Informatique 3OC

A. Ridard

Bases de données

Gymnase Yverdon

canton de vaud

Table des matières

I. Introduction . . . 4
1. Excel . . . 4
2. SGBD . . . 5
II. Modèle relationnel . . . 5
1. Contraintes d'intégrité . . . 5
2. Deux représentations . . . 7
III. Définition des données . . . 8
1. Création et destruction de tables . . . 8
2. Implémentation des contraintes d'intégrité . . . 9
IV. Modification des données . . . 9
1. Insertion . . . 9
2. Mise à jour et suppression . . . 10
V. Consultation des données . . . 12
1. Projection, restriction (filtre) et tri . . . 12
2. Jointure . . . 14
3. Fonctions de groupe . . . 15
4. Regroupement . . . 15
VI. Interaction entre un SGBD et un programme . . . 19

I. Introduction

1. Excel

Tri et filtre

A l'aide d'Excel, afficher :

1. les compagnies, dans l'ordre alphabétique.
2. les compagnies françaises.
3. les pilotes, dans l'ordre décroissant du nombre d'heures de vol.
4. les pilotes de la compagnie 1 ayant moins de 2000 heures de vol.
5. les pilotes ayant plus de 1000 heures de vol, dans l'ordre alphabétique.

Peut-on afficher les pilotes travaillant pour une compagnie low cost ?

Jointure

A l'aide d'Excel, afficher :

1. la jointure entre Pilote et Compagnie.
2. les pilotes travaillant pour une compagnie low cost.
3. les pilotes travaillant pour Air France, dans l'ordre alphabétique.

Stocker les données dans des tables différentes permet d'éviter la **redondance** pouvant être source d'**incohérence** dans les données.

Peut-on afficher le nombre de pilotes travaillant pour une compagnie low cost ?

Fonctions de groupe

A l'aide d'Excel, afficher :

1. le nombre de pilotes travaillant pour une compagnie low cost.
2. la moyenne du nombre d'heures de vol.
3. le minimum du nombre d'heures de vol pour une compagnie française.

Peut-on afficher, pour chaque compagnie, le nombre de pilote ?

Regroupement

A l'aide d'Excel, afficher :

1. pour chaque compagnie désignée par son id, le nombre de pilote.
2. pour chaque compagnie désignée par son id, la somme des heures de vol.
3. pour chaque compagnie désignée par son nom, la somme des heures de vol.

Peut-on afficher la(les) compagnie(s) ayant le maximum de pilotes ?

2. SGBD

Comme on le verra dans les sections suivantes, un Système de Gestion de Bases de Données (SGBD) permet de définir, de modifier et de consulter des données, mais il assure aussi le contrôle des données en garantissant la **confidentialité** et l'**intégrité**.

Il garantit la confidentialité des données en gérant :
- l'accès aux données [1] (en consultation)
- le niveau externe [2]

Il garantit l'intégrité des données :
- la **sécurité** en se protégeant des attaques (volontaires)
- la **cohérence** en se préservant des erreurs (involontaires)

La sécurité est assurée en gérant :
- l'accès aux données [3] (en modification)

La cohérence est assurée en gérant :
- les accès concurrents [4]
- les **contraintes d'intégrité**

Qu'entendons-nous par « contraintes d'intégrité » ?

II. Modèle relationnel

1. Contraintes d'intégrité

Le modèle relationnel [5] est un modèle mathématique permettant de stocker des données dans des relations (tables) composées d'attributs (colonnes) respectant des propriétés logiques (contraintes d'intégrité).

Les contraintes d'intégrité sont donc des propriétés logiques, préservées à tout instant par la base de données, garantissant ainsi la cohérence des données.

On distingue quatre types de contraintes d'intégrité :
- Contraintes de domaine
- Contraintes d'entité
- Contraintes de référence
- Contraintes utilisateurs

Les contraintes de domaine doivent :
- permettre de représenter toutes les valeurs possibles pour une colonne
- limiter autant que possible la saisie de valeurs illégitimes

Concrètement, cela revient à choisir pour chaque colonne un **type** [6] voire un ensemble de valeurs plus précis (entier compris entre 1 et 6).

