

Les bases de données en langage SQL

Jean-Claude AZIAHA



Les objectifs de la formation



Créer une base de données



Créer, modifier et supprimer des tables



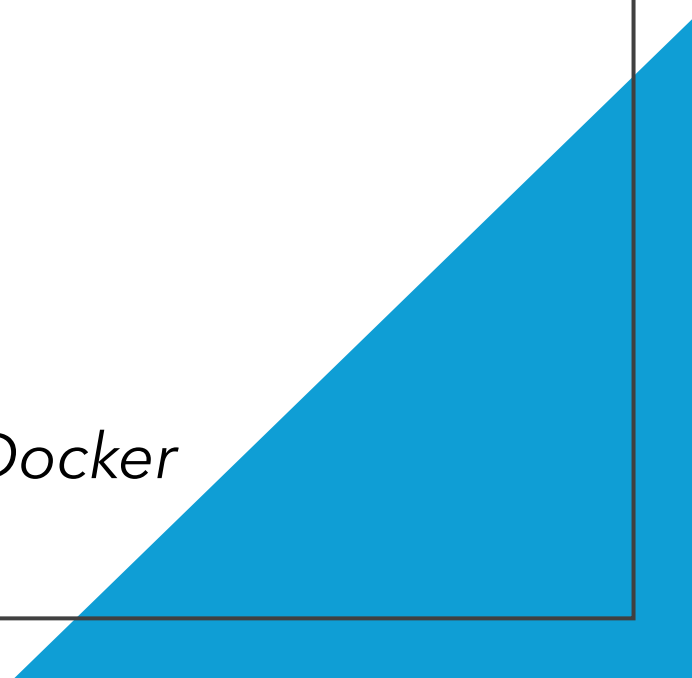
Insérer, sélectionner, modifier,
supprimer des données dans les tables



Garantir la cohérence des données
avec les clés et les relations

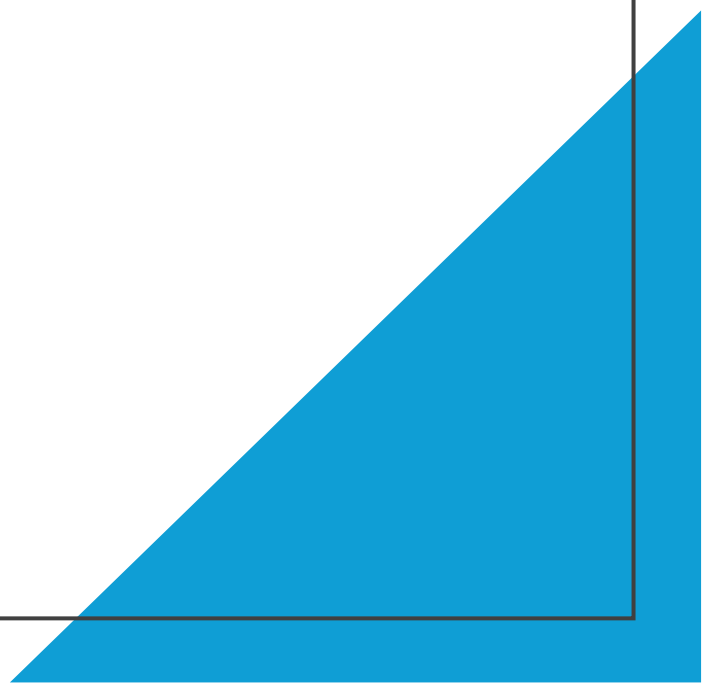
Programme

Partie A: Découverte

1. Rappel sur le principe du Web
 2. Qu'est-ce qu'une base de données?
 3. Le langage SQL et son importance
 4. Le SGBDR et son importance
 5. La liste des SGBDR
 6. Présentation de *MySQL*
 7. Installation de *MySQL* avec *Docker*
 8. Installation de l'interface *phpmyadmin* avec *Docker*
- 

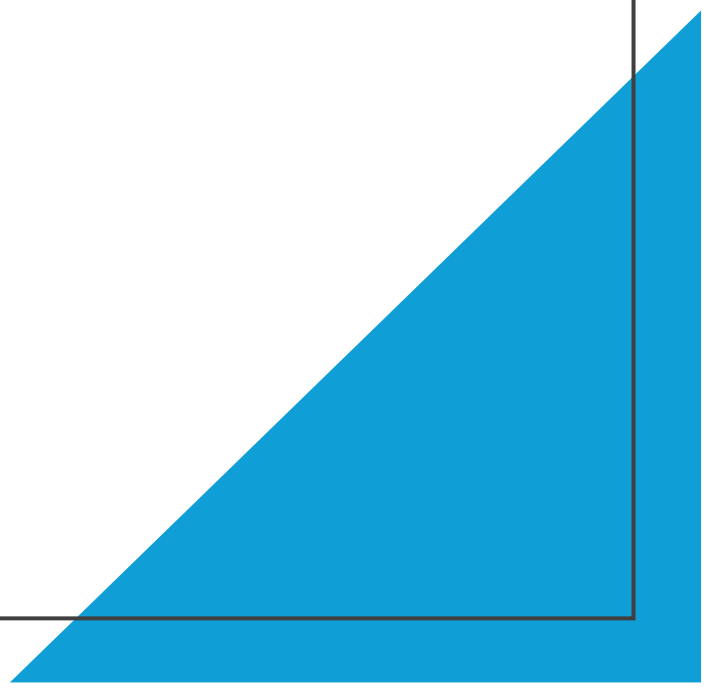
Programme

Partie B: Langage de définition des données (LDD)

1. Création et suppression d'une base de données
 2. La table et ses colonnes
 3. Les principaux types de données
 4. Création d'une table
 5. La clé primaire
 6. La clé candidate
 7. Modification d'une table
 8. Suppression d'une table
- 
- A large blue right-angled triangle is positioned in the bottom right corner of the slide, with its hypotenuse running from the bottom left towards the top right.

