Documentations complètes
Python3 : `https://docs.python.org/3/`
Nympy : `https://docs.scipy.org/doc/`
Matplotlib : `http://matplotlib.org/`
SQL : `http://sql.sh/`

3 Base de données - SQL Partie I

3.1 Base de données

3.1.1 Introduction

Historiquement l'informatique est née de la volonté d'ordonnancer des informations et de les traiter. Aujourd'hui on comprend sans peine la nécessité de stocker des données en très grande quantité.

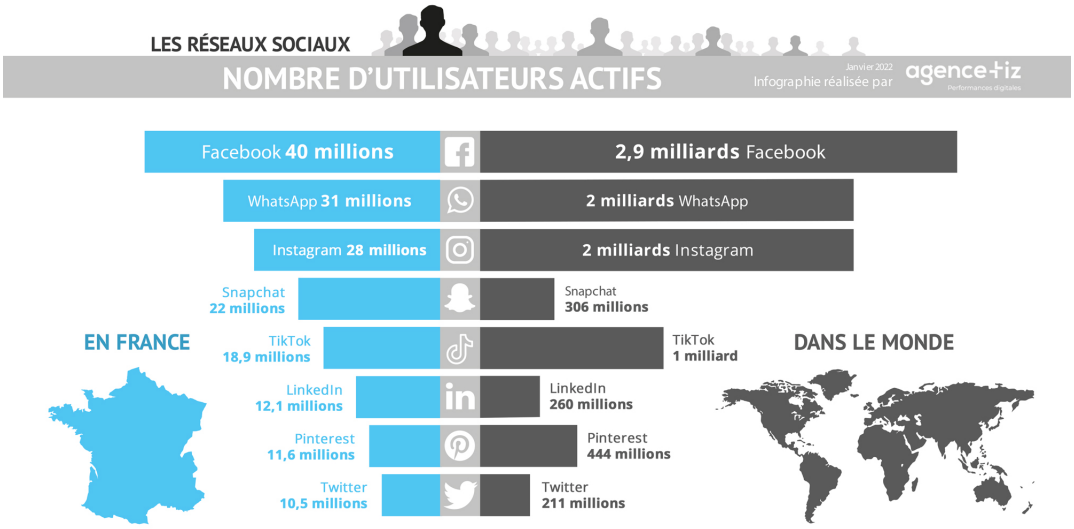

FIGURE 3.1 – Nombre d'utilisateurs par réseau social en 2022

Il faut stocker ces informations de telles sorte que l'on puisse :

1. Représenter ces données.
2. Permettre des recherches parmi ces données.
3. Assurer l'intégrité de ces données.

Jusqu'en 1970 environ, on manipulait directement les fichiers contenant les données. Ce qui présente de nombreux inconvénients :

— Il faut connaître l'organisation des différents, fichiers, dossiers etc... C'es-à-dire la hiérachie des informations.
— Il faut des programmes pour manipuler les fichiers, suivant leur type.
— Les fichiers sont particuliers en fonction du traitement.
— Il peut y avoir une grande redondance des données.

— Il y peut n'y avoir aucune standardisation des données.

— La rechercher des données est compliquée.

Jusqu'à présent nous avons utilisé des tableaux, des listes, en Python pour stocker des données. Illustrons les limites de cette représentation par un exemple.

La figure (3.1.1) représente la liste des étudiants de la PTSI-B ainsi que leur f-groupe de SI, leur LV1 et leur groupe de colle. Si l'on souhaite représenter notre classe sous forme de tableaux on peut s'y prendre de

Nom	Groupe SI	Langue	Groupe Colle
Erdös	A	ANG	1
Germain	A	ALL	1
Cantor	A	ANG	2
Abel	B	ALL	2
Cauchy	B	ESP	3
Gödel	B	ESP	3

FIGURE 3.2 – Liste de la PTSI-B

plusieurs manières :

1. En représentant toutes les informations :

```
PTSIB = [
("Erdös","A","ANG",1) , ("Germain","A","ALL",1),
 ("Cantor","A","ANG",2) , ("Abel","B","ALL",2),
 ("Cauchy","B","ESP",3) , ("Gödel","B","ESP",3)
]
```