---

[1]. avec les *utilisateurs*, les *rôles* et les *privilèges*
[2]. avec les *vues* qui agissent comme des « fenêtres »
[3]. avec les *utilisateurs*, les *rôles* et les *privilèges*
[4]. avec les *transactions*
[5]. défini en 1970 par l'informaticien britannique Edgar Frank Codd alors qu'il était employé par IBM
[6]. entier, décimal, texte, booléen, date

Indiquer les contraintes de domaine pour la table suivante.

idComp	nomComp	pays	estLowCost
1	Air France	France	0
2	Corsair	France	0
3	EasyJet	Angleterre	1
4	Swiss	Suisse	0
5	Ryanair	Irelande	1

Les contraintes d'entité permettent d'identifier sans ambiguïté (de manière unique) chaque ligne d'une table.

Concrètement, cela revient à choisir pour chaque table une colonne [7] ayant une valeur unique pour chaque ligne. Cette colonne « privilégiée » est appelée **clé primaire**.

La clé primaire d'une table permet aussi de servir de référence dans une autre.

Indiquer la clé primaire de la table suivante.

idComp	nomComp	pays	estLowCost
1	Air France	France	0
2	Corsair	France	0
3	EasyJet	Angleterre	1
4	Swiss	Suisse	0
5	Ryanair	Irelande	1

Les contraintes de référence permettent de « faire des liens » entre les tables d'une base de données.

Concrètement, une colonne d'une table qui porte la même information que la clé primaire d'une autre table est appelée une **clé étrangère**.

Indiquer une clé étrangère dans la table suivante, et préciser à quelle clé primaire elle fait référence.

idPilote	nomPilote	nbHVol	compPil
1	Tintin	1500	1
2	Haddock	450	3
3	Dupond	450	5
4	Dupont	3000	1
5	Tournesol	900	4
6	Castafiore	900	
7	Rastapopoulos	3000	4

Cette clé étrangère :

- évite d'ajouter des valeurs fictives ne correspondant à aucune compagnie
- empêche de supprimer des compagnies utilisées dans la table Pilote

idPilote	nomPilote	nbHVol	compPil
1	Tintin	1500	1
2	Haddock	450	3
3	Dupond	450	5
4	Dupont	3000	1
5	Tournesol	900	4
6	Castafiore	900	
7	Rastapopoulos	3000	4

idComp	nomComp	pays	estLowCost
1	Air France	France	0
2	Corsair	France	0
3	EasyJet	Angleterre	1
4	Swiss	Suisse	0
5	Ryanair	Irelande	1

[7]. voire plusieurs

Cette clé étrangère permet aussi de « joindre » les tables contenant la clé étrangère d'une part, et la clé primaire référencée d'autre part.

idPilote	nomPilote	nbHVol	compPil	idComp	nomComp	pays	estLowCost
1	Tintin	1500	1	1	Air France	France	0
2	Haddock	450	3	3	EasyJet	Angleterre	1
3	Dupond	450	5	5	Ryanair	Irelande	1
4	Dupont	3000	1	1	Air France	France	0
5	Tournesol	900	4	4	Swiss	Suisse	0
6	Castafiore	900					
7	Rastapopoulos	3000	4	4	Swiss	Suisse	0

Les contraintes utilisateurs (ou métier) sont toutes les autres contraintes !

Dans les cas simples, elles sont directement implémentées lors de la création des tables, avec un *CHECK*, mais le plus souvent elles nécessitent de la programmation...

2. Deux représentations

Pour représenter le modèle relationnel (la base de données), on peut utiliser :

- une forme textuelle appelée **schéma relationnel**
- une forme graphique appelée **diagramme relationnel**

Voici le schéma relationnel « allégé [8] » des deux tables précédentes :

- Compagnie (idComp, nomComp, pays, estLowCost)
- Pilote = (idPilote, nomPilote, nbHVol, #compPil = @Compagnie(idComp))

Compléter ce qui précède afin d'obtenir le schéma relationnel de **bd_aviation**.

Voici le diagramme relationnel des deux tables précédentes :

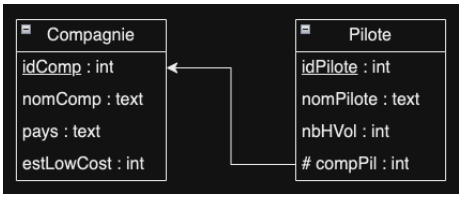

Diagramme relationnel

Représenter le diagramme relationnel de **bd_aviation** avec draw.io.