Programme

Partie C: Langage de manipulation des données (**LMD**)

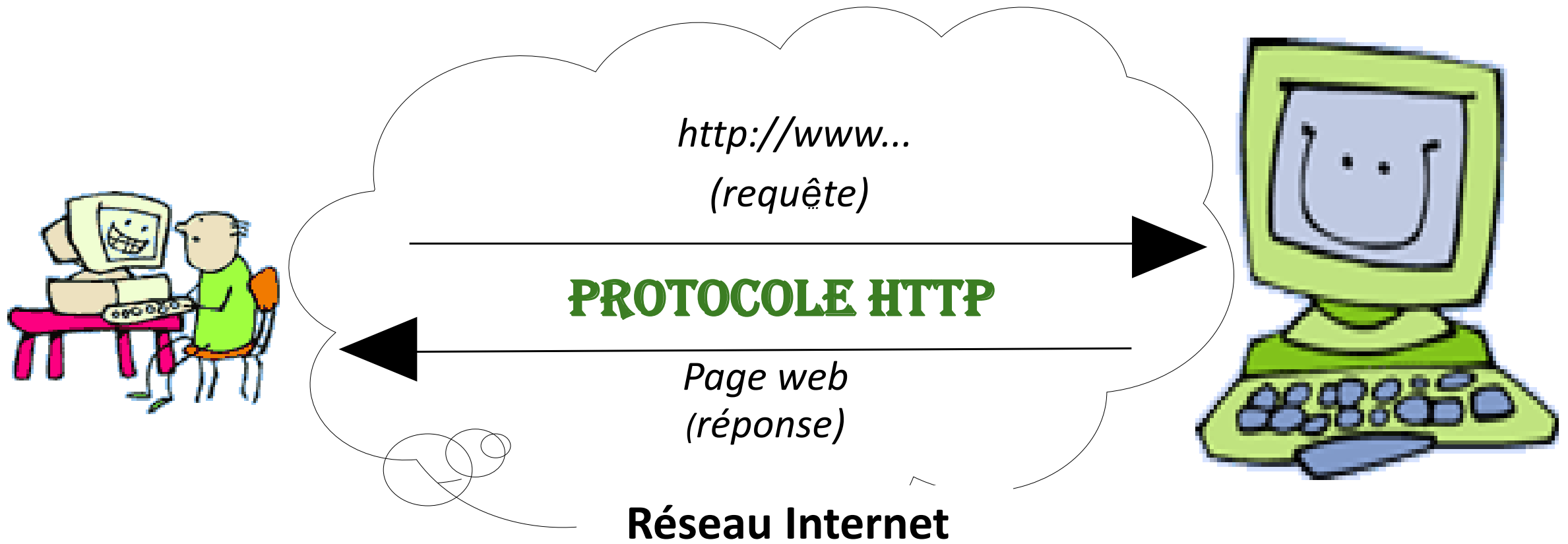
1. Insertion des données dans une table
 2. Sélection des données d'une table
 3. Order, Limit
 4. Les fonctions d'agrégation
 5. Mise à jour des données d'une table
 6. Suppression des données d'une table
 7. Les clés étrangères et jointures
- 

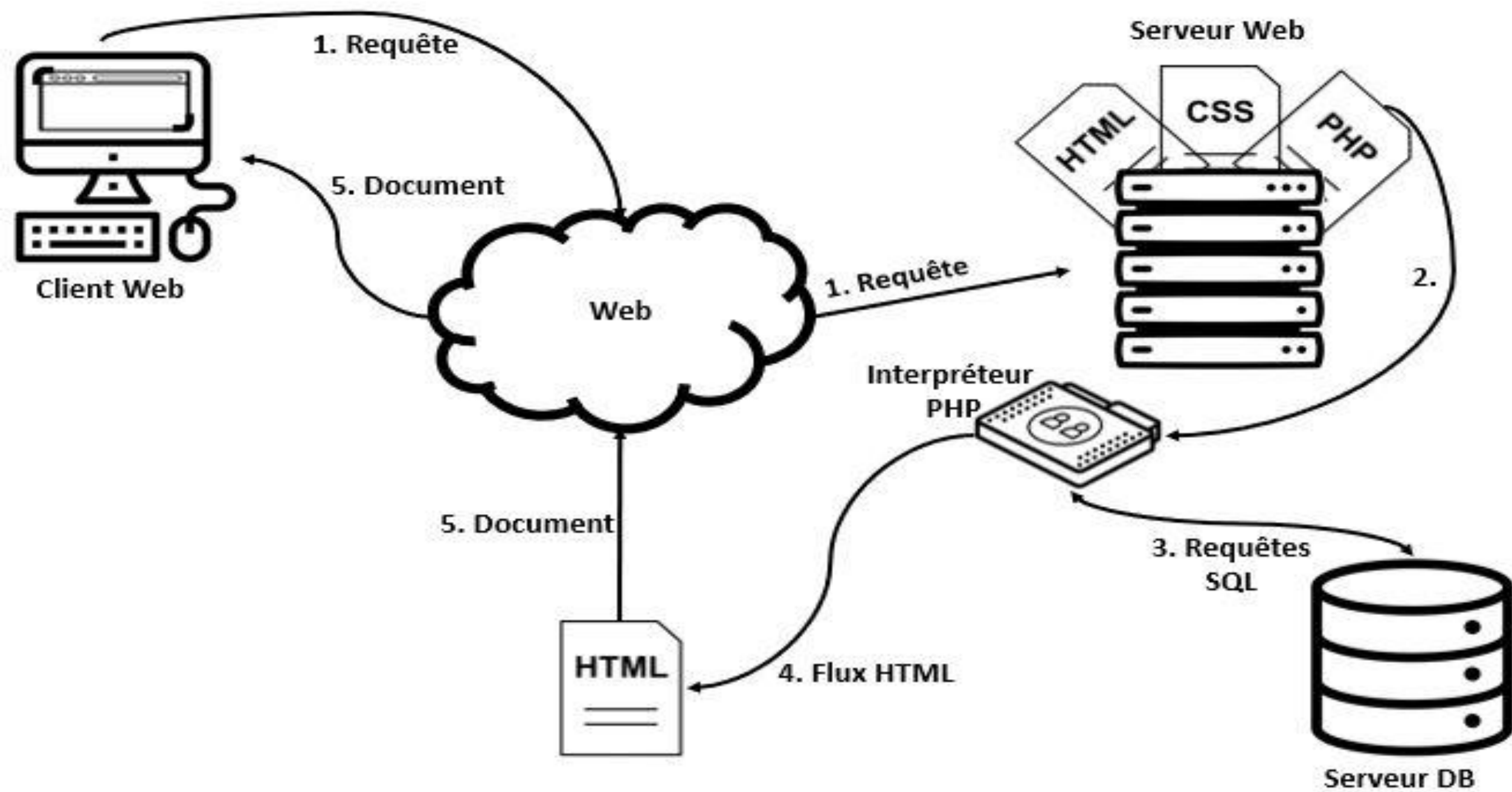
Partie A

Découverte



A1- Rappel sur le principe du Web





A2- Qu'est-ce qu'une base de données?