2. En regroupant les étudiants par groupe de SI :

```
PTSIB = [
("A",[("Erdös","ANG",1),("Germain","ALL",1),("Cantor","ANG",2)]),
 ("B",[("Abel","ALL",2),("Cauchy","ESP",3),("Gödel","ESP",3)])
]
```

3. En regroupant les étudiants par groupe de colle :

```
PTSIB = [
(1,[("Erdös","A","ANG") , ("Germain","A","ALL")]),
 (2,[("Cantor","A","ANG") , ("Abel","B","ALL")]),
 (3,[("Cauchy","B","ESP") , ("Gödel","B","ESP")])
]
```

Exercice 3.1. Supposons que l'on choisisse la première représentation, écrire une fonction `liste_langue(classe,langu` qui retourne la liste des élèves de la classe `classe` étudiant la langue `langue`.

Solution.

Exercice 3.2. Supposons que l'on choisisse la seconde représentation, écrire une fonction `liste_groupeSI(classe,gro` qui retourne la liste des élèves du groupe `groupe`, et une fonction `recherche_groupeColle(classe,groupe)` qui retourne la liste des élèves du groupe de colle `groupe` de la classe `classe`.

On peut remarquer un certain nombre de choses :

1. La seconde recherche a été plus complexe que la première.
2. Si nous avions choisi la troisième représentation de la classe c'est la recherche de la liste d'un groupe de colle qui aurait été facilitée et la recherche des élèves d'un groupe de SI qui aurait été plus compliquée.
3. La recherche des élèves pratiquant une même langue est aussi complexe dans les deux cas, il aurait fallu représenter la classe en regroupant les élèves par langue pour faciliter cette recherche.
4. La meilleure solution semble de ne pas privilégier un lien d'appartenance à un sous-groupe ou à un autre.

3.1.2 L'organisation en BD

Une base de données est constituée de données, celles-ci sont collectées, stockées, mise à jour par un système de gestion de base de données (SGBD).

C'est aussi le SGBD qui nous permet d'accéder au données via des requêtes, toutes le opérations effectuées par le SGBD se font de manière transparente pour l'utilisateur qui n'a pas besoin de comprendre les mécanismes complexes mis en oeuvre par le SGBD pour effectuer ces tâches.

Définition 3.1.

— Base de données : Ensemble exhaustif d'informations, structurées, non redondantes et persistante, disponibles pour une ou plusieurs applications en parallèle.

— SGBD : Logiciel, qui sert d'interface entre l'utilisateur et la BD lui permettant de décrire, de créer/modifier (administrateur) ou d'interroger la BD (utilisateur ou client).

Dans ce cours nous nous intéresserons à un type particulier de base de données, les bases de données relationnelles (BDR). Dans ces bases les données sont organisées sous forme de tables reliées entre elles par des relations.

Pour interagir avec un SGBD on utilise le langage SQL : Structured Query Language développé dans les années 1970.

3.1.3 L'architecture trois-tiers

Les réseaux informatiques permettent la transmission d'informations entre deux machines, l'une est appelée client, c'est celle qui envoie des requêtes, l'autre est appelée serveur, c'est celle qui traite la requête. Cette organisation est appelée : architecture à deux niveaux.

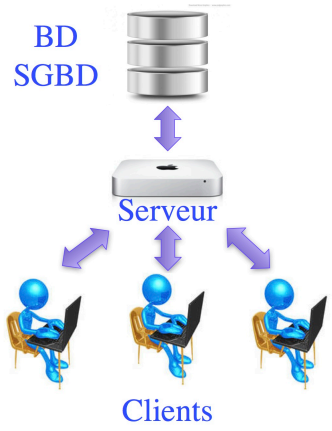

FIGURE 3.3 – Architecture à deux niveaux

Dans la pratique, c'est une autre architecture qui est utilisée, il s'agit de l'architecture trois-tiers[i]. Dans cette architecture, il y a un serveur de données et un serveur d'applications. Le serveur d'applications est placé entre les clients et le serveur de données.