[8]. On omet les types pour ne pas alourdir la description

III. Définition des données

Le modèle relationnel est un modèle logique (mathématique), il est implémenté[9] physiquement par un SGBD relationnel[10].

Le langage SQL[11] utilisé est un langage *déclaratif* avec lequel on « déclare » les résultats que l'on souhaite obtenir, pas la manière de les « calculer ».

1. Création et destruction de tables

CREATE

```
CREATE TABLE Compagnie
     (
     idComp INTEGER PRIMARY KEY,
     nomComp TEXT,
     pays TEXT,
     estLowCost INTEGER CHECK( estLowCost IN (0,1) )
     )
;

CREATE TABLE Pilote
     (
     idPilote INTEGER PRIMARY KEY,
     nomPilote TEXT,
     nbHVol INTEGER,
     compPil INTEGER REFERENCES Compagnie(idComp)
     )
;
```

L'ordre de création est important

Il faut tenir compte des contraintes de référence.

Lorsqu'une table est créée, il est impossible de la créer à nouveau, donc pour pouvoir exécuter le script de création plusieurs fois[12], on commence par des destructions.

DROP

```
DROP TABLE Pilote;
DROP TABLE Compagnie;
```

L'ordre de destruction est important

Là encore, les contraintes de référence imposent l'ordre inverse de création.

[9]. anglicisme signifiant mis en œuvre
[10]. Nous utiliserons SQLite qui permet de faire l'économie d'un serveur
[11]. Structured Query Language
[12]. utile pour débugger par exemple

2. Implémentation des contraintes d'intégrité

Lors de la création d'une table, les déclarations de colonne doivent être séparées par une virgule et contenir les éléments suivants :

nom type autres contraintes éventuelles

Voici les directives permettant d'implémenter les contraintes :

- de domaine : type (INTEGER, REAL ou TEXT) + CHECK au besoin
- d'entité : PRIMARY KEY
- de référence : REFERENCES
- utilisateurs : CHECK

Bonne pratique

Le SQL est insensible à la casse, mais on recommande les règles suivantes :

- les noms de tables en minuscules avec la première lettre en majuscule
- les noms de colonnes en minuscules
- les mots-clés du langage SQL en majuscules

.

Définition

Écrire[a] le script de création de bd_aviation, puis l'exécuter[b].

a. avec *VS Code*
b. avec *DB Browser*

IV. Modification des données

Maintenant que la structure[13] de la base est définie, nous allons pouvoir l'utiliser.

1. Insertion

INSERT

```
-----------------------------------------------------
--  Table Compagnie
-----------------------------------------------------

INSERT INTO Compagnie VALUES (1,'Air France','France',0);
INSERT INTO Compagnie VALUES (2,'Corsair','France',0);
INSERT INTO Compagnie VALUES (3,'EasyJet','Angleterre',1);
INSERT INTO Compagnie VALUES (4,'Swiss','Suisse',0);
INSERT INTO Compagnie VALUES (5,'Ryanair','Irelande',1);

-----------------------------------------------------
--  Table Pilote
-----------------------------------------------------

INSERT INTO Pilote VALUES (1,'Tintin',1500,1);
INSERT INTO Pilote VALUES (2,'Haddock',450,3);
INSERT INTO Pilote VALUES (3,'Dupond',450,5);
INSERT INTO Pilote VALUES (4,'Dupont',3000,1);
INSERT INTO Pilote VALUES (5,'Tournesol',900,4);
INSERT INTO Pilote VALUES (6,'Castafiore',900,null);
INSERT INTO Pilote VALUES (7,'Rastapopoulos',3000,4);
```

[13]. tables, colonnes, contraintes d'intégrité

L'ordre d'insertion est important

Il faut tenir compte des contraintes de référence.

Lorsqu'une ligne est insérée, il est impossible de l'insérer à nouveau [14], donc pour pouvoir exécuter le script d'insertion plusieurs fois [15], on commence par des suppressions.

DELETE

```
DELETE FROM Pilote;
DELETE FROM Compagnie;
```

L'ordre de suppression est important

Là encore, les contraintes de référence imposent l'ordre inverse d'insertion.

Insertion

Exécuter le script de remplissage de bd_aviation.

