitié de la population de son pays
9. Quel(s) est/sont le(s) pays dont la densité est supérieure celle de la Belgique, classés par ordre croissant de densité

## 10.7 Gestion d'une école

- 1) Prenez la base de données pour la gestion des points d'une école. L'idée générale est de pouvoir encoder les points des étudiants inscrits à un cours donné par un professeur. Les informations sur les étudiants sont : Numéro de matricule, Nom, Prénom, Adresse, CP, Ville, Sexe. Les informations sur les professeurs : numéro de matricule, Nom, Prénom, Diplôme, Statut (AS = Assistant, MCF = maître de conférence, PR = Professeur).



**Professeurs :**

Nom : Bernair

Prénom : Michel

Matricule : 007

Diplôme : Master Histoire

Statut : Maitre de conférences

Chargé de cours : Histoire, Sociologie

Nom : Delbar

Prénom : Benjamin

Matricule : 009

Diplôme : Informaticien, Master en Biologie

Statut : Assistant

Chargé de cours : Biologie

Nom : Martin

Prénom : Bruno

Matricule : 008

Diplôme : Doctorat Mathématique

Statut : Professeur

Chargé de cours : Mathématique, Informatique, Œnologie

Nom : Vanassche

Prénom : Nathalie

Matricule : 003

Diplôme : Régendat en Histoire

Statut : Professeur

Chargé de cours : Economie

## Etudiants :

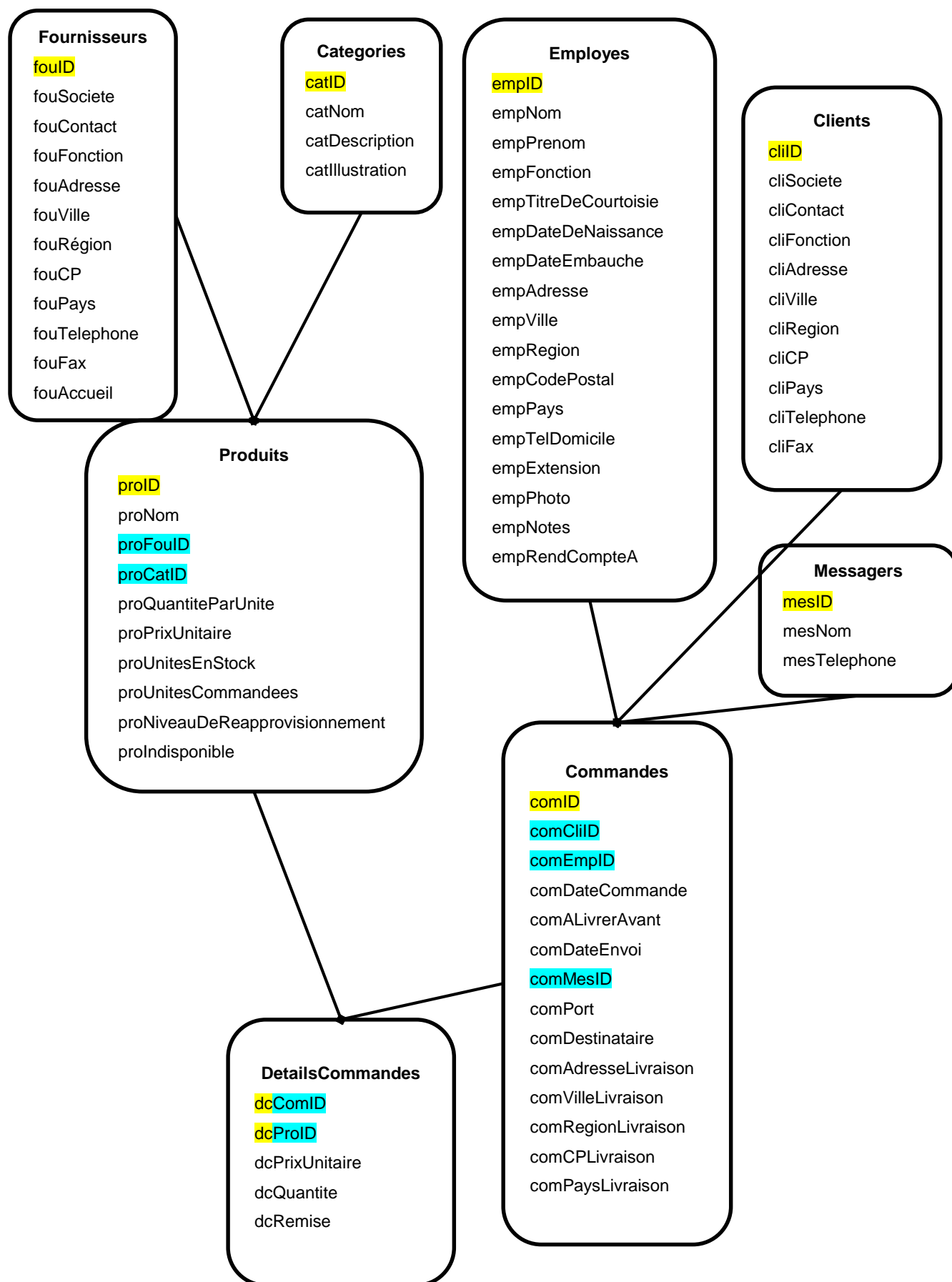