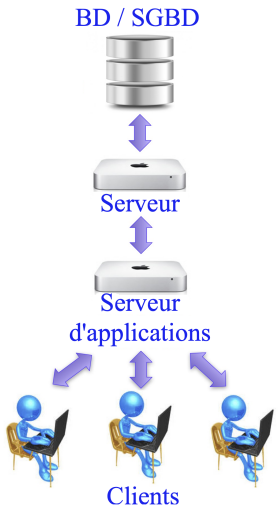

FIGURE 3.4 – Architecture à trois niveaux

Cette architecture offre de nombreux avantages :

1. L'allègement du poste de travail client (notamment vis-à-vis des architectures classiques client-serveur de données – typiques des applications dans un contexte Oracle/Unix) ;
2. Les données et les programmes sont indépendants (Indépendance logique et physique). Cette rupture du lien de propriété exclusive entre application et données permet de normaliser la base de données.
3. Limitation maximale de la redondance des données.
4. Standardisation des traitements généraux.

i. Comprendre « tier » comme niveau, en anglais

5. Accès par un langage déclaratif accessible aux non-initiés, l'utilisateur ne s'occupe ni du stockage, ni de l'archivage ni des algorithmes de recherche ou de tri.
6. Partageabilité et sécurité des données : grande simultanéité maximum lecture/écriture, sécurisation par authentification et droits d'accès (+ cryptage + restauration après panne).

3.2 Le model relationnel

3.2.1 Le model relationnel

Le model relationnel a été mis au point par Codd dans les années 1970 pour le compte d'I.B.M.. Ce modèle repose sur , qui dérivée de la théorie des ensemble, cette algèbre utilise des opérateurs ensemblistes et des opérateurs relationnels qui permettent de définir comment les informations sont liées entre elles et comment on peut y accéder. À partir d'un nombre basique d'opérations, il est ainsi possible d'accéder à des données qui font sens entre elles.

Dans ce modèle, les données sont organisées en table 2D qui représente ce que l'on appelle relation. On manipule ces tables grâce à l'algèbre relationnelle.

Prenons un exemple : ma *car_wish_list*.

Il s'agit d'une liste de voitures, elle constitue une table dans la base de données, elle est définie pas son nom : *car_wish_list* et sa relation : l'ensemble des champs qui la constitue (Marque, Modèle, Couleur, Année de production).

car_wish_list				
id	Marque	Modèle	Couleur	Année

La table va accueillir des enregistrements, un 4-uplet du produit cartésien Marque $\times$ Modèle $\times$ Couleur $\times$ Année.
Voici un exmple d'enregistrement :

car_wish_list				
id	Marque	Modèle	Couleur	Année
1	Porsche	918 - Spyder	Grise	2010

L'enregistrement est constitué d'attributs : Marque, Modèle, Couleur et Année.

Le nombre d'attributs d'une relation est son degré, la relation *car_wish_list* est de degré 4.
L'id est un numéro de ligne qui identifie l'enregistrement dans la table (nous y reviendrons plus tard).

Chaque attribut A prend ses valeurs dans un certain domaine qu'on note $\text{dom}(A)$:

$$\text{dom(Marque)} = \{\text{Porsche, Ferrari, Aston-Martin, Ford}\}$$
$$\text{dom(Modèle)} = \{\text{918 - Spyder}, \ldots, \text{Shelby GT500}\}$$
$$\text{dom(Couleur)} = \{\text{Gris, Jaune, Bleu, Rouge, Noir}\}$$
$$\text{dom(Année)} = \{1953, \ldots, 2015\}$$

— De manière générale, si on appelle $\mathcal{A}$ un ensemble fini d'attributs. On appelle schéma relationnel un n-uplet d'attributs $S = (A_1, \ldots, A_n)$ deux à deux distincts du produit cartésien $\mathcal{A}^n$.