- Une base de données est une **collection structurée d'informations**.
- Elle permet donc de collecter des informations de manière organisée afin qu'elles puissent être facilement récupérées, manipulées ou mises à jour.



A3- Le langage SQL et son importance

- Pour créer et manipuler une base de données, nous avons besoin d'un langage permettant de le faire. Ce langage est le SQL.
- **SQL** est en réalité un acronyme qui signifie : **S**tructured **Q**uery **L**anguage.
- C'est un langage structuré, conçu spécifiquement pour créer et manipuler des bases de données relationnelles.
- **Importance:**
 - Permet de créer et manipuler des bases de données
 - C'est le standard dans l'industrie
 - Très utilisé dans de nombreuses entreprises

A4- Le SGBDR et son importance

- Nous venons de dire que la création d'une base de données nécessite un langage adapté; En l'occurrence le SQL.
- Mais ce langage doit s'exécuter dans un environnement pour qu'il soit fonctionnel.
- Cet environnement s'appelle un **S**ystème de **G**estion de **B**ase de **D**onnées **R**elationnelles (SGBDR).
- Un SGBDR est tout simplement un serveur ou un service qui permet gérer des bases de données.

A4- Le SGBDR et son importance

➤ **Importance:**

- Gérer l'accès aux bases de données
- Permettre aux utilisateurs de stocker, de récupérer et de manipuler des données via le langage SQL

❖ En somme, pour manipuler une base de données SQL, nous devons passer par un Système de Gestion de Base de Données Relationnel.

A5- La liste des SGBDR

Les SGBDR Open Source

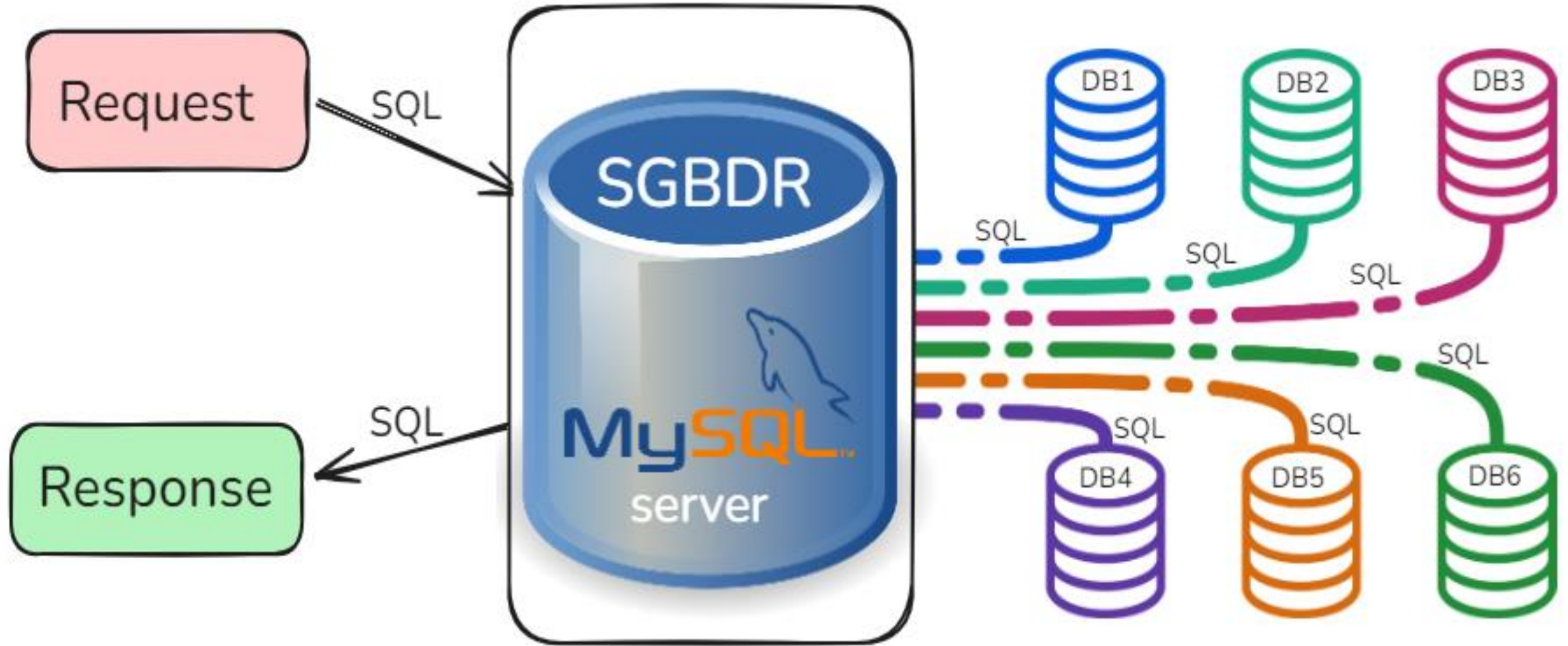
- MySQL
- MariaDB
- PostgreSQL
- SQLite

Les SGBDR Propriétaires

- Oracle Database
- Microsoft SQL Server
- IBM Db2

Les autres SGBDR Notables

- SAP HANA
- Amazon Aurora
- Teradata



A6- Présentation de MySQL

- MySQL est le SGBDR le plus utilisé dans le monde
- Particulièrement dans le développement Web
- ❖ C'est donc le SGBDR que nous utiliserons dans cette formation.



A6- Présentation de MySQL



A7- Installation de MySQL avec Docker

MySQL peut être installé de plusieurs façons différentes:

- Installation Native en fonction du système d'exploitation (Windows, Mac, Linux)
- Installation via des logiciels comme Wamp/Xampp/Mamp/Lamp
- Installation via la containerisation avec Docker
- Installation via des Services Cloud (Amazon, Google Cloud, Microsoft Azure)
- ...

❖ Dans cette formation, nous installerons MySQL via **Docker**.

A7- Installation de MySQL avec Docker

❑ COMMENÇONS PAR DOCKER

- Docker est une plateforme qui permet de créer et de gérer des conteneurs.
- Un conteneur est un environnement léger et auto-suffisant contenant tous les services dont une application a besoin pour fonctionner.