2. Mise à jour et suppression

Pour illustrer les modifications, commençons par insérer une nouvelle compagnie.

```
INSERT INTO Compagnie VALUES (1, 'Voloteo', 'Espagne', 0) ;
```

RESULT

```
L'exécution s'est terminée avec des erreurs.
Résultat : UNIQUE constraint failed: Compagnie.idComp
À la ligne 1 :
INSERT INTO Compagnie VALUES (1, 'Voloteo', 'Espagne', 1);
```

Recommençons en respectant cette fois la contrainte d'entité.

```
INSERT INTO Compagnie VALUES (6, 'Voloteo', 'Espagne', 0) ;
```

RESULT

```
L'exécution s'est terminée sans erreur.
Résultat : Requête exécutée avec succès. Elle a pris 0 ms , 1 enregistrements affectés
À la ligne 2 :
INSERT INTO Compagnie VALUES (6, 'Voloteo', 'Espagne', 0);
```

En fait, il s'agit de la compagnie low cost Volotea, corrigeons les données.

```
UPDATE Compagnie SET nomComp = 'Volotea', estLowCost = 1
WHERE idComp = 6 ;
```

[14]. à cause de la contrainte d'entité
[15]. utile pour débugger par exemple

RESULT

idComp	nomComp	pays	estLowCost
Filtre	Filtre	Filtre	Filtre
1	Air France	France	0
2	Corsair	France	0
3	EasyJet	Angleterre	1
4	Swiss	Suisse	0
5	Ryanair	Irelande	1
6	Volotea	Espagne	1

Insérons une deuxième compagnie espagnole.

```
INSERT INTO Compagnie VALUES (7, 'Iberia', 'Espagne', 0) ;
```

RESULT

idComp	nomComp	pays	estLowCost
Filtre	Filtre	Filtre	Filtre
1	Air France	France	0
2	Corsair	France	0
3	EasyJet	Angleterre	1
4	Swiss	Suisse	0
5	Ryanair	Irelande	1
6	Volotea	Espagne	1
7	Iberia	Espagne	0

Pour revenir aux données initiales, supprimons les deux dernières lignes.

```
DELETE FROM Compagnie
WHERE pays = 'Espagne' ;
```

RESULT

idComp	nomComp	pays	estLowCost
Filtre	Filtre	Filtre	Filtre
1	Air France	France	0
2	Corsair	France	0
3	EasyJet	Angleterre	1
4	Swiss	Suisse	0
5	Ryanair	Irelande	1

Pour chaque question, exécuter l'instruction et expliquer le message d'erreur.

1.

```
INSERT INTO Compagnie VALUES (6, Volotea, 'Espagne', 2) ;
```

2.

```
INSERT INTO Compagnie VALUES (6, 'Volotea', 'Espagne', 2) ;
```

3.

```
INSERT INTO Pilote VALUES (8, 'Ridard', 1000, 6) ;
```

4.

```
DELETE FROM Compagnie
WHERE idComp = 3 ;
```

Écrire un script permettant de tester, au niveau de la table *Avion* :

- la contrainte de domaine sur *leTypeAvion*
- la contrainte d'entité
- la contrainte de référence sur *compAv* à l'aide d'une insertion
- la contrainte de référence sur *leTypeAvion* à l'aide d'une suppression

V. Consultation des données

Pour illustrer cette section, reprenons les questions traitées sur Excel.

1. Projection, restriction (filtre) et tri

Afficher les compagnies, dans l'ordre alphabétique.

On peut trier avec ORDER BY

```
SELECT *
FROM Compagnie
ORDER BY nomComp
;
```

Afficher les compagnies françaises.

On peut filter avec WHERE

```
SELECT *
FROM Compagnie
WHERE pays = 'France'
;
```

Afficher les pilotes, dans l'ordre décroissant du nombre d'heures de vol.

On peut trier dans l'ordre décroissant avec DESC

```
SELECT *
FROM Pilote
ORDER BY nbHVol DESC
;
```

Afficher le nom et le nombre d'heures de vol des pilotes, dans l'ordre décroissant du nombre d'heures de vol.

On peut projeter sur certaines colonnes uniquement

```
SELECT nomPilote, nbHVol
FROM Pilote
ORDER BY nbHVol DESC
;
```

Afficher le nom des pilotes, dans l'ordre décroissant du nombre d'heures de vol.

On n'est pas obligé de projeter sur la colonne de tri

```
SELECT nomPilote
FROM Pilote
ORDER BY nbHVol DESC
;
```

Afficher les pilotes de la compagnie 1 ayant moins de 2000 heures de vol.