Par exemple V=(Marque,Modèle,Couleur,Année) est un schéma relationnel.

car_wish_list				
id	Marque	Modèle	Couleur	Année
1	Porsche	918 - Spyder	Gris	2010
2	Porsche	911 - GT3 Cup	Bleu	2013
3	Ford	Mustang GT	Bleu	2011
4	Porsche	550 - Spyder	Gris	1953
5	Ferrari	GTE	Noir	2015
6	Ferrari	288 - GTO	Rouge	1984
7	Porsche	918 - Spyder	Noir	2012
8	Ferrari	288 - GTO	Jaune	1987
9	Ford	Mustang GT	Rouge	2014
10	Aston-Martin	DB5	Gris	1963
11	Ford	Shelby GT500	Bleu	2013

FIGURE 3.5 – La table *car_wish_list*

— Une relation R, ou une table, associé à un schéma relationnel est un ensemble fini d'enregistrements, c'est-à-dire de n-uplets du produit cartésien $D_1 \times \ldots \times D_n$, des domaines $D_1, \ldots, D_n$. On note $\#R$ le nombre d'enregistrements de R.

Par exemple *car_wish_list* est une relation du schéma V.

— On notera $B \in S$, pour dire que B est un attribut de S.

Par exemple Marque $\in V$, et Année $\in V$.

On notera aussi $S' \subset S$ pour dire que $S' \subset \{A_1, \ldots, A_n\}$.

Par exemple S'=(Marque,Modèle)$\subset V$.

— Si R est une relation de schéma S, on notera $R(S)$.

Par exemple on notera *car_wish_list*(V), pour signifier que la relation est associée au schéma V.

— Si e est un enregistrement de la relation R, de schéma S on note $e \in R(S)$.

— Si $e \in R(S)$, et si $A \in S$ est un attribut de S, on notera $e.A$ la composante de e suivant l'attribut A.

Par exemple si e = (Porsche,918 - Spyder,Gris,2010) alors e.Année = 2010 et e.Marque= Porsche.

— Si $e \in R(S)$ et $S' = (A_{i_1}, \ldots, A_{i_p}) \subset S$, on note $e(S') = (e.A_{i_1}, \ldots, e.A_{i_p})$.

Par exemple si e = (Porsche,918 - Spyder,Gris,2010) et si S' = (Marque,Couleur), alors : $e(S')$ =(Porsche,Gris).

3.2.2 L'algèbre relationelle

Pour utiliser une BD on utilise le langage SQL pour décrire des opérateurs relationnels que l'on applique à notre base de données. Nous allons décrire maintenant ces opérateurs.

L'ensemble des opérations que l'on effectue sur base de données sont les opérateurs de l'algèbre relationnelle.

Sélection simple et projection

Définition 3.2 (Sélection simple)**.**

Si $R(S)$ est une relation de schéma S, $A \in S$ et $a \in \text{dom}(A)$.
On appelle sélection d'une relation R suivant la condition $A = a$, et on note $\sigma_{A=a}(R)$, la relation obtenue en sélectionnant dans $R(S)$ les enregistrement e telles que $e.A = a$. On a donc :

$$\sigma_{A=a}(R) = \{e \in R \mid e.A = a\}.$$

Par exemple :

$\sigma_{Marque="Ford"}(car\_wish\_list)$

id	Marque	Modèle	Couleur	Année
1	Ford	Mustang GT	Bleu	2011
2	Ford	Mustang GT	Rouge	2014
3	Ford	Shelby GT500	Bleu	2013

Informatique 3.1 (SQL : Sélection)**.**

La sélection des éléments dont la marque est Ford :

```
SELECT *
FROM car_wish_list
WHERE Marque = "Ford"
```

Définition 3.3 (Projection)**.**

Si $R(S)$ est une relation de schéma S et $S' \subset S$.
On appelle projection d'une relation R suivant la condition S', et on note $\pi_{S'}(R)$, la relation :

$$\pi_{S'}(R) = \{e(S') \mid e \in R\}.$$

Remarque 3.1.

Le schéma de la relation $\pi_{S'}(R)$ est S'.