➤ **Téléchargement de Docker**

- Se rendre sur le site officiel: <https://www.docker.com/products/docker-desktop/>
- Télécharger l'exécutable de Docker Desktop en fonction du système d'exploitation

➤ **Installation de Docker**

- Exécuter le fichier téléchargé
- Suivre les instructions de l'assistant d'installation
- Dans le terminal de votre machine: *docker --version*

A7- Installation de MySQL avec Docker

❑ PUIS DOCKER-COMPOSE

- Docker-Compose est un outil permettant de définir et configurer **simplement** tous les services à mettre en place avec **Docker**.

➤ Téléchargement de Docker-Compose

- Se rendre sur le site officiel: <https://docs.docker.com/compose/install/>
- Télécharger l'exécutable de Docker-Compose en fonction du système d'exploitation

➤ Installation de Docker-Compose

- Exécuter le fichier téléchargé
- Suivre les instructions de l'assistant d'installation
- Dans le terminal de votre machine: `docker-compose --version`

A7- Installation de MySQL avec Docker

❑ ENFIN MYSQL

- Créer un nouveau dossier en local sur votre machine: « **mysql-with-docker** »
- Y créer le fichier: « **docker-compose.yml** »
- Y rajouter de fichier de configuration de MySQL « **docker/mysql/mysql.cnf** »

❖ *Lien GitHub du repository:*

https://github.com/jc-aziaha/imie_sql_with_docker.git

EXPLORER



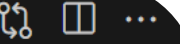
✓ MYSQL-WITH-D...

✓ docker\mysql

mysql.cnf

docker-compose.yml M

docker-compose.yml M X



docker-compose.yml > {} services

docker-compose.yml - The Compose specification establishes a standard for the definition of multi-container platform-agnost

```
1  networks:
2    mysql-network:
3      driver: bridge
4
5  services:
6    mysql:
7      image: mysql:8.0
8      container_name: mysql-server
9      restart: unless-stopped
10     tty: true
11     environment:
12       MYSQL_ROOT_PASSWORD: azerty1234A*
13       TZ: Europe/Paris
14     ports:
15       - 3306:3306
16     volumes:
17       - ../docker/mysql/mysql.cnf:/etc/mysql/conf.d/custom.cnf:ro
18       - ../docker/mysql/data:/var/lib/mysql
19     networks:
20       - mysql-network
```

EXPLORER



✓ MYSQL-WITH-DOCKER

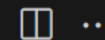
✓ docker\mysql

mysql.cnf

docker-compose.yml M

docker-compose.yml M

mysql.cnf X



docker > mysql > mysql.cnf

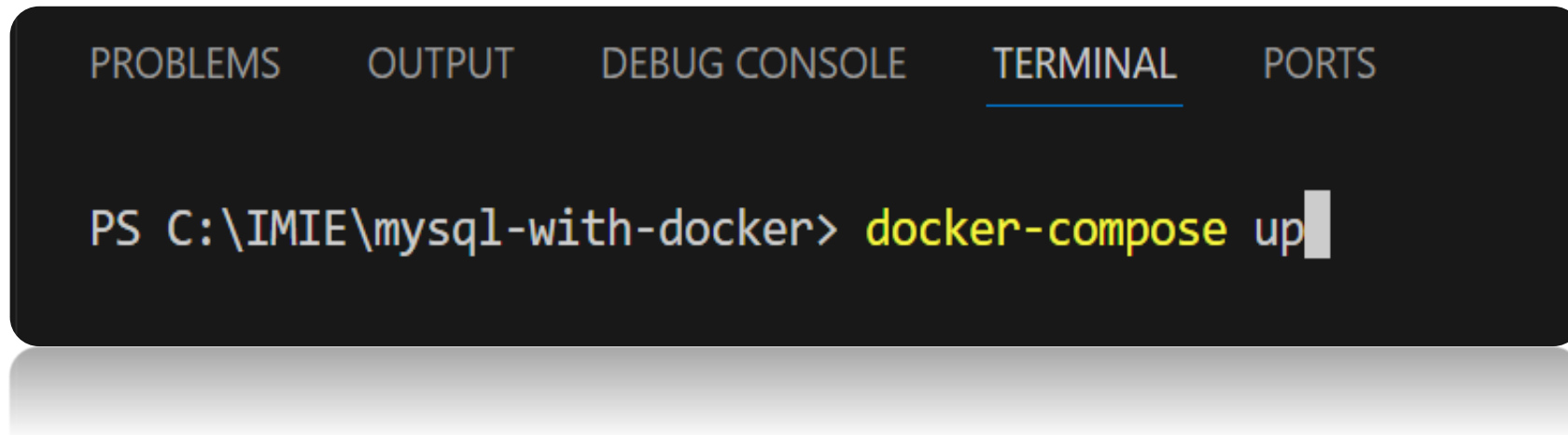
```
1 [mysqld]
2 bind-address = 0.0.0.0
3 innodb_buffer_pool_size = 1024M
4 default-authentication-plugin = mysql_native_password
5
6 # Unicode
7 character-set-server = utf8mb4
8 collation-server = utf8mb4_unicode_ci
9
10 [client]
11 ssl-mode = DISABLED
12 default-character-set = utf8mb4
13
14 [mysql]
15 default-character-set = utf8mb4
16
```


MySQL 8.0.27
MySQL 8.0.27
MySQL 8.0.27

```
10 [client]
11 ssl-mode = DISABLED
12 default-character-set = utf8mb4
13
14 [mysql]
15 default-character-set = utf8mb4
```