On peut filtrer avec une condition combinée

```
SELECT *
FROM Pilote
WHERE compPil = 1 AND nbHVol < 2000
;
```

Afficher le nom des pilotes de la compagnie 1 ayant moins de 2000 heures de vol.

On n'est pas obligé de projeter sur les colonnes de filtre

```
SELECT nomPilote
FROM Pilote
WHERE compPil = 1 AND nbHVol < 2000
;
```

Afficher les pilotes ayant plus de 1000 heures de vol, dans l'ordre alphabétique.

On peut filtrer et trier

```
SELECT *
FROM Pilote
WHERE nbHVol > 1000
ORDER BY nomPilote
;
```

L'ordre des directives est important

SELECT, FROM, WHERE, ... , ORDER BY

2. Jointure

Afficher la jointure entre Pilote et Compagnie.

On peut joindre deux tables avec JOIN ... ON ...

```
SELECT *
FROM Pilote JOIN Compagnie ON compPil = idComp
;
```

Afficher les pilotes travaillant pour une compagnie low cost.

On peut filtrer une jointure

```
SELECT *
FROM Pilote JOIN Compagnie ON compPil = idComp
WHERE estLowCost = 1
;
```

Afficher les pilotes travaillant pour Air France, dans l'ordre alphabétique.

On peut filtrer et trier une jointure

```
SELECT *
FROM Pilote JOIN Compagnie ON compPil = idComp
WHERE nomComp = 'Air France'
ORDER BY nomPilote
;
```

Consultation 1

1. Afficher les identifiants des compagnies possédant un Boeing.
2. Pour chaque pilote désigné par son identifiant, afficher les types d'avions pour lesquels il est qualifié.
3. Pour chaque compagnie désignée par son identifiant, afficher les informations sur ses avions.
4. Pour chaque pilote désigné par son nom, afficher les types d'avions pour lesquels il est qualifié.
5. Pour chaque compagnie désignée par son nom, afficher les informations sur ses avions.
6. Afficher les noms des compagnies possédant un A320.
7. Afficher les noms des compagnies possédant un Airbus.
8. Afficher les noms des pilotes qualifiés pour des avions de plus de 300 passagers.

3. Fonctions de groupe

Afficher le nombre de pilotes travaillant pour une compagnie low cost.

On peut compter le nombre de lignes avec COUNT(*)

```
SELECT COUNT(*)
FROM Pilote JOIN Compagnie ON compPil = idComp
WHERE estLowCost = 1
```

On peut nommer la colonne calculée avec AS

```
SELECT COUNT(*) AS nb_pilotes_lowCost
FROM Pilote JOIN Compagnie ON compPil = idComp
WHERE estLowCost = 1
;
```

Afficher la moyenne du nombre d'heures de vol.

On peut calculer la moyenne d'une colonne avec AVG

```
SELECT AVG(nbHVol)
FROM Pilote
;
```

Afficher le minimum du nombre d'heures de vol pour une compagnie française.

On peut calculer le minimum d'une colonne avec MIN

```
SELECT MIN(nbHVol)
FROM Pilote JOIN Compagnie ON compPil = idComp
WHERE pays = 'France'
;
```

4. Regroupement

Afficher, pour chaque compagnie désignée par son id, le nombre de pilote.

On peut regrouper des lignes et calculer par groupe avec GROUP BY

```
SELECT compPil, COUNT(*)
FROM Pilote
GROUP BY compPil
;
```

Ne pas oublier « l'étiquette »

On projette sur la colonne du regroupement pour identifier chaque groupe.

Afficher, pour chaque compagnie désignée par son id, la somme des heures de vol.

On peut utiliser toutes les fonctions de groupe!

```
SELECT compPil, SUM(nbHVol)
FROM Pilote
GROUP BY compPil
;
```

Afficher, pour chaque compagnie désignée par son nom, la somme des heures de vol.

On privilégie les étiquettes « compréhensibles »

```
SELECT nomComp, SUM(nbHVol)
FROM Pilote JOIN Compagnie ON compPil = idComp
GROUP BY nomComp
;
```

On est limité au niveau de la projection

On peut projeter uniquement sur la colonne du regroupement (étiquette) et sur les colonnes « calculées ».

En complément des questions précédentes, intéressons-nous à la gestion des doublons.