Par exemple :

$\pi_{\text{Marque,Couleur}}(car\_wish\_list)$		
id	Marque	Couleur
1	Porsche	Gris
2	Porsche	Bleu
3	Ford	Bleu
4	Ferrari	Noir
5	Ferrari	Rouge
6	Porsche	Noir
7	Ferrari	Jaune
8	Ford	Rouge
9	Aston-Martin	Gris

Informatique 3.2 (SQL : Projection)**.**

La projection suivant (Marque,Couleur) :

```
SELECT DISTINCT Marque , Couleur
FROM car_wish_list
```

Le mot clef `DISTINCT` permet d'éviter les doublons.

Remarque 3.2.

La sélection et la projections sont les deux opérations les plus naturelles, mais elles sont insuffisantes pour décrire des recherches dites , c'est-à-dire se faisant sur plusieurs tables.

Considérons la relation *specifications* suivante :

specifications			
id	Type	Puissance	Poids
1	918 - Spyder	887	1675
2	911 - GT3 Cup	435	1395
3	Mustang GT	412	1580
4	550 - Spyder	110	760
5	288 - GTO	400	1224
6	DB5	286	1465
7	Shelby GT500	671	1747

À l'aide de sélections et de projections on ne peut pas répondre à la question : Quelles sont les marques qui offrent un modèle dont la puissance est supérieure à 400 ch ?

Opérations Ensemblistes

Définition 3.4 (Définition : Union - Intersection et différence)**.**

Soient $R_1(S)$ et $R_2(S)$ deux relations de même schéma S.
On définit :

- La réunion, notée $R_1(S) \cup R_2(S)$, la relation de schéma S dont les enregistrements sont dans $R_1(S)$ ou dans $R_2(S)$.
- L'intersection, notée $R_1(S) \cap R_2(S)$, la relation de schéma S dont les enregistrements sont dans $R_1(S)$ et dans $R_2(S)$.
- La différence, notée $R_1(S) - R_2(S)$, la relation de schéma S dont les enregistrements sont dans $R_1(S)$ mais pas dans $R_2(S)$.

Exercice 3.3. Décrire $Liste1 \cup Liste2$, $Liste1 \cap Liste2$ et $Liste1 - Liste2$.

Liste1		
id	Titre	Auteur
1	Ainsi parlait Zarathoustra	Frédéric Nietzsche
2	Le cas Wagner	Frédéric Nietzsche
3	Le Traité des cinq roues	Musashi Miyamoto

et

Liste2		
id	Titre	Auteur
1	Par-delà bien et mal	Frédéric Nietzsche
2	Le cas Wagner	Frédéric Nietzsche

Solution.

Informatique 3.3 (SQL : Réunion, intersection et différence)**.**

— La réunion de R_1 et R_2 :

```
SELECT * FROM R1 UNION SELECT  * FROM R2
```

— L'intersection de R_1 et R_2 :

```
SELECT * FROM R1 INTERSECT SELECT  * FROM R2
```

— La différence de R_1 et R_2 :

```
SELECT * FROM R1 EXCEPT SELECT  * FROM R2
```

Sélections composées

Grâce aux opérations ensemblistes on peut exprimer des sélections suivant des conditions complexes.

Si C_1 et C_2 sont deux conditions et R une relation posons $R_1 = \sigma_{C_1}(R)$ et $R_2 = \sigma_{C_2}(R)$ alors :

— les éléments de R vérifiant C_1 et C_2 sont ceux de $R_1 \cap R_2$.

— les éléments de R vérifiant C_1 ou C_2 sont ceux de $R_1 \cup R_2$.

— les éléments de R ne vérifiant pas C_1 sont ceux de $R - R_1$.

Exercice 3.4. Comment obtenir les voitures de la relation *car_wish_list* qui sont des Porsches ou des Ford bleues qui ne sont pas de 2011 ?

Sélections par comparaison

— Lorsque le domaine d'un attribut le permet on peut aussi faire des sélections simples suivant des conditions du type « Année $<$ 2000 ».

```
SELECT *
FROM car_wish_list
WHERE Annee < 2000
```

Il aussi possible d'utiliser `>`, `>=` et `<=`.