A7- Installation de MySQL avec Docker

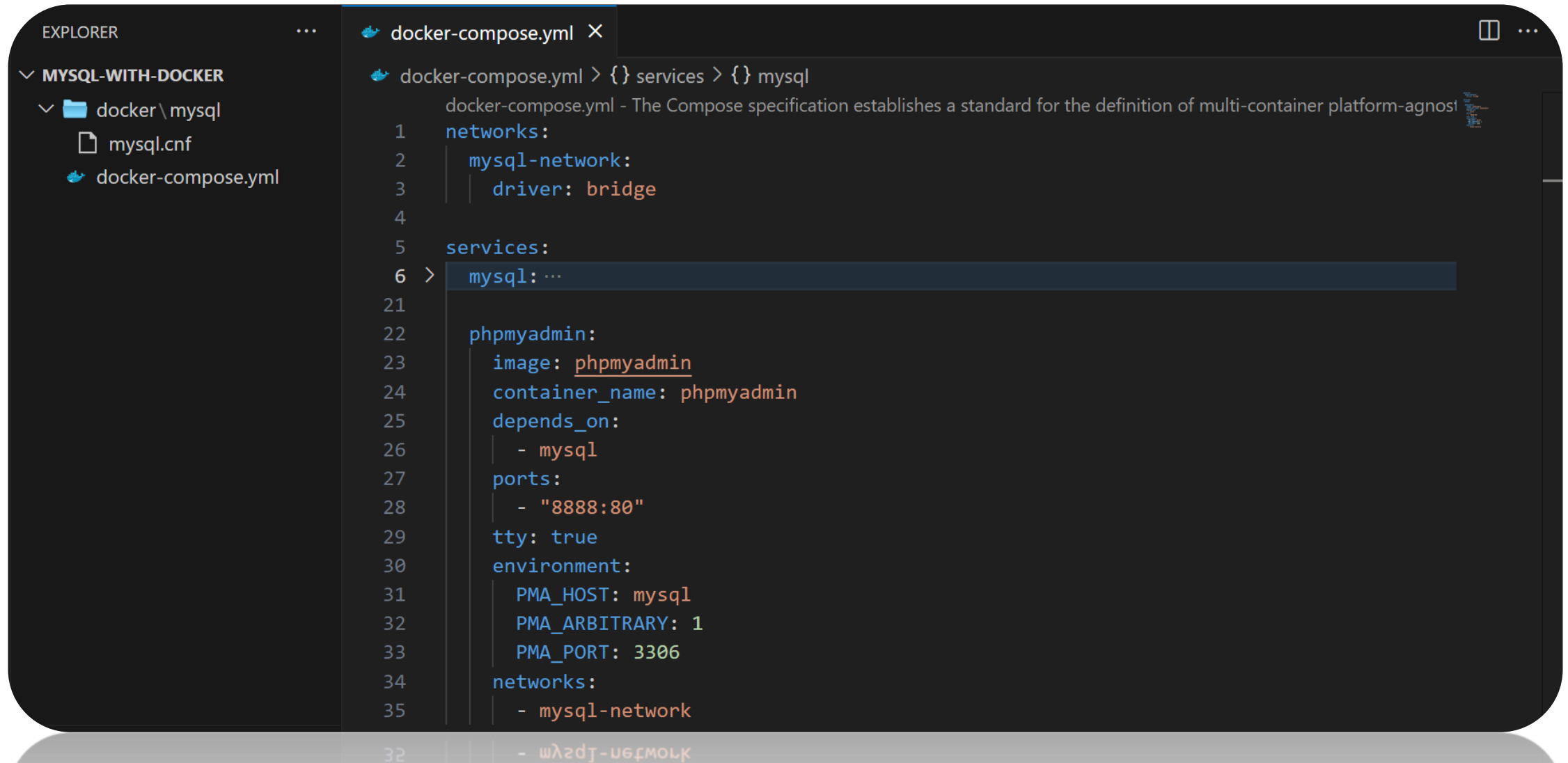
Cette commande permet de démarrer les services définis dans le fichier
« docker-compose.yml »

A screenshot of a Visual Studio Code terminal window. The terminal has a dark background with a light gray border. At the top, there are five tabs: 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', 'TERMINAL' (which is selected and underlined with a blue line), and 'PORTS'. Below the tabs, the terminal shows a command prompt 'PS C:\IMIE\mysql-with-docker>' followed by the command 'docker-compose up' in yellow text. A white cursor is positioned at the end of the command.

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\IMIE\mysql-with-docker> docker-compose up
```

A8- Installation de phpmyadmin



The screenshot shows a code editor with a dark theme. On the left, the 'EXPLORER' sidebar displays a file tree for a project named 'MYSQL-WITH-DOCKER'. It contains a folder 'docker\mysql' with files 'mysql.cnf' and 'docker-compose.yml'. The main editor area shows the 'docker-compose.yml' file, which is open in a tab labeled 'docker-compose.yml X'. The breadcrumb navigation at the top of the editor reads 'docker-compose.yml > {} services > {} mysql'. The code defines a 'mysql' service and a 'phpmyadmin' service. The 'mysql' service is partially visible, showing 'networks:' and 'driver: bridge'. The 'phpmyadmin' service is fully visible, including 'image: phpmyadmin', 'container_name: phpmyadmin', 'depends_on: - mysql', 'ports: - "8888:80"', 'tty: true', 'environment: PMA_HOST: mysql, PMA_ARBITRARY: 1, PMA_PORT: 3306', and 'networks: - mysql-network'.

```
docker-compose.yml > {} services > {} mysql
docker-compose.yml - The Compose specification establishes a standard for the definition of multi-container platform-agnost

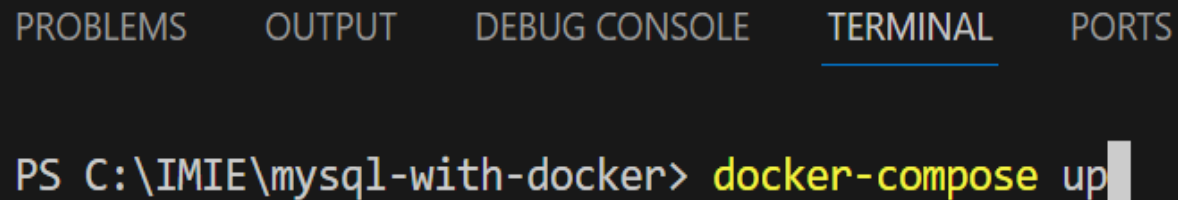
1 networks:
2   mysql-network:
3     driver: bridge
4
5 services:
6 > mysql: ...
21
22 phpmyadmin:
23   image: phpmyadmin
24   container_name: phpmyadmin
25   depends_on:
26     - mysql
27   ports:
28     - "8888:80"
29   tty: true
30   environment:
31     PMA_HOST: mysql
32     PMA_ARBITRARY: 1
33     PMA_PORT: 3306
34   networks:
35     - mysql-network
36
```


A8- Installation de phpmyadmin

- Nous venons de rajouter un nouveau service nommé **phpmyadmin**.
- Or les services étaient déjà démarrés
- Afin que nos modifications soient prises en compte, coupons et relançons le serveur

➤ Pour couper le serveur: **CTRL C**

➤ Pour le relancer:



A screenshot of a terminal window with a dark background. At the top, there are five tabs: 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', 'TERMINAL' (which is selected and underlined), and 'PORTS'. Below the tabs, the terminal shows a command prompt 'PS C:\IMIE\mysql-with-docker>' followed by the command 'docker-compose up' in yellow text. A white cursor is at the end of the command.

➤ Dans le navigateur, en chargeant l'url <http://localhost:8888>, l'interface d'accueil de phpmyadmin devrait s'afficher.

PARTIE B

Langage de définition des données



B1- Création et suppression d'une base de données?