N°	Sexe	nom	prenom	Rue	Numéro	CP	Ville	Date de Naissance	Histoire	Sociologie	Mathématique	Informatique	Oenologie	Economie	Biologie
1	M	AUBOISDORMANT	Abel	Rue des arbres	12	1300	Wavre	24/6/1977	13	4	10	16	11	13	20
2	M	ZIEUVAIR	Bruno	Clos de l'Armoise	1323	1300	Wavre	23/8/1975	11		17	14	15	12	2
3	F	HONNETE	Camille	Rue des fraises	1	1342	Limelette	25/8/1973	18	18	15	12	15	15	16
4	M	ALAVANILLESIOUPLAIT	Douglas	Place de l'Ornoy	2	1348	Louvain-La-Neuve	23/5/1976	18	14			12	9	16
5	F	AVULEUR	Edith	Chemin de traverses	2	1332	Genval	25/4/1974	13	18	4	9	13	4	17
6	M	UJOUR	Fred	Rond-Point de la gare	32	1332	Genval	12/2/1977	2	4	0	9	5	7	3
7	M	MENSOIF	Gérard	Rue de la boisson	89	1348	Louvain-La-Neuve	12/5/1977	8	16	13	15	17	10	14
8	F	AVRANTE	Hélène	Avenue des combattants	12b	1300	Wavre	5/5/1977	15	4	6			11	17
9	M	DRÉSSAMÈRE	Ivan	Avenue des combattants	1	1348	Louvain-La-Neuve	6/7/1977	12	9	6	16	10	3	16
10	M	UMUL	Jacques	Rue des indépendants	55	1300	Wavre	5/5/1978	8	7	6	2	17	9	14
11	M	ORDINE	Kid	Place rouge	1	1348	Louvain-La-Neuve	5/3/1972	16	13	5	7	16	14	16
12	F	TROUILLE	Lassie	Rue des légumes	122	1332	Genval	3/5/1973	16	1	8	14	19	11	10
13	F	HONNETE	Marie	Rue des fraises	1	1342	Limelette	5/8/1973	14	15	5	19	17	8	19
14	M	OTINE	Nick	Rue de la pharmacie	5	1332	Genval	5/4/1977	10	19	15	19	12	12	10
15	M	HERME	Olaf	Rue du silence	0	1332	Genval	6/7/1975		14	11	10		14	

Requêtes :