— L'opérateur ! = signifie différent et peut s'utiliser ainsi :

```
SELECT *
FROM car_wish_list
WHERE Marque != "Porsche"
```

— On peut aussi comparer différents attributs entre eux.

```sql
SELECT *
FROM specifications
WHERE Poids / Puissance < 2.5
```

Si je veux juste les modèles :

```sql
SELECT Modele
FROM specifications
WHERE Poids / Puissance < 2.5
```

— D'autres opérateurs encore sont disponibles :

1. `IN` pour savoir si une valeurs appartient à une liste de valeurs.

```sql
SELECT *
FROM specifications
WHERE Poids IN {760, 1224}
```

2. `BETWEEN` pour savoir si une valeur est dans un intervalle de valeurs.

```sql
SELECT *
FROM specifications
WHERE Puissance BETWEEN 100 AND 300
```

3. `LIKE` pour spécifier le début, milieu ou fin d'un mot dans notre recherche.

```sql
SELECT *
FROM car_wish_list
WHERE Marque LIKE "%P"
```

Voir ici pour plus d'exemples : https://sql.sh/cours/where/like.

— Evidemment il y a les connecteurs logiques `AND`, `OR` et `NOT`.

Exercice 3.5. On considère la base formée des deux tables *car_wish_list* et *specifications*.
Traduire les requêtes suivantes à l'aide d'opérateurs relationnels, puis en SQL :

1. Obtenir tous les marques de la liste.
2. Obtenir le modèle, l'année et la couleur des Porsches.
3. Obtenir la puissance des voitures ayant un poids < 1300.
4. Obtenir le modèle et la puissance des voitures ayant une puissance < 400 .
5. Obtenir le modèle et la puissance des voitures ayant un rapport poids/puissance < 3.

Renommage

Définition 3.5.

Définition : Renommage Soit $S = (A_1, \ldots, A_n)$ un schéma, A_i un attribut de S et B un attribut de même domaine que A_i.
On appelle renommage l'opération qui consiste à remplacer A_i par B, on la note :

$$\rho_{A_i \leftarrow B}(S) = (A_1, \ldots, A_{i-1}, B, A_{i+1}, \ldots, A_n).$$

On notera aussi :

$$\rho_{A_{i_1},\ldots,A_{i_p} \leftarrow B_{i_1},\ldots,B_{i_p}}(S) = \rho_{A_{i_1} \leftarrow B_{i_1}} \circ \ldots \circ \rho_{A_{i_p} \leftarrow B_{i_p}}(S),$$

pour $(i_1, \ldots, i_p)$ un p-uplet de nombre distincts de $[\![1, n]\!]$.

Remarque 3.3.

1. L'opération de renommage est a priori définie sur un schéma S, mais on peut considérer qu'elle opère directement sur une relation R en substituant à son schéma un schéma obtenu par renommage, on note alors :

$$\rho_{A_{i_1},\ldots,A_{i_p} \leftarrow B_{i_1},\ldots,B_{i_p}}(R(S)) = R\left(\rho_{A_{i_1},\ldots,A_{i_p} \leftarrow B_{i_1},\ldots,B_{i_p}}(S)\right).$$

2. L'opération de renommage ne modifie pas les enregistrements de la relation, juste le nom des attributs.

Informatique 3.4 (SQL : Renommage)**.**

Si R est une relation de schéma $S = (A_1, \ldots, A_n, B_1, \ldots, B_m)$ et que l'on souhaite renommer les A_i en C_i en SQL :

```
SELECT A1 AS C1 , A2 AS C2 , . . . , An AS Cn , B1 , . . . , Bn
FROM R
```

Remarquer qu'il est nécessaire d'indiquer tous les attributs !