Par défaut, on affiche les doublons

```
SELECT pays
FROM Compagnie
;
```

RESULT

pays
France
France
Angleterre
Suisse
Irelande

On peut supprimer les doublons avec DISTINCT

```
SELECT DISTINCT pays
FROM Compagnie
;
```

RESULT

pays
France
Angleterre
Suisse
Irelande

Comment la fonction COUNT se comporte-t-elle avec les doublons ?

On peut compter les valeurs (non NULL)

```
SELECT COUNT(pays)
FROM Compagnie
;
```

RESULT

COUNT(pays)
5

On peut compter les valeurs (non NULL) distinctes

```
SELECT COUNT(DISTINCT pays)
FROM Compagnie
;
```

RESULT

COUNT(DISTINCT pays)
4

Pour terminer cette section, intéressons-nous à la question non traitée sur Excel.

Afficher la(les) compagnie(s) ayant le maximum de pilotes ?

Affichons d'abord le maximum de pilotes.

```
SELECT MAX(nb_pilotes)
FROM
    (
    SELECT compPil, COUNT(*) nb_pilotes
    FROM Pilote
    GROUP BY compPil
    )
;
```

RESULT

MAX(nb_pilotes)
2

Sous-requête

On peut utiliser le résultat d'une requête à l'intérieur d'une requête « principale ».

Affichons ensuite la(les) compagnie(s) ayant exactement deux pilotes.

```
SELECT compPil
FROM Pilote
GROUP BY compPil
HAVING COUNT(*) = 2
;
```

RESULT

compPil
1
4

Complétons l'ordre des directives

SELECT, FROM, WHERE, GROUP BY, HAVING , ORDER BY

Affichons enfin la(les) compagnie(s) ayant le maximum de pilotes.

```
SELECT compPil
FROM Pilote
GROUP BY compPil
HAVING COUNT(*) =
    (
    SELECT MAX(nb_pilotes)
    FROM
        (
        SELECT compPil, COUNT(*) nb_pilotes
        FROM Pilote
        GROUP BY compPil
        )
    )
;
```

Modifier la requête précédente de manière à afficher les noms des compagnies.

Consultation 2

1. Afficher le maximum de places dans un avion (de la base).
2. Afficher le type d'avion qui possède le maximum de places.
3. Afficher les noms des pilotes ayant un nombre d'heures de vol supérieur à la moyenne.
4. Afficher le nombre de places totales pour les avions d'Air France.
5. Pour chaque pilote, désigné par son nom, afficher le nombre de qualifications.
6. Afficher les noms des pilotes ayant 2 qualifications.
7. Afficher les noms des pilotes ayant le maximum de qualifications.
8. Pour chaque compagnie, désignée par son nom, afficher le nombre de places totales.

VI. Interaction entre un SGBD et un programme

Voici comment on peut utiliser un SGBD depuis le langage de programmation Python.

Ordre SQL

```
import sqlite3 as sgbd

# on établit une connection avec le SGBD
cnx = sgbd.connect("/Users/anthonyridard/Desktop/bd_aviation.db")

# on crée un curseur qui représente un ordre SQL
c = cnx.cursor()

# on peut alors exécuter cet ordre
c.execute( "SELECT * FROM Compagnie")

# on peut aussi afficher les lignes obtenues
for l in c.fetchall() :
    print(l)
```

RESULT

```
(1, 'Air France', 'France', 0)
(2, 'Corsair International', 'France', 0)
(3, 'EasyJet', 'Angleterre', 1)
(4, 'Swiss', 'Suisse', 0)
(5, 'Ryanair', 'Irelande', 1)
```

On peut insérer, dans un ordre SQL, une valeur saisie par l'utilisateur du programme.

Ordre SQL paramétré

```
texte = input('Choisissez un pays :')

c.execute( "SELECT * FROM Compagnie WHERE pays = ?", [texte])
```

RESULT

```
Choisissez un pays :France
(1, 'Air France', 'France', 0)
(2, 'Corsair International', 'France', 0)
```

L'utilisation du paramètre "?" est indispensable pour éviter les **injections SQL**.

Ordre SQL « paramétré » vulnérable

```
texte = input('Choisissez un pays :')

c.executescript( "SELECT * FROM Compagnie WHERE pays = '" + texte + "'")
```

RESULT

```
Choisissez un pays :'; DROP TABLE Pilote; SELECT * FROM Compagnie WHERE pays = 'Suisse
air-de-anthony:~ anthonyridard$
```