1. Afficher quel est l'âge moyen des garçons et des filles ?"
2. Afficher le nom des enseignants d'histoire.
3. Afficher les noms des étudiants qui n'ont pas de notes en Sociologie.
4. Afficher le nom des matières qui sont enseignées par des maîtres de conférences ou des assistants.
5. Afficher par ordre alphabétique d'étudiant (nom et prénom), les points qu'il a obtenus dans chaque matière.
6. Afficher le nom, l'âge et le sexe des étudiants qui ont eu une note d'informatique supérieure à la moyenne générale du cours d'informatique de la classe.
7. Afficher le nom et le statut des enseignants qui enseignent dans plus d'une matière.
8. Afficher par cours le nombre d'élève qui ont réussi.
9. Afficher le nom, le prénom et le sexe des étudiants qui ont une note en informatique supérieure à leur note de Mathématique.
10. Pour les étudiants n'ayant pas de note dans une matière, afficher le nom de l'étudiant et le nom de la matière concernée.
11. Afficher, les matières pour lesquelles la moyenne des notes est inférieure à 10. Afficher le nom de l'enseignant correspondant.
12. Afficher, pour chaque matière, la meilleure note et le nom ou les noms des étudiants qui l'ont obtenue.
13. Afficher le nombre de garçons et le nombre de filles qui ont réussi tous les cours.
14. Afficher, pour chaque sexe (Homme, Femme) la moyenne des notes, par cours, dans les matières enseignées par M Bernair.
15. Afficher le nombre de cours réussi par les étudiants (même ceux qui ont raté tous les cours)

## 10.8 Comptoir

Base de données comptoir



1. Quels sont les employés qui travaillent à « London »?
2. Quelles sont toutes les informations du client dont le cliid est « dracd » ?
3. Afficher les clients dont la fonction n'est ni acheteur ni chef des ventes
4. Quel est/sont le ou les client(s) qui a/ont acheté le moins de produit « tofu » ?
5. Quel est le produit qui a le prix unitaire le plus élevé ?
6. Combien d'employé ont effectué une commande en juillet 1997 ?
7. Affichez pour chaque employé la somme de son chiffre d'affaire (les remises (%) doivent entrer en ligne de compte).
8. On considère les employés comme des commerciaux.  
Quel est celui qui a le chiffre d'affaires le moins élevé (les remises (%) doivent entrer en ligne de compte) ?
9. Quelle(s) est/sont la catégorie de produits que le fournisseur «Karkki Oy» a le plus vendu ?
10. Quel(s) est le pays abritant à la fois des employés, des clients et des fournisseurs ?
11. Parmi les commandes dont le prix total est supérieur à la moyenne des prix totaux quelle est la moyenne de celles-ci ?
12. Quels sont les produits dont le prix unitaire est supérieur à la moyenne des prix unitaires et qu'alfki n'a pas achetés ?



## 12. Evaluation

## BES WEBDEVELOPER (5XBDR)

### Système de gestion de bases de données

**Pour atteindre le seuil de réussite,  
l'étudiant sera capable :**

*à partir d'un cahier des charges  
développant une problématique liée à une  
base de données, en disposant d'une  
station informatique opérationnelle équipée  
d'un logiciel "Bases de données",*

- ♦ d'établir un schéma compatible de la situation présentée dans le cahier des charges;
- ♦ de construire la structure de la base de données sur un système de gestion de bases de données relationnelles en produisant les contenus nécessaires ;
- ♦ de construire des requêtes sous un langage tel que SQL.

**Pour la détermination du degré de  
maîtrise, il sera tenu compte des  
critères suivants :**

- ♦ la pertinence des choix et des techniques,
- ♦ le respect des consignes,
- ♦ le degré d'autonomie atteint.

## BACHELIER EN INFORMATIQUE DE GESTION (5IBD1)

### Initiation aux bases de données

**Pour atteindre le seuil de réussite,  
l'étudiant sera capable :**

*à partir d'un cahier des charges, en  
disposant d'une station informatique  
opérationnelle équipée d'un logiciel  
« Bases de données »,*

- ♦ de développer et de gérer une base de données sur un système de gestion de bases de données relationnelles et de manipuler des requêtes sous un langage tel que SQL,... dans des cas simples.

**Pour la détermination du degré de  
maîtrise, il sera tenu compte des critères  
suivants :**

- ♦ l'utilisation pertinente de toutes les techniques présentées dans le programme,
- ♦ le degré d'autonomie atteint.