```
1 -- Création de notre première base de données
2 CREATE DATABASE my_first_db;
```

Ou



```
1 -- Création de notre première base de données
2 CREATE DATABASE IF NOT EXISTS my_first_db
3 CHARACTER SET utf8mb4
4 COLLATE utf8mb4_unicode_ci;
```

B1- Création et suppression d'une base de données?

```
6 -- Consulter la liste des bases de données
7 SHOW DATABASES;
```

Database

information_schema

my_first_db

mysql

performance_schema

sys

B1- Création et suppression d'une base de données?

```
9  -- Supprimer une base de données
10 DROP DATABASE my_first_db;
```

Ou



```
9  -- Supprimer une base de données
10 DROP DATABASE IF EXISTS my_first_db;
```

B2- Les tables et ses colonnes

- Dans une base de données, les données sont regroupées par table.
- Chaque table est subdivisée en plusieurs colonnes.
- À chaque colonne est associé le type de données.

Exemple de structure d'une table nommée « **contacts** »

	#	Nom	Type	Interclassement	Attributs	Null	Valeur par défaut	Commentaires	Extra
<input type="checkbox"/>	1	first_name	varchar(255)	utf8mb4_unicode_ci		Non	<i>Aucun(e)</i>		
<input type="checkbox"/>	2	last_name	varchar(255)	utf8mb4_unicode_ci		Non	<i>Aucun(e)</i>		
<input type="checkbox"/>	3	age	int			Oui	<i>NULL</i>		

B2- Les tables et ses colonnes

- ❖ Cela veut dire que pour créer une table, nous devons connaître le type qui sera attribué à chacune de ses colonnes.
- ❖ Prenons donc le temps de connaître les principaux types qui existent.
 - a) Types de Données Numériques Entiers: tinyint, smallint, mediumint, int, bigint
 - b) Types de Données Numériques à Virgule Flottante: float, double, decimal
 - c) Types de Données Chaînes de Caractères: varchar, tinytext, text, mediumtext, longtext
 - d) Types de Données Date et Heure: date, time, datetime, timestamp, year
 - e) Types de Données Booléens: boolean
 - f) Autres types: json

B3- Les principaux types de données

a. Types de Données Numériques Entiers

- **TINYINT** : Petit entier, plage de -128 à 127 ou 0 à 255 (non signé).
- **SMALLINT** : Petit entier, plage de -32,768 à 32,767 ou 0 à 65,535 (non signé).
- **MEDIUMINT** : Entier moyen, plage de -8,388,608 à 8,388,607 ou 0 à 16,777,215 (non signé).
- **INT** (ou INTEGER) : Entier, plage de -2,147,483,648 à 2,147,483,647 ou 0 à 4,294,967,295 (non signé).
- **BIGINT** : Grand entier, plage de -9,223,372,036,854,775,808 à 9,223,372,036,854,775,807 ou 0 à 18,446,744,073,709,551,615 (non signé).

B3- Les principaux types de données

b. Types de Données Numériques à Virgule Flottante

- **FLOAT**: Nombre à virgule flottante simple précision.
- **DOUBLE**: Nombre à virgule flottante double précision.
- **DECIMAL**: Nombre à virgule fixe

B3- Les principaux types de données

c. Types de Données Chaînes de Caractères

- **VARCHAR**: Chaîne de caractères de longueur variable, jusqu'à 65,535 caractères. La longueur dépend de l'encodage et de la taille de la ligne.
- **TINYTEXT** : Texte de petite taille, jusqu'à 255 caractères.
- **TEXT** : Texte de taille moyenne, jusqu'à 65,535 caractères.
- **MEDIUMTEXT** : Texte de grande taille, jusqu'à 16,777,215 caractères.
- **LONGTEXT** : Texte de très grande taille, jusqu'à 4,294,967,295 caractères.

B3- Les principaux types de données

d. Types de Données Date et Heure

- **DATE** : Date au format YYYY-MM-DD.
- **TIME** : Heure au format HH:MM:SS.
- **DATETIME** : Combinaison de date et d'heure, au format YYYY-MM-DD HH:MM:SS.
- **TIMESTAMP** : Horodatage, stockage du nombre de secondes depuis l'époque Unix (1970-01-01 00:00:00 UTC).
- **YEAR** : Année au format YYYY, plage de 1901 à 2155.

B3- Les principaux types de données

e. Types de Données Booléens et Autres

- **BOOLEAN** : Alias pour **TINYINT**(1). Stocke 0 (faux) ou 1 (vrai).

f. Autres Types

- **JSON** : Stocke des données au format JSON.

B4- Création d'une table

Utilisation d'une base de données

```
12  -- Utiliser une base de données  
13  USE my_first_db;
```

B4- Création d'une table

```
15 -- Création de notre première table
16 CREATE TABLE IF NOT EXISTS contacts (
17     first_name VARCHAR(255) NOT NULL,
18     last_name VARCHAR(255) NOT NULL,
19     age INT NULL
20 )
21 ENGINE=InnoDB,
22 CHARACTER SET utf8mb4,
23 COLLATE utf8mb4_unicode_ci;
```

B5- La clé primaire

- Lorsque cette table sera remplie, plusieurs contacts peuvent avoir le même prénom, nom et âge. Ce qui est possible dans la « vraie vie ».

first_name	last_name	age
Jean	Dupond	22
Sophie	Durand	35
Jean	Dupond	48
Jean	Dupond	48

- Il faut donc trouver un moyen de les distinguer afin de ne jamais les confondre.
- Pour se faire, nous allons attribuer à chaque contact, une nouvelle colonne nommée « id ». Cette colonne sera l'identifiant permettant donc d'identifier chaque contact de la liste.

B5- La clé primaire

- ❖ Au final, la clé primaire est une contrainte d'intégrité qui permet d'identifier de manière unique chaque enregistrement d'une table.
- Rajoutons donc à la table « contacts », une nouvelle colonne « id » qui sera:
 - unique
 - non nulle
 - auto-incrémentée (commencera à partir de 1 pour le premier enregistrement de la table)

B5- La clé primaire

```
25 -- Ajouter une nouvelle colonne "id"
26 -- qui représente la clé primaire de la table
27 ALTER TABLE contacts
28 ADD COLUMN id INT AUTO_INCREMENT NOT NULL PRIMARY KEY FIRST;
```