3.3 Exercices

Exercice 3.6. Le service de réservation d'un hôtel utilise un logiciel qui accède à une base de données qui contient toutes les informations sur les chambres, les clients, les séjours... La base de données de cet hôtel est constituée de trois tables (ou entités) :

chambres (id : `int`, numero : `int`, etage : `int`, nb_places : `int`, equip : `str`, tarif : `float`)
sejours (id : `int`, date_a : `date`, date_d : `date`, id_chambre : `int`, id_client : `int`)
clients (id : `int`, nom : `str`, prenom : `str`, adresse : `str`, cp : `int`, ville : `str`, email : `str`, tel : `str`)

1. Monsieur Bob Morane arrive le lundi le 17 septembre 2018 à l'accueil de l'hôtel et souhaite prendre sa chambre. Donner la requête SQL permettant de connaître le numéro et l'étage de la chambre qui lui a été attribuée.
2. Le réceptionniste curieux veut connaître tous les séjours que Monsieur Bob Morane a passé dans l'hôtel. Donner la requête SQL permettant de satisfaire sa curiosité.
3. Encore plus curieux, il veut connaître tous les séjours de Monsieur Bob Morane effectués en 2017. Donner la requête SQL.
4. Une famille de quatre personnes appelle pour réserver une chambre du lundi 17 septembre 2018 au dimanche suivant. Donner la requête SQL permettant de connaître le numéro et l'étage des chambres libres.
5. Plus précisément, la famille souhaiterait avoir une télévision. Donner la requête SQL permettant de connaître le numéro et l'étage des chambres libres.

Exercice 3.7. Une base de données qui recense des informations sur des oeuvres cinématographiques est composée de trois tables :

acteurs (id : `int`, prénom : `text`, nom : `text`)
castings (id_acteur : `int`, id_films : `int`, role : `text`)
films (id : `int`, titre : `text`, annee : `int`)

1. Déterminer la requête SQL permettant de récupérer toutes les informations des acteurs dont le prénom est « Bob ».
2. Déterminer la requête SQL permettant de récupérer les prénoms et noms des acteurs dont le prénom est « Bob ».
3. Déterminer la requête SQL permettant de récupérer l'identifiant des films dans lesquels l'acteur « Bob Morane » joue.
4. Déterminer la requête SQL permettant de récupérer les titres et années des films dans lesquels l'acteur « Bob Morane » joue.
5. Déterminer la requête SQL permettant de récupérer les identifiants des films dans lesquels jouent les acteurs « Bob Morane » et « Sophie Randmo » ensemble.
6. Modifier la requête précédente pour obtenir les titres et années des films dans lesquels jouent les acteurs « Bob Morane » et « Sophie Randmo » ensemble.
7. Déterminer la requête SQL permettant de récupérer les acteurs (référencés) jouant dans le films Brasil.

3.4 Corrections

Corrigé 3.1 Le code :

```
def recherche_langue(classe,langue):
    eleves = []
    for eleve in classe:
        if eleve[2] == langue:
            eleves.append(eleve[0])
    return eleves
```

```
>>> recherche_langue("ALL",PTSIB)
['Germain', 'Abel']
```

Corrigé 3.2 Le code :

```
def liste_groupeSI(classe,grpe):
    eleves = []
    for groupe in classe:
        if groupe[0] == groupe:
            return groupe[1]
```

```
>>> liste_groupeSI(PTSIB,"B")
[('Abel', 'ALL', 2), ('Cauchy', 'ESP', 3), ('Godel', 'ESP', 3)]
```

```
def liste_groupeColle(classe,num_grpe):
    eleves = []
    for groupe in classe:
        for eleve in groupe[1]:
            if eleve[2] == num_grpe:
                eleves.append(eleve[0])
    return eleves
```

```
>>> liste_groupeColle(PTSIB,2)
['Cantor', 'Abel']
```

Corrigé 3.4 En SQL :

```
SELECT *
FROM car_wish_list
WHERE (Marque = "Ford" OR Marque = "Porsche") AND Couleur = "Bleu"
EXCEPT
SELECT *
FROM car_wish_list
WHERE Annee = 2011
```

Ce qui retourne :

R				
id	Marque	Modèle	Couleur	Année
1	Porsche	911 - GT3 Cup	Bleu	2013
2	Ford	Shelby GT500	Bleu	2013

Corrigé 3.5 1. $\pi_{\text{Marque}}(\text{car\_wish\_list})$.

```sql
SELECT Marque
FROM car_wish_list
```

2. $\pi_{\text{Modele, Annee, Couleur}} \circ \sigma_{\text{Marque} = \text{"Porsche"}}(\text{car\_wish\_list})$.

```sql
SELECT Modèle, Année, Couleur
FROM car_wish_list
WHERE Marque = "Porsche"
```

3. $\pi_{\text{Puissance}}(\sigma_{\text{Poids} < 1300}(\text{specifications}))$.

```sql
SELECT Puissance
FROM specifications
WHERE Poids < 1300
```

4. $\sigma_{\text{Puissance} < 400}(\pi_{\text{Modèle,Puissance}}(\text{spécification}))$.

```sql
SELECT Modèle, Puissance
FROM specifications
WHERE Puissance < 400
```

5. $\pi_{\text{Modèle,Puissance}}(\sigma_{\text{Poids / Puissance} < 3}(\text{spécifications}))$

```sql
SELECT Modèle, Puissance
FROM specifications
WHERE Poids/Puissance < 3
```

Corrigé 3.6 Les requêtes :

1.

```sql
SELECT numero , etage
FROM chambres
WHERE id = (SELECT id_chambre
FROM sejours
WHERE  id_client = (SELECT id FROM clients WHERE nom = " Morane " AND prenom = " Bob " ) AND  date_a = " 2018-09-17 " ))
```

2.

```sql
SELECT *
FROM sejours
WHERE  id_client = (SELECT id FROM clients WHERE nom = " Morane " AND prenom = " Bob ")
```

3.

```
SELECT *
FROM sejours
WHERE  id_client = (SELECT id FROM clients WHERE nom = " Morane " AND prenom = " Bob ")
AND date_a>= " 2017 -01 -01 "
AND date_d <= " 2017 -12 -31 "
```

4.

```
SELECT numero , etage
FROM chambres
WHERE nb_places >= 4
AND id NOT IN (SELECT id_chambre FROM sejours WHERE
date_a BETWEEN " 2018-09-17 " AND " 2018-09- 23" OR
date_d BETWEEN " 2018-09-17 " AND " 2018-09-23 " OR (date_a <= " 2018-09-17 " AND date_d >= " 2018-09-23 " ) )
```

5.

```
SELECT numero , etage
FROM chambres
WHERE id NOT IN (SELECT id_chambre FROM sejours WHERE
date_a BETWEEN " 2018-09-17 " AND " 2018-09- 23" OR
date_d BETWEEN " 2018-09-17 " AND " 2018-09-23 " OR (date_a <= " 2018-09-17 " AND date_d >= " 2018-09-23 " ) )
AND equip = " TV " AND nb_places >= 4
```

Corrigé 3.7 Les requêtes :

1.

```
SELECT * FROM acteurs WHERE nom = "Bob"
```

2.

```
SELECT prenom, nom
FROM acteurs
WHERE prenom = "Bob"
```

3.

```
SELECT id_film
FROM casting
WHERE id_acteur = (SELECT id FROM acteurs WHERE prenom = "Bob" and nom = "Morane")
```

4.

```
SELECT titre , annee
FROM films
WHERE id in (SELECT id_film FROM casting WHERE id_acteur = (SELECT id FROM acteurs WHERE prenom = "Bob" and nom = "Morane"))
```

5.

```
SELECT id_film
FROM casting
WHERE id_acteur = (SELECT id FROM acteurs WHERE prenom = "Sophie" and nom = "Randmo")
INTERSECT
SELECT id_film
FROM casting
WHERE id_acteur = (SELECT id FROM acteurs WHERE prenom = "Bob" and nom = "Morane")
```

6.

```
SELECT titre, annee
FROM films
WHERE id in (SELECT id_film
FROM casting
WHERE id_acteur = (SELECT id FROM acteurs WHERE prenom = "Sophie" and nom = "Randmo")
```

```
INTERSECT
SELECT id_film
FROM casting
WHERE id_acteur = (SELECT id FROM acteurs WHERE prenom = "Bob" and nom = "Morane"))
```

7.

```
SELECT nom, prenom
FROM acteurs
WHERE id IN (SELECT DISTINCT id_acteur FROM casting
WHERE id_film = (SELECT id FROM films WHERE titre = "Brasil"))
```