```
30 -- Description de la table
31 DESCRIBE contacts;
```

Field	Type	Null	Key	Default	Extra
id	int	NO	PRI	NULL	auto_increment
first_name	varchar(255)	NO		NULL	
last_name	varchar(255)	NO		NULL	
age	int	YES		NULL	

B6- La clé candidate

- Une clé candidate est une clé qui possède les mêmes attributs qu'une clé primaire.
- Ses caractéristiques:
 - unique
 - non nulle
- On l'appelle clé **candidate** parce qu'elle peut, si besoin, jouer le rôle de clé primaire.

B6- La clé candidate

```
33 CREATE TABLE IF NOT EXISTS users (  
34     id INT AUTO_INCREMENT NOT NULL PRIMARY KEY,  
35     first_name VARCHAR(255) NOT NULL,  
36     last_name VARCHAR(255) NOT NULL,  
37     email VARCHAR(180) NOT NULL UNIQUE, ←  
38     password VARCHAR(255) NOT NULL,  
39     created_at DATETIME DEFAULT NULL COMMENT '(DC2Type:datetime_immutable)',  
40     verified_at DATETIME DEFAULT NULL COMMENT '(DC2Type:datetime_immutable)',  
41     updated_at DATETIME DEFAULT NULL COMMENT '(DC2Type:datetime_immutable)'  
42 )  
43 ENGINE=InnoDB,  
44 CHARACTER SET utf8mb4,  
45 COLLATE utf8mb4_unicode_ci;
```

B7- Modification d'une table

- Modifier une table signifie changer sa structure.
- Soit en:
 1. Renommant la table:
 - `RENAME TABLE old_table_name TO new_table_name;`
 - **Exemple:** `RENAME TABLE categories TO posts_categories;`
 2. Rajoutant une nouvelle colonne:
 - `ALTER TABLE table_name ADD column_name column_definition;`
 - **Exemple:** `ALTER TABLE users ADD age VARCHAR(255) NOT NULL;`
 3. Modifiant une colonne:
 - `ALTER TABLE table_name MODIFY colum_name new column_definition;`
 - **Exemple:** `ALTER TABLE users MODIFY first_name VARCHAR(200);`

B7- Modification d'une table

5. Supprimant une colonne

- ALTER TABLE *table_name* DROP COLUMN *column_name*;
- **Exemple:** ALTER TABLE *users* DROP COLUMN *column_name*;

6. Modifiant l'encodage des caractères la table:

- ALTER TABLE *table_name* CONVERT TO CHARACTER SET *charset_name*;
- **Exemple:** ALTER TABLE *categories* CONVERT TO CHARACTER SET *utf8mb4*;

7. Modifiant le moteur de stockage:

- ALTER TABLE *table_name* ENGINE = *engine_name*;
- **Exemple:** ALTER TABLE *users* ENGINE = *InnoDB*;

B8- Suppression d'une table

- La suppression d'une table est une opération irréversible qui supprime la table ainsi que toutes les données qu'elle contient.

- Cette opération se réalise soit en:
 1. Supprimant une table à la fois:
 - DROP TABLE IF EXISTS *table_name*;
 - **Exemple:** DROP TABLE IF EXISTS *users*;
 2. Supprimer plusieurs tables à la fois:
 - DROP TABLE IF EXISTS *table1*, IF EXISTS *table2*, IF EXISTS *table3*;
 - **Exemple:** DROP TABLE IF EXISTS *users*, IF EXISTS *orders*, IF EXISTS *products*;

PARTIE C

Langage de manipulation
des données

A solid orange horizontal bar is positioned below the text, spanning most of the width of the white background area.

C1- Insertion des données dans une table

Insertion d'une ligne à la fois:

```
INSERT INTO table_name (column_name1, column_name2, ...)  
VALUES ('valeur 1', 'valeur 2', ...);
```

Exemple:

```
50 INSERT INTO contacts (first_name, last_name, age)  
51     VALUES ('picsou', 'Balthazar', 70);
```


C1- Insertion des données dans une table

Insertion d'une ligne à la fois:

```
INSERT INTO table (column_name1, column_name2, ...)  
VALUES ('valeur1', 'valeur 2', ...) ('valeur 3', 'valeur 4', ...);
```

```
53 INSERT INTO contacts (first_name, last_name, age)  
54     VALUES ('Donald', 'Duck', 35),  
55             ('Daisy', 'Duck', 33),  
56             ('Riri', 'Duck', 12),  
57             ('Firi', 'Duck', 12),  
58             ('Loulou', 'Duck', 12);
```

C2- Sélection des données d'une table

Formule: `SELECT * FROM table_name;`

Exemple:

```
60 -- Selectionnons toutes les colonnes
61 -- de tous les enregistrements de la table
62 SELECT * FROM contacts;
```

id	first_name	last_name	age
1	picsou	Balthazar	70
2	Donald	Duck	35
3	Daisy	Duck	33
4	Riri	Duck	12
5	Firi	Duck	12
6	Loulou	Duck	12

C2- Sélection des données d'une table

Formule: `SELECT column1, column2 FROM table_name;`

Exemple:

```
64 -- Selectionnons pour chaque contact de la table,  
65 -- seulement son prénom et nom.  
66 SELECT first_name, last_name FROM contacts;
```

first_name	last_name
picsou	Balthazar
Donald	Duck
Daisy	Duck
Riri	Duck
Firi	Duck
Loulou	Duck

C2- Sélection des données d'une table

Formule: `SELECT * FROM table_name WHERE condition;`

Exemple:

```
68 -- Selectionnons seulement les contacts
69 -- qui portent le nom Balthazar
70 SELECT * FROM contacts WHERE last_name="Balthazar";
```

id	first_name	last_name	age
1	picsou	Balthazar	70

C2- Sélection des données d'une table

Formule: `SELECT * FROM table_name WHERE condition AND condition;`

Exemple:

```
72 -- Selectionnons seulement les contacts
73 -- qui portent le nom Duck et qui ont 12 ans
74 SELECT * FROM contacts WHERE last_name="Duck" AND age=12;
```

id	first_name	last_name	age
4	Riri	Duck	12
5	Firi	Duck	12
6	Loulou	Duck	12

C3- ORDER et LIMIT

➤ ORDER BY :

- Sert à trier les résultats d'une requête selon une ou plusieurs colonnes, soit en ordre croissant (ASC), soit en ordre décroissant (DESC).
- **Exemple** : SELECT * FROM employés **ORDER BY** salaire **DESC**;

➤ LIMIT :

- Permet de limiter le nombre de lignes retournées par une requête.
- **Exemple** : SELECT * FROM employés **LIMIT 5**;

C4- Les fonctions d'agrégation

➤ **COUNT()** :

- Compte le nombre de lignes correspondant à une condition ou dans un ensemble de données.
- **Exemple** : `SELECT COUNT(*) FROM employés;`

➤ **SUM()** :

- Calcule la somme d'une colonne numérique.
- **Exemple** : `SELECT SUM(salaire) FROM employés;`

➤ **AVG()** :

- Calcule la moyenne des valeurs d'une colonne numérique.
- **Exemple** : `SELECT AVG(salaire) FROM employés;`

C4- Les fonctions d'agrégation

➤ **MIN()** :

- Retourne la valeur minimale d'une colonne.
- Exemple : `SELECT MIN(salaire) FROM employés;`

➤ **MAX()** :

- Retourne la valeur maximale d'une colonne.
- Exemple : `SELECT MAX(salaire) FROM employés;`

C5- Mise à jour des données d'une table

Formule: UPDATE *table_name* SET *column_name* = 'value' WHERE *condition*;

Exemple:

```
76 -- Mettons à jour le prénom de Balthazar
77 -- en changeant "picsou" en "Picsou"
78 UPDATE contacts SET first_name = "Picsou" WHERE id=1;
```

```
80 -- Selectionnons les valeurs associée au contact
81 -- dont l'identifiant est 1. C'est à dire Picsou Balthazar.
82 SELECT * FROM contacts WHERE id=1;
```

id	first_name	last_name	age
1	Picsou	Balthazar	70

C6- Suppression des données d'une table

Formule: DELETE FROM *contacts* WHERE *condition*;

Exemple:

```
84 -- Supprimons Picsou de la table "contacts".  
85 DELETE FROM contacts WHERE id=1;
```

```
87 -- Selectionnons tous les enregistrements de la table "contacts".  
88 SELECT * FROM contacts;
```

id	first_name	last_name	age
2	Donald	Duck	35
3	Daisy	Duck	33
4	Riri	Duck	12
5	Firi	Duck	12
6	Loulou	Duck	12

C7- Les jointures et clés étrangères

- Les jointures permettent de combiner les données de plusieurs tables en une seule requête SQL.
- Cela est utile lorsque les informations recherchées sont réparties entre plusieurs tables liées par des clés étrangères.
- Il existe plusieurs types de jointures, chacune ayant un objectif différent.
 - INNER JOIN
 - LEFT JOIN
 - RIGHT JOIN
 - FULL OUTER JOIN

C7- Les jointures et clés étrangères

Type de Join	Que fait-il?	Résultat si pas de correspondance
INNER JOIN	Retourne uniquement les lignes qui ont une correspondance dans les deux tables	Ignore la ligne
LEFT JOIN	Retourne toutes les lignes de la table de gauche, même sans correspondance	Valeurs NULL pour la table droite
RIGHT JOIN	Retourne toutes les lignes de la table de droite, même sans correspondance	Valeurs NULL pour la table gauche
FULL JOIN	Retourne toutes les lignes des deux tables, correspondance ou non	Valeurs NULL pour la table sans correspondance

C7- Les jointures et clés étrangères

Des exemples pour mieux comprendre:

Id_client	nom	Id_commande	Id_client	Montant (€)
1	Sophie	101	1	50
2	Issa	102	1	30
3	Benoît	103	2	70
4	Fatou	104	2	20
		105	5	100

C7- Les jointures et clés étrangères

❑ **INNER JOIN** (Jointure interne) : Seules les lignes qui ont une correspondance dans les deux tables sont affichées.

▪ **Exemple:**

```
SELECT client.nom, commande.montant  
FROM client  
INNER JOIN commande  
ON client.id_client = commande.id_client;
```

nom	Montant (€)
Sophie	50
Sophie	30
Issa	70
Issa	20

- ❖ Seuls Alice et Bob apparaissent car ils ont des commandes.
- ❖ Charlie et David n'ont pas de commandes, ils ne sont pas affichés.

C7- Les jointures et clés étrangères

❑ **LEFT JOIN** (Jointure à gauche): Toutes les lignes de la table de gauche sont affichées, même si aucune correspondance n'existe dans la table de droite.

▪ **Exemple:**

```
SELECT client.nom, commande.montant  
FROM client  
LEFT JOIN commande  
ON client.id_client = commande.id_client;
```

nom	Montant (€)
Sophie	50
Sophie	30
Issa	70
Issa	20
Benoît	NULL
Fatou	NULL

- ❖ Charlie et David sont inclus dans le résultat, même s'ils n'ont pas de commandes.
- ❖ Les montants de leurs commandes sont NULL car aucune commande n'est associée à eux.

C7- Les jointures et clés étrangères

❑ **RIGHT JOIN** (Jointure à droite): Toutes les lignes de la table de droite sont affichées, même si aucune correspondance n'existe dans la table de gauche.

▪ **Exemple:**

```
SELECT client.nom, commande.montant  
FROM client  
RIGHT JOIN commande  
ON client.id_client = commande.id_client;
```

nom	Montant (€)
Sophie	50
Sophie	30
Issa	70
Issa	20
NULL	100

- ❖ Une commande (id_commande = 105) est associée à un id_client inexistant (id_client = 5).
- ❖ Elle est donc affichée avec des informations sur le client à NULL.

C7- Les jointures et clés étrangères

❑ **FULL OUTER JOIN** (Jointure complète externe): Toutes les lignes des deux tables sont affichées, que la correspondance existe ou non.

▪ **Exemple:**

SELECT Client.nom, Commande.montant

FROM Client

LEFT JOIN Commande **ON** Client.id_client = Commande.id_client

UNION

SELECT Client.nom, Commande.montant

FROM Client

RIGHT JOIN Commande **ON** Client.id_client = Commande.id_client;

C7- Les jointures et clés étrangères

Ce résultat combine toutes les lignes des deux tables.

Charlie et David apparaissent sans commande (NULL), et la commande associée à un client inexistant (id_client = 5) apparaît également avec un nom NULL.JOIN

Commande ON Client.id_client = Commande.id_client;

nom	Montant (€)
Sophie	50
Sophie	30
Issa	70
Issa	20
Benoît	NULL
Fatou	NULL
NULL